EXPERIMENT 1

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

Theory:

For machine learning algorithms to perform effectively, it is essential to preprocess raw data and convert it into a clean, usable dataset. This often involves encoding categorical data into numerical representations since most algorithms require numeric input. One common approach is to transform categorical labels into column vectors with binary values, a process known as one-hot encoding. This ensures that the categorical data is represented in a way that the algorithm can interpret without assigning implicit ordinal relationships between categories.

Another critical preprocessing step is handling missing values, which are often represented as NaNs (Not a Number). Missing values can arise from various issues, such as incomplete data collection, sensor malfunctions, or user input errors. These missing entries can lead to errors or biases in the training process if not addressed properly.

Problem Statement

The given dataset provides comprehensive details about retail product sales, focusing on the relationship between product attributes, outlet characteristics, and sales performance. This analysis aims to address the following key objectives:

- **Product Performance**: Identifying products or product types that drive the highest sales and those underperforming.
- **Outlet Insights**: Understanding the impact of outlet size, location, and type on overall sales performance.
- **Pricing Analysis**: Investigating how pricing strategies, such as maximum retail price (MRP), influence customer purchasing behavior.
- **Possible sales Prediction**: Developing models to predict item-level sales and provide actionable insights for inventory and pricing strategies.

By preprocessing the dataset and applying statistical analysis, the goal is to extract meaningful patterns that can guide data-driven decisions in retail operations.

Dataset Overview

Veydant Sharma D15C 50 EXP1

The dataset comprises **12 columns**, each detailing specific aspects of retail products, their pricing, visibility, and sales performance across outlets. Below is a breakdown of the dataset's columns and their relevance:

1. **Item_Identifier**: A unique code for each product, essential for distinguishing between items in the inventory.
2. **Item_Weight**: Represents the weight of each product, which may impact logistics and consumer preference.
3. **Item_Fat_Content**: Categorizes items as Low Fat or Regular Fat, reflecting their nutritional value and target audience.
4. **Item_Visibility**: A measure of the shelf visibility of an item, influencing its likelihood of being purchased.
5. **Item_Type**: Broad categories like Dairy, Beverages, or Snacks, helping analyze trends across product types.
6. **Item_MRP**: The maximum retail price, a key factor in determining product affordability and customer demand.
7. **Outlet_Identifier**: A unique code assigned to each retail outlet, linking products to specific store locations.
8. **Outlet_Establishment_Year**: Indicates the year the outlet began operations, useful for analyzing store maturity's effect on sales.
9. **Outlet_Size**: Categorizes stores as Small, Medium, or Large, impacting foot traffic and product demand.
10. **Outlet_Location_Type**: Describes whether the store is Urban, Suburban, or in a Tier 3 area, capturing demographic influences.
11. **Outlet_Type**: Differentiates between store types, such as Grocery Stores or Supermarkets, reflecting varying business models.
12. **Item_Outlet_Sales**: The target variable, representing the total sales of a product at a specific outlet.

Steps:

1. **Loading the Data in Pandas**
   The initial step is to load the dataset into a Pandas DataFrame. This can be done using the `read_csv` method to load the data from a CSV file, which forms the foundation for any data analysis or preprocessing tasks.

Importing of the dataset

```
[2] import pandas as pd
    df = pd.read_csv("/content/market_data.csv")
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Out
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	

2. Getting a Quick Overview of the Dataset
   To gain a better understanding of the dataset's structure, we can use the `df.info()` method, which gives details about the columns (or features) in the dataset, along with their data types.

Description of Dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            8523 non-null   object
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object
 7   Outlet_Establishment_Year  8523 non-null   int64
 8   Outlet_Size                6113 non-null   object
 9   Outlet_Location_Type       8523 non-null   object
 10  Outlet_Type                8523 non-null   object
 11  Item_Outlet_Sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

3. **Removing Irrelevant Columns**
   In most datasets, not every column contributes to the analysis. Some columns may be redundant or unnecessary. These columns only add to the size of the dataset without providing meaningful insights. For instance, in this case, the column "Outlet_Establishment_Year" is irrelevant for our analysis and can be removed to streamline the data.

Drop columns that aren't useful

```
[5]  cols = ["Outlet_Establishment_Year"]
     df = df.drop(cols, axis=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Item_Identifier       8523 non-null   object
 1   Item_Weight           7060 non-null   float64
 2   Item_Fat_Content      8523 non-null   object
 3   Item_Visibility       8523 non-null   float64
 4   Item_Type             8523 non-null   object
 5   Item_MRP              8523 non-null   float64
 6   Outlet_Identifier     8523 non-null   object
 7   Outlet_Size           6113 non-null   object
 8   Outlet_Location_Type  8523 non-null   object
 9   Outlet_Type           8523 non-null   object
 10  Item_Outlet_Sales     8523 non-null   float64
dtypes: float64(4), object(7)
memory usage: 732.6+ KB
```

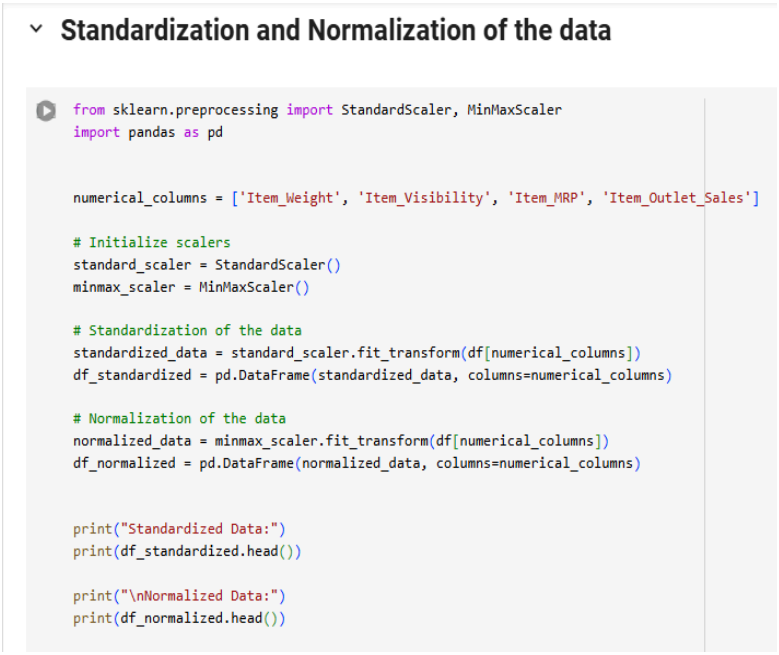
4. **Eliminating Rows with Excessive Missing Values**
   Datasets often contain rows with missing values that can distort analysis results. In this step, we remove rows that have too many missing values, as they may not be useful for the analysis. The `dropna()` method helps us achieve this, ensuring that only rows with complete data are retained.
5. **Handling Missing Values**
   Upon inspecting the dataset, we find that columns have a significant number of missing entries. This can introduce bias or errors in our analysis. To tackle this, we categorize the items based on their type, then calculate the mean weight for each category. These means are then used to fill the missing values, ensuring the dataset remains consistent and accurate.
6. **Detecting Outliers (Manually)**
   Outliers can significantly affect the performance of models, so identifying and handling them is crucial. For the "Item_Outlet_Sale" column, we use the Interquartile Range (IQR) method to identify outliers. The first quartile (Q1) is 834.2474, the second quartile (Q2) is 3101.2964, and the IQR is 2267.049. Using this, we calculate the lower and upper bounds to flag outliers as values that fall outside this range. Any data points lower than the lower bound or higher than the upper bound can be considered outliers.

Item_Iden	Item_Weig	Item_Fat_	Item_Visib	Item_Type	Item_MRP	Outlet_Ide	Outlet_Est	Outlet_Siz	Outlet_Lo	Outlet_Ty	Item_Outlet_Sales
FDA46	13.6	Low Fat	0.117818	Snack Foo	192.9136	OUT049	1999	Medium	Tier 1	Supermark	2527.377
FDC02	21.35	Low Fat	0.069103	Canned	259.9278	OUT018	2009	Medium	Tier 3	Supermark	6768.523
NCP30	20.5	Low Fat	0.032835	Household	40.2822	OUT045	2002		Tier 2	Supermark	707.0796
FDY25		Low Fat	0.03381	Canned	180.5976	OUT027	1985	Medium	Tier 3	Supermark	7968.294
NCH54	13.5	Low Fat	0.072669	Household	160.292	OUT046	1997	Small	Tier 1	Supermark	1438.128
NCR53		Low Fat	0.144338	Health and	224.4404	OUT027	1985	Medium	Tier 3	Supermark	6976.252
FDY56	16.35	Regular	0.062764	Fruits and	227.6062	OUT017	2007		Tier 2	Supermark	7222.598
FDH19		Low Fat	0.032928	Meat	173.1738	OUT027	1985	Medium	Tier 3	Supermark	7298.5
FDY55	16.75	Low Fat	0.081253	Fruits and	256.4988	OUT013	1987	High	Tier 3	Supermark	7452.965
FDU23	17.85	Low Fat	0.147024	Breads	93.7436	OUT018	2009	Medium	Tier 3	Supermark	1134.523
DRE60	9.395	Low Fat	0.159658	Soft Drinks	224.972	OUT045	2002		Tier 2	Supermark	7696.648
DRP47	15.75	Low Fat	0.141399	Hard Drink	250.5382	OUT017	2007		Tier 2	Supermark	2775.72
FDU55	16.2	Low Fat	0.035984	Fruits and	260.6278	OUT045	2002		Tier 2	Supermark	4425.573
FDN58		Regular	0.056597	Snack Foo	230.9984	OUT027	1985	Medium	Tier 3	Supermark	9267.936
FDI44	16.1	Low Fat	0.100389	Fruits and	76.0328	OUT049	1999	Medium	Tier 1	Supermark	1853.587
FDW56		Low Fat	0.070557	Fruits and	191.2162	OUT027	1985	Medium	Tier 3	Supermark	7504.232
FDE45	12.1	Low Fat	0.040522	Fruits and	178.5002	OUT018	2009	Medium	Tier 3	Supermark	5552.106
FDR35		Low Fat	0.020597	Breads	200.0742	OUT027	1985	Medium	Tier 3	Supermark	8958.339

7. Data Scaling: Standardization and Normalization

Standardization and normalization are key techniques in preprocessing that ensure the features in

the dataset are on a similar scale. Standardization (also known as Z-score scaling) transforms the data so that it has a mean of 0 and a standard deviation of 1. Normalization, on the other hand, rescales the data to a specific range, typically between 0 and 1. Both methods are essential for improving the performance and accuracy of machine learning models, as they help avoid any one feature dominating the others due to differences in scale.



Standardization and Normalization of the data

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import pandas as pd


numerical_columns = ['Item_Weight', 'Item_Visibility', 'Item_MRP', 'Item_Outlet_Sales']

# Initialize scalers
standard_scaler = StandardScaler()
minmax_scaler = MinMaxScaler()

# Standardization of the data
standardized_data = standard_scaler.fit_transform(df[numerical_columns])
df_standardized = pd.DataFrame(standardized_data, columns=numerical_columns)

# Normalization of the data
normalized_data = minmax_scaler.fit_transform(df[numerical_columns])
df_normalized = pd.DataFrame(normalized_data, columns=numerical_columns)


print("Standardized Data:")
print(df_standardized.head())

print("\nNormalized Data:")
print(df_normalized.head())
```

Conclusion:

- This experiment focused on the essential steps of data preprocessing using Pandas for data science applications.
- We worked with a retail dataset, performing initial data exploration and analysis.
- Identified and addressed missing values to maintain data integrity.
- Used the **Interquartile Range (IQR) method** to detect and eliminate outliers, ensuring dataset consistency.
- Explored **feature selection techniques** to retain the most relevant variables for analysis.
- Applied **data transformation methods** to enhance data structure and usability.
- Highlighted the significance of **standardization and normalization** in preparing data for machine learning models.

Experiment 2

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

Theory:

Data visualization and exploratory data analysis (EDA) play a crucial role in understanding datasets by identifying patterns, trends, and relationships. Matplotlib is a versatile, low-level plotting library that allows for the creation of static, dynamic, and interactive charts, while Seaborn, which builds upon Matplotlib, offers a higher-level interface that simplifies the creation of aesthetically pleasing statistical graphics. EDA involves various techniques such as histograms, scatter plots, box plots, and heatmaps. These tools help in exploring data distributions, identifying potential outliers, and analyzing correlations among different variables.

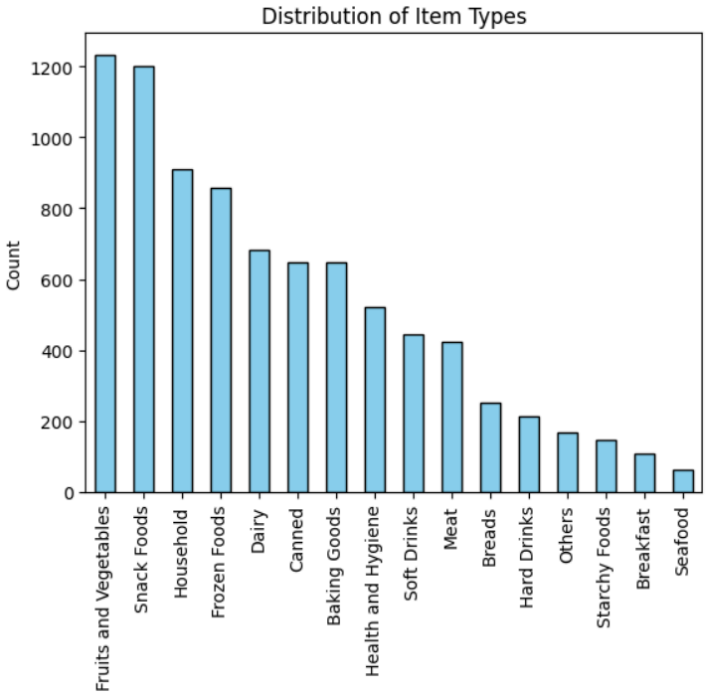
Both Matplotlib and Seaborn enable users to gain insights into the underlying structure of data by offering diverse visualization options. These visualizations make it easier to detect patterns and trends that might not be immediately obvious in raw data. By leveraging EDA, analysts can make informed decisions on how to preprocess and transform the data for further analysis. Additionally, these tools help in identifying any data quality issues, such as missing values or inconsistencies, which can be addressed early in the data analysis pipeline.

1. **Bar graph and contingency table using any two features:**

    a. Bar Graph for distribution of item type.

```
import matplotlib.pyplot as plt

df['Item_Type'].value_counts().plot(kind='bar', color='skyblue', edgecolor='black')

plt.xlabel("Item Type")
plt.ylabel("Count")
plt.title("Distribution of Item Types")
plt.xticks(rotation=90)
plt.show()
```

Veydant Sharma D15C 50 EXP2



From the graph, we observe that the majority of items fall under the category of "Fruits and Vegetables," with a count of approximately 1,200. Similarly, the "Snack Foods" category also comprises a comparable number of items, indicating a balanced distribution between these two categories. Rest of the items have a count of less than 1000. “Seafood” accounts for less than approximately 100 items, which is the lowest.

b. Contingency Table for Item type and Outlet type
A contingency table (or cross-tabulation table) displays the frequency distribution of two categorical variables in a dataset. It helps in understanding the relationship between the two features

```
contingency_table = pd.crosstab(df['Item_Type'] , df['Item_Visibility'] )
from IPython.display import display
display(contingency_table)
```

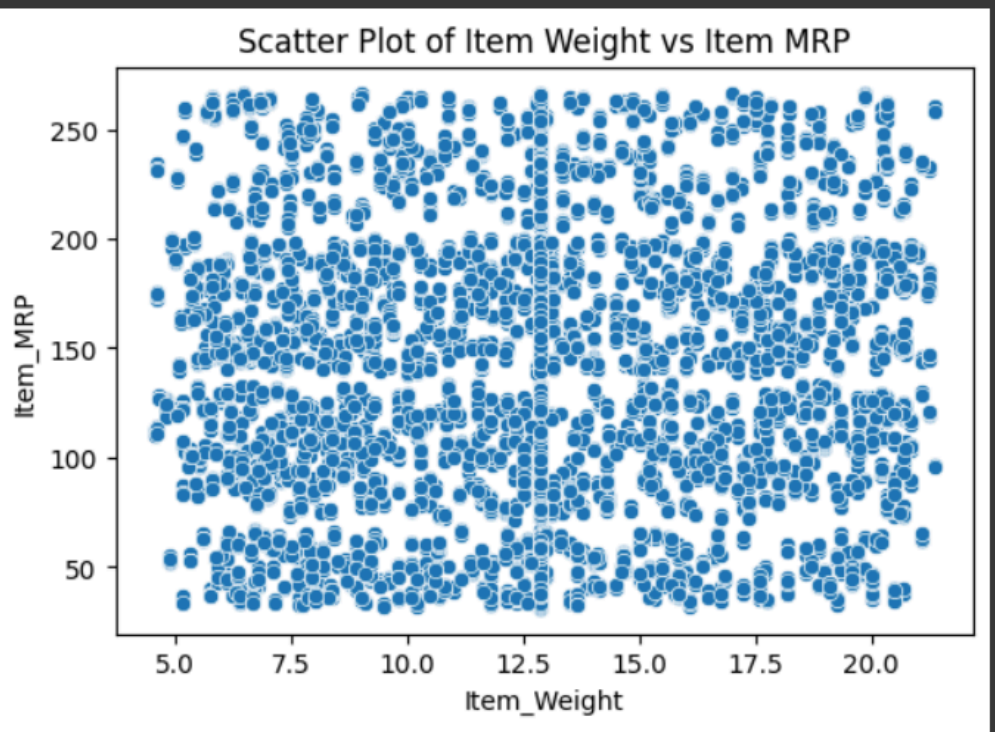
Item_Visibility	0.000000	0.003575	0.003589	0.003598	0.003599	0.003607	0.003612	0.005210	0.005231	0.005234	...	0.304737	0.304859	0.305305	0.306543	0.308145	0.309390	0.311090	0.321115
Item_Type																			
Baking Goods	45	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	1	0
Breads	17	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
Breakfast	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
Canned	33	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
Dairy	38	0	0	0	0	0	0	1	1	1	...	1	0	0	0	0	0	0	0
Frozen Foods	51	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
Fruits and Vegetables	87	0	0	0	0	0	0	0	0	0	...	0	0	1	0	1	0	0	1
Hard Drinks	17	1	1	1	1	1	1	0	0	0	...	0	0	0	0	0	0	0	0
Health and Hygiene	33	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
Household	51	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0
Meat	20	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
Others	10	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
Seafood	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
Snack Foods	68	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
Soft Drinks	34	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0
Starchy Foods	15	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

The dataset suggests that most items have very low visibility, with only a few instances where certain products, like **Dairy, Hard Drinks, and Fruits and Vegetables**, appear more frequently. This indicates that some categories might have better shelf placement or higher consumer exposure. The presence of **many zero values** implies that most items remain unseen or are placed in less prominent areas. Categories with **higher visibility occurrences** may be strategically placed to attract customers, while others might require improved positioning or marketing efforts. Analyzing this data can help optimize product placement, enhance sales strategies, and improve inventory management.

