EXPERIMENT 1

Load Data in Pandas

```
import pandas as pd

df = pd.read_csv("/content/Top_10000_Movies.csv", on_bad_lines="skip")
df.head()
```

	id	original_language	original_title	popularity	release_date	vote_average	vote_count	genre	overview	revenue	runtime	tagline
0	580489.0	en	Venom: Let There Be Carnage	5401.308	2021-09-30	6.8	1736.0	['Science Fiction', 'Action', 'Adventure']	After finding a host body in investigative rep...	424000000.0	97.0	NaN
1	524434.0	en	Eternals	3365.535	2021-11-03	7.1	622.0	['Action', 'Adventure', 'Science Fiction', 'Fa...	The Eternals are a team of ancient aliens who ...	165000000.0	157.0	In the beginning...
2	438631.0	en	Dune	2911.423	2021-09-15	8.0	3632.0	['Action', 'Adventure', 'Science Fiction']	Paul Atreides, a brilliant and gifted young ma...	331116356.0	155.0	Beyond fear, destiny awaits.
3	796499.0	en	Army of Thieves	2552.437	2021-10-27	6.9	555.0	['Action', 'Crime', 'Thriller']	A mysterious woman recruits bank teller Ludwig...	0.0	127.0	Before Vegas, one locksmith became a legend.
4	550988.0	en	Free Guy	1850.470	2021-08-11	7.8	3493.0	['Comedy', 'Action', 'Adventure', 'Science Fic...	A bank teller called Guy realizes he is a back...	331096766.0	115.0	Life's too short to be a background character.

Description of the Dataset

```
df.info()

df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10014 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 10002 non-null  float64
 1   original_language  10002 non-null  object
 2   original_title     10001 non-null  object
 3   popularity         10000 non-null  float64
 4   release_date       9962 non-null   object
 5   vote_average       10000 non-null  float64
 6   vote_count         10000 non-null  float64
 7   genre              10000 non-null  object
 8   overview           9900 non-null   object
 9   revenue            9998 non-null   float64
 10  runtime            9989 non-null   float64
 11  tagline            7079 non-null   object
dtypes: float64(6), object(6)
memory usage: 1.2+ MB
```

	id	popularity	vote_average	vote_count	revenue	runtime
count	10002.000000	10000.000000	10000.00000	10000.000000	9.998000e+03	9989.000000
mean	250003.082683	34.516871	6.29875	1315.084900	5.737536e+07	98.792772
std	261732.329571	100.693958	1.43426	2501.899103	1.480897e+08	28.771525
min	0.000000	6.269000	0.00000	0.000000	0.000000e+00	0.000000
25%	11864.500000	11.908000	5.90000	118.000000	0.000000e+00	89.000000

Drop columns that aren’t useful.

Drop Columns

```
df = df.drop(["popularity", "runtime"], axis=1)

df.head()
```

	id	original_language	original_title	release_date	vote_average	vote_count	genre	overview	revenue	tagline
0	580489.0	en	Venom: Let There Be Carnage	2021-09-30	6.8	1736.0	['Science Fiction', 'Action', 'Adventure']	After finding a host body in investigative rep...	424000000.0	NaN
1	524434.0	en	Eternals	2021-11-03	7.1	622.0	['Action', 'Adventure', 'Science Fiction', 'Fa...	The Eternals are a team of ancient aliens who ...	165000000.0	In the beginning...
2	438631.0	en	Dune	2021-09-15	8.0	3632.0	['Action', 'Adventure', 'Science Fiction']	Paul Atreides, a brilliant and gifted young ma...	331116356.0	Beyond fear, destiny awaits.
3	796499.0	en	Army of Thieves	2021-10-27	6.9	555.0	['Action', 'Crime', 'Thriller']	A mysterious woman recruits bank teller Ludwig...	0.0	Before Vegas, one locksmith became a legend.
4	550988.0	en	Free Guy	2021-08-11	7.8	3493.0	['Comedy', 'Action', 'Adventure', 'Science Fic...	A bank teller called Guy realizes he is a back...	331096766.0	Life's too short to be a background character.

Standardization and Normalization of columns

Standardization

```
[16] from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df[['vote_average_std', 'vote_count_std', 'revenue_std']] = scaler.fit_transform(df[['vote_average', 'vote_count', 'revenue']])

df.head()
```

	id	original_title	release_date	vote_average	vote_count	overview	revenue	tagline	original_language_be	original_language_bn	...	genre_['Western', 'Horror']	genre_['West 'Myst 'Thril 'Dr
0	580489.0	Venom: Let There Be Carnage	2021-09-30	6.8	1736.0	After finding a host body in investigative rep...	424000000.0	Based on a true story.	False	False	...	False	
1	524434.0	Eternals	2021-11-03	7.1	622.0	The Eternals are a team of ancient aliens who ...	165000000.0	In the beginning...	False	False	...	False	
						Paul Atreides, a		Beyond fear,					

Normalization

```
[17] from sklearn.preprocessing import MinMaxScaler

min_max_scaler = MinMaxScaler()
df[['vote_average_norm', 'vote_count_norm', 'revenue_norm']] = min_max_scaler.fit_transform(df[['vote_average', 'vote_count', 'revenue']])

df.head()
```

	id	original_title	release_date	vote_average	vote_count	overview	revenue	tagline	original_language_be	original_language_bn	...	genre_['Western', 'TV Movie']	genre_['West 'Thril
0	580489.0	Venom: Let There Be Carnage	2021-09-30	6.8	1736.0	After finding a host body in investigative rep...	424000000.0	Based on a true story.	False	False	...	False	
1	524434.0	Eternals	2021-11-03	7.1	622.0	The Eternals are a team of ancient aliens who ...	165000000.0	In the beginning...	False	False	...	False	
2	438631.0	Dune	2021-09-15	8.0	3632.0	Paul Atreides, a brilliant and gifted young ma...	331116356.0	Beyond fear, destiny awaits.	False	False	...	False	

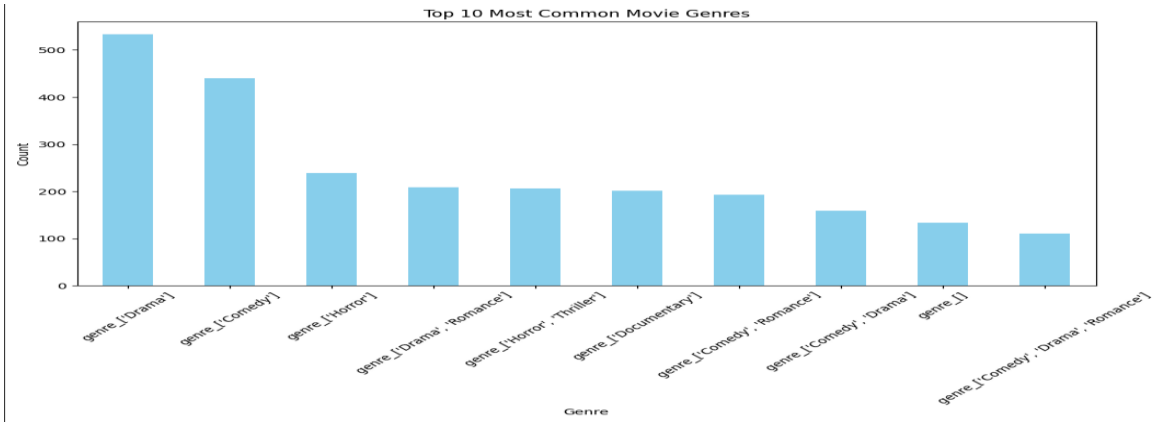
EXPERIMENT 2

1. Bar Graph

This graph provides an overview of genre distribution in the dataset. The dominance of Drama and Comedy aligns with industry trends.

Bar Graph

```
import matplotlib.pyplot as plt

genre_columns = [col for col in df.columns if col.startswith("genre_")]

genre_counts = df[genre_columns].sum().sort_values(ascending=False)

plt.figure(figsize=(12, 6))
genre_counts.head(10).plot(kind="bar", color="skyblue")

plt.title("Top 10 Most Common Movie Genres")
plt.xlabel("Genre")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```

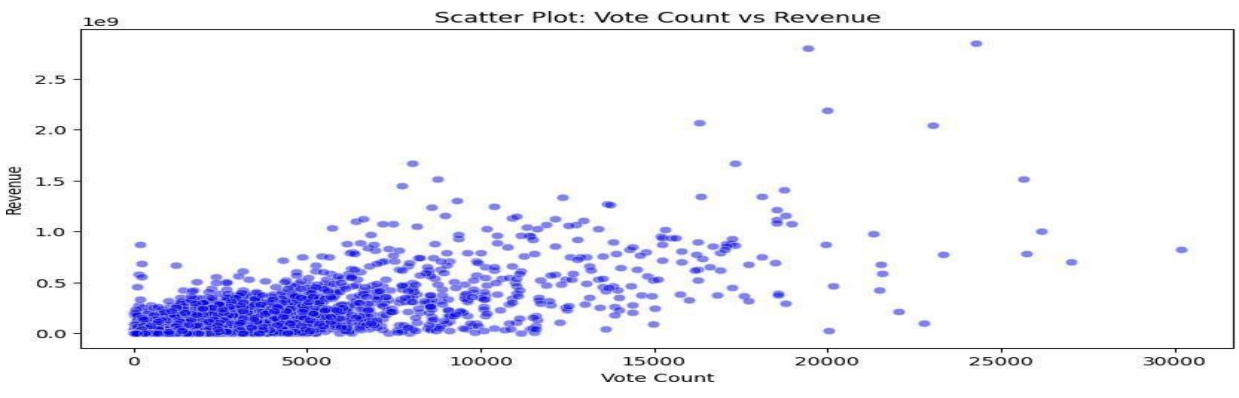


2. Scatter Plot

The scatter plot suggests that while higher vote counts often correspond to higher revenue, other factors likely play a significant role in determining a movie's financial success.

Scatter plot

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

plt.figure(figsize=(10, 5))
sns.scatterplot(x=df["vote_count"], y=df["revenue"], alpha=0.5, color="blue")

plt.title("Scatter Plot: Vote Count vs Revenue")
plt.xlabel("Vote Count")
plt.ylabel("Revenue")
plt.show()
```



Aim: Perform Data Modeling on the dataset

Output :

1. Partition of the dataset that is dividing into training and testing of data in **75-25** where 75% of the records are included in the training data and rest 25% are included in the test data.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from scipy import stats
```

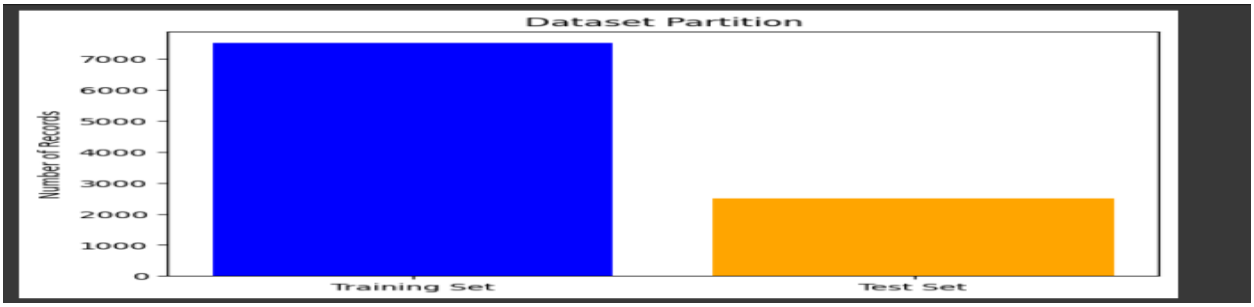
2. Visualization using a bar graph to confirm the proportions of the data split into training and

```
proportions = [len(train_data), len(test_data)]
labels = ['Training Set', 'Test Set']

plt.bar(labels, proportions, color=['blue', 'orange'])
plt.title('Dataset Partition')
plt.ylabel('Number of Records')
plt.show()
```

test sets and checking the records that are split up

```
Total records: 10014
Training set records: 7510
Test set records: 2504
```

Using a **two-sample Z-test**, we evaluated whether the data split introduced any bias in the




```
[11] train_mean = np.mean(train_data['popularity'])
     test_mean = np.mean(test_data['popularity'])
     train_std = np.std(train_data['popularity'], ddof=1)
     test_std = np.std(test_data['popularity'], ddof=1)

     z_score = (train_mean - test_mean) / np.sqrt((train_std**2/len(train_data)) + (test_std**2/len(test_data)))
     p_value = stats.norm.sf(abs(z_score)) * 2

     print(f"Z-Score: {z_score}")
     print(f"P-Value: {p_value}")
```

popularity distribution

3. Now this test is performed for the comparison of **Popularity** column in training and testing dataset. The Z-statistic was calculated based on the means, standard deviations, and sizes of both samples.

