Market Segmentation Customer Data Project

Madhumita Mondal

11/14/2023

In marketing, market segmentation is the process of dividing a broad consumer or business market, normally consisting of existing and potential customers, into sub-groups of consumers based on some type of shared characteristics.

Task:

This project focuses on customer segmentation, a technique commonly used in marketing and business strategy. The primary goal is to categorize customers into distinct groups based on their behavior, preferences, or characteristics. The project involves both unsupervised learning, specifically K-Means clustering, and supervised learning using a Decision Tree classifier.

Type of Learning/Algorithms:

Unsupervised Learning (Clustering):

Algorithm: K-Means Clustering Purpose: Group customers into clusters based on similarities in their features without the need for labeled target variables.

Supervised Learning (Classification):

Algorithm: Decision Tree Classifier Purpose: Train a model to predict the cluster (previously identified through K-Means) of a customer based on their features. This step involves a labeled dataset where the clusters serve as the target variable.

Goal:

The project seeks to leverage data-driven insights to enhance business strategies and decision-making by understanding and categorizing customer behavior. The combination of unsupervised learning for segmentation and supervised learning for prediction contributes to a holistic approach in addressing business challenges related to customer engagement and satisfaction.A case requires to develop a customer segmentation to give recommendations like saving plans, loans, wealth management, etc. on target customers groups.

Dataset:

The sample Dataset summarizes the usage behavior of about 9000 active credit card holders during the last 6 months. The file is at a customer level with 18 behavioral variables. Dataset is collected from Kaggle. It has 8950 rows × 18 columns.

https://www.kaggle.com/code/fabiendaniel/customer-segmentation/input (https://www.kaggle.com/code/fabiendaniel/customer-segmentation/input)

Variables of Dataset

Balance □ Balance Frequency □ Purchases □ One-off Purchases □ Installment Purchases □ Cash Advance □ Purchases Frequency □ One-off Purchases Frequency □ Purchases Installments Frequency □ Cash Advance Frequency □ Cash Advance TRX □ Purchases TRX □ Credit Limit □ Payments □ Minimum Payments □ PRC Full payment □ Tenure □ Cluster

Importing Libraries

```python
In [ ]:  import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         from sklearn.preprocessing import StandardScaler
         scalar=StandardScaler()
         from sklearn.decomposition import PCA
         from sklearn.cluster import KMeans,AgglomerativeClustering,DBSCAN,Spectr
         from sklearn.mixture import GaussianMixture
         from sklearn.metrics import silhouette_samples, silhouette_score
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import classification_report
         from sklearn import tree
         from sklearn import metrics

         import warnings
         warnings.filterwarnings("ignore")
```

Loading the dataset

```python
In [ ]:  df = pd.read_csv("Customer Data.csv")
         df
```

EDA

```
In [ ]:    ▶  df.shape
```

```
In [ ]:    ▶  df.info()
```

```
In [ ]:    ▶  df.describe()
```

```
In [ ]:    ▶  df.isnull().sum()
```

Data Cleaning Steps:

Handling Missing Values:

Why: Missing values in the dataset, especially in features like "MINIMUM_PAYMENTS" and "CREDIT_LIMIT," can impact the accuracy of analysis and modeling.

How: Filled missing values in "MINIMUM_PAYMENTS" and "CREDIT_LIMIT" columns with the mean values of those columns using fillna()

```
In [ ]:    ▶  # filling mean value in place of missing values in the dataset
              df["MINIMUM_PAYMENTS"] = df["MINIMUM_PAYMENTS"].fillna(df["MINIMUM_PAYMI
              df["CREDIT_LIMIT"] = df["CREDIT_LIMIT"].fillna(df["CREDIT_LIMIT"].mean()
```

```
In [ ]:    ▶  df.isnull().sum()
```

Handling Duplicate Rows:

Why: Duplicate rows can distort analysis and lead to inaccurate insights.

How: Checked for and dropped duplicate rows using duplicated().sum() and drop_duplicates().

```
In [ ]:    ▶  # checking for duplicate rows in the dataset
              df.duplicated().sum()
```

Drop Unnecessary Columns:

Why: The "CUST_ID" column was dropped as it doesn't contribute to the analysis.