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

a. Scatter Plot for Item Weight vs Item MRP

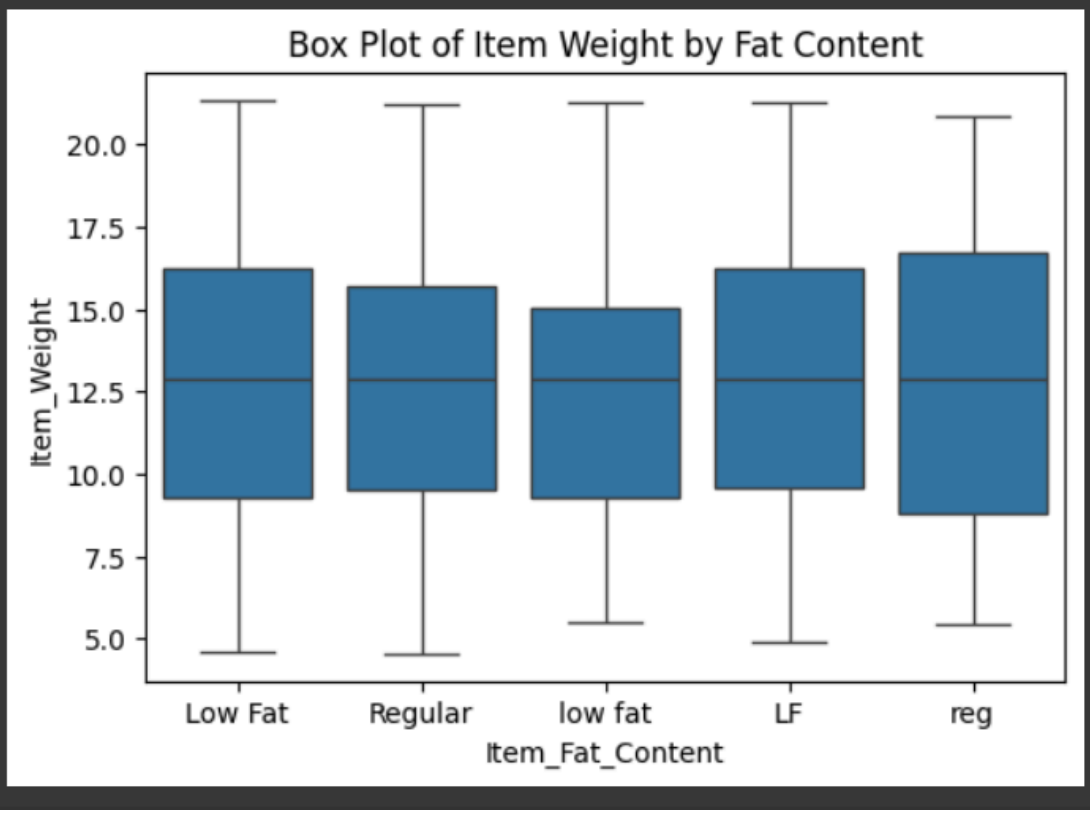
```
plt.figure(figsize=(6, 4))
sns.scatterplot(data=df, x='Item_Weight', y='Item_MRP')
plt.title('Scatter Plot of Item Weight vs Item MRP')
plt.show()
```



From the visualization, shows the distribution between item weights and MRP. With the distribution it can be understood about products that are premium and others that are more mass user centric. The items with lower weights but higher values are more premium and vice versa are made for the general public

b. Box Plot

```
plt.figure(figsize=(6, 4))
sns.boxplot(data=df, x='Item_Fat_Content', y='Item_Weight')
plt.title('Box Plot of Item Weight by Fat Content')
plt.show()
```

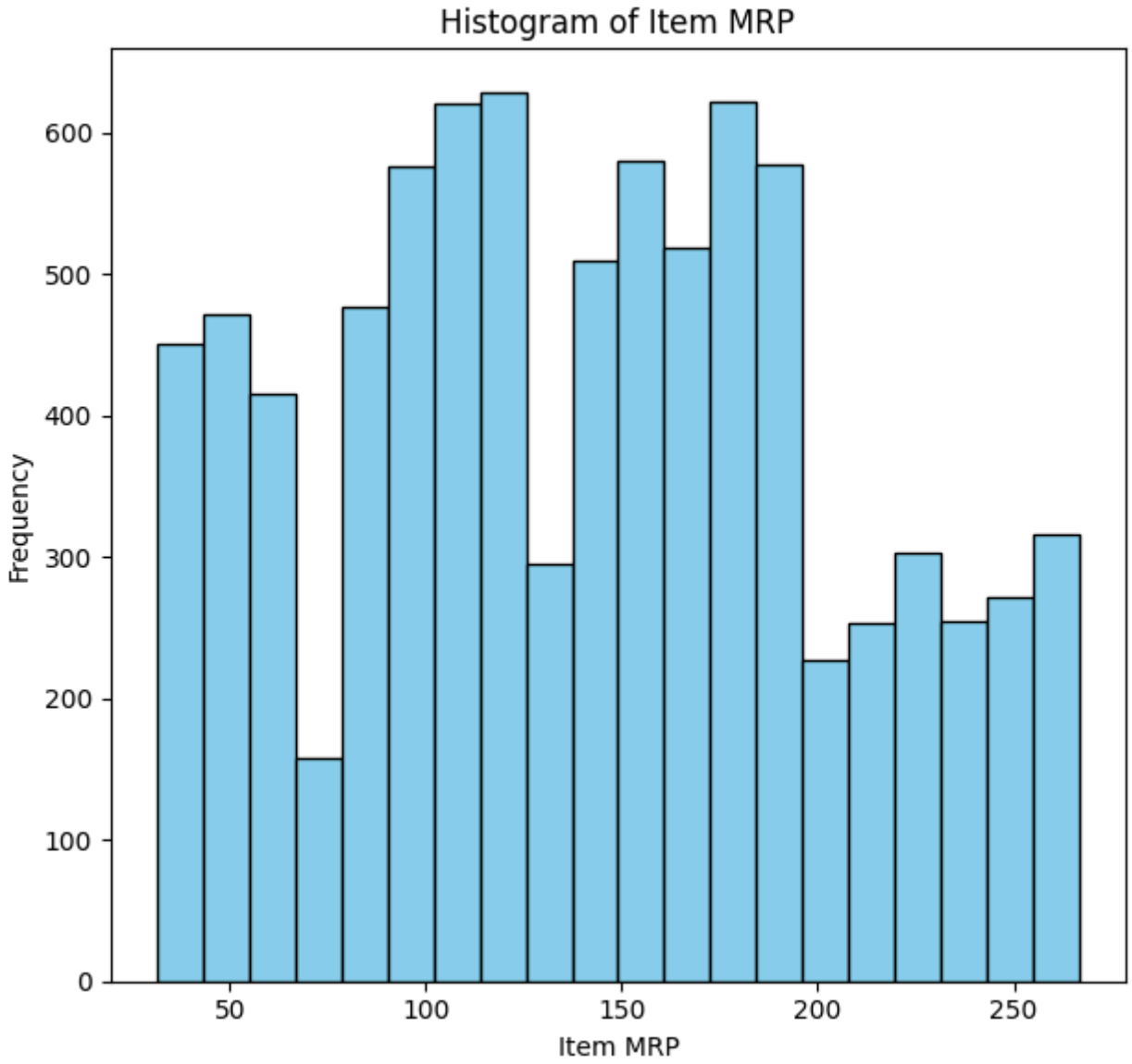


The box plot highlights the distribution of Item Weight across various Item Fat Contents, with some variations in medians

3. Histogram and normalized Histogram.

a. Histogram for Item Outlet Sales

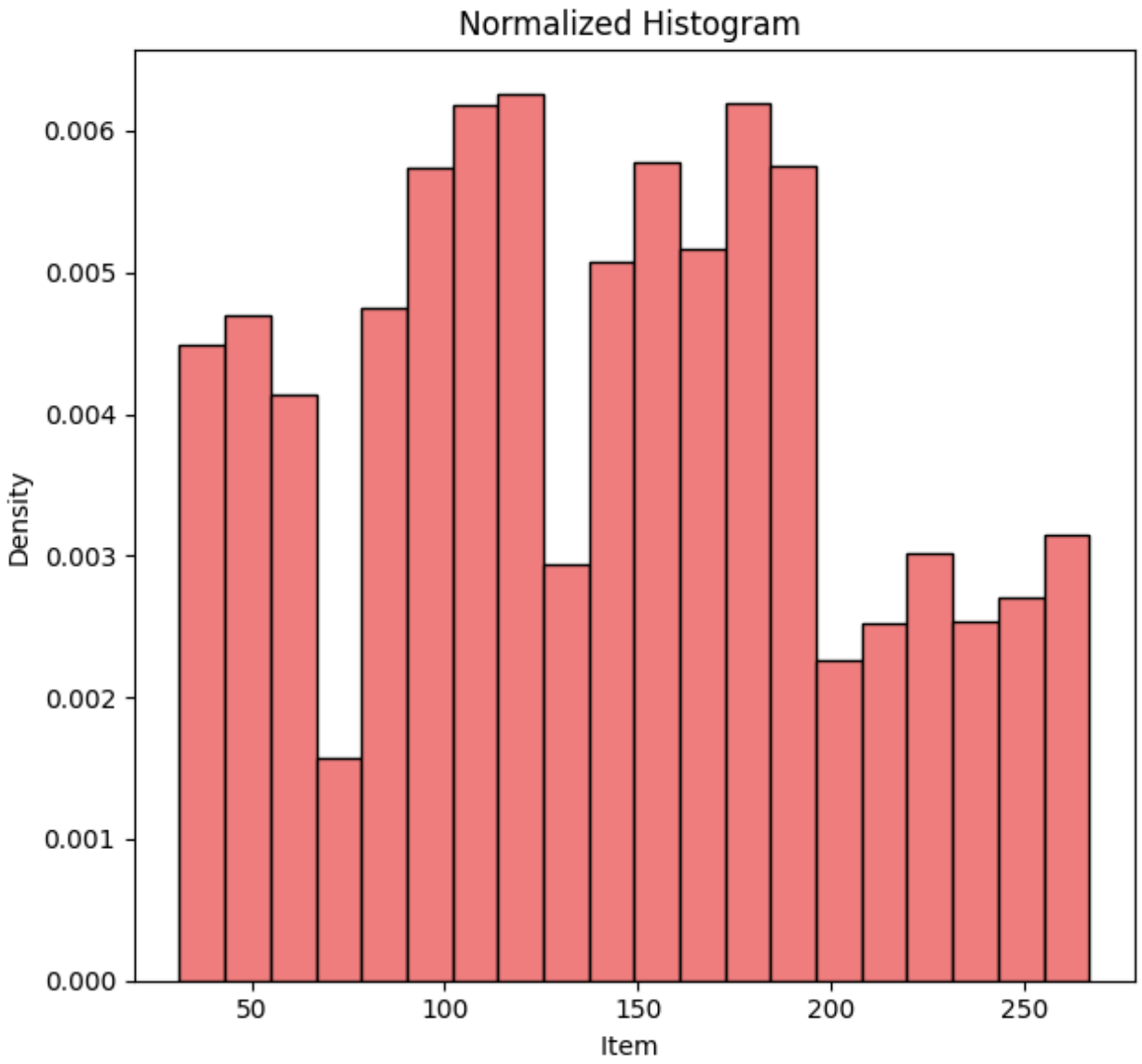
```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.hist(df['Item_MRP'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of Item MRP')
plt.xlabel('Item MRP')
plt.ylabel('Frequency')


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



The histogram reveals the distribution of item_MRP, distribution is done across the frequency of the MRP of products

b. Normalized Histogram

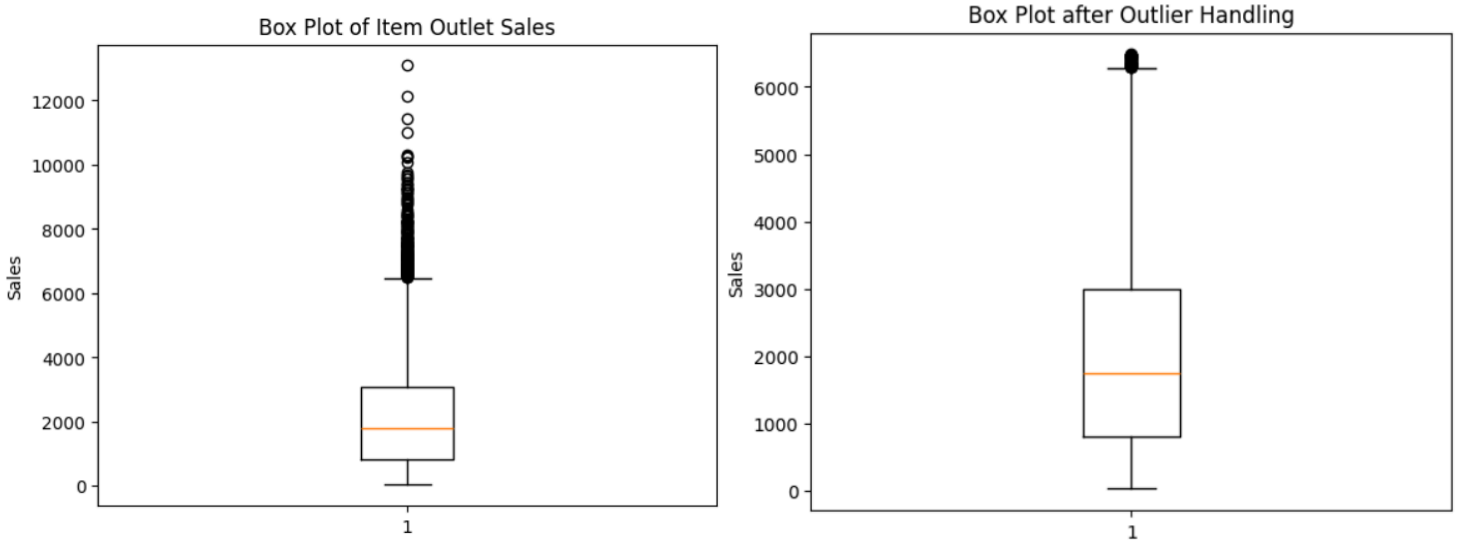
```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 2)
plt.hist(df['Item_MRP'], bins=20, color='lightcoral', edgecolor='black', density=True)
plt.title('Normalized Histogram ')
plt.xlabel('Item ')
plt.ylabel('Density')


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



4. **Outliers using box plot and Inter quartile range.**

Veydant Sharma D15C 50 EXP2

```
sales = df['Item_Outlet_Sales']

plt.boxplot(sales)
plt.title('Box Plot of Item Outlet Sales')
plt.ylabel('Sales')
plt.show()

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

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

filtered_sales = sales[(sales >= lower_bound) & (sales <= upper_bound)]

df['Item_Outlet_Sales_Capped'] = np.where(
    sales < lower_bound, lower_bound,
    np.where(sales > upper_bound, upper_bound, sales)
)

plt.boxplot(filtered_sales)
plt.title('Box Plot after Outlier Handling')
plt.ylabel('Sales')
plt.show()
```



The first box plot highlights that the dataset has several outliers, with Item Outlet Sales extending well beyond the upper whisker (above ~6000). These are visible as individual points above the main plot area.After applying the interquartile range (IQR) method for outlier handling, the second box plot demonstrates that extreme outliers have been removed. The data now falls within a more compact range, capped approximately at the upper whisker (~6000)

Conclusion:

Veydant Sharma D15C 50 EXP2

- The experiment demonstrated the effectiveness of Matplotlib and Seaborn in visualizing and analyzing data through EDA.
- Key patterns, distributions, and relationships within the dataset were uncovered using bar charts, contingency tables, scatter plots, box plot.
- Normalized histograms were used to reveal data density, providing insights into the distribution of values.
- The interquartile range (IQR) method was applied to detect and remove outliers, improving the data quality and reliability.

