

# Customer Segmentation Report for Arvato Financial Solutions

Jean-Thierry BONTINCK – Janvier 2020

---

*Udacity Machine Learning Nanodegree Capstone Project Report*

## 1. Summary

### 1.1. Project Overview

#### 1.1.1. Domain Background

This project comes from the marketing domain. It has been provided by Arvato Financial Solutions, a Bertelsmann subsidiary. It is a “real life” problem that a data scientist working at Arvato would work on. The aim of the project is to have a better understanding of customer’s profiles and, therefore, be able to target more efficiently marketing actions on prospects to gain new customers. This sounds like a classic data science problem and it is also very interesting. It is related to other domains like, obviously, marketing but also sociology.

#### 1.1.2. Personal motivation:

I’ve chosen to work on this project for different reasons. First, I’ve graduated from a Deep Learning Nanodegree, and I’ve already done project like the dog breed classifier project. I want to build experience something different and, also, to challenge myself. Furthermore, I live in an area where are located a lot of companies from the retail industry and the finance/insurance sector. This kind of project is very close to what is done daily by their data science teams. From a professional perspective, I therefore wish to develop practical knowledge on such a type of data science project that I don’t have yet in my portfolio.

#### 1.1.3. Datasets and Inputs

There are 4 datasets to be used in the scope of this project and injection of exogenous data is not allowed. Those datasets are supplied in a csv format (“;” separator) and are the following:

- **Udacity\_AZDIAS\_052018.csv:** Demographics data for the general population of Germany; 891,211 persons (rows) x 366 features (columns). This dataset will be used as a reference in order to reveal the features of the customers.
- **Udacity\_CUSTOMERS\_052018.csv:** Demographics data for customers of a mail-order company; 191,652 persons (rows) x 369 features (columns). With this dataset, we will compare how different/similar are the features of the customers in comparison with the previous dataset that covers the general population.
- **Udacity\_MAILOUT\_052018\_TRAIN.csv:** Demographics data for individuals who were targets of a marketing campaign; 42,982 persons (rows) x 367 features (columns). This dataset will be the base to create our prediction model.

- **Udacity\_MAILOUT\_052018\_TEST.csv**: Demographics data for individuals who were targets of a marketing campaign; 42,833 persons (rows) x 366 features (columns). This dataset will be used to test and evaluate the model created.

The dataset comes with 2 documents that explains what the columns in the datasets are and what is the meaning of the values that each feature can take.

## 1.2. Problem Statement

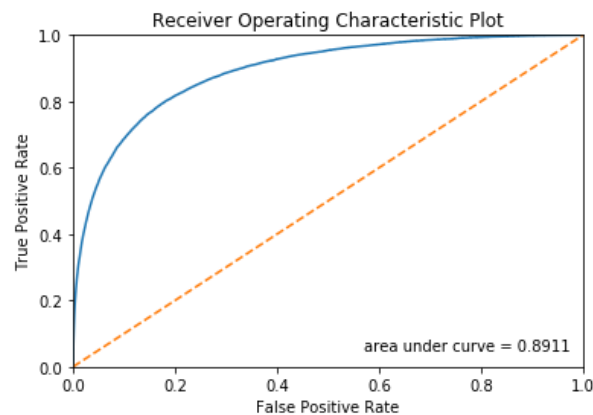
This project is a guided project and the key steps are given as follow:

1. A segmentation of the customers will be created using the whole population dataset and the customer dataset. This step implies using unsupervised machine learning methods.
2. A prediction model will be built to predict if a prospect will respond to the marketing campaign or not. This will be done using the 3<sup>rd</sup> dataset that contains information about prospects and whether they respond or not to a previous campaign.
3. The prediction model will be tested on data unseen by our model. This data comes from the 4<sup>th</sup> dataset that contains the same information about prospect (except if they respond or not). The test predictions given by the model will be evaluated and ranked on the Kaggle website against other student predictions.

To achieve this, there are a several steps that are part of the solution and critical for its performance that I want to mention: pre-processing, features selection and model prediction analysis. Before working with the dataset on segmentation, we will have to work on cleaning and pre-processing the dataset. We have been provided with raw data and it will be a crucial step. Then we will build our segmentation. We have more than 360 variables in the dataset, and we can anticipate that not all of them will be relevant for our goal. At the end, the segmentation should give us insights to perform a good feature selection that will allow our model to work efficiently and make accurate predictions. The last important step will be to understand the predictions that are made and their relationship with the test data. This is always an important step and, in our case, should also be done in the specific context of a Kaggle competition.

### 1.3. Metrics

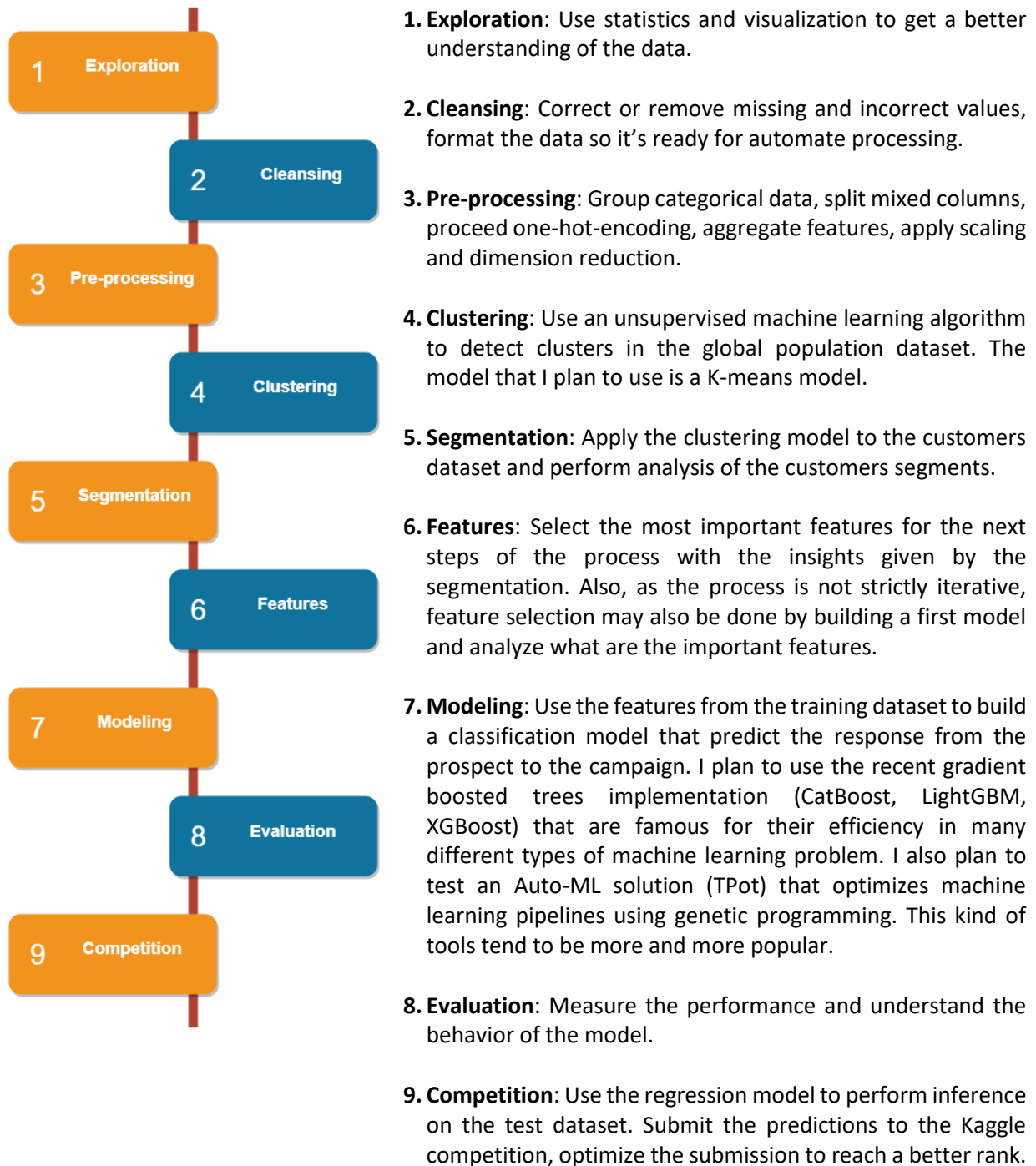
The final evaluation metric is already defined for this project and is the **area under receiver operating characteristic (ROC) curve**. It is often used to evaluate the performance of a binary classifier. The classifier will output a value between 0 and 1 that represent the probability that the outcome is “Yes” against “No” (or “Pass” against “Fail”, or any binary outcome). Often a threshold value will be used to define at what level the prediction will be understood as a 1. This threshold is in relation with the aim of the model and what we want to achieve in term of precision and recall. The idea of the ROC curve is to move this threshold from 1 to 0 and plot how the model is responding. The X axis represents the false positive rate (number of negative incorrectly detected positive). The Y axis the true positive rate (number of positive correctly detected positive). At a threshold of 1, there is no positive at all, the false positive rate and the true positive rate are 0. Inversely, at a threshold of 0, all is detected positive, the false positive rate and the true positive rate are both 1. When the classifier is random, both rates progress linearly and the ROC curve is a line from (0,0) to (1,1). The area under curve is then 0.5. When the classifier is perfect, the curve is squared (a horizontal line from (0,0) to (0,1) then a vertical line from there to (1,1)) and the area under curve value is 1. The ROC curve will be between those 2 limits and the area under curve value will be between 0.5 (random) and 1 (perfect).



