APPLIED DATA SCIENCE

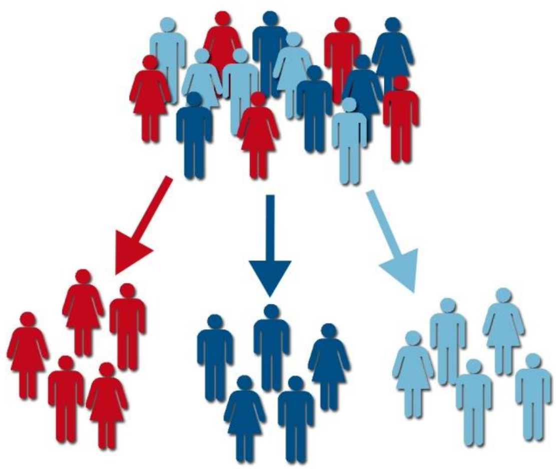
IBM NAAN MUTHALVAN PHASE 3

TEAM MEMBERS:

- **PRIYADHARSHINI.E**
- **KEERTHANA.S**
- **GAYATHRI.B**
- **DHARSNI RITIKA .KG**

PROJECT TITLE:

CUSTOMER SEGMENTATION USING DATA SCIENCE.



DEVEOPMENT PART:

- In this part you will begin building your project by loading and preprocessing the dataset.
- Begin the customer segmentation project by loading and preprocessing the customer data.
- Collect and preprocess the customer data for analysis.

SOURCE   TOOLS:

- Data Loading.
- Data preprocessing.
- Data Set Explanation.

- Building the Project by Load The Data Set.
- Preprocess Data Set.
- Different Analysis Needs.

DATA LOADING:

- Ensure your data is in a structured format like CSV, Excel, or a database.
- Use appropriate libraries in your programming language (e.g., pandas in Python) to load the data into a Data Frame or any suitable data structure.

Exploring the Data:

- Check the first few rows to understand the structure and format of the data.
- Investigate the data types of each column (numerical, categorical, etc.).
- Look for missing values, outliers, or any inconsistencies in the data.

Data Cleaning:

- Handle missing values by removing or imputing them based on the context.
- Address outliers either removing them or transforming them to be within an acceptable range.

Splitting the Data:

- Split the data into training and testing sets to evaluate the model's performance.

Example in Python using the pandas library:

```
python
import pandas as pd

# Load data from a CSV file
customer_data = pd.read_csv('customer_data.csv')

# Explore the data
print(customer_data.head()) # Display the first few rows
print(customer_data.info()) # Get data types and check for missing values
```

DATA PREPROCESSING:

```
# Data preprocessing steps go here (handling missing values.)

# Split data into features (X) and target variable (y)
```

```
X = customer_data.drop('target_column', axis=1) # Features
y = customer_data['target_column'] # Target variable

# Split data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42).
```

DATASET EXPLANATION:

- In the context of customer segmentation, a dataset typically consists of various attributes or features related to customers.
- These features can include demographic information (such as age, gender, Spending score , Annual income).

Customer segmentation involves dividing the customer base into distinct groups based on similarities in these attributes.

For example, Dataset for customer segmentation, you might have the following columns:

- Customer ID : A unique identifier for each customer.
- Age : Age of the customer.
- Gender : Gender of the customer (male, female, other).
- Annual Income : Income level of the customer.
- Spending Score: score of purchasing.

Dataset Link: https://www.kaggle.com/datasets/akram24/mall-customers

BUILDING THE PROJECT BY LOAD THE DATASET.

Import Libraries.

```
python
import pandas as pd
```

Load the Dataset.

```
# Assuming your dataset is in a CSV file
dataset = pd.read_csv('your_dataset.csv')
```

Explore the Data.

```
# Print the first few rows of the dataset to understand its structure
  print(dataset.head())
```

python program,

```
import pandas as pd

# Load the dataset from a CSV file
file_path = 'path/to/your/dataset.csv' # Replace this with the actual file path
dataset = pd.read_csv(file_path)

# Now 'dataset' holds your data and you can start working with it
print(dataset.head())
 # Display the first few rows to verify the data has been loaded correctly
```

In the code above:

- `pd.read_csv(file_path)` loads the CSV file into a pandas DataFrame.
- `print(dataset.head())` displays the first few rows of the dataset, allowing you to check if the data is loaded correctly.

PREPROCESS DATASET:

1. Handling Missing Values.

- Identify and handle missing values in the dataset.
- You can either remove rows with missing values or fill them using techniques like mean, median, or interpolation, depending on the context.

```
# Remove rows with missing values
    dataset_clean = dataset.dropna()

# Or fill missing values with mean
    dataset_filled = dataset.fillna(dataset.mean())
```

2. Encoding Categorical Variables.

- Convert categorical variables into numerical representations. One-hot encoding creates binary columns for each category,
- while label encoding assigns a unique number to each category.

```
# One-hot encoding
```

```
dataset_encoded = pd.get_dummies(dataset, columns=['categorical_column'])

# Label encoding
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
dataset['categorical_column']=label_encoder.fit_transform(dataset['categorical_column'])
```

3.Feature Scaling.

