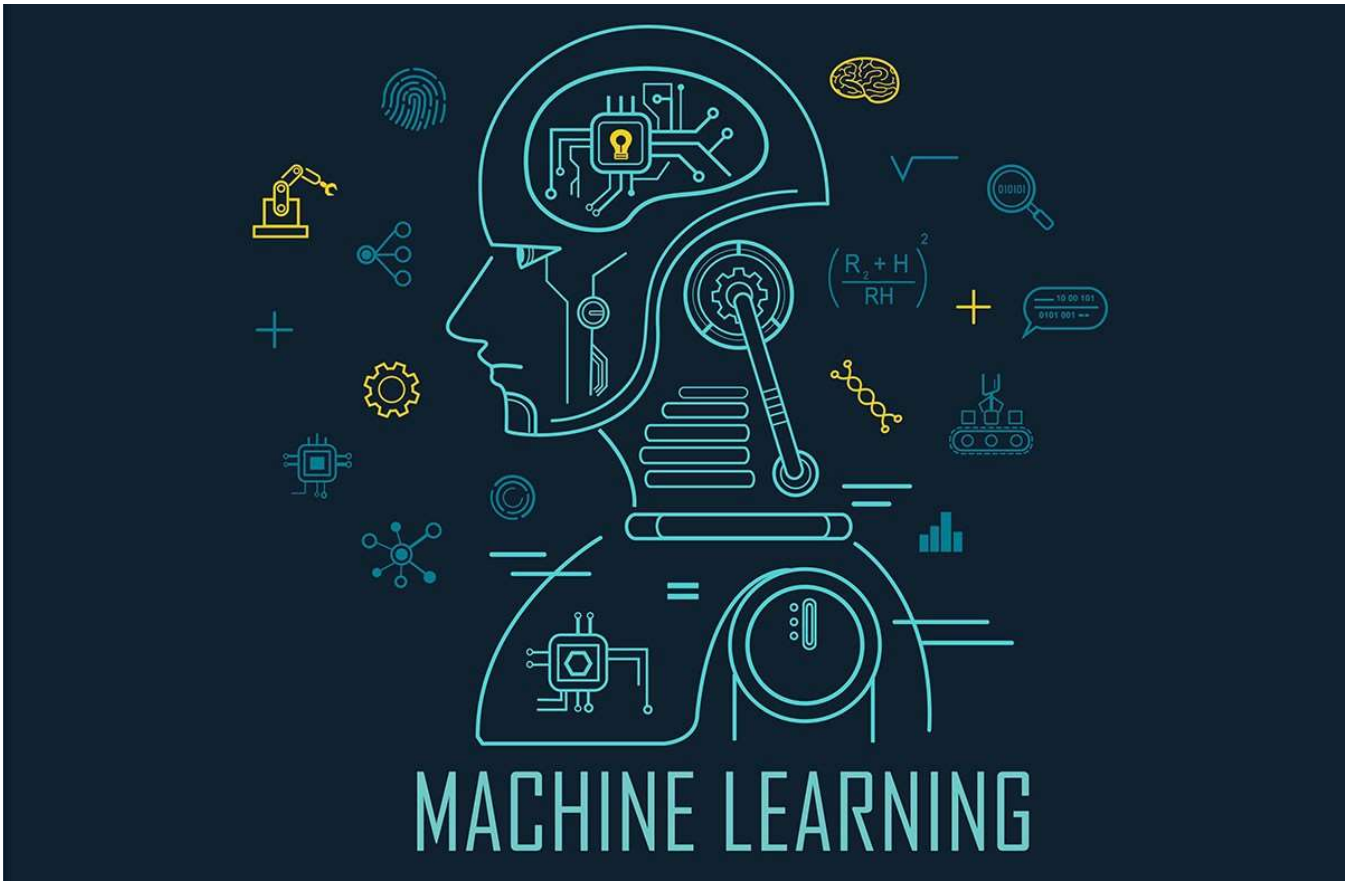
Prithvi Raj

About

Arvato Financial Services Segmentation and Predictive Modeling

Prithvi Raj 6 days ago · 21 min read



A. Project Definition

1. **Project Overview:** *

This post is based on a Capstone project for the Data Science Nano Degree program on Udacity.com platform.

For the project, I chose to work on Arvato's data. Arvato is a financial services company * based in Germany. There were two main tasks in the project:

1. Customer segmentation and profiling: Analyze company's data and similar data for Germany's general population to identify sub-segments of the German population, which are similar in characteristics to the company's own customer base
2. Predictive modeling: Predict which customers are most likely to respond to a marketing campaign to be done in future.

Both of these problems can and are solved using machine learning techniques in Python. While the 1st problem will be solved using an Unsupervised Learning technique of K-Means clustering, the 2nd problem will be solved using a binary classification technique such as logistic regression to arrive at probability to respond to marketing campaigns.

The following data files are provided in the project:

1. Customers.csv: It contained the list of customers of the company, and ~400 characteristics of them. All the attributes were demographics related
2. Azdias.csv: It contained a different list of customers. But these were not the company's customers, but belonged to the general population of Germany. The provided attributes were exactly the same as the attributes for provided for "Customers.csv" file
3. DIAS Information Levels — Attributes 2017.csv: It contained the list of the customer demographic attributes present in the above two files and their detailed descriptions
4. DIAS Attributes — Values 2017.csv: It contained the list of the customer demographic attributes, their description, and also the interpretation of each of the values taken by the ~400 attributes
5. mailout_train.csv: This file contains a list of ~42k customers and ~350 characteristics, along with a flag to indicate whether the customer responded to a marketing campaign in the past or not
6. mailout_test.csv: This file contains a new list of ~42k customers and same ~350 characteristics. Hence, this file is similar to the mailout.train.csv file above in

structure. But it does not contain any flag to indicate whether they responded to a marketing campaign or not.

2. Problem Statement

We have to solve two problems here:

1. The first problem is to segment the company's customer base and also the overall German population into multiple segments. The segments need to be created in such a manner that customers belonging to the same segment are very similar to each other, while customers belonging to different segments are very different from each other.

After the segments have been created, we need to identify sub-segments of overall German population which are very similar to the company's customer base.

We will be using K-Means clustering method to create the 5–10 customer segments. K-Means is an Unsupervised machine learning method. Further after the segments have been created, we would look at the characteristics of each of these segments, and try to assess which parts of the Germany population are similar to the company's customer base.

2. The second problem is to predict the probability to respond to a marketing campaign for a new set of customers. For this problem, we will use a binary classification technique in Python's machine learning library — sci-kit learn.

3. Metrics:

1. For the first problem of creating customer segments, we will use elbow method to arrive at the right number of clusters in the K-Means clustering technique. We will see in more detail later on what the elbow method is.

Further, we will calculate the silhouette score for different number of clusters. The number of clusters where the silhouette score is the highest will be chosen as the optimum number of clusters or segments.

2. For the second problem of predictive modeling, we will use Area Under ROC curve (AUROC) to choose the best model. The higher the value for AUROC on a test sample, would imply that that model is the best one.

B. Analysis

1. **Data Exploration:**

Let's import the datasets and understand their contents in detail.

For the problem of segmentation, we imported the "Azdias" and "Customers" datasets. Since their structure is same, we appended the datasets one below the other and created a flag to indicate the source of the data.

```
# import libraries here; add more as necessary
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# magic word for producing visualizations in notebook
%matplotlib inline

azdias.shape
```

```
(891221, 366)
```

```
customers.shape
```

```
(191652, 369)
```

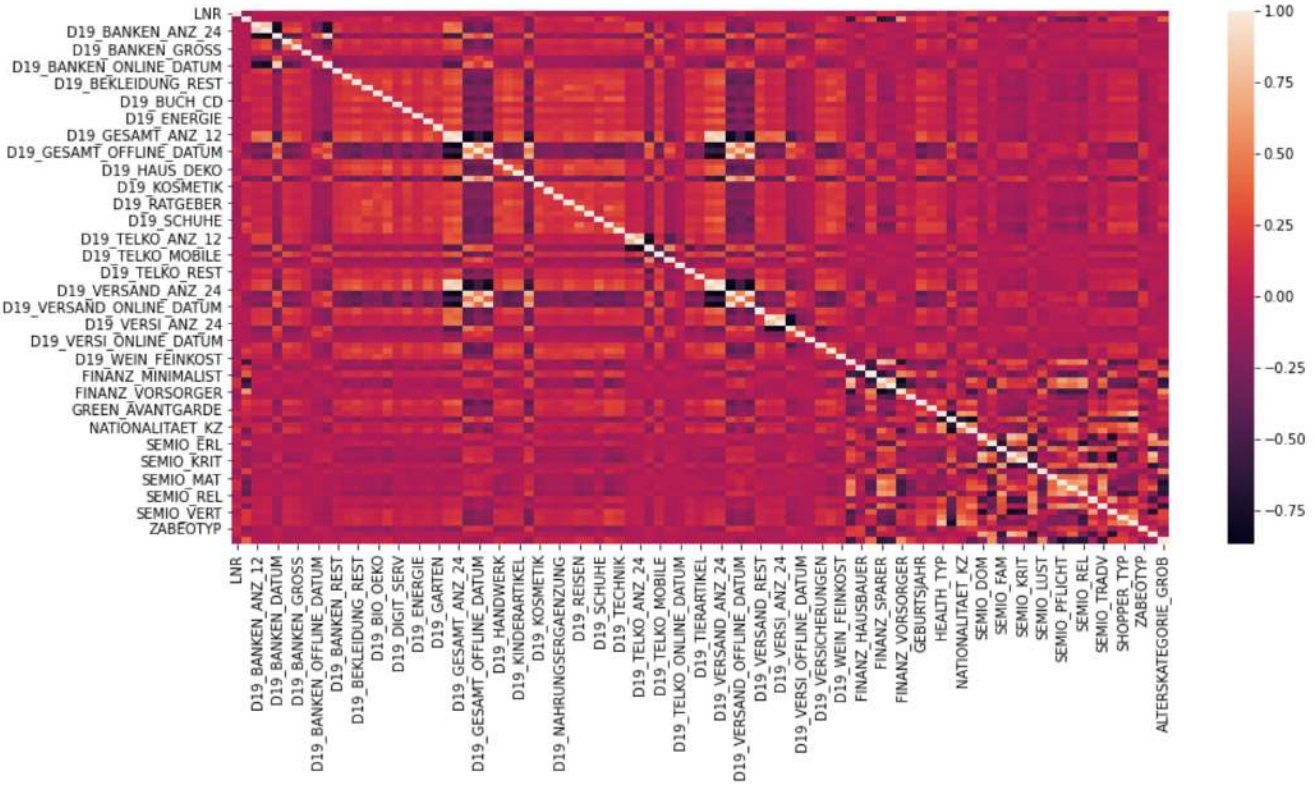
The combined dataset has the shape:

```
(1082873, 361)
```