How: Used drop(columns=["CUST_ID"], axis=1, inplace=True).

```
In [ ]:  ▶  # drop CUST_ID column because it is not used
            df.drop(columns=["CUST_ID"],axis=1,inplace=True)
```

```
In [ ]:  ▶  df.columns
```

Visualizing Data Distributions:

Why: Understanding the distribution of features helps in identifying outliers and assessing the overall data quality.

How: Utilized various visualizations such as KDE plots and histograms to visualize the distribution of numerical features.

```
In [ ]:  ▶  plt.figure(figsize=(30,45))
            for i, col in enumerate(df.columns):
                if df[col].dtype != 'object':
                    ax = plt.subplot(9, 2, i+1)
                    sns.kdeplot(df[col], ax=ax)
                    plt.xlabel(col)

            plt.show()
```

This step, generates a set of distribution plots for numerical features, supporting the exploration and understanding of the data's underlying patterns and characteristics during the initial stages of the project.Its a set of 17 vertical histograms, one for each numerical feature in the df DataFrame.

```
In [ ]:  ▶  plt.figure(figsize=(10,60))
            # Loop through each numerical feature in the DataFrame
            for i in range(0,17):
                # Create subplots in a single column layout
                plt.subplot(17,2,i+1)
                # Plot a distribution plot (histogram with KDE) for the current feat
                sns.distplot(df[df.columns[i]],kde_kws={'color':'b','bw': 0.1,'lw':
                # Set the title for the subplot based on the feature name
                plt.title(df.columns[i])
            # Adjust layout for better spacing & Display the plot
            plt.tight_layout()
```

Correlation Analysis:

Explanation: Created a heatmap to visualize the correlation matrix, highlighting relationships between features.

Outcome: Identified correlations between features, aiding in understanding the interdependencies within the dataset.

```
In [ ]:  ▶| plt.figure(figsize=(12,12))
            sns.heatmap(df.corr(), annot=True)
            plt.show()
```

Conclusions and Discussions of EDA:

Handling Missing Values:

The imputation of missing values in "MINIMUM_PAYMENTS" and "CREDIT_LIMIT" was performed to ensure completeness in the dataset. The use of mean values is a common strategy, but other methods like median or advanced imputation techniques could be explored based on the nature of the data.

Handling Duplicate Rows:

No duplicate rows were found, ensuring the dataset's integrity.

Drop Unnecessary Columns:

The "CUST_ID" column was dropped as it does not contribute to the analysis. This simplifies the dataset without losing relevant information.

Visualizing Data Distributions:

Visualizations revealed the distribution of numerical features, helping identify potential outliers or skewed data. The heatmap showed correlations between features, aiding in understanding the relationships within the dataset.

```
In [ ]:  ▶|
```

Scaling the DataFrame

Scaling the DataFrame is a prerequisite for various machine learning algorithms to ensure fair and meaningful contributions from all features, enhance model stability, and facilitate consistent interpretation of results across different models. It is a fundamental step in the preprocessing pipeline to improve the overall performance and reliability of machine learning models, especially when features in the dataset have different scales or units.Here are the key requirements and reasons for scaling the DataFrame:

1.Homogenizing Feature Scales 2.Distance-Based Algorithms 3.Gradient Descent Optimization 4.PCA (Principal Component Analysis) 5.Neural Networks 6.Regularization Techniques 7.Support Vector Machines (SVM) 8.Consistent Model Interpretation

```
In [ ]:    scaled_df = scalar.fit_transform(df)
```

Dimensionality reduction

Converting the DataFrame into 2D DataFrame for visualization

```
In [ ]:    pca = PCA(n_components=2)
           principal_components = pca.fit_transform(scaled_df)
           pca_df = pd.DataFrame(data=principal_components ,columns=["PCA1","PCA2"]
           pca_df
```