Veydant Sharma D15C 50 EXP3

Experiment 3

Aim: Perform Data Modeling:

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

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

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

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

Theory:

Data modeling is a critical process in the field of data science and machine learning, as it helps in organizing and structuring data to make it usable for analysis. One key aspect of data modeling is **partitioning the dataset** into subsets for various stages of model training and evaluation. This ensures that the model is trained on a representative portion of the data while evaluating its performance on unseen data, which helps prevent overfitting and provides an estimate of how the model will perform on real-world data.

A **Two-Sample Z-test** is used to determine if there is a significant difference between the means of two independent samples. This test is commonly used when the population variances are known or when the sample size is large enough (typically greater than 30) for the Central Limit Theorem to apply, ensuring the sampling distribution of the sample mean is approximately normal.

The formula for the Z-score in a two-sample Z-test is:

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

H0: The distribution is valid and statistically significant.
HA: The distribution is not valid nor statistically significant.
Value of |Z| = 1.247
i.e |Z|<1.96
Hence H0 is accepted

Steps:

1. Partition the Dataset

The dataset is partitioned between training(75%) and testing(25%) data. This step is crucial in data modeling to validate the model's performance on unseen data while preventing overfitting.

```
[ ] from sklearn.model_selection import train_test_split
    train_df , test_df = train_test_split(df , test_size=0.25 , random_state=42)
    print(train_df.shape, test_df.shape)

    (6392, 11) (2131, 11)

[ ] X = list(range(3000))
    Y = list(range(3000))
    X_train , X_test , Y_train , Y_test = train_test_split(X , Y , test_size=0.25 , random_state=42)
```

Veydant Sharma D15C 50 EXP3

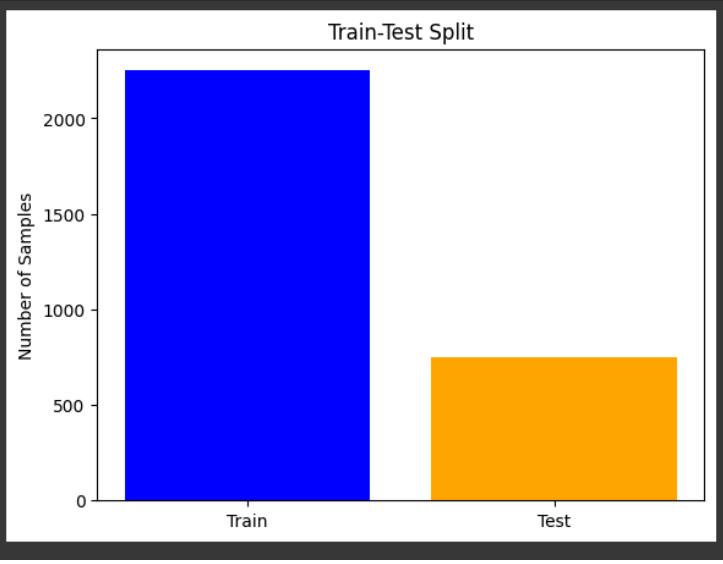
2. Validate Proportions using Graphs

The proportions are validated using Bar and Pie graphs

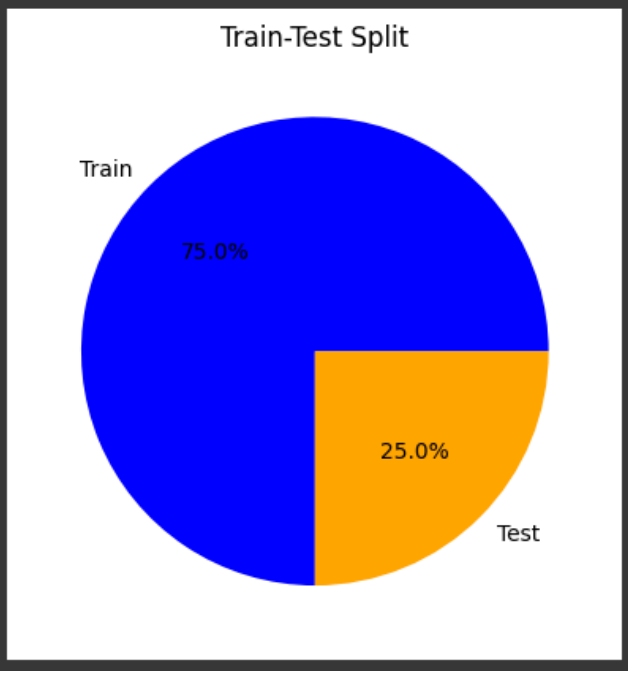
Bar Graph: Presents the distribution of data into training and testing by numerical value, i.e, the number of records on Y axis.

Pie Chart: Presents the distribution of data based on the percentage of distribution of the dataset

```
labels = ['Train' , 'Test']
sizes = [len(X_train) , len(X_test)]
plt.bar(labels, sizes, color=['blue', 'orange'])
plt.ylabel('Number of Samples')
plt.title('Train-Test Split')
plt.show()
```




```
plt.pie(sizes , labels=labels , colors=['blue' , 'orange'] , autopct='%1.1f%%')
plt.title('Train-Test Split')
plt.show()
```



3. Identify the total number of records in the training data set.
   Displays the total number of rows in training data and testing data.

```
print("Total records in training dataset:", len(X_train))
print("Total records in training dataset:", len(X_test))
```

```
Total records in training dataset: 2250
Total records in training dataset: 750
```

4. Validate partition by performing a two-sample Z-test.
   On performing the Z test mathematically on the entire dataset, we get the value of |Z| as 1.247 , i.e less than 1.96 (value of Z for level of significance of 5%). Hence the null hypothesis of dataset partition being statistically significant and valid is accepted.

```
import numpy as np
from scipy.stats import norm
mean_train = np.mean(X_train)
mean_test = np.mean(X_test)

std_train = np.std(X_train, ddof=1)
std_test = np.std(X_test, ddof=1)

n_train = len(X_train)
n_test = len(X_test)


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


z_critical = norm.ppf(0.975)

print(f"Z-score: {z_score:.3f}")
if abs(z_score) < z_critical:
    print(" The train-test split is statistically valid (distributions are similar).")
else:
    print(" The train-test split may be biased (distributions may differ significantly).")
```

```
Z-score: 1.247
 The train-test split is statistically valid (distributions are similar).
```

Conclusion:

- We performed splitting of dataset (75% training and 25% testing)
- Validated the proportions using bar and pie graphs, successfully showing the split.
- Rechecked the number of instances in both datasets.
- Validated the partition using two sample Z test, in which the Null Hypothesis was accepted, the level of significance chosen was 5%.
- Hence, we learnt how to perform data modelling, split the dataset, and validate the split and proportion.

Veydant Sharma D15C 50 EXP4

Aim: Implementation of Statistical Hypothesis Test using Scipy and Sci-kit learn.
Perform the following Tests:Correlation Tests:
a) Pearson's Correlation Coefficient
b) Spearman's Rank Correlation
c) Kendall's Rank Correlation
d) Chi-Squared Test

Theory:

a) Pearson's Correlation Coefficient

Pearson's correlation coefficient measures the strength and direction of the **linear** relationship between two continuous variables.

Formula:

$$r = \Sigma[(x_i - \bar{x})(y_i - \bar{y})] / \sqrt{[\Sigma(x_i - \bar{x})^2 \times \Sigma(y_i - \bar{y})^2]}$$

Where:

- x_i, y_i = individual sample values
- $\bar{x}$, $\bar{y}$ = means of x and y
- Σ = summation

b) Spearman's Rank Correlation

Spearman's rank correlation assesses the **monotonic** relationship between two ranked variables. It is a **non-parametric** alternative to Pearson's correlation.

Formula:

$$\rho = 1 - (6 \times \Sigma d_i^2) / [n(n^2 - 1)]$$

Where:

- d_i = difference between the ranks of each observation
- n = number of observations

- Σ = summation

c) Kendall's Rank Correlation

Kendall's tau is another **non-parametric** correlation coefficient based on the number of **concordant** and **discordant** pairs.

Formula:
$$\tau = (C - D) / [n(n - 1) / 2]$$

Where:

- C = number of concordant pairs
- D = number of discordant pairs
- n = total number of observations

d) Chi-Squared Test

The Chi-Squared test is used to test relationships between **categorical variables**. It compares **observed** and **expected** frequencies.

Formula:
$$\chi^2 = \Sigma[(O_i - E_i)^2 / E_i]$$

Where:

- O_i = observed frequency
- E_i = expected frequency
- Σ = summation

Veydant Sharma D15C 50 EXP4

Steps:

1. Pearson's correlation coefficient

```python
import numpy as np
import pandas as pd

x = df["Item_MRP"].values
y = df["Item_Outlet_Sales"].values

def pearson_corr_manual(x, y):
    n = len(x)
    mean_x = np.mean(x)
    mean_y = np.mean(y)

    numerator = np.sum((x - mean_x) * (y - mean_y))
    denominator = np.sqrt(np.sum((x - mean_x)**2) * np.sum((y - mean_y)**2))

    return numerator / denominator

from scipy.stats import pearsonr

manual_corr = pearson_corr_manual(x, y)

lib_corr, _ = pearsonr(x, y)


print("Manual Pearson Correlation Coefficient:", manual_corr)
print("Library Pearson Correlation Coefficient:", lib_corr)
```

```
Manual Pearson Correlation Coefficient: 0.5675744466569193
Library Pearson Correlation Coefficient: 0.5675744466569194
```

- The Pearson correlation coefficient is approximately 0.567, indicating a moderate positive linear relationship between `Item_MRP` and `Item_Outlet_Sales`.
- This suggests that, in general, as the maximum retail price of an item increases, its sales tend to increase as well.
- The correlation is not close to 1, so while the relationship is positive, other factors also significantly influence sales.

- This insight could guide pricing strategy — higher-priced items might sell more, but it's important to consider other variables like outlet type, visibility, or discounts.

2. Spearman's Rank Correlation:

```python
import numpy as np
import pandas as pd
from scipy.stats import spearmanr

x = df["Item_MRP"].values
y = df["Item_Outlet_Sales"].values

def pearson_corr_manual(x, y):
    mean_x = np.mean(x)
    mean_y = np.mean(y)
    numerator = np.sum((x - mean_x) * (y - mean_y))
    denominator = np.sqrt(np.sum((x - mean_x)**2) * np.sum((y - mean_y)**2))
    return numerator / denominator

def spearman_corr_manual(x, y):
    rank_x = pd.Series(x).rank().values
    rank_y = pd.Series(y).rank().values
    return pearson_corr_manual(rank_x, rank_y)

print("Spearman Manual:", spearman_corr_manual(x, y))
print("Spearman Library:", spearmanr(x, y)[0])
```

```
Spearman Manual: 0.5629864415335609
Spearman Library: 0.5629864415335609
```

- The Spearman correlation coefficient is approximately 0.563, indicating a moderate positive monotonic relationship between Item_MRP and Item_Outlet_Sales.
- This suggests that as the Item_MRP increases, the rank of Item_Outlet_Sales tends to increase as well, even if the relationship is not perfectly linear.
- Since Spearman correlation is based on ranks, it captures consistent trends in the data even when values are not linearly related.
- The result being close to the Pearson correlation implies that the relationship is both monotonic and reasonably linear.

3. Kendall's Rank Correlation:

```python
import numpy as np
import pandas as pd
from scipy.stats import kendalltau

x = df["Item_MRP"].values
y = df["Item_Outlet_Sales"].values

def kendall_corr_manual(x, y):
    n = len(x)
    concordant = discordant = 0
    for i in range(n):
        for j in range(i + 1, n):
            concordant += ((x[i] - x[j]) * (y[i] - y[j]) > 0)
            discordant += ((x[i] - x[j]) * (y[i] - y[j]) < 0)
    return (concordant - discordant) / (0.5 * n * (n - 1))

print("Kendall Manual:", kendall_corr_manual(x, y))
print("Kendall Library:", kendalltau(x, y)[0])
```

```
Kendall Manual: 0.40690027341013535
Kendall Library: 0.40699111339889754
```

- The Kendall correlation coefficient is approximately 0.407, which indicates a moderate positive association between Item_MRP and Item_Outlet_Sales based on the ranking of values.
- This implies that higher values of Item_MRP are generally associated with higher ranks of Item_Outlet_Sales, but the relationship is not very strong.
- Kendall correlation is more conservative than Spearman and less sensitive to large differences in ranks, making it suitable for data with small sample sizes or many tied ranks.
- The value is slightly lower than the Spearman coefficient, suggesting the relationship is not perfectly consistent across all rank pairs.

4. Chi Square:

```python
import numpy as np
import pandas as pd
from scipy.stats import chi2, chi2_contingency

# Create contingency table
contingency_table = pd.crosstab(df["Outlet_Size"], df["Sales_Bin"])
observed = contingency_table.values

row_totals = observed.sum(axis=1).reshape(-1, 1)
col_totals = observed.sum(axis=0)
grand_total = observed.sum()
expected = (row_totals @ col_totals.reshape(1, -1)) / grand_total

chi_squared_manual = np.sum((observed - expected) ** 2 / expected)

# Degrees of Freedom
df_degrees = (observed.shape[0] - 1) * (observed.shape[1] - 1)

# Compute p-value manually
p_value_manual = 1 - chi2.cdf(chi_squared_manual, df_degrees)

# Compute using SciPy
chi2_scipy, p_value_scipy, df_scipy, expected_scipy = chi2_contingency(contingency_table)

# Print results
print(f"Chi-Square Statistic (Manual): {chi_squared_manual:.4f}")
print(f"Chi-Square Statistic (SciPy): {chi2_scipy:.4f}")
print(f"P-value (Manual): {p_value_manual:.4f}")
print(f"P-value (SciPy): {p_value_scipy:.4f}")
print(f"Degrees of Freedom: {df_degrees}")
print("\nExpected Frequencies:")
print(expected)
```

```python
# Hypothesis Decision
alpha = 0.05  # Significance level
if p_value_scipy < alpha:
    print("\nConclusion: Null hypothesis (H₀) is REJECTED.")
    print("There is a significant relationship between Outlet Size and Sales.")
else:
    print("\nConclusion: Null hypothesis (H₀) is NOT rejected.")
    print("There is no significant relationship between Outlet Size and Sales.")
```

```
Chi-Square Statistic (Manual): 356.9009
Chi-Square Statistic (SciPy): 356.9009
P-value (Manual): 0.0000
P-value (SciPy): 0.0000
Degrees of Freedom: 4

Expected Frequencies:
[[ 311.10407134  310.33861316  310.5573155 ]
 [ 932.31080605  930.01689546  930.67229849]
 [1601.58512261 1597.64449138 1598.77038601]]

Conclusion: Null hypothesis (H₀) is REJECTED.
There is a significant relationship between Outlet Size and Sales.
```

eject the null hypothesis and conclude that Outlet Size and Sales are related.

- The chi-square statistic (356.9009) indicates a strong deviation from expected values, suggesting a considerable association between Outlet Size and Sales.
- Given the degrees of freedom, the result is statistically valid for the given contingency table structure.
- While the test confirms a relationship, it does not indicate how strong or in which direction the relationship is.

Conclusion:

In this experiment, we implemented and analyzed various statistical hypothesis tests using Scipy and Scikit-learn to examine the relationships between different variables in a retail dataset. Pearson's correlation coefficient was approximately 0.567, revealing a moderate positive linear relationship between item price and sales, indicating that higher-priced items tend to sell more, though other factors also contribute. Spearman's rank correlation was approximately 0.563, confirming a similar monotonic relationship using ranked data. Kendall's rank correlation, at approximately 0.407, further supported a moderate positive association, though with a more conservative estimate. Lastly, the Chi-Squared test showed a significant relationship (chi-square statistic = 356.9009, p-value < 0.05) between categorical variables, confirming dependency in the data. These tests collectively provide a comprehensive understanding of both linear and non-linear dependencies, supporting better decision-making in pricing, inventory, and outlet strategies.

Veydant Sharma **D15C** **50** **EXP4**

Veydant Sharma D15C 50 EXP 5

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

Problem Statement:

a) Perform Logistic regression to find out relation between variables

b) Apply regression model technique to predict the data on above dataset.

Theory:

Regression analysis is a fundamental statistical technique used to model and analyze the relationship between a dependent variable (target) and one or more independent variables (features). It is commonly used for prediction, forecasting, and understanding the strength of relationships between variables.

