# Capstone report for obtaining the Udacity Machine Learning Engineer Nanodegree

# Contents

[Capstone report for obtaining the Udacity Machine Learning Engineer Nanodegree 1](#_Toc44570306)

[Problem definition 1](#_Toc44570307)

[Analysis 2](#_Toc44570308)

[Data exploration and exploratory visualization 2](#_Toc44570309)

[A specific notebook (Arvato Project Workbook\_DataExploration.ipynb) is dedicated to data exploration and analysis. This notebook has been used to determine statistics such as : 2](#_Toc44570310)

[Amount of missing values for each feature 2](#_Toc44570311)

[Imbalance of mailout training dataset 4](#_Toc44570312)

[Conclusion 5](#_Toc44570313)

[Algorithm and techniques 5](#_Toc44570314)

[Benchmark 6](#_Toc44570315)

[Methodology 6](#_Toc44570316)

[Data pre-processing 6](#_Toc44570317)

[Implementation 6](#_Toc44570318)

[Refinement 6](#_Toc44570319)

[Results 6](#_Toc44570320)

[Conclusion 7](#_Toc44570321)

# Problem definition

This capstone project deals with customers data coming from Arvato Financial Solutions, a Bertelsmann subsidiary. Arvato is a global services company including customer support, IT, logistics and finance.

The issue at hand is customer segmentation and classification i.e. being able to derive the traits of a customer of the specific company Arvato is providing services to compared to general population and then be able to state whether or not a person, based on his traits, will convert to a customer?

The following datasets were provided as four different files :

* A general population file containing 891 211 samples (persons) with 366 features for each sample;
* A customer file containing 191652 samples with 369 features i.e. 366 features are identical to the general population file while three additional ones are provided : CUSTOMER\_GROUP, ONLINE\_PURCHASE and PRODUCT\_GROUP identifying the type of customer being described;
* A TRAIN mail campaign file containing 42982 samples and 367 features i.e. 366 features being identical to the general population file and one additional feature identifying whether or not the person became a customer following the campaign;
* A TEST mail campaign file containing 42833 samples being identical to the TRAIN file except that the target (became customer) has been removed.

Additionally, Excel files providing more information about the features of the customers and general population (levels, type of information, …) were also provided.

The problem at hand is therefore a supervised leaning problem. One needs to use the general population and customer files to try devising the features that distinguish a customer from a non-customer. Once these features are identified, one will build a model to infer if a given person will become a customer or the likelihood of this event for that particular person.

To solve this problem, the following techniques have been used :

* Dataset exploration using visualizing techniques (histogram, …);
* Feature-cleaning, engineering and cleaning as well as feature space reduction using Principal Component Analysis;
* Supervised model in the form of XGBoost.

For the supervised model, given the high imbalance of the dataset (only 2% of customers (target response = yes) among the data), using an accuracy scoring metric would be ill-advised. Therefore, one uses the Area Under Curve (AUC) metric. The objective is set to binary logistic given a binary classification is at hand.

# Data Analysis

## Data exploration and exploratory visualization

A specific notebook (Arvato Project Workbook\_DataExploration.ipynb) is dedicated to data exploration and analysis. This notebook has been used to determine statistics such as :

* Amount of missing data for each feature and for each dataset,
* Amount of missing data among samples for each dataset
* Type of features and number of levels for categorical features
* Imbalance of the mailout training dataset

### Amount of missing values : features analysis

#### For the features for which information is available in the Excel files

Figure 1 shows the ratio of natural[[1]](#footnote-1) NaN per feature for the population, customers as well as the mailout datasets and sorted by the highest number of occurrence in the population dataset. Figure 2 shows the same information but for the total number of NaN. The equivalent NaN have been identified based on Excel file information. Note that this was only possible for the features featured in those Excel files i.e. 260 out of 366.

One first striking information is that the number of total NaN if far higher than the number of natural NaN and that therefore these so-called “equivalent NaN” must be well understood. They could either be considered as true NaN and replaced by an appropriate imputing method or considered as a category of their own and left as is.

Focusing on the information for the total NaN, one sees that their distribution does not change much between the datasets. Although, **based on a 65% threshold**, the following features would be removed based on the population dataset but not based on the customers and mailout training datasets :

* EXTSEL992
* AGER\_TYP
* KK\_KUNDENTYP
* D19\_VERSAND\_OFFLINE\_DATUM

It’s therefore important to consider this during the cleaning steps since the end-goal is to perform binary classification on the mailout dataset. Therefore, one would suggest to base the cleaning steps on the information contained in the mailout training dataset (and therefore customers dataset which is nearly similar).

#### For the features for which no information is available in the Excel files

The above analysis stands for the features that either feature a high ratio of natural NaN or those for which detailed information is available in the Excel file and therefore for which equivalent-NaN could be automatically inferred.

