Machine Learning Engineer Nanodegree

Capstone project

Customer Segmentation Report for
Arvato Financial Services

Ganesh Bhat
4 May 2021

Contents

Project Overview (Domain Background)

Customer Segmentation Report for Arvato Financial Services

In this project, we will analyze demographics data for customers of a mail-order sales company in Germany, comparing it against demographics information for the general population.

We have been given the data-set as outlined below and we will use a combination of unsupervised and supervised learning techniques to answer few questions for the customer as outlined in the problem statement.

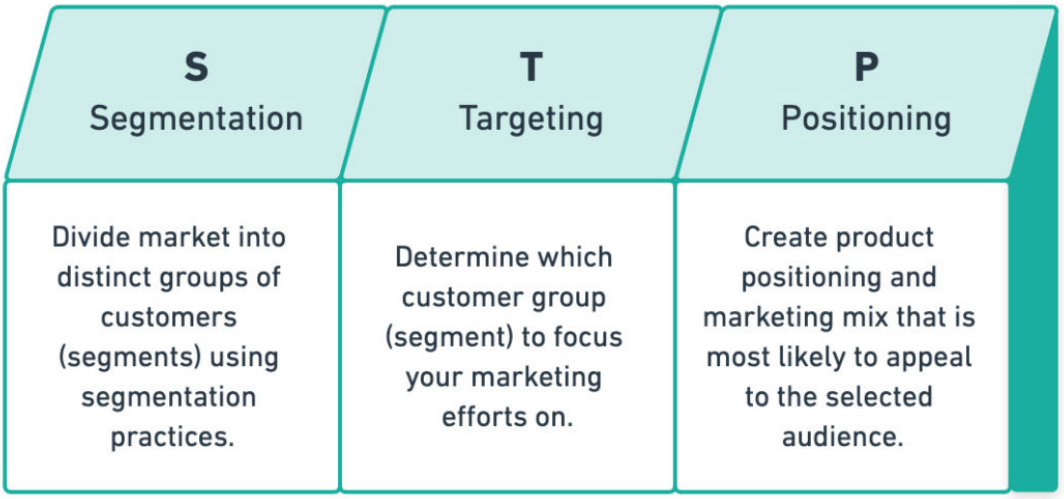


Image Reference

Personal motivation:

My personal curiosity stays with the time-series forecasting of financial data, personal finance, I wanted to apply the learnings from this course and do Machine Learning on Aws Sagemaker for this data.

This project nicely covers many areas such as PCA, Unsupervised and Supervised learning areas, Neural Networks and also provides an opportunity to apply all the different techniques such as Hyperparameter tuning jobs, balancing for certain metrics etc, which I learnt as part of this course.

I am looking at it as a practice project, where I can apply all my learnings and solidify my understanding of all the concepts, instead of focusing on one single larger problem at this stage.

Problem Statement

1. Identify the set of people mail-order sales company in Germany must target with their marketing campaign, instead of targeting all the people across germany.
2. Also identify customer segmentation, identify part of the population that best describes the core customer base (ex: regions, gender and income group etc) of people the company shall focus on, for their marketing campaign.
3. Company needs to know the prediction on which of those individuals are most likely to convert into becoming customers for the company so that they can target such individuals better using instruments like coupons, ads etc.

Datasets and Inputs

The dataset(s) and/or input(s) to be used in the project are thoroughly described. Information such as how the dataset or input is (was) obtained, and the characteristics of the dataset or input, should be included. It should be clear how the dataset(s) or input(s) will be used in the project and whether their use is appropriate given the context of the problem.

There were four data files provided by Arvato for this project.
(As part of the terms and conditions of Arvato, the files cannot be shared) This was obtained from Arvato, shared via Udacity for capstone project purpose.

Below describes the characteristic of the dataset.

- Udacity_AZDIAS_052018.csv: Demographics data for the general population of Germany; 891 211 persons (rows) x 366 features (columns).
- Udacity_CUSTOMERS_052018.csv: Demographics data for customers of a mail-order company; 191 652 persons (rows) x 369 features (columns).
- Udacity_MAILOUT_052018_TRAIN.csv: Demographics data for individuals who were targets of a marketing campaign; 42 982 persons (rows) x 367 (columns).
- Udacity_MAILOUT_052018_TEST.csv: Demographics data for individuals who were targets of a marketing campaign; 42 833 persons (rows) x 366 (columns).

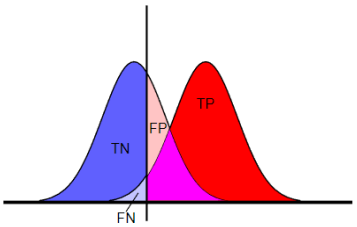
Each row of the demographics files represents a single person, but also includes information outside of individuals, including information about their household, building, and neighborhood.

The "CUSTOMERS" file contains three extra columns ('CUSTOMER_GROUP', 'ONLINE_PURCHASE', and 'PRODUCT_GROUP'), which provide broad information about the customers depicted in the file.

The original "MAILOUT" file included one additional column, "RESPONSE", which indicated whether or not each recipient became a customer of the company.

For the "TRAIN" subset, this column has been retained, but in the "TEST" subset it has been removed; it is against that withheld column that your final predictions will be assessed in the Kaggle competition.

Metrics



In the provided data the class is imbalanced, very similar to credit card fraud detection. We aim to use the AWS provided mechanism to add the balancing factor.

To evaluate unsupervised algorithm efficiency, we will use Silhouette score and measure the efficiency.

As we can see, adding additional individuals for ad-targeting will not have a major side effect. However leaving out an individual from ad-targeting can result in loss of customers.

Hence we will try to increase the ad-coverage than losing them. We will tune our model for metric 'recall'.

An alternate possibility is to consider the metric such as ROC and AUR (reference)

Solution Statement

Solution is

1. Cluster Analysis: Perform dimensionality reduction to reduce the number of features using PCA and then apply a few unsupervised learning algorithms such as clustering (K-Means, DBScan) to show relationship between existing customers and rest of the people in germany. We will use elbow method to choose K for K-Means.

These could be clustered based on age,gender,income & occupation, cultural background, family status,

Various distance metrics such as Cosine Distance, Euclidean distance, Jaccard distance shall be considered.

2. Use one or many Supervised learning models to understand which model performs better in finding whether a person will become a customer or not - XGBoost, Neural Networks with sigmoid (Binary classification) .

Benchmark Model

We will use Logistic Regression (Linear Learner) or Naive Bayes for benchmarking purposes .
(Reference: Segmentation based modelling for advanced targeted marketing)

Reference: benchmarking predictive models

Evaluation Metrics

As we can see, adding additional individuals for ad-targeting will not have a major side effect. However leaving out an individual from ad-targeting can result in loss of customers. We will tune our model for metric 'recall'.

An alternate possibility is to consider the metric such as ROC and AUR (Reference: Beyond accuracy - precision and recall)

Project Design

Overall design

There are 4 parts we are going to address

1. **Data Preprocessing (current notebook)**
   In this part we need to preprocess data for further analysis. Missing values by columns and rows will be analysed, data will be divided by types followed by subsequent transformations.
2. **Customer Segmentation & Unsupervised learning (unsupervised learning notebook)**
   In this part we need to analyze general population and customer segment data sets and use unsupervised learning techniques to perform customer segmentation, identifying the parts of the population that best describe the core customer base of the company. I will use principal component analysis (PCA) technique for dimensionality reduction. Then, elbow curve will be used to identify the best number of clusters for KMeans algorithm. Finally, I will apply KMeans to make segmentation of population and customers and determine description of target cluster for the company.
3. **Supervised Learning Model (supervised learning notebook)**
   In this part we need to build machine learning model using response of marketing campaign and use model to predict which individuals are most likely to convert into becoming customers for the company. I will use several machine learning classifiers and choose the best using analysis of learning curve. Then, I will parametrize the model and make predictions.
4. **Kaggle Competition (submission and results captured as part of supervised learning notebook)**
   The results of this part need to be submitted for Kaggle competition

Detailed design

Data exploration and feature engineering

- Load and prepare data
- Data exploration
- Data Cleaning
- Handling missing values
- Encoding categorical values
- Feature Scaling
- Remove highly co-related features

Unsupervised Learning & clustering

- Dimensionality reduction using PCA
- Analyze component makeup
- Apply unsupervised learning algorithm such as KMeans
- Choose correct K using elblow method (experiment with different k)

- Analyze the clusters using heatmap

Supervised learning

- Load and prepare data
- Explore and preprocess data
- Split into train & test
- Prepare benchmark model and evaluate
- Use following models and check the accuracy

Report

Data exploration and feature engineering

we will be doing following as part of feature engineering and data exploration

- Explore and understand the data
- Clean the data and prepare it for unsupervised learning tasks
- Save the data into files so that other notebook can load it

Data exploration

Shape of the data : Azdias

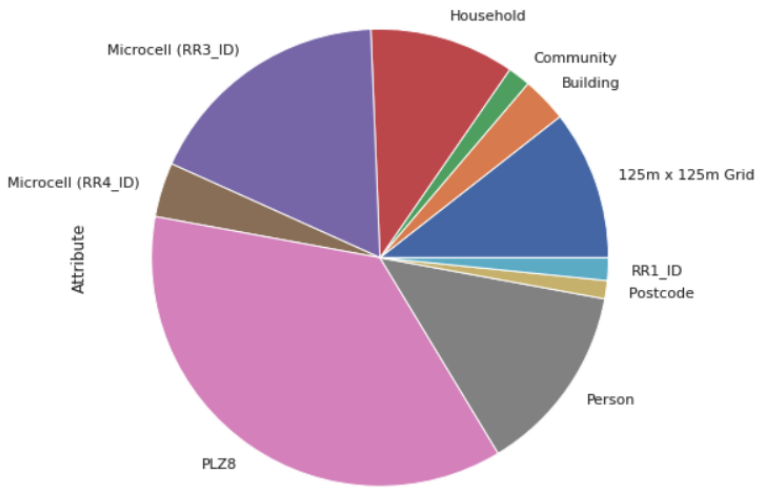
```
In [7]: print(azdias.shape)
        (891221, 366)
```

Shape of the data : Customers

```
In [13]: customers.shape
Out[13]: (191652, 369)
```

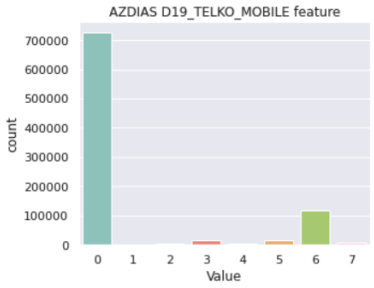
Exploring attributes info provided with respect to information class

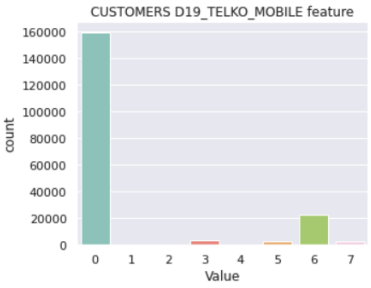
```
In [26]: attributes_info.groupby('Information level').count()['Attribute'].plot(kind='pie',subplots=True, figsize=(8,8))
Out[26]: array([<AxesSubplot:ylabel='Attribute'>], dtype=object)
```

