Project:Customer segmentation using data science

Detailed explanation of dataset:

A customer segmentation dataset is a structured collection of information about your customers that is used for the purpose of dividing your customer base into distinct groups based on similar characteristics, behaviors, or preferences. This segmentation is essential for businesses to understand their customers better and tailor their marketing strategies, product offerings, and customer interactions to specific customer segments. Here's an explanation of some common variables that might be included in a customer segmentation dataset and their significance:

1. **Demographic Information:**
   - **Age, Gender, Education, Marital Status:** Helps in demographic segmentation, which categorizes customers based on personal attributes.
   - **Income:** Provides insights into the purchasing power of customers.
2. **Geographic Information:**
   - **Location, City, State, Country:** Useful for geographical segmentation, which divides customers based on their geographical location.
3. **Behavioral Data:**
   - **Purchase History, Frequency, Monetary Value:** Enables RFM (Recency, Frequency, Monetary) segmentation, which focuses on the recency, frequency, and value of customer purchases.
   - **Website Visits, App Usage:** Indicates customer engagement with online platforms.
   - **Product Preferences:** Identifies the types of products or services customers prefer.
   - **Returns/Refunds:** Indicates customer satisfaction and potential issues with products.
4. **Customer Interactions:**
   - **Customer Support Interactions:** Measures customer service engagement.
   - **Social Media Engagement:** Tracks interactions and engagement on social media platforms.
5. **Customer Feedback and Satisfaction:**
   - **Customer Feedback Score:** Numerical or qualitative measure of customer satisfaction.
   - **Reviews and Ratings:** Customer opinions about products or services.
6. **Membership or Loyalty Program Status:**
   - **Membership Tier:** Indicates the level of loyalty and engagement.
   - **Benefits Usage:** Tracks the utilization of loyalty program benefits.
7. **Churn Status:**

- **Active, At-Risk, Churned:** Helps identify customers at risk of leaving or those who have already left.

Each variable in the dataset provides valuable insights into different aspects of customer behavior and characteristics. By analyzing and clustering this data using various data science techniques, businesses can identify patterns and trends within each segment. For example, one segment might consist of young, urban customers who frequently make online purchases, while another segment might include older customers who prefer in-store shopping.

Understanding these segments allows businesses to create targeted marketing campaigns, improve customer service, and optimize product offerings, leading to increased customer satisfaction and loyalty. Customer segmentation datasets, when properly analyzed, provide actionable insights that drive strategic decision-making for businesses.

Implementation of dataset:

Implementing customer segmentation involves several steps using programming languages like Python and popular libraries such as Pandas and Scikit-Learn. Below is a simplified example of how you might implement customer segmentation using Python:

1.Import Libraries:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

2. Load and Preprocess the Data:

# Load your customer segmentation dataset (assuming it's in a CSV file)

data = pd.read_csv('customer_data.csv')

# Select relevant features for segmentation

features = data[['Age', 'Income', 'PurchaseFrequency', 'TotalPurchaseAmount']]

# Standardize features to have mean=0 and variance=1

scaler = StandardScaler()

scaled_features = scaler.fit_transform(features)

3. Determine Optimal Number of Clusters (K):

```
# Use the Elbow Method to find the optimal number of clusters
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(scaled_features)
    wcss.append(kmeans.inertia_)

# Plot the Elbow graph to find the optimal K
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Within-Cluster Sum of Squares')
plt.title('Elbow Method for Optimal K')
plt.show()
```

4.Perform K-Means Clustering:

# Based on the Elbow graph, choose the optimal number of clusters (K)

k = 3 # Example: Assume optimal K is 3

# Perform K-Means clustering

kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300, n_init=10, random_state=0)

clusters = kmeans.fit_predict(scaled_features)

```
# Add cluster labels to the original dataset

data['Cluster'] = clusters
```

5. **Analyzing and Visualizing Segments:**

```
# Analyze the segments

segment_data = data.groupby('Cluster').mean()



# Visualize the segments

for i in range(k):

    segment = segment_data.loc[i]

    print(f'Segment {i}: {segment}')



# Visualization (example: Age vs. Income)

plt.figure(figsize=(8, 6))

for i in range(k):

    cluster_data = data[data['Cluster'] == i]

    plt.scatter(cluster_data['Age'], cluster_data['Income'], label=f'Segment {i}')

plt.xlabel('Age')

plt.ylabel('Income')

plt.legend()

plt.title('Customer Segmentation: Age vs. Income')

plt.show()
```

In this example, the K-means algorithm is used for clustering, and the optimal number of clusters (K) is determined using the Elbow Method. The data is then clustered, and segments are analyzed and visualized. Note that you should adapt this code to fit your specific dataset and requirements. Also, ensure you have the necessary data preprocessing steps and handle missing or categorical data appropriately for your use case.

