Customer Segmentation Report for Arvato Financial Services

Parham Dehghani

May 26, 2023

1 Introduction

In this report, we present the findings and insights from a customer segmentation project conducted for Arvato Financial Services. The objective of the project was to analyze demographic data provided by Bertelsmann Arvato Analytics and identify distinct customer segments. By understanding the characteristics of different customer segments, Arvato Financial Services can tailor their marketing strategies and improve customer targeting.

The project involved two main datasets: AZDIAS, representing the general population, and CUSTOMERS, representing Arvato's customer base. We aimed to compare the demographic attributes of the customer base with the general population and uncover patterns and differences that could help identify valuable customer segments.

The project encompassed several steps, including data preprocessing, dimensionality reduction, clustering analysis, and predictive modeling. Each step aimed to extract meaningful insights from the data and enable the identification of distinct customer segments.

2 Data Preprocessing

The first step in the project was to preprocess the datasets, AZDIAS and CUSTOMERS, to ensure data quality and consistency [1]. Initially, we loaded the datasets and examined their structure and content, employing NumPy [2] and Pandas [3].

During the data cleaning process, we identified various issues that needed to be addressed. This included converting date columns to numerical format to facilitate analysis. We also encountered missing values in categorical columns, which we handled by imputing the missing values using appropriate techniques. Additionally, we converted object-type columns to numeric format where applicable to enable further analysis.

To improve the quality of the data, we dropped irrelevant columns with excessive missing data. This ensured that we focused on the most informative features during the subsequent analysis. For the remaining columns, we performed imputation techniques to fill in missing values. Imputation method based on the mode of each column was used based on the characteristics of the data. Following are the steps that are fulfilled for preprocessing data.

```
Step 1: Importing the required libraries and packages

import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

Step 2: Loading the data

azdias = pd.read_csv('./Data/Udacity_AZDIAS_052018.csv', sep=';')
customers = pd.read_csv('./Data/Udacity_CUSTOMERS_052018.csv', sep=';')


Step 3: Preprocessing the data

# Converting date columns to numeric values
azdias['EINGEFUEGT_AM'] = pd.to_datetime(azdias['EINGEFUEGT_AM']).\
astype(int)
customers['EINGEFUEGT_AM'] = pd.to_datetime(customers['EINGEFUEGT_AM']).\
astype(int)

# Converting object columns to numeric values
azdias['CAMEO_DEUG_2015'] = azdias['CAMEO_DEUG_2015'].replace('X', -1).\
astype(int)
azdias['CAMEO_INTL_2015'] = azdias['CAMEO_INTL_2015'].replace('XX', -1).\
astype(int)
customers['CAMEO_DEUG_2015'] = customers['CAMEO_DEUG_2015'].\
replace('X', -1).astype(int)
customers['CAMEO_INTL_2015'] = customers['CAMEO_INTL_2015'].\
replace('XX', -1).astype(int)

# Dropping irrelevant and high-null columns
azdias.drop(['EXTSEL992'], axis=1, inplace=True)
azdias.drop(azdias.columns[4:8], axis=1, inplace=True)

customers.drop(['EXTSEL992'], axis=1, inplace=True)
customers.drop(customers.columns[4:8], axis=1, inplace=True)

# Imputing null values
for col in azdias.columns:
    try:
        azdias[col] = azdias[col].\
        transform(lambda x: x.fillna(x.mode()[0]))
    except:
        print('That␣broke...')

# Applying one-hot encoding to categorical variables
azdias = pd.get_dummies(azdias)
customers = pd.get_dummies(customers)
```

3 Dimensionality Reduction

Dimensionality reduction was performed to reduce the number of variables in the datasets while preserving the most important information. This step was essential to address the curse of dimensionality and simplify the subsequent clustering analysis.

We applied Principal Component Analysis (PCA) [4] from sklearn library [5] to 30% of both the AZDIAS and CUSTOMERS datasets, as data is hyper-dimensional and huge in terms of processing on a stand-alone system. PCA transformed the original variables into a new set of linearly uncorrelated variables called principal components [6]. These components were ranked based on their ability to explain the variance in the data.

To determine the optimal number of components to retain, we calculated the explained variance ratio for different numbers of components. By plotting the explained variance ratio, we identified the number of shared features to be 232 at which adding more components did not contribute significantly to the overall explained variance. This allowed us to select an appropriate number of components for further analysis.