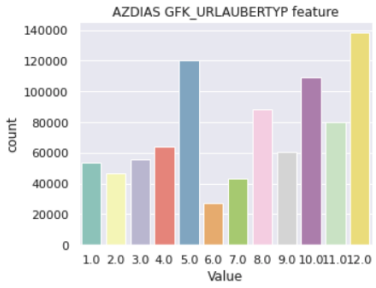


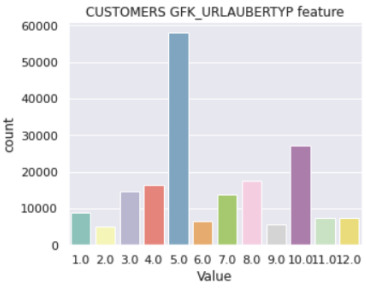
```
columns common to all 3 daframes:  272
columns common to between customer and azdias:  272
columns specific to azdias: 0
columns specific to customers: 3
columns unique to attributes: 42
```

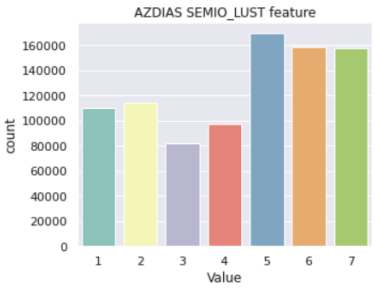
Random sampling of features between customers and general population

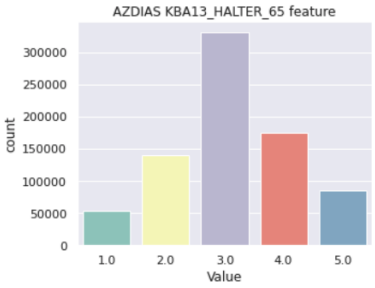


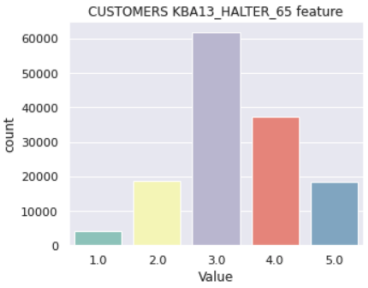












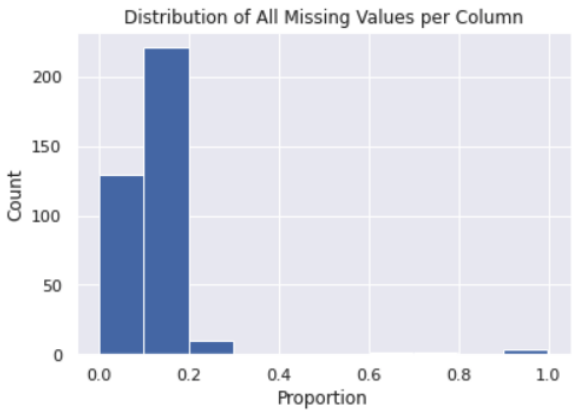
Columns with Unknown and XX, X values

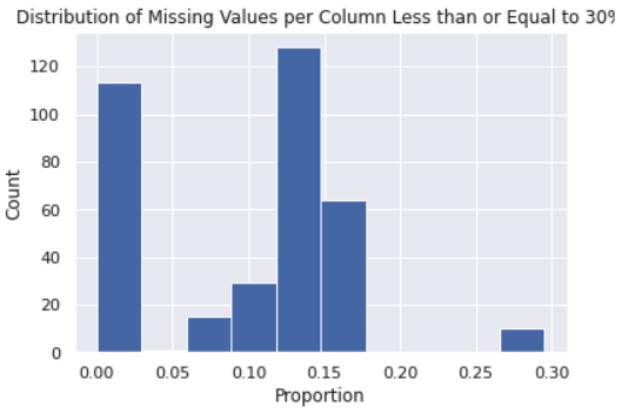
Unknown values are turned into NaN

XX and X values set for the columns are also turned into NaN

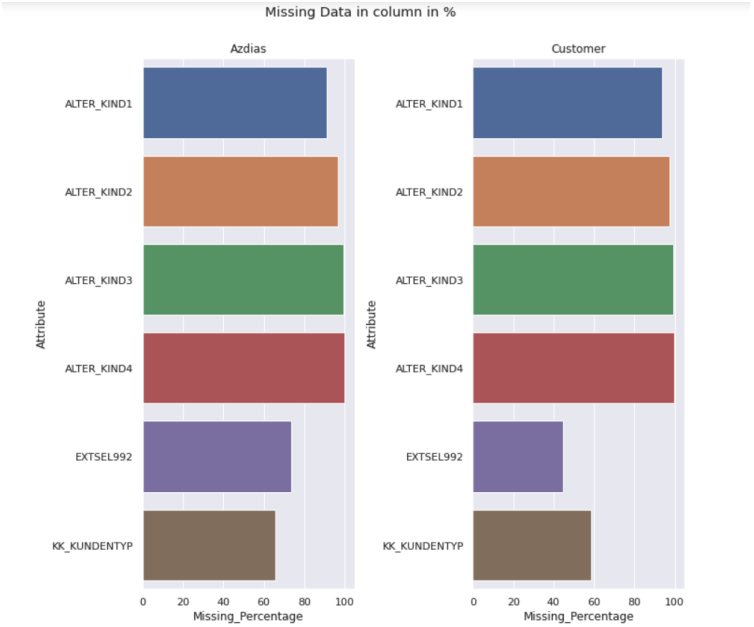
Exploring missing data (Columns)

Distribution of missing values per column





column wise analysis of missing data



Columns with more than 50% missing data is dropped. Notice the columns with a large number of missing features are gone.

```
Number of columns Before dropping
Azdias - 366
Customers - 369
                Removed 6 columns from given dataframes
Number of columns After dropping
Azdias - 360
Customers - 363
```

Exploring missing data (Rows)

Row wise missing data is explored and rows with large number of missing data is dropped

Before dropping

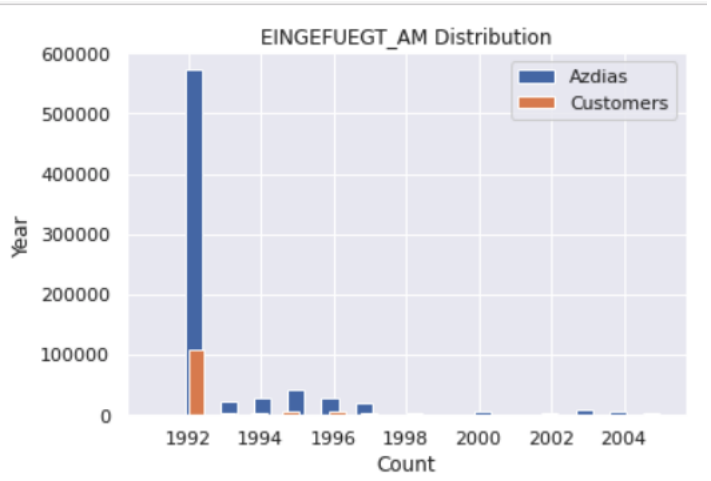
After dropping

Column Type Analysis (Encoding categorical column types)

Following are the non-numeric and non-boolean column types, that must be treated.
We have encoded them using Label encoder or using pandas transform (manual encoding techniques)

```
Azdias columns with dtype Object
 ['CAMEO_DEU_2015' 'D19_LETZTER_KAUF_BRANCHE' 'EINGEFUEGT_AM' 'OST_WEST_KZ']
Customer columns with dtype Object
 ['CAMEO_DEU_2015' 'D19_LETZTER_KAUF_BRANCHE' 'EINGEFUEGT_AM' 'OST_WEST_KZ'
 'PRODUCT_GROUP' 'CUSTOMER_GROUP']
```

column EINGEFUEGT_AM contains date and it is checked with histogram and we retain the year alone.



Removal of extra unnecessary columns

These extra columns are removed as they dont carry lot of useful information

Remove extra columns

```
remove_extra_cols = ["D19_LETZTER_KAUF_BRANCHE", "CAMEO_DEU_2015", "LP_FAMILIE_FEIN", "LP_STATUS_FEIN"]

azdias = remove_columns(azdias, remove_extra_cols)
customers = remove_columns(customers, remove_extra_cols)
```

- column LNR is a unique column and has no special meaning

```
In [94]: len(azdias.LNR.unique()), len(azdias)
Out[94]: (751331, 751331)
```

Lets drop it

```
In [95]: azdias = azdias.drop("LNR", axis=1)
         customers = customers.drop("LNR", axis=1)
```

Feature Scaling

We used a standard scaler to scale our features into uniform value levels, so that learning algorithms can perform better and without less bias.

Columns after scaling

Out[101]:

	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PERSONEN	ANZ_STATISTISCHE
0	-0.572859	1.249446	-1.416979	1.561904	0.159045	-0.128345	-0.296854	0.233463	
1	-0.572859	1.249446	0.809514	0.676059	0.095201	-0.128345	-0.296854	-0.626760	
2	1.858228	-0.942562	0.285633	-0.209786	-0.479387	-0.128345	-0.296854	-1.486983	
3	-0.572859	-0.942562	1.202425	0.011676	-0.351701	-0.128345	-0.296854	1.953909	
4	2.668591	-0.942562	-0.107277	-0.874169	-0.224014	-0.128345	-0.296854	-0.626760	

```
In [102]: customers_scaled.head()
```

Out[102]:

	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PERSONEN	ANZ_STATISTISCHE
0	1.858228	-0.942562	-0.107277	-0.874169	-0.479387	-0.128345	-0.296854	0.233463	
1	-0.572859	-0.942562	-0.631158	-3.088782	-0.479387	-0.128345	-0.296854	-0.626760	
2	1.047866	-0.942562	-0.369218	-1.317092	-0.543230	-0.128345	-0.296854	-1.486983	
3	-0.572859	-0.942562	1.202425	0.011676	-0.096328	-0.128345	-0.296854	1.953909	
4	1.047866	-0.942562	0.023693	-0.874169	-0.479387	-0.128345	-0.296854	0.233463	

Summary

We have done these steps as part of the process

- Explore and understand the data
- Perform data cleanup and turn the data into numeric representation
- Identify categorical columns and label encode them
- Drop the rows from training set that has null values
- Drop the columns that have 50% or more null values
- Identify columns having wrong values such as XX, X and turn into NaN
- Turn values containing unknown to NaN
- Split the RESPONSE column and move into label
- Fix certain columns encoded in one or two letter words into numbers
- Remove the LNR column, that will be later appeneded to resut for Kaggle compitation
- Remove columns and rows without data and without much meaning
- Perform imputation to fill average data into the missing data columns
- Perform scaling

Unsupervised Learning & Clustering

(2. UnsupervisedLearning.pynb file)

Now that we have done data cleaning and feature engineering in our last section, we have the following now

- azdias_scaled_with_header.csv - 751331 rows and 356 columns
- customers_scaled_with_header.csv - 135144 rows and 356 columns

We will be doing following as part of Unsupervised learning

- Identify the clusters of population between customers and general german population
- Find out set of people from general german population who must be targeted for our advertising campaign