For the other features, a custom function has been written to display interesting information about each feature and to try identifying the ones that again contain a significant amount of natural NaN or equivalent NaN values.

Here is an example of the outputs given by the custom function :

feature D19\_DIGIT\_SERV is categorized as int64 per panda

It means D19 DIGIT SERV in english

it has 8 different values

value 0 has 183539.0 samples and represents 95.77% of data

value 6 has 4233.0 samples and represents 2.21% of data

value 3 has 1582.0 samples and represents 0.83% of data

value 7 has 922.0 samples and represents 0.48% of data

value 5 has 765.0 samples and represents 0.40% of data

value 2 has 434.0 samples and represents 0.23% of data

value 4 has 121.0 samples and represents 0.06% of data

value 1 has 56.0 samples and represents 0.03% of data

it has no natural NaN

The function allows printing the levels for each feature and the number of samples fitting each level. Natural NaN are identified as equal to np.nan. In the above example, one sees that more than 95% of the data has the value 0 which for the features described in the Excel file corresponds to an “unknown value” so this particular feature could be understood as one containing significant number of equivalent NaN and could be choses as candidate for discarding. Remember that associating the value 0 with NaN here is an assumption since nothing guarantees that this is actually true.

This step is relatively time consuming because it requires a manual analysis of the output since no information exist about the levels of the feature, … At the time of writing, the analysis has been performed on the population dataset only. As analyzed before, it should be also performed on the customers and mailout training datasets to reinforce the conclusion and the appropriateness of the analysis. As a remedy, it has at least been verified that deleted columns based on population datasets should also be deleted based on mailout training dataset.

### Amount of missing values : sample analysis

Figure 3, Figure 4 and Figure 5 show the ratio of NaN (natural and equivalent i.e. total) per sample (i.e. per row) respectively for the population, customers and mailout training datasets. One sees that most sample (around 80% for all three datasets) have nearly complete information (bin [0-10]%) for all three datasets of NaN while another significant bin is the [70-80]% meaning that a relatively significant number of samples (around 20% for all three datasets) have between 70 and 80% missing values.

Therefore, choosing **a threshold of 70%** missing value for dropping a row could be seen as appropriate for all three datasets.

### Type of features and number of levels for categorical features

Now that one has been crunching numbers and defined sensible thresholds for dropping non-important features and samples as well as identifying the features that shall be dropped based on manual analysis and assumptions, a last stage consists in analyzing some features more in details and using “subject matter expert” information.

A careful analysis of the features led to the following decisions :

* GEBURTSJAHR and ALTERSKATEGORIE\_GROB provides redundant information; GEBURTSJAHR is kept while ALTERSKATEGORIE will be dropped
* The three CAMEO columns offer identical information. Only CAMEO\_INTL\_2015 will be kept and will be further processed as it provides in fact two information, encoded as double digit i.e. AB. A provides the wealth and B provides the family situation
* LP\_LEBENSPHASE\_FEIN and LP\_LEBENSPHASE\_GROB provide similar information, one is the refined classification the other the gross classification. They provide information about life stage & income. This seems already provided by CAMEO columns so they will both be dropped
* LP\_FAMILIE\_FEIN and LP\_FAMILIE\_GROB are again gross and detailed classification. They provide the family situation. So this is similar to CAMEO\_INTL\_2015 after processing. Both will be dropped
* LP\_STATUS\_GROB and LP\_STATUS\_FEIN are again gross and detailed classification. They provide the income of the person. This information could be deemed similar to CAMEO info but these columns provide more direct information so LP\_STATUS\_GROB will be kept.
* CAMEO\_INTL\_2015 and PRAEGENDE\_JUGENDJAHRE require specific processing to extract relevant information. Specifically, CAMEO\_INTL\_2015 will be split into two features and PRAEGENDE\_JUGENDJAHRE will be transformed into a binary feature to identify whether the person is mainstream or avantgarde i.e. diminishing the amount of categories and information because the rest of the information contained in PRAEGENDE\_JUGENDJAHRE seems similar to other already available features

### Imbalance of mailout training dataset

The mailout training dataset is highly imbalanced as shown by Figure 1. i.e. the dataset features many more non-customers than customers. This is something to keep in mind during the training process and that should be handled specifically.



Figure 1

With the analysis on samples performed in §1.1.1.3, it is interesting to verify if part of the non-customers in the mailout training dataset could be removed based on a ratio of NaN per sample criterion e.g. if a sample contains more than 60% NaN then it gets removed. This could alleviate part of the imbalance in the training dataset.

However, performing this analysis, it turns out that with a threshold of 60% (the one coming from the analysis performed in §1.1.1.3 for the mailout training dataset), 93 samples out of the 532 customers samples would get removed as well. This means one would lose around 16% of data for the customers within this dataset while the information is already scarce.