Example of ROC curve

## 1.4. Project Design

The project includes 9 phases that are described below. In real life, those steps are not covered in a linear way and closed once achieved, but more executed in iterative loops with forward and backward moves. It's common that findings in one phase will require to move back and adapt a previous phase.



CatBoost <https://catboost.ai>  
 XGBoost <https://xgboost.readthedocs.io>

LightGBM <https://lightgbm.readthedocs.io>  
 TPOT <https://epistasislab.github.io/tpot>

## 2. Analysis

### 2.1. Data Exploration

I've mainly focus on the AZDIAS dataset that will serve as a reference in the project and because all datasets share the same data structure. We will have a look at the 3 other datasets and see what is specific to them. There are more than 360 features, so I won't provide detailed analysis for each column. Instead we will focus on the most important and highlight the key findings.

#### 2.1.1. Documentation, outliers & missing values

The 2 documents that explains the datasets are a very important to understand and handle the datasets. The first one (information level), enlighten about what are each field in the dataset, the name and the information encoded. All fields are grouped in different levels of granularity (person, household, building, microcell, cell, ...). The second one gives, for each field, a more detailed description of the information, the range of values, the meaning of each value if necessary. Here are the important points that I've spotted:

- Not all the data is described, some fields of the dataset are not in the documentation (about 57).
- Some fields are in the documentation but not in the dataset (about 9).
- There is some misspelling in some names given int the documentation compared to the names really in the datasets. For most of them, we can easily guess and corrected (D19\_HAUS\_DEKO\_RZ in the documentation, D19\_HAUS\_DEKO in the datasets).
- For most of the fields describes in the documentation, we have a range of typical or expected values.
- For many fields, there is a specific value for "unknown" that represents missing values. We have also some missing values where the field is left empty. It's then too early to analyze them as those unknow values should also be treated as missing value so we can have a meaningful overview.

#### 2.1.2. Type of data

The dataset contains a mix of different types of features:

- Ordinal: most of the features of the dataset are ordinal. For example, giving a level of activity from low to high. But I've also spotted that some are not well ordered. For example, D19\_HAUS\_DEKO\_RZ that starts with 0 (no transaction, then 1 (Multi-buyer) and ends with 7 (Prospects) where the order should be reversed on the 1 to 7 range to have a correct ordinal feature.
- Categorical: We have some are categorical features (about 30,35 features).
- Mixed: That's categories we must be careful with as they are mixing different kind of information We have for example PRAEGENDE\_JUGENDJAHRE that represents the dominating movement in the person's youth but mixed with generation and the area of the country (where 6 represent 60ies - generation 68 / student protestors, Avantgarde, W).
- Numeric: We have just a few of them, less than 10.
- Binary: 5 features.

### 2.1.3. Specific characteristics of datasets

I've pointed out some important differences between the datasets in a high-level view:

- There are no duplicates in the AZDIAS and Customers datasets, but we have a lot of them in Mail Train and Mail Test datasets, respectively 5,767 for 42,962 (13,42%) and 5,633 for (13,15%).
- There is a specific field in the Mail Train dataset ("RESPONSE") that will be our label class. The 2 classes are heavily unbalanced with 532 positive values (1,24%).
- There are 3 specific fields for the Customers that are not available in the other datasets (PRODUCT\_GROUP, CUSTOMER\_GROUP, ONLINE\_PURCHASE)

	<b>count</b>	<b>mean</b>	<b>std</b>	<b>min</b>	<b>25%</b>	<b>50%</b>	<b>75%</b>	<b>max</b>
AGER_TYP	891221	-0.358435	1.198724	-1	-1	-1	-1	3
AKT_DAT_KL	817722	4.421928	3.638805	1	1	3	9	9
ALTER_HH	817722	10.86413	7.639683	0	0	13	17	21
ALTER_KIND1	81058	11.74539	4.09766	2	8	12	15	18
ALTER_KIND2	29499	13.40266	3.2433	2	11	14	16	18
ALTER_KIND3	6170	14.47601	2.712427	4	13	15	17	18
ALTER_KIND4	1205	15.08963	2.452932	7	14	15	17	18
ALTERSKATEGORIE_FEIN	628274	13.70072	5.079849	0	11	14	17	25
ANZ_HAUSHALTE_AKTIV	798073	8.287263	15.62809	0	1	4	9	595
ANZ_HH_TITEL	794213	0.040647	0.324028	0	0	0	0	23
ANZ_KINDER	817722	0.154018	0.502389	0	0	0	0	11
ANZ_PERSONEN	817722	1.727637	1.155849	0	1	1	2	45
ANZ_STATISTISCHE_HAUSHALTE	798073	7.599356	14.3322	0	1	3	9	449
ANZ_TITEL	817722	0.004162	0.068855	0	0	0	0	6
ARBEIT	794005	3.167854	1.002376	1	3	3	4	9
BALLRAUM	797481	4.153043	2.18371	1	2	5	6	7
CJT_GESAMTTYP	886367	3.632838	1.595021	1	2	4	5	6
CJT_KATALOGNUTZER	886367	3.335264	1.493633	1	2	4	5	5
CJT_TYP_1	886367	3.368086	1.368331	1	2	3	5	5
CJT_TYP_2	886367	3.195014	1.401382	1	2	3	5	5
CJT_TYP_3	886367	3.35129	1.396508	1	2	3	5	5
CJT_TYP_4	886367	3.336151	1.373077	1	2	3	5	5
CJT_TYP_5	886367	3.360684	1.378992	1	2	3	5	5
CJT_TYP_6	886367	3.46598	1.328456	1	2	4	5	5
D19_BANKEN_ANZ_12	891221	0.122336	0.53595	0	0	0	0	6
D19_BANKEN_ANZ_24	891221	0.219907	0.747903	0	0	0	0	6
D19_BANKEN_DATUM	891221	9.26742	1.735725	1	10	10	10	10
D19_BANKEN_DIREKT	891221	0.892735	2.011838	0	0	0	0	7
D19_BANKEN_GROSS	891221	0.56858	1.643764	0	0	0	0	6
D19_BANKEN_LOKAL	891221	0.106769	0.808179	0	0	0	0	7
D19_BANKEN_OFFLINE_DATUM	891221	9.926794	0.605641	1	10	10	10	10