```
Z-Score: -1.59015224967257
P-Value: 0.11180049083548155
```

This validates that the partitioning process preserves the original distribution, ensuring that both sets are representative of the overall data.

Name:Himanshu Naik Roll no:63 Div:D15C

Exp 4

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

Theory and Output:

1. Loading dataset:

Data loading is the first step in data analysis. The dataset is stored in a CSV file and read using `pandas.read_csv()`.

The first few rows are displayed to understand the dataset structure

```
[1] import pandas as pd
    import scipy.stats as stats
```

```
df = pd.read_csv('/content/Employee.csv')
```

```
df.head()
```

	Education	JoiningYear	City	PaymentTier	Age	Gender	EverBenched	ExperienceInCurrentDomain	LeaveOrNot
0	Bachelors	2017	Bangalore	3	34	Male	No	0	0
1	Bachelors	2013	Pune	1	28	Female	No	3	1
2	Bachelors	2014	New Delhi	3	38	Female	No	2	0
3	Masters	2016	Bangalore	3	27	Male	No	5	1
4	Masters	2017	Pune	3	24	Male	Yes	2	1

Next steps: Generate code with df | View recommended plots | New interactive sheet

2. Pearson's Correlation Coefficient:

Pearson's Correlation Coefficient (denoted as **r**) measures the **linear** relationship between two continuous variables.
Values range from **-1 to +1**:

- **+1**: Perfect positive correlation
- **0**: No correlation
- **-1**: Perfect negative correlation

The formula for Pearson's Correlation Coefficient is:

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2}}$$

```
pearson_corr, pearson_p = stats.pearsonr(df['Age'], df['ExperienceInCurrentDomain'])

print(f"Pearson's Correlation Coefficient: {pearson_corr}")
print(f"P-value: {pearson_p}")
```

```
Pearson's Correlation Coefficient: -0.13464285083693067
P-value: 2.8637816441811323e-20
```

3. Spearman's Rank Correlation

- Spearman's Rank Correlation (denoted as ρ, rho) measures the monotonic relationship between two variables.
- It does not require normally distributed data.
- If ranks of two variables are related, it indicates correlation.
- The formula is:

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

```
[13] spearman_corr, spearman_p = stats.spearmanr(df['Age'], df['ExperienceInCurrentDomain'])

     print(f"Spearman's Rank Correlation Coefficient: {spearman_corr}")
     print(f"P-value: {spearman_p}")
```

```
Spearman's Rank Correlation Coefficient: -0.14172932292026683
P-value: 2.6218815420869774e-22
```

4. Kendall's Rank Correlation

Theory:

- Kendall's Tau (τ) measures the **ordinal association** between two variables.
- It counts **concordant** and **discordant** pairs:
  - **Concordant pairs**: If one variable increases, the other also increases.
  - **Discordant pairs**: One increases while the other decreases.
- The formula is:

$$\tau = \frac{(C - D)}{\frac{1}{2}n(n-1)}$$

```
[14] kendall_corr, kendall_p = stats.kendalltau(df['Age'], df['ExperienceInCurrentDomain'])

     print(f"Kendall's Rank Correlation Coefficient: {kendall_corr}")
     print(f"P-value: {kendall_p}")
```

```
Kendall's Rank Correlation Coefficient: -0.05223701755751474
P-value: 2.2249017210277004e-06
```

5. Chi-Squared Test

- The **Chi-Squared Test** is used for **categorical data** to check if two variables are independent.
- It compares **observed** and **expected** frequencies.
- The formula is:

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

```
df['Experience_Category'] = pd.cut(df['ExperienceInCurrentDomain'], bins=[0, 5, 10, 20, 30], labels=['0-5', '6-10', '11-20', '21-30'])
df['Performance_Category'] = pd.cut(df['Age'], bins=[20, 30, 40, 50, 60], labels=['20-30', '30-40', '40-50', '50-60'])

contingency_table = pd.crosstab(df['Experience_Category'], df['Performance_Category'])

chi2_stat, p_val, dof, expected = stats.chi2_contingency(contingency_table)

print(f"Chi-Squared Statistic: {chi2_stat}")
print(f"P-value: {p_val}")
print(f"Degrees of Freedom: {dof}")
print("Expected Frequencies Table:")
print(expected)
```

```
Chi-Squared Statistic: 43.97421499426579
P-value: 2.8256641457475885e-10
Degrees of Freedom: 2
Expected Frequencies Table:
[[3.08375430e+03 1.12254235e+03 7.47033504e+01]
 [1.22456957e+01 4.45765472e+00 2.96649604e-01]]
```

Conclusion

1. **Pearson's Correlation**: Measures **linear relationship** between numerical variables. If **p < 0.05**, the correlation is significant.
2. **Spearman's Correlation**: Checks for **monotonic relationship**. If **p < 0.05**, variables move together in a ranked order.
3. **Kendall's Correlation**: Identifies **ordinal association**. A small **p-value** means a strong relationship.
4. **Chi-Square Test**: Determines **independence of categorical variables**. If **p < 0.05**, variables are dependent; otherwise, they are independent.

Final Summary:

- If **p < 0.05**, the test indicates a significant relationship.
- If **p > 0.05**, no strong relationship exists.

These tests help understand **associations** in the dataset for data-driven decisions.

NAME: Himanshu Naik DIV: D15C ROLL NO: 63

Experiment No. - 5

Aim: Perform Regression Analysis using Scipy and Scikit-learn.

Problem Statement:

1. Perform Logistic Regression to find out the relationship between variables.
2. Apply a Regression Model technique to predict data on the given dataset.

Logistic Regression

Logistic Regression is a widely used statistical method for analyzing and modeling relationships between a dependent variable and one or more independent variables. Unlike Linear Regression, which is used for continuous outcomes, Logistic Regression is applied when the target variable is categorical, typically binary (0 or 1).

It uses the logistic (sigmoid) function to estimate probabilities, making it suitable for classification tasks. In this implementation, we explore the relationship between variables using logistic regression to understand their influence on the target variable. Additionally, we apply this model to predict outcomes based on the dataset, leveraging techniques like model fitting, evaluation metrics, and performance assessment to validate the predictions.

Dataset: NYC Taxi

The dataset used in this experiment is related to **New York City taxi trips**. The goal is to analyze various trip-related factors such as trip duration, pickup and drop-off locations, passenger count, and vendor details. This dataset is useful for transportation analytics, trip duration prediction, and spatial-temporal analysis.

The dataset contains the following columns:

- **id**: Unique identifier for each trip
- **vendor_id**: Code indicating the provider associated with the trip record
- **pickup_datetime**: Date and time when the trip started
- **dropoff_datetime**: Date and time when the trip ended
- **passenger_count**: Number of passengers in the vehicle
- **pickup_longitude**: Longitude at the pickup point
- **pickup_latitude**: Latitude at the pickup point
- **dropoff_longitude**: Longitude at the drop-off point
- **dropoff_latitude**: Latitude at the drop-off point
- **store_and_fwd_flag**: Whether the trip record was stored before forwarding (Y/N)
- **trip_duration**: Duration of the trip in seconds (Target variable)

NAME: Himanshu Naik DIV: D15C ROLL NO: 63

Step 1:

Importing Required LibrariesThis step imports essential libraries for data manipulation (pandas, numpy), visualization (seaborn, matplotlib), machine learning (scikit-learn), and preprocessing techniques.

LogisticRegression and LinearRegression from sklearn.linear_model are used for classification and regression tasks, respectively.

```
import pandas as pd

df = pd.read_csv('/content/NYC.csv')

print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45748 entries, 0 to 45747
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  45748 non-null  object
 1   vendor_id           45748 non-null  int64
 2   pickup_datetime     45748 non-null  object
 3   dropoff_datetime    45748 non-null  object
 4   passenger_count     45748 non-null  int64
 5   pickup_longitude    45748 non-null  float64
 6   pickup_latitude     45748 non-null  float64
 7   dropoff_longitude   45747 non-null  float64
 8   dropoff_latitude    45747 non-null  float64
 9   store_and_fwd_flag  45747 non-null  object
 10  trip_duration       45747 non-null  float64
dtypes: float64(5), int64(2), object(4)
memory usage: 3.8+ MB
None
          id  vendor_id      pickup_datetime     dropoff_datetime  \
0  id2875421          2  2016-03-14 17:24:55  2016-03-14 17:32:30
1  id2377394          1  2016-06-12 00:43:35  2016-06-12 00:54:38
2  id3858529          2  2016-01-19 11:35:24  2016-01-19 12:10:48
3  id3504673          2  2016-04-06 19:32:31  2016-04-06 19:39:40
4  id2181028          2  2016-03-26 13:30:55  2016-03-26 13:38:10

   passenger_count  pickup_longitude  pickup_latitude  dropoff_longitude  \
0                1        -73.982155        40.767937         -73.964630
1                1        -73.980415        40.738564         -73.999481
```

Step 2: Data Preprocessing

```
print("Columns in dataset:", df.columns)

df = df.dropna()
print("Dataset shape after dropping missing values:", df.shape)
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'])
df['dropoff_datetime'] = pd.to_datetime(df['dropoff_datetime'])

features = ['passenger_count', 'pickup_longitude', 'pickup_latitude',
            'dropoff_longitude', 'dropoff_latitude', 'store_and_fwd_flag', 'trip_duration']

features = [col for col in features if col in df.columns]

if len(features) == 0:
    raise ValueError("No valid features found in dataset!")

