Customer Segmentation Report

Arvato Financial Solutions

1. Project Definition

a. Project Overview

Arvato Financial Solutions is an international financial service provider, offering advanced analytics solutions across the entire customer journey from risk and fraud management to Accounts Receivables Management.

For this project, Arvato provides support to a German mail-order company selling organic products by giving them insights on the segmentation of their customers vis a vis general German population, and by delivering a predictive model in order to improve the chances of customers acquisition through mail campaigns.

The datasets provided for this project are:

- 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 train dataset is used to train and tune the parameters of the predictive model, and the test dataset is the model unseen data on which predictions will be performed and scored on the Kaggle platform.

The other inputs provided are:

- DIAS Information Levels - Attributes 2017.xlsx is a top-level list of attributes and descriptions, organized by informational category.
- DIAS Attributes - Values 2017.xlsx is a detailed mapping of data values for each feature in alphabetical order.

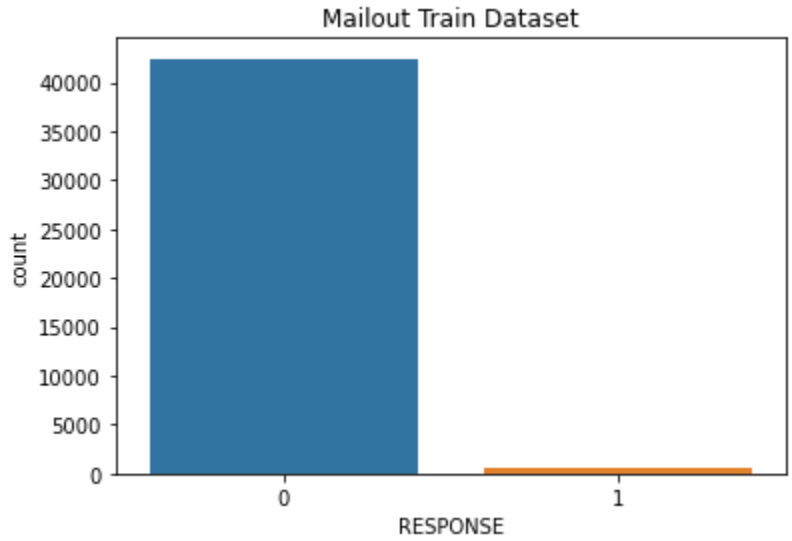
b. Problem Statement

The problem is two parts. First, understanding what are the key attributes of existing customers vis a vis German general population. Are there segments in the German population which are more or less likely to be customers of the mail-order company given their demographic data ? To answer this question an unsupervised clustering analysis of the general population will be performed and applied to the customers of the company.

Second, how to improve success of mail marketing campaigns by targeting the population with the highest chance of becoming a customer ? To answer this question a supervised model will be applied to predict the probability of a person becoming a customer.

c. Metrics

The mailout train dataset is very unbalanced. As can be seen below.



The performance of our model will be assessed using the AUC metric, as accuracy would not be meaningful.

2. Analysis

a. Data Exploration

We first performed a series of checks on the data.

- Range of values for each feature: we wanted to ensure that for each feature listed in the attributes file, our datasets had values within an expected range. We built a function to perform this check. We found 6 categorical variables having values outside of the expected range. As they are categorical, the interpretation of a value which is missing from the dictionary can not be done and we therefore replaced these values by NaN. We found 7 numerical variables which

were outside of ‘typical’ range. As they are numerical, they might be outliers but could still be correct. We replaced the 0 value by NaN for 5 of them as they were supposed to start at 1, and also because one of them was the year of birth, which can not be 0.

- Missing features from dictionary: 94 features are present in the files but missing from the dictionary. For these features the range check mentioned above can not be performed. Looking at the names of these features they are very likely categorical features.

- Missing Data: In the attribute file, some values typically -1 or 9 mean ‘unknown’. We therefore decided to replace by NaN these values. There are a lot of missing values either by rows or columns especially for azdias and customers dataset. For example, customers file has half of the columns with more than 27% missing values and 25% of rows with more than 62% missing.

b. Categorical and Numerical Features

We can use the attribute file to confirm that 7 features are numerical features. For all the other features we consider them as categorical, which when we look at their description seems to make sense.