- Scale numerical features to a standard range (e.g., using Min-Max scaling or Z-score normalization) to ensure they have a similar scale.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
dataset_scaled = scaler.fit_transform(dataset[['numeric_column1',
'numeric_column2']])
```

4. Handling Outliers.

- Identify and handle outliers in numerical features using statistical methods or visualization techniques.
- You can remove outliers or transform them to be within an acceptable range.

5. Feature Engineering (Optional).

.

- Create new features based on existing ones to enhance the model's performance.
- This could involve operations like combining features, extracting relevant information, or creating interaction terms.

```
# Creating a new feature by combining existing features
dataset['new_feature'] = dataset['feature1'] * dataset['feature2']
```

6. Data Splitting.

- Split the data into features (X) and the target variable (y) for model training and testing purposes.

```
X = dataset.drop('target_column', axis=1)
y = dataset['target_column']
```

Different Analysis Needs:

1.Descriptive Statistics.

- Calculate basic statistics such as mean, median, standard deviation, minimum, maximum, and percentiles.

- This helps in understanding the central tendency and spread of numerical features in your dataset.

```
# Descriptive statistics
print(dataset.describe())
```

2.Data Visualization.

- Visualize our data using charts and graphs.
- Common plots include histograms, box plots, scatter plots, and bar charts.
- Visualization provides an intuitive understanding of the data distribution and relationships between variables.

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

# Example: Histogram
plt.hist(dataset['numeric_column'])
plt.xlabel('Numeric Column')
plt.ylabel('Frequency')
plt.show()

 # Example: Scatter plot
plt.scatter(dataset['feature1'], dataset['feature2'])
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

3.Correlation Analysis.

- Compute the correlation between numerical variables to identify relationships.
- Positive and negative correlations indicate the direction of the relationship.

```
# Correlation matrix
correlation_matrix = dataset.corr()
print(correlation_matrix)
```

4.Hypothesis Testing.

- Use statistical tests (t-tests, ANOVA, etc.) to test hypotheses about your data.
- For example, you might want to test if the means of two groups are significantly different.

```
from scipy.stats import ttest_ind
```

```
group1 = dataset[dataset['group'] == 1]['target_column']
group2 = dataset[dataset['group'] == 2]['target_column']

t_stat, p_value = ttest_ind(group1, group2)
print(f'T-statistic: {t_stat}, p-value: {p_value}')
```

5.Predictive Modeling.

- ❖ Build machine learning models to predict the target variable based on the features.
- ❖ Evaluate the model's performance using metrics like accuracy, precision, recall, or mean squared error, depending on the type of problem (classification or regression).

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)

predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f'Accuracy: {accuracy}')
```

6. Clustering Analysis.

- ❖ Apply clustering algorithms like K-means to group similar data points together.
- ❖ This is useful for discovering patterns or segments within your data.

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3).
clusters = kmeans.fit_predict(X).
```

PROGRAM :

```
# Importing necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Load your dataset (replace 'data.csv' with your dataset file)
data = pd.read_csv('data.csv')

# Handling missing values (replace 'column_name' with the appropriate column name)
data['column_name'].fillna(data['column_name'].mean(), inplace=True)

# Encoding categorical variables using one-hot encoding
data = pd.get_dummies(data, columns=['categorical_column'])

# Feature scaling using StandardScaler (replace 'feature_column' with the appropriate
column name)
scaler = StandardScaler()
data['feature_column'] = scaler.fit_transform(data['feature_column'].values.reshape(-1,
1))

# Splitting the data into features and target variable
X = data.drop('target_column', axis=1) # Features
y = data['target_column'] # Target variable

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

# Now you can use X_train, X_test, y_train, y_test for your machine learning model
training.
```

CONCLUSION:

- In conclusion, data preprocessing is a vital step in the field of data science. It involves cleaning, transforming, and organizing raw data into a format suitable for analysis.
- Proper data preprocessing ensures that the data used for analysis is accurate, consistent, and relevant, leading to more meaningful insights and better decision-making.
- Key steps in data preprocessing include handling missing values, encoding categorical variables, scaling features, and splitting the data into training and testing sets.
- These steps prepare the data for machine learning algorithms, enabling the development of accurate predictive models.
- It's important to note that the specific preprocessing techniques used may vary based on the nature of the data and the requirements of the analysis or modeling task.
- Regular exploration and understanding of the data are essential for choosing the most appropriate preprocessing methods.