Let's check if there is any overlap between customers in the "Azdias" and the "Customers" datasets. We find that there is no overlap between the customers in the 'azdias' dataset and the 'customers' dataset

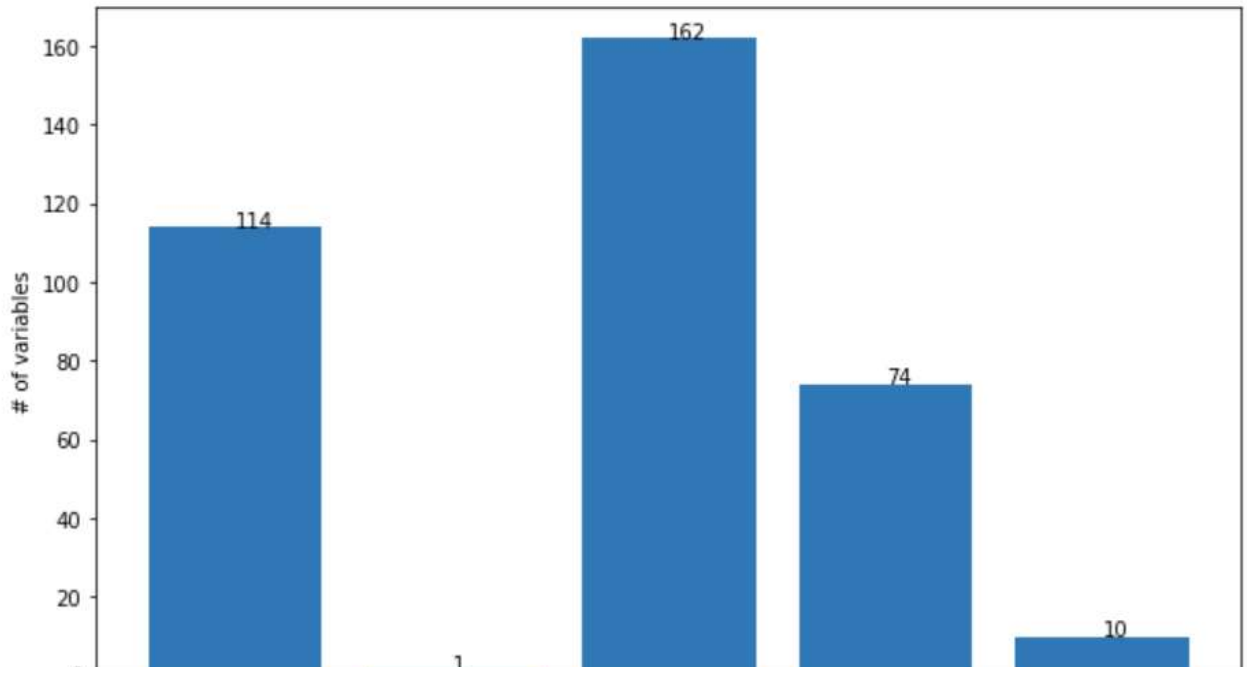
2. Data Visualization:

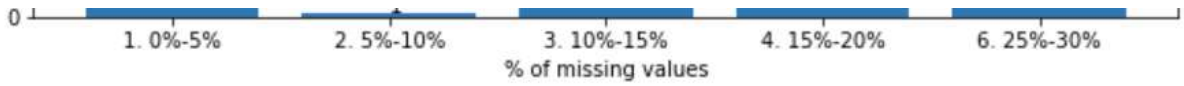
Let's check if we can get some insights into how correlated the variables are with each other:



There are too many variables to get a clear picture of correlation between them with a single correlation-matrix based heatmap, so, we will look into this aspect later while performing Principal Component Analysis (PCA)

Let's now checking the extent of missing values in the dataset. After checking the % of missing values in each of the 360 variables, we observe the below distribution:





So, out of 360 variables, only114 variables have less than 5% of missing values.

Many variables have high percentage of missing values, and replacing the missing values with mean is taking too high a time and was not converging with my laptop's memory resources, For the clustering exercise, we'll take only those variables which have all values present

```
(a['percent_miss'] == 0).sum()
```

```
94
```

So, there are only 94 variables which do not have missing values

```
(a['percent_miss'] <= 0.01).sum()
```

```
113
```

And, 113 variables with less than or equal to 1% of their values as missing

```
# Checking if a variable takes just single values for all customer ids
b = combs.nunique().reset_index()
b.columns = ['var', 'no_of_unique']

b.sort_values(by = 'no_of_unique')
```

So, no variable has same values across all rows

C. Methodology

1. **Data Preprocessing:**

KMeans algorithm cannot take categorical variables as input. So, we reduced the dataset to have only numeric variables. There is another algorithm to do clustering which can take categorical variables as input called as KPrototype, but, it was crashing

my laptop due to high memory requirements training. So, we will use only numeric variables and use KMeans instead of KPrototype.

Further, we are only keeping variables with no missing values, so that we don't have to do any missing value treatment. Again, missing values with mean or median was crashing my laptop due to memory constraint, so, we would avoid missing value treatment step by taking only those variables which have 100% of their values present.

At this stage we end up with 94 variables.

Now, for KMeans, we need to scale the variables. For this we used MinMaxScaler from sklearn.preprocessing

Here is a view of the scaled data

```
(1082873, 92)

array([[0.        , 0.        , 0.        , ..., 0.4       , 0.
,
        0.125     ],
       [0.        , 0.        , 0.        , ..., 0.8       , 1.
,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.8       , 1.
,
        0.25      ],
       ...,
       [0.75      , 0.        , 0.        , ..., 0.4       , 0.
,
        0.375     ],
       [1.        , 0.16666667, 0.16666667, ..., 0.4       , 1.
,
        0.25      ],
       [1.        , 0.16666667, 0.16666667, ..., 0.        , 0.
,
        0.125     ]])
```

Right now, the scaled dataset has 92 variables.

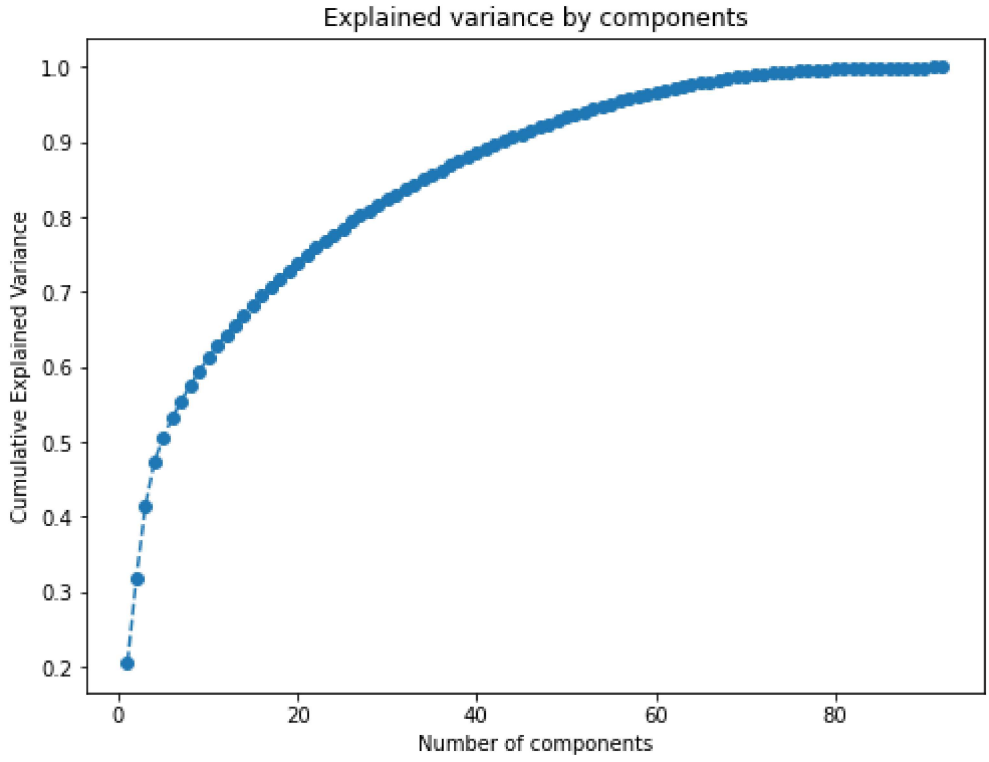
We can get better separation between clusters if we reduce the number of dimension going into the KMeans model. So, we will use Principal Component Analysis (PCA) to reduce the number of variables going as input to the KMeans program.

Using PCA, we will also be consolidating correlated variables.

```
from sklearn.decomposition import PCA
```

Let's plot the cumulative variance explained by the principal components to determine a cutoff of the number of principal components to be used in KMeans clustering

```
plt.figure(figsize = (8, 6))
plt.plot(range(1, 93), pca.explained_variance_ratio_.cumsum(), marker = 'o', linestyle = '—')
plt.title('Explained variance by components')
plt.xlabel('Number of components')
plt.ylabel('Cumulative Explained Variance');
```



Based on this Cumulative Variance plot, we need to decide on the number of Principal Components we need to keep in our model.