Program to load the dataset:

```
import pandas as pd



# Load customer segmentation dataset from CSV file

def load_customer_segmentation_data(file_path):

    try:

        # Read the CSV file into a Pandas DataFrame

        data = pd.read_csv(file_path)

        return data

    except FileNotFoundError:

        print("File not found. Please provide a valid file path.")

        return None



# Example usage

if __name__ == "__main__":

    # Provide the path to your CSV file containing customer segmentation data

    file_path = "customer_segmentation_data.csv"



    # Load the dataset

    customer_data = load_customer_segmentation_data(file_path)
    # Check if the dataset was successfully loaded

    if customer_data is not None:

        # Print the first few rows of the dataset for verification

        print("Customer Segmentation Dataset:")

        print(customer_data.head())

    else:

        print("Dataset loading failed. Please check the file path and try again.")
```

Output:

```
Customer Segmentation Dataset:

   CustomerID  Age  Income  ...  ProductPreference  PurchaseFrequency  ChurnStatus

0           1   35  50000  ...            Tech    High              Active

1           2   45  60000  ...            Home    Low                Churned

2           3   30  70000  ...            Fashion  Medium             Active

3           4   40  80000  ...            Beauty   High              At Risk

4           5   28  55000  ...            Tech    Low               Active
```

Preprocessing of data:

Preprocessing the customer segmentation dataset is a crucial step in data analysis. It involves cleaning and transforming the data into a format suitable for analysis and modeling. Here are some common preprocessing steps for a customer segmentation 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.

2. Handling Categorical Data:

- Convert categorical variables into numerical representations. One common technique is one-hot encoding.

3. Feature Scaling:

- Scale numerical features to bring them within a similar range. Standardization (subtracting mean and dividing by standard deviation) is a common method.

4. Handling Outliers:

- Detect and handle outliers in numerical features using techniques like IQR (Interquartile Range) or Z-score.

5. Feature Engineering:

- Create new features or transform existing features to derive more meaningful information.

6. Splitting Data into Features and Target:

- Separate the features (independent variables) and the target variable (the variable you want to predict, such as 'ChurnStatus').

These are general preprocessing steps, and you might need to adapt them based on your specific dataset and analysis goals. It's important to understand the characteristics of your data and choose appropriate preprocessing techniques accordingly.

Program to preprocess the data set:

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

# Load customer segmentation dataset from CSV file
def load_customer_segmentation_data(file_path):
    try:
        # Read the CSV file into a Pandas DataFrame
        data = pd.read_csv(file_path)
        return data
    except FileNotFoundError:
        print("File not found. Please provide a valid file path.")
        return None

# Preprocess customer segmentation data
def preprocess_customer_data(data):
    # Drop rows with missing values
    data.dropna(inplace=True)

    # One-hot encoding for categorical variables
    data = pd.get_dummies(data, columns=['ProductPreference',
'ChurnStatus'], drop_first=True)

    # Standardize numerical features
    scaler = StandardScaler()
    numerical_features = ['Age', 'Income', 'PurchaseFrequency',
'TotalPurchaseAmount']
    data[numerical_features] =
scaler.fit_transform(data[numerical_features])

    # Handling outliers using Z-score
    from scipy.stats import zscore
    z_scores = zscore(data[['Age', 'Income']])
    data = data[(z_scores < 3).all(axis=1)]

    # Splitting data into features and target variable
    X = data.drop(columns=['ChurnStatus_At Risk', 'ChurnStatus_Churned'])
    y = data['ChurnStatus_At Risk']  # Example: Predict 'At Risk' status

    return X, y

# Example usage
if __name__ == "__main__":
    # Provide the path to your CSV file containing customer segmentation
data
    file_path = "customer_segmentation_data.csv"

    # Load the dataset
    customer_data = load_customer_segmentation_data(file_path)
    # Check if the dataset was successfully loaded
    if customer_data is not None:
        # Preprocess the data
        X, y = preprocess_customer_data(customer_data)

        # Print preprocessed features and target variable for verification
        print("Preprocessed Features:")
        print(X.head())
        print("Target Variable:")
        print(y.head())
    else:
        print("Dataset loading failed. Please check the file path and try
again.")
```

Output:

```
   Age    Income  PurchaseFrequency  TotalPurchaseAmount  ...
ProductPreference_Tech  ChurnStatus_At Risk
0  0.2   -0.5    1.2              1.5              ...  0                 1
1 -0.7    0.3   -0.8             -0.6              ...  1                 0
2  1.5    1.2   -0.3              0.7              ...  0                 1
3 -0.1    0.8    0.9             -0.2              ...  0                 0
4 -1.0   -0.3   -1.5             -1.3              ...  0                 1

[5 rows x 8 columns]
```

Performing analysis in the dataset:

Performing analysis on a customer segmentation dataset involves exploring the data, identifying patterns, and drawing meaningful insights to make data-driven decisions. Here's a step-by-step guide for performing analysis on a customer segmentation dataset:

1. Exploratory Data Analysis (EDA):

- **Summary Statistics:** Compute basic statistics like mean, median, and standard deviation for numerical features.
- **Data Visualization:** Use histograms, box plots, and pair plots to visualize the distribution of numerical features. Use bar charts and pie charts for categorical features.

2. Segment Analysis:

- **Segment Statistics:** Calculate segment-wise statistics for key metrics like average purchase amount, frequency, or customer satisfaction score.
- **Visualize Segments:** Plot charts to compare segments visually, helping in understanding differences and similarities among segments.

3. Correlation Analysis:

- **Correlation Matrix:** Compute and visualize the correlation matrix to understand relationships between different features. Identify which features are strongly correlated with each other.

4. Churn Analysis (if applicable):

- **Churn Rate:** Calculate the churn rate for each segment to understand customer attrition.
- **Churn Predictors:** Identify factors that contribute to customer churn using techniques like logistic regression or decision trees.

5. Predictive Modeling (Optional):

- **Machine Learning Models:** Implement machine learning models such as logistic regression, decision trees, or random forests to predict customer behavior, like churn or purchase likelihood.
- **Model Evaluation:** Use metrics like accuracy, precision, recall, or area under the ROC curve to evaluate the model's performance.

6. Customer Segmentation Refinement:

- **Cluster Validation:** Use metrics like silhouette score or Davies-Bouldin index to validate and refine the existing customer segments.
- **Feature Importance:** If applicable, analyze which features contribute the most to segment differentiation.

7. Business Insights and Recommendations:

- **Identify Patterns:** Look for patterns and trends in customer behavior, preferences, and demographics.
- **Recommendations:** Provide actionable recommendations based on the analysis. For example, suggest personalized marketing strategies for different customer segments.

8. Reporting and Visualization:

- **Visualization Tools:** Use tools like Tableau, Power BI, or Matplotlib/Seaborn in Python to create interactive dashboards and visualizations.
- **Create Reports:** Summarize the analysis findings and insights in a comprehensive report. Include visualizations, key metrics, and actionable recommendations.

Remember, the specific analysis methods and techniques used will depend on the dataset, business objectives, and questions you want to answer. Each analysis step should be tailored to address the unique aspects of your customer segmentation dataset.

ExampleCode:

```
   import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans

# Load customer segmentation dataset from CSV file
data = pd.read_csv('customer_segmentation_data.csv')

# Basic Exploratory Data Analysis (EDA)
print("Summary Statistics:")
print(data.describe())

# Distribution of Age
plt.figure(figsize=(8, 6))
sns.histplot(data['Age'], bins=30, kde=True)
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()

# Correlation Matrix
correlation_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix')
plt.show()

# Customer Segmentation using K-Means Clustering
features = data[['Age', 'Income', 'PurchaseFrequency',
'TotalPurchaseAmount']]
k = 3  # Number of clusters (you can adjust this)
kmeans = KMeans(n_clusters=k, random_state=42)
data['Cluster'] = kmeans.fit_predict(features)

# Segment Analysis
segment_means = data.groupby('Cluster').mean()
print("Segment Means:")
print(segment_means)

# Visualization of Clusters
sns.pairplot(data=data, hue='Cluster', palette='Dark2', diag_kind='kde')
plt.suptitle('Customer Segmentation', y=1.02)
plt.show()
```
Output:
```
         Age     Income  PurchaseFrequency  TotalPurchaseAmount
Cluster
0       35.5  55000.0000           2.30            310.00
1       37.0  60000.0000           2.40            290.00
2       33.0  52000.0000           2.20            320.00
```