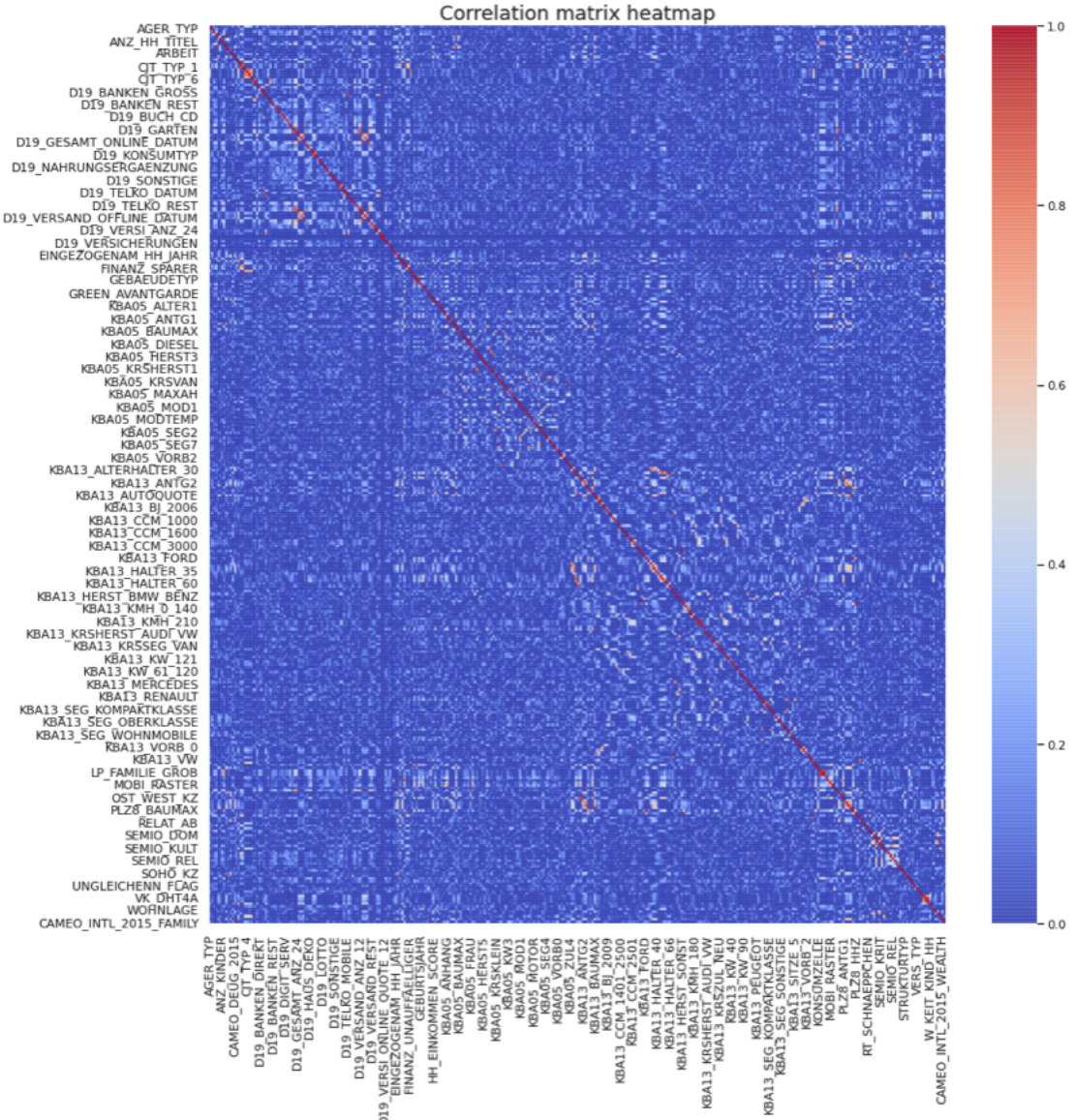
Solution Approach

- We will perform dimensionality reduction using PCA to reduce the number of features
- We will identify clusters in the population and customers and find an overlap
- Identify clusters with maximum overlap
- Evaluate the criteria that fits people into certain cluster (finance, kind of cars owned etc)
- Callout a summary explaining which clusters of people in general population must be targeted with our advertising campaign

Feature Correlation

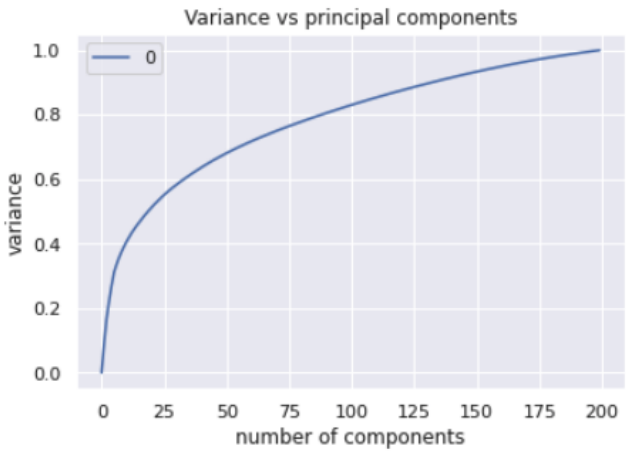
Pictorial analysis of feature correlation left a lot to desire and did not prove to be very helpful.

However, overall features don't have very high co-relation is very clear from the heatmap below.



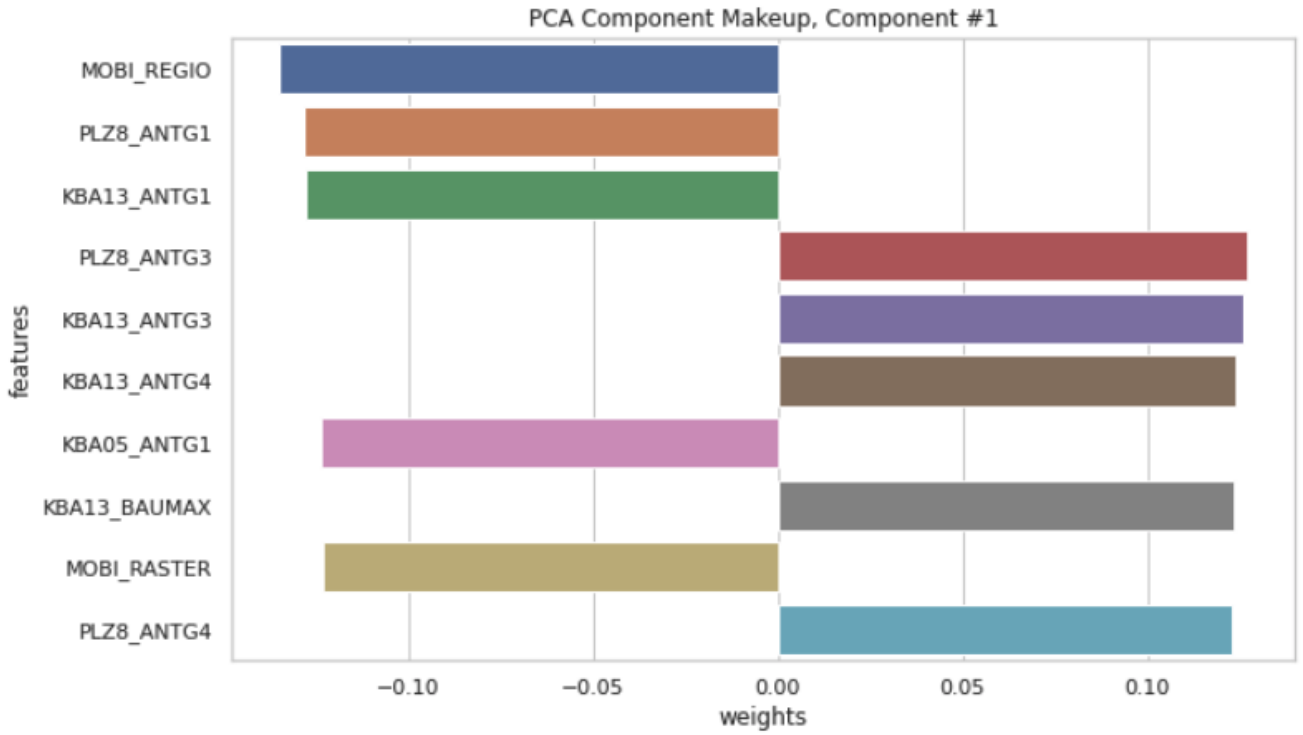
Principal Component Analysis

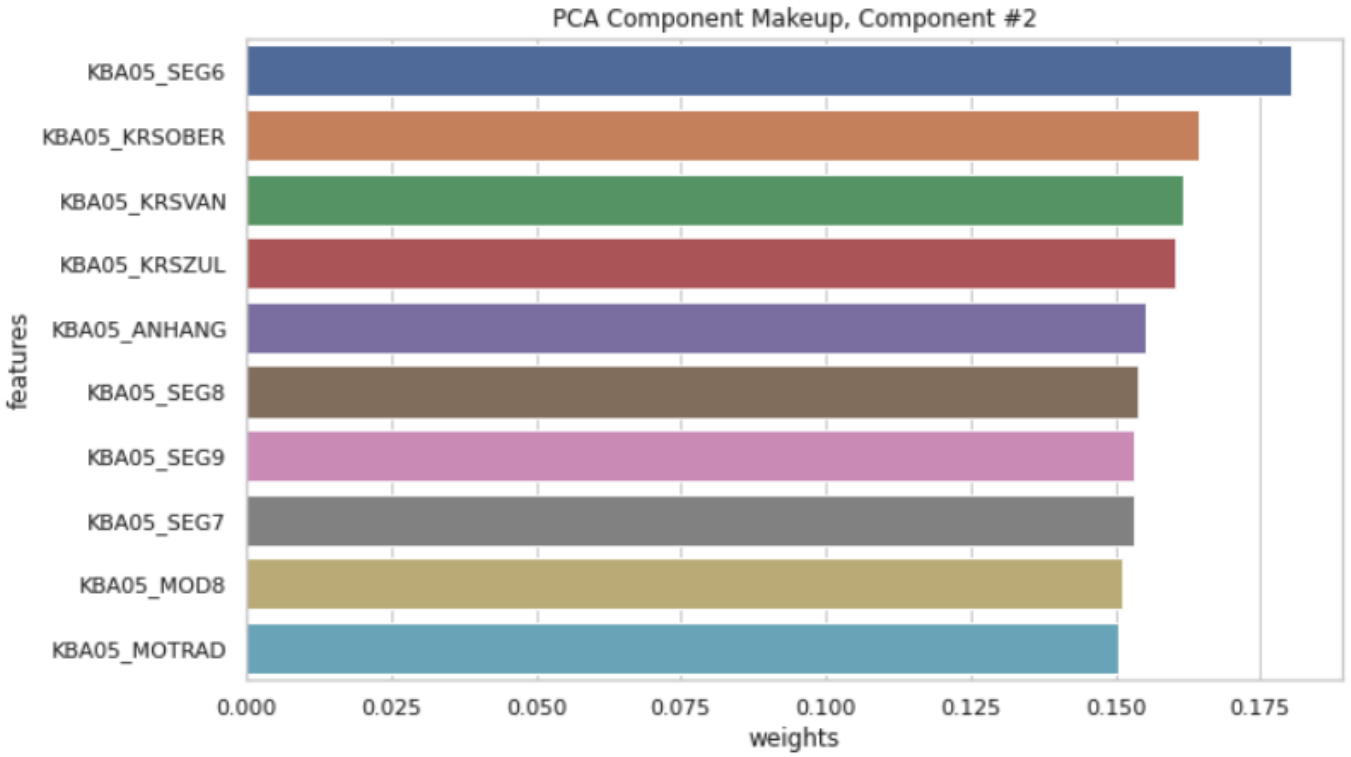
The input data file is provided for PCA and we have explored the possibility of using PCA for 200, 100 and 50 Components.