Based on this, the conclusion is that one should not remove samples

### Conclusion

This paragraph summarizes the cleaning and processing steps that are undertaken for all three datasets i.e. population, customers and mailout datasets. These steps are based on the analyses performed in §1.1.1 to §1.1.4.

First, the following features will be dropped because detailed analysis (on population dataset) showed they contain a significant number of NaN or assumed[[2]](#footnote-2) equivalent-NaN values :

"ALTER\_KIND2", "ALTER\_KIND3", "ALTER\_KIND4", "ALTER\_KIND1", "D19\_DIGIT\_SERV" , "D19\_BANKEN\_LOKAL", "D19\_VERSI\_OFFLINE\_DATUM", "D19\_BANKEN\_REST", "D19\_VERSI\_ONLINE\_DATUM", "D19\_GARTEN" , "D19\_TELKO\_ANZ\_12", "D19\_BANKEN\_ANZ\_24", "D19\_ENERGIE", "D19\_VERSI\_ANZ\_12", "D19\_BANKEN\_ANZ\_12", "D19\_BANKEN\_GROSS", "D19\_BIO\_OEKO", "D19\_NAHRUNGSERGAENZUNG" , "D19\_TELKO\_ANZ\_24", "D19\_TELKO\_ONLINE\_QUOTE\_12", "D19\_SAMMELARTIKEL", "D19\_KOSMETIK", "D19\_DROGERIEARTIKEL", "D19\_WEIN\_FEINKOST", "D19\_VERSAND\_REST", "D19\_TELKO\_MOBILE", "D19\_TELKO\_REST", "D19\_VERSI\_ANZ\_24", "D19\_VERSICHERUNGEN", "D19\_VERSICHERUNGEN", "D19\_VERSI\_DATUM", "D19\_LEBENSMITTEL", "D19\_SCHUHE" , "D19\_VERSI\_ONLINE\_QUOTE\_12", "D19\_KINDERARTIKEL", "D19\_HAUS\_DEKO", "D19\_BANKEN\_DIREKT", "D19\_BILDUNG", "D19\_RATGEBER", "D19\_HANDWERK", "D19\_FREIZEIT", "ANZ\_KINDER", "D19\_LOTTO", "ALTERSKATEGORIE\_FEIN", "EINGEZOGENAM\_HH\_JAHR", "EINGEFUEGT\_AM"

Then, the following features will be dropped because they are either judged as redundant information with other available features or because they feature a very high number of levels which would be problematic during the process of one-hot-encoding for categorical variables :

'CAMEO\_DEUG\_2015', 'CAMEO\_DEU\_2015', 'LP\_LEBENSPHASE\_FEIN', 'LP\_LEBENSPHASE\_GROB', 'LP\_FAMILIE\_FEIN', 'LP\_FAMILIE\_GROB', 'LP\_STATUS\_FEIN'

This brings the number of features from 366 down to 314.

Next, a specific processing of the PRAEGENDE\_JUGENDJAHRE and CAMEO\_INTL\_2015 columns is performed as identified in §1.1.3. CAMEO\_INTL\_2015 will be split into two columns while PRAEGENDE\_JUGENDJAHRE will be reduced to a binary categorical. Also, during data exploration, it was spotted that CAMEO\_INTL\_2015 contains undesirable data i.e. ‘X’ or ‘XX’. These will be replaced by NaN. This brings the number of features to 315.

The next step consists in replacing the equivalent NaN identified in §1.1.1.1 thanks to the information available in the Excel file. First, since some features can have two different values reprensting an equivalent NaN, a custom function performs replacements so that the dataset becomes consistent i.e. one value representing an equivalent NaN per feature. Only 88 replacements are performed across the entire mailout dataset for that step. Then, the equivalent NaN are actually replaced by true NaN values (np.nan).

Eventually, the following columns are dropped based on a NaN threshold of 65% (§1.1.1.1) per feature :

D19\_BANKEN\_DATUM, D19\_BANKEN\_OFFLINE\_DATUM, D19\_BANKEN\_ONLINE\_DATUM, D19\_TELKO\_DATUM, D19\_TELKO\_OFFLINE\_DATUM, D19\_TELKO\_ONLINE\_DATUM, TITEL\_KZ

These columns correspond to the ones identified in Figure 3. This brings the number of features from 315 down to 308.

It’s very important to clean the customers and population dataset in the same manner so that trained algorithms are consistent across the datasets and consider the same type/amount of information.

## Impute, scale, transform

The next step in the data treatment is to impute missing values using appropriate methods, to scale all features so that learning algorithms are not influenced by the absolute value of the features (i.e. an amount of money of 40 000$ should not weight more in the model than the age of buyer which would be at least a 3 digits number) and eventually to transform categorical values not having an ordinal order into one-hot-encoded values .

### Determining the data type

The data type i.e. numerical vs categorical is determined based on a manual analysis of the features for which information is available in the Excel files.