3. Clustering

a. Choice of the algorithm

The datasets azdias and customers have a lot of missing values. The features are largely categorical and when we look at the attribute file, a relatively large number of features are not ordered.

In order to avoid loss of information by deleting a lot of rows, columns and in order to avoid using a high number of inputting strategies for the missing values which would remain, we decided to use the KPrototype algorithm to perform the clustering.

The following version of the algorithm will be used:

“Huang, Z.: Clustering large data sets with mixed numeric and categorical values, Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conference, Singapore, pp. 21-34, 1997”

It will enable to:

- Create an ‘unknown’ category for all the missing data on categorical features
- Avoid making assumptions whether or not a categorical feature is ordered by keeping it as categorical
- Avoid one hot encoding of categorical features as it would result in way too large datasets
- Take into account the numerical feature as numerical

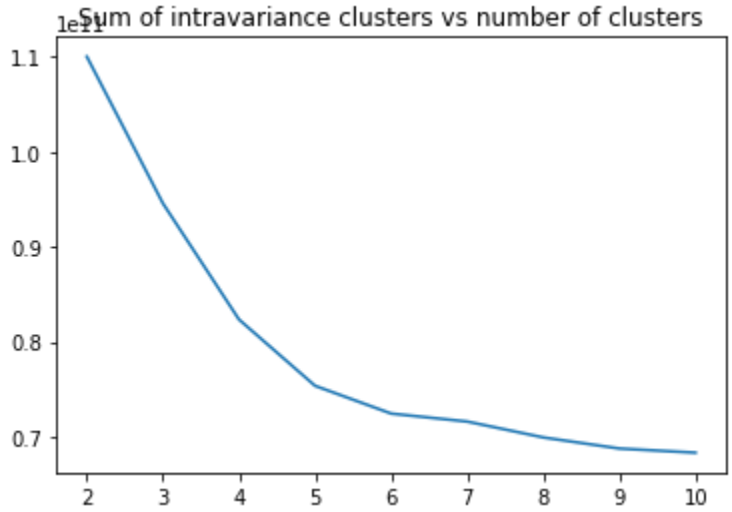
The library used to run the Kprototype algorithm is the following:

https://github.com/nicodv/kmodes

The algorithm is run with 2 to 10 clusters. The number of initialization of the random points has been chosen to 1. It would have been better to increase this number but it was becoming too expensive in terms of computation. By using this algorithm we have avoided to delete a lot of information or to input a lot of data, but the drawback is that the computations take a lot of time.

b. Results

To choose the optimal number of clusters, we visualize the sum of intravariance clusters, depending on the number of clusters chosen.



With the elbow method we chose 7 clusters as it seems to decrease almost linearly after.

We then look at the relative distribution of the general population and customers amongst clusters.

We see 2 clusters as particularly interesting:

- Cluster 3: under-represented for customers at 13% vs 21% for general population
- Cluster 4: over-represented for customers at 27% vs 12% for general population

For the other clusters, the percentage of customers is close to the percentage in the general population so it does not indicate that these clusters characteristics are correlated with being a customer or not.

We want to understand the specificities of clusters 3 and 4. We therefore built a function which retrieved for categorical features present in the attribute file, the features of these clusters if they are present less than a certain threshold in the general population and if the feature value is not ‘unknown’ for the cluster.

The reason is that we have more than 300 categorical features, and we want to focus on what makes the profile of these clusters standout vs the general population. Also even if an 'unknown' value is present in proportion less than the requested threshold in the general population, it won't tell us much about this cluster.

The function also compares the numerical features with the distribution in the general population

Fixing the threshold at 25%, we have the following picture:

- Cluster 3:
    - Low income earner without vacation habit
    - Working class
    - Live in area with low number of cars, especially for more expensive ones
    - Live in area with high share of multifamily buildings
    - On the numerical attributes a couple of features are quite far away from the median:
        - Live mostly in 10 households buildings, meaning large buildings. This above the 75% percentile.
        - Live in a geographical area with a small number of cars below 25% percentile.

- Cluster 4:
    - High income but not materialistic.
    - Nature fan
    - Dreamily person
    - Social and cultural minded
    - On the numerical attributes all the features are very close to the median for the general population.