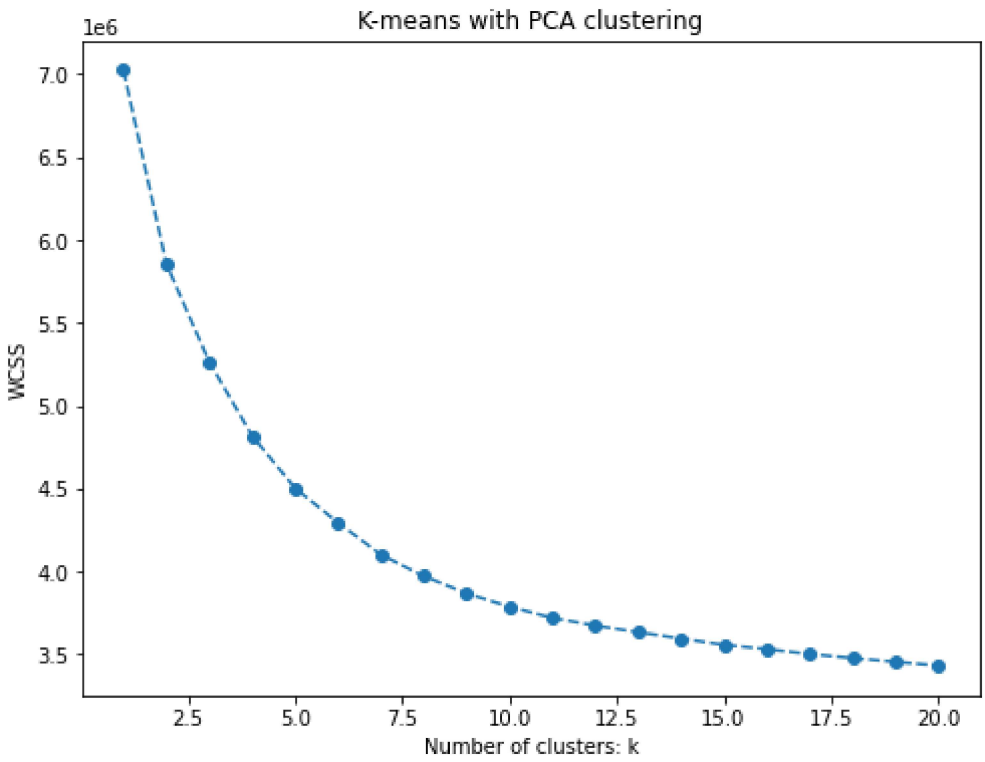
Let's take 40 components for clustering, as ~90% of the variance in the original variables is explained by 40 principal components

2. Implementation:

Next, We need to determine the optimum number of clusters for the KMeans clustering. For doing this, we will calculate Within Cluster Sum of Squares (WCSS) for clustering results with different number of clusters.

```
sse = []
for i in range(1, 21):
model = KMeans(n_clusters = i, init = 'k-means++')
model.fit(scaled_pca)
sse.append(model.inertia_)

plt.figure(figsize = (8, 6))
plt.plot(range(1, 21), sse, marker = 'o', linestyle = '--')
plt.title('K-means with PCA clustering')
plt.xlabel('Number of clusters: k')
plt.ylabel('WCSS');
```



It can be seen that as the number of clusters increase the Within Cluster Sum of Squares (WCSS) keeps on decreasing. From the above chart, we don't see a clear elbow in the curve as the WCSS value seems to decrease at the same rate as the number of clusters increase.

Hence, we can't determine the number of clusters just by this graph

3. Refinement:

Let's calculate Silhouette Score for different number of clusters. The number of clusters where the Silhouette Score is maximum can be taken as the optimum number of

clusters

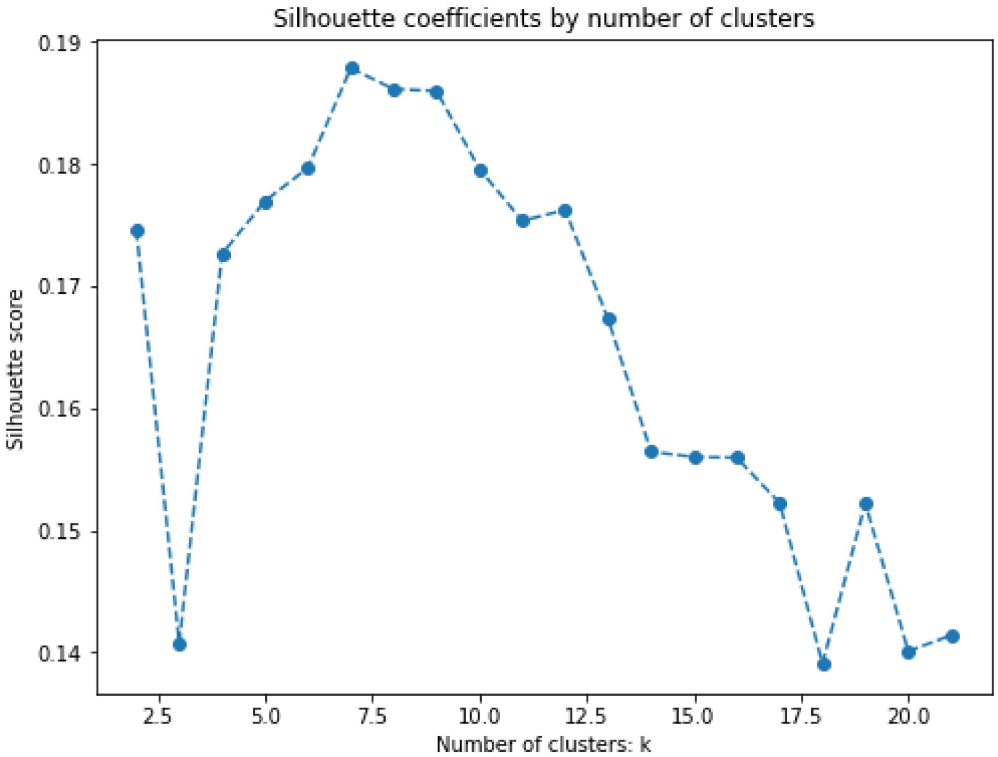
```
from sklearn.metrics import silhouette_score

sc = []
sample_size = 100000
sample = pd.DataFrame(scaled_pca).sample(n = sample_size).values

for i in range(2, 22):
model = KMeans(n_clusters = i)
model.fit(sample)
score = silhouette_score(sample, model.labels_)
sc.append(score)

plt.figure(figsize = (8, 6))
plt.plot(range(2, 22), sc, marker = 'o', linestyle = '— ')
plt.title('Silhouette coefficients by number of clusters')
plt.xlabel('Number of clusters: k')
plt.ylabel('Silhouette score');
```

We can see the the Silhouette score is highest when n = 7. So, 7 is the right number of clusters



D. Results:

1. Model Evaluation and Validation:

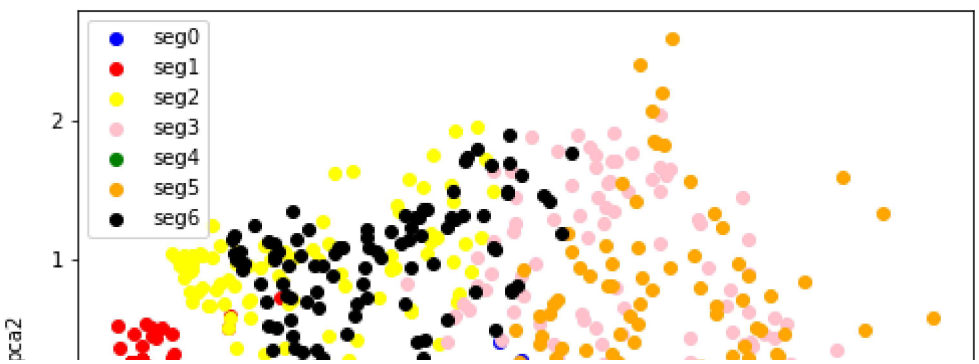
As determined above, the optimum number of clusters for the KMeans algorithm with our data is 7. So, we will run KMeans with 7 as the desired number of clusters.

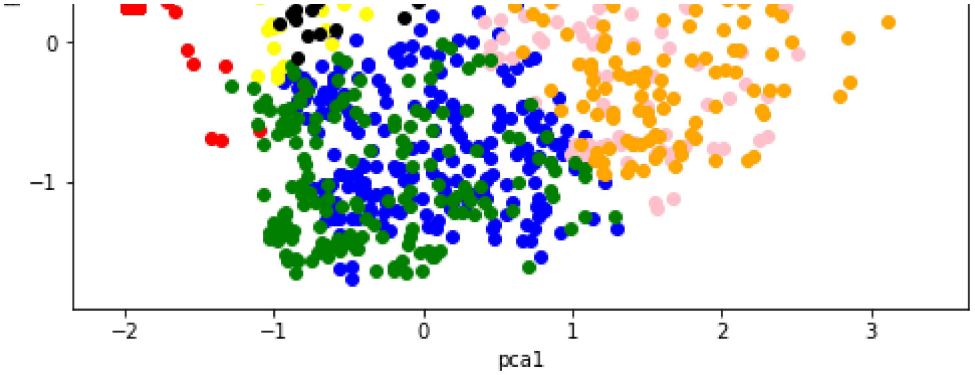
```
model = KMeans(n_clusters = 7, init = 'k-means++')
model.fit(scaled_pca)
```

Visualizing the 7 clusters using two principal components — pca1, and pca2 — on a 2-d graph

```
plt.figure(figsize = (8, 6))
plt.scatter(s[s['label'] == 0]['pca1'], s[s['label'] == 0]['pca2'], color = 'blue', label = 'seg0')
plt.scatter(s[s['label'] == 1]['pca1'], s[s['label'] == 1]['pca2'], color = 'red', label = 'seg1')
plt.scatter(s[s['label'] == 2]['pca1'], s[s['label'] == 2]['pca2'], color = 'yellow', label = 'seg2')
plt.scatter(s[s['label'] == 3]['pca1'], s[s['label'] == 3]['pca2'], color = 'pink', label = 'seg3')
plt.scatter(s[s['label'] == 4]['pca1'], s[s['label'] == 4]['pca2'], color = 'green', label = 'seg4')
plt.scatter(s[s['label'] == 5]['pca1'], s[s['label'] == 5]['pca2'], color = 'orange', label = 'seg5')
plt.scatter(s[s['label'] == 6]['pca1'], s[s['label'] == 6]['pca2'], color = 'black', label = 'seg6')

plt.xlabel('pca1')
plt.ylabel('pca2')
plt.legend();
```





Let's look at the characteristics of customers in each of these 7 clusters, and check if they are distinct from each other.