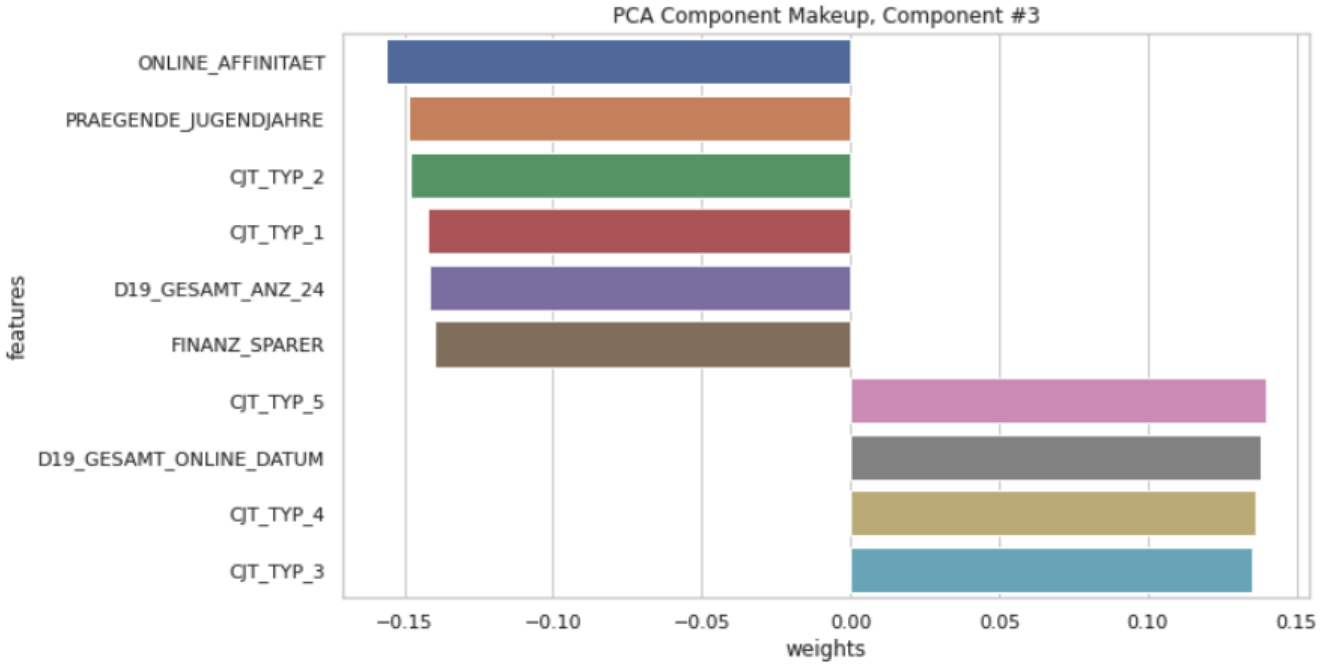


As we can see, 100 components sufficiently covers 80% + data variance. Hence 100 is the value chosen for performing PCA of customer and general population data

Exploring few components to understand how PCA components are mapped to features







This shows the pretty good composition of components.

Following are the outputs from PCA

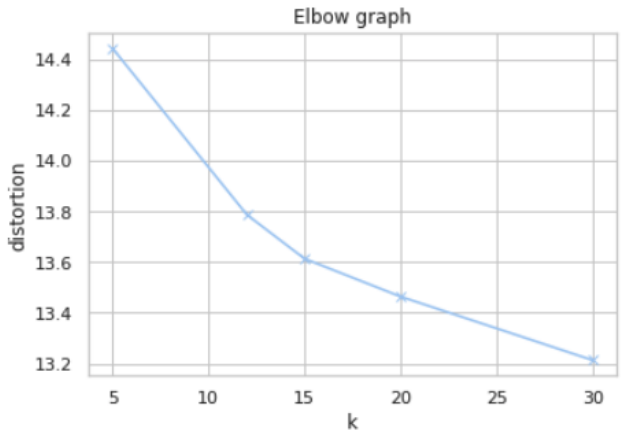
- PCA transformed data for customers - pca_customer_final.csv
- PCA transformed data for general public - pca_azdias_final.csv

Same is saved in AWW S3 for clusting use-case

Clustering

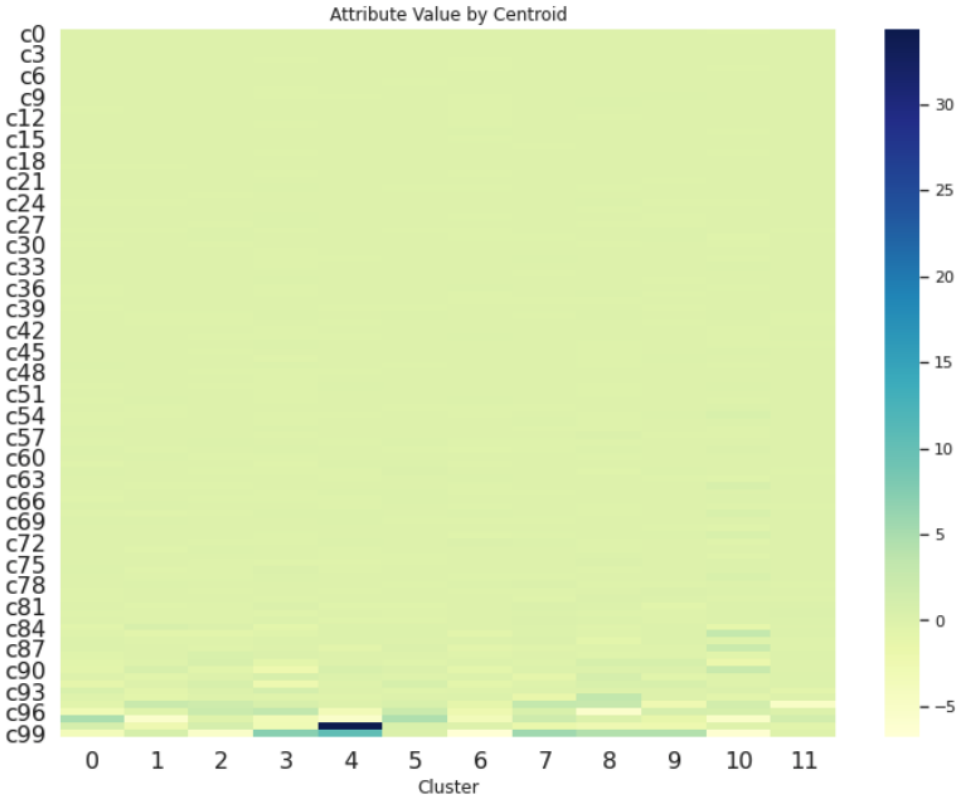
We will be using AWS parallel job running infrastructure to run the K-Means clustering for larg number of K values.

Reference:
https://aws.amazon.com/blogs/machine-learning/k-means-clustering-with-amazon-sagemaker/

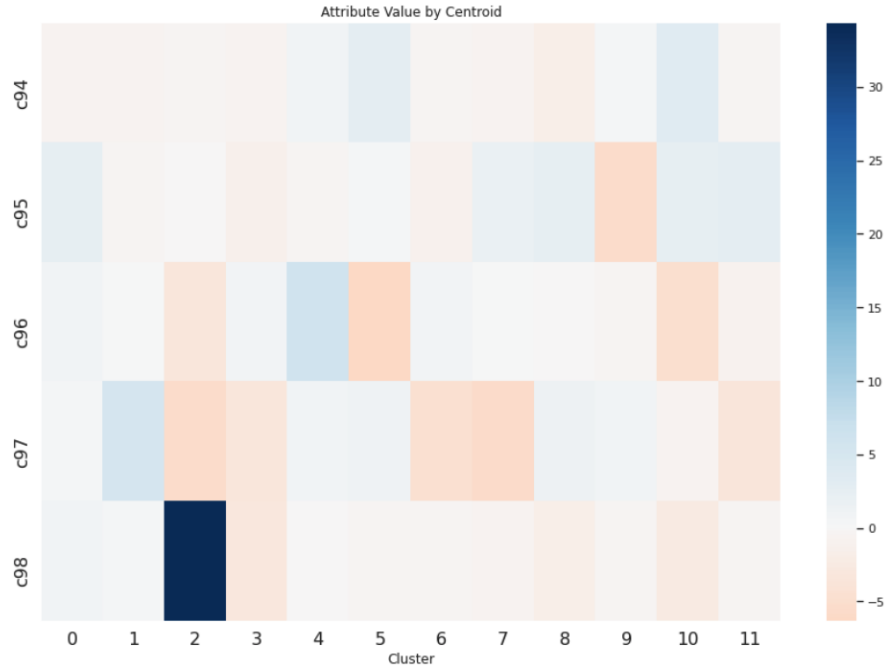


This is the elbow graph created indicates an elbow happening at K=12, indicating a good cluster value for our K.

Clustering is run with K=12 and here is the visualization of centroids



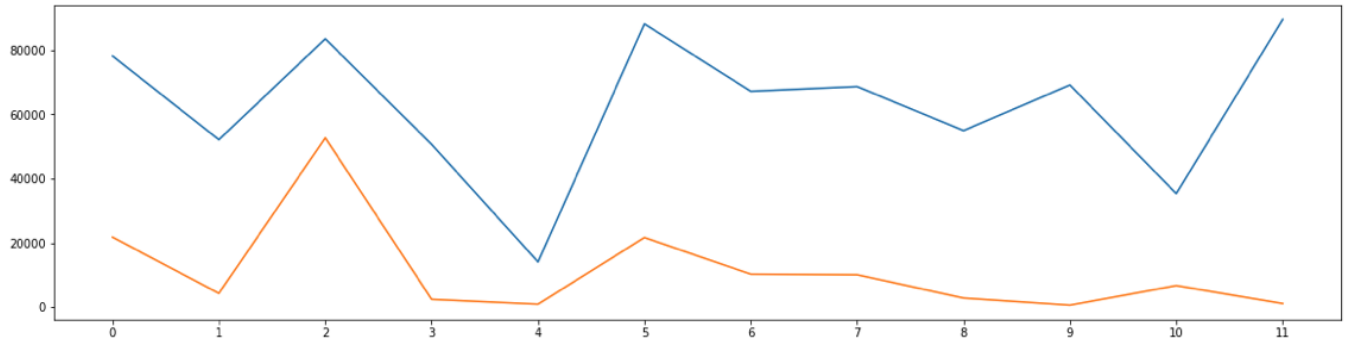
As we can see, most of the data is concentrated around top 5 (94-99), let's visualize those.