To be noted that the tentative to interpret these clusters would be much easier with a deeper understanding of the meaning of the various features, beyond what is provided in the attribute file.

4. Unsupervised learning

a. Models and metric

As explained earlier, the train dataset is very unbalanced. The metric to validate the model is therefore the AUC.

In terms of model choice, because we have a very large number of categorical features we chose the Catboost model. This is a gradient boosting model, which has the advantages of:

- Being fast to trained compared to other libraries such as XGboost, light GBM
- Handle categorical variables without having to encode them

4 Other models were also tried: logistic regression, random forest, K nearest neighbors and a soft voting classifier combining catboost, logistic regression and random forest. As shown in the results summary, the catboost model achieved the best performance and hence we will do a deeper dive on this model.

b. Catboost Classifier

i. First implementation

We want a baseline comparison, we therefore use the default catboost parameters with the following exceptions:

- logging_level='Silent': we don't want the output of the by default 1000 iterations
- eval_metric='AUC': this enables to select the best model for the AUC on the validation set (which has been seet at 33% of the mailout train dataset)
- early_stopping_round=200 : if there is no improvement of the AUC on the validation set after 200 iterations, there are no further iterations. This prevents overfitting and also reduce computation time

We obtained an AUC score of 0.778 on the validation set, for our baseline model. We achieved 0.79226 on Kaggle for the predictions on the mailout_test dataset.

We then wanted to understand what features are the most important. Catboost library proposes several methodologies to assess the importance of features, but after going through the documentation I struggled to understand the computation of the various measures.

We therefore implemented feature permutation to measure the importance of each feature. For each feature the column of the feature is randomly shuffled 5 times, and the measure of the feature importance is computed as the AUC of the baseline model minus the average AUC obtained after shuffling the feature 5 times. The higher the score, the more the model performance deteriorated after shuffling the feature and hence the more important the feature is.
The result is the below:

```
In [114]: importance[importance!=0]

Out[114]: D19_SOZIALES                  0.242544
          EINGEFUEGT_AM                 0.041555
          ANZ_STATISTISCHE_HAUSHALTE    0.031231
          EXTSEL992                     0.001917
          CAMEO_DEU_2015                0.000857
          VERDICHTUNGSRAUM             -0.000320
          dtype: float64
```

ii. Features engineering

We have no information on these features in the dictionary at the exception of CAMEO_DEU_2015.

- D19_SOZIALES: D19 variables refer to transaction activity in some group of products. Here it must be the group SOZIALES. The value of this feature is treated as categorical and goes from 0 to 5 or unknown. It is mostly unknown information for cluster 4 which was the over-represented one for customers.
- EINGEFUEGT_AM: Lots of different time stamps are present. We tried to engineer this feature by just extracting the month and the year, but it deteriorated the performance from 0.778 to 0.776.
- ANZ_STATISTICHE_HAUSHALTE: No information but looking at the number of unique values at 174 it rather looks like a numerical feature with integers rather than a categorical variable. We

rerun our baseline model but treating this feature as numerical. We obtain an improvement from 0.778 to 0.789.

iii. Features selection

Starting from our improved model with ANZ_STATISTICHE_HAUSHALTE as a numerical feature, we computed again the features importance using as previously feature permutation.
We obtain the following result:

	Unnamed: 0	0
0	D19_SOZIALES	0.269052
1	EINGEFUEGT_AM	0.054603
2	EXTSEL992	0.012945
3	VERDICHTUNGSRAUM	0.004648
4	CAMEO_INTL_2015	0.001438
5	CAMEO_DEU_2015	0.000387

We reran the improved model but we only select the above feature. We achieved a better AUC at 0.7925.

When we made predictions on the mailout_test dataset, we achieved an AUC of 0.79223 on Kaggle vs. 0.79226 for the baseline. It is a very slight decrease, but the number of features being significantly reduced, it enabled us to perform hyperparameters tuning from a computational standpoint.

iv. Hyperparameters tuning with reduced number of features

We will try to tune the following parameters in the catboost model:

- Iterations: Number of trees to fit. We specify an early stopping round at 300 to prevent over-fitting and reduce computation time
- Learning_rate: Controls the gradient step coefficient
- Depth: Depth of the trees which are fitted
- L2_leaf_reg: Coefficient at the L2 regularization term of the cost function.
- Class_weights:The values are used as multipliers for the object weights. This parameter could help us as the dataset is extremely imbalanced.