We would also analyze whether the 7 clusters are different from each other with respect to the distribution of customers from Azdias and Customers datasets

The below table shows the following:

label: Segment label from KMeans

Azdias: # of customers in the segment from "Azdias"

Customers: # of customers in the segment from "Customers"

Azdias_percent_dist: % of the "Azdias" customers in the segment

Cust_percent_dist: % of "Customers" customers in the segment

Azdias_percentage_in_seg: % of customers in the segment from "Azdias: base

Azdias_proportion_index = Azdias_percentage_in_seg/% of Azdias in overall audience base

	label	azdias	customers	azdias_percent_dist	cust_percent_dist	azdias_perentage_in_seg	Azdias_proportion_index
1	0	166503	40536	18.6826	21.1508	80.4211	0.977152
2	1	96065	47041	10.779	24.545	67.1286	0.815642
3	2	116306	1286	13.0502	0.671008	98.9064	1.20176
4	3	127179	20332	14.2702	10.6088	86.2166	1.04757
5	4	166222	26171	18.651	13.6555	86.3971	1.04976
6	5	106826	54182	11.9865	28.271	66.3483	0.806161
7	6	112120	2104	12.5805	1.09782	98.158	1.19266

As we have combined the “Azdias” and “Customers” users into the KMeans exercise, overall “Azdias” proportion in the combined data is: 82.3%

as calculated below:

> *(pt[‘azdias’].sum()/(pt[‘azdias’].sum()+pt[‘customers’].sum()))*100*

```
82.30152566367431
```

It can be seen that compared to overall user base, “Azdias” proportion is much higher in segments labeled 2, and 6, and, much lower in segments labeled 1, and 5

Also, the segments labeled 0, 3, 4 have the same proportion of “Azdias” as in the overall population

Further, it can be interpreted that since segments 2 and 6 are “Azdias” dominant, users belonging to them are likely to be very different from users from “Customers”

While, since segments 1 and 5 are “Customers” dominant, users belonging to them are likely to be very similar to “Customers” type of users. Hence, we can interpret that “Azdias” users belonging to these segments are likely to be very similar to ‘Customers’

Now, let’s look at the characteristics of each of these segments

	var	0	1	2	3	4	5	6	overall_mean	index_0	index_1	index_2	index_3	index_4	index_5	index_6	max_index	azdias_mean	customers_mean
1	D19_VERSAND_ANZ_12	0.094697	0.010768	0.182300	2.044505	0.112951	1.804426	0.264936	0.634138	0.149332	0.016981	0.287477	3.224069	0.178118	2.845478	0.417788	3.224069	0.604646	0.771283
2	D19_VERSAND_ANZ_24	0.228759	0.020558	0.393649	3.025002	0.260727	2.700176	0.507976	1.002657	0.228153	0.020504	0.392606	3.016986	0.260036	2.693022	0.506630	3.016986	0.958868	1.20629
3	D19_GESAMT_ANZ_12	0.194422	0.017735	0.276405	2.449621	0.187164	2.257844	0.476607	0.822460	0.236391	0.021563	0.336071	2.978406	0.227566	2.745232	0.579490	2.978406	0.797683	0.937679
4	D19_TIERARTIKEL	0.060114	0.004815	0.154509	0.711994	0.076541	0.599268	0.123223	0.241597	0.248820	0.019928	0.639531	2.947032	0.316814	2.480445	0.510034	2.947032	0.243178	0.234247
5	D19_SCHUHE	0.114645	0.013053	0.386268	1.532150	0.171768	1.271123	0.308762	0.526388	0.217796	0.024798	0.733809	2.910688	0.326315	2.414804	0.586567	2.910688	0.510818	0.598788
6	D19_DROGERIEARTIKEL	0.168582	0.015171	0.445932	1.986089	0.217030	1.739143	0.359565	0.688284	0.244931	0.022041	0.647889	2.885567	0.315320	2.526782	0.522409	2.885567	0.673292	0.757999
7	D19_LEBENSMITTEL	0.154198	0.008819	0.091180	0.899180	0.173114	1.047699	0.148218	0.365206	0.422222	0.024147	0.249666	2.462117	0.474018	2.868788	0.405846	2.868788	0.319525	0.577636
8	D19_BIO_OEKO	0.125527	0.006163	0.069690	0.789277	0.168436	0.875708	0.071684	0.307592	0.408096	0.020037	0.226567	2.565986	0.547597	2.846980	0.233048	2.846980	0.257938	0.538492
9	D19_BANKEN_ANZ_12	0.023189	0.002893	0.038931	0.325054	0.017152	0.330437	0.107079	0.116797	0.198540	0.024769	0.333325	2.783073	0.146857	2.829164	0.916799	2.829164	0.122336	0.09104
10	D19_FREIZEIT	0.145127	0.011949	0.342966	1.687332	0.118882	1.656818	0.417959	0.607976	0.238705	0.019654	0.564110	2.775325	0.195537	2.725137	0.687460	2.775325	0.590678	0.688414

> *Based on these above variable index, the following can be interpreted*
> *Segment profiles:*
> ***# Segment 0:***
> *# No transaction known or very low mail-order transactions in last 12 months*
> *# No transaction known or very low mail-order transactions in last 24 months*
> *# No or very low “Total Pool” transactions in last 12 months*
> *# No or very low “Animal Products” transactions*

 # No or very low "Shoes" transactionns
 # No or very low "Drugstores" transactions
 # No or very low "Food Products" transactions
 # No or very low "Ecologicals" transactions
 # No or very low "Banks" transactions in last 12 months
 # No or very low "Leisure Products" transactions

 # Segment 1:
 # No transaction known or extremely low mail-order transactions in last 12 months
 # No transaction known or extremely low mail-order transactions in last 24 months
 # No or extremely low "Total Pool" transactions in last 12 months
 # No or extremely low "Animal Products" transactions
 # No or extremely low "Shoes" transactionns
 # No or extremely low "Drugstores" transactions
 # No or extremely low "Food Products" transactions
 # No or extremely low "Ecologicals" transactions
 # No or extremely low "Banks" transactions in last 12 months
 # No or extremely low "Leisure Products" transactions

 # Segment 2:
 # No transaction known or very low mail-order transactions in last 12 months
 # No transaction known or very low mail-order transactions in last 24 months
 # No or very low "Total Pool" transactions in last 12 months
 # No or very low "Animal Products" transactions
 # No or very low "Shoes" transactionns
 # No or very low "Drugstores" transactions
 # No or very low "Food Products" transactions
 # No or very low "Ecologicals" transactions
 # No or very low "Banks" transactions in last 12 months
 # No or very low "Leisure Products" transactions

 # Segment 3:
 # Very high "mail-order" transactions in last 12 months
 # Very high "mail-order" transactions in last 24 months
 # Very high "Total Pool" transactions in last 12 months
 # Very high "Animal Products" transactions
 # Very high "Shoes" transactionns

 # Very high "Drugstores" transactions
 # Very high "Food Products" transactions
 # Very high "Ecologicals" transactions
 # Very high "Banks" transactions in last 12 months
 # Very high "Leisure Products" transactions

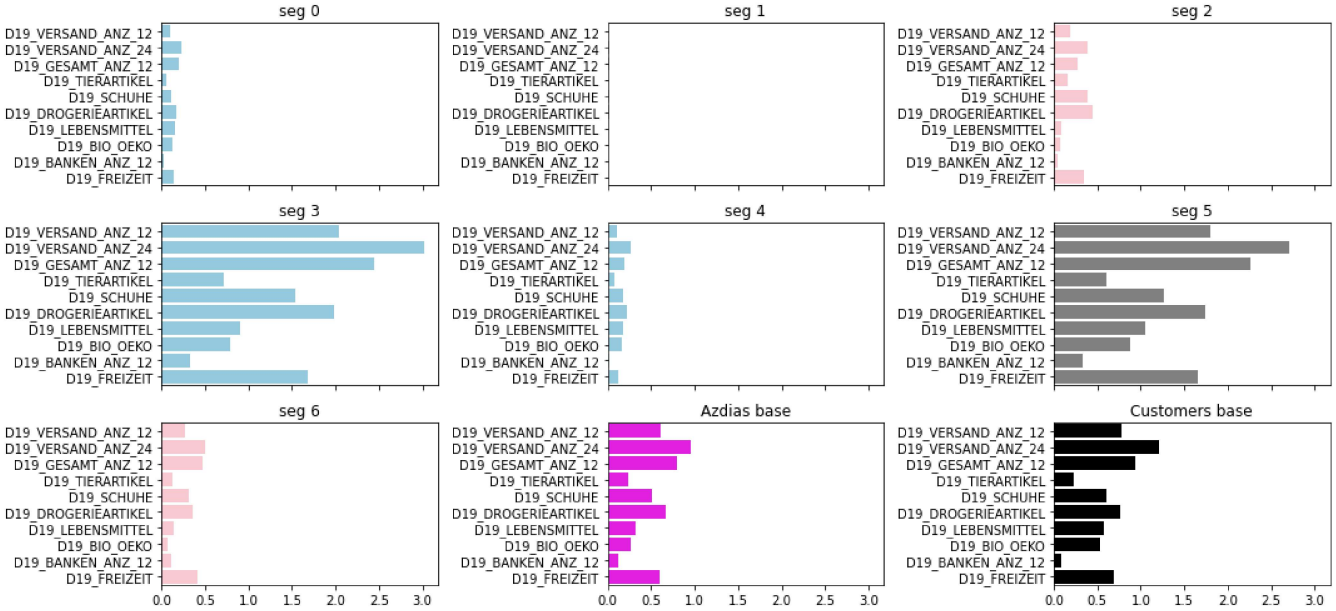
 # Segment 4:
 # No transaction known or very low mail-order transactions in last 12 months
 # No transaction known or very low mail-order transactions in last 24 months
 # No or very low "Total Pool" transactions in last 12 months
 # No or very low "Animal Products" transactions
 # No or very low "Shoes" transactionns
 # No or very low "Drugstores" transactions
 # No or very low "Food Products" transactions
 # No or very low "Ecologicals" transactions
 # No or very low "Banks" transactions in last 12 months
 # No or very low "Leisure Products" transactions