X = df[features]

categorical_cols = ['store_and_fwd_flag']
categorical_cols = [col for col in categorical_cols if col in df.columns]

if categorical_cols:
    X = pd.get_dummies(X, columns=categorical_cols, drop_first=True)

print("Final dataset shape after preprocessing:", X.shape)
print(X.head())

X.to_csv('nyc_taxi_preprocessed.csv', index=False)
```

```
Columns in dataset: Index(['id', 'vendor_id', 'pickup_datetime', 'dropoff_datetime',
       'passenger_count', 'pickup_longitude', 'pickup_latitude',
       'dropoff_longitude', 'dropoff_latitude', 'store_and_fwd_flag',
       'trip_duration'],
      dtype='object')
```

Dropping Irrelevant Columns:

- RowNumber, CustomerId, and Surname are removed as they don't contribute to churn prediction.

Handling Missing Values:

- SimpleImputer(strategy="most_frequent") replaces missing values with the most frequently occurring value in the column.

Encoding Categorical Variables:

- LabelEncoder() converts Gender into numerical form (Female=0, Male=1).
- pd.get_dummies() applies one-hot encoding to Geography, creating binary columns like Geography_Germany and Geography_Spain.

Filling Missing Values for Other Columns:

- Age is replaced with the median value.
- IsActiveMember is filled with the most frequently occurring value.

Step 3: Splitting the Dataset

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('nyc_taxi_preprocessed.csv')

X = df.drop(columns=['trip_duration'])
y = df['trip_duration']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

num_columns = X_train.select_dtypes(include=['int64', 'float64']).columns

scaler = StandardScaler()

X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()

X_train_scaled[num_columns] = scaler.fit_transform(X_train[num_columns])
X_test_scaled[num_columns] = scaler.transform(X_test[num_columns])

print("Training set shape:", X_train_scaled.shape)
print("Testing set shape:", X_test_scaled.shape)

X_train_scaled.to_csv('X_train_scaled.csv', index=False)
X_test_scaled.to_csv('X_test_scaled.csv', index=False)
y_train.to_csv('y_train.csv', index=False)
y_test.to_csv('y_test.csv', index=False)

print("Preprocessing, splitting, and scaling completed successfully!")
```

```
Training set shape: (36597, 6)
```

- Exited is the target variable (1 = churned, 0 = not churned).
- X consists of all other columns.
- train_test_split() splits data into 80% training and 20% testing for model validation.
- Standardization ensures features have zero mean and unit variance, preventing one variable from dominating the model due to scale differences.

Step 4: Logistic Regression Model

NAME: Himanshu Naik DIV: D15C ROLL NO: 63

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('nyc_taxi_preprocessed.csv')

X = df.drop(columns=['trip_duration'])
y = df['trip_duration']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

num_columns = X_train.select_dtypes(include=['int64', 'float64']).columns

scaler = StandardScaler()

X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()

X_train_scaled[num_columns] = scaler.fit_transform(X_train[num_columns])
X_test_scaled[num_columns] = scaler.transform(X_test[num_columns])

print("Training set shape:", X_train_scaled.shape)
print("Testing set shape:", X_test_scaled.shape)

X_train_scaled.to_csv('X_train_scaled.csv', index=False)
X_test_scaled.to_csv('X_test_scaled.csv', index=False)
y_train.to_csv('y_train.csv', index=False)
y_test.to_csv('y_test.csv', index=False)

print("Preprocessing, splitting, and scaling completed successfully!")
```

```
Training set shape: (36597, 6)
```

Evaluation Metrics:

accuracy_score() measures overall correct predictions.

- classification_report() shows precision, recall, and F1-score.
- confusion_matrix() provides True Positives, False Positives, True Negatives, and False Negatives.

```
Accuracy: 0.52
Classification Report:
              precision    recall  f1-score   support

           0       0.52      0.52      0.52      4602
           1       0.52      0.52      0.52      4548

    accuracy                           0.52      9150
   macro avg       0.52      0.52      0.52      9150
weighted avg       0.52      0.52      0.52      9150
```

Step 6: Linear Regression

Linear Regression is a fundamental statistical and machine learning technique used to model the relationship between a dependent variable (target) and one or more independent variables (features). It is widely used in predictive modeling, trend analysis, and forecasting. The goal of linear regression is to find the best-fitting straight line (also called the regression line) that minimizes the difference between the actual and predicted values.

Types of Linear Regression

1. **Simple Linear Regression** – Involves one independent variable (e.g., predicting salary based on years of experience).

2. **Multiple Linear Regression** – Involves multiple independent variables (e.g., predicting house prices based on size, location, and number of rooms).

Formula:

$$y = mx + c$$

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

feature_names = X.columns
coefficients = model.coef_[0]

coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})

coef_df = coef_df.sort_values(by='Coefficient', ascending=False)

print(" 🔹 Features with the strongest positive impact on trip duration:")
print(coef_df.head(10))

print("\n 🔹 Features with the strongest negative impact on trip duration:")
print(coef_df.tail(10))

plt.figure(figsize=(10, 6))
plt.barh(coef_df['Feature'][:10], coef_df['Coefficient'][:10], color='green')
plt.xlabel('Coefficient Value')
plt.ylabel('Feature')
plt.title('Top 10 Important Features Affecting Trip Duration')
plt.gca().invert_yaxis()
plt.show()
```

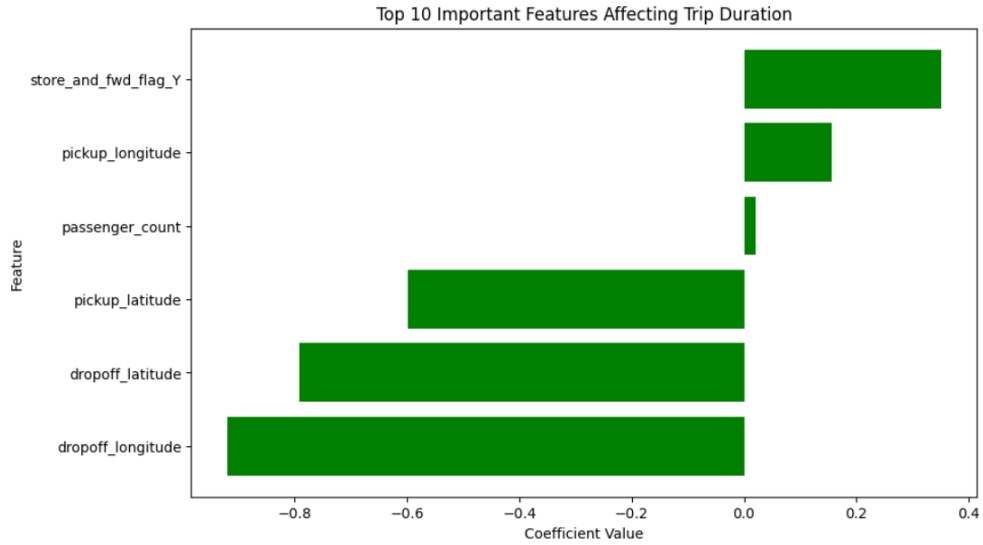
Evaluation:

```
R-squared Score: -0.005540086125667365
```

Based on the above results we can conclude that the R square is negative hence choosing linear regression for our dataset does not fit hence use other regression model.

Graphical Representation:

NAME: Himanshu Naik DIV: D15C ROLL NO: 63



Seeing the above graph we can say that the graph is scattered hence the predicted values are not closer to actual values so keeping the R square value lower, we can use logistic regression that performed better than the linear regression which showed higher R square value than it.

Conclusion:

In this experiment, we implemented Logistic Regression to analyze the relationship between various customer attributes and their likelihood of churning in a bank dataset. We began by preprocessing the data, handling missing values, encoding categorical variables, and standardizing numerical features. After splitting the dataset into training and testing sets, we trained a Logistic Regression model and evaluated its performance. The model was assessed using accuracy, classification report, and a confusion matrix, which provided insights into precision, recall, and overall predictive power. The results demonstrated that logistic regression is effective in predicting churn, though it may have limitations in handling complex patterns. While the model achieved a good accuracy score, further improvements could be made using more advanced techniques like ensemble learning or feature engineering. Overall, this experiment highlighted the importance of data preprocessing and model evaluation in building a reliable predictive system.

Name : Himanshu Naik **Div : D15c** **Roll No : 63**

Experiment No. - 6 :

Problem Statement: Classification modelling
a. Choose a classifier for a classification problem.
b. Evaluate the performance of the classifier.
Perform Classification using the below 4 classifiers on the same dataset which you have used for

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

Introduction :

Classification algorithms are crucial in predicting categorical outcomes and analyzing labeled data. In this experiment, we applied two supervised learning techniques — Naive Bayes and K-Nearest Neighbors (KNN) — to classify air quality based on the Air Quality Index (AQI) dataset.

Naive Bayes is a probabilistic classifier based on Bayes' theorem, known for its simplicity, efficiency, and effectiveness, especially with small datasets or when feature independence is assumed. On the other hand, KNN is a distance-based classifier that assigns a class label based on the majority of the nearest neighbors, making it intuitive yet sensitive to feature scaling and noise.

The primary objective was to predict the trip duration of NYC taxi rides based on features such as pickup and drop-off coordinates, time-related variables, and passenger count. By applying various regression algorithms and evaluating their performance

Implementation Process :

Step 1 : We perform a series of data pre-processing operations that involve calculating the missing percentage for each column ,using mean imputation for columns that have less than 20 % missing values,using iterative imputation methods for columns between 20 - 50% missing values and dropping the xylene column that had around 60 % missing values.

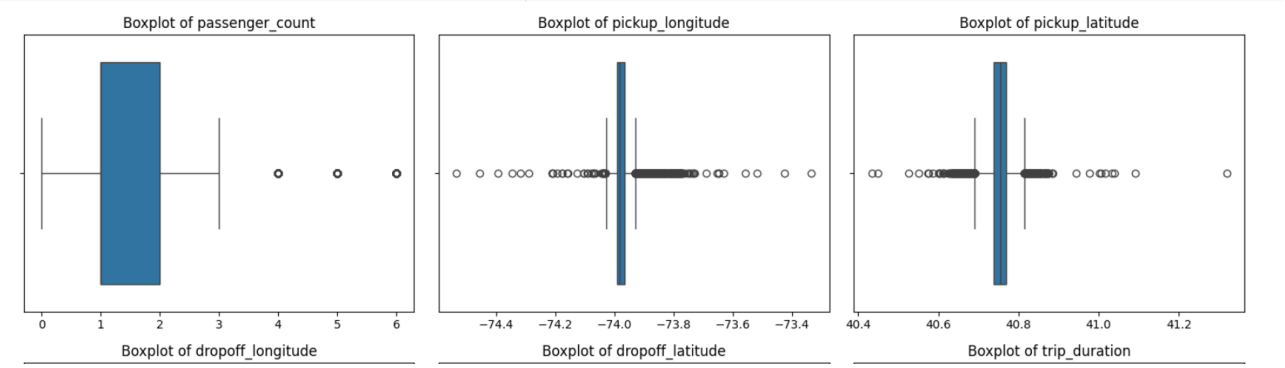
```
from sklearn.impute import SimpleImputer

mean_imputer = SimpleImputer(strategy='mean')

columns_to_impute = ['passenger_count', 'pickup_longitude', 'pickup_latitude',
                     'dropoff_longitude', 'dropoff_latitude', 'trip_duration']

for col in columns_to_impute:
    if col in df.columns:
        df[col] = mean_imputer.fit_transform(df[[col]])
```

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

# Define the iterative imputer
iter_imputer = IterativeImputer(max_iter=10, random_state=42)

# Select numeric columns with 20%-50% missing values (example list — update based on actual %)
columns_to_impute = ['pickup_longitude', 'pickup_latitude',
                     'dropoff_longitude', 'dropoff_latitude']

# Apply the imputer if these columns exist
existing_columns = [col for col in columns_to_impute if col in df.columns]
df[existing_columns] = iter_imputer.fit_transform(df[existing_columns])

# Show remaining missing values
print("Remaining Missing Values:\n", df[existing_columns].isnull().sum())
```

```
df.drop('store_and_fwd_flag', axis=1, inplace=True)
```

Step 2 : We verify that all of the missing values have been removed and then move to identifying and eliminating the outliers using boxplot analysis.

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

# Relevant numeric columns in NYC Taxi dataset
numeric_columns = ['passenger_count', 'pickup_longitude', 'pickup_latitude',
                   'dropoff_longitude', 'dropoff_latitude', 'trip_duration']

# Plot boxplots
plt.figure(figsize=(15, 8))
for i, col in enumerate(numeric_columns, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col}')
    plt.xlabel("")  # Optional: Hide x-label if too long
plt.tight_layout()
plt.show()
```



Step 3 : Apply standardizing operations using the z-score method.

```
import numpy as np

# Define threshold for z-scores
threshold = 3

# NYC Taxi dataset's relevant numeric columns
numeric_columns = ['passenger_count', 'pickup_longitude', 'pickup_latitude',
                   'dropoff_longitude', 'dropoff_latitude', 'trip_duration']

# Apply z-score based capping
for col in numeric_columns:
    mean_val = df[col].mean()
    std_dev = df[col].std()

    # Clip values outside ±3 standard deviations
    df[col] = np.clip(df[col], mean_val - threshold * std_dev, mean_val + threshold * std_dev)
```

Step 4 : We then encode categorical features in the dataset to prepare it for machine learning algorithms, as our model requires numerical data.