Extract of the statistical description of features

## 2.2. Exploratory Visualization

### 2.2.1. *Main correlations between variables*

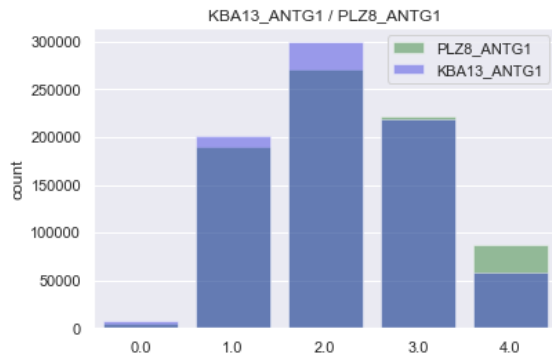
Some data science algorithm can be sensitive to highly correlated values. That's the case of the PCA that I've planned to use (<https://stats.stackexchange.com/questions/50537/should-one-remove-highly-correlated-variables-before-doing-pca>). I've therefore analyzed correlations between variables. You will find below a heatmap of correlations. Some features seem to be highly correlated and to provide information around the same topic; some are easy to catch on the plot below on colored area. We can notice data about transactions (like D19\_BANKEN\_DATUM and close one), about cars owning (like KBA05\_AUTOQUOT and close), about age of people in the area, about type of building...



*Correlations between features heatmap*

### 2.2.2. Combine correlated features

We could have dropped some highly correlated features, but I've analyzed the idea to combine some of them. The principle would be to take 2 correlated features with a close information space, to average them when they are both values are present, to keep the value of one or the other in case of one missing values. This should be done carefully with features on the same scale. My goal is to have more reliable and less correlated features. To illustrate that, I've taken the features PLZ8\_ANTG1 and KBA13\_ANTG1 that represents both number of 1-2 family houses in the area but at different geographical levels.



PLZ8\_ANTG1 - KBA13\_ANTG1

Correlation: 0.9189

Missing values for PLZ8\_ANTG1: 116,515 (13.07%)

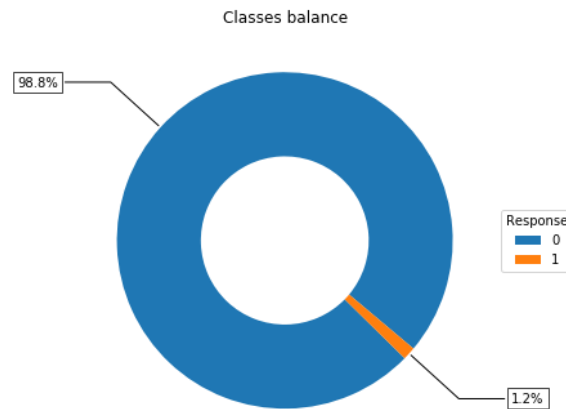
Missing values for KBA13\_ANTG1: 105,800 (11.87%)

Resulting missing values for PLZ8\_ANTG1 and KBA13\_ANTG1 combined: 104,957 (**11.77%**)

Example of features combination

### 2.2.3. Classes balance for the Mail Train dataset

The "donut plot" below shows the class imbalance for the Mail Train dataset.



Classes balance (Mail Train, RESPONSE)



## 2.3. Algorithms and Techniques

There are 2 data science problems to solve: identify the main characteristic of the customers and predict prospect that are likely to respond to a mail campaign. We will then start with a n unsupervised learning algorithm for the first problem and perform a supervised learning algorithm for the second.

### 2.3.1. Clustering

For the customer segmentation, I've chosen to use the K-means algorithm. K-means is an unsupervised learning technique that aims to find coherent groups inside a dataset. Before the algorithm starts, the number of K centers is fixed, and the initial centers are chosen, usually randomly. K-means is then run iteratively in a 2 steps process. First each point of the dataset is assigned to the closer cluster center (based on the Euclidean distance), and secondly, each cluster center is moved at the center of all points assigned to it. The process stops after a given number of iterations is reached or, better, when the distance is under a given value.

This algorithm is well suited for a first level exploration. The difficult part of unsupervised learning is the evaluation of the model. For the K-means, we can plot the distance for each K number of clusters and use the "elbow method" to help us choose the best K value. It's often a good first step to engage a clustering project. It could be followed then by another algorithm, for example, a hierarchical clustering, that exploit use the K value found suing the K-means.

One drawback of K-means is that it suffers from the curse of dimensionality. We will apply PCA before to help our model. This is a classic schema.

### 2.3.2. Classification

On the classification side of the project, we will first use a logistic regression as a benchmark model. Logistic regression is a binary classifier. It's very close to the linear regression but the outcome of the approximation function is bound between 0 and 1 and represents a probability. The logistic regression will use a linear combination of all the input variables and use a sigmoid function on top of that. The value of the coefficients is found iteratively. Also, common implementation (like the one from Scikit-Learn that we will use) includes a regularization to the logistic regression. The regularization will constraint the algorithm complexity in order to avoid overfitting.

We will then try get improvement by implementing the best-in-class gradient boosting algorithms. Those algorithms come from the "ensemble" family of models. An ensemble, more than a model, is a group of models. The underlying idea behind is that, with the combination of a lot a weak models, you can build a final model that will be stronger than each of the weak models used. For example, random forest is an ensemble of decision trees. To combine the model together, there are different techniques like "bagging" or "boosting". "Bagging" could be seen as a simple vote between the model. "Boosting" is a more elaborated way of combining models together, where models are added by layers with each new layer trying to correct the specific weakness of the previous layers. The additive model in the ensemble is then more trying to predict the residual between the label and the existing layer than the label itself. The gradient term means that the way layers are added together includes the gradient of the loss function (and a coefficient, the learning rate). This gradient term will then lead the overall model in the direction of minimizing the loss function.

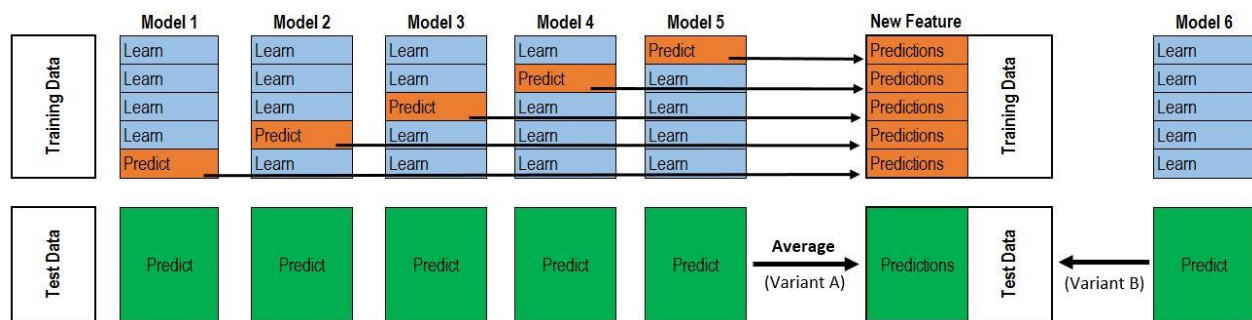
XGBoost, Light-GBM and CatBoost are common implementation of gradient boosting that we will use. Each of those algorithms comes with many parameters, some for the algorithm itself (like the number of iterations or the learning rate), some related to the weak learner used. XGBoost is the first of the three to have been released and it has been widely used. XGBoost uses decision trees as weak learner and introduced new regularization techniques very efficient to avoid overfitting. Light-GBM has been designed by Microsoft and implements a different way to grow the trees. Light-GBM brought better accuracy, large scale handling and GPU learning. CatBoost, the latest released, came also with its own way to grow trees (symmetrically) and to handle categorical inputs (reason of the “Cat” is the name).