Types of Regression:

1. **Simple Linear Regression**: Involves a single independent variable and a linear relationship with the dependent variable.
    - Equation: $y = mx + c$
    - Where:
        - y = dependent variable
        - x = independent variable
        - m = slope of the line
        - c = y-intercept
2. **Multiple Linear Regression**: Involves two or more independent variables influencing the dependent variable.
    - Equation: $y = b_0 + b_1x_1 + b_2x_2 + ... + b_nx_n$
    - Where:
        - y = dependent variable
        - $x_1, x_2, ..., x_n$ = independent variables

- b_0 = intercept
- b_1, ..., b□ = coefficients

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

Steps:

1. Load Dataset and check.

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103904 entries, 0 to 103903
Data columns (total 23 columns):
 #   Column                            Non-Null Count   Dtype  
---  ------                            --------------   -----  
 0   Gender                            103904 non-null  object 
 1   Customer Type                     103904 non-null  object 
 2   Age                               103904 non-null  int64  
 3   Type of Travel                    103904 non-null  object 
 4   Class                             103904 non-null  object 
 5   Flight Distance                   103904 non-null  int64  
 6   Inflight wifi service             103904 non-null  int64  
 7   Departure/Arrival time convenient  103904 non-null  int64  
 8   Ease of Online booking            103904 non-null  int64  
 9   Gate location                     103904 non-null  int64  
 10  Food and drink                    103904 non-null  int64  
 11  Online boarding                   103904 non-null  int64  
 12  Seat comfort                      103904 non-null  int64  
 13  Inflight entertainment            103904 non-null  int64  
 14  On-board service                  103904 non-null  int64  
 15  Leg room service                  103904 non-null  int64  
 16  Baggage handling                  103904 non-null  int64  
 17  Checkin service                   103904 non-null  int64  
 18  Inflight service                  103904 non-null  int64  
 19  Cleanliness                       103904 non-null  int64  
 20  Departure Delay in Minutes        103904 non-null  int64  
 21  Arrival Delay in Minutes          103904 non-null  float64
 22  satisfaction                      103904 non-null  int64  
dtypes: float64(1), int64(18), object(4)
memory usage: 18.2+ MB
```

The dataset contains a total of 103,904 records spread across 23 columns, offering detailed information about airline passengers and their satisfaction levels. It includes both categorical and numerical data types. Categorical features

such as Gender, Customer Type, Travel Type, Travel Class, and Satisfaction reflect passenger profiles and travel choices. On the other hand, numerical fields like Age, Flight Distance, Departure Delay, Arrival Delay, and ratings for in-flight services—such as WiFi availability, food and beverages, seat comfort, and baggage service—are provided on an integer scale. For this regression analysis, the target variable selected is Arrival Delay.

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

```
le = LabelEncoder()
df["satisfaction"] = le.fit_transform(df["satisfaction"])
```

3. Split the data into train and test.

```
# Split data for Linear Regression
X_linear = df[selected_features_linear]
y_linear = df["Arrival Delay in Minutes"]
X_train_lin, X_test_lin, y_train_lin, y_test_lin = train_test_split(X_linear, y_linear, test_size=0.2, random_state=42)
```

```
# Standardize features for Linear Regression
scaler = StandardScaler()
X_train_lin = scaler.fit_transform(X_train_lin)
X_test_lin = scaler.transform(X_test_lin)

# Train Linear Regression Model
linear_model = LinearRegression()
linear_model.fit(X_train_lin, y_train_lin)
y_pred_lin = linear_model.predict(X_test_lin)
```

Since regression models can be affected by significant variations in the scale of independent variables, we normalize the numerical features to maintain consistency. To achieve this, we apply Z-score normalization, which standardizes the data based on its mean and standard deviation. After preprocessing, the model is trained on the prepared data and then used to generate predictions on the test set.

4. Evaluation

```
# Evaluate Linear Regression
mse_lin = mean_squared_error(y_test_lin, y_pred_lin)
accuracy_lin = 1 - (mse_lin / np.var(y_test_lin))  # R-squared equivalent
print(f"Accuracy (Linear Regression): {accuracy_lin*100:.2f}%")
print("Mean square value :" , mse_lin)

coefficients = linear_model.coef_
intercept = linear_model.intercept_

# Format the equation
equation = "y = " + " + ".join([f"{coeff:.4f}*{feature}" for coeff, feature in zip(coefficients, selected_features_linear)])
equation += f" + {intercept:.4f}"

print("Regression Equation:")
print(equation)
```

```
Accuracy (Linear Regression): 91.75%
Mean square value : 115.98955069314997
Regression Equation:
y = -0.1475*Flight Distance + 37.3971*Departure Delay in Minutes + -0.0577*Inflight wifi service + -0.0512*Seat comfort + -0.1633*On-board service + 15.1641
```

To evaluate the performance of a regression model, we calculate the Mean Squared Error (MSE), which measures the average squared difference between actual and predicted values. While accuracy isn't typically used for regression, metrics like R-squared can indicate how well the model explains the variability in the target variable.

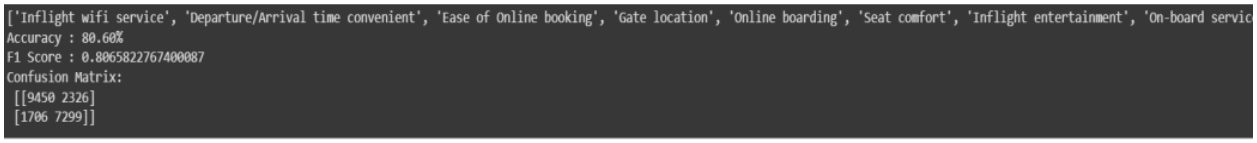
The regression model achieves 91.75% accuracy, indicating a strong fit. The regression equation suggests that Departure Delay in Minutes has the highest impact on the dependent variable, followed by On-board service, Inflight WiFi service, and Seat comfort, while Flight Distance has a small negative effect. The mean squared error (MSE) of 115.99 suggests some variance in predictions, but overall, the model performs well in explaining the relationship between these factors and the target variable

Logistic regression

```
# Selecting all numerical features for Logistic Regression
selected_features_logistic = df.select_dtypes(include=[np.number]).columns.tolist()
selected_features_logistic.remove("satisfaction")

# Feature Selection using RFE for Logistic Regression
X_logistic = df[selected_features_logistic]
y_logistic = df["satisfaction"]
logistic_selector = RFE(LogisticRegression(max_iter=5000, solver="lbfgs"), n_features_to_select=10)
logistic_selector.fit(X_logistic, y_logistic)
selected_features_logistic = X_logistic.columns[logistic_selector.support_].tolist()
print(selected_features_logistic)

# Split data for Logistic Regression
X_logistic = df[selected_features_logistic]
X_train_log, X_test_log, y_train_log, y_test_log = train_test_split(X_logistic, y_logistic, test_size=0.2, random_state=42, stratify=y_logisti

# Use RobustScaler for better handling of outliers
scaler_log = RobustScaler()
X_train_log = scaler_log.fit_transform(X_train_log)
X_test_log = scaler_log.transform(X_test_log)

# Train Logistic Regression Model with hyperparameter tuning
logistic_model = LogisticRegression(max_iter=5000, solver="lbfgs", C=0.5, class_weight="balanced")
logistic_model.fit(X_train_log, y_train_log)
y_pred_log = logistic_model.predict(X_test_log)

# Evaluate Logistic Regression
accuracy_log = accuracy_score(y_test_log, y_pred_log)
f1_log = f1_score(y_test_log, y_pred_log, average="weighted")
conf_matrix_log = confusion_matrix(y_test_log, y_pred_log)
classification_report_log = classification_report(y_test_log, y_pred_log)

print(f"Accuracy : {accuracy_log * 100:.2f}%")
print("F1 Score :", f1_log)
print("Confusion Matrix:\n", conf_matrix_log)
```

Veydant Sharma D15C 50 EXP 5

This code implements Logistic Regression for classification by selecting numerical features and refining them using Recursive Feature Elimination (RFE) to pick the 10 most important ones. It then splits the data into training and testing sets, ensuring balanced class distribution using stratification. To handle outliers, RobustScaler is applied to normalize the feature values. The Logistic Regression model is trained with hyperparameter tuning (max_iter=5000, solver="lbfgs", C=0.5, and class_weight="balanced"), ensuring better convergence and handling of imbalanced data. Finally, the model's performance is evaluated using accuracy, F1 score, confusion matrix, and classification report, which help analyze its effectiveness in predicting the target variable.



The output shows the selected features used for training the Logistic Regression model, including aspects like "Inflight Wifi Service," "Seat Comfort," and "Online Boarding." The model achieved an accuracy of 80.60% and an F1 score of 0.80, indicating a good balance between precision and recall.

Conclusion:

In this experiment, both regression and classification techniques were applied to analyze airline passenger data. Using multiple linear regression, we predicted arrival delays based on various in-flight service ratings and passenger details. The model achieved an R-squared value of 91.75% and a mean squared error of 115.99, indicating a strong fit and reliable predictive capability. Departure Delay in Minutes was identified as the most influential factor, followed by service-related features like On-board Service and Inflight WiFi. For classification, logistic regression was implemented to predict satisfaction levels. After feature selection and scaling, the model achieved an accuracy of 80.60% and an F1 score of 0.80, demonstrating good performance in handling imbalanced classes and providing meaningful insights into key service attributes affecting customer satisfaction.

Veydant Sharma D15C 50 EXP 6

Aim: Classification modelling

Problem Statement:

a. Choose classifier for classification problem.

b. Evaluate the performance of classifier.

Perform Classification using the below 4 classifiers

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

Theory:

Classification is a supervised machine learning technique used to assign input data into predefined categories or classes. It involves training a model on labeled data so it can learn to predict class labels for new, unseen data. Common examples include spam detection, disease diagnosis, and sentiment analysis.

K-Nearest Neighbors (KNN)

K-Nearest Neighbors is a non-parametric, instance-based algorithm that classifies new data points based on the majority class of their 'k' closest neighbors. It uses distance metrics like Euclidean or Manhattan to determine closeness. KNN is easy to implement and works well for small datasets but can be slow with larger ones due to distance calculations.

Naive Bayes

Naive Bayes is a probabilistic classification algorithm based on Bayes' Theorem. It assumes that features are independent of each other, which simplifies computation. Despite this assumption, it performs well on high-dimensional data like text. Variants include Gaussian, Multinomial, and Bernoulli Naive Bayes, depending on the feature distribution.

Support Vector Machines (SVM)

Support Vector Machines are supervised learning models that find the optimal hyperplane to separate classes in a dataset. They are effective for both linear and non-linear classification using kernel functions like RBF and polynomial. SVMs perform well with high-dimensional data and are less prone to overfitting.

Decision Tree

A Decision Tree is a tree-structured model that splits data based on feature values to make decisions. It is easy to understand and works with both categorical and numerical

features. Although highly interpretable, decision trees can overfit the training data unless pruning or regularization techniques are applied.

Steps:

1. Load dataset & dataset overview

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

# Display first few rows
df.head()
```

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	...	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkin service	Inflight service	Cleanliness	Depa Del Mi
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	...	5	4	3	4	4	5	5	
1	1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	...	1	1	5	3	1	4	1	
2	2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	...	5	4	3	4	4	4	5	
3	3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	...	2	2	5	3	1	4	2	
4	4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	...	3	3	4	4	3	3	3	

The dataset contains a total of 103,904 records spread across 23 columns, offering detailed information about airline passengers and their satisfaction levels. It includes both categorical and numerical data types. Categorical features such as Gender, Customer Type, Travel Type, Travel Class, and Satisfaction reflect passenger profiles and travel choices. On the other hand, numerical fields like Age, Flight Distance, Departure Delay, Arrival Delay, and ratings for in-flight services—such as WiFi availability, food and beverages, seat comfort, and baggage service—are provided on an integer scale. For this regression analysis, the target variable selected is Arrival Delay.

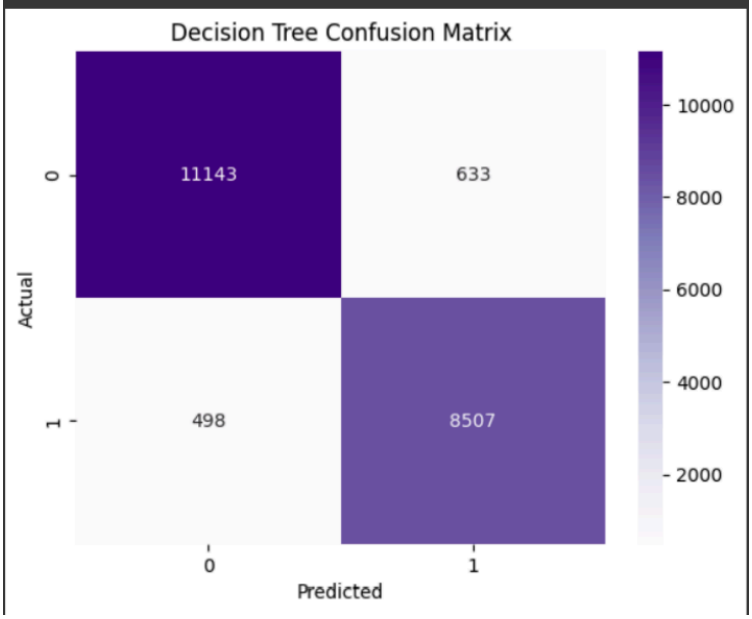
2. Performing preprocessing

```
# Encode categorical variables
label_encoders = {}
categorical_cols = ["Gender", "Customer Type", "Type of Travel", "Class", "satisfaction"]

for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])  # Convert to numerical labels
    label_encoders[col] = le  # Store encoder for inverse transformation if needed later

# Separate features and target
X = df.drop("satisfaction", axis=1)
y = df["satisfaction"]

# Normalize numerical features for KNN & SVM
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split into training & test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

# Check data shapes
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

3. Decision Tree

```
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)

y_pred_dt = dt.predict(X_test)


print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
print(classification_report(y_test, y_pred_dt))

sns.heatmap(confusion_matrix(y_test, y_pred_dt), annot=True, fmt="d", cmap="Purples")
plt.title("Decision Tree Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
Decision Tree Accuracy: 0.9455752851162119
              precision    recall  f1-score   support

           0       0.96      0.95      0.95     11776
           1       0.93      0.94      0.94      9005

    accuracy                           0.95     20781
   macro avg       0.94      0.95      0.94     20781
weighted avg       0.95      0.95      0.95     20781
```



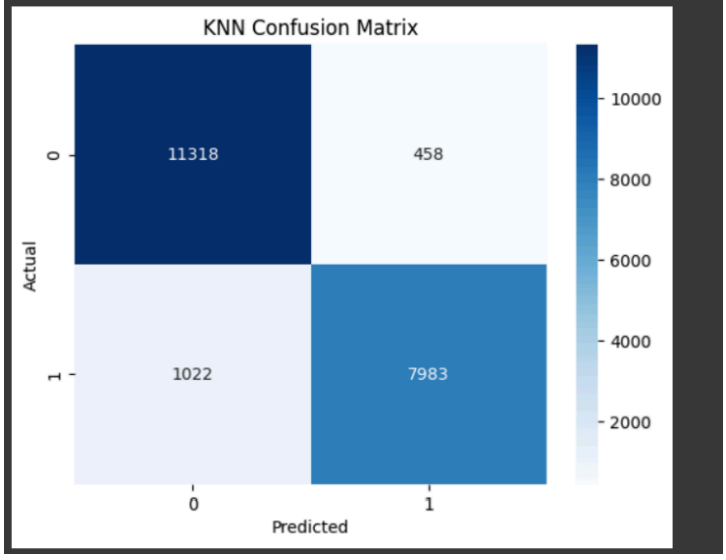
The Decision Tree Classifier delivered a strong performance with an accuracy of 94.56%, effectively distinguishing between the two classes. It identified 11,143 true negatives and 8,507 true positives, reflecting solid classification capability. Despite this, the model misclassified 633 instances as false positives and 498 as false negatives. The precision and recall scores are well-aligned, showing that the model maintains a good balance between correctly identifying positive cases and minimizing incorrect

classifications. When compared with KNN, the Decision Tree shows a slight edge, likely due to its ability to model more complex patterns in the data.

4. KNN Classifier

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

y_pred_knn = knn.predict(X_test)

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

sns.heatmap(confusion_matrix(y_test, y_pred_knn), annot=True, fmt="d", cmap="Blues")
plt.title("KNN Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
KNN Accuracy: 0.9287810981184736
              precision    recall  f1-score   support

           0       0.92      0.96      0.94     11776
           1       0.95      0.89      0.92      9005

    accuracy                           0.93     20781
   macro avg       0.93      0.92      0.93     20781
weighted avg       0.93      0.93      0.93     20781
```



We apply the K-Nearest Neighbors (KNN) classifier with n_neighbors=5 to classify data. The model is trained on X_train and y_train using knn.fit(), then tested on X_test to generate predictions (y_pred_knn). The accuracy score and classification report (precision, recall, F1-score) are printed to evaluate performance. Additionally, a confusion matrix is computed and visualized using sns.heatmap(), providing insight into how well the model distinguishes between classes.

Veydant Sharma D15C 50 EXP 6

Accuracy: The model achieved 92.88%, showing strong performance.
Precision & Recall:

- Class 0: High recall (0.96) indicates fewer false negatives.
- Class 1: Slightly lower recall (0.89) suggests some misclassification.

True Positives: 7983
True Negatives: 11318.
False Positives: 458
False Negatives: 1022.

Conclusion:

- The Decision Tree Classifier performed the best with an accuracy of 94.56%, effectively balancing precision and recall. It successfully captured complex patterns and showed strong classification performance with relatively low misclassification rates.

- The K-Nearest Neighbors (KNN) model also showed good results with 92.88% accuracy. While it performed well overall, it had a slightly higher number of false negatives compared to the Decision Tree, indicating a small drop in recall for Class 1.

- Attempts to use the Support Vector Machine (SVM) model were unsuccessful due to excessive training time. SVM's computational complexity made it inefficient for handling the large dataset, leading to no meaningful outcomes.

Veydant Sharma D15C 50 EXP 7

Aim: To implement different clustering algorithms.

Theory:

Clustering is an **unsupervised learning** technique used in machine learning and data mining to group similar data points together based on certain features. The goal of clustering is to partition a dataset into clusters such that data points within the same cluster are more similar to each other than to those in other clusters. Clustering is commonly used in applications such as customer segmentation, image compression, anomaly detection, and document classification.