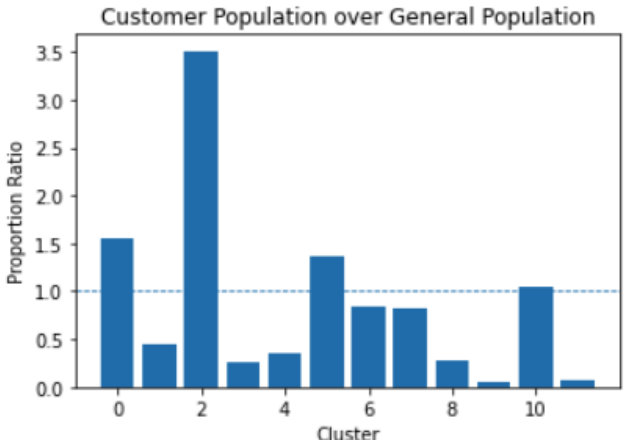
We will perform batch clustering for general population and customer data using Sagemaker

```
kmeans_transformer = kmeans.transformer(
    #model_name = 'pca-2021-05-01-10-42-21-922',
    instance_count=1,
    instance_type='ml.m4.xlarge',
    output_path='s3://{}/{}/output'.format(session.default_bucket(), prefix2)
)
kmeans_transformer.transform(pca_azdias_input_s3loc, content_type='text/csv', split_type='Line')
```

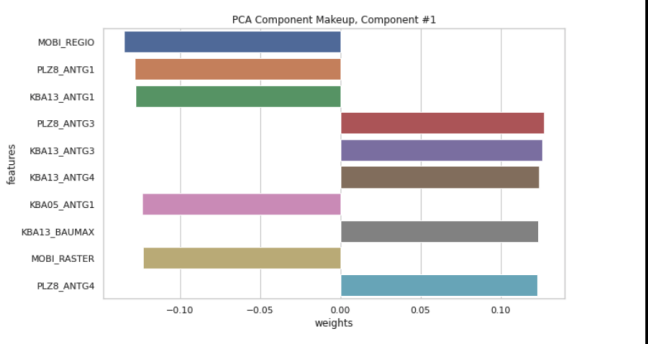
Lets try to evaluate cluster distribution between customer and general public



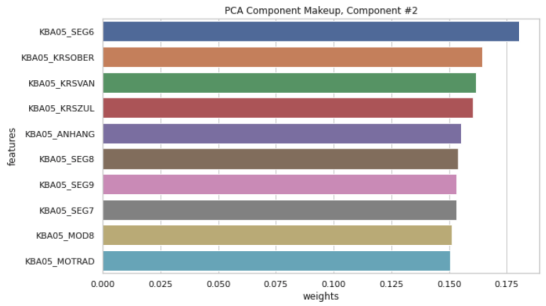
Lets perform weighed overlap comparison to understand the share.



Details of each cluster is looked at as well using the charts specific to cluster along with the details about each column (below shows 2 samples of the same)



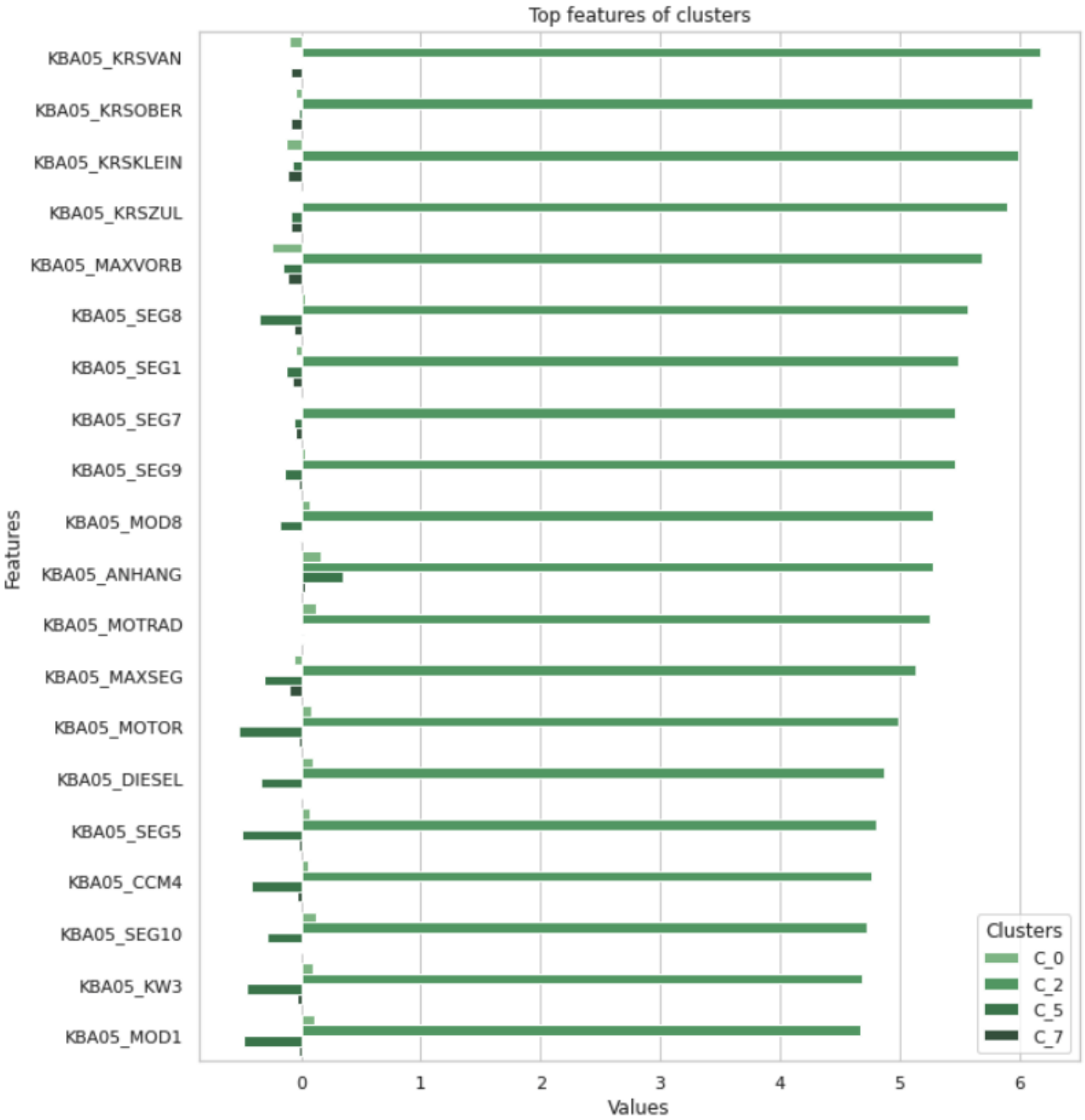
	weights	features	abs_weights	Information level	Description	Additional notes
301	-0.134590	MOBI_REGIO	0.134590	RR1_ID	moving patterns	NaN
306	-0.128075	PLZ8_ANTG1	0.128075	PLZ8	number of 1-2 family houses in the PLZ8	NaN
179	-0.127557	KBA13_ANTG1	0.127557	NaN	NaN	NaN
308	0.126775	PLZ8_ANTG3	0.126775	PLZ8	number of 6-10 family houses in the PLZ8	NaN
181	0.126115	KBA13_ANTG3	0.126115	NaN	NaN	NaN
182	0.123579	KBA13_ANTG4	0.123579	NaN	NaN	NaN
115	-0.123249	KBA05_ANTG1	0.123249	Microcell (RR3_ID)	number of 1-2 family houses in the cell	NaN
186	0.123157	KBA13_BAUMAX	0.123157	NaN	NaN	NaN
300	-0.122749	MOBI_RASTER	0.122749	NaN	NaN	NaN
309	0.122678	PLZ8_ANTG4	0.122678	PLZ8	number of >10 family houses in the PLZ8	NaN



	weights	features	abs_weights	Information level	Description	Additional notes
164	0.180107	KBA05_SEG6	0.180107	Microcell (RR3_ID)	share of upper class cars (BMW 7er etc.) in th...	NaN
139	0.164214	KBA05_KRSOBER	0.164214	Microcell (RR3_ID)	share of upper class cars (referred to the cou...	NaN
140	0.161615	KBA05_KRSVAN	0.161615	Microcell (RR3_ID)	share of vans (referred to the county average)	NaN
141	0.160324	KBA05_KRSZUL	0.160324	Microcell (RR3_ID)	share of newbuilt cars (referred to the county...	NaN
114	0.155209	KBA05_ANHANG	0.155209	Microcell (RR3_ID)	share of trailers in the microcell	NaN
166	0.153776	KBA05_SEG8	0.153776	Microcell (RR3_ID)	share of roadster and convertables in the micr...	NaN
167	0.153001	KBA05_SEG9	0.153001	Microcell (RR3_ID)	share of vans in the microcell	NaN
165	0.152999	KBA05_SEG7	0.152999	Microcell (RR3_ID)	share of all-terrain vehicles and MUVs in the ...	NaN
154	0.151241	KBA05_MOD8	0.151241	Microcell (RR3_ID)	share of vans (in an AZ specific definition)	NaN
157	0.150369	KBA05_MOTRAD	0.150369	Microcell (RR3_ID)	share of motorcycles per household	NaN

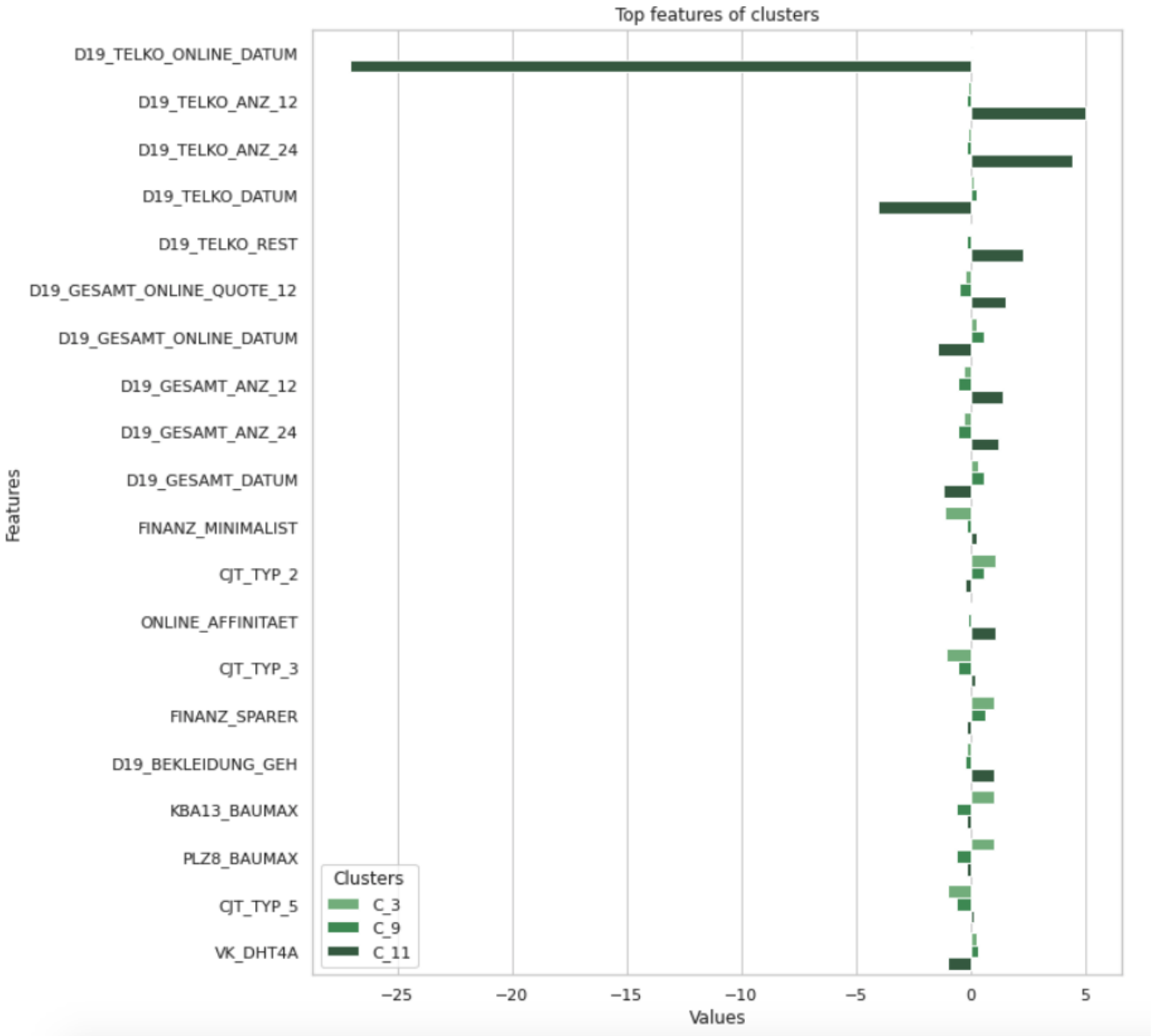
Based on the above analysis, we will be considering the cluster 0, 2 , 5 and optionally 10 for considering them to be better overlap of customers with general public