Hyperparameter tuning

Finding 'k' value by Elbow Method

The hyperparameter tuning process, specifically using the Elbow Method to find the optimal 'k' value, plays a pivotal role in ensuring the efficacy of the K-Means clustering algorithm. It provides a data-driven approach to determining the number of clusters that best represent the underlying patterns in the dataset, enhancing the reliability and utility of the clustering results in the project.

```
In [ ]:    inertia = []
           range_val = range(1,15)
           for i in range_val:
               kmean = KMeans(n_clusters=i)
               kmean.fit_predict(pd.DataFrame(scaled_df))
               inertia.append(kmean.inertia_)
           plt.plot(range_val,inertia,'bx-')
           plt.xlabel('Values of K')
           plt.ylabel('Inertia')
           plt.title('The Elbow Method using Inertia')
           plt.show()
```

Summary and Analysis of the Model and Predictions:

Customer Segmentation using K-Means:

Summary:

Employed K-Means clustering to categorize customers into distinct segments based on their features. Determined the optimal number of clusters using the Elbow Method. Visualized the clustered data in a 2D space using Principal Component Analysis (PCA).

Analysis:

Identified natural groupings of customers, allowing for targeted marketing and personalized strategies. Utilized PCA for dimensionality reduction, facilitating visualization and interpretation.

Decision Tree Classification Model:

Summary:

Utilized a Decision Tree classifier to predict the cluster labels of customers. Trained the model on the segmented data from K-Means. Evaluated model performance using a confusion matrix and classification report.

Analysis:

Provided a interpretable model for predicting customer clusters. Evaluated precision, recall, and F1-score for each cluster, assessing the model's ability to correctly classify customers into their respective segments.

Feature Importance Analysis:

Summary:

While a Decision Tree was used, there is no explicit mention of analyzing feature importance.

Analysis:

It's crucial to investigate feature importance to understand which features contribute most to the clustering and classification processes. This analysis can guide business decisions.

Handling of Interaction/Collinearity:

Summary:

Feature correlations were visualized using a heatmap during EDA.

Analysis:

Collinearity may impact the interpretation of the Decision Tree model, and it would be beneficial to explicitly address this issue using statistical methods.

Handling of Overfitting/Data Imbalance:

Summary:

No explicit mention of techniques used to reduce overfitting or address data imbalance.

Analysis:

Considering techniques like pruning for Decision Trees and evaluating the balance of clusters in the K-Means analysis could enhance the robustness of the models.

Exploration of New Techniques/Models:

Summary:

Primarily used standard techniques like K-Means and Decision Trees.

Analysis:

While the chosen models are appropriate, exploring additional clustering algorithms or ensemble methods could provide insights into alternative modeling approaches.

Model Building using KMeans

K-Means Clustering Model Analysis:

Objective:

The goal of the K-Means clustering model is to categorize customers into distinct segments based on their features. The primary metrics for evaluation are inertia (within-cluster sum of squares) and silhouette score, which were used to determine the optimal number of clusters during the Elbow Method analysis.

Data Scaling:

Before applying K-Means, the dataset was standardized using the StandardScaler to ensure that features with different scales do not disproportionately influence the clustering process. Scaling is crucial for K-Means as it relies on the Euclidean distance between data points.

Dimensionality Reduction with PCA:

Principal Component Analysis (PCA) was used to reduce the dimensionality of the data to two principal components, facilitating visualization of the clusters. PCA aids in retaining as much information as possible while simplifying the data for visualization.

Optimal Number of Clusters (K):

The Elbow Method was employed to determine the optimal number of clusters by plotting the inertia (within-cluster sum of squares) against different values of K. The "elbow" point in the plot is considered the optimal K. Beyond this point, the reduction in inertia becomes marginal. The optimal K value was determined for the subsequent application of K-Means.

Application of K-Means:

The K-Means algorithm was applied with the determined optimal number of clusters. Each customer was assigned a cluster label based on their features. The clustered data was visualized in a 2D space using the two principal components obtained from PCA.

Analysis of Cluster Centers:

The cluster centers were extracted and inverse-transformed to the original scale using the StandardScaler. The cluster centers represent the average feature values for each cluster, providing insights into the characteristics of customers in each segment.