Those algorithms are efficient but also very robust. They can perform well even in case of unscaled input data (decision trees behind) and in case of large number of and/or correlated features (due to the gradient process that will help to balance the importance of a feature). To achieve the best level of performance, we will use grid search for hyperparameters.

We will also try to use an Auto-ML framework, **TPot**, to help us find alternative solutions and maybe more performant solutions. TPot uses genetic programming concepts to build a large base of models (population) and make them mutate to improve performance (generations). It’s a lot of computational time as the process will test and try thousands of models, but also less human time involved. It can give interesting insights on the type of model to use.



Lastly, we will assemble the models in an ensemble using stacking. The principle is to use K-folding and fit each model to create K versions for each model types. If 3 type of model are used over 5 folds, we will then create 5 models of each types = 15 models. We will use the evaluation set of each fold to generate new data and then build a meta-model to mitigate the results of each type model. When the model is used for prediction, the result of the 5 models of our example are averaged, we have then 3 results. And a final result is given by the meta model using those 3 averaged predictions. We can expect to have more accuracy of the model, but it would cost more in terms of computation. To give a prediction, our example model perform inference on 5 models 3 times and then on the meta model = 16 models inference. This has been inspired by this: <https://www.kaggle.com/serigne/stacked-regressions-top-4-on-leaderboard>.



Model stacking principle

Furthermore, the way the meta-model balances the different types of model can also give interesting insights on the relative performance of model types.

## 2.4. Benchmark model

As a benchmark model, I plan to use a simple logistic regression. This model will be trained and test on the same data that our final model. This will give us a reference for the measured performance of our final model. It's important to keep in mind that the complete evaluation of a model is also linked with the objective that is followed. Depending on the context, the human time spent, the computational effort to design, fit or perform inference on the model and many other parameters can weight in the balance. And this can lead the choice in the direction of an algorithm or another.

### 3. Methodology

#### 3.1. Data cleansing and preparation

Here are the steps covered during the pre-processing stage.

- Remove features: I've removed several features for various reason. The list is given in the table below. Concerning the 57 features that are in the dataset but not documented, I've decided to keep them and to guess the type for each (24 ordinal, 6 categorical, 5 numeric, 4 binary, 1 date).

LNR	Line number, not a feature
EINGEFUEGT_AM	Date. Not referenced and, from the translation, seems to be the insertion date
ONLINE_PURCHASE CUSTOMER_GROUP PRODUCT_GROUP	Customer data only
ALTERSKATEGORIE_FEIN LP_LEBENSPHASE_FEIN LP_STATUS_FEIN LP_FAMILIE_FEIN CAMEO_DEU_2015	Categorical features with a high number of categories

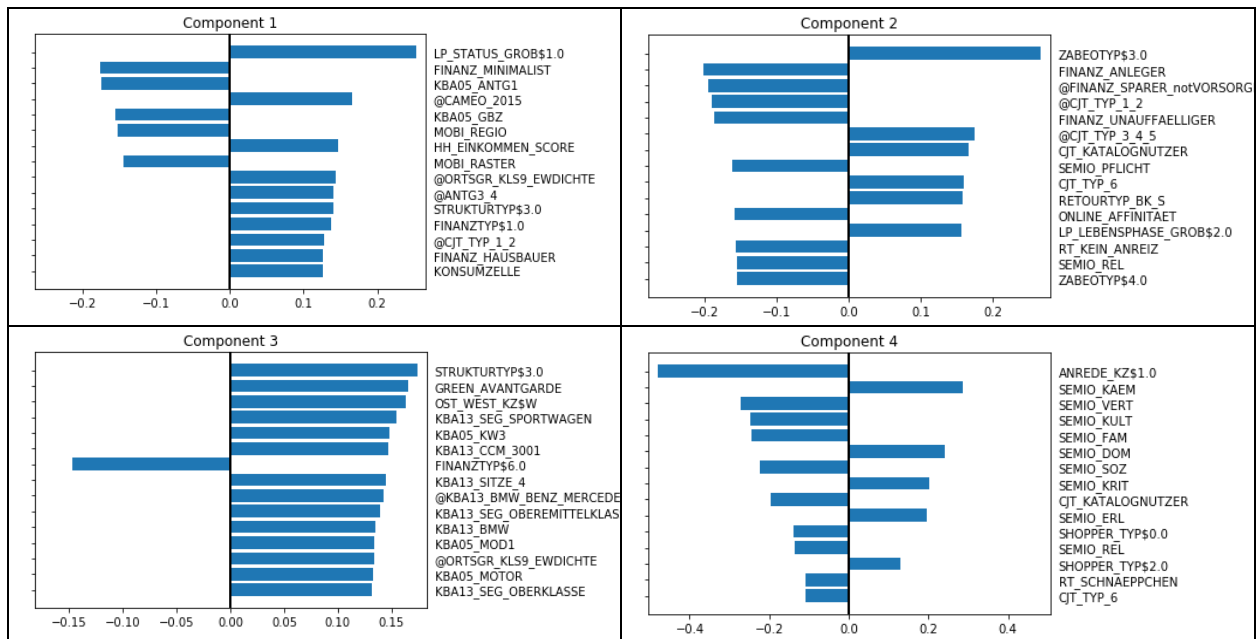
#### Features removed

- Perform special transformations: this step groups all the transformations that I haven't been able to handle in a generic way with parameters. For example, I've scaled the GEBURTSJAHR (year of birth) to match the decade and combine it with some other features related to the age.
- Check expected values: when needed, this step check that values are in the expected set of values. If not, values are replaced with n/a values.
- Create n/a: given the unknow values from the documentation, those values are transformed to n/a values to be treated the same way as missing values.
- Copy columns: copy some columns in order to perform distinct transformation. For example, I've copied PRAEGENDE\_JUGENDJAHRE that represents the decade, the person youth's movement and the area of origin (east/west). I will use the original column to transform it into a decade features and the copied one to extract the youth's movement features. The area of origin is already coded in another feature and match exactly.
- Apply mapping: this step covers a lot of features. It maps original values to new one. I have often used that to have really ordinal values for a feature not properly ordered, but also to scale some features before combining them.
- Categorical encoding: categorical features are one-hot encoded in this step.
- Combine features: as explained previously, this step combines features together, averaging them and covering the missing values when encountered.
- Drop features: features removed at the end of the process (for example, binary features are encoded like categorical but one of the two is removed during this stage).

## 3.2. Implementation

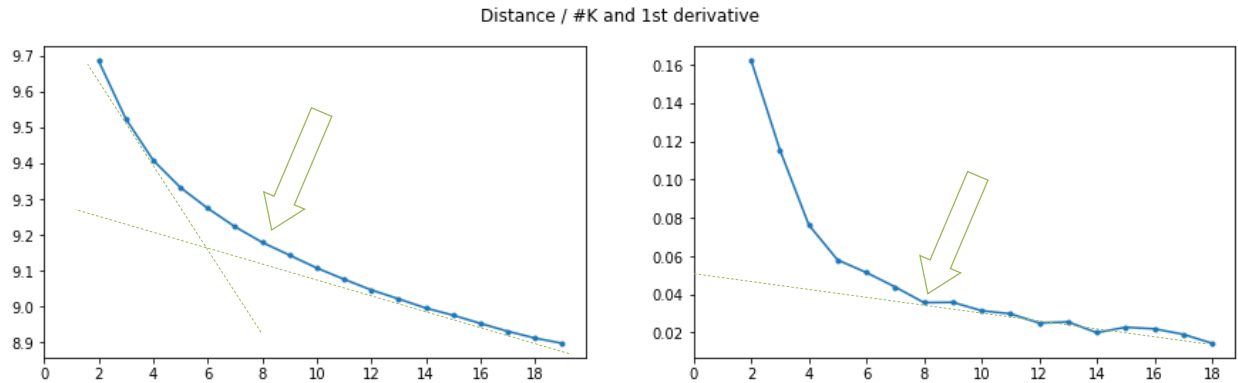
### 3.2.1. K-means

Before implementing the K-means, the data needs to be complete some pre-processing steps. It must be cleaned and prepare like described previously, but also have the missing values imputed, scaled, and decomposed in principal components. I've used a simple imputation strategy (median constant value), a min/max scaler and the PCA implementation of Scikit-Learn. A plot of the main features used by the 4 first components is shown below. The component 1 is mostly about the wealth and the living area. The component 2 includes also features about the finance but more about the customer behavior. The component 3 takes in account the youth-s movement and several features about types of car in the living area. The component 4 is about the gender, the affinity and personality.