We try two different approaches using the Hyperopt library:

1. Random search

After 100 iterations using a 3 folds stratified cross validation, the best model found has the following parameters:

- class_weights: None,
- depth: 9.0
- iterations: 500
- l2_leaf_reg: 5.0

- learning_rate: 0.03677614050093655

The score achieved on the mailout_test dataset on Kaggle is 0.78435 less good than baseline.

2. Tree Parzen Estimator

After 100 iterations using a 3 folds stratified cross validation, the best model found has the following parameters:

- class_weights: None,
- depth: 8.0,
- iterations: 500,
- l2_leaf_reg: 7.0
- learning_rate: 0.05878812536974758

The score achieved on the mailout_test dataset on Kaggle is 0.7843 less good than baseline.

v. Hyperparameters tuning with all features

The results above, made us think that we probably needed more features to have a better performance. The measure of features importance that we previously performed has some drawbacks:

- Features permutation measure can be misleading if some features are correlated (because we only permute one feature at a time while the correlated feature(s) would be kept unchanged)
- The baseline model was trained with default parameters

Because it took a lot of computation power, we decided to only try hyperparameters tuning for 50 iterations, solely using a random search approach. The reason is that for the Tree Parzen Estimator method to be efficient we likely need a much higher number of iterations, especially as there are some dependencies between some hyperparameters such as the number of trees, depth and all the regularization parameters. The TPE method implemented in Hyperopt does not account for this dependency. If a large number of iterations can be performed, I would assume it can still outperform random search, but it would take too much time for us. Also to speed-up the process, we reduced the number of early stopping rounds from 300 to 100.

After performing a random 3 folds stratified cross validation we retrieve the following best model:

- class_weights: (1, 100)
- depth: 7.0
- iterations: 500
- l2_leaf_reg: 5.0
- learning_rate: 0.01291491309200495

What is interesting is that similarly to the previous results with a reduced number of features, the regularization parameters are increased compared to the default settings of Catboost (l2_leaf_reg default is 3, and the default learning rate is around 0.05) and the depth of the trees is increased (default is 6).

The difference here is that the minority class takes a weight of 100, to compensate for the very imbalance structure of the mailout_train dataset.

The score achieved on the mailout_test dataset on Kaggle is 0.80612. This was our best submission and ranked 30 out 361 at the time of writing.

vi. Features importance on the final model

We performed feature permutation to understand what are the important features on the final model.

	Unnamed: 0	0
0	D19_SOZIALES	0.198505
1	D19_KONSUMTYP_MAX	0.006600
2	KBA05_KRSHERST3	0.006152
3	KBA13_HERST_EUROPA	0.004832
4	RT_SCHNAEPPCHEN	0.004449
...	...	...
336	D19_VERSAND_DATUM	0.000045
337	MIN_GEBAEUDEJAHR	0.000043
338	MOBI_RASTER	0.000039
339	D19_BANKEN_ANZ_12	0.000005
340	ALTER_KIND4	0.000003

D19_SOZIALES remains by far the most important feature. As opposed to previous models, almost all features seem to have a contribution, but these are very small contributions.

	Feature	Importance
0	D19_SOZIALES	0.198505
1	D19_KONSUMTYP_MAX	0.006600
2	KBA05_KRSHERST3	0.006152
3	KBA13_HERST_EUROPA	0.004832
4	RT_SCHNAEPPCHEN	0.004449
5	VHN	0.004215
6	WOHNLAGE	0.004198
7	SEMIO_RAT	0.003953
8	KBA13_KW_60	0.003922
9	KBA13_OPEL	0.003739
10	KBA13_BJ_2008	0.003605

Below the top 10. We don't have information on all the meaning of features. But quite a few features are related to information about cars (all the KBA13 and KBA05), one feature on type of neighborhood, one on consumption and one indicating if the person has a rational mind.

This seems more related with