 # Segment 5:
 # Very high "mail-order" transactions in last 12 months
 # Very high "mail-order" transactions in last 24 months
 # Very high "Total Pool" transactions in last 12 months
 # Very high "Animal Products" transactions
 # Very high "Shoes" transactionns
 # Very high "Drugstores" transactions
 # Very high "Food Products" transactions
 # Very high "Ecologicals" transactions
 # Very high "Banks" transactions in last 12 months
 # Very high "Leisure Products" transactions

 # Segment 6:
 # No transaction known or low mail-order transactions in last 12 months
 # No transaction known or low mail-order transactions in last 24 months
 # No or low "Total Pool" transactions in last 12 months
 # No or low "Animal Products" transactions
 # No or low "Shoes" transactionns
 # No or low "Drugstores" transactions

```
# No or low "Food Products" transactions
# No or low "Ecologicals" transactions
# No or low "Banks" transactions in last 12 months
# No or low "Leisure Products" transactions
```

Let's compare these 7 segments with respect to these above variables in charts form. We will keep the scale to be same for each segment's plot so that the values can be compared between them



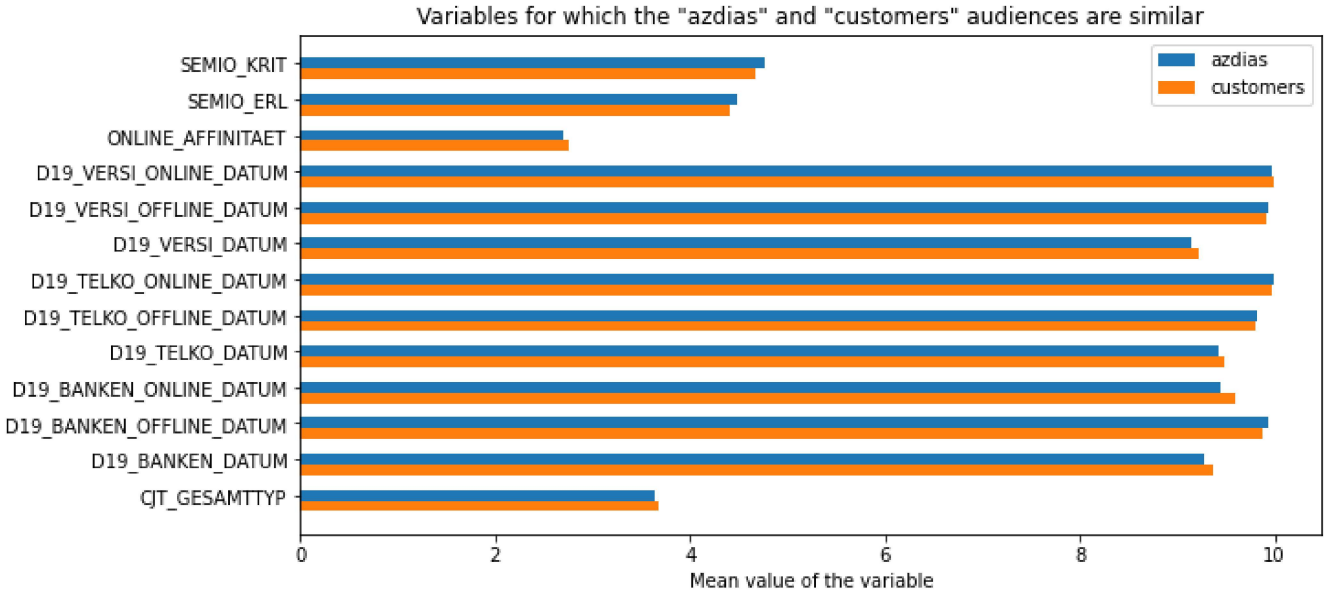
Now, let's now look at the differentiating characteristics of the Overall population and the customer base of the company. This will help us in knowing these things:
1. What is the profile of a typical customer of the company
2. What is the profile of a typical German consumer as present in the Azdias data
3. How are the two populations different from each other

Let's see on which parameters the overall population and the company's customers are similar

```
plt.figure(figsize = (10, 5))
plt.barh(y = np.arange(ft_same.shape[0]), width = ft_same['azdias'], height = 0.3,
label = 'azdias')
plt.yticks(np.arange(ft_same.shape[0]), ft_same['var'])

plt.barh(y = np.arange(ft_same.shape[0]) — 0.3, width = ft_same['customers'], height
= 0.3, label = 'customers')
plt.xlabel('Mean value of the variable')
```

```
plt.title('Variables for which the "azdias" and "customers" audiences are similar')
plt.legend();
```



Hence, the factors on which the two audiences are very similar are:

 # CJT_GESAMTTYP — customer journey typology

 # D19_BANKEN_DATUM — actuality of the last transaction for the segment banks TOTAL

 # D19_BANKEN_OFFLINE_DATUM — actuality of the last transaction for the segment banks OFFLINE

 # D19_BANKEN_ONLINE_DATUM —

 # D19_TELKO_DATUM — actuality of the last transaction for the segment telecommunication TOTAL

 # D19_TELKO_OFFLINE_DATUM — actuality of the last transaction for the segment telecommunication OFFLINE

 # D19_TELKO_ONLINE_DATUM — actuality of the last transaction for the segment telecommunication ONLINE

 # D19_VERSI_DATUM — actuality of the last transaction for the segment insurance TOTAL

 # D19_VERSI_OFFLINE_DATUM — actuality of the last transaction for the segment insurance OFFLINE

 # D19_VERSI_ONLINE_DATUM — actuality of the last transaction for the segment insurance ONLINE

 # ONLINE_AFFINITAET — ONLINE_AFFINITAET

 # SEMIO_ERL — affinity indicating in what way the person is eventful orientated

 # SEMIO_KRIT — affinity indicating in what way the person is critical minded

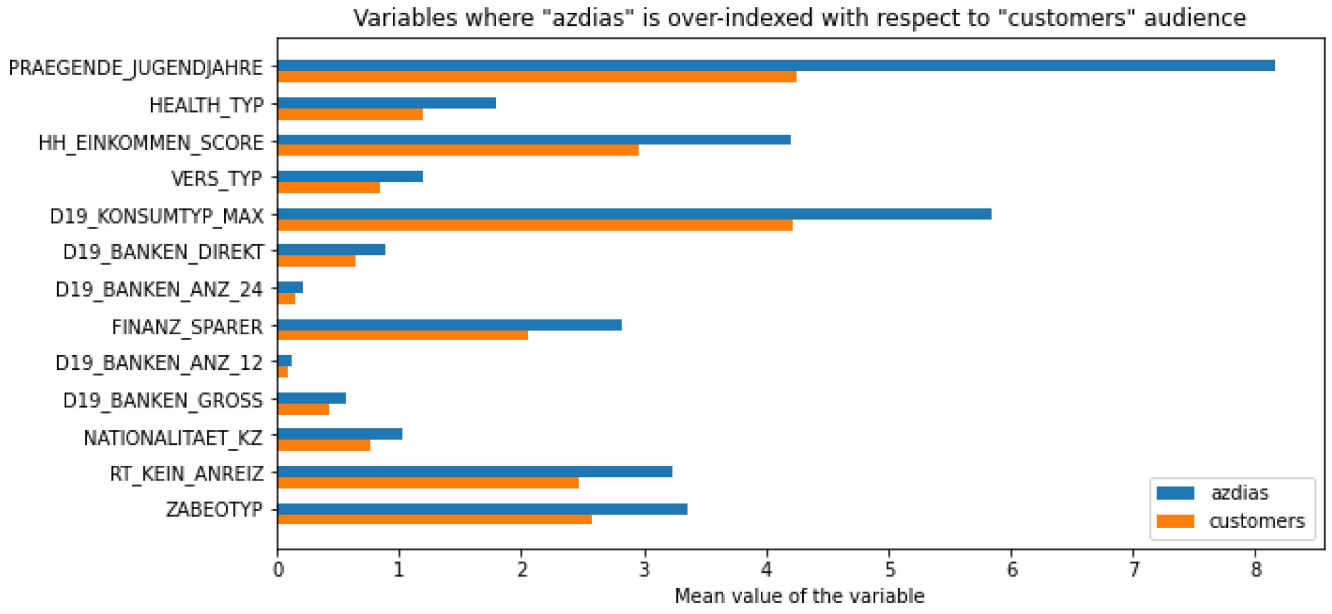
So, in summary, the audiences are similar with respect to banking, and insurance transactions majorly

Now, let's see on what parameters there is stark difference between the two groups.