During the analysis, one classified data according to the following categories :

* Binary
* Categorical
* Ordinal
* Numeric

Binary and categorical will be classified as simply categorical and receive the same data treatment. Similarly, Ordinal and Numeric data will receive also the same data treatment. Ordinal features are categorical features having a numerical order i.e. a feature indicating that one is possessing 1, 2 or 3 houses has a numerical order while a feature indicating your political preference (socialist, liberal, …) has no intrinsic numerical order. Such distinction is important because categorical features will be one-hot encoded will others will simply be scaled.

This analysis identified the following columns, still being part of the dataset, as being categorical :

'AGER\_TYP',

'ANREDE\_KZ',

'CJT\_GESAMTTYP',

'D19\_KONSUMTYP',

'FINANZTYP',

'GEBAEUDETYP',

'GEBAEUDETYP\_RASTER',

'GFK\_URLAUBERTYP',

'GREEN\_AVANTGARDE',

'HEALTH\_TYP',

'KBA05\_MAXHERST',

'KBA05\_MAXVORB',

'KBA05\_MODTEMP',

'KBA05\_SEG6',

'LP\_STATUS\_GROB',

'NATIONALITAET\_KZ',

'OST\_WEST\_KZ',

'PRAEGENDE\_JUGENDJAHRE',

'REGIOTYP',

'RETOURTYP\_BK\_S',

'SHOPPER\_TYP',

'VERS\_TYP',

'WOHNLAGE',

'ZABEOTYP'

One must not forget that the Excel files are not complete and some features present in the datasets are not present in the Excel files. Therefore, to identify the additional categorical columns, one uses the dtype inferred automatically by pandas upon loading the dataset. If the dtype equals object then the feature is classified as categorical. The following features are added tot he categorical list through this step :

'D19\_LETZTER\_KAUF\_BRANCHE', CAMEO1', 'CAMEO2'

Note that CAMEO1' and CAMEO2 are the split of the CAMEO\_DEUINTL\_2015which was also flagged during the analysis of the Excel file so this is consitent. This is due to the modification of that feature during the processing step (§1.1.5).

In conclusion, 27 features are categorical and 280 are numerical.

### Applying the appropriate imputing, scaling and transform

For categorical variables, missing values will be imputed through scikit-learn ‘most frequent’ imputer. Also, they will be one-hot-encoded again using scikit-learn one-hot-encoder.

The numerical variables will see their missing values imputed through the mean imputing method of scikit-learn and will be scaled using MinMaxScaler from scikit-learn also.

Following this step, the dataset is expanded from 307 features to 438 features due to one-hot encoding.

# Algorithms

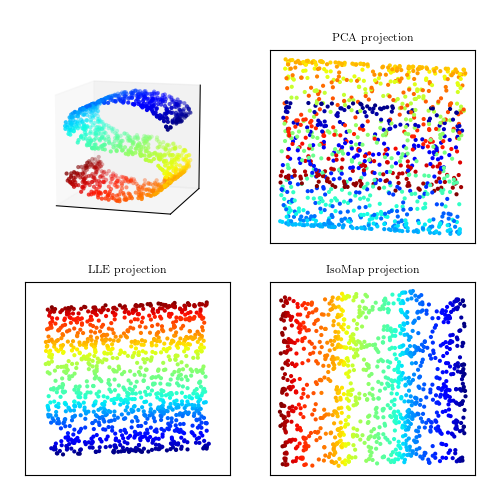
## Generalities

The following algorithms and techniques will be used based on the objectives of the study and the EDA performed:

* Common to clustering and classification :
  + Principal Component Analysis to reduce the features space
* Specific to clustering
  + Use K-Means algorithm to identify clusters from general population and analyze where customers fit into those clusters
* Specific to classification
  + Use XGBoost to classify whether or not one person is likely to become a customer
  + Hyperparameter tuning will be performed using AWS Sagemaker capabilities

For reducing the features space, PCA will be used because this is a very common technique across the data science space and a simple technique. It’s a good approach to quickly reduce the amount of features in order to perform the first studies on a dataset and use it subsequently for specific models e.g. clustering or classification.

It’s important to note though that PCA is a linear technique i.e. it is incapable of capturing non-linear relationships between data. A textbook illustration of that is the S-shape data distribution which shows how non-linear technique are superior to PCA for such datasets :



One sees that non-linear (bottom row) methods preserves data separation (mixing of colors) when projecting data onto the reduced features space contrarily to PCA (top right).

Similarly to PCA, K-means will be used to find clusters among the population and the customers dataset as it is a well-known and widely used technique across the data science space. It also matches the requirement that one has to use an unsupervised clustering algorithm. Indeed, the customers and population datasets have no samples in common i.e. one does not know in the population dataset which samples are customers and which ones are not. And to be complete, K-means is also the only clustering algorithm that the author has experience with.