Creation of Target Column "Cluster":

A target column, "Cluster," was added to the original dataset, storing the cluster labels assigned by K-Means. This target variable is then used for further analysis and potentially for training a supervised learning model.

Exploration of Each Cluster:

Individual datasets were created for each cluster to explore the characteristics and behaviors of customers within each segment. Count plots and histograms were created to visualize the distribution of features within each cluster.

Visual Validation:

Scatterplots were used to visualize the clustered data in the 2D space. Visual inspection of the plots helps validate the effectiveness of the clustering algorithm in forming distinct groups.

Saving Results:

The K-Means model and the clustered dataset were saved for future use. The model was saved using joblib, and the clustered dataset was saved as a CSV file.

In [ ]: ▶|

```python
kmeans_model=KMeans(4)
kmeans_model.fit_predict(scaled_df)
pca_df_kmeans= pd.concat([pca_df,pd.DataFrame({'cluster':kmeans_model.la
```

Visualizing the clustered dataframe

The visualization of the clustered dataframe provides a crucial lens through which to understand the distribution and characteristics of customer segments. The visualization aims to present a clear and intuitive representation of how customers are grouped into distinct segments based on their features. This visualization serves as a powerful tool for both exploratory data analysis and communicating insights derived from clustering algorithms, particularly K-Means. Utilized scatterplots to display the distribution of data points in a 2D space, with each point representing an individual customer. Different colors or markers are employed to distinguish between clusters.Utilized scatterplots to display the distribution of data points in a 2D space, with each point representing an individual customer. Different colors or markers are employed to distinguish between clusters.

```
In [ ]: ▶| plt.figure(figsize=(8,8))
         ax=sns.scatterplot(x="PCA1",y="PCA2",hue="cluster",data=pca_df_kmeans,pa
         plt.title("Clustering using K-Means Algorithm")
         plt.show()
```

```
In [ ]: ▶| # find all cluster centers
         cluster_centers = pd.DataFrame(data=kmeans_model.cluster_centers_,colum
         # inverse transform the data
         cluster_centers = scalar.inverse_transform(cluster_centers)
         cluster_centers = pd.DataFrame(data=cluster_centers,columns=[df.columns]
         cluster_centers
```

```
In [ ]: ▶| # Creating a target column "Cluster" for storing the cluster segment
         cluster_df = pd.concat([df,pd.DataFrame({'Cluster':kmeans_model.labels_
         cluster_df
```

```
In [ ]: ▶| cluster_1_df = cluster_df[cluster_df["Cluster"]==0]
         cluster_1_df
```

```
In [ ]: ▶| cluster_2_df = cluster_df[cluster_df["Cluster"]==1]
         cluster_2_df
```

```
In [ ]: ▶| cluster_3_df = cluster_df[cluster_df["Cluster"]==2]
         cluster_3_df
```

```
In [ ]: ▶| cluster_4_df = cluster_df[cluster_df["Cluster"] == 3]
         cluster_4_df
```

The count plot

The count plot of cluster distribution is a valuable visualization in the project, providing a high-level overview of how data points are distributed among different clusters. It offers insights into the composition and significance of each cluster, aiding in the interpretation of the clustering results and informing subsequent business strategies

In [ ]: ▶| `#Visualization`
```python
#Visualization
sns.countplot(x='Cluster', data=cluster_df)
```

Visual representation of how the distribution of each feature varies across different clusters

The next step using the Seaborn library to create a set of histograms for each feature in the cluster_df DataFrame, grouped by the 'Cluster' column.This plot is used to display the grid of histograms. Each row in the grid corresponds to a different feature, and each column corresponds to a different cluster. This visualization allows for a quick comparison of the distributions of each feature across different clusters.

This set of histograms is valuable in the exploratory data analysis (EDA) phase of the project. It provides a visual representation of how the distribution of each feature varies across different clusters. Analyzing these histograms helps in understanding the characteristics and patterns that distinguish one cluster from another, aiding in the interpretation of the clustering results.