First, let's see which variables have "Azdias" overindexed on "Customers"

```
plt.figure(figsize = (10, 5))
plt.barh(y = np.arange(ft_diff1.shape[0]), width = ft_diff1['azdias'], height = 0.3, label = 'azdias')
plt.yticks(np.arange(ft_diff1.shape[0]), ft_diff1['var'])

plt.barh(y = np.arange(ft_diff1.shape[0]) — 0.3, width = ft_diff1['customers'], height = 0.3, label = 'customers')
plt.xlabel('Mean value of the variable')
plt.title('Variables where "azdias" is over-indexed with respect to "customers" audience')
plt.legend();
```



Hence, the variables which have higher values for "Azdias" compared to "Customers" are:

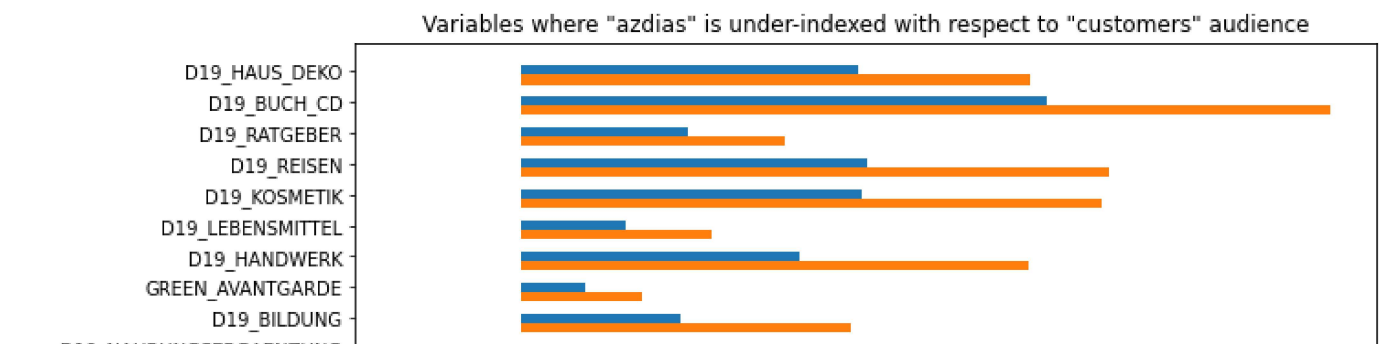
```
# ZABEOTYP — typification of energy consumers ("Azdias" are more price driven; while "customers" are more "green" driven)
# RT_KEIN_ANREIZ —
# NATIONALITAET_KZ — nationaltity (scored by prename analysis) ("Azdias" have more proportion of foreigners, while "customers" are more German)
```

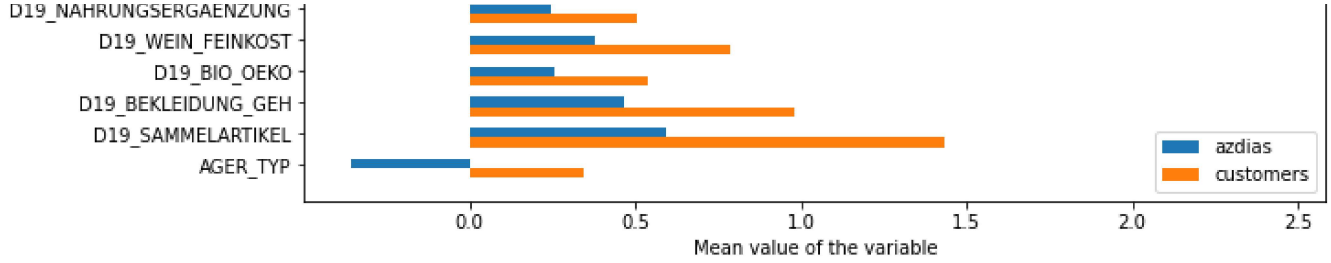
```
# D19_BANKEN_GROSS — transactional activity based on the product group BIG BANKS (“Azdias” has more transaction activity for “Big banks”)
# D19_BANKEN_ANZ_12 — transaction activity BANKS in the last 12 months (“Azdias” have increased banking activity in the last 12 months compared to “customers”)
# FINANZ_SPARER — financial typology: money saver (“Azdias” are low money savers, while “customers” are high money savers)
# D19_BANKEN_ANZ_24 — transaction activity BANKS in the last 24 months (“Azdias” have increased banking activity in the last 24 months much more than “customers”)
# D19_BANKEN_DIREKT — transactional activity based on the product group DIRECT BANKS (“Azdias” has more transaction activity for “Direct Banks”)
# D19_KONSUMTYP_MAX —
# VERS_TYP — insurance typology (“Azdias” has more risky behavior than “customers”)
# HH_EINKOMMEN_SCORE — estimated household net income (“Azdias” household income is lower than “customers”)
# HEALTH_TYP — health typology (“Azdias” are less healthy than “customers”)
# PRAEGENDE_JUGENDJAHRE — dominating movement in the person’s youth (avantgarde or mainstream) (“Azdias” are younger than “customers”)
```

Now, let’s see which variables are under indexed with respected to “Azdias” versus “Customers”

```
plt.figure(figsize = (10, 5))
plt.barh(y = np.arange(ft_diff2.shape[0]), width = ft_diff2['azdias'], height = 0.3, label = 'azdias')
plt.yticks(np.arange(ft_diff2.shape[0]), ft_diff2['var'])

plt.barh(y = np.arange(ft_diff2.shape[0]) — 0.3, width = ft_diff2['customers'], height = 0.3, label = 'customers')
plt.xlabel('Mean value of the variable')
plt.title('Variables where “azdias” is under-indexed with respect to “customers” audience')
plt.legend();
```



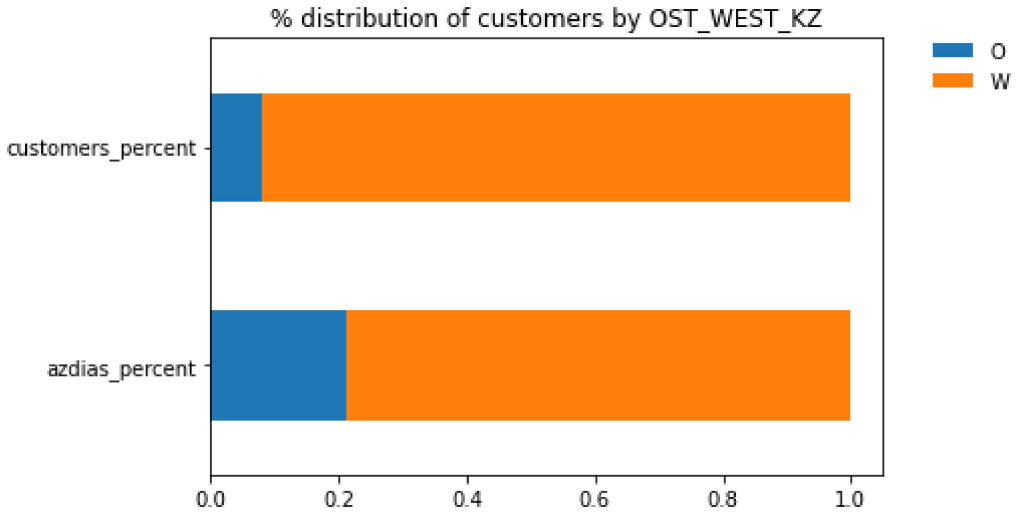


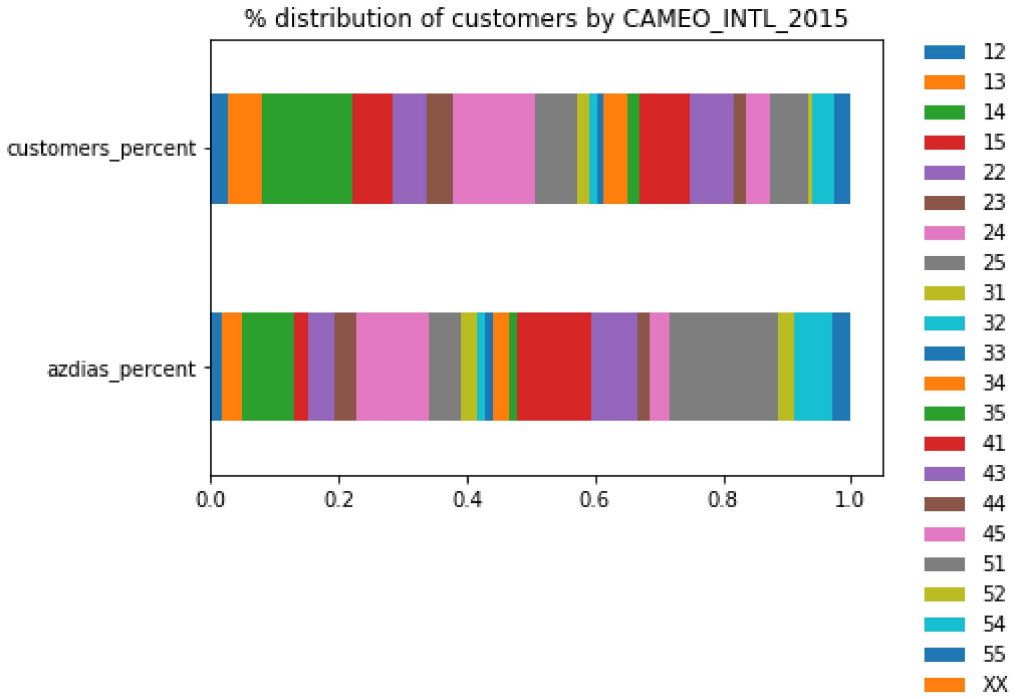
Here is the list of such variables:

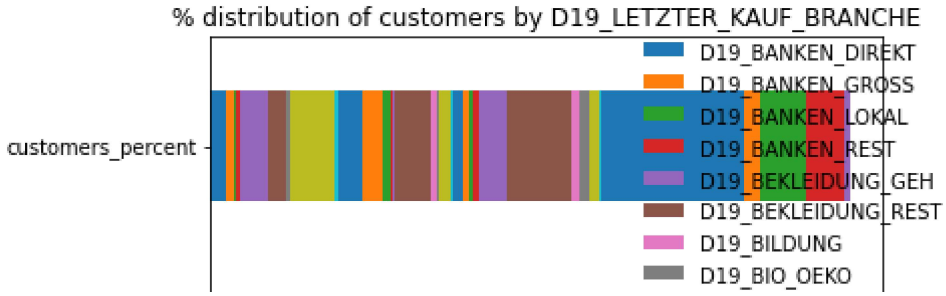
> *# AGER_TYP — best-ager typology (Age if "Azdias" is not well known, while age of "customers" is well known or elderly)*
> *# D19_SAMMELARTIKEL — transactional activity based on the product group COLLECTABLE ITEMS ("Azdias" has less transaction activity for Collectable items)*
> *# D19_BEKLEIDUNG_GEH — transactional activity based on the product group LUXURY CLOTHING ("Azdias" has less transaction activity for "Luxury" items)*
> *# D19_BIO_OEKO — transactional activity based on the product group ECOLOGICALS ("Azdias" has less transaction activity for "Ecologicals")*
> *# D19_WEIN_FEINKOST — transactional activity based on the product group WINE AND GOURMET FOOD ("Azdias" has less transaction activity for "Wine and Gourmet Foods")*
> *# D19_NAHRUNGSERGAENZUNG — transactional activity based on the product group DIETARY SUPPLEMENTS ("Azdias" has less transaction activity for "Dietary supplements")*
> *# D19_BILDUNG — transactional activity based on the product group EDUCATION ("Azdias" has less transaction activity for "Education" related products)*
> *# GREEN_AVANTGARDE — the environmental sustainability is the dominating movement in the youth of these consumers ("Azdias" has less score for "Green avantgarde")*
> *# D19_HANDWERK — transactional activity based on the product group DO-IT-YOURSELF PRODUCTS ("Azdias" has less transaction activity for "Do-it yourself" products)*
> *# D19_LEBENSMITTEL — transactional activity based on the product group FOOD PRODUCTS ("Azdias" has less transaction activity for "Food products")*
> *# D19_KOSMETIK — transactional activity based on the product group COSMETIC PRODUCTS ("Azdias" has less transaction activity for "Cosmetic" products)*
> *# D19_REISEN — transactional activity based on the product group TRAVEL RELATED PRODUCTS ("Azdias" has less transaction activity for "Travel related products")*
> *# D19_RATGEBER — transactional activity based on the product group GUIDEBOOKS ("Azdias" has less transaction activity for "Guidebooks")*

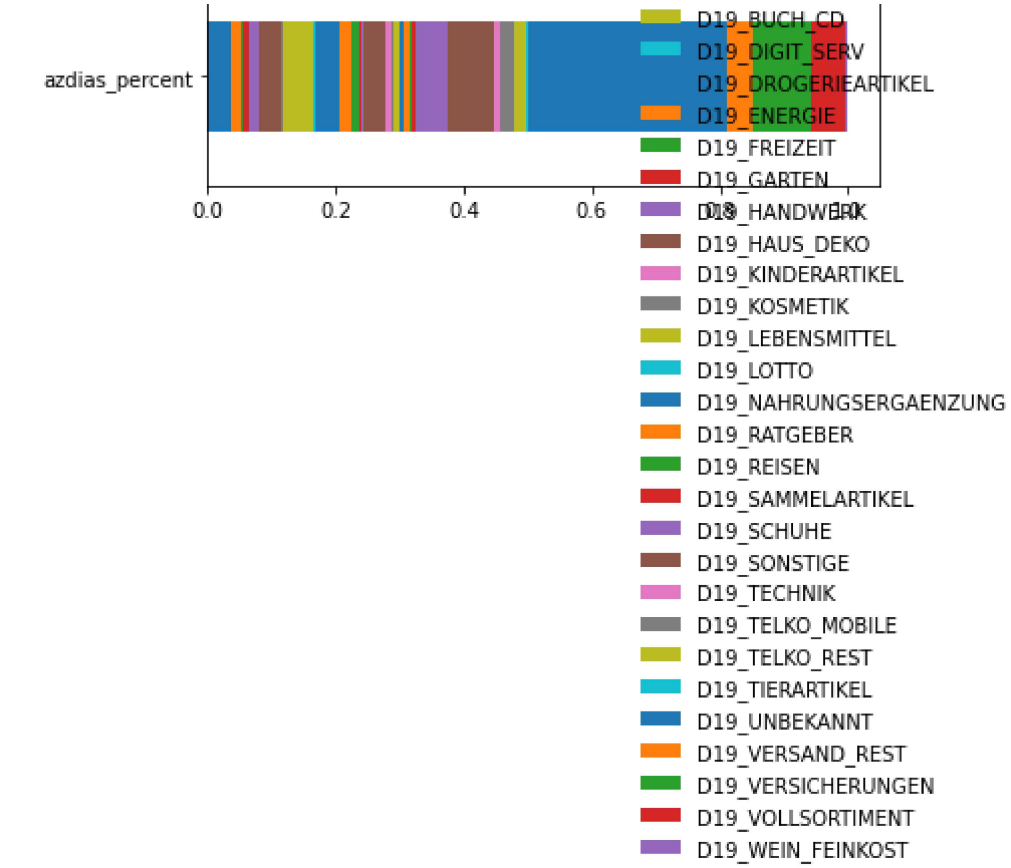
> *# D19_BUCH_CD*
> *# D19_HAUS_DEKO — transactional activity based on the product group HOUSE DECORATION (“Azdias” has less transaction activity for “House decoration” items)*

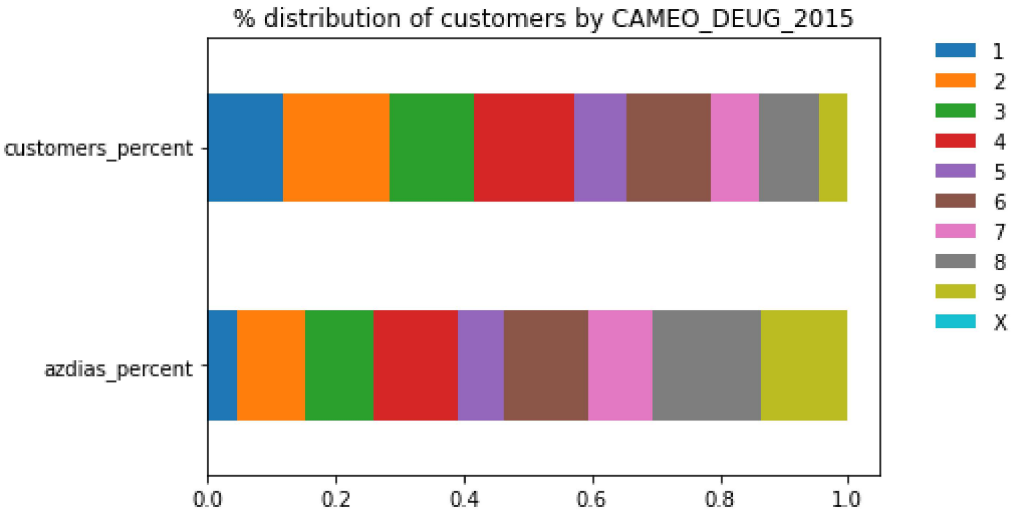
Till now we have profied “Azdias” and “Customers” users using only numeric variables. Now, let’s look at the profiles of Azdias and Customers using character variables

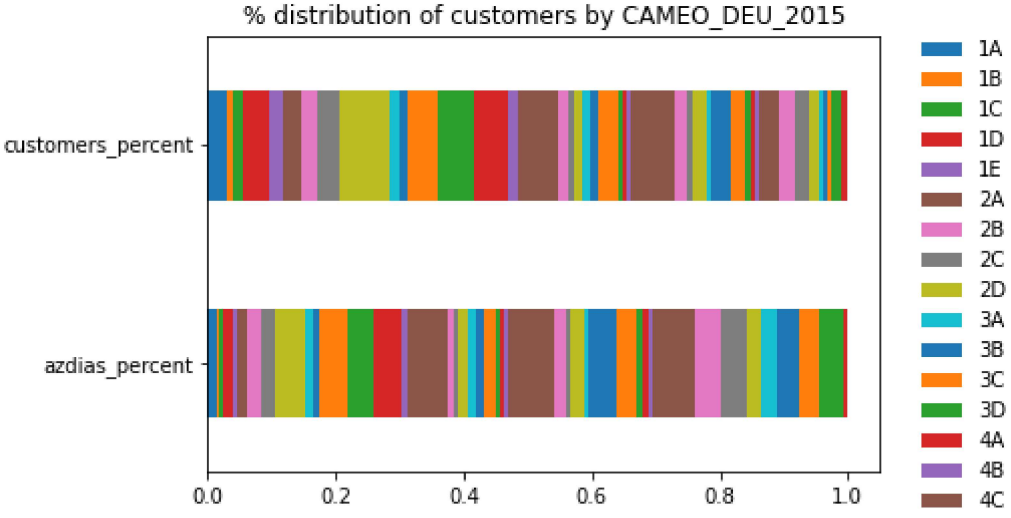







azdias_percent
0.0
0.2
0.4
0.6
D19_BUCH_CD
D19_DIGIT_SERV
D19_DROGERIEARTIKEL
D19_ENERGIE
D19_FREIZEIT
D19_GARTEN
D19_HANDWERK
D19_HAUS_DEKO
D19_KINDERARTIKEL
D19_KOSMETIK
D19_LEBENSMITTEL
D19_LOTTO
D19_NAHRUNGSERGAENZUNG
D19_RATGEBER
D19_REISEN
D19_SAMMELARTIKEL
D19_SCHUHE
D19_SONSTIGE
D19_TECHNIK
D19_TELKO_MOBILE
D19_TELKO_REST
D19_TIERARTIKEL
D19_UNBEKANNT
D19_VERSAND_REST
D19_VERSICHERUNGEN
D19_VOLLSORTIMENT
D19_WEIN_FEINKOST


% distribution of customers by CAMEO_DEUG_2015
customers_percent
azdias_percent
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5
6
7
8
9
X


% distribution of customers by CAMEO_DEU_2015
customers_percent
azdias_percent
0.0
0.2
0.4
0.6
0.8
1.0
1A
1B
1C
1D
1E
2A
2B
2C
2D
3A
3B
3C
3D
4A
4B
4C

Based on these charts, the following observations can be made:
1. “Customers” have a higher proportion of users from “West (FRG)” compared to “Azdias”
2. “Customers” have a higher proportion of wealthy users compared to “Azdias”
3. “Customers” have a higher proportion of upper and upper middle class users compared to “Azdias”

Next we will work on the 2nd project, which is the predictive modeling exercise to predict which customers are most likely to respond to a marketing campaign.