The selected principal components represented combinations of the original features that captured the maximum amount of information. These components were used as inputs for the subsequent clustering analysis, effectively reducing the dimensionality of the data.

```
Step 4: Data Analysis and Dimensionality Reduction

We use 30 percent of data as it is hyper-dimensional and huge in
number of records for dimensional reduction

# sampling original processed data
customers_sample = customers.sample(frac=0.3).reset_index(drop=True)
azdias_sample = azdias.sample(frac=0.3).reset_index(drop=True)

# customers standardization
# Select columns to standardize (excluding the first column)
columns_to_standardize = customers_sample.columns[1:]

# Initializing the StandardScaler
scaler = StandardScaler()

# Fitting the scaler to the data
scaler.fit(customers_sample[columns_to_standardize])

# Transforming the data
customers_sample[columns_to_standardize] = scaler.\
transform(customers_sample[columns_to_standardize])


# azdias standardization
# Select columns to standardize (excluding the first column)
columns_to_standardize = azdias_sample.columns[1:]

# Initializing the StandardScaler
scaler = StandardScaler()
```

```
# Fitting the scaler to the data
scaler.fit(azdias_sample[columns_to_standardize])

# Transforming the data
azdias_sample[columns_to_standardize] = scaler.\
transform(azdias_sample[columns_to_standardize])

# copying to new dataframes
azdias=azdias_sample.copy()
customers=customers_sample.copy()

# Finding the common features between the two datasets
common_features = list(set(azdias.columns) & set(customers.columns))
# remove identifier column
common_features.remove('LNR')
# Concatenating two datasets considering common features
combined_df = pd.concat([azdias[common_features],\
 customers[common_features]], ignore_index=True)

best_components = 0  # Variable to store the best number of components
max_features = combined_df.shape[1]

for num_components in range(1, max_features + 1):
    pca = PCA(n_components=num_components)
    df_reduced = pca.fit_transform(combined_df)

    explained_variance = sum(pca.explained_variance_ratio_)

    if explained_variance > 0.9:

        best_components = num_components
        break

--> Best number of features that can explain 90 percent of
the variance is:  232

We found that ncomponents=232 can explain 90% of the variance in the
original combined dataset. We then proceed with doing dimensional
reduction for azdias and customers with this number of components

ncomponents=232
# azdias dimensional reduction
pca = PCA(n_components=ncomponents)
df_reduced1 = pca.fit_transform(azdias.iloc[:,1:])
azdias_reduced = pd.concat([azdias.iloc[:,0],\
pd.DataFrame(df_reduced1)],axis=1)

# customers dimensional reduction
pca = PCA(n_components=ncomponents)
```

```
df_reduced2 = pca.fit_transform(customers.iloc[:,1:])
customers_reduced = pd.concat([customers.iloc[:,0],\
pd.DataFrame(df_reduced2)],axis=1)

# Writing to drive
azdias_reduced.to_csv('./Data/azdias_reduced.csv',index=False)
customers_reduced.to_csv('./Data/customers_reduced.csv',index=False)
```

4 Clustering Analysis

Clustering analysis was employed to identify distinct customer segments based on the reduced-dimensional data. We used the K-means algorithm [7], a popular unsupervised learning technique, to cluster the data points into homogeneous groups.

To determine the optimal number of clusters, we employed the elbow method. We ran the K-means algorithm for a range of cluster numbers and calculated the sum of squared distances between data points and their respective cluster centroids. By plotting the sum of squared distances against the number of clusters, we identified the elbow point to be 10, indicating the optimal number of clusters where the decrease in distortion diminished significantly.

After determining the optimal number of clusters, we performed K-means clustering on the principal components of the AZDIAS dataset. Each individual was assigned to a specific cluster based on their similarity to other individuals in terms of their demographic attributes.

We analyzed the similarity of each cluster with the company customers data by examining the inverse centroid distance as a measure of affinity between a chosen cluster and the company customers. This allowed us to interpret and describe the profiles of each segment in terms of the demographic attributes that contributed to their formation. By comparing these profiles with the customers population data, we gained insights into the distinguishing characteristics of each segment.

```
Step 5: Clustering Analysis

from sklearn.cluster import KMeans
wcss = []
for i in range(1, 20):
    kmeans = KMeans(n_clusters = i, init = 'k-means++')
    kmeans.fit(azdias_reduced.iloc[:,1:].values)
    wcss.append(kmeans.inertia_)

# Plotting the elbow curve
plt.plot(range(1, 20), wcss)
plt.title('The␣Elbow␣Method')
plt.xlabel('Number␣of␣clusters␣for␣demography␣data')
plt.ylabel('WCSS')
plt.show()

--> we see that n_clusters=10 would be a good choice for
clustering as inertia is not meaningfully changing after
this number
```

```
Step 6: Applying K-means clustering

# Applying K-means clustering with the chosen number of clusters
kmeans1 = KMeans(n_clusters = 10, init = 'k-means++')
azdias_kmeans = kmeans1.fit_predict(azdias_reduced.iloc[:,1:])


kmeans2 = KMeans(n_clusters = 1, init = 'k-means++')
customer_kmeans = kmeans2.fit_predict(customer_reduced.iloc[:,1:])

Step 7: Analyzing clustered data with respect to company data

from sklearn.metrics import pairwise_distances

# Calculating cluster similarity by calculating Euclidean
# pairwise distance between clusters in two datasets
centroid_distances = pairwise_distances(kmeans1.cluster_centers_, kmeans2.\
cluster_centers_)

# Converting Euclidean distances to similarities (using inverse distance)
similarity_matrix = 1 / (1 + centroid_distances)

array([[0.02488811],
       [0.08678226],
       [0.10243211],
       [0.1591847 ],
       [0.1184324 ],
       [0.10843472],
       [0.134481  ],
       [0.09939991],
       [0.15072899],
       [0.09954378]])

--> we choose clusters 3 and 8 of the demography as the most similar
 clusters to the company customers


# Assigning cluster labels to each dataset
azdias = azdias_reduced.copy()
customer = customer_reduced.copy()
azdias['Cluster_label']=kmeans1.labels_
customer['Cluster_label']=kmeans2.labels_

We have now two datasets that are clustered and each cluster from
azdias is mapped to the cluster in customer. We make a dictionary
including all individuals that are most likely similar to company
customers

affine_individuals = {'Similar Individuals':[]}
for i in (3,8):
```

```
    affine_individuals['Similar␣Individuals'].extend(azdias.\
    loc[azdias['Cluster_label']==i,'LNR'].values)

# Saving dictionary using pickling
import pickle

with open('affinity.pickle', 'wb') as file:
    pickle.dump(affine_individuals, file)
```