Eventually, for the classification task, XGBOOST will be used. XGBOOST has demonstrated solid performances in many tasks in the recent years and also during Kaggle competitions. It’s also an algorithm that is ready to use within the AWS Sagemaker platform which reinforces this choice as hyperparameters tuning is therefore easier. One has to take care with XGBOOST since it cannot work with categorical data that are not one-hot-encoded and thereby forcing an increase of the feature space. To solve this problem, an alternative would be to use LightBGM for example but the author has no experience with such algorithm and it’s not readily available in AWS Sagemaker. Another thing to be careful with XGBOOST is that it can overfit the training dataset if the trees are constructed too deep.

## Metrics

The area under curve (AUC) metric will be used to determine the performance of the classification task while the objective metric will be a binary logistic loss given the task at hand.

For the clustering and PCA, no specific performance metric is defined.

Indeed, PCA simply makes a projection of the features space onto a reduced feature space. One can use the explained variance into the feature space to gauge the right number of components to be used and to balance the feature space size against the desired retention of information.

For the clustering, one needs to choose the right number of clusters, right being defined as the number that allows correctly clustering customers against non-clusters. The elbow method could be used but since information about the right number of clusters is already available from work already performed on this project, one simply chooses 5 as the number of clusters [1].

## Benchmark

As the AUC metric will be used for the classification task, one knows that a lower bound for the performance is 0.5. Indeed, achieving an AUC of 0.5 means your model does not fare better than random guessing.

Now, based on Kaggle’s leaderboard [2] for this project, a target should be to reach 0.8.

# Methodology

## Data pre-processing

Based on the EDA performed, cleaning steps have been defined. To perform these cleaning steps in an identical manner on each dataset, a custom cleaning & processing function has been defined in utils/clean.py using functions defined in utils/helper.py.

At first, a custom scikit-learn transformer was used and is still present in utils/ custom\_transformers.py but in the end this way of working proved to be not flexible enough and did not provide many benefits compared to a simple function for the present use case.

## Implementation

### PCA

Once the cleaning and preprocessing steps (including imputing missing values, scaling and one-hot encoding) are completed, the feature space reduction PCA technique has been applied using AWS PCA container and storing the model on AWS S3 for future usage. This allowed reducing the features from 300+ to a chosen number by specifying this number as the number of components desired out of the PCA algorithm. As a first step, 100 components has been chose based on the explained variance metric (see §1.4.

### Clustering

With this reduced feature space, clustering could be applied using K-MEANS container again from AWS. A number of 10 cluster was chosen as a first step. As for PCA, the K-MEANS model was stored on S3 for future usage. For illustration, Figure 2 shows how clustering can be used to identify which clusters of population are more susceptible to hold potential customers. Here, clusters 2 and 3 appear to be promising.

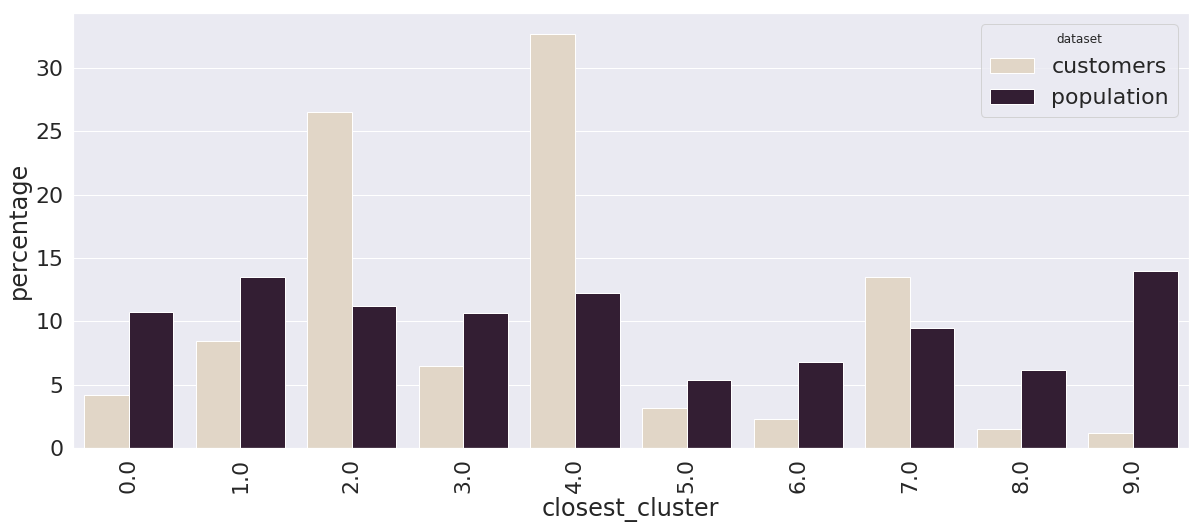


Figure 2 : Clustering using k-means algorithm and 10 clusters

With clusters available, one can then look at the most prominent features for each cluster and identify which features are significantly differencing customers vs non-customers. For illustrative purposes, Figure 3 shows the top 1 component of cluster 2 as well as the features composing it.