In [ ]: ▶|
```python
# Loop through each feature (column) in the DataFrame excluding the 'Clu
for c in cluster_df.drop(['Cluster'],axis=1):
    # Create a FacetGrid for subplots, with separate columns for each cl
    grid= sns.FacetGrid(cluster_df, col='Cluster')
     # Map a histogram for the current feature onto the grid
    grid= grid.map(plt.hist, c)
# Display the grid of histograms
plt.show()
```

Type *Markdown* and LaTeX: α^2

Saving the kmeans clustering model and the data with cluster label

In [ ]: ▶|
```python
#Saving Scikitlearn models
import joblib
joblib.dump(kmeans_model, "kmeans_model.pkl")
```

In [ ]: ▶| ```python
cluster_df.to_csv("Clustered_Customer_Data.csv")
```

# Decision Tree Classification Model:

Results:

Trained a Decision Tree classifier to predict the cluster labels of customers. Evaluated the model's performance using a confusion matrix and classification report.

Analysis:

Precision, recall, and F1-score for each cluster provide insights into the model's ability to correctly classify customers. Interpretability of the Decision Tree aids in understanding the key features influencing cluster predictions.

Definition:

Accuracy is the ratio of correctly predicted instances to the total instances. It is commonly used for evaluating the overall performance of a classification model.

Calculation:

Accuracy = Number of Correct Predictions/Total Number of Predictions

Interpretation:

A high accuracy indicates that the model is making correct predictions. However, accuracy might be misleading if the dataset is imbalanced (unequal distribution of classes), as the model could achieve high accuracy by predicting the majority class.

# Training and Testing the model accuracy using decision tree

In [ ]: ▶| ```python
#Split Dataset
X = cluster_df.drop(['Cluster'],axis=1)
y= cluster_df[['Cluster']]
X_train, X_test, y_train, y_test =train_test_split(X, y, test_size=0.3)
```

In [ ]: ▶| ```python
X_train
```

In [ ]: ▶ `X_test`

In [ ]: ▶
```python
#Decision_Tree
model= DecisionTreeClassifier(criterion="entropy")
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

In [ ]: ▶
```python
#Confusion_Matrix
print(metrics.confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Saving the Decision tree model for future prediction

In [ ]: ▶
```python
import pickle
filename = 'final_model.sav'
pickle.dump(model, open(filename, 'wb'))

# some time later...

# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_test, y_test)
print(result,'% Acuuracy')
```

Overall Assessment:

Our project did a great job in sorting out customers into meaningful groups using K-Means clustering, which helped us learn a lot about how customers behave and what they prefer. The Decision Tree model we used is like a clear guide for predicting which group a customer belongs to. To make our findings even more reliable, we suggest digging deeper into which features really matter, handling any tricky similarities between features, and making sure our model doesn't get too focused on our training data. It's also a good idea to explore other methods to see if they might work even better. This way, we keep refining and improving our understanding of customers, making our insights even more valuable for making smart business decisions.

Next Steps:

This project may help in understanding the customer behavior and preferences specific to each cluster. Additionally, we can perform further statistical tests or machine learning models within each cluster to uncover more detailed patterns and potentially tailor marketing or customer engagement strategies based on these distinct customer groups. We can also make sure that when some factors are too similar, a common issue called collinearity, we

can handle it with statistical tools so our results are trustworthy. For Decision Trees, which help us make predictions, we can prune them a bit to avoid being too focused on our training data. When we're dividing our customers into groups, we want those groups to be fair, so we can check if they're balanced.And finally we can deploy this model to an app for real world application usage.

The real-world value of this project

This project brings a lot of real-world benefits. It helps businesses understand their customers better, making it easier to create effective marketing strategies and improve how things operate. By using data to figure out what customers need, the project supports efficient operations and helps keep customers happy, which is essential for any business. Plus, it encourages a culture of using data to make smart decisions, which is a key factor in long-term success. The project making it a valuable tool for businesses aiming to do well in a competitive world where understanding and serving customers is crucial.