5 Model Training: Supervised Learning

In addition to clustering analysis, we utilized predictive modeling techniques to understand the factors driving customer behavior and identify key predictors for customer segmentation. We employed random forest algorithm [8] to predict customer response or behavior based on the available demographic attributes.

First, we preprocessed `Udacity_MAILOUT_052018_TRAIN.csv` and then split the dataset into training (75%) and testing (25%) subsets to train and evaluate the predictive model. We used accuracy and confusion matrix to assess the performance of the model.

Secondly, we incorporated the trained model to make predictions for the individuals listed in `Udacity_MAILOUT_052018_TEST.csv`, evaluating the RESPONSE value for each of them. At last, we added a new column to this dataset that included the predicted RESPONSE for each individual given the demographic features. We found that 39 individuals are potential customers based on the trained model.

The predictive model provided valuable insights into the relationship between demographic attributes and customer behavior. These insights could be utilized to develop targeted marketing strategies for each customer segment, optimizing the effectiveness of Arvato Financial Services' campaigns.

```
Step 8: Supervised learning employing Udacity_MAILOUT_052018_TRAIN.csv

# Reading train data
train_dataset = pd.read_csv('./Data/Udacity_MAILOUT_052018_TRAIN.csv',\
sep=';')

# Preprocessing train data
train_dataset['EINGEFUEGT_AM'] = pd.\
to_datetime(train_dataset['EINGEFUEGT_AM'])

train_dataset['EINGEFUEGT_AM'] = train_dataset.\
loc[~train_dataset['EINGEFUEGT_AM'].isnull(),'EINGEFUEGT_AM'].\
astype(int)

train_dataset.loc[train_dataset['CAMEO_INTL_2015']=='XX',\
'CAMEO_INTL_2015']=-1

train_dataset['CAMEO_INTL_2015'] = train_dataset.\
loc[~train_dataset['CAMEO_INTL_2015'].isnull(),'CAMEO_INTL_2015'].\
astype(int)
```

```
train_dataset.loc[test_dataset['CAMEO_DEUG_2015']=='X',\
'CAMEO_DEUG_2015']=-1

train_dataset['CAMEO_DEUG_2015'] = train_dataset.\
loc[~train_dataset['CAMEO_DEUG_2015'].isnull(),'CAMEO_DEUG_2015'].\
astype(int)

train_dataset.iloc[:,300].fillna(-1,inplace=True)
dropped_cols = train_dataset.columns[4:8]
train_dataset.drop(dropped_cols,axis=1,inplace=True)
for col in train_dataset.columns:
    try:
        train_dataset[col] = train_dataset[col].\
        transform(lambda x: x.fillna(x.mode()[0]))
    except:
        print('That broke...')

# Dealing with categorical varibles
train_dataset = pd.get_dummies(train_dataset)
```

Now preprocessing is done and we can start training the model with train dataset and employing random forest algorithm

```
X = train_dataset.drop('RESPONSE',axis=1).values
y = train_dataset.loc[:, 'RESPONSE'].values

# spliting dataset: 75% train data, 25% test data
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = \
train_test_split(X, y, test_size = 0.25)

# Training the model
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators = 10,\
 criterion = 'entropy')
classifier.fit(X_train, y_train)

# Predicting the response for test set
y_pred = classifier.predict(X_test)

# Making confusion matrix
from sklearn.metrics import confusion_matrix, accuracy_score

cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
--> [[10618     9]
    [  114     0]]
0.9885485522763243
```