PCA first components explained

I've then fit the data transformed into PCA in the K-means algorithm with several values of K (from 2 to 19). For each case, I've calculated the mean distance to the cluster center. As it was not easy to detect the elbow, I've also plotted the difference with the previous average distance to help detect a change in the curve.



#### Choosing K: the elbow method

The best K value is in between 4/5 and 8/9. I've chosen the value of 8 because it seems to me that it's the last value before a change in the slope occurs. I've also had a look at the results of several values to strengthen my choice. I have then fitted the K-means and used the customers data to see to identify the cluster for each customer.

#### 3.2.2. Classification

**Missing values and duplicates:** I've chosen to keep both the lines duplicated and the one with a high level of n/a values. I did that simply because the final score on the Kaggle competition were better when keeping them. But in another context, I'd rather remove most of them. Also, on the prediction side, I've noted that there are only 7 different lines in the 5,633 duplicates lines and that means that they will have a significant weight in the final score (we will dig into that later). I've also noticed that those lines come with many missing values, so even the prediction may be not very accurate for those. This means that if we were about to launch a marketing action, we would have to decide if we want to include those one or not, rather than rely on the prediction.

**Feature creation:** I've reused the clustering model to find the best cluster for each line of the Mail Train and Test datasets (I've then performed one-hot encoding). I've also added the distance to each cluster center as additional features.

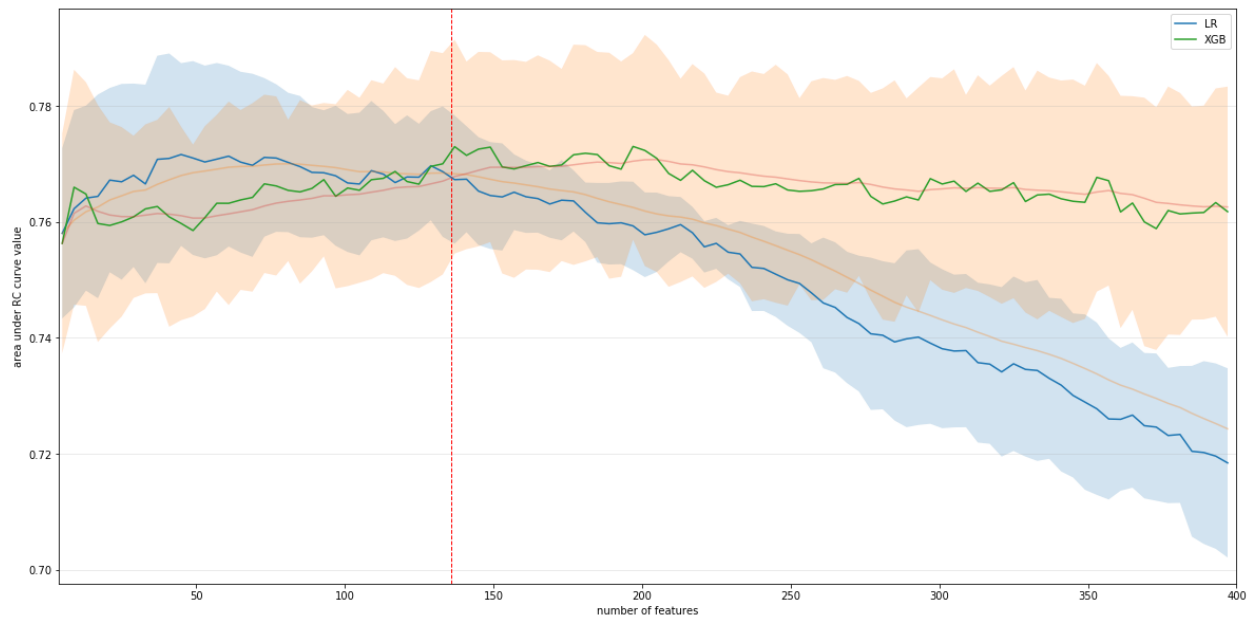
**Pre-processing:** Like for clustering, the data should be pre-processed before being fitted in the models. I've use a KNN imputer to fill the missing values as I thought it could give a more accurate imputation. I've then scaled the data using a standard scaler.

**Dimension reduction:** I've computed the PCA in the project of a parallel branch of the project using this data, but I've kept the original pre-processed dataset as the main work dataset.

**Benchmark model:** at this stage, I've fitted a basic logistic regression model and a first XGB model with out-of-box parameters. Both models will serve in the process to measure the progression made and the effects of the treatments considered. I've also chosen a value of 5 for the folding number.

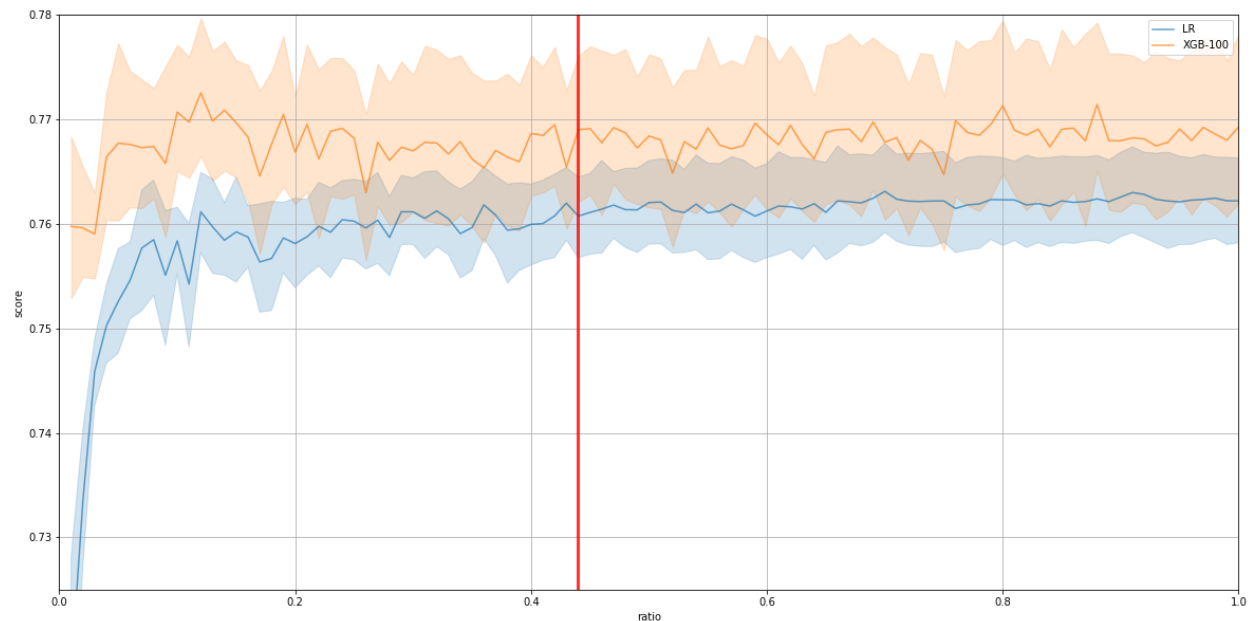
**Feature selection:** we have a lot of features, and I've then considered using a feature selection tool. I've used the SelectKbest utility from Scikit-Learn that rank features using a correlation measure with the label. I've decreased the number of features selected and plot the area under ROC curve score value for both

the benchmark model and the XGB simple model (plot below). It's interesting to notice that feature selection benefits the logistic regression but not really to the XGB model.