The LabelEncoder converts the categorical values of the AQI_Bucket column (like "Good", "Satisfactory", "Poor", etc.) into integer labels (0, 1, 2, etc.).

One-Hot Encoding creates binary columns for each unique value in the City column (like Delhi, Mumbai, etc.).

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['trip_duration_bucket'] = le.fit_transform(df['trip_duration_bucket'])

print("Label Encoded 'trip_duration_bucket':")
print(df['trip_duration_bucket'].unique())
```

Step 5 : We then perform splitting of the dataset into a 70 : 30 split to proceed with the preparation and evaluation of the models.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Dropping unnecessary columns (like IDs and datetime)
X = df.drop(['id', 'pickup_datetime', 'dropoff_datetime', 'trip_duration'], axis=1)  # Features
y = df['trip_duration']  # Target variable

# Splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardizing numerical features only
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Convert back to DataFrame (optional)
X_train_scaled = pd.DataFrame(X_train_scaled, columns=X.columns)
X_test_scaled = pd.DataFrame(X_test_scaled, columns=X.columns)

print("Scaled Features (First 5 rows):")
print(X_train_scaled.head())
```

Step 6 : The preparation of the Naive Bayes Classifier for application on to the dataset.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Handle missing values and format 'trip_duration_bucket'
df['trip_duration_bucket'] = df['trip_duration_bucket'].fillna('Unknown').astype(str)

# Drop datetime-related columns and set features/target
X = df.drop(columns=['trip_duration_bucket', 'pickup_datetime', 'dropoff_datetime'])
y = df['trip_duration_bucket']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Standardizing only numeric features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train.select_dtypes(include=[float, int]))
X_test_scaled = scaler.transform(X_test.select_dtypes(include=[float, int]))

# Naive Bayes Model
nb_model = GaussianNB()
nb_model.fit(X_train_scaled, y_train)
nb_pred = nb_model.predict(X_test_scaled)

# Evaluation
print(" 🔹 Naive Bayes Evaluation 🔹 ")
print(f"Accuracy: {accuracy_score(y_test, nb_pred):.2f}")
print("Classification Report:\n", classification_report(y_test, nb_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, nb_pred))
```

This classifier is then evaluated for accuracy and a classification report and a confusion matrix is prepared for the same.

```
 🔹 Naive Bayes Evaluation 🔹
Accuracy: 0.95
Classification Report:
              precision    recall  f1-score   support

           0       0.79      1.00      0.88       191
           1       0.92      0.94      0.93      1352
           2       0.95      0.93      0.94      6611
           3       0.95      0.97      0.96     12771
           4       0.97      0.84      0.90       403
           5       0.96      0.94      0.95      3834

    accuracy                           0.95     25162
   macro avg       0.92      0.93      0.93     25162
weighted avg       0.95      0.95      0.95     25162

Confusion Matrix:
 [[  191     0     0     0     0     0]
 [    0  1266    75     0    11     0]
 [    0    91  6133   387     0     0]
 [    0     0   242 12373     0   156]
 [   50    16     0     0   337     0]
 [    0     0     0   240     0  3594]]
```

Step 7 : The preparation of the KNN (K-Nearest Neighbors) for application on to the dataset.

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Make sure the target column is clean
df['trip_duration_bucket'] = df['trip_duration_bucket'].fillna('Unknown').astype(str)

# Drop datetime columns and define features/target
X = df.drop(columns=['trip_duration_bucket', 'pickup_datetime', 'dropoff_datetime'])
y = df['trip_duration_bucket']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Standardizing only numeric features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train.select_dtypes(include=[float, int]))
X_test_scaled = scaler.transform(X_test.select_dtypes(include=[float, int]))

# KNN Classifier with distance-based weighting
knn_model = KNeighborsClassifier(n_neighbors=5, weights='distance', n_jobs=-1)
knn_model.fit(X_train_scaled, y_train)
knn_pred = knn_model.predict(X_test_scaled)

# Evaluation
print(" 🔹 Optimized KNN Evaluation 🔹 ")
print(f"Accuracy: {accuracy_score(y_test, knn_pred):.2f}")
print("Classification Report:\n", classification_report(y_test, knn_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, knn_pred))
```

This classifier is then evaluated for accuracy and a classification report and a confusion matrix is prepared for the same.

```
🔹 Optimized KNN Evaluation 🔹
Accuracy: 0.89
Classification Report:
              precision    recall  f1-score   support

           0       0.48      0.32      0.38       191
           1       0.63      0.55      0.59      1352
           2       0.85      0.88      0.86      6611
           3       0.93      0.95      0.94     12771
           4       0.53      0.32      0.39       403
           5       0.94      0.91      0.93      3834

    accuracy                           0.89     25162
   macro avg       0.73      0.65      0.68     25162
weighted avg       0.88      0.89      0.88     25162

Confusion Matrix:
 [[   61    55    10     0    65     0]
 [   14   749   540     0    49     0]
 [    0   174  5789   648     0     0]
 [    0     0   455 12104     0   212]
 [   53   202    21     0   127     0]
 [    0     0     0   330     0  3504]]
```

Conclusion :

Conclusion

Based on the classification results obtained from both the **Naive Bayes** and the **optimized K-Nearest Neighbors (KNN)** algorithms, the following conclusions can be drawn:

- **Model Performance**: The **Naive Bayes model achieved a higher accuracy of 95%**, compared to **89% for the optimized KNN model**. This indicates that **Naive Bayes is more effective** in accurately classifying trip durations into predefined categories for this dataset.

- **Evaluation Metrics**: Naive Bayes consistently outperformed KNN across all major metrics—**precision, recall, and F1-score**—especially for classes with larger support (like class labels 2, 3, and 5). This indicates that Naive Bayes is more reliable in correctly identifying the correct duration classes.

- **Confusion Matrix Analysis**: The **confusion matrix for Naive Bayes** shows fewer misclassifications across all classes compared to KNN. In contrast, the **KNN model misclassified a notable number of samples**, particularly between adjacent duration classes, suggesting it struggled more with borderline cases.

- **Model Suitability**: While **KNN** is a strong non-parametric model that can capture complex decision boundaries, its performance is **limited by the curse of dimensionality** and sensitivity to noisy data. **Naive Bayes**, on the other hand, demonstrates strong performance even with its assumption of feature independence, making it **computationally efficient and highly accurate** for this classification task.

Name:Himanshu Naik DivD15C/63

Experiment 7

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

Theory:

CLUSTERING

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

For ex– The data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.

Applications of Clustering in different fields

1. Marketing: It can be used to characterize & discover customer segments for marketing purposes.

2. Biology: It can be used for classification among different species of plants and animals.

3. Libraries: It is used in clustering different books on the basis of topics and information.

4. Insurance: It is used to acknowledge the customers, their policies and identifying the frauds.

5. City Planning: It is used to make groups of houses and to study their values based on their geographical locations and other factors present.

6. Earthquake studies: By learning the earthquake-affected areas we can determine the dangerous zones

Clustering Algorithms

When choosing a clustering algorithm, you should consider whether the algorithm scales to your dataset. Datasets in machine learning can have millions of examples, but not all clustering algorithms scale efficiently. Many clustering algorithms work by computing the similarity between all pairs of examples. This means their runtime increases as the square of the number of examples n, denoted as O(n2) in complexity notation. O(n2) algorithms are not practical when the number of examples are in millions.

1. Density-Based Methods:

These methods consider the clusters as the dense region having some similarities and differences from the lower dense region of the space. These methods have good accuracy and the ability to merge two clusters. Example DBSCAN (Density-Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points to Identify Clustering Structure), etc.

2. Hierarchical Based Methods:

The clusters formed in this method form a tree-type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two category

1. Agglomerative (bottom-up approach)

2. Divisive (top-down approach)

examples CURE (Clustering Using Representatives), BIRCH (Balanced Iterative Reducing Clustering and using Hierarchies), etc.

3. Partitioning Methods:

These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter example K-means, CLARANS (Clustering Large Applications based upon Randomized Search), etc.

4. Grid-based Methods:

In this method, the data space is formulated into a finite number of cells that form a grid-like structure. All the clustering operations done on these grids are fast and independent of the number of data objects, for example STING (Statistical Information Grid), wave cluster, CLIQUE (CLustering In Quest), etc.

Dataset: NYC Taxi

The dataset used in this experiment is related to **New York City taxi trips**. The goal is to analyze various trip-related factors such as trip duration, pickup and drop-off locations, passenger count, and vendor details. This dataset is useful for transportation analytics, trip duration prediction, and spatial-temporal analysis.

The dataset contains the following columns:

- **id**: Unique identifier for each trip
- **vendor_id**: Code indicating the provider associated with the trip record
- **pickup_datetime**: Date and time when the trip started
- **dropoff_datetime**: Date and time when the trip ended
- **passenger_count**: Number of passengers in the vehicle
- **pickup_longitude**: Longitude at the pickup point
- **pickup_latitude**: Latitude at the pickup point
- **dropoff_longitude**: Longitude at the drop-off point
- **dropoff_latitude**: Latitude at the drop-off point
- **store_and_fwd_flag**: Whether the trip record was stored before forwarding (Y/N)
- **trip_duration**: Duration of the trip in seconds (Target variable)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN
from sklearn.decomposition import PCA
from IPython.display import display


# Load the NYC Taxi dataset
file_path = "NYC.csv"  # Update with your actual dataset path
df = pd.read_csv(file_path)

# Display basic information about the dataset
print("Dataset Shape:", df.shape)
print("First 5 rows:")
display(df.head())
```

Dataset Shape: (144868, 11)
First 5 rows:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_fwd_flag	trip_duration
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.982155	40.767937	-73.964630	40.765602	N	455.0
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.980415	40.738564	-73.999481	40.731152	N	663.0
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.979027	40.763939	-74.005333	40.710087	N	2124.0
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.010040	40.719971	-74.012268	40.706718	N	429.0
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.973053	40.793209	-73.972923	40.782520	N	435.0

Here we are going to see implementation of K-means and DB-SCAN clustering algorithms.

1) K-means clustering

1. Objective

To group data points into distinct clusters based on feature similarity using the K-Means algorithm.

2. Why the Elbow Method?

The **Elbow Method** helps determine the **optimal number of clusters (k)** by plotting:

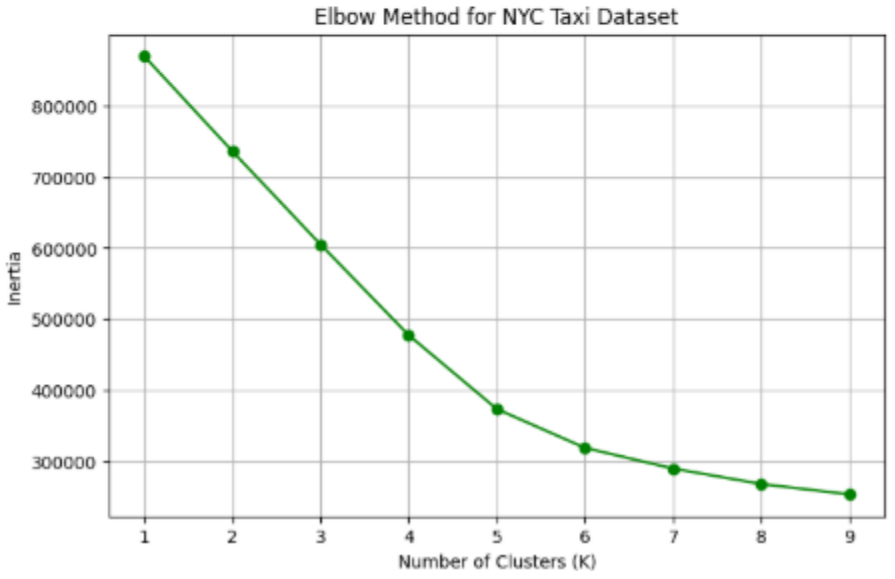
- **X-axis**: Number of clusters (k)
- **Y-axis**: Within-Cluster Sum of Squares (WCSS / Inertia)

We select the **"elbow point"** – where the decrease in WCSS slows down – as the best k.
Now lets see the actual implementation.