Analyzing each cluster to understand what they represent as overall set of customers in 0,2,5,7 clusters



27	KBA05_SEG7	5.468613	C_2	Microcell (RR3_ID)	share of all-terrain vehicles and MUVs in the microcell	NaN
28	KBA05_SEG9	5.461203	C_2	Microcell (RR3_ID)	share of vans in the microcell	NaN
29	KBA05_MOD8	5.282201	C_2	Microcell (RR3_ID)	share of vans (in an AZ specific definition)	NaN
30	KBA05_ANHANG	5.274980	C_2	Microcell (RR3_ID)	share of trailers in the microcell	NaN
31	KBA05_MOTRAD	5.253604	C_2	Microcell (RR3_ID)	share of motorcycles per household	NaN
32	KBA05_MAXSEG	5.137200	C_2	Microcell (RR3_ID)	most common car segment in the microcell	NaN
33	KBA05_MOTOR	4.984557	C_2	Microcell (RR3_ID)	most common engine size in the microcell	NaN
34	KBA05_DIESEL	4.865035	C_2	Microcell (RR3_ID)	share of cars with Diesel-engine in the microcell	NaN
35	KBA05_SEG5	4.803234	C_2	Microcell (RR3_ID)	share of upper middle class cars and upper class cars (BMW5er, BMW7er etc.)	NaN
36	KBA05_CCM4	4.771638	C_2	Microcell (RR3_ID)	share of cars with more than 2499ccm	NaN
37	KBA05_SEG10	4.723862	C_2	Microcell (RR3_ID)	share of more specific cars (Vans, convertables, all-terrains, MUVs etc.)	NaN
38	KBA05_KW3	4.689517	C_2	Microcell (RR3_ID)	share of cars with an engine power of more than 119 KW	NaN
39	KBA05_MOD1	4.673879	C_2	Microcell (RR3_ID)	share of upper class cars (in an AZ specific definition)	NaN

Analyzing each cluster to understand what they represent as overall set of customers in 3,9,11 clusters



45	D19_GESAMT_ONLINE_QUOTE_12	1.520283	C_11	Household	amount of online transactions within all transactions in the complete file
46	D19_GESAMT_ONLINE_DATUM	-1.463496	C_11	Household	actuality of the last transaction with the complete file ONLINE
47	D19_GESAMT_ANZ_12	1.382835	C_11	NaN	NaN
48	D19_GESAMT_ANZ_24	1.226397	C_11	NaN	NaN
49	D19_GESAMT_DATUM	-1.176568	C_11	Household	actuality of the last transaction with the complete file TOTAL
50	FINANZ_MINIMALIST	0.240001	C_11	Person	financial typology: low financial interest
51	CJT_TYP_2	-0.230610	C_11	NaN	NaN
52	ONLINE_AFFINITAET	1.082423	C_11	RR1_ID	online affinity
53	CJT_TYP_3	0.197107	C_11	NaN	NaN
54	FINANZ_SPARER	-0.149157	C_11	Person	financial typology: money saver
55	D19_BEKLEIDUNG_GEH	1.023013	C_11	125m x 125m Grid	transactional activity based on the product group LUXURY CLOTHING

Summary

Steps we performed

- perform dimensionality reduction using PCA
- Analyze component makeup
- Apply unsupervised learning algorithm such as KMeans
- Choose correct K using elblow method (experiment with different k)
- Analyze the clusters using heatmap

Conclusion

As you can see most of our customers are in the clusters 0,2,5,7

Which represents the general population with

- good spending habbits
- with top model cars
- large houses
- better savings habbits
- online spenders

People in clusters 3,9,11 should NOT be targetted for our marketing campain

Which represents the general population with

- bad financial habbits
- not good at savings
- own small cars
- doesnt own luxury clothing
- low online interest, less etc.

Supervised Learning

(3. Supervised Learning.ipynb)

Problem Overview

Now that you've found which parts of the population are more likely to be customers of the mail-order company, it's time to build a prediction model. Each of the rows in the "MAILOUT" data files represents an individual that was targeted for a mailout campaign. Ideally, we should be able to use the demographic information from each individual to decide whether or not it will be worth it to include that person in the campaign.

The "MAILOUT" data has been split into two approximately equal parts, each with almost 43 000 data rows. In this part, you can verify your model with the "TRAIN" partition, which includes a column, "RESPONSE", that states whether or not a person became a customer of the company following the campaign. In the next part, you'll need to create predictions on the "TEST" partition, where the "RESPONSE" column has been withheld.

Problem Statement

As we have understood the data in the previous analysis we did while clustering, we have identified

- Set of german population that closely overlaps with the customers of the company
- We know the set of features/clusters to be targeted for our mail campaign.

Expected outcome

- Now the problem statement is to identify with larger certainty that who are all those people from these identified clusters would respond to ad-campaigns when we target them with a variety of advertisements.
- We have been provided with two data files that contain the manually curated list of customers identified by LNR, who responded vs not-responded to advertising campaigns.
- As we have the labelled data available, our job here is to train multiple supervised learning models and evaluate the metrics to understand which is performing better among them and identify the potential customers.
- As an extension of this exercise, we will also run this against Kaggle competition to know the results of our prediction

Data Loading and Exploring

We have received two data files for this task

```
mailout_train = pd.read_csv('Udacity_MAILOUT_052018_TRAIN.csv', sep=';', dtype={18:'str',19:'str'})
mailout_test = pd.read_csv('Udacity_MAILOUT_052018_TEST.csv', sep=';', dtype={18:'str',19:'str'})
```

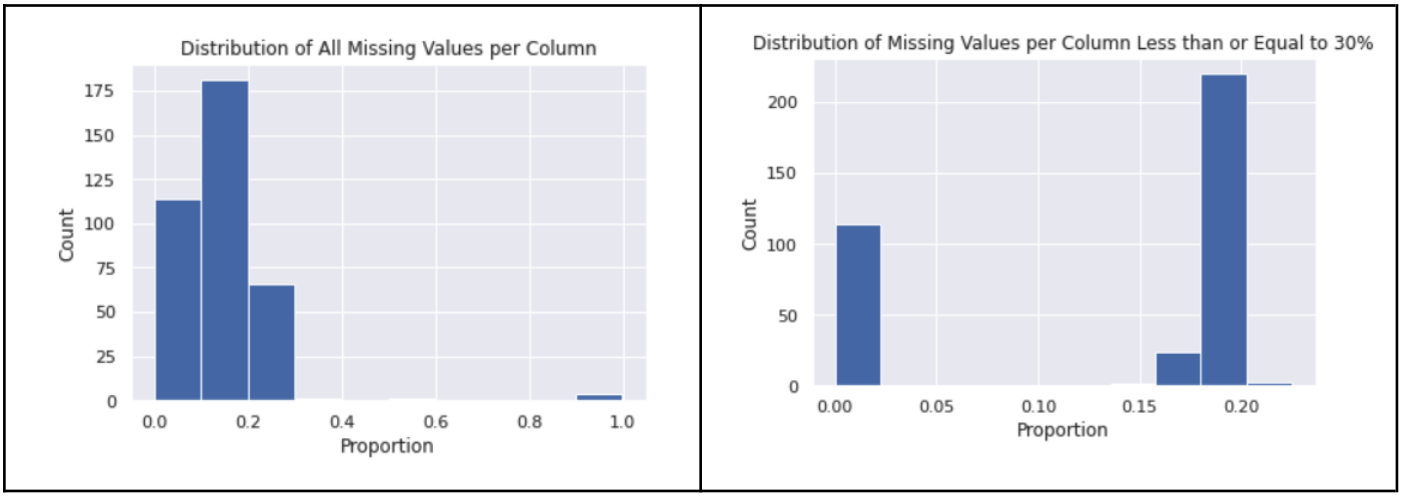
with the shape

```
: mailout_train.shape, mailout_test.shape
: ((42962, 367), (42833, 366))
```

Data Cleaning

similar to what we done before in unsupervised model, as we know the common steps we have taken, lets do this now

Analyzing missing values



Similar to the initial data cleaning steps, we have done these

- Identify categorical columns and label encode them
- Drop the rows from training set that has null values
- Drop the columns that have 50% or more null values
- Identify columns having wrong values such as XX, X and turn into NaN
- Turn values containing unknown to NaN
- Split the RESPONSE column and move into label
- Fix certain columns encoded in one or two letter words into numbers
- Remove the LNR column, that will be later appeneded to resut for Kaggle compitation
- Using imputer we will fill in avergae value for the missing columns
- Scale the data for the algorithm

We never drop the rows from training set, as that is pre-requsite for Kaggle submission

Resulting columns before scaling looks like this

```
In [73]: mailout_train.head()
```

Out[73]:

	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PERSONEN	ANZ_STATISTISCHE
0	2	1.0	8.0	8.0	15.0	0.0	0.0	1.0	
1	1	4.0	13.0	13.0	1.0	0.0	0.0	2.0	
2	1	1.0	9.0	7.0	0.0	NaN	0.0	0.0	
3	2	1.0	6.0	6.0	4.0	0.0	0.0	2.0	
4	2	1.0	9.0	9.0	53.0	0.0	0.0	1.0	

```
In [74]: mailout_test.head()
```

Out[74]:

	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PERSONEN	ANZ_STATISTISCHE
0	2	1.0	7.0	6.0	2.0	0.0	0.0	2.0	
1	-1	1.0	0.0	0.0	20.0	0.0	0.0	1.0	
2	2	9.0	16.0	11.0	2.0	0.0	0.0	4.0	
3	-1	7.0	0.0	0.0	1.0	0.0	0.0	0.0	
4	1	1.0	21.0	13.0	1.0	0.0	0.0	4.0	

and scaled columns

	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PERSONEN	ANZ_STATISTISCHE
0	0.863541	-0.294962	-0.375361	-0.416422	0.548565	-0.131367	-0.220536	-0.750986	
1	0.120172	1.455271	0.447323	0.743341	-0.379287	-0.131367	-0.220536	-0.018142	
2	0.120172	-0.294962	-0.210824	-0.648374	-0.445562	-0.131367	-0.220536	-1.483831	
3	0.863541	-0.294962	-0.704435	-0.880327	-0.180462	-0.131367	-0.220536	-0.018142	
4	0.863541	-0.294962	-0.210824	-0.184469	3.067020	-0.131367	-0.220536	-0.750986	

This is the shape of the data at the end of data cleaning

```
In [84]: mailout_train_scaled.shape, mailout_test_scaled.shape
Out[84]: ((34214, 360), (42833, 360))
```

click to expand output; double click to hide output

Evaluation metrics

In the provided data the class is imbalanced, very similar to credit card fraud detection. We aim to use the AWS provided mechanism to add the balancing factor. To evaluate unsupervised algorithm efficiency, we will use Silhouette score and measure the efficiency.