A. Project Definition

1. **Metrics:**

Since this is classification problem, we can use accuracy as a metric to evaluate how good the fit of the model is. But, we noticed that data is highly imbalanced. That is, the number of responders is very low compared to the number of non-responders. Specifically, the response rate (#responders/# total customers) = 532/42,962

= 1.2%

So, in that case, even if the model predicts all customers to be non-responders, which is obviously incorrect, the accuracy would still be very high (42430/42962) = 98.8%

So, a confusion matrix based accuracy score is not the right metric for such an imbalanced data.

Instead we will use Area Under ROC curve as the metric to measure the performance of the model (To read more about confusion matrix, and Area Under ROC curve, you can read here).

For the AUROC curve calculation, we will not predict the actual 1 or 0 flag for every customer, but we will only predict the probability to respond. This probability will taken values between 0 and 1.

B. Analysis

1. **Data Exploration:**

Two datasets are provided for this exercise — train and test — with these shapes:

```
train: (42962, 368)

test: (42833, 367)
```

There is one column less in the test data, because it does not contain the "response" variable, which needs to be predicted.

Further, several variables have large percentage of their values missing. We will only use those variables in the model which have no more than 20% of their values as missing.

```
a = pd.DataFrame(train.isnull().sum()/train.shape[0]).reset_index()
a.columns = ['var', 'per_miss']
(a['per_miss'] <= 0.2).sum()
```

```
295
```

With that logic, we will have 295 variables instead of the original 368 in the train dataset.

Out of these 295 variables, 287 variables are numeric, while the remaining 6 variables are categorical.

```
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
numeric_data = X.select_dtypes(include = numerics)

numeric_data.shape
```

```
(42962, 287)
```

C. Methodology

1. **Data preprocessing:**

Before training the model, the "train" dataset created above with 287 variables will be need to be divided into "training" and "testing" samples. The "testing" sample will be used for evaluating the model on a new dataset.

We will keep 20% of the samples for testing and 80% for training.

Further, we will drop date variables from the modeling exercise.

```
# Dropping the date variable
char = char.drop('EINGEFUEGT_AM', axis = 1)
```

We will use train_test_split function to create the training and testing datasets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state=42)
```

We will need to perform missing value treatment for the variables with this approach for each variable type:

1. Numeric variables: Replace missing values with mean
2. Categorical variables: Replace missing values with "Missing" word

Next, the numeric variables need to be scaled. We will use MinMaxScaler for this.

```
scaler = MinMaxScaler()
num = scaler.fit_transform(num)
```

While the numeric variables can be directly used in any machine learning classification model, the categorical variables need to be converted into numeric variables.

```
char = pd.get_dummies(char)
```

Next, all the numeric and the dummy variables created from categorical variables will be combined together.

```
# Combining numeric and character variables again
X_train = pd.concat([k, char.reset_index(drop = True)], axis = 1)
```

We will need to perform the same data-preprocessing steps on the X_test data as well.

2. Implementation:

First, we will use the simple machine learning technique of logistic regression to train the model and check the AUROC curve metric. This model will be the benchmark model which we will improve on.

```
lr_model = LogisticRegression(max_iter = 200)
lr_model.fit(X_train, y_train)

pred = lr_model.predict_proba(X_test)[:,1]
roc_auc_score(y_test, pred)
```

```
0.6798554246292932
```

AUROC curve of 67% is achieved on test data using logistic regression

Next, let's use Random Forest technique to train the model.

```
rf_model = RandomForestClassifier()

rf_model.fit(X_train, y_train)

pred = rf_model.predict_proba(X_test)[:,1:]
roc_auc_score(y_test, pred)
```

```
0.6270048818331397
```

Here, the AUROC value is actually lower than what was observed for the logistic regression model.

3. Refinement:

Now, let's try several other machine learning techniques and compare their performance without performing any hyperparameter tuning in each of the techniques. Basically, we will use the default hyper parameters for now for each of the machine learning techniques.

We tried the following machine learning methods:

- Logistic Regression
- Random Forest
- Decision Tree
- Support Vector Machine (SVM)
- Gradient Boosted Tree
- XGBoost
- Neural Network
- KNearest Neighbor
- NaiveBayes
- Stochastic Graident Descent
- AdaBoost
- LightGBM

Here is the a comparison of each of these techniques in terms of testing sample's AUROC:

	method	training_AUROC	testing_AUROC	duration

4	([DecisionTreeRegressor(criterion='friedman_ms...	0.91	0.75	122
10	(DecisionTreeClassifier(max_depth=1, random_st...	0.84	0.73	34
9	SGDClassifier(loss='log')	0.79	0.69	2
11	LGBMClassifier()	0.99	0.69	3
0	LogisticRegression(max_iter=500)	0.81	0.68	13
5	XGBClassifier(base_score=0.5, booster='gbtree'...	0.99	0.68	21
1	(DecisionTreeClassifier(max_features='auto', r...	0.98	0.60	16
3	SVC(probability=True)	0.97	0.59	2612
8	GaussianNB()	0.59	0.56	1
6	MLPClassifier()	0.99	0.55	141
7	KNeighborsClassifier()	0.91	0.50	2358
2	DecisionTreeClassifier()	0.99	0.49	4

The above table is sorted in decreasing order of the “testing_AUROC” values. So, it can be seen that the best performance is achieved using Gradient Boosted Tree model.

So, we will further improve this performance (testing AUROC = 75%) by tuning hyper parameters of the model.

D. Results

4. Model Evaluation and Validation:

In the beginning, I tried hyper parameter tuning using Grid Search Cross Validation, with ~100 models in the resulting grid. But, it did not converge in ~12 hours.

So, I tried hyper parameter tuning using Randomized Search Cross Validation with these parameters:

```
rscv = RandomizedSearchCV(GradientBoostingClassifier(),
scoring = 'roc_auc',
param_distributions = {
'learning_rate': [0.01, 0.005],
'n_estimators': [200, 300, 500],
'max_depth': [5, 6, 7],
'subsample': [0.4, 0.5, 0.6, 0.7],
'min_samples_leaf': [1, 2, 3, 4, 5],
'min_samples_split': [2, 3, 4, 5]
},
```

```
cv = 5,
n_iter = 10
)
```

And, there is top 6 models from the Grid Search:

	params	mean_test_score	rank_test_score
5	{'subsample': 0.4, 'n_estimators': 300, 'min_s...	0.772169	1
7	{'subsample': 0.6, 'n_estimators': 500, 'min_s...	0.770521	2
9	{'subsample': 0.5, 'n_estimators': 200, 'min_s...	0.770059	3
4	{'subsample': 0.7, 'n_estimators': 300, 'min_s...	0.769989	4
8	{'subsample': 0.4, 'n_estimators': 200, 'min_s...	0.768258	5
1	{'subsample': 0.7, 'n_estimators': 300, 'min_s...	0.767623	6

Here the best model's test score is: 77.2%

The parameters for the best model are:

```
{'subsample': 0.4,
 'n_estimators': 300,
 'min_samples_split': 4,
 'min_samples_leaf': 3,
 'max_depth': 6,
 'learning_rate': 0.005}
```

Now, let's check the how much is the testing AUROC for this model with the best score.

```
pred = rscv.predict_proba(X_test)[:,1:]
test_roc = roc_auc_score(y_test, pred)

test_roc
```

```
0.7581069190488456
```

So, the testing AUROC is 75.8%

5. Justification:

We are fairly confident that is a decently robust model to be used on a new data. This is because we have used Cross Validation while training the model, and also tested the model on a new test dataset.

Further, the metric used for evaluation of the model, AUROC is appropriate for this task where the data is highly imbalanced.

Finally, while we did not run the complete Grid Search CV, the best result obtained with 10 iteration of the Randomized Search CV is likely to be close to the best model.

Overall, the AUROC values has increased from 67.9% with logistic regression to 75.8% with Gradient Boosted Tree model using Random Search CV

Finally, we will score the completely new, test dataset named as "mailout_test.csv" with this model, and upload it on a Kaggle competition for this task here:

Udacity+Arvato: Identify Customer Segments

Apply machine learning techniques to predict customers using data provided by Arvato Financial Solutions.

www.kaggle.com

We got a score of 0.794 on the Leaderboard of the Kaggle competition, which gives a rank of 156 out of 379 participants.

E. Conclusion

1. **Reflection:**

In summary, in this project I got exposure to doing both Unsupervised Learning (with the KMeans clustering exercise), and Supervised Learning (predictive modeling exercise).

In the KMeans, I realized the importance of being careful with not including variables without cleaning and scaling them. With just raw variables, the KMeans gave poor results, and also did not converge in short time. Further, I realized the importance of using PCA to ensure that number of variables going into KMeans are reduced and hence the convergence occurs faster.

I also learned about the Elbow and Silhouette methods to identify the right number of clusters.

Lastly, the profiling exercise was really interesting via which I identified the key differentiating characteristics between different identified clusters, and also between “Azdias” and “Customers” populations.

I enjoyed doing the predictive modeling exercise more. Through this, I learned how to train and validation the model using multiple machine learning methods.

Further, I learned about the Grid Search CV and the more practical Randomized Search CV.

2. Improvement:

In future, if I would want to try to improve modeling exercises as follows:

a. Segmentation: I would use KPrototype to make use of categorical features also in the model.

b. Predictive modeling: I would use an ensemble of few modeling techniques to improve the AUROC. For example, an ensemble of logistic regression with Gradient Boosted Tree can be tried.

Lastly, I will try to create some derived variables — such as log, square, and square root of existing variables and check if the accuracy improves.

Udacity+Arvato: Identify Customer Segments

Apply machine learning techniques to predict customers using data provided by Arvato Financial Solutions.

www.kaggle.com

Github link for detailed documentation of the project:

priithvi/Udacity_Capstone_Arvato

As part of the Udacity Data Science Nano Degree program, the capstone project is for data provided by 'Arvato Financial...

github.com

Arvato Machine Learning K Means Supervised Learning Unsupervised Learning