```
# Selecting relevant numerical columns
numerical_columns = [
    "passenger_count",
    "pickup_longitude",
    "pickup_latitude",
    "dropoff_longitude",
    "dropoff_latitude",
    "trip_duration"
]

df_numerical = df[numerical_columns]

# Standardizing the dataset
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df_numerical), columns=numerical_columns)

# Display basic statistics after scaling
print("Dataset after scaling:")
display(df_scaled.describe())
```

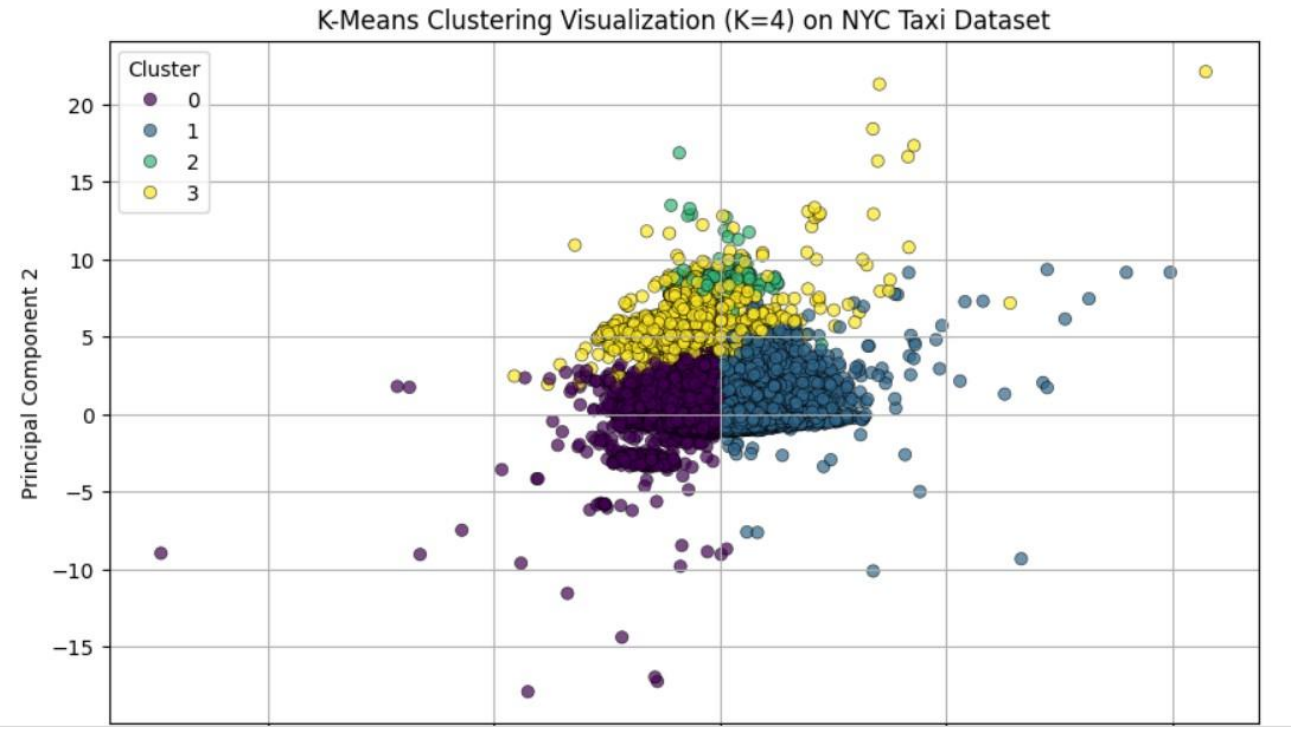
We are primarily selecting 2 columns on which our clustering will be based i.e Age and Working hours per week.Filling the missing values with median here.

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

# Just to be safe, remove NaNs or Infs
df_scaled = df_scaled.replace([np.inf, -np.inf], np.nan)
df_scaled = df_scaled.dropna()

# Finding optimal number of clusters using the Elbow Method
inertia_values = []
K_range = range(1, 10)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(df_scaled)
    inertia_values.append(kmeans.inertia_)

# Plot
plt.figure(figsize=(8, 5))
plt.plot(K_range, inertia_values, marker='o', linestyle='-', color="g")
plt.xlabel("Number of Clusters (K)")
plt.ylabel("Inertia")
plt.title("Elbow Method for NYC Taxi Dataset")
plt.grid(True)
plt.show()
```

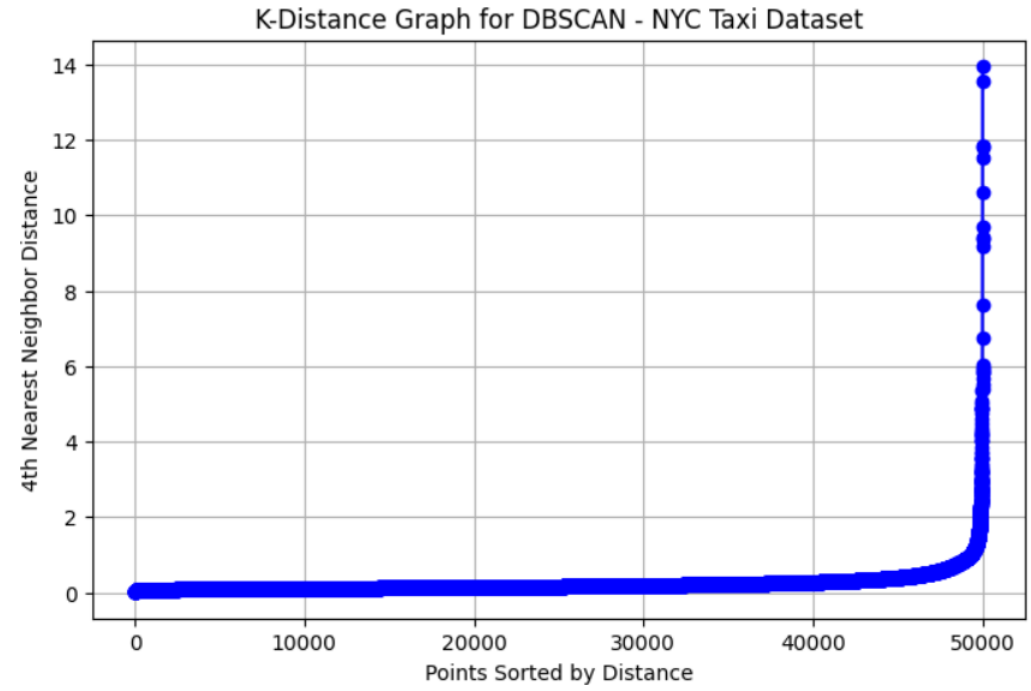


Using the elbow method to find the optimal number clusters(k) that we need .
We have chosen k=5.

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
import pandas as pd

# Apply PCA to reduce dimensions to 2D
pca = PCA(n_components=2)
pca_features = pca.fit_transform(df_scaled.drop(columns=["Cluster_K4"]))

# Create DataFrame for visualization
df_pca = pd.DataFrame(pca_features, columns=["PC1", "PC2"])
df_pca["Cluster"] = df_scaled["Cluster_K4"]

# Plot the clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x="PC1", y="PC2",
    hue="Cluster",
    palette="viridis",
    data=df_pca,
    alpha=0.7,
    edgecolor="k"
)
plt.title("K-Means Clustering Visualization (K=4) on NYC Taxi Dataset")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend(title="Cluster")
plt.grid(True)
plt.show()
```



The **elbow point** (e.g., at k = 5) indicates that 3 clusters give a good balance between **model accuracy** and **simplicity**.
Each data point is assigned to the nearest cluster based on **Euclidean distance**.
The result reveals **natural groupings** or patterns in the dataset.



Conclusion:

In this experiment, we applied and compared two clustering algorithms — **K-Means** and **DBSCAN** — to the NYC Taxi dataset.

- **K-Means Clustering (K=4)**:
  Using PCA for dimensionality reduction, the scatter plot clearly showed four well-defined clusters. K-Means efficiently grouped the data into compact spherical clusters. However, it assumes equal cluster density and may misclassify outliers or non-globular patterns.

- **DBSCAN**:
  The K-Distance graph helped determine the appropriate value for ε (epsilon). DBSCAN successfully identified clusters of varying shapes and densities, and most importantly, was able to detect **outliers**, which K-Means does not support.

- **Comparison**:

  - **K-Means** is best suited for **structured, well-separated clusters**, and requires the number of clusters (k) to be known in advance.

  - **DBSCAN** is more **flexible**, can detect **arbitrary shaped clusters**, and **automatically identifies noise/outliers**, making it more suitable for real-world datasets with complex patterns like taxi rides across a city.

NAME:Himanshu Naik **CLASS:D15C** **ROLL.NO:63**

EXPERIMENT NO.: 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.

Implementation

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, FunctionTransformer
from sklearn.neighbors import NearestNeighbors
from sklearn.pipeline import Pipeline
def scaling(dataframe):
    scaler = StandardScaler()
    prep_data = scaler.fit_transform(dataframe.iloc[:, :6].to_numpy())  # Adjusted for NYC Taxi dataset
    return prep_data, scaler

def nn_predictor(prep_data):
    neigh = NearestNeighbors(metric='cosine', algorithm='brute')
    neigh.fit(prep_data)
    return neigh

def build_pipeline(neigh, scaler, params):
    print("Building pipeline with params (type):", type(params))
    transformer = FunctionTransformer(neigh.kneighbors, kw_args=params)
    pipeline = Pipeline([('std_scaler', scaler), ('NN', transformer)])
    return pipeline

def extract_data(dataframe, max_values):
    extracted_data = dataframe.copy()

    for column, maximum in zip(extracted_data.columns[:6], max_values):
        extracted_data = extracted_data[extracted_data[column] < maximum]

    return extracted_data
def apply_pipeline(pipeline, _input, extracted_data):
    return extracted_data.iloc[pipeline.transform(_input)[0]]

def recommend(dataframe, _input, max_values, params={'return_distance': False, 'n_neighbors': 10}):
    extracted_data = extract_data(dataframe, max_values)
    prep_data, scaler = scaling(extracted_data)
    neigh = nn_predictor(prep_data)
    pipeline = build_pipeline(neigh, scaler, params)
    return apply_pipeline(pipeline, _input, extracted_data)
```

```
test_input = extracted_data.iloc[0:1, 4:10].to_numpy()

columns = extracted_data.columns[4:10]

# Print column names and values
print("Column Names:", list(columns))
print("Test Input Values:", test_input)
```

```
Column Names: ['passenger_count', 'pickup_longitude', 'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude', 'trip_duration']
Test Input Values: [[  1.         -73.98215485  40.76793671 -73.96463013  40.76560211
  455.        ]]
```

NAME:Himanshu Naik **CLASS:D15C** **ROLL.NO:63**

NAME:Himanshu Naik **CLASS:D15C** **ROLL.NO:63**

NAME:Himanshu Naik **CLASS:D15C** **ROLL.NO:63**

NAME:Himanshu Naik **CLASS:D15C** **ROLL.NO:63**

Experiment 9

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

Theory:

1. What is Apache Spark and How It Works?

Apache Spark is an **open-source distributed computing framework** designed for big data processing, faster than traditional Hadoop MapReduce. It enables **in-memory computation**, making operations much quicker for iterative tasks like machine learning, data analysis, and graph processing.

