PHASE 2

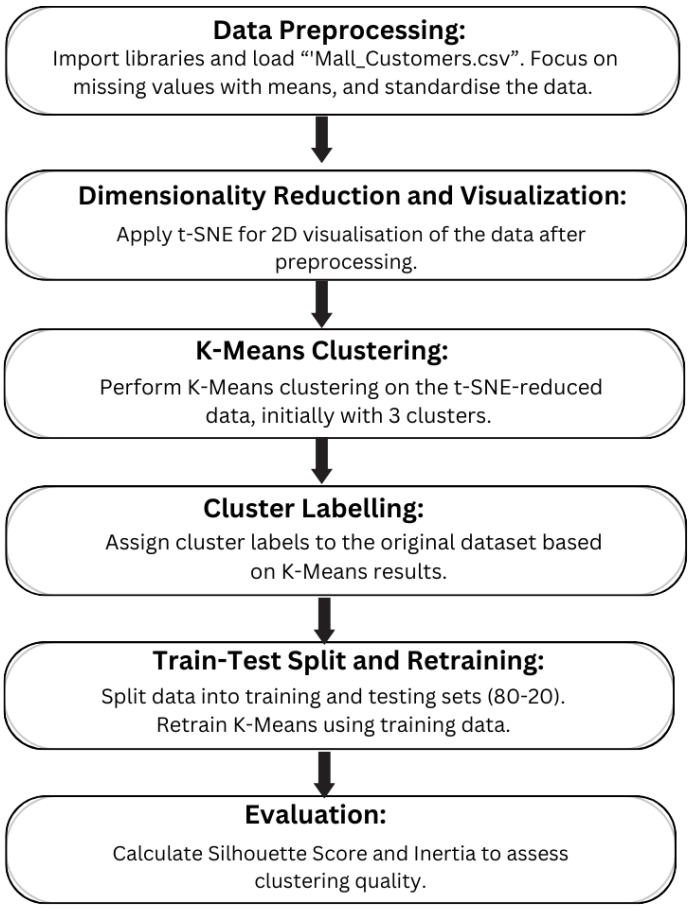
Customer Segmentation using Data Science

Thoufeeq A -71772117146
Sudharsan M-71772117144
Sandhiya S -71772117138
Srikanth B -71772117143

DESCRIPTION:

This document outlines the methodology and steps to develop a data science model for customer segmentation. The main goal of this initiative is to assist businesses in gaining deeper insights into their customer base and tailoring marketing strategies to match each segment's unique characteristics.

PROJECT FLOW:



DATASET LINK:

The dataset employed in this analysis originates from Kaggle, a renowned platform for data science competitions and datasets. You can access the specific dataset focused on analysing and predicting mall customer behaviour at
https://www.kaggle.com/datasets/akram24/mall-customers

COLUMNS:

The columns are typically found in a dataset often used for customer segmentation, profiling, and analysis, particularly in the context of a mall or retail business.

1. **CustomerID**:
This column typically represents a unique identifier for each customer. Each customer is assigned a distinct ID that can be used to track and differentiate them in the dataset..

2. **Genre (or Gender):**
The Genre column indicates the gender of each customer, such as 'Male' or 'Female'. Gender can be valuable for understanding customer demographics and tailoring marketing strategies.

3. **Age**:
The Age column represents the age of each customer. Age is a crucial demographic factor as it can influence customer preferences, purchasing behaviour, and product choices.

4. **Annual Income**:
Annual Income represents the annual income of each customer in thousands of dollars. It's a key factor for understanding the spending potential and purchasing power of different customer segments.

5. **Spending Score (1-100)**:
The Spending Score, ranging from 1 to 100, categorises customers based on spending behaviour, considering factors like purchase amounts, visit frequency, and product types. Higher scores indicate more active or higher-spending customers.

COLUMNS USED:

The 'Spending Score (1-100)' column drives clustering, which is based on customer spending behaviour using K-Means.

Utilising t-SNE, the code reduces the feature space to two dimensions for visualisation. K-Means then clusters data into three groups, labelling them as 'Cluster.' The dataset is split into training and testing sets, and K-Means is restrained on the training data. Cluster labels for the testing data are predicted.

Finally, the Silhouette Score assesses clustering quality, favouring higher scores for better cluster separation and cohesion.

LIBRARIES USED:

1.**Pandas (pd):**

Pandas is a powerful library for data manipulation and analysis. It provides data structures and functions to work with structured data, such as tables or spreadsheets.

2.**NumPy (np):**

NumPy is a fundamental library for numerical computations in Python. It provides support for large, multi-dimensional arrays and matrices, as well as a wide range of mathematical functions.

3.**Matplotlib (plt):**

Matplotlib is a widely-used library for creating static, animated, and interactive visualisations in Python. It provides a variety of plotting functions and customization options.