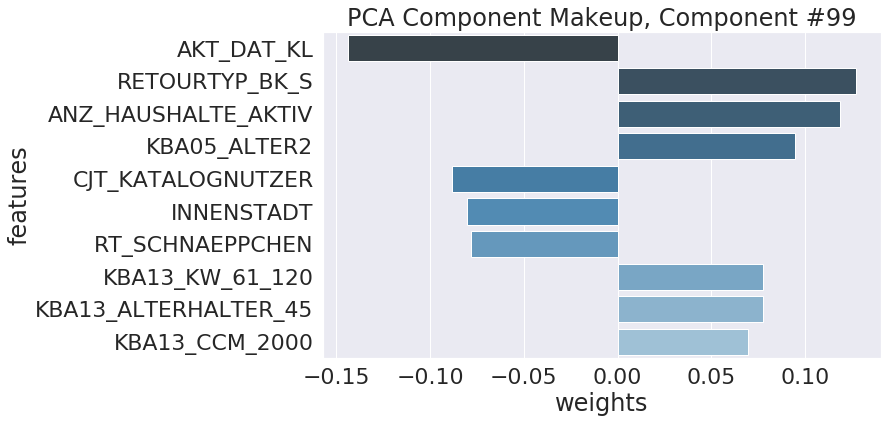


Figure 3 : Top 1 component for cluster 2

### Classification

With clustering part done, one then moved to the binary classification task.

For performing the classification task, a XGBoost model was trained on the complete mailout training dataset after cleaning & transforming steps have been performed (see §4 for details) to predict whether or not a person is likely to become a customer. It’s important to note that PCA results were not used i.e. the complete set of features (~400) was fed to the XGBOOST model.

As already identified in §1.4, the loss function was set to binary logistic and the metric to Area Under Curve (AUC). The dataset was split between a training and a validation set representing 30% of the data and using stratified split i.e. keeping the ratio between customers and non-customers equal within the training and the validation set. This is important given the imbalance (see §1.1.4) of the dataset.

A hyperparameter tuning job was launched using AWS SageMaker capabilities. The following training parameters were retained as giving the best performance :

|  |  |
| --- | --- |
| Parameter | Value |
| Num\_round | 200 |
| alpha | 18 |
| colsample\_bytree | 0.55 |
| colsample\_bylevel | 0.23 |
| colsample\_bynode | 0.67 |
| eta | 0.32 |
| lambda | 16 |
| Max\_depth | 3 |
| min\_child\_weight | 2.75 |
| Subsample | 0.81 |
| objective | Binary:logistic |
| Eval\_metric | auc |

Figure 4 : Final training parameters

Once the XGBoost model was trained, the test set was evaluated through it and sent to Kaggle.

## Results

For the clustering part, applying the K-MEANS algorithm on both the customers and general population datasets allowed identifying that customers are more represented in two particular clusters and analyzing the features that stand out in those clusters.

For the classification task, a score of 0.79 was obtained meaning that the model clearly outperform random guesses (0.5) and nearly achieves the set benchmark of 0.80.

## Refinements

Cleaning & transformation refinements have already been embedded in the custom cleaning function as more and more insights were gained about the data through exploration and first modeling. It’s unlikely that major improvement can be made regarding data cleaning. Maybe removing columns based on correlation or Chi-squared tests could help as well as using standardization rather than MinMax scaling.

Regarding the clustering, maybe improvements could be achieved by applying the elbow method and choosing a better number of clusters. Indeed, as identified in §1.4, this method allows defining that 5 is a good number although the present analysis uses 10.

Regarding the classification task, the following could improve the performance:

* Use the reduced feature space from PCA instead of the complete space
* Include the cluster number into the set of features
* Perform oversampling of positive examples given the dataset imbalance
* Use an embedding method to better translate categorical features compared to one-hot encoding;
* Using other imputing methods for missing values than “frequent” or “mean” e.g. using nearest neighbors;

# Conclusion

The project allowed working on a true dataset with real-life features such as missing values, missing information, outliers, wrongly imputed values, … This required particular attention during data exploration and cleaning.

Also, the project allowed gaining familiarity with AWS Sagemaker and using advanced features such as hyperparameter tuning.

In the end, the proposed models allow achieving a score of 0.79 thereby nearing the top 100 on Kaggle despite using a rough approach of training the model on the complete feature space. It’s highly likely that performance could be improved using feature space reduction techniques.