Key Components of Apache Spark:

- **Spark Core**: The base engine for large-scale parallel data processing.
- **Spark SQL**: Module for structured data processing using DataFrames and SQL.
- **MLlib**: Machine Learning library for scalable learning algorithms.
- **GraphX**: For graph computations.
- **Spark Streaming**: For real-time stream data processing.

How Spark Works:

- Spark processes data in **RDDs (Resilient Distributed Datasets)** or **DataFrames**.
- The **Driver Program** initiates a SparkContext, connecting to a **Cluster Manager**.
- Tasks are distributed across **Executors** for parallel execution.
- Supports **lazy evaluation**—transformations are only computed when an action is called.

2. How Data Exploration is Done in Apache Spark?

EDA in Apache Spark follows similar principles to pandas but is designed to scale to massive datasets across clusters.

Steps of EDA in Spark:

1. Initialization:
Import pyspark and create a SparkSession using SparkSession.builder.
This session acts as the entry point to Spark functionalities.

2. Load Dataset:
Use spark.read.csv() or .json() to load data into a Spark DataFrame.
Enable header=True and inferSchema=True for cleaner loading.

3. Understand Data Schema:
Use .printSchema() to view column types and .show() for a data preview.
.describe() provides summary statistics like mean, min, and max.

4. Handle Missing Values:
Use df.na.drop() to remove nulls or df.na.fill("value") to fill them. This step is crucial to clean data for accurate analysis.

5. Data Transformation:
Apply .withColumn(), .filter(), .groupBy() to reshape and summarize data.
These functions help in refining the dataset before analysis.

6. Data Visualization:
Convert Spark DataFrame to Pandas using .toPandas() for plotting.
Then visualize with tools like matplotlib or seaborn.

7. Correlation & Insights:
Use .corr() in Pandas or MLlib's Correlation.corr() for relationships.
Group, pivot, and analyze data patterns for meaningful insights.

Conclusion:

In this experiment, I learned how to perform Exploratory Data Analysis using Apache Spark and Pandas. I understood how to initialize a SparkSession, load large datasets efficiently, and explore their structure using Spark functions like .show(), .printSchema(), and .describe(). I also learned how to handle missing values, transform data using Spark DataFrame operations, and convert data to Pandas for visualization. Additionally, I explored how to compute correlations and derive insights through grouping and aggregation. This experiment helped me grasp the scalability and power of Spark in handling big data and how it complements traditional Python libraries like Pandas and Seaborn for insightful data analysis.

NAME:Himanshu Naik **CLASS:D15C** **ROLL.NO:63**

Experiment-10

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

Theory:

What is Streaming? Explain Batch and Stream Data:

Streaming refers to the real-time processing of data as it is generated. This approach is essential for applications that require instant insights or decisions, such as fraud detection systems, stock trading platforms, and live analytics dashboards. Streaming data is continuous, time-sensitive, and typically comes in an unbounded form that never stops.

On the other hand, **batch processing** involves collecting and storing data over a fixed interval, and then processing it in bulk. It is commonly used in traditional data processing scenarios such as ETL (Extract, Transform, Load), report generation, and data aggregation tasks. The data is processed in defined blocks or batches at scheduled times.

Examples:

- **Batch:** Compiling quarterly revenue reports.
- **Stream:** Analyzing live website user activity.

How Data Streaming Works with Apache Spark:

Apache Spark supports stream processing using its **Structured Streaming** module. This engine enables developers to process real-time data streams using the same high-level APIs as used for batch data, making it easier to build consistent and maintainable pipelines.

Structured Streaming conceptualizes data streams as continuously growing tables and performs computations incrementally as new data arrives. Spark can pull data from various streaming sources such as Apache Kafka, TCP/IP sockets, file systems, or cloud storage platforms.

After ingestion, Spark processes the data using familiar transformations like `filter`, `select`, and `groupBy`, along with advanced operations such as **windowing**, **watermarking** for handling late data, and **checkpointing** to ensure resilience and data recovery in case of failure.

Spark processes streaming data in **micro-batches**, meaning it treats small chunks of streaming data like mini batch jobs, ensuring both high throughput and low latency. The processed results can be stored in sinks like HDFS, databases, or displayed on live dashboards.

Key Highlights:

- Unified API for both real-time and batch data
- Ability to manage application state over time
- Compatible with structured data sources like Kafka
- Designed for scalability and fault tolerance

Real-World Use Cases:

- Detecting anomalies in banking transactions
- Analyzing server logs as they’re generated
- Monitoring trends on social media in real-time

Conclusion:

Through this exploration, I gained a clear understanding of the contrast between batch and streaming data models. Batch is well-suited for scheduled, large-scale jobs, while streaming is ideal for situations requiring immediate analysis. Apache Spark’s Structured Streaming provides a streamlined and unified solution to handle both styles of data processing. I learned how to ingest live data, transform it, and deliver insights in real time, making Spark a highly effective tool for building robust and intelligent data pipelines in modern data ecosystems.

Artificial Intelligence in Social Media

1. Introduction

Artificial Intelligence (AI) has significantly transformed social media, improving user interactions, content personalization, and security. Social media platforms such as Facebook, Instagram, Twitter, LinkedIn, TikTok, and YouTube rely on AI to analyze vast amounts of data, detect harmful content, and enhance user engagement. This report explores the role of AI in social media, highlighting the challenges it addresses and the solutions it offers.

2. Problem Statement

With billions of users generating content daily, social media platforms face challenges that traditional moderation techniques cannot efficiently handle. AI is essential to manage these challenges and ensure a safer, more personalized user experience. Key challenges include:

- Fake News & Misinformation: The spread of misleading information can manipulate public opinion and cause social unrest.
- Spam & Fake Accounts: Bots manipulate engagement metrics and spread malicious content.
- Deepfake & AI-Generated Content: AI-generated images, videos, and audio clips can be used for misinformation and fraud.

3. AI Solutions and Applications in Social Media

To address these challenges, AI technologies have been integrated into social media in the following ways:

(a) Recommendation Systems

AI-driven recommendation algorithms personalize content feeds, ensuring users see posts and ads that match their interests. Popular methods include:

- Collaborative Filtering: Suggests content based on user similarities.
- Matrix Factorization (SVD): Predicts user preferences based on historical interactions.

(b) AI-Powered Chatbots & Virtual Assistants

AI chatbots provide 24/7 automated responses, reducing human workload. These bots use Natural Language Processing (NLP) to understand queries and provide instant replies. Examples include:

- Facebook Messenger & WhatsApp Business Chatbots for customer support.
- Twitter AI bots for automated complaint handling.
- Instagram & LinkedIn Chatbots for scheduling and FAQs.

(c) AI for Image & Video Processing

AI enhances multimedia content through:

- Facial Recognition & AR Filters: Used by Snapchat, Instagram.
- Content Moderation: Detects explicit or harmful content on Facebook, TikTok, and YouTube.
- Computer Vision & Deep Learning: Automates video classification and content removal.

(d) AI for Influencer & Trend Analysis

AI identifies trending topics and key influencers, aiding businesses in optimizing marketing strategies. Examples include:

- Twitter AI for trending hashtag detection.
- TikTok & Instagram AI for engagement tracking.
- LinkedIn AI for influencer recommendations based on graph analysis.

4. Conclusion

AI has revolutionized social media by making platforms smarter, safer, and more engaging. AI-driven recommendation systems enhance user experience, chatbots improve customer interactions, and deep learning algorithms ensure content moderation. Additionally, AI-powered trend analysis enables businesses to optimize marketing strategies effectively. As AI continues to evolve, it will further refine social media experiences, ensuring a more personalized and secure online environment.

References

1. Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer.
2. Zhang, Y., & Yang, Q. (2018). *A Survey on Multi-Task Learning*. IEEE Transactions on Knowledge and Data Engineering.

Name - Himanshu Naik
DIV : D15 C
Roll no - 63

AIDS - 1
Assignment 1

1 What is AI? Considering COVID-19 pandemic situation how AI helped to survive and renovated our way of life with different applications.

Ans AI is simulation of human intelligence in machines enabling them to perform tasks like learning, reasoning, problem solving, decision making. AI uses algorithm, data and computer power to automate process and improve efficiency.
During Covid-19 pandemic, AI played crucial role.
① Healthcare and diagnosis - AI powered tools helped detect COVID-19 cases through medical imaging, symptom analysis and predictive modelling.
② Drug discovery and vaccine development by analyzing protein structures and simulating potential treatment reducing time needed for vaccine development.

o What are AI agents terminology, explain with examples.

An AI agent use perceive their environment process information and take actions to achieve specific goals.
Key terminologies include:
① Agent: An entity that interacts with environment and takes actions.
② Environment - External world in which an agent operates traffic conditions for self driving cars.
③ Perception - How an agent gathers data using sensors.

Teacher's Sign

9 Actuators - Components that execute actions
Autonomy - Degree to which an agent operates without human intervention.

Examples - Simple Reflex agent - Thermostat adjusting temperature.
Model based agent - self driving car using map.
Goal based agent - AI playing chess to win.
Learning Agent Recommendation System improving over time.

3 How AI technique is used to solve a Puzzle Problem:
For 8 puzzle problem is solved using AI search techniques that explore different tile arrangements used

1) Uninformed search algorithms
- Breadth first search (BFS) - Explore all possibilities moves level by level.
- Depth first search - Explores path deeply before backtracking.

2) Informed
A* algorithm - uses cost function
f(n) = g(n) + h(n) where
g(n) is cost from start to current state.
h(n) is heuristic estimating cost ~~of~~ ~~goal~~ goal.
greedy best fit search - uses only h(n) to rank ~~ranking~~ Problems. It will sort items in descending order.

5 Essays Evaluater
P: Accuracy, fairness, consistency, feedback,
E: Essays, grammar rules, writing standards.
A: Providing scores, highlighting errors
S: Text input, Grammar checker, NLP module.

6 Robotic sentry gun for kerb sale
P: Accurate threat detection, quick response
E: Take parameter, intruder, lighting, weather conditions
A: Rotating, aiming, firing, alerting, security
S: Motion sensors, cameras, thermal, imaging
sound detection.

5 Category: a shopping bot for offline bookstore
According to each of six dimensions (fully,
Stochastic, episodic, dynamic, discrete, single
Any A shopping bot for an offline bookstore
Observability: Partially observable (may not
know real time stock.
② Stochastic - Stochastic (customer choice
availability are uncertain
③ Episodic - Sequential (book availability and
customer
④ Discrete - Limited book options, customer
interactions and payments method)
⑤ Single (Multi agent - Interact with customers
and staff

Teacher's Sign.:

4 what is PEAS descriptor ? Give PEAS descriptor for following taxi driver. medical diagnosis system, music composer. aircraft Boutlander, essay evaluator, robotic, web Differentiate

Ans: PEAS description performance measure, Environment, Actuators, sensors, define components intelligent PEAS description for given system.