As we can see, adding additional individuals for ad-targeting will not have a major side effect. However leaving out an individual from ad-targeting can result in loss of customers. Hence we will try to increase the ad-coverage than loosing them. We will tune our model for metric 'recall'.

The data is highly imbalanced. Accuracy will be a bad metric to choose for this problem since the accuracy might always be more than 98% even if the model predicts all zeros. To address this imbalance while evaluating the model we need to chose a metric which will take this class

imbalance into accounts. The usual metrics used for imbalanced classification are Precision and Recall or Area under Receiver Operating Curve (AUROC).

Lets understand confusion matris as basis for understanding AUROC

Confusion Matrix: is a 2x2 matrix in case of binary classification problems which shows the number of values predicted correctly and the number of values predicted wrongly. The confusion matrix is the base for any classification metric, even accuracy can be calculated from the values inside confusion matrix.

Terms inside confusion matrix:

- True Positives (TP) - Number of positive observations predicted as positive
- True Negatives (TN) - Number of negative observations predicted as negative
- False Negatives (FN) - Number of positive observations predicted as negative
- False Positives (FP) - Number of negative observations predicted as positive

Precision and Recall:

- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)

ROC curve

- True Positive Rate = TP / (TP + FN)
- False Positive Rate = FP / (FP + TN)

More details can be found here (Beyond accuracy and recall blog)

Algorithms and Techniques

As explained in the proposal, we plan to use these algorithms & techniques to solve this problem

- Baseline model such as Linear Learner
- Neural Networks with PyTorch
- XgBoost and also use with Hyperparameter tuner for better results
- KNN & SVM (optional, if time permits)

Prepare the data

We will retain 1% of the data for testing, we will split the train and validation set into 80:20

Split data into training, validation and testing

Keep the testing as 1% of the data

In [101]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(mailout_train_scaled, labels_df, test_size=0.01, shuffle=True)
```

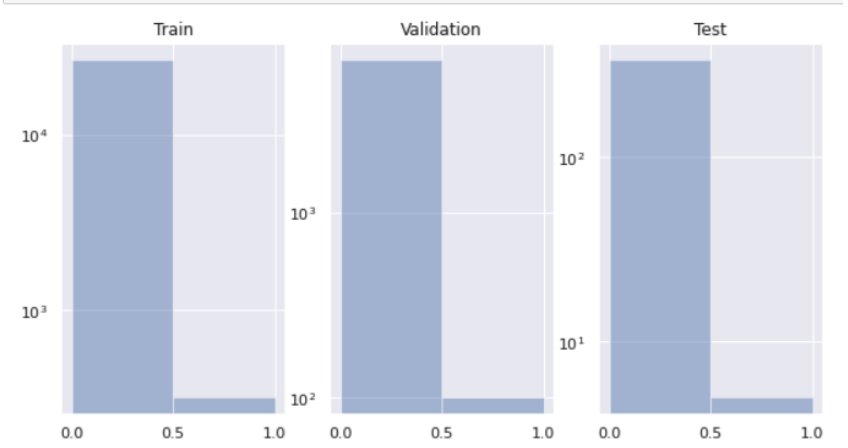
In [102]:
```
len(X_train), len(X_test), len(y_train), len(y_test)
```

Out[102]:
```
(33871, 343, 33871, 343)
```

In [104]:
```
#further split train into train & validation
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.20, shuffle=True)

print(len(X_train), len(X_test), len(X_val))
```

```
27096 343 6775
```

Distribution of data



Benchmark model

We will be using Linear Learner as benchmark model

In [132]:
```
from sagemaker import LinearLearner


# specify an output path
prefix = 'mailout_linear'
output_path = 's3://{}/{}'.format(bucket_name, prefix)

# instantiate LinearLearner
linear_balanced = LinearLearner(role=role,
                                train_instance_count=1,
                                train_instance_type='ml.c4.xlarge',
                                predictor_type='binary_classifier',
                                output_path=output_path,
                                sagemaker_session=session,
                                epochs=15,
                                binary_classifier_model_selection_criteria='precision_at_target_recall', # target r
                                target_recall=0.9,
                                positive_example_weight_mult='balanced')
```

using the 1% testing data, we will evaluate the effectiveness of the model

Deploy the predictor

In [136]:
```
%%time
# deploy and create a predictor
balanced_predictor = linear_balanced.deploy(initial_instance_count=1, instance_type='ml.t2.medium')
```

```
Parameter image will be renamed to image_uri in SageMaker Python SDK v2.
```

```
----------------------!CPU times: user 287 ms, sys: 27.1 ms, total: 314 ms
Wall time: 10min 33s
```

Test with sample value

In [148]:
```
balanced_predictor.predict(test_dataX.to_numpy()[:1].astype('float32'))
```

Out[148]:
```
[label {
   key: "predicted_label"
   value {
     float32_tensor {
       values: 1.0
     }
   }
 }
 label {
   key: "score"
   value {
     float32_tensor {
       values: 0.49471575021743774
     }
   }
 }]
```

Perform evaluation

In [156]:
```
print('Metrics for balanced, LinearLearner.\n')
# get metrics for balanced predictor
metrics = evaluate(balanced_predictor,
                   x_test_data_np.astype('float32'),
                   y_test_data_np)
```

```
Metrics for balanced, LinearLearner.


Recall:     0.350
Precision:  0.015
Accuracy:   0.646
```

Lets perform batch transformation of the test file supplied to validate against Kaggle (which uses AUROC)

In [36]:
```
linear_transformer = linear_balanced.transformer(instance_count = 1, instance_type = 'ml.m4.xlarge')
```

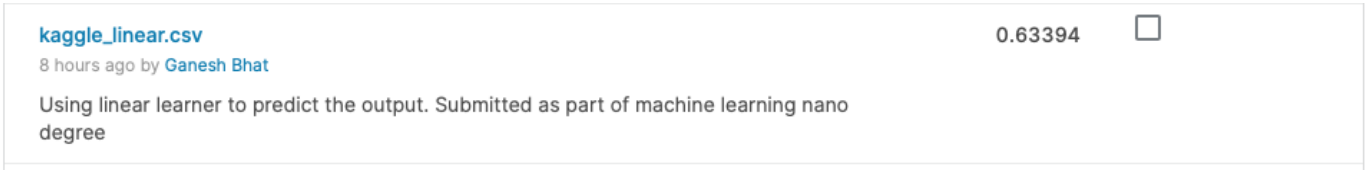
```
Parameter image will be renamed to image_uri in SageMaker Python SDK v2.
```

In [37]:
```
# save to s3 for training job
mailout_test_scaled = pd.read_csv('mailout_test_scaled_header.csv')
mailout_test_scaled.to_csv('mailout_test_scaled.csv', index=False, header=False);

prefix2 = 'mailout'
mailout_test_location = session.upload_data('mailout_test_scaled.csv', key_prefix=prefix2)
```

In [38]:
```
linear_transformer.transform(mailout_test_location, content_type='text/csv', split_type='Line')
```

In [39]:
```
linear_transformer.wait()
```

Here is the submission result of **63.394%**



PyTorch and Neural Networks

Create a binary classifier with 1 hidden layer that uses Sigmoid activator, with dropout 0.3 to cater to bias.

```
In [44]: !pygmentize source_pytorch/model.py
```

```
        Initialize the model by setting up linear layers.
        Use the input parameters to help define the layers of your model.
        :param input_features: the number of input features in your training/test data
        :param hidden_dim: helps define the number of nodes in the hidden layer(s)
        :param output_dim: the number of outputs you want to produce
        """
        super(BinaryClassifier, self).__init__()

        # define any initial layers, here
        self.fc1 = nn.Linear(input_features, hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, output_dim)
        self.drop = nn.Dropout(0.3)

        # sigmoid layer
        self.sig = nn.Sigmoid()
```

create a trainer to use 80 hidden neurons

```
## TODO: Add args for the three model parameters: input_features, hidden_dim, output_dim
# Model Parameters
parser.add_argument('--input_features', type=int, default=360, metavar='IN')
parser.add_argument('--hidden_dim', type=int, default=80, metavar='H')
parser.add_argument('--output_dim', type=int, default=1, metavar='OUT')
parser.add_argument('--lr', type=float, default=0.001, metavar='LR',
                    help='learning rate (default: 0.001)')
```

use pytorch to train the mode

In [99]:
```
# your import and estimator code, here
from sagemaker.pytorch import PyTorch

from sagemaker.estimator import Estimator

output_path = f"s3://{bucket_name}/{prefix}"

estimator = PyTorch(
    entry_point="train.py",
    source_dir="source_pytorch",
    role=role,
    framework_version="1.0",
    py_version = "py3",
    train_instance_count=1,
    train_instance_type="ml.c4.xlarge",
    output_path=output_path,
    sagemaker_session=session,
    hyperparameters={
        "input_features": 360,
        "hidden_dim": 80,
        "output_dim": 1,
        "epochs": 160
    })
```

In [100]:
```
%%time

# Train your estimator on S3 training data

estimator.fit({'train': train_location})
```

Model is trained for 160 epochs with 0.0012 loss.

```
Epoch: 135, Loss: 0.0023246333622153233
Epoch: 136, Loss: 0.0015705725289617187
Epoch: 137, Loss: 0.0013665378349616967
Epoch: 138, Loss: 0.001986339102085015
Epoch: 139, Loss: 0.0006160821103210616
Epoch: 140, Loss: 0.0006260442701973102
Epoch: 141, Loss: 0.0022906240571799728
Epoch: 142, Loss: 0.002646572396390357
Epoch: 143, Loss: 0.004568177540015756
Epoch: 144, Loss: 0.0033961092157784854
Epoch: 145, Loss: 0.0032962243187657663
Epoch: 146, Loss: 0.003630483833117028
Epoch: 147, Loss: 0.0011275577883072493
Epoch: 148, Loss: 0.0026344207759058263
Epoch: 149, Loss: 0.0018407670559794174
Epoch: 150, Loss: 0.0011256384915214477
Epoch: 151, Loss: 0.0011032939251286447
Epoch: 152, Loss: 0.0020193832066887018
Epoch: 153, Loss: 0.00215300327903104
Epoch: 154, Loss: 0.0012558328581495019
Epoch: 155, Loss: 0.0026039970939978584
Epoch: 156, Loss: 0.0030240577301451827
Epoch: 157, Loss: 0.004914673675298753
Epoch: 158, Loss: 0.0008577220536573687
Epoch: 159, Loss: 0.0033641486560330104
Epoch: 160, Loss: 0.0012232039593301756
2021-05-04 08:49:05,931 sagemaker-containers INFO     Reporting training SUCCESS
```

Deploy the mode

In [104]:
```
%%time

# uncomment, if needed
# from sagemaker.pytorch import PyTorchModel
from sagemaker.pytorch import PyTorchModel

model = PyTorchModel(
    entry_point="predict.py",
    role=role,
    framework_version="1.0",
    py_version='py3',
    model_data=estimator.model_data,
    source_dir="source_pytorch"
)
```

In [105]:
```
# deploy your model to create a predictor
predictor = model.deploy(initial_instance_count=1, instance_type="ml.t2.medium")
```

```
'create_image_uri' will be deprecated in favor of 'ImageURIProvider' class in Sage

