DON BOSCO INSTITUTE OF TECHNOLOGY
Bengaluru, Karnataka – 74

TEAM COUNT: THREE

PROJECT TITLE: Customer journey analysis using clustering and dimensionality reduction enhancing user experience

TEAM LEAD NAME: AFTAF AYUB MANIYAR

TEAM LEAD CAN ID: CAN_33697094

1) NAME: AFTAF A MANIYAR
CAN ID: CAN_33697094
ROLE: MACHINE LEARNING

2) NAME: HARSHA N A
CAN ID: CAN_33738478
ROLE: DATA ANALYST

3) NAME: RAKSHITH K L
CAN ID: CAN_33116494
ROLE: PROJECT MANAGER

4)GOKUL KANNAN P
CAN ID:CAN_33196444
ROLE:INFORMATION

Advanced Market Segmentation Using Deep Clustering

Phase 2: Data Preprocessing and Model Design

2.1 Overview of Data Preprocessing

After completing the initial data exploration in Phase 1, Phase 2 focuses on preparing the dataset for deep learning. This involves cleaning, transforming, and scaling the data to ensure it is suitable for training the deep autoencoder model. The primary goal of this phase is to handle missing values, outliers, and data inconsistencies, and to apply appropriate transformations such as feature scaling, encoding, and dimensionality reduction.

2.2 Data Cleaning: Handling Missing Values, Outliers, and Inconsistencies

Cleaning the dataset is a critical step to ensure that the input data is accurate and ready for modeling. In this phase, we address the following issues:

- **Missing Values**: Missing values can cause models to fail or generate biased results. For this project, missing data were identified using statistical methods such as descriptive statistics (mean, median) and visualization techniques like heatmaps. The following strategies were employed to handle missing values:

- **Outliers**: Outliers can significantly skew model results, especially for algorithms that are sensitive to extreme values. For this project, outliers were detected using visualization methods such as boxplots and statistical methods like the **Z-score**.
- **Inconsistencies**: Inconsistencies within the data, such as duplicate entries or contradictory information (e.g., age and income), were also cleaned. Duplicate rows were identified and removed, and any contradictory entries were flagged for further review or corrections.

Methods you used to pre process the data

Coding of your project for preprocessing with Screen Shots

2.3 Feature Scaling and Normalization

Scaling the features ensures that they are comparable in magnitude, which is particularly important for deep learning models like autoencoders.

Steps for normalization an scaling with coding and its screen shots

2.4 Feature Transformation and Dimensionality Reduction

Transforming features helps improve the performance of the deep learning model by reducing noise or irrelevant information and highlighting important patterns. This phase also includes applying dimensionality reduction techniques to handle high-dimensional data.

- **Dimensionality Reduction**: Given the high number of features in the dataset, it was important to reduce the dimensionality to speed up the training process and prevent overfitting. Several dimensionality reduction techniques were considered:
  - **Principal Component Analysis (PCA)**: PCA was applied to reduce the dimensionality of the dataset while retaining the maximum amount of variance. This helped remove multicollinearity between highly correlated features and provided a more compact representation of the data.
  - **Feature Selection**: Based on the initial analysis, redundant features that provided minimal value to the clustering process were removed. This was done by analyzing feature importance or using techniques like **Variance Thresholding**, which eliminates features with low variance across samples.

Importance of dimensionality reduction in your project

Source code of DR with screenshots

2.5 Autoencoder Model Design

With the data cleaned and transformed, we now turn to the model design. The focus in this project is on using an **autoencoder** for deep clustering. Autoencoders are unsupervised neural networks that learn to represent input data in a compressed latent space. The architecture of the autoencoder was designed as follows:

- **Encoder Architecture**: The encoder takes the preprocessed and transformed data as input and compresses it into a latent feature space. The encoder has the following layers:
  - An input layer that takes the feature vector.
  - Several dense layers with progressively decreasing units, such as 64 neurons in the first hidden layer, followed by 32 neurons, and the final latent layer with 8 neurons.
- **Decoder Architecture**: The decoder mirrors the encoder, expanding the latent features back into the original feature space:
  - Dense layers with progressively increasing units to reconstruct the original input.
  - The output layer uses the **sigmoid activation function** to ensure the reconstructed values are between 0 and 1, which matches the scaled features.
- **Loss Function and Optimizer**: The model uses **Mean Squared Error (MSE)** as the loss function since the goal is to minimize the reconstruction error between the input data and the model's reconstruction. The **Adam optimizer** was chosen for efficient gradient-based optimization.

2.6 Model Training and Validation

After designing the autoencoder architecture, the next step was training the model. The data was split into **training** and **validation** sets to evaluate the model's performance. The model was trained for **50 epochs** with a batch size of 32.

During training, the model’s reconstruction error was monitored on both the training and validation datasets to ensure that the model was generalizing well and not overfitting. Hyperparameters, such as the learning rate and batch size, were fine-tuned to achieve optimal performance.

Once trained, the encoder was used to extract the **latent features**, which represent the compressed version of the original data. These features will be used for the clustering phase in the next step.

Code for the preprocessing, training, epochs with its screenshots

2.7 Conclusion of Phase 2

Phase 2 has focused on preparing the dataset for deep learning by cleaning the data, handling missing values and outliers, and transforming the features. Scaling and encoding were applied to ensure that the data was ready for the auto encoder model. Dimensionality reduction techniques like PCA helped improve model efficiency by reducing noise and redundancy. With the model designed and trained, the latent features are now ready for clustering, which will be the next step in the segmentation process. This phase has established a strong foundation for the clustering and segmentation in subsequent phases of the project.