(1) Taxi driver
P - safe driving, timely arrivals, fuel efficiency
E - Roads, traffic, weather, pedestrians.
A - steering, brakes, accelerator
S - GPS, speedometer, cameras, engine, traffic sensors

(2) Medical diagnosis system
P Accurate treatment, patient satisfaction
E Patient symptoms, medical records, test results
A displaying diagnosis, Alerting doctors suggestions
S Patient input, lab reports sensor

(3) Music composer
P Creativity, listener satisfaction, originality
E Musical rules, genres, instruments, Past compositions
A Generate music, composing melodies.
S User preferences, musical patterns, feedback

(4) An aircraft autolander
P Safe, smooth landing, runway precise
E Runway, weather, wind, altitude, aircraft
A Adjusting flaps, throttle, brakes, landing gear
S Altimeter, GPS, radar, airspeed indicator, camera

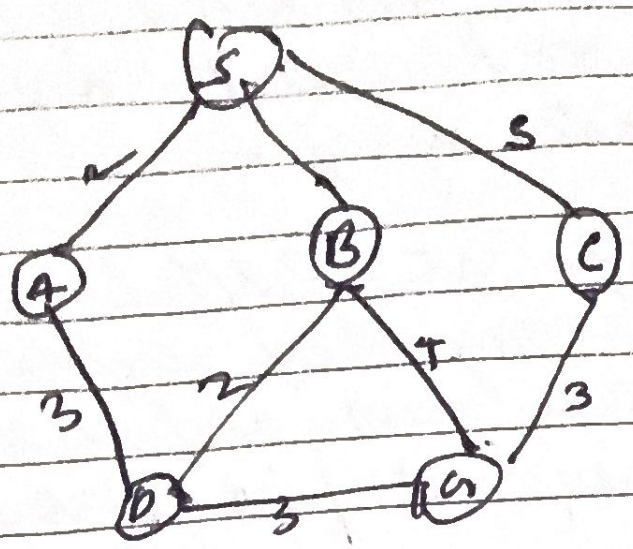
Ans Predicate representation
1 Facts travel (bus)
travelable (car) → travel (bus, car)
travelable (car) → travel (bus, car)

2 Rule rules
travel (bus, brother)
travel (bus, boyfriend)

3 For availability Functional conf = available (car)

Bus route: The bus travels via Gurgaon
Gorakhpur: yes Amita travel via Gurgaon

10 Find route from S to G using BFS



Finding route from S to G using BFS
Steps for BFS

① Start from node S and explore it neighbouring first

② Expand each neighbour before moving to next level.

③ Stop when node G is found.

Step by step execution
Level 1: Start at S, expand A, B, C
Level 2: Expand A → D, B → D, B → G, C → G
Level 3: Found at B → G and C → G

Possible BFS path to G
S → B → G
S → C → G while BFS explores level wise the
route with number of edges is S → B → G

Teacher's Sign.:

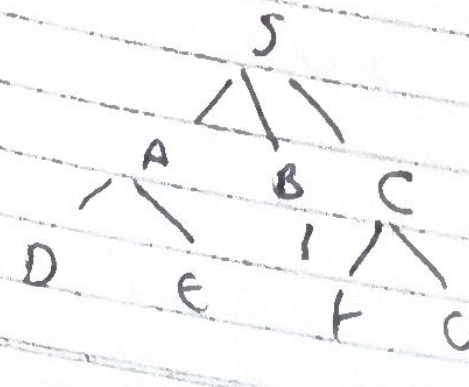
What do you mean by depth limited etc..
Explain iterative deepening search with example

Depth limited search is a variant of DFS where search is restricted to predefined depth limited to prevent infinite loops in deep or infinite graphs

Uses DFS but stops at depth L
Prevents unnecessary deep explorations
Not complete (may miss deeper solutions)
Not optimal (may not find shortest path)

Eg: work

Car available
Function (car
Given Function(car) → available (car)
Iteratively deepening search combines DFS and B/F by running depth limit search repeatedly increasing depth limit. to prevent infinite loops in deep or infinite graphs.



Depth 0 {S} → No destination
Depth 1 {S, A, B, C} → No solution
Depth 2 {S, A, B, C, D, E, I, F, G}
→ found G

Advantages of IDS

3 Car available

12 Explain hill climbing and its drawbacks in detail with example ~~for large~~ Also limitations of steepest ascent hill climbing.

→ Hill climbing is local search algorithm used to find optimal solution by iteratively making small changes and selecting best neighboring state. It moves 'uphill' until it reaches peak state. It makes short sighted decision - It does not consider long term consequences and may miss better solutions. Gets stuck in plateaus and local maxima if no better moves exist. It stops prematurely.

13 Explain simulated annealing and write it algorithm

Ans Simulated annealing is optimization algorithm inspired by annealing process in metallurgy where are heated and once cooled slowly where structure. It determined to escape local optima by allowing occasional downhill moves solutions

Simulated annealing algorithm

① Initialize start an initial solution and initial temperature

Define cooling schedule (temperature reduction functions

② Repeat until stopping condition is met

Generate new solution by making small random changes to calculate change in cost

If is better () move

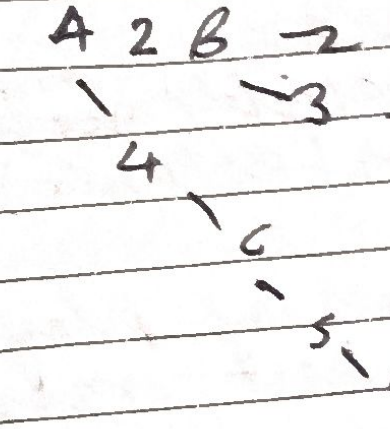
If is worse accept with probability

Teacher's Sign

Steepest ascent hill climbing.
Simple hill climbing only considers neighbouring node
Steepest ascent hill climbing examines all neighbouring nodes and selects one closest to solution state as next node
Attempts to maximize its distance from current state by selecting successive states that are progressively higher than the previous one
Stochastic hill climbing chooses at random from among the uphill moves; the probability of selection can vary with the steepness of the uphill move
Local maxima: No algorithm may stop at that is not global maximum because it only considers nearby solutions
Plateau - If a flat area it reached, algorithm has no direction is not, algorithm may struggle to find optimal path
Ridge and Values: If problem has ridges, algorithm may struggle to find optimal path
Termination issues: There is no way to determine if global maximum has been reached unless entire space is explored.
Limitations of steepest ascent hill climbing
Selects best node/move at each step

14 Explain A* algorithm with example.
A* is an informed search algorithm used for pathfinding and graph traversal. It finds the shortest path from start node to goal node by combining
(1) g(n) → The actual cost from start node to current node
(2) h(n) → Estimated cost from current node to goal (heuristic)
(3) f(n) = g(n) + h(n) → total estimated cost.

Example finding shortest path in a graph
Consider a graph where nodes represent cities
an edges represent travel distances
Graph representation. Let say we need to
find shortest path from A to G.



Step by step execution
Start at A calculate
f(n) = g(n) + h(n) Assume

2. Expand lowest f(n) node find

A → B or A → C Suppose h(C) = 0 (good node)

3. Continue exploring nodes with lowest f(n) until we reach G.

4. Find shortest path is determined once we reach with lowest cost.

15 Explain min max Explain algorithm and draw game tree for Tic Tac Toe game.

And the minimum algorithm is a decision making algorithm used in two player games like tic tac toe, chess and checkers. It helps find optimal moves by assuming that one Player. How min max works.

Generate game tree with all possible moves.

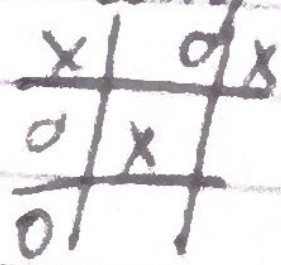
Assign scores +1 for G (max)

minimizing player picks minimum value of.

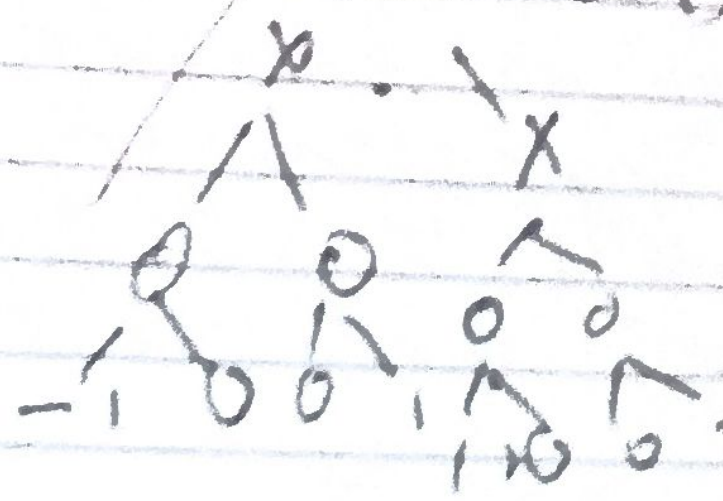
Choose best move based on root node value

Minimax algorithm is a decision making algorithm used in two player games like tic tac toe, chess and checkers where players take turns and each player's goal is minimize the loss and maximize the gain. Consider game tree with all possible moves (states) of tic tac toe and assign values (+1 for win, 0 for draw, -1 for loss) to leaf nodes. Backpropagate the values. Maximizing player picks maximum value & choose best move based on root node value.

Game tree for tic tac toe
Consider nearly complete tic tac toe board where it is X's turn to play



Root (X to move)



Maximizing player (X) chooses highest value from available branches.
Minimizing player (O) chooses lowest value

explain Minimax decision making algorithm for adversarial search with example
Minimax pruning is optimization for minimax algorithm used in adversial game
It prunes branches that do not need exploration, making minimax more efficient
- Alpha (α) : Best already guaranteed
- Beta (β) : Best min can guarantee
- If β ≤ α further branches are pruned (pruning)

Consider game tree with max and min levels,

max (3) ≥ min (2) ≤ Alpha pruning applied

Alpha starts at -∞, Beta at +∞
First branch (3,5) is explored → max chooses 3
Alpha (max) = 3 now all to second branch
Sees a value (2) less than 3 → Beta (min) = 2
Since Beta (2) < Alpha (3), pruning happens (remaining branches)

17 Explain WUMPUS world environment giving PEAS description. Explain how percept sequence is generated

→ WUMPUS world is partially observable, stochastic environment used in AI for planning and decision making
It consist of agent navigating the world
WUMPUS: A monster that kills the agent if encountered. It deadly traps. Goal

find and grab it
Walls boundaries that block.
PEAS description
Component | Description
P (Performance measure) | 1000 for grabbing gold
-100 for falling into
E (environment) | Put
9x9 grid with WUMPUS, Pits
A (actuators) | move forward, left right,
grab shoot.
S (sensor) | stench (near WUMPUS), Breeze
near (Pit), Glitter (near gold)

* Percept sequence generation
The percept sequence is series of observations
received by agent as it moves.
Agent starts at (1,1) perceives a breeze (if Pit is
moves to (1,2) senses a stench (if WUMPUS
is adjacent
Reach (2,2) sees glitter (if gold - present) grabs
it.
Navigates back to (1,0) exits the cave)
At each step, agent updates it knowledge
base and decides the next moves based on
logical inference.

Teacher's Sign.: _____

Solve the following cryptoarithmetic problem
SEND + MORE = MONEY

Ans The Cryptoarithmetic problem SEND+MORE=MONEY resolves assigning unique digits to each letter
Assigning values to Letters

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

Solution is
$$\begin{array}{r} 9567 \\ +\ 1085 \\ \hline 10652 \end{array}$$

19 Consider following axioms
All people who are graduating are happy.
Someone is graduating
Explain the following
① Represent these axioms in first predicate logic
② Convert each formula to clause form
③ Prove that "If someone is smiling using resolution technique
Draw the resolution tree

Ans First Order Predicate logic representation
Let
G(x): x is Graduating
H(x): x is happy
S(x) is smiling
Axiom
$\forall x (G(x) \rightarrow H(x))$ Graduating people are happy.