----------------!
```

Perform prediction in batches and collect the result

In [109]:
```
prediction_batches = [predictor.predict(batch) for batch in np.array_split(test_x, 100)]
```

In [110]:
```
test_preds = np.concatenate([np.array([x for x in batch])
                             for batch in prediction_batches])
```

Write to csv and obtain the AUROC score from Kaggle

Submission and Description	Public Score	Use for Final Score
neural_kaggle-2.csv 5 hours ago by Ganesh Bhat This is based on neural network model using pytorch in Amazon Sagemaker	0.50048	☐

This model underperformed our base linear learner model and scored 50.048%

Create an XGBoost model

In [5]:
```
from sagemaker.amazon.amazon_estimator import get_image_uri

container = get_image_uri(session.boto_region_name, 'xgboost')
prefix = 'mailout'

xgb = sagemaker.estimator.Estimator(container, # The location of the container we wish to use
                                    role,                                    # What is our current IAM Role
                                    train_instance_count=1,                  # How many compute instances
                                    train_instance_type='ml.m4.xlarge',      # What kind of compute instances
                                    output_path='s3://{}/{}/output'.format(session.default_bucket(), prefix),
                                    sagemaker_session=session)

xgb.set_hyperparameters(max_depth=5,
                        eta=0.2,
                        gamma=4,
                        min_child_weight=6,
                        subsample=0.8,
                        silent=0,
                        objective='binary:logistic',
                        early_stopping_rounds=10,
                        num_round=500)

s3_input_train = sagemaker.s3_input(s3_data=train_location, content_type='csv')
s3_input_validation = sagemaker.s3_input(s3_data=val_location, content_type='csv')
```

```
'get_image_uri' method will be deprecated in favor of 'ImageURIProvider' class in SageMaker Python SDK v2.
There is a more up to date SageMaker XGBoost image. To use the newer image, please set 'repo_version'='1.0-1'. Fo
r example:
        get_image_uri(region, 'xgboost', '1.0-1').
Parameter image_name will be renamed to image_uri in SageMaker Python SDK v2.
's3_input' class will be renamed to 'TrainingInput' in SageMaker Python SDK v2.
's3_input' class will be renamed to 'TrainingInput' in SageMaker Python SDK v2.
```

We will use hyperparameter tuner job to find out best configuration.

Create hyperparameter tuner

In [6]:
```
from sagemaker.tuner import IntegerParameter, ContinuousParameter, HyperparameterTuner

# Create the hyperparameter tuner object
xgb_hyperparameter_tuner = HyperparameterTuner(estimator = xgb, # The estimator object to use as the basis for the
                                               objective_metric_name = 'validation:auc', # The metric used to compa
                                               objective_type = 'Maximize', # Whether we wish to minimize or maximi
                                               max_jobs = 10, # The total number of models to train
                                               max_parallel_jobs = 3, # The number of models to train in parallel
                                               hyperparameter_ranges = {
                                                    'max_depth': IntegerParameter(10, 100),
                                                    'eta'      : ContinuousParameter(0.05, 0.5),
                                                    'min_child_weight': IntegerParameter(2, 8),
                                                    'subsample': ContinuousParameter(0.5, 0.9),
                                                    'gamma': ContinuousParameter(0, 20),
                                               })
```

In [7]:
```
xgb_hyperparameter_tuner.fit({'train': s3_input_train, 'validation': s3_input_validation})
```

In [8]:
```
xgb_hyperparameter_tuner.wait()
```

We will create a batch transformer using best training job

Transform and save results.

```
In [11]: xgb_transformer.transform(test_location, content_type='text/csv', split_type='Line')

In [12]: xgb_transformer.wait()
```

```
.............................Arguments: serve
[2021-05-03 17:47:40 +0000] [1] [INFO] Starting gunicorn 19.9.0
[2021-05-03 17:47:40 +0000] [1] [INFO] Listening at: http://0.0.0.0:8080 (1)
[2021-05-03 17:47:40 +0000] [1] [INFO] Using worker: gevent
[2021-05-03 17:47:40 +0000] [20] [INFO] Booting worker with pid: 20
[2021-05-03 17:47:40 +0000] [21] [INFO] Booting worker with pid: 21
/opt/amazon/lib/python3.7/site-packages/gunicorn/workers/ggevent.py:65: MonkeyPatchWarning: Monkey-patching ssl
after ssl has already been imported may lead to errors, including RecursionError on Python 3.6. It may also sile
ntly lead to incorrect behaviour on Python 3.7. Please monkey-patch earlier. See https://github.com/gevent/geven
t/issues/1016. Modules that had direct imports (NOT patched): ['urllib3.util.ssl_ (/opt/amazon/lib/python3.7/sit
e-packages/urllib3/util/ssl_.py)', 'urllib3.util (/opt/amazon/lib/python3.7/site-packages/urllib3/util/__init__.
py)'].
  monkey.patch_all(subprocess=True)
[2021-05-03:17:47:40:INFO] Model loaded successfully for worker : 20
/opt/amazon/lib/python3.7/site-packages/gunicorn/workers/ggevent.py:65: MonkeyPatchWarning: Monkey-patching ssl
after ssl has already been imported may lead to errors, including RecursionError on Python 3.6. It may also sile
ntly lead to incorrect behaviour on Python 3.7. Please monkey-patch earlier. See https://github.com/gevent/geven
t/issues/1016. Modules that had direct imports (NOT patched): ['urllib3.util.ssl_ (/opt/amazon/lib/python3.7/sit
e-packages/urllib3/util/ssl_.py)', 'urllib3.util (/opt/amazon/lib/python3.7/site-packages/urllib3/util/__init__.
```

```
In [13]: data_dir = '../data/mailout'
         !aws s3 cp --recursive $xgb_transformer.output_path $data_dir
```

```
download: s3://sagemaker-us-east-1-333536604061/xgboost-210503-1725-004-59072e94-2021-05-03-17-42-59-815/testing.
csv.out to ../data/mailout/testing.csv.out
```

use the best training job

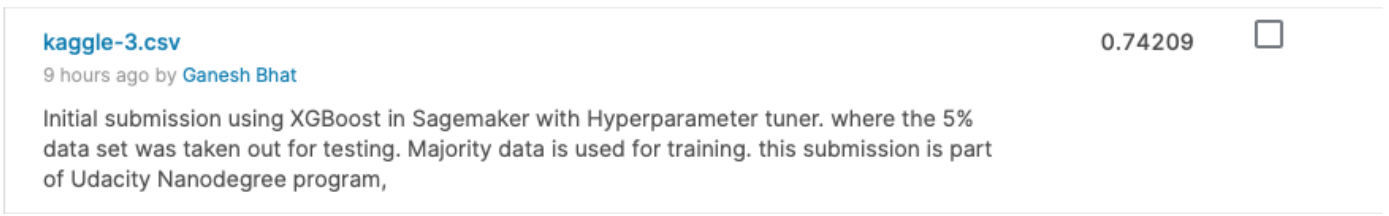
```
In [9]: xgb_hyperparameter_tuner.best_training_job()
Out[9]: 'xgboost-210503-1725-004-59072e94'
```

```
In [10]: # Create a new estimator object attached to the best training job found during hyperparameter tuning
         xgb_attached = sagemaker.estimator.Estimator.attach(xgb_hyperparameter_tuner.best_training_job())

         # Create a transformer object from the attached estimator.
         xgb_transformer = xgb_attached.transformer(instance_count = 1, instance_type = 'ml.m4.xlarge')
```

```
Parameter image_name will be renamed to image_uri in SageMaker Python SDK v2.

2021-05-03 17:33:38 Starting - Preparing the instances for training
2021-05-03 17:33:38 Downloading - Downloading input data
2021-05-03 17:33:38 Training - Training image download completed. Training in progress.
2021-05-03 17:33:38 Uploading - Uploading generated training model
2021-05-03 17:33:38 Completed - Training job completedArguments: train
[2021-05-03:17:33:21:INFO] Running standalone xgboost training.
[2021-05-03:17:33:21:INFO] Setting up HPO optimized metric to be : auc
[2021-05-03:17:33:21:INFO] File size need to be processed in the node: 153.87mb. Available memory size in the
de: 8411.38mb
[2021-05-03:17:33:21:INFO] Determined delimiter of CSV input is ','
[17:33:21] S3DistributionType set as FullyReplicated
[17:33:21] 15204x360 matrix with 5473440 entries loaded from /opt/ml/input/data/train?format=csv&label_column
delimiter=,
```

We received an accuracy score of 98% but less recall and precision.

To obtain AUROC, we will submit to Kaggle.



We received 74.209% score for AUROC, being the best score so far.

Summary

We have done the following as part of this section

Data Loading & Cleaning

- Load and prepare data
- Data exploration
- Data Cleaning
- Handling missing values
- Encoding categorical values
- Feature Scaling
- Remove highly co-related features

Supervised learning

- Split into train & test
- Prepare benchmark model and evaluate
- Use following models and check the accuracy
    - Benchmark model with Linear learner
    - Neural Networks with PyTorch
    - XgBoost
    - KNN & SVM (optional, if time permits)
        - Use hyperparameter tuner for betterment of results
- Evaluate the results using confusion metric and ROC score

Kaggle

Overview

Now that you've created a model to predict which individuals are most likely to respond to a mailout campaign, it's time to test that model in competition through Kaggle. If you click on the link here, you'll be taken to the competition page where, if you have a Kaggle account, you can enter.

Your entry to the competition should be a CSV file with two columns. The first column should be a copy of "LNR", which acts as an ID number for each individual in the "TEST" partition. The second column, "RESPONSE", should be some measure of how likely each individual became a customer – this might not be a straightforward probability. As you should have found in Part 2, there is a large output class imbalance, where most individuals did not respond to the mailout. Thus, predicting individual classes and using accuracy does not seem to be an appropriate performance evaluation method. Instead, the competition will be using AUC to evaluate performance. The exact values of the "RESPONSE" column do not matter as much: only that the higher values try to capture as many of the actual customers as possible, early in the ROC curve sweep.

Submission format

Submission must have 42833 rows

The csv should be of format

LNR	RESPONSE
4523	0.23544

We have created 3 models

- Linear Learner model
- Pytorch model with neural network
- XGBoost model

Here are the scores and leaderboard image

Linear Learner model : 0.63394

Pytorch model with neural network : 0.50048

XGBoost model with hyperparameter tuner: 0.74209

XGBoost has provided best possible scoare so far. I will try to improve this model further, however for the completeness of the course, this notebook will be submitted with this result.

Scorecard and ranking

References

1. https://towardsdatascience.com/customer-segmentation-in-online-retail-1fc707a6f9e6
2. https://towardsdatascience.com/customer-segmentation-with-machine-learning-a0ac8c3d4d84
3. https://www.researchgate.net/publication/263574034_Customer_segmentation_in_private_banking_sector_using_machine_learning_techniques
4. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.3440&rep=rep1&type=pdf
5. https://blog.dominodatalab.com/benchmarking-predictive-models/
6. https://towardsdatascience.com/benchmarking-simple-machine-learning-models-with-feature-extraction-against-modern-black-box-80af734b31cc
7. https://towardsdatascience.com/creating-benchmark-models-the-scikit-learn-way-af227f6ea977
8. https://towardsdatascience.com/get-interactive-plots-directly-with-pandas-13a311ebf426
9. https://www.proud2becloud.com/a-clustering-process-with-sagemaker-experiments-a-real-world-use-case/
10. https://www.brandwatch.com/blog/intelligent-market-segmentation/