Score vs. number of selected features

**Down-sampling:** Because of the unbalanced class that we have already noticed, I've considered down-sampling as a way to improve to positive label ratio in the dataset. I've tried to assess what would be the effect on down-sampling on the score by measuring the area under curve for different down-sampling ratio. This is shown on the visualization below. We can see that with the ratio from about 0.4 to 1.0, the score is about the same. I would have expected some improvements.



Score vs. down-sampling ratio

**Note about down-sampling:** I've created a class to perform down-sampling the way I wanted to do it. But I discovered that it was not compatible with the way Scikit-Learn works with folding especially with the `cross_validate` function and `GridSearchCV` object.

**Gradient boosted models hyperparameters grid-search:** I've used grid search to look for the optimal parameters for the XGBoost model. I've performed several rounds starting with the high-level parameters (booster, n\_estimators, learning\_rate, max\_depth), then more fine grain parameters (colsample\_bylevel, colsample\_bynode, colsample\_bytree, subsample). I finally tuned parameters linked to the class balance (scale\_pos\_weight, min\_child\_weight) and to the regularization (gamma, reg\_alpha, reg\_lambda). I've then done the same process but with related hyperparameters for Cat-Boost and Light-GBM model.

**TPot Auto-ML:** Implementing TPot Auto-ML is straight forward, but it runs for days. I've let it run in the daytime a couple of days, about 24h in total.

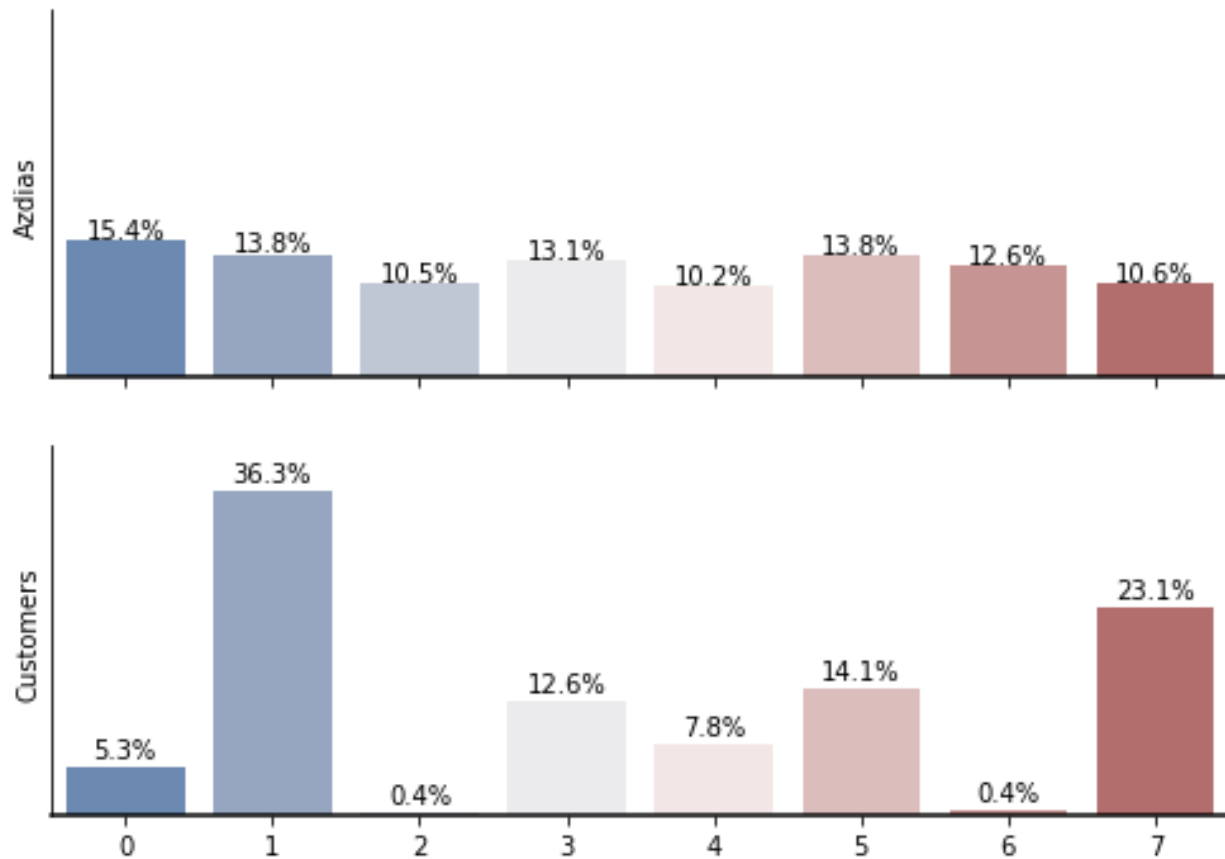
**Stacked models:** Stacking is, with bagging and boosting, a common way to build ensemble models. I've already experienced the implementation of stacking. It needs to produce a coding effort and, if N models are stacked, the resulting model is made of N+1 models. In consequence, it affects the training time, the size of the model and the inference time.



## 4. Results

### 4.1. Customer segmentation

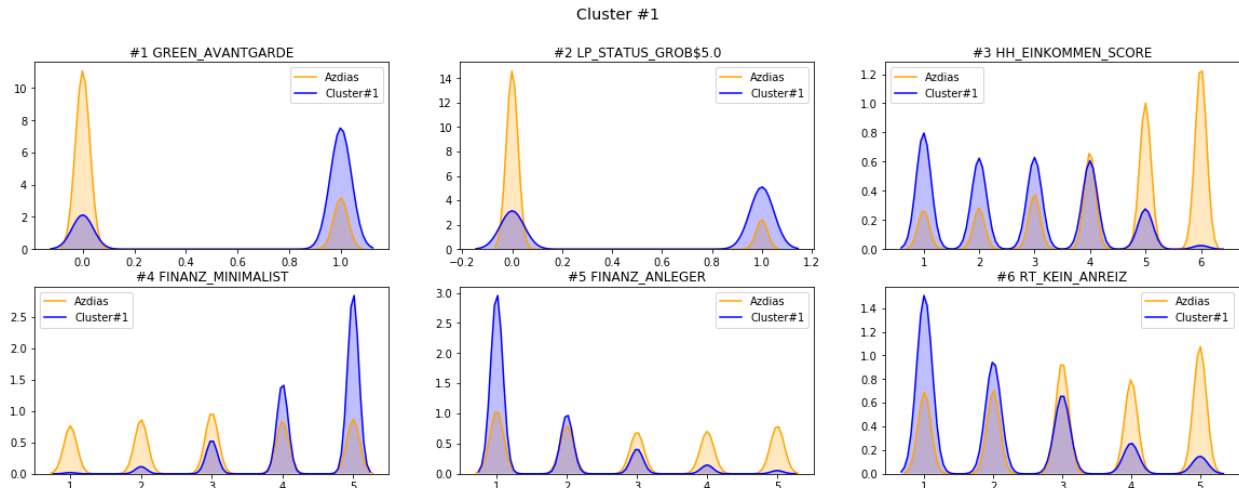
The bar chart below shows the distribution into clusters for the Azdias and Customer datasets.