4 ∀x (H(x) → S(x))
5) ∃x S(x) (Someone is graduating)

2) Convert to clause form
1) remove implications
∀x (¬H(x) → ¬G(x)) becomes ¬G(x) ∨ H(x)
∀x (H(x) → S(x)) becomes ¬H(x) ∨ S(x)
∃x S(x) introduces Skolem constant (let x = a, since someone is graduating)
3) standardized clause form
C1 : ¬G(x) ∨ H(a)
C2 ¬H(x) ∨ S(a)
C3: G(a) Someone is graduating

3) Prove "Someone is smiling" using resolution
Resolve C1 and C3
C1 ¬G(a) ∨ H(a)
C3. G(a)
Resolution H(a)
Resolve G(a) with C2
Resolution S(a) since is derived, we have proved. that someone is smiling

Solution tree → S(a) → G(a) ∨ H(a)

S(a)

¬H(a)

H(a)

¬S(a) ⊥

Explain modus ponens with suitable example
Modus ponens (Law of detachment) is a fundamental rule of inference in logic and statistics

If
1 $P \rightarrow q$ (If P, then q)
2 P (P is true)
then we can conclude
q (q must be true)

Example
1 If it rains, the grass will be wet ($P \rightarrow q$)
2 It is raining (P)
3 Therefore the ground is wet (q)
This rule is widely used in mathematical proofs, AI reasoning and programming logic

a) Explain forward chaining and backward chaining algorithm with help of example. Both are inference techniques used in rule based systems and AI.

- Forward Chaining (Data driven Reasoning) starts from known facts and applies rules to derive new facts until the goal is reached.
- Used in expert systems, AI planning and production systems.

example rules
- If it rains (R) then ground is wet (W → R)
- If the ground is wet (W) then grass is slippery (W → S)

Given KB is Tuesday
Apply R → W to see if ground is wet is inferred
Apply W → S to (both is supported to infer)

2 Backward Chaining (goal driven reasoning)
- starts from and works backwards to see if facts support it

example goal is grass slippery (S)?

① check S → W (Grass is slippery if ground is wet)
② check W → R (Ground is where it rained
③ If R (rain) is found is known fact, then it is true

NAME HIMANSHU NAIK **Roll no - 63** **D15C**

AIDS-I Assignment No: 2

Q.1: Statistical Analysis of the Dataset

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

ANS - Mean (10 pts): The mean is calculated by summing all values and dividing by the total number of values. Mean = (82 + 66 + 70 + 59 + 90 + 78 + 76 + 95 + 99 + 84 + 88 + 76 + 82 + 81 + 91 + 64 + 79 + 76 + 85 + 90) / 20 = 1601 / 20 = **80.05**

Median (10 pts): Arrange the data in ascending order: 59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99 Median = (10th value + 11th value) / 2 = (81 + 82)/2 = **81.5**

Mode (10 pts): The most frequent number is 76, which appears 3 times. Mode = **76**

Interquartile Range (20 pts): Q1 (25th percentile) = median of first half = (5th + 6th)/2 = (76 + 76)/2 = 76 Q3 (75th percentile) = median of second half = (15th + 16th)/2 = (88 + 90)/2 = 89 IQR = Q3 - Q1 = 89 - 76 = **13**

Q.2: Comparison of Machine Learning Tools for Kids

ANS - 1. Machine Learning for Kids

- **Target Audience**: School-aged children (ages 8–16), educators
- **Use by Audience**: Allows students to create ML projects using image, text, or number-based data. Integrated with Scratch and Python.
- **Benefits**: Intuitive, easy to use, free, integrates with existing educational platforms.
- **Drawbacks**: Limited model complexity; less suitable for advanced users.
- **Analytic Type**: **Predictive Analytic** - as it uses trained models to make predictions.
- **Learning Type**: **Supervised Learning** - users provide labeled examples for training.

2. **Teachable Machine**

- **Target Audience**: Beginners, educators, artists, and developers.
- **Use by Audience**: Users can train models using images, audio, or poses through a webcam/microphone.
- **Benefits**: No coding required, real-time interaction, fast model training.
- **Drawbacks**: Models may overfit, not suitable for large-scale applications.
- **Analytic Type**: **Predictive Analytic** - it predicts based on real-time input.
- **Learning Type**: **Supervised Learning** - trained using labeled input data.

Q.3: Misinformation in Data Visualization

ANS - Data visualization is a powerful tool to convey complex information quickly and intuitively. However, when misused—intentionally or unintentionally—it can lead to serious misinformation. This often happens during crises such as the COVID-19 pandemic, where public perception directly influences behavior and policy.

Key Articles and Insights:

1. Arthur Kakande – "What's in a chart?" (Medium)
    - This article provides a step-by-step guide to identifying misleading charts. It highlights common red flags such as:
        - Truncated or manipulated axes
        - Misleading use of color
        - Unlabeled data points or scales
        - Charts without context
    - The goal is to teach readers how to critically evaluate visual data, emphasizing that beautiful graphics can still be deceptive.

2. Katherine Ellen Foley – "How bad Covid-19 data visualizations mislead the public" (Quartz)
    - The article discusses real examples where poor design choices contributed to public misunderstanding.
    - It stresses the responsibility of media and data scientists in avoiding:
        - Overuse of cumulative data (which hides trends)
        - Inappropriate chart types (e.g., pie charts for time series)
        - Inconsistent scales and color coding

Example of Real-World Misinformation:

Case Study: Misleading COVID-19 Charts in U.S. States During the early months of the pandemic, some state governments (notably Georgia and Florida) published bar graphs of COVID-19 cases where:

- The x-axis (dates) was out of order (e.g., jumping from May 7 to May 4 and back to May 6).
- The colors representing counties were reused inconsistently.
- Y-axes were truncated, making trends appear flat even as case numbers surged.

Why This Was Misleading:

- The non-chronological order gave the illusion of declining cases.
- Truncated y-axes minimized visual differences, suggesting improvement
- Inconsistent color mapping confused which regions were most affected.

Impact:

- These visualizations gave citizens and policymakers a false sense of security.
- Some interpreted the misleading visuals as evidence that restrictions could be lifted safely.
- Public health experts criticized these charts as dangerous.

Better Practices:

- Always keep axes clearly labeled and in proper order.
- Avoid truncated y-axes unless justified—and always indicate truncation.

- Use consistent and meaningful color schemes.
- Present both raw and per-capita numbers for clearer comparison.

Cited Source:

- The New York Times, "The Worst Charts of the Pandemic—and How We Can Do Better" https://www.nytimes.com/2020/07/02/upshot/coronavirus-data-charts.html

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

- Data File: Classification data.csv
- Class Label: Last Column
- Use any Machine Learning model (SVM, Naïve Base Classifier)

Requirements to satisfy

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

ANS -

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Features where 0 is invalid and should be considered missing
cols_with_zero_invalid = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

# Replace 0s with NaN
df[cols_with_zero_invalid] = df[cols_with_zero_invalid].replace(0, np.nan)

# Impute missing values with median
df.fillna(df.median(), inplace=True)

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

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-Validation-Test Split
X_train_val, X_test, y_train_val, y_test = train_test_split(X_scaled, y, test_size=0.10, random_state=42, stratify=y)
X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val, test_size=2/9, random_state=42, stratify=y_train_val)  # (2

# Final shapes
print("Train set:", X_train.shape)
print("Validation set:", X_val.shape)
print("Test set:", X_test.shape)
```

```
Train set: (537, 8)
Validation set: (154, 8)
Test set: (77, 8)
```

```
[7] from imblearn.over_sampling import SMOTE
    from collections import Counter
```

NAME HIMANSHU NAIK **Roll no - 63** **D15C**

```
from imblearn.over_sampling import SMOTE
from collections import Counter

# Apply SMOTE only on the training data (NOT on validation or test)
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Check the class distribution before and after
print("Before SMOTE:", Counter(y_train))
print("After SMOTE:", Counter(y_train_resampled))
```

```
Before SMOTE: Counter({0: 350, 1: 187})
After SMOTE: Counter({0: 350, 1: 350})
```

```
print("\nMissing values:\n", df.isnull().sum())
```

```
Shape of dataset: (768, 9)

Data types:
 Pregnancies                 int64
Glucose                      int64
BloodPressure                int64
SkinThickness                int64
Insulin                      int64
BMI                        float64
DiabetesPedigreeFunction   float64
Age                          int64
Outcome                      int64
dtype: object

Sample data:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1

Missing values:
 Pregnancies                 0
Glucose                      0
BloodPressure                0
SkinThickness                0
Insulin                      0
BMI                          0
DiabetesPedigreeFunction     0
Age                          0
Outcome                      0
dtype: int64
```

```
best_svm = grid_search.best_estimator_

# Evaluate on validation set
y_val_pred = best_svm.predict(X_val)
print("🔷 SVM Validation Accuracy:", accuracy_score(y_val, y_val_pred))
print("🔷 Classification Report:\n", classification_report(y_val, y_val_pred))
```

```
🔷 SVM Validation Accuracy: 0.6948051948051948
🔷 Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.73      0.76       100
           1       0.56      0.63      0.59        54

    accuracy                           0.69       154
   macro avg       0.67      0.68      0.67       154
weighted avg       0.71      0.69      0.70       154
```

+ Code + Text

```
[10] from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()
nb.fit(X_train_resampled, y_train_resampled)

# Evaluate on validation set
y_val_pred_nb = nb.predict(X_val)
print("🔷 Naive Bayes Validation Accuracy:", accuracy_score(y_val, y_val_pred_nb))
print("🔷 Classification Report:\n", classification_report(y_val, y_val_pred_nb))
```

```
🔷 Naive Bayes Validation Accuracy: 0.7662337662337663
🔷 Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.80      0.82       100
           1       0.66      0.70      0.68        54

    accuracy                           0.77       154
   macro avg       0.74      0.75      0.75       154
weighted avg       0.77      0.77      0.77       154
```

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

- Data File: Regression data.csv
- Independent Variable: 1st Column
- Dependent variables: Column 2 to 5

Use any Regression model to predict the values of all Dependent variables using values of 1st column.
Requirements to satisfy:

- Programming Language: Python
- OOP approach must be followed
- Hyper parameter tuning must be used
- Train and Test Split should be 70/30
- Train and Test split must be randomly done
- Adjusted R2 score should more than 0.99
- Use any Python library to present the accuracy measures of trained model

ANS -

```
# Split the data into training and testing sets
regression_model.split_data()

# Tune hyperparameters and train the best model
regression_model.tune_and_train_model()

# Evaluate the model and print the Adjusted R² score
adj_r2 = regression_model.evaluate_model()
```

```
Best parameters from GridSearch: {'poly__degree': 2, 'ridge__alpha': 10}
R²: 0.6492937127071428
Adjusted R²: 0.6314611896244551
```

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).

ANS - Key Features:

- Fixed acidity, Volatile acidity, Citric acid, Residual sugar, Chlorides, Free sulfur dioxide, Total sulfur dioxide, Density, pH, Sulphates, Alcohol

- Importance:
  - Alcohol: Most correlated with quality (higher alcohol, better quality)
  - Volatile Acidity: Negative correlation (lower acidity, better quality)
  - Sulphates and Citric Acid: Add to flavor; positively correlated
- Handling Missing Data:
  - Technique Used: Mean/median imputation for numerical columns
  - Alternatives: KNN Imputation, model-based imputation
- Advantages and Disadvantages:
  - Mean/Median: Easy, fast; may reduce variability
  - KNN: Captures local structure; computationally expensive
  - Model-based: More accurate; risk of bias, overfitting
- Dataset Source: Kaggle Wine Quality Dataset (https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009)