# References

1. <https://github.com/maitreytalware/Arvato-Bertelsmann-Customer-Segmentation-Capstone>
2. <https://www.kaggle.com/c/udacity-arvato-identify-customers/leaderboard>

APPENDIX A

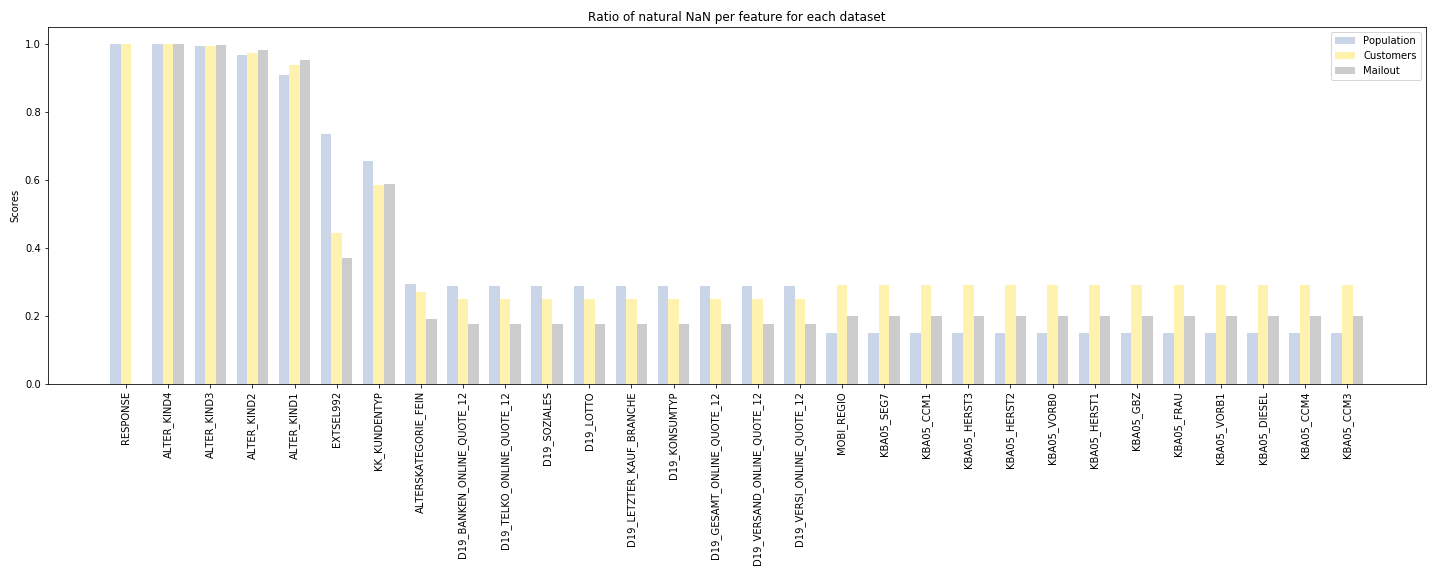


Figure 5 : Ratio of natural NaN per feature for population, customers and mailout training datasets. Results are sorted according to the population dataset data.