#### Choosing K: the elbow method

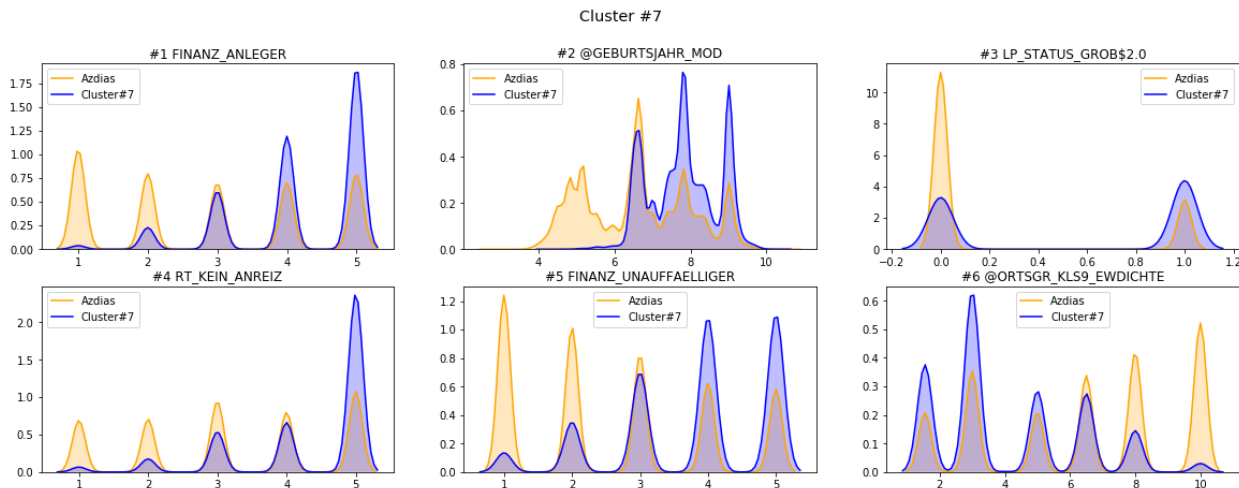
We have 2 clusters (#1 and #7) more represented in the customer dataset than in the Azdias dataset. There are 3 clusters with a close distribution in both datasets (#3, #4, #5). Finally, there are 3 clusters not rarely represented into the customers dataset (#0, and strongly #1, #6).

To extract the main representative features for each cluster, I've used the centroids of each cluster. I've first applied the inverse PCA. And I've then calculated the difference between the average for the cluster and the average for the Azdias reference dataset that I've normalized by dividing by the standard deviation. The features where the absolute values for this indicator are the most important have been spotted as the representative features of the cluster.



*Representative features of cluster #1 (value distribution in the cluster vs reference dataset)*

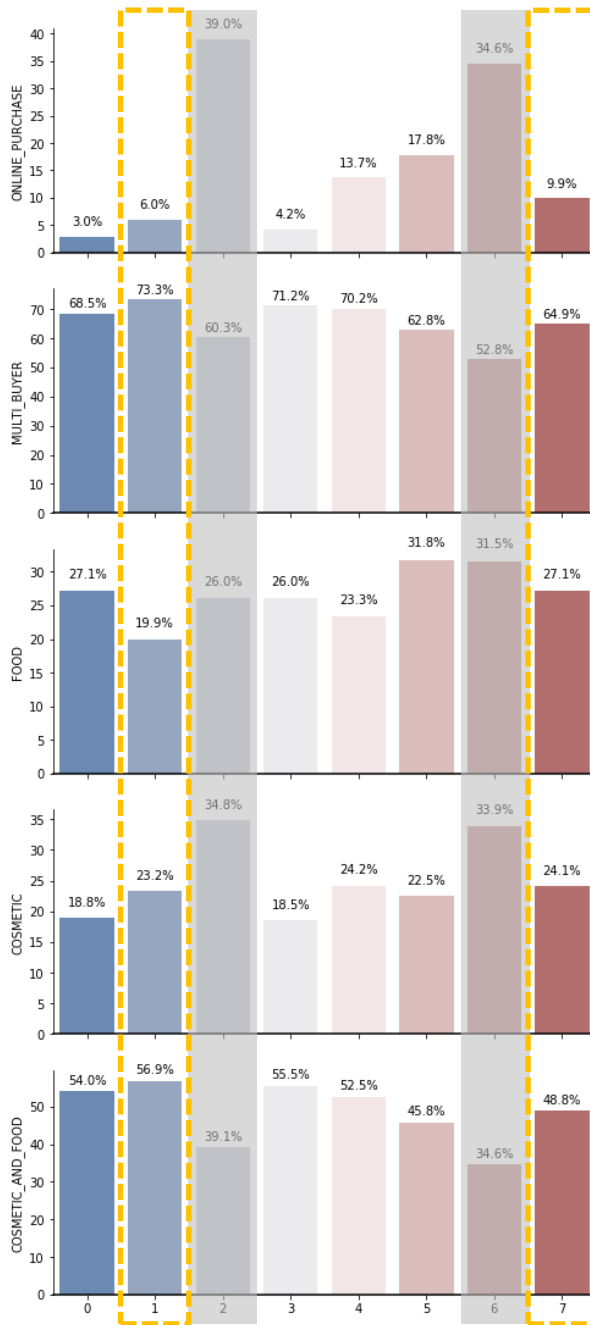
**Cluster #1:** The environmental sustainability is most likely to be the dominating movement in the youth of these consumers. They are most likely in the top earners, with net income in the household most likely in the high to highest categories. They have most likely not a low financial interest and most likely investor. Unfortunately, the RT\_KEIN\_ANREIZ is not documented, but, from the translation, it seems related to an incentive.



*Representative features of cluster #7 (value distribution in the cluster vs reference dataset)*

**Cluster #7:** Customers from cluster #7 are more likely from the 80's, 90's decades and live more likely in areas where the population density is lower. It's interesting to note that the customers that belong to this cluster seems opposite on some main characteristics to the one of the cluster #1. First on their relationship with finance, as they are not likely investors and most likely average earners. They are also more likely opposed on the RT\_KEIN\_ANREIZ (?) feature to the customers of cluster #1.

Finally, I've analyzed the customer specific data with the segmentation lenses. The bar charts below represent the percentage of each feature in the different clusters. The 2 gold clusters are surrounded with a yellow dashed line and the 2 out of scope clusters are grayed.



Customer data per cluster

If we compare the cluster #1 to the cluster #7 we can note that customers from the cluster #1 are less likely online purchaser, a more likely multi-buyer, more likely customers in the cosmetic&food group of products and a more likely cosmetic products purchaser in general ( $56.9\%+23.2\%=80.1\%$  vs.  $48.8\%+24.1\%=72.9\%$ ). They are both equivalent on the food products group.

The rare customers from the cluster #2 and #6 (that represent each only 0.4% of the customer base) are clearly more online buyer.

*Note: All those comments should be taken with caution as they may not be statistically relevant and would need further investigations to be proven.*


























































I'm personally surprised that a mail order company can have most of its customers as catalog buyer and not more online purchasers. This tickles my curiosity about they are and what is their business.

As a final thought, the qualitative results are aligned with company profile. The Arvato client is a mail-order company selling organic products, so the clusters found seems coherent. We have on one side, people wealthy from the green avant-garde and, on the other, younger people living in away from the population dense area.

## 4.1. Classification

### 4.1.1. Model Evaluation and Validation

The table below presents the different metrics for all models built.