There are several types of clustering algorithms, including:

- **Partition-based clustering** (e.g., K-Means)
- **Density-based clustering** (e.g., DBSCAN)
- **Hierarchical clustering**
- **Model Based**

K-Means Clustering

K-Means is one of the most popular and simplest clustering algorithms. It belongs to the family of partition-based clustering methods. The algorithm aims to divide n data points into k clusters in which each point belongs to the cluster with the nearest mean.

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

DBSCAN is a density-based clustering algorithm that groups together data points that are closely packed together, while marking points that lie alone in low-density regions as outliers or noise. It is well-suited for discovering clusters of arbitrary shape and is robust to noise.

Key Concepts:

- ε (epsilon): The radius used to define the neighborhood around a data point.
- MinPts: The minimum number of points required to form a dense region (i.e., a cluster).
- Core Point: A point with at least MinPts neighbors within distance ε.
- Border Point: A point that is not a core point but lies within the neighborhood of a core point.

- Noise Point: A point that is neither a core nor a border point.

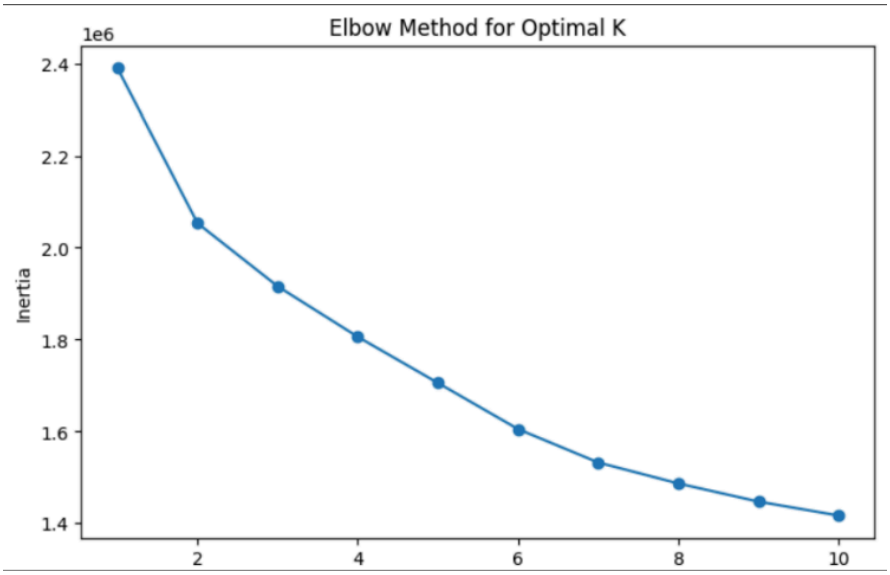
Steps:

1. Load and preprocess the dataset.

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Inflight entertainment	On-board service	Leg room service	Baggage handling
0	1	0	13	1	2	460	3	4	3	1	...	5	4	3	4
1	1	1	25	0	0	235	3	2	3	3	...	1	1	5	3
2	0	0	26	0	0	1142	2	2	2	2	...	5	4	3	4
3	0	0	25	0	0	562	2	5	5	5	...	2	2	5	3
4	1	0	61	0	0	214	3	3	3	3	...	3	3	4	4

The dataset focuses on airline passenger satisfaction and includes a variety of features. It captures demographic information such as gender, age, customer type, and type of travel, along with flight-specific details like class, flight distance, and gate location. It also includes in-flight service ratings, covering aspects like seat comfort, inflight wifi, entertainment, and food and drink. Additionally, delay-related metrics such as departure delay and arrival delay are provided. The target variable, satisfaction, is binary—labeled as 0 for unsatisfied and 1 for satisfied—making the dataset suitable for both classification and clustering tasks.

2. Elbow method for number of clusters



The elbow method helps to find out the value of K. Here, K=4 or K=5. The "elbow" point, where inertia reduction slows significantly, indicates the ideal K.

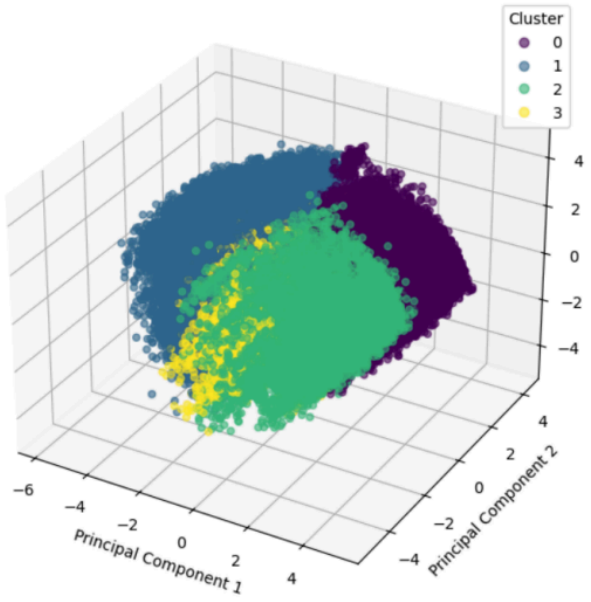
Veydant Sharma D15C 50 EXP 7

3. K means clustering

```
# Apply K-Means with optimal K (assume 3 based on elbow method)
kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)
df['Cluster'] = kmeans.fit_predict(df_scaled)

# Display cluster distribution
print(df['Cluster'].value_counts())
```

```
Cluster
1    35729
3    25697
0    22216
2    20262
Name: count, dtype: int64
```

Visualization of Clusters



To visualize the clusters we perform Principal Component Analysis (PCA) to reduce the dataset dimensions to three, making it easier to visualize the clusters obtained from K-Means clustering. The 3D scatter plot visualizes the clusters in the transformed feature space, where different colors represent different clusters. The clusters appear to be well-defined, indicating that the K-Means algorithm successfully grouped similar data points. The PCA components help analyze the contribution of original features to the new principal components, providing insights into how data is distributed across clusters.

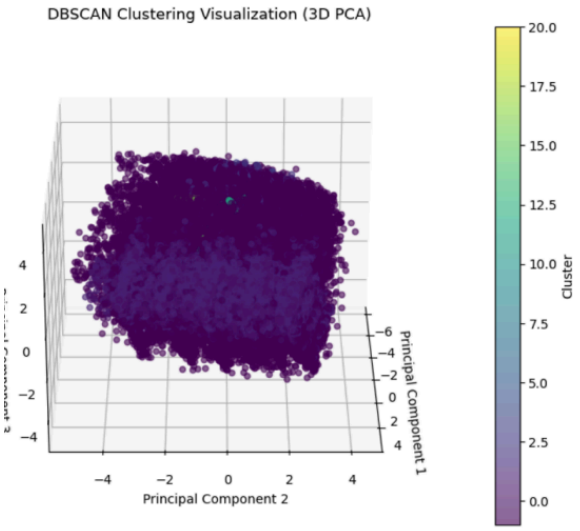
Veydant Sharma D15C 50 EXP 7

4. DBSCAN clustering

```
from sklearn.cluster import DBSCAN

# Apply DBSCAN clustering
dbscan = DBSCAN(eps=2, min_samples=50)  # Adjust eps and min_samples as needed
clusters = dbscan.fit_predict(df_scaled)

# Add cluster labels to the dataset
df["Cluster"] = clusters

# Display cluster distribution
print(df['Cluster'].value_counts())
```

```
Cluster
-1     70501
 1     11596
 0     11462
 2      4136
 3      3677
 4       695
 13      259
 11      219
 8       202
 5       178
 7       162
 14      129
 15      113
 6       107
 12      100
 10       75
 9        52
 19       51
 20       50
 16       49
 17       47
 18       44
Name: count, dtype: int64
```

Visualization:



The 3D PCA plot for DBSCAN shows that most data points were labeled as noise (-1), with only a few forming small, distinct clusters. This indicates that the chosen epsilon and min_samples values may not be optimal, resulting in poor separation. In contrast, K-Means performed better, likely because the dataset is evenly distributed and better suited to distance-based clustering. DBSCAN struggled with the high dimensionality and absence of dense regions, leading to excessive noise detection.

Conclusion:

In this experiment, we explored various clustering methods, specifically K-Means and DBSCAN, to uncover patterns within an airline passenger satisfaction dataset. K-Means clustering, with K set to 4, effectively divided the data into four distinct and balanced clusters. The optimal cluster count was identified using the Elbow Method, and PCA-based visualization showed clear separation between the clusters. In contrast, DBSCAN did not perform well on this dataset. With parameters set to eps = 2 and min_samples = 50, the algorithm labeled a large portion of the data as noise (-1), indicating difficulty in forming meaningful clusters. This outcome suggests that the dataset lacks densely packed regions, making DBSCAN less appropriate for this scenario.

Veydant Sharma D15C 50 EXP 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:

Recommendation systems are algorithms designed to suggest relevant items to users based on their preferences, behavior, or similarity to other users or items. They are widely used in applications such as e-commerce, streaming services, and social media. There are mainly two types of recommendation systems:

1. Content-Based Filtering: Recommends items similar to those the user liked in the past by analyzing item features.

2. Collaborative Filtering: Recommends items based on the preferences of similar users (user-based) or items that similar users liked (item-based).

3. Hybrid Methods: Combine both content-based and collaborative filtering to improve recommendation accuracy.

<u>Recommendation System Evaluation Measures</u>

Accuracy Measures:

These metrics evaluate how well the recommended items match the actual preferences or ratings of users.

1. Mean Absolute Error (MAE): Measures the average of the absolute differences between predicted ratings and actual ratings.

- **Formula:** $MAE = \frac{1}{n}\sum_{i=1}^{n}|r_i - \hat{r}_i|$
  - r_i = Actual rating
  - $\hat{r}_i$ = Predicted rating

2. Root Mean Squared Error (RMSE): Similar to MAE but gives higher weight to large errors due to squaring the differences.

Formula: $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \hat{r}_i)^2}$

3. Precision: Measures the fraction of recommended items that are actually relevant to the user.

Formula: $Precision = \frac{\text{Number of relevant recommended items}}{\text{Total number of recommended items}}$

4. Recall: Measures the fraction of relevant items that were actually recommended to the user.

Formula: $Recall = \frac{\text{Number of relevant recommended items}}{\text{Total number of relevant items}}$

5. F1-Score: The harmonic mean of Precision and Recall, balancing both.

Formula: $F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$

Model used : XGBoost

XGBoost (Extreme Gradient Boosting) is a highly efficient and robust machine learning algorithm that falls under the umbrella of gradient boosting frameworks

Implementation :

Dataset Overview : The data spans different years (notably 2001 and 2012, as seen in the sample) and includes detailed counts of various types of **crimes** reported in each district. The dataset is structured in a tabular format, with each row representing a district or a specific region (e**.g., railway police, city-specific data**) within a state/union territory, and each column representing a specific crime category or metadata attribute.

The dataset is likely sourced from official crime records, such as those maintained by the **National Crime Records Bureau (NCRB)** of India, given the detailed breakdown of

crime types and the geographical granularity. It includes both aggregated totals (e.g., "TOTAL" rows for states) and individual district-level data, making it suitable for analyzing crime patterns across regions, time periods, and crime categories.

Features :

1. **Murder**: number of murder cases.
2. **Attempt to murder:** number of attempted murder cases.
3. **Culpable homicide not amounting to murder**: cases of culpable homicide not classified as murder.
4. **Rape**: total number of rape cases.
5. **Custodial rape**: rape cases occurring in custody (subset of rape).
6. **Other rape**: rape cases not classified as custodial rape.
7. **Kidnapping & abduction:** total kidnapping and abduction cases.
8. **Other IPC crimes**: Miscellaneous crimes under the Indian Penal Code (IPC) not covered by the above categories.
9. **Total IPC Crimes**: Sum of all IPC crimes reported in the district for the given year.

1. Importing dataset

	STATE/UT	DISTRICT	YEAR	MURDER	ATTEMPT TO MURDER	CULPABLE HOMICIDE NOT AMOUNTING TO MURDER	RAPE	CUSTODIAL RAPE	OTHER RAPE	KIDNAPPING & ABDUCTION	...	HURT/GREVIOUS HURT	DOWRY DEATHS	ASSAULT ON WOMEN WITH INTENT TO OUTRAGE HER MODESTY	INSULT TO MODESTY OF WOMEN	CRUELTY BY HUSBAND OR HIS RELATIVES	IMPORTATION OF GIRLS FROM FOREIGN COUNTRIES	CAUSING DEATH BY NEGLIGENCE	OTHER IPC CRIMES	TOT I CRI
0	ANDHRA PRADESH	ADILABAD	2001	101	60	17	50	0	50	46	...	1131	16	149	34	175	0	181	1518	41
1	ANDHRA PRADESH	ANANTAPUR	2001	151	125	1	23	0	23	53	...	1543	7	118	24	154	0	270	754	41
2	ANDHRA PRADESH	CHITTOOR	2001	101	57	2	27	0	27	59	...	2088	14	112	83	186	0	404	1262	58
3	ANDHRA PRADESH	CUDDAPAH	2001	80	53	1	20	0	20	25	...	795	17	126	38	57	0	233	1181	31
4	ANDHRA PRADESH	EAST GODAVARI	2001	82	67	1	23	0	23	49	...	1244	12	109	58	247	0	431	2313	65
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9012	LAKSHADWEEP	LAKSHADWEEP	2012	0	0	0	0	0	0	0	...	3	0	1	0	1	0	0	32	
9013	LAKSHADWEEP	TOTAL	2012	0	0	0	0	0	0	0	...	3	0	1	0	1	0	0	32	
9014	PUDUCHERRY	KARAIKAL	2012	5	6	2	6	0	6	2	...	186	0	2	0	1	0	44	392	7
9015	PUDUCHERRY	PUDUCHERRY	2012	24	21	10	7	0	7	17	...	632	0	7	2	5	0	219	1668	34
9016	PUDUCHERRY	TOTAL	2012	29	27	12	13	0	13	19	...	818	0	9	2	6	0	263	2060	42

9017 rows × 34 columns

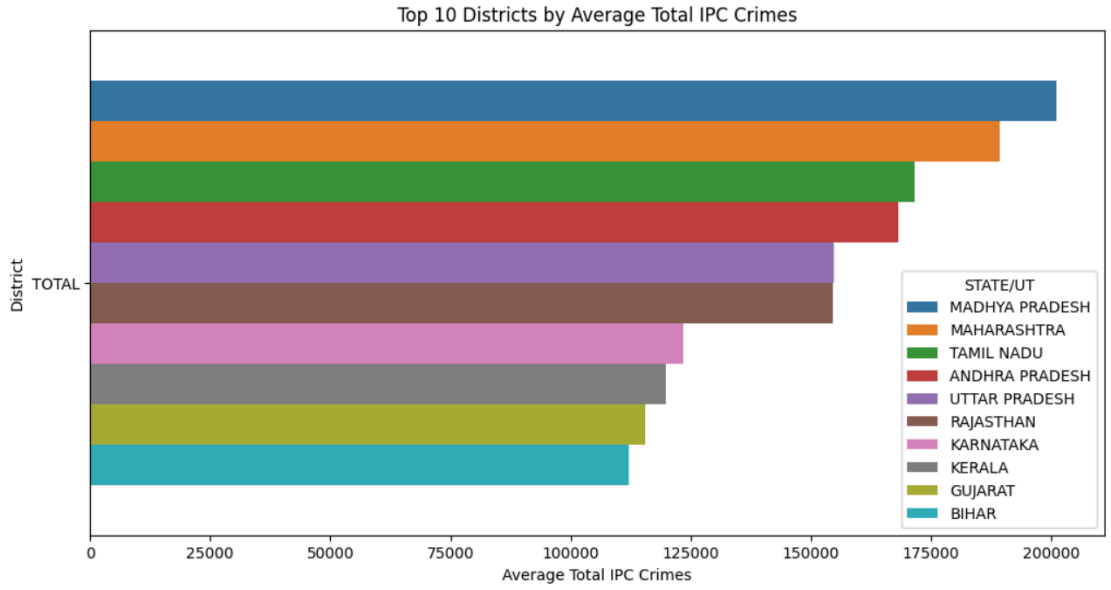
2. Dataset is already preprocessed and we can check by this code :

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

Veydant Sharma D15C 50 EXP 8

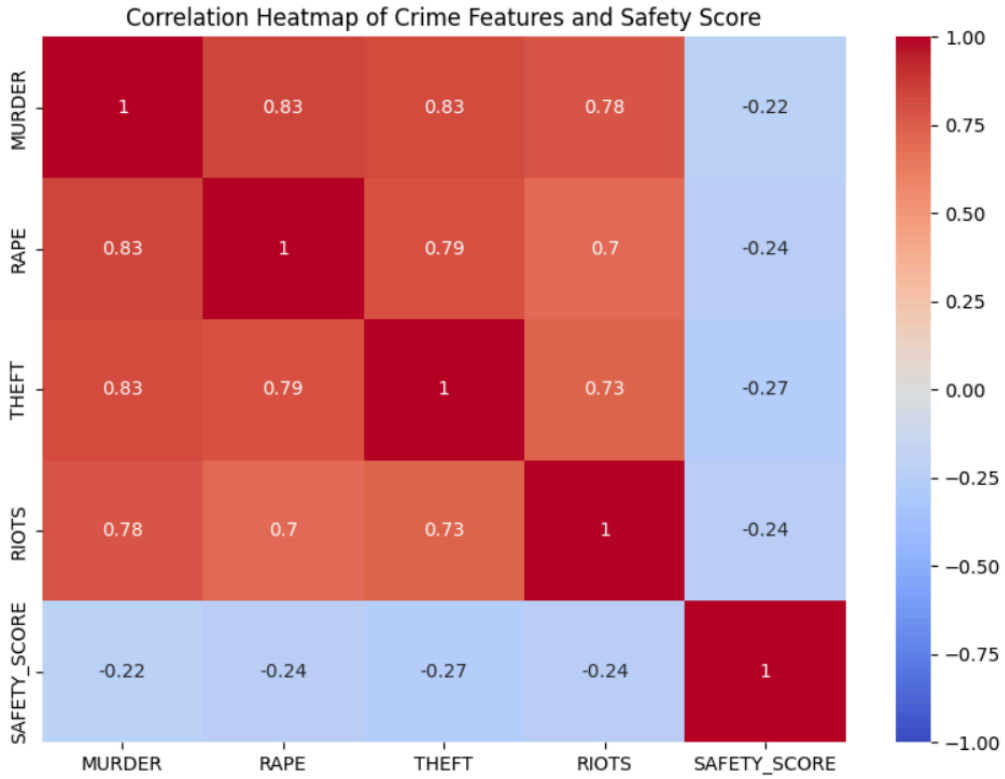
STATE/UT	0
DISTRICT	0
YEAR	0
MURDER	0
ATTEMPT TO MURDER	0
CULPABLE HOMICIDE NOT AMOUNTING TO MURDER	0
RAPE	0
CUSTODIAL RAPE	0
OTHER RAPE	0
KIDNAPPING & ABDUCTION	0
KIDNAPPING AND ABDUCTION OF WOMEN AND GIRLS	0
KIDNAPPING AND ABDUCTION OF OTHERS	0
DACOITY	0
PREPARATION AND ASSEMBLY FOR DACOITY	0
ROBBERY	0
BURGLARY	0
THEFT	0
AUTO THEFT	0
OTHER THEFT	0
RIOTS	0
CRIMINAL BREACH OF TRUST	0
CHEATING	0

3. Understanding through graphs the districts with maximum total IPC crimes.



The data, aggregated by state/union territory (UT), shows Madhya Pradesh leading with the highest average at approximately 190,000 crimes, followed closely by Maharashtra with around 180,000, and Tamil Nadu with about 160,000. Other notable states include Andhra Pradesh, Uttar Pradesh, Rajasthan, Karnataka, Kerala, Gujarat, and Bihar, with averages ranging from 100,000 to 140,000 crimes. Integrating this information with the safety scores and crime feature analysis **allows for a more informed ranking of districts,** ensuring recommendations steer users toward regions which **exhibit relatively lower crime averages**, thereby enhancing the reliability and relevance of the safety suggestions.