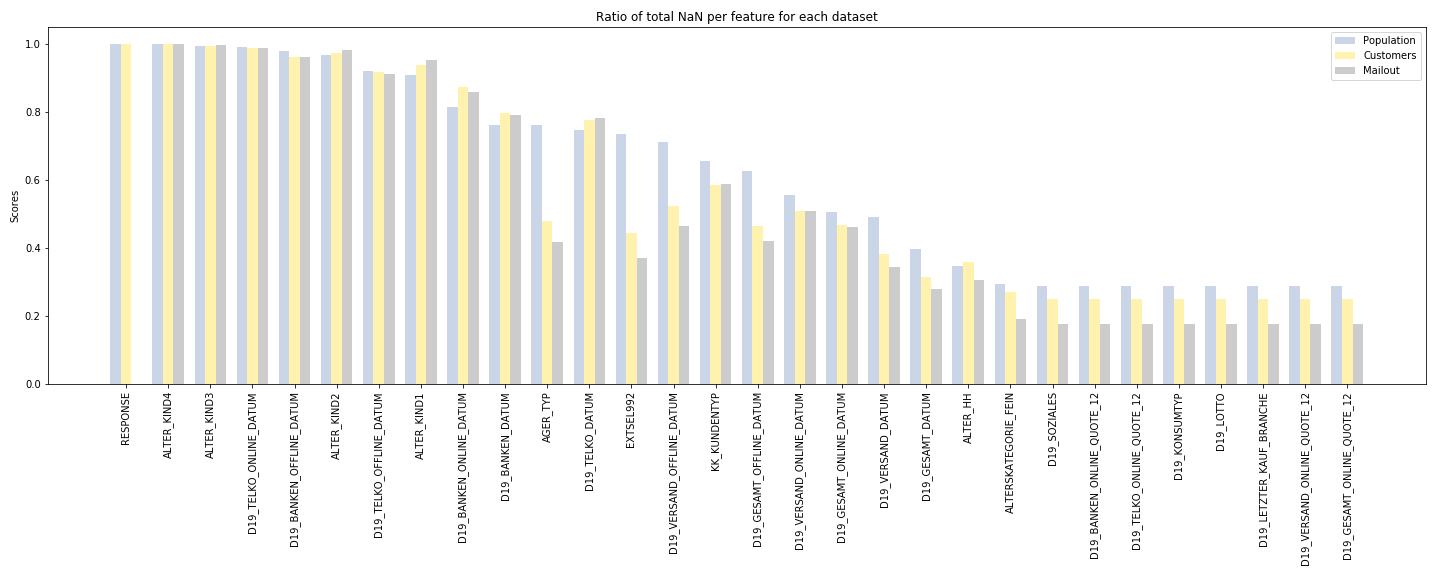


Figure 6 : Ratio of natural NaN per feature for population, customers and mailout training dataset. Results are sorted according to the population dataset data.I

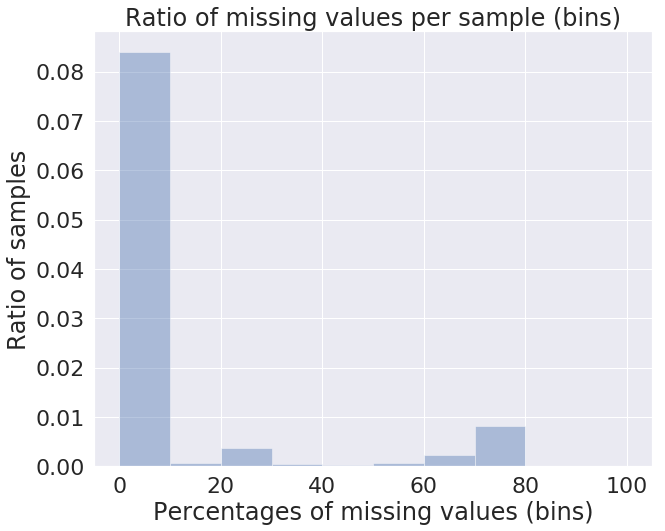


Figure 7 : Populaton dataset

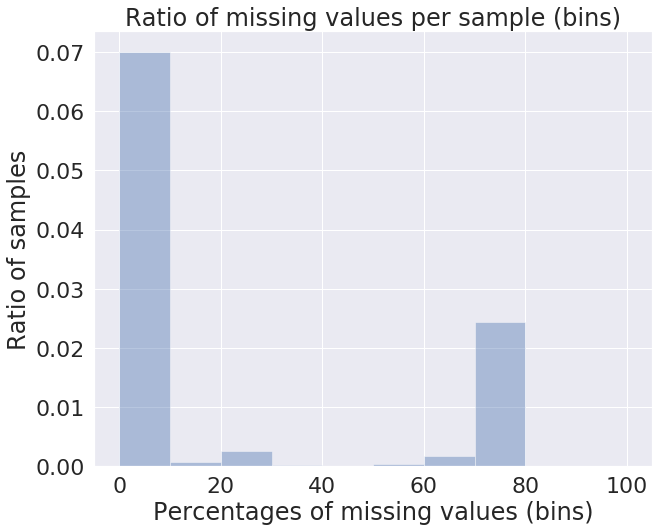


Figure 8 : Customers dataset

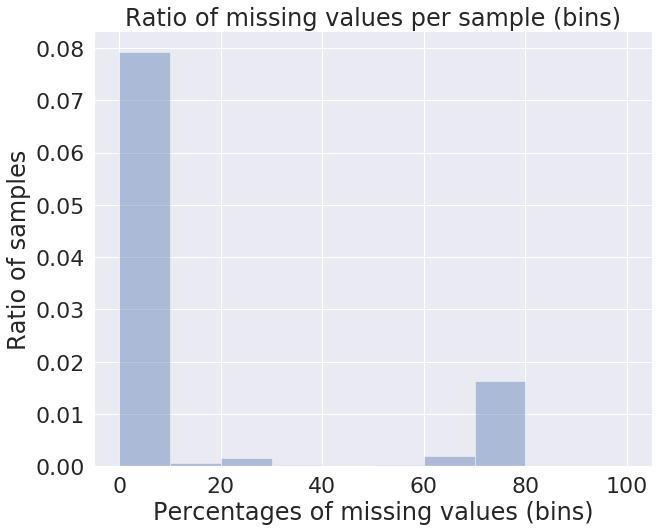


Figure 9 : Mailout dataset

1. In the context of the present report, natural NaN means values that are immeditaly identified as np.nan upon loading in pandas. Some values are equivalent to NaN according to the information provided in the Excel files but require additional processing to be recognized as such. [↑](#footnote-ref-1)
2. Assumed because for those features, one does not have an Excel file detailed the signification of the levels [↑](#footnote-ref-2)