Model	AUC Score CV	AUC Score CV - Std Dev.	Fit time (seconds)	Stacked model Meta-mod. coeff.	Exec. Time (seconds)	AUC Score Test
Logistic Regression (LR)	 0.721	0.011	 62.3	-	 0.2	 0.759
LR +PCA	 0.739	0.018	 5.3	-	 0.1	 0.787
LR +feature selection	 0.770	0.011	 4.4	 +14.03%	 0.1	 0.789
LR +feature selection +downsampling	 0.764	0.007	 1.8	-	 0.1	 0.780
XGB	 0.759	0.016	 7.3	-	 0.4	 0.803
XGB +PCA	 0.714	0.013	 9.2	-	 0.3	 0.715
XGB +feature selection	 0.769	0.013	 6.7	-	 0.2	 0.800
XGB +feature selection +downsampling	 0.763	0.015	 3.0	-	 0.2	 0.796
XGB +feature selection +grid search	 0.764	0.024	 9.9	 +23.35%	 0.3	 0.799
CatBoost +feature selection +grid search	 0.778	0.017	 3.4	 +38.27%	 0.1	 0.805
Light GBM +feature selection +grid search	 0.768	0.018	 0.7	 +7.40%	 0.2	 0.793
Tpot model (Extra Trees Classifier)	 0.771	0.011	 9.7	 +16.96%	 0.6	 0.782
<b>Stacked model</b>	 0.778	0.015	 113.0	-	 3.5	 0.799

Classification models results table

The table is composed with the following information:

- **Model:** Model type and version
- **AUC Score CV:** average area under ROC curve score returned by the cross validation (training)
- **AUC Score CV – Std Dev.:** standard deviation of the scores returned by the cross validation
- **Fit time (seconds):** is the average time in second to fit 1 model
- **Stacked model Meta-mod. coeff.:** for the 5 models that have been stacked in the last model, it gives the relative contribution (percentage) to the meta-model. In other words, it tells us how much the stacked model trusts this model to build the final prediction
- **Exec. time (seconds):** the execution time to build the prediction on for the entire test dataset
- **AUC Score Test:** the AUC score returns after submission on the Kaggle website

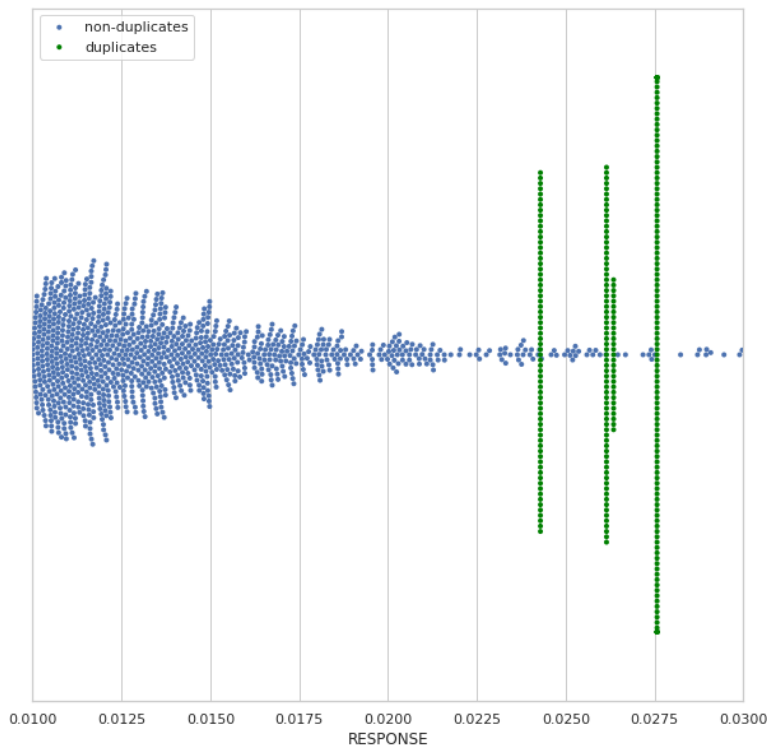
### 4.1.2. Justification and comments

Here are all my comments about those results:

- **About logistic regression models:** With a bit a work on features, we have achieved a good score on a simple logistic regression model, close to what I've seen on the Kaggle competition board.
- **About XGBoost models:** I'm not surprised that we had a good score with our only cleaned dataset and before performing any type of as XGBoost is famous for its robustness and its capacity to handle a lot of features and extract the most of it. But I'm surprised that we did not achieved a better score by using grid search
- **AUC Score CV – Std Dev.:** This value shows that our training score is not very accurate.
- **AUC Score CV / AUC Score Test:** I've calculated the correlation between the 2 (0.841). So, the 2 are highly correlated but there is still room for a difference.

## 5. Conclusion

### 5.1. Hacking the Kaggle leaderboard



[Focus on RESPONSE distribution in the submission file](#)

The visualization beside represents a sample of the final prediction submitted to the Kaggle competition. Each point represents a prediction case, a line in the submission file. The X-axis represents the value for the response column.

I mentioned earlier in the document that the Mail Test dataset had more than 10% duplicates. But there are only 7 different lines in those duplicates! That's why we have those values with a high density of points in green.

As they weight a lot in the score by their number (10%) and as they are just a few (7), it is possible to adjust their values in order to achieve a better score on the leaderboard Kaggle competition....

### 5.2. Reflection

I spent much more time on this project than I initially planned to. That was the first time I was using a clustering algorithm outside of simple exercises and I found that it was very stimulating from a data science perspective and on a more general point of view. I surprised myself late in the evening reading articles about persona in marketing or the history of Germany.

This project was also very challenging. The number of features made in more difficult than other projects I've worked on. Furthermore, it came with a wide scope of issues that can be found in real life project: all kind of data, missing or incorrect values, duplicates, documentation not up to date, unbalanced classes and all the ones I've probably not seen. This was a great playground.

On another side, I feel I've improved my coding experience the hard way. I had issue because of a small bug in the pre-processing at the very beginning of the project that totally ruined the score of the model at the very end of it. I remembered that it was told in the lecture that a bug in data science can sometimes be very quiet and append without raising a big error message to many people. But it has a very bad effect at the end.

### 5.3. Improvement

There is a lot of room for improvement and experience. Here are a few tracks to follow:

- **Clustering:** Other algorithm could be applied to valid or improve the current segmentation.
- **Classification/Features:** I've worked a bit on preparing the features, and the logistic regression model benefited from that. But maybe having more raw features could have help to have slightly better gradient boosted type models. Also, Cat-Boost and Light-GBM can accept categorical features without prior one-hot encoding. Looking at the good score achieved by the Cat-Boost model, I think it would be worth the try.
- **Classification/Duplicates & n/a rows:** I've kept duplicated and rows with a high level of missing values in the train set, but I wouldn't have done that in another context. I've done this in the context of the Kaggle competition seeing I was having a better score. But I'm wondering if this could help models to be a little more optimized during the grid search or the auto-ml processes.
- **Classification/Number of folds:** I've used a value of 5 mostly because it's a common value and to limit the computation time. Using a higher value may lead in more consistent results in the AUC score and therefore could help the grid search and auto-ml processes.
- **Classification/Class imbalance:** I've tried to deal with the class imbalance using down-sampling without having significant results. I've just seen that it was not affecting the score much, but not increasing it. I would like to try other technique like SMOTE (Synthetic Minority Over-sampling Technique) for example. SOMTE allows to generate more observations for a less represented class.

### 5.4. Thanks

I've started my learning journey in September 2018 by a Deep Learning Nanodegree and I'm closing the loop with this Nanodegree, that's why I would like to first thank **Udacity** that have had a real impact on the course of my professional life. I wish their inspiring vision about the place of learning in life will continue to find a fertile soil in me to grow. I would like to thank **AWS & Udacity** that gave a sit in this nanodegree course through the AWS Deep Racer challenge. I would like to thank **Kaggle** that hosted the competition of this project but also provides a great community and knowledge base for data science aficionados. And I want to thank **Arvato/Bertelsmann** that provided this challenging dataset and use case!



## 6. Table of Contents

1. Summary .....	1
1.1. Project Overview .....	1
1.2. Problem Statement .....	2
1.3. Metrics .....	3
1.4. Project Design .....	4
2. Analysis .....	5
2.1. Data Exploration .....	5
2.2. Exploratory Visualization .....	7
2.3. Algorithms and Techniques .....	9
2.4. Benchmark model .....	11
3. Methodology.....	12
3.1. Data cleansing and preparation.....	12
3.2. Implementation .....	13
4. Results.....	17
4.1. Customer segmentation .....	17
4.1. Classification .....	20
5. Conclusion.....	21
5.1. Hacking the Kaggle leaderboard .....	21
5.2. Reflection .....	21
5.3. Improvement .....	22
5.4. Thanks .....	22
6. Table of Contents .....	23