We see that our accuracy for the trained model is 98.8% which is acceptable.Now the trained model is verified and we can apply it to Udacity_MAILOUT_052018_TEST.csv

Step 9: Finding potential customers in Udacity_MAILOUT_052018_TEST.csv dataset

```
# Reading csv file
test_dataset = pd.read_csv('./Data/Udacity_MAILOUT_052018_TEST.csv',
 sep=';')

# Preprocessing as done for the training dataset
test_dataset['EINGEFUEGT_AM'] = pd.\
to_datetime(test_dataset['EINGEFUEGT_AM'])
test_dataset['EINGEFUEGT_AM'] = test_dataset.\
loc[~test_dataset['EINGEFUEGT_AM'].isnull(),'EINGEFUEGT_AM'].\
astype(int)

test_dataset.loc[test_dataset['CAMEO_INTL_2015']=='XX',\
'CAMEO_INTL_2015']=-1

test_dataset['CAMEO_INTL_2015'] = test_dataset.\
loc[~test_dataset['CAMEO_INTL_2015'].isnull(),'CAMEO_INTL_2015'].\
astype(int)

test_dataset.loc[test_dataset['CAMEO_DEUG_2015']=='X',\
'CAMEO_DEUG_2015']=-1

test_dataset['CAMEO_DEUG_2015'] = test_dataset.\
loc[~test_dataset['CAMEO_DEUG_2015'].isnull(),'CAMEO_DEUG_2015'].\
astype(int)

test_dataset.iloc[:,300].fillna(-1,inplace=True)
dropped_cols = test_dataset.columns[4:8]
test_dataset.drop(dropped_cols,axis=1,inplace=True)
for col in test_dataset.columns:
    try:
        test_dataset[col] = test_dataset[col].\
        transform(lambda x: x.fillna(x.mode()[0]))
    except:
        print('That broke...')

test_dataset = pd.get_dummies(test_dataset)

# Feature scaling
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
X = sc.fit_transform(test_dataset.values)


# Predicting the response for test set
y = classifier.predict(X)

# Adding RESPONSE column to test dataset
labeled_test_data = pd.read_csv('./Data/Udacity_MAILOUT_052018_TEST.csv',\
 sep=';')
labeled_test_data['RESPONSE'] = y

# Saving new labled test file as a csv file
labeled_test_data.to_csv('./Data/Udacity_MAILOUT_052018_TEST_LABELED.csv',\
index=False)

# Checking for RESPONSE=1 in the labeled data
potential_customers = labeled_test_data.\
loc[labeled_test_data.iloc[:,-1]==1,:].LNR

--> We see that our trained model predicts 39 individuals would
become customer of the company given the demographic features

potential_customers.to_csv('./potential_customers.csv',index=False)
```

6 Conclusion

In conclusion, this customer segmentation project for Arvato Financial Services aimed to identify distinct customer segments based on demographic data. By employing various techniques, including data preprocessing, dimensionality reduction, clustering analysis, and predictive modeling, we extracted valuable insights and generated actionable recommendations.

Through the exploration of demographic data from the general population and Arvato's customer base, we identified distinguishing characteristics and patterns among different customer segments. These insights enable Arvato Financial Services to tailor their marketing strategies and improve customer targeting, leading to more effective campaigns and enhanced customer satisfaction.

It is important to note that the success of customer segmentation relies on continuous monitoring and adaptation. As customer preferences and demographics evolve, regular updates to the segmentation model will be necessary to ensure its relevance and effectiveness.

Overall, this project provided a primary understanding of Arvato Financial Services' customer base and the potential for targeted marketing strategies. By leveraging the power of customer segmentation, Arvato Financial Services can maximize their marketing efforts and strengthen their position in the competitive landscape.

References

[1] T. Hastie, R. Tibshirani and J. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*. Springer Science and Business Media, 2009.

[2] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau et al., *Array programming with numpy*, *Nature* **585** (2020) 357–362.

[3] W. McKinney, *Data structures for statistical computing in python*, *Proceedings of the 9th Python in Science Conference* **445** (2010) .

[4] I. T. Jolliffe, *Principal component analysis*, *Wiley Online Library* (2002) .

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel et al., *Scikit-learn: Machine learning in python*, *Journal of Machine Learning Research* **12** (2011) 2825–2830.

[6] S. Wold, K. Esbensen and P. Geladi, *Principal component analysis*, *Chemometrics and Intelligent Laboratory Systems* **2** (1987) 37–52.

[7] A. K. Jain, *Data clustering: 50 years beyond k-means*, *Pattern Recognition Letters* **31** (2010) 651–666.

[8] L. Breiman, *Random forests*, *Machine Learning* **45** (2001) 5–32.