4.**Scikit-learn (sklearn)**:

Scikit-learn is a machine learning library for Python. It provides a wide range of tools for data mining and data analysis. In your code, you're using specific modules from scikit-learn.

5.**StandardScaler (from sklearn.preprocessing):**

StandardScaler is a preprocessing technique provided by scikit-learn for standardising features by removing the mean and scaling to unit variance. It's often used to prepare data for machine learning models.

6.**TSNE (from sklearn.manifold):**

t-SNE (t-distributed Stochastic Neighbour Embedding) is a dimensionality reduction technique used for visualising high-dimensional data. It's often used for exploratory data analysis and visualisation.

7.**SimpleImputer (from sklearn.impute):**

SimpleImputer is a scikit-learn tool for handling missing data by imputing or filling in missing values with a specified strategy, such as mean, median, or a constant value.

8.**KMeans (from sklearn.cluster):**

KMeans is a clustering algorithm provided by scikit-learn. It's used to partition data into clusters based on similarity, with the number of clusters specified by the user.

Installation:

To install the libraries,use the pip command followed by the library name in the terminal.

TRAIN AND TEST:

Import the Required Libraries:

The train_test_split function is imported from sklearn.model_selection.

Load and Preprocess the Dataset:

Load the dataset and perform preprocessing steps, including handling missing values and standardising the data.

Choose a Split Ratio:

Determine the data split ratio for training and testing. Common ratios are 70-30, 80-20, or 90-10. In this code, an 80-20 split is chosen, allocating 80% for training and 20% for testing.

Perform the Train-Test Split:

Utilise the train_test_split function to divide the dataset into training and testing sets. This function requires:

- X: The feature matrix (independent variables).
- y (optional): The target variable (in this case, cluster labels).
- Test_size: The proportion (set to 0.2 for 20%).
- Rrandom_state (optional): A seed for reproducibility.

Obtain Training and Testing Data:

After the split, obtain X_train and X_test datasets. In this case, there are no specific y_train and y_test sets as clustering uses cluster labels.

Training and Testing:

Train the machine learning model using the X_train dataset. For K-Means clustering, retrain the model using the training data. Utilise the X_test dataset to evaluate the model's performance. Predict cluster labels for the testing data and assess clustering quality using metrics like Silhouette Score and Inertia.

Model Evaluation:

Evaluate the model's performance on the testing data to determine generalisation and prevent overfitting. This step is crucial for ensuring the model performs well on unseen data.

METRICS USED FOR ACCURACY CHECK:

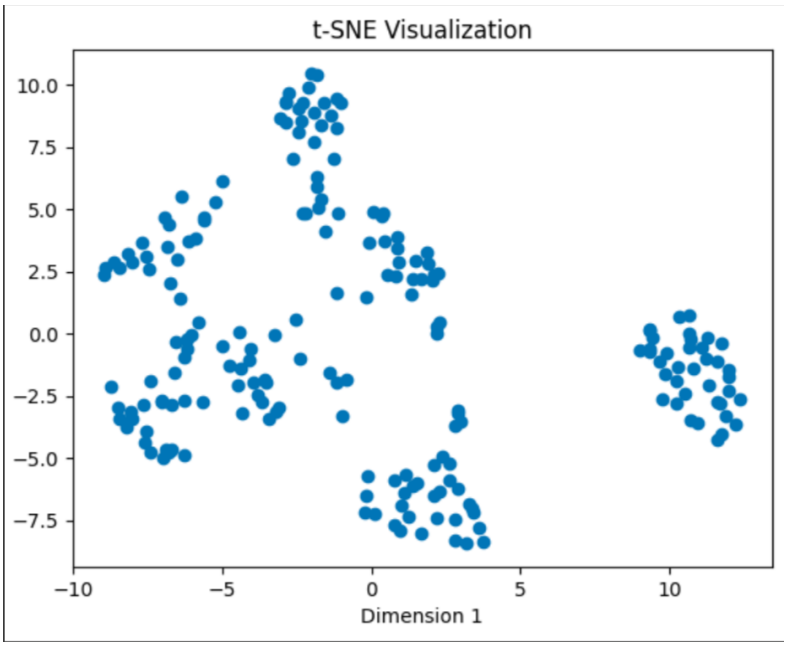
Two vital metrics are utilised for evaluation:

1. **Silhouette Score:**
   - Measures clustering quality by evaluating cluster separation and data point similarity within clusters.
   - Higher score signifies well-defined, appropriately assigned clusters, calculated for testing data.

2. **Inertia**:
   - Gauges cluster compactness by summing squared distances to cluster centres.
   - Lower values indicate more compact clusters, computed for both training and testing data.

These metrics aid in assessing clustering performance and guiding decisions regarding cluster number and algorithm fine-tuning.

OUTPUT:



Silhouette Score: 0.4522377848625183
Inertia for Training Data: 2713.7197265625
Inertia for Testing Data: 2713.7197265625