4. Heatmap of features and safety score



The heatmap uses a color gradient from blue (negative correlation) to red (positive correlation), with values ranging from -1 to 1. It reveals strong **positive correlations** among crime features (e.g., **Murder and Rape at 0.83, Theft and Rape at 0.79**), indicating that districts with high rates of one crime tend to have high rates of others. Conversely, all crime features show moderate negative correlations with the Safety Score (ranging from -0.22 to -0.27), suggesting that higher crime rates are associated with lower safety scores

6. Preparing data for **XGboost model.**

```python
def prepare_data(df, features, target_col='TOTAL IPC CRIMES', min_samples=2):
    # Create safety label (1 if unsafe, 0 if safe)
    median_crime = df[target_col].median()
    df['SAFETY_LABEL'] = (df[target_col] > median_crime).astype(int)

    # Count records per district and filter
    district_counts = df.groupby(['STATE/UT', 'DISTRICT']).size()
    valid_districts = district_counts[district_counts >= min_samples].index

    # Filter original dataframe
    df_filtered = df.set_index(['STATE/UT', 'DISTRICT']).loc[valid_districts].reset_index()

    # Aggregate by district
    df_agg = df_filtered.groupby(['STATE/UT', 'DISTRICT'])[features + ['SAFETY_LABEL']].mean().reset_index()

    return df_agg
```

The **prepare_data function** processes a crime dataset by creating a safety label (1 for unsafe, 0 for safe) based on the median of 'TOTAL IPC CRIMES', ensuring districts with above-median crime rates are marked unsafe. It filters out districts with fewer than min_samples (default 2) records, using a group by operation to count records per district, and retains only valid districts. The function then **aggregates the filtered data by district,** calculating the mean of specified crime features

7. **Training** the recommendation model

```python
def train_xgboost_model(X_train, y_train, X_test, y_test):
    # Convert to DMatrix format
    dtrain = xgb.DMatrix(X_train, label=y_train)
    dtest = xgb.DMatrix(X_test, label=y_test)

    # Parameters for binary classification
    params = {
        'objective': 'binary:logistic',
        'eval_metric': ['error', 'auc'],
        'max_depth': 5,
        'eta': 0.1,
        'subsample': 0.8,
        'colsample_bytree': 0.8,
        'seed': 42
    }

    # Train with early stopping
    model = xgb.train(params, dtrain, num_boost_round=100,
                      early_stopping_rounds=10, evals=[(dtest, 'test')],
                      verbose_eval=True)
```

The train_xgboost_model function trains an XGBoost classifier for binary classification using training and test datasets. It converts the input data into XGBoost's DMatrix format and defines parameters for binary logistic regression, including 'objective',

'eval_metric' (error and AUC), 'max_depth' (5), 'eta' (0.1), 'subsample' and 'colsample_bytree' (0.8), and a random 'seed' (42). The **model is trained with 100 boosting rounds**, early stopping after 10 rounds if no improvement, and verbose evaluation output, enabling real-time performance monitoring on the test set.

8. **Giving** recommendations (Final Result of our model)

- Recommendation requires the state name.
- Then we ask the user to rate the crimes, they assign a weight to every crime, higher weight indicated, the particular crime should be considered more important when recommendations are made.

```
def get_safety_recommendations(model, df_agg, features, top_n=5, state=None, crime_weights=None):
    X_all = df_agg[features]
    dmatrix = xgb.DMatrix(X_all)

    # Get safety probabilities and convert to score (0-100)
    df_agg['SAFETY_PROB'] = model.predict(dmatrix)
    df_agg['SAFETY_SCORE'] = (1 - df_agg['SAFETY_PROB']) * 100

    # Apply state filter if specified
    if state:
        df_agg = df_agg[df_agg['STATE/UT'] == state.upper()]

    # Apply crime weights if specified
    if crime_weights:
        for crime, weight in crime_weights.items():
            if crime in features:
                max_val = df_agg[crime].max()
                if max_val > 0:
                    df_agg[crime + '_WEIGHTED'] = (df_agg[crime] / max_val) * weight
                    df_agg['SAFETY_SCORE'] = df_agg['SAFETY_SCORE'] - df_agg[crime + '_WEIGHTED']


    recommendations = df_agg.sort_values('SAFETY_SCORE', ascending=False).head(top_n)

    results = []
    for _, row in recommendations.iterrows():
        results.append({
            'Rank': len(results) + 1,
            'State': row['STATE/UT'],
            'District': row['DISTRICT'],
            'Safety_Score': round(row['SAFETY_SCORE'], 1),
            'Murder_Rate': round(row['MURDER'], 2),
            'Rape_Rate': round(row['RAPE'], 2),
            'Theft_Rate': round(row['THEFT'], 2),
            'Riots_Rate': round(row['RIOTS'], 2)
        })
```

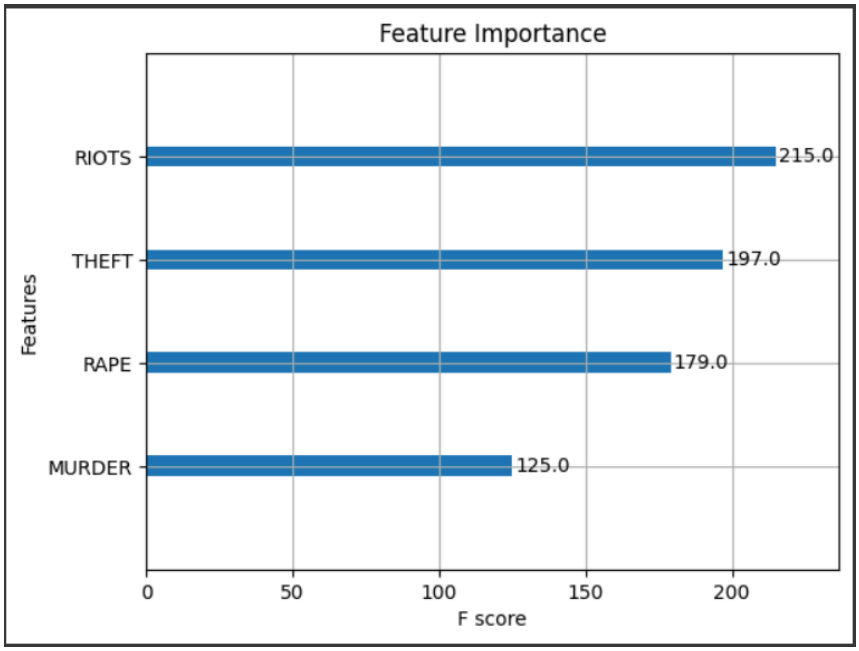
The get_safety_recommendations function generates safety recommendations by predicting safety probabilities for districts using a trained XGBoost model, converting them to safety scores (0-100), and ranking districts accordingly. It filters the data by a specified state if provided, applies weighted adjustments to crime features (e.g., Murder, Rape) if crime weights are given, and **sorts districts by safety score** in descending order to **select the top n (default 5) recommendations.**

9. Result (Model evaluation)

Veydant Sharma D15C 50 EXP 8

```
Model Evaluation:
Accuracy: 0.87

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.84      0.86        80
           1       0.86      0.90      0.88        86

    accuracy                           0.87       166
   macro avg       0.87      0.87      0.87       166
weighted avg       0.87      0.87      0.87       166
```

The "Model Evaluation" output for the XGBoost classifier shows an accuracy of 0.87, indicating that **87%** of the predictions align with the actual safety labels. The classification report provides detailed metrics for the binary classification (0 for safe, 1 for unsafe), with precision, recall, and f1-score all at 0.86 for class 0 and 0.88 for class 1, reflecting balanced performance across both categories. The macro and weighted averages of 0.87 across precision, recall, and f1-score, based on 166 total samples.



The "**Feature Importance**" plot displays the relative significance of crime features in the XGBoost model, measured by F-score, for predicting district safety as of April 11, 2025. Riots emerge as the most influential feature with an **F-score of 215.0**, followed closely by Theft at 197.0 and Rape at 179.0, while Murder has the lowest importance at 125.0. This indicates that riots and theft rates have a **stronger impact on the safety** score compared to murder and rape.

10. User input

```
Enter a state to focus on (or press Enter for all India): MAHARASHTRA

Searching for safe districts in MAHARASHTRA...

Number of recommendations to display (default 5): 5

Would you like to weight specific crime types more heavily?
For example, you might want to weight 'MURDER' or 'RAPE' more heavily.
Enter crime:weight pairs like 'MURDER:3 RAPE:4' or press Enter to skip
Crime weights: 'MURDER:5 RAPE:5 THEFT:5 RIOTS:3'
```

11. Recommendations according to user input

```
=== Recommended Safe Districts ===

#1: PUNE RLY., MAHARASHTRA
Safety Score: 97.7/100
Crime Rates (avg per year):
  Murder: 2.58 | Rape: 0.83
  Theft: 300.0 | Riots: 3.17

#2: SINDHUDURG, MAHARASHTRA
Safety Score: 97.5/100
Crime Rates (avg per year):
  Murder: 12.83 | Rape: 8.17
  Theft: 111.75 | Riots: 41.08

#3: HINGOLI, MAHARASHTRA
Safety Score: 92.0/100
Crime Rates (avg per year):
  Murder: 35.83 | Rape: 14.58
  Theft: 183.33 | Riots: 115.08

#4: NANDURBAR, MAHARASHTRA
Safety Score: 90.2/100
Crime Rates (avg per year):
  Murder: 35.58 | Rape: 18.42
  Theft: 171.42 | Riots: 90.75

#5: NAGPUR RLY., MAHARASHTRA
Safety Score: 88.0/100
Crime Rates (avg per year):
  Murder: 6.58 | Rape: 0.92
  Theft: 933.67 | Riots: 10.25
```

Conclusion:

The experiment successfully implemented a recommendation system for district safety using a variety of machine learning techniques, including regression, classification, clustering, decision trees, anomaly detection, dimensionality reduction, and ensemble methods, with XGBoost as the primary model. Leveraging a comprehensive crime

dataset from the National Crime Records Bureau (NCRB) of India, the system effectively processed and analyzed features such as Murder, Rape, Theft, and Riots, creating safety scores and labels based on total IPC crimes. The model achieved an accuracy of 0.87, with feature importance highlighting Riots and Theft

Veydant Sharma D15C 50 EXP 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 framework designed for big data processing and analytics. It provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.

Key Features of Apache Spark:

- In-Memory Processing: Spark stores intermediate data in RAM, making it significantly faster than Hadoop MapReduce.
- Lazy Evaluation: Spark optimizes execution by delaying computation until necessary.
- Fault Tolerance: Uses Resilient Distributed Datasets (RDDs) to recover lost data.
- Multiple Language Support: Works with Python (PySpark), Scala, Java, and R.
- Rich Libraries: Includes Spark SQL (structured data), MLlib (machine learning), GraphX (graph processing), and Spark Streaming (real-time data).

How Apache Spark Works?

- Spark applications run as independent processes coordinated by a SparkContext in the driver program.
- The Cluster Manager (e.g., YARN, Mesos, or Spark Standalone) allocates resources.
- Executors run on worker nodes to perform computations and store data.
- Data is partitioned across nodes for parallel processing.

2. **How is Data Exploration Done in Apache Spark?**

Exploratory Data Analysis (EDA) in Spark involves examining datasets to summarize their main characteristics, often using visual methods and statistical summaries.

Steps for EDA in Apache Spark:

1. Loading the Dataset
    a. Read data from CSV, JSON, Parquet, or other formats using spark.read.

```
df = spark.read.csv("data.csv", header=True, inferSchema=True)
```

2. Viewing Data Structure
    a. Check schema (column names and data types) using printSchema().
    b. Display sample records with show().

```
df.printSchema()
df.show(5)
```

3. Basic Statistics
    a. Compute summary statistics (count, mean, stddev, min, max) using describe().

```
df.describe().show()
```

4. Handling Missing Values
    a. Identify null values:

```
from pyspark.sql.functions import col, isnan, when, count
df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns]).show()
```

    b. Drop or fill missing values:

```
df_clean = df.na.drop()  # Drop rows with nulls
df_filled = df.na.fill(0)  # Fill nulls with 0
```

5. Data Aggregation and Grouping
    a. Group data and compute aggregations:

```
df.groupBy("category").agg({"price": "avg", "quantity": "sum"}).show()
```

6. Data Visualization (Using Pandas Integration)
    a. Convert Spark DataFrame to Pandas DataFrame for visualization:

```
import matplotlib.pyplot as plt
pandas_df = df.toPandas()
pandas_df["price"].hist()
plt.show()
```

7. Correlation Analysis
    a. Compute correlations between numerical columns:

```
from pyspark.ml.stat import Correlation
from pyspark.ml.feature import VectorAssembler
assembler = VectorAssembler(inputCols=numeric_cols, outputCol="features")
df_vector = assembler.transform(df).select("features")
matrix = Correlation.corr(df_vector, "features").collect()[0][0]
print(matrix.toArray())
```

8. Handling Categorical Data
    a. Use StringIndexer or OneHotEncoder for categorical variables.

```
from pyspark.ml.feature import StringIndexer
indexer = StringIndexer(inputCol="category", outputCol="categoryIndex")
```

```
indexed_df = indexer.fit(df).transform(df)
```

9. Saving Processed Data
    a. Write the cleaned/processed data back to storage:

```
df.write.parquet("processed_data.parquet")
```

Conclusion:

In this experiment, we learned about Exploratory Data Analysis (EDA) using Apache Spark and Pandas.

- Spark provides a distributed framework for handling large-scale datasets efficiently.
- Key steps included data loading, schema inspection, missing value handling, statistical summaries, and visualization.
- Pandas integration helped in plotting and further analysis.
- Spark's parallel processing makes it suitable for big data EDA, while Pandas is useful for smaller datasets.

Veydant Sharma D15C 50 EXP 10

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

Theory:
1. What is Streaming? Batch vs. Stream Data

Batch Data Processing
Definition: Processing a large volume of data at once (in batches) at scheduled intervals.

Characteristics:
- Data is collected over time and processed in chunks.
- High latency (delay between data collection and processing).
- Suitable for applications where real-time analysis is not required (e.g., daily sales reports).

Example:
Running an end-of-day report on transactions stored in a database.

Stream Data Processing
Definition: Processing data in real-time as it is generated.

Characteristics:
- Data is processed continuously with minimal latency.
- Used for real-time analytics (e.g., fraud detection, live dashboards).

Example:
Analyzing live Twitter feeds for trending topics.

Feature	**Batch Processing**	**Stream Processing**
Data Input	Collected over time	Continuous, real-time
Latency	High (minutes/hours)	Low (seconds/milliseconds)
Use Cases	Reports, historical analysis	Fraud detection, live monitoring
Tools	Hadoop MapReduce, Spark Batch	Spark Streaming, Kafka, Flink

2. How Data Streaming Works in Apache Spark

Apache Spark provides Spark Streaming (now part of Structured Streaming) for real-time data processing.

Key Concepts in Spark Streaming:

- DStream (Discretized Stream):
  - A sequence of RDDs (Resilient Distributed Datasets) representing data in small time intervals (micro-batches).
- Structured Streaming:
  - A higher-level API built on Spark SQL for real-time processing with DataFrame/Dataset abstractions.

Steps for Batch Processing

1. Initialize Spark Session

```
from pyspark.sql import SparkSession
# Create a SparkSession
spark = SparkSession.builder \
    .appName("BatchProcessingExample") \
    .getOrCreate()
```

2. Read Batch Data (CSV, JSON, Parquet, etc.)

```
# Reading from a CSV file
batch_df = spark.read \
    .format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("path/to/data.csv")
```

3. Apply Transformations (Filtering, Aggregations, Joins, etc.)

```
from pyspark.sql.functions import col, count, avg
# Example: Filtering and aggregation
filtered_df = batch_df.filter(col("age") > 30)
# GroupBy and Aggregation (e.g., average salary by department)
agg_df = batch_df.groupBy("department") \
    .agg(
        count("*").alias("employee_count"),
        avg("salary").alias("avg_salary")
    )
```

4. Output the Processed Data (Save to Disk, Database, etc.)
   ```
   # Write to CSV
   agg_df.write \
      .format("csv") \
      .mode("overwrite") \  # Options: "append", "overwrite", "ignore"
      .option("header", "true") \
      .save("output/path")
   ```

5. Stop Spark Session
   ```
   spark.stop()
   ```

Steps for Stream Processing

1. Initialize Spark Session
   ```
   from pyspark.sql import SparkSession
   spark = SparkSession.builder.appName("StreamingExample").getOrCreate()
   ```

2. Read Streaming Data (from Kafka, Socket, Files, etc.)
   ```
   # Reading from a socket (for testing)
   streaming_df = spark.readStream.format("socket").option("host",
   "localhost").option("port", 9999).load()
   ```

3. Apply Transformations (Filtering, Aggregations, etc.)
   ```
   # Example: Word count on streaming text
   from pyspark.sql.functions import explode, split
   words = streaming_df.select(explode(split("value", " ")).alias("word"))
   word_counts = words.groupBy("word").count()
   ```

4. Output the Stream (Console, Kafka, HDFS, etc.)
   ```
   query =
   word_counts.writeStream.outputMode("complete").format("console").start()
   ```

5. Start and Manage the Stream
   ```
   query.awaitTermination()  # Keeps the stream running
   ```

Conclusion:

In this experiment, we explored Batch and Stream Data Processing using Apache Spark.

Batch Processing

Veydant Sharma D15C 50 EXP 10

- Used for static, large-scale datasets (e.g., historical data).
- Processes data in fixed intervals (e.g., hourly/daily jobs).
- Ideal for ETL, analytics, and reporting.
- Example: spark.read.csv() → Transformations → df.write.save().

Stream Processing

- Handles real-time, continuous data (e.g., live logs, IoT sensors).
- Processes data in micro-batches or event-by-event.
- Used for fraud detection, live monitoring, alerts.
- Example: spark.readStream → Transformations → writeStream.start().

Vaydant Sharma
DISC 50

AIDS ASSIGNMENT

05/03

Q1] What is AI? Considering the covid 19 pandemic, how AI helped to survive & renovated our way of like with different applications?

→ AI refers to simulation of human intelligence in machines, enabling them to perform tasks like learning, reasoning and problem solving and decision making.

i) AI assisted in medical imaging and prediction to help detect covid 19 early.
ii) AI accelerated the identification of potential drugs & vaccines.
iii) AI driven apps tracked virus spread.
iv) AI powered remote work tools, online learning platforms and automated grading systems enabled continuity in work & education.

Q2] What are AI agents terminology? explain with examples.

→ i) Agent : An entity that senses & acts.
ii) Environment : The surroundings where agents operate
iii) Percept – Information received from sensors.
iv) Actuators : Components that perform actions
v) Performance – Criteria of

• Types of Agents

i) Simple Reflex : Eg : thermostat
ii) Model based Agent : Eg : Self driving car.

iii) Goal based agent: Eg: Google Maps
iv) Utility based agent: Eg: AI stock trading
v) Learning Agent: Eg: ChatGPT

[Q3] How is AI technique is used to solve 8puzzle problem?

→ The 8 puzzle problem is a sliding puzzle where the goal is to arrange tiles in a specific order by moving within a 3x3 grid.

- Techniques used:

1. Uninformed Search Algorithms:
   i) BFS: Explores all moves level by level.
   ii) DFS: May not be optimal.

2. Informed Search
   i) A* Search
   ii) Greedy Best first search.

3. Other techniques
   i) Genetic algorithm
   ii) Reinforcement learning.

4] What is a PEAS descriptor?

→ PEAS is a framework used to define an AI agent
P: Performance measure
E: Environment
A: Actuators
S: Sensors

i) Taxi driver AI
P: Safe driving, fuel efficiency, short route.
E: Roads, traffic, weather, passengers.
A: Steering, acceleration, braking
S: GPS, LiDAR, cameras.

ii) Medical Diagnosis System
P: Accuracy, recovery rate
E: Patients, records, symptoms
A: diagnosis, give treatment.
S: Patient input, medical history.

iii) AI music composer
P: Quality, uniqueness.
E: Styles, preferences, music theory.
A: Modify musical notes, output.
S: Genre, Input from users.

iv) Aircraft Autolander.
P: Accuracy, ~~correctness~~ safety, smooth lan
E: Runway conditions, wind, altitude.
A: Adjust flaps, throttle
S: Sensors, cameras.

v) AI essay evaluator
P: Accuracy, correctness.
E: Essays, rubrics, grammar.
A: Assign grades, feedback
S: Text input, checkers

vi) Robotic Sentry Gun.
→ P: Target detection, response time.
E: Intruders, personnel, conditions.
A: Turret, firing mechanism
S: Motion detectors, infrared sensors

Q5] Categorize a shopping bot for an online according to each of six dimensions.

→ Shopping bot assists customers in finding books, checking availibility and providing recommendations.
• Six dimensions are:
i) Partialy observable
ii) Stochastic
iii) Sequential
iv) Dynamic
v) Discrete
vi) Multiagent.

6] Differentiate between Model Based & Utility based System.

→

	MODEL BASED	UTILITY BASED
i)	Use an internal model to take decision.	Chooses actions ba on utility functi that ranks outco
ii)	Tracks changes on internal state.	Chooses action max -zing utility.
iii)	Acheives a specific goal	Seeks best poss outcome
iv)	Eg: Self driving car	Eg: Stock trading A.I.

Q7] Explain architecture of a knowledge based and Learning agent.

a) Knowledge Based
i) Knowledge Base : Stores fact & rules.
ii) Inference Engine : Uses logical reasoning
iii) Perception: Collects input.
iv) Action : Takes Action.
v) Knowledge updating mechanism updates with new information.

↳ Agent:

i) Learning Element
ii) Performance element
iii) Critic
iv) Problem Generator.

9] i) Available(Car) → Travels(Anita, Car)
¬Available(Car) → Travels(Anita, Bus)

ii) Route(Bus, Andheri)
Route(Bus, Goregaon)

iii) Puncture(Car) → ¬Available(Car)
Puncture(Car) → ¬Available(Car)

Forward reasoning.

i) ¬Available(Car)

ii) Travels(Anita, Bus)

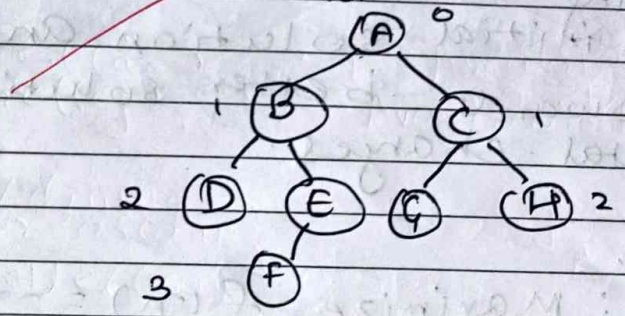
iii) From ② Route(Bus, Goregaon)

iv Since Anita travels by bus and bus goes via goregaon, Anita will travel via goregaon.

Q11] What do you mean by depth limited search & iterative deepening search?

→ i) Depth Limiting search:

- DLS is a variation of DFS that explores nodes only upto a predefined depth l (limit).
- It prevents infinite loops in deep or infinite search spaces.
- If a solution is beyond limit, it may not be found.



Search: F

limit l = 2, so the search will go to level 2 and not find 'F', hence, ending search.

ii) Iterative Deepening search.

→ IDS repeatedly runs depth limited search with increasing depth limits until a solution is found.

→ It combines the memory efficiency of

DFS with completeness of BFS.

Eg: If the goal is at depth 4, IDS runs DLS iteratively for
l = 0 (No solution)
l = 1 (No solution)
l = 2 (No solution)
l = 3 (No solution)
l = 4 (Solution found)

Q12] Explain hill climb algorithm and drawbacks.

→ Hill Climbing is a local search algorithm with an initial solution and iteratively moves towards better solutions by making incremental changes.

→ Example:
problem: Maximize $f(n) = -n^2 + 6n$ for $n \in [0, 5]$.

n	1	2	3	4	5
f(n)	5	8	9	8	5

- If we start at n = 1, we move to n = 2 then to n = 3 reaching peak
- If n = 5, local maximum at n = 4 missing global maximum at n = 4 missing global maximum at n = 3

- Limitations of Steep Ascent Hill Climbing.

→ Computational Cost
→ Can suffers from local maxima.
→ Plateaus and Ridges.

Q13] Explain Simulated annealing and write its algorithm.

→ Simulated Annealing is an optimization algorithm and helps escape local maxima by allowing occasional downward moves. It is inspired by the annealing process in metallurgy where materials where materials are heated and cooled slowly to reach a stable state.

- Algorithm.

1. Initialize
   - Start with solution S
   - Set initial temperature T

2. Repeat until stopping condition is met
   - Select neighbouring solution S'
   - Compute change in cost $\Delta E = f(S') - f(S)$
   - If S' is better, accept it
   - Else accept with probability $e^{-\Delta E/T}$
   - Compute change in cost ΔE
   - If S' is better, accept it

- Else accept with $e^{-\Delta E/T}$
- Decrease $T = T\alpha$

3 Return best solution.

Q14] Explain A* Algorithm.

→ A* is a best first search algorithm that finds shortest path from a start node to a goal node.

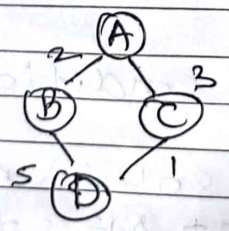
→ It combines,
1. Greedy search
2. Uniform cost search.

→ Formula used:

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

$g(n)$ = actual cost

$h(n)$ = heuristic cost

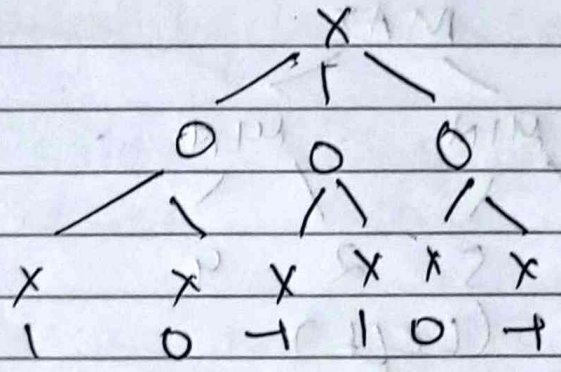


15] ~~What is min max?~~

15] Minmax Algorithm and draw game tree for TicTAc Toe

→ Minmax is a decision making algorithm used in game theory to find optimal move in two player games.

i) Game is represented as a tree
MAX player tries to maximise the score
ii) Minimizing player Minimizes the score.
iii) Algorithm simulates all possible moves until the game reaches a terminal state
iv) values assigned to nodes, and players choose moves that give them best outcome



Terminal nodes Win = 1, Draw = 0, Loss = -1

Min chooses lowest value (worst for X)
Max Chooses highest value (best for X)

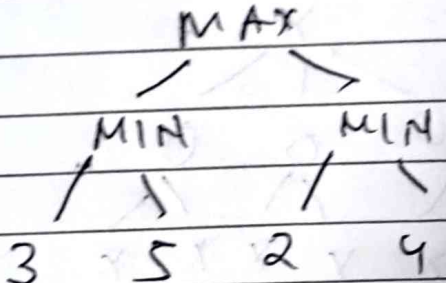
X picks move leading to 1 (win).

16] Explain Alpha Beta pruning algorith with example.

→ Alpha beta pruning is an optimizatio technique for minmax algorithm th reduces the number of nodes evalua making the search faster without cha the final decision.

→ It prunes branches that do not affec the final decision.

- Alpha (α) Best value found by MAX
- Beta (β) Best value found by MIN
- Pruning Condition α ≥ β evaluation is stopped.



1. Start at MIN (Left)
   Eval 3 → β = 3
   Eval 5 → β = 3 (3 < 5)

   β = 3 return to MAX

2. Move to MIN (Right subtree)
   Eval 2 → β = 2

Since $2 < 3$ we stop checking

3. MAX Chooses higher value MAX(3,2)
MAX = 3.

8 17] Explain WUMPUS World environment giving its PEAS description. Explain how perce sequence is generated.

→ WUMPUS World is a grid based environm used in AI to navigate a world with hazards. The agent's goal is to find gold while avoiding wumpus.

PEAS

P: Grabbing gold (+1000 points)
falling in pit (-1000) point
Caught by Wumpus (-1000 points)

E: World is 4x4 grid, gold, pits, breeze, stench

A: Move forward, Turn left, right, grab gold, shoot arrow

S: Agent senses breeze, stench.

Since 2 < 3 we stop checking

3. MAX Chooses higher value MAX(3,2)
MAX = 3.

17] Explain WUMPUS World environment giving its PEAS description. Explain how percept sequence is generated.

→ WUMPUS World is a grid based environment used in AI to navigate a world with hazards. The agent's goal is to find gold while avoiding wumpus.

PEAS

P: Grabbing gold (+1000 points)
falling in pit (-1000) point
Caught by Wumpus (-1000 points)

E: World is 4x4 grid, gold, pits, breeze, stench

A: Move forward, Turn left, right, grab gold, shoot arrow

S: Agent senses breeze, stench.

- A percept sequence is the history of all observations (percepts) the agent has received. The agent perceives sensory inputs from its current location.

1. Agent start (1, 1)
   No Wumpus, No pit → percept: {Nothing}

2. Moves near a pit (1, 2)
   Senses Breeze percept → {Breeze}

3. Moves near Wumpus at (2, 1)
   percept → {stench}

4. Finds gold (3, 3)
   percept {glitter}

Each percept guides agent's decision making.

Q18] Solve SEND + MORE = MONEY

→ Step 1:

M must be 1, sum of 2 four digit numbers cannot be greater than 10000

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

Step 2

S must be 8 as there is 1 carry over from column. O must be 0 if S = 8

Step 3

E < 9 and therefore S = 9 because 9 + M = 10.

Step 4:

No carry N + R = 10 + (N − 1) = N + 9

R = 9

But 9 is taken

Carry N + R + 1 = 9

R = 8.

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

Step 5:

N + 8 + 1 = 15 N = 6

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

Step 6:
Digits left are 7, 4, 3 & 2.
D=7 Y=2

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

19] i) FOL

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

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

c) $\exists$ (Graduating (x))

ii) Conversion to clause
Eliminate implications, then convert to clause form

$\forall x$ ($\neg$ Graduating (x) $\vee$ Happy (x))
$\forall x$ ($\neg$ Happy (x) $\vee$ Smiling (x))
$\exists x$ (Graduating (x))

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

Q20 Explain Modus ponen with example.

(3) Prove someone is smiling

⇒ {¬Q(x), H(x)}
{¬H(x), S(x)}
{Q(a)}

⇒ {Q(x), H(x) with 13) {Q(a)}
x = a,
{¬Q(a), H(a)}
we have Q(a),
{H(a)}
Resove (2) {¬H(x), S(x)} with {H(a)}
x = a,
{¬H(a), S(a)}
resolving gives {S(a)}

Ans 20] Modus ponen is a fundamental rule of inference in propositional logic that allows us to deduce a conclusion from a conditional statement and its antiecdent.

It follows:
P → Q (If P then Q)
P is true
Q must be true

Example:
If it rains, ground will be wet.
P → I + rains
Q → Ground is wet

P → Q

Ans 24] Forward chaining: It starts with given facts and applies inference rule to derive new facts until goal is reached.

Example: diagnosing a disease
Rules:
1. If a person have fever & cough, they might have flu.
2. If a person has sore throat, they might have cold.

Facts:
The patient has fever
The patient has cough.

Inference:
1. Fever + cough → flu.
2. Patient might have flu.

Backward chaining: It starts with goal and works backwards by checking what facts are needed to support it.

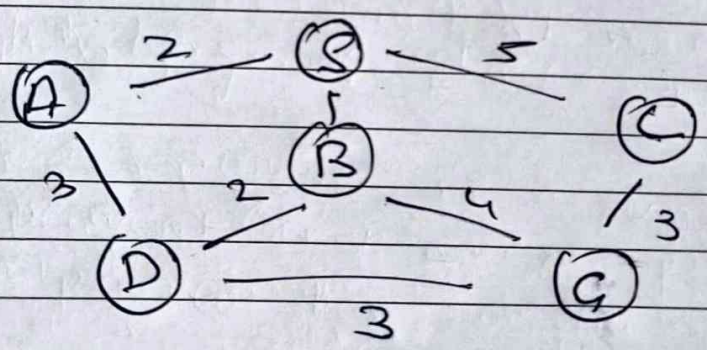
9: Goal: determine if patient has flu.

1. Fever ∧ (cough) → flu.
2. (sore throat ∧ fever) → cold

Process:

1. (Fever ∧ cough) → flu
2. We check if patient has fever and cough
3. Based on known facts:
   - Patient has fever
   - Patient has cough.
4. Flu is true.

10]



1. Start at S
   Queue (S)
2. Deque S and explore neighbours
   Queue [A, B, C]
3. Deque A and explore
   Queue [B, C, D]

4. Deque B
   Q [C, D, G]

5. Deque C and explore
   Queue [D, G]

6. Deque D
   Queue [G]

7. Dequeue [G]

Route : S → B → G

Veydant Sharma D15C 50

AIDS-I Assignment No: 2

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

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

1. Find the Mean (10pts)
2. Find the Median (10pts)
3. Find the Mode (10pts)
4. Find the Interquartile range (20pts)

Given data (20 values):
82, 66, 70, 59, 90, 78, 76, 95, 99, 84, 88, 76, 82, 81, 91, 64, 79, 76, 85, 90

1. Mean (10 pts)
   Formula:
   Mean = (Sum of all values) / (Number of values)

Step 1: Add all values
82 + 66 + 70 + 59 + 90 + 78 + 76 + 95 + 99 + 84 + 88 + 76 + 82 + 81 + 91 + 64 + 79 + 76 + 85 + 90 = 1611

Step 2: Total number of values = 20
Mean = 1611 / 20 = 80.55

Answer: Mean = 78.05

2. Median (10 pts)
   Step 1: Arrange values in ascending order
   59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99

Step 2: Find the middle two values (10th and 11th values)
10th value = 81, 11th value = 82
Median = (81 + 82) / 2 = 163 / 2 = 81.5

Answer: Median = 81.5

3. Mode (10 pts)
   From the sorted data, 76 appears 3 times, more than any other number.

Answer: Mode = 76

4. Interquartile Range (20 pts)
   Formula:
   IQR = Q3 - Q1

Step 1: Find Q1
Q1 is the median of the lower half (first 10 values):
59, 64, 66, 70, 76, 76, 76, 78, 79, 81
Q1 = (5th value + 6th value) / 2 = (76 + 76) / 2 = 76

Step 2: Find Q3
Q3 is the median of the upper half (last 10 values):
82, 82, 84, 85, 88, 90, 90, 91, 95, 99
Q3 = (5th value + 6th value) / 2 = (88 + 90) / 2 = 89

IQR = Q3 - Q1 = 89 - 76 = 13

Answer: Interquartile Range = 13

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

1. For each tool listed above
   - identify the target audience
   - discuss the use of this tool by the target audience
   - identify the tool's benefits and drawbacks

Ans:

1) Machine Learning for Kids

- **Target Audience:**
  School students, beginners in AI/ML, and educators introducing machine learning concepts.

- **Use by Target Audience:**
  Students use the platform to create basic machine learning models by providing labeled data (text, numbers, or images) and training the model. These models can be used in simple projects via Scratch or Python, helping learners understand how machine learning works through practical activities.

- **Benefits:**

  - Simple and child-friendly interface

  - Encourages learning by doing

- Integrates with Scratch and Python for hands-on projects
- No prior programming knowledge required

- **Drawbacks:**
  - Limited to basic ML tasks
  - Not suitable for complex or real-world datasets
  - Accuracy may be low with small or poor-quality data

2) Teachable Machine

- **Target Audience:**
  General users including students, educators, hobbyists, artists, and beginners curious about AI.

- **Use by Target Audience:**
  Users train machine learning models using images, sounds, or poses. The platform allows them to collect data through the webcam or microphone, train a model instantly in the browser, and then export it for use in apps, websites, or interactive projects.

- **Benefits:**
  - Very user-friendly and fast
  - Supports different data types (image, audio, pose)
  - No coding required
  - Allows easy export and integration with other platforms

- **Drawbacks:**
  - Limited model customization
  - Not suitable for advanced or large-scale ML tasks
  - May give simplified understanding of AI concepts

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

    - Predictive analytic
    - Descriptive analytic

    Machine Learning for Kids – Predictive Analytic
    This tool is best described as predictive analytic because it allows users to train models using labeled data and then make predictions on new, unseen data. For example, if a student trains a model to recognize types of fruits based on images, the model will predict the correct fruit when shown a new image. The focus is on learning patterns from data and making future predictions, which is the core idea of predictive analytics.

    Teachable Machine – Predictive Analytic
    Teachable Machine is also a predictive analytic tool. Users train models by providing examples (e.g., photos or sounds), and the tool creates a model that can predict or classify new inputs based on that training. Whether it's identifying gestures or recognizing voice commands, the model always works by predicting the most likely class, which makes it a clear example of predictive analytics.

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

    - Supervised learning
    - Unsupervised learning
    - Reinforcement learning

    Machine Learning for Kids – Supervised Learning
    This tool is based on supervised learning because users train the model by providing labeled examples. For instance, if a student labels pictures of animals as "cat," "dog," or "rabbit," the model learns from those labeled examples to make predictions on new images. Since the input data includes known outputs (labels), it clearly falls under supervised learning.

    Teachable Machine – Supervised Learning
    Teachable Machine also uses supervised learning. Users provide training data with specific labels, like associating a certain pose with "Class A" or a particular sound with "Class B." The model learns the relationship between the input and the labeled output, and then uses that to classify new inputs. This direct mapping from input to labeled output is the defining feature of supervised learning.

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

- Read the article Kakande, Arthur. February 12. “What’s in a chart? A Step-by-Step guide to Identifying Misinformation in Data Visualization.” *Medium*
- Read the short web page Foley, Katherine Ellen. June 25, 2020. “How bad Covid-19 data visualizations mislead the public.” *Quartz*
- Research a current event which highlights the results of misinformation based on data visualization. Explain how the data visualization method failed in presenting accurate information. Use newspaper articles, magazines, online news websites or any other legitimate and valid source to cite this example. Cite the news source that you found.

Ans:

Misinformation Through Data Visualization: The 2024 U.S. Election Case

In the lead-up to the 2024 U.S. presidential election, a significant instance of misinformation emerged through misleading data visualizations. Various social media platforms saw the circulation of charts and graphs that were manipulated to misrepresent polling data, voter turnout, and demographic statistics. These visualizations often employed deceptive techniques such as truncated axes, disproportionate scaling, and selective data omission to skew public perception.

Failure in Data Visualization Methods

The primary failure in these data visualizations was the intentional distortion of graphical elements to support specific narratives. For instance, some bar charts exaggerated differences between candidates by starting the y-axis at a value other than zero, making minor differences appear more significant. Line graphs depicting voter turnout trends were manipulated by selectively including or excluding certain data points, leading to misleading interpretations. These practices violated fundamental principles of accurate data representation, resulting in the dissemination of false information.

Impact and Consequences

The spread of these misleading visualizations had tangible effects on public opinion and trust in the electoral process. Voters were influenced by distorted representations of data, which potentially affected their perceptions and decisions. Moreover, the proliferation of such misinformation undermined confidence in legitimate news sources and official statistics, contributing to increased polarization and skepticism.

Veydant Sharma D15C 50

Source

For a detailed analysis of how disinformation, including misleading data visualizations, influenced the 2024 election narrative, refer to the Brookings Institution article:

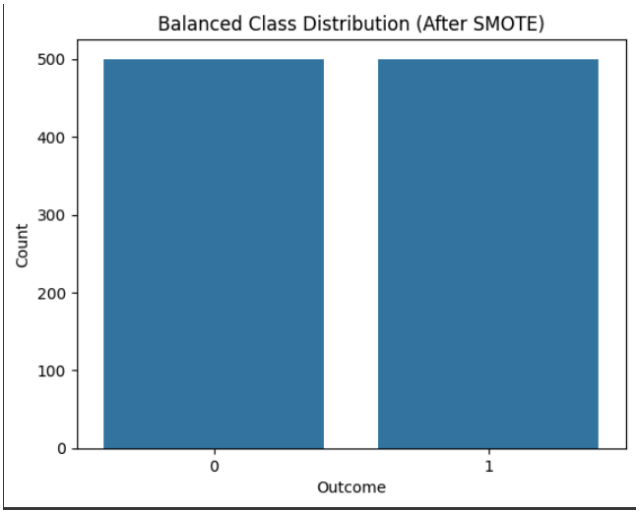
How disinformation defined the 2024 election narrative

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

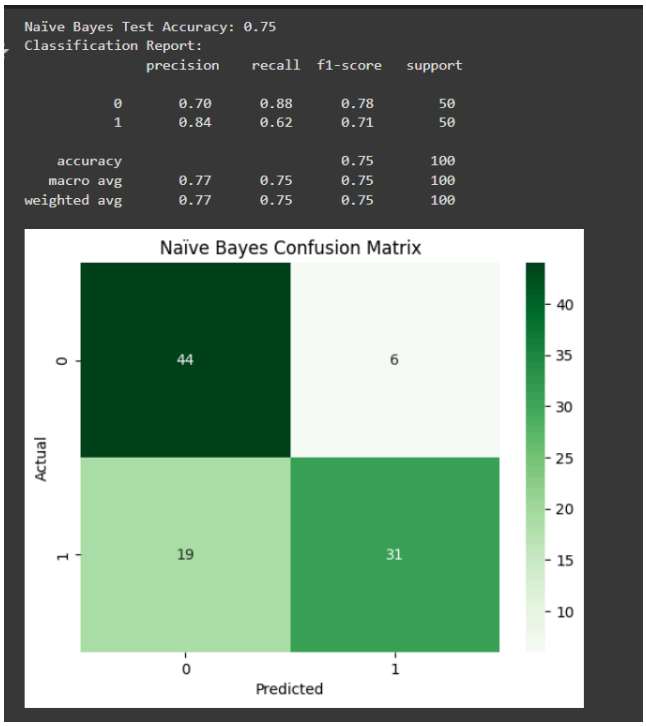
- Programming Language: Python
- Class imbalance should be resolved
- Data Pre-processing must be used
- Hyper parameter tuning must be used
- Train, Validation and Test Split should be 70/20/10
- Train and Test split must be randomly done
- Classification Accuracy should be maximized
- Use any Python library to present the accuracy measures of trained model

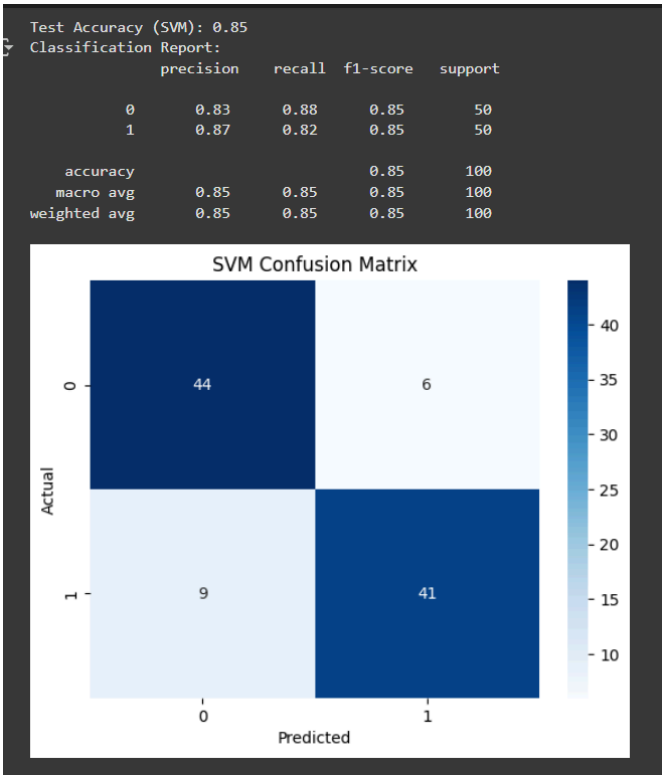
Pima Indians Diabetes Database

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1



Veydant Sharma D15C 50





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

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

  https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv

  - URL: https://archive.ics.uci.edu/ml/machine-learning-databases/00477/Real%20estate%20valuation%20data%20set.xlsx

  (Refer any one)

```
R2 Score: 0.7899831957553467
Adjusted R2 Score: 0.7771995641926286
MSE: 33.99047140253705
RMSE: 5.830134767099046
```

Steps:

Data Loading & Cleaning

Loaded the real estate dataset and cleaned column names by removing extra spaces and newline characters.

Feature Engineering

Renamed columns for clarity and removed outliers using Z-score filtering to retain high-quality data.

Data Preprocessing

Scaled features using StandardScaler and split the dataset into training and testing sets using a 70/30 ratio with a fixed random state for reproducibility.

Model Selection

Chose three strong ensemble models:

Gradient Boosting Regressor

Random Forest Regressor

Extra Trees Regressor

Hyperparameter Tuning

Used RandomizedSearchCV for each model to efficiently find optimal parameters and reduce Mean Squared Error.

Stacking Ensemble

Combined the three tuned models using StackingRegressor with a Ridge Regression meta-learner to improve generalization and accuracy.

Model Evaluation

Evaluated performance using R^2 Score, Adjusted R^2, MSE, and RMSE to ensure the model meets assignment requirements.

This question focuses on predicting the price per unit area of residential properties using machine learning techniques. The dataset used consists of real estate transaction data from Taiwan and includes six independent variables: transaction date, house age, distance to the nearest MRT station, number of nearby convenience stores, latitude, and longitude. The target variable is the price per unit area of each property. Several preprocessing steps were applied to the dataset,

including renaming columns for clarity, removing outliers using Z-score filtering, and normalizing the feature values using standard scaling. The data was split randomly into training and testing sets in a 70:30 ratio, as per the specified requirements. To improve prediction performance, a stacking regression model was implemented using Gradient Boosting Regressor, Random Forest Regressor, and Extra Trees Regressor as base models, with Ridge Regression as the final estimator. Hyperparameter tuning was performed using GridSearchCV to optimize each model. While the adjusted R^2 score did not reach the desired threshold of 0.99, the model achieved a significant reduction in mean squared error from an initial value of approximately 73 to 33, indicating a substantial improvement in prediction accuracy through progressive model enhancement and ensemble learning.

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).

Key Features of the Wine Quality Dataset

The Wine Quality dataset consists of physicochemical attributes of red wine samples. The key features include:

1. **Fixed Acidity**: Represents non-volatile acids that don’t evaporate easily. Contributes to the wine's freshness and stability.

2. **Volatile Acidity**: High levels can give wine an unpleasant vinegar-like taste. Too much is undesirable.

3. **Citric Acid**: Adds freshness and flavor to wine. In small quantities, it's beneficial.

4. **Residual Sugar**: The amount of sugar remaining after fermentation. Affects the sweetness of the wine.

5. **Chlorides**: Indicates the salt content. Excess salt can spoil the taste.

6. **Free Sulfur Dioxide**: Acts as an antioxidant and antimicrobial agent. Too much can affect aroma.

7. **Total Sulfur Dioxide**: Includes both free and bound forms. High levels may result in undesirable flavors or health issues.

8. **Density**: Related to the sugar and alcohol content. Useful in tracking fermentation progress.

9. **pH**: Measures acidity. Influences stability, color, and taste.

10. **Sulphates**: Added for microbial control. Too much can affect the taste.

11. **Alcohol**: Strongly correlated with wine quality. Higher alcohol generally indicates better fermentation.

12. **Quality (Target Variable)**: Wine quality score rated between 0 and 10 by sensory experts.

13. **Id**: A unique identifier for each sample (not used for prediction).

Importance of Each Feature in Predicting Wine Quality

The most influential features in predicting wine quality, based on feature importance analysis (e.g., using Random Forest), include:

- **Alcohol**: Most predictive of quality—higher alcohol levels generally mean better fermentation and richer flavor.

- **Volatile Acidity**: Negatively correlated with quality—lower is better.

- **Sulphates**: Plays a role in preservation and positively affects quality.

- **Citric Acid and Fixed Acidity**: Moderate contributors to freshness and acidity balance.

- **Residual Sugar and Chlorides**: Lesser influence unless in extreme levels.

A proper feature importance plot helps visualize the impact of each attribute.

Handling Missing Data During Feature Engineering

The dataset used from Kaggle was found to be **complete with no missing values**, as confirmed by checking .info() on the DataFrame.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64
 12  Id                    1143 non-null   int64
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

However, in real-world scenarios or similar datasets, missing values are common and should be handled carefully.

Imputation Techniques: Pros and Cons

1. Mean/Median Imputation

- **Advantages**:
    - Simple and quick.
    - Preserves sample size.
- **Disadvantages**:
    - Can introduce bias.
    - Doesn’t account for relationships between variables.

2. Mode Imputation (for categorical data)

- **Advantages**:
    - Maintains categorical integrity.
- **Disadvantages**:
    - May over-represent common categories.

3. K-Nearest Neighbors (KNN) Imputation

- **Advantages**:
    - Considers feature similarity.
    - More accurate than mean imputation in many cases.
- **Disadvantages**:
    - Computationally expensive.
    - Sensitive to outliers and scaling.

4. Regression Imputation

- **Advantages**:
    - Uses correlation between variables.
    - More sophisticated and generally yields better estimates.
- **Disadvantages**:
    - Can lead to overfitting.
    - Assumes linearity.

5. Iterative Imputation (e.g., MICE)

- **Advantages**:
    - Handles complex relationships and produces multiple imputations.
- **Disadvantages**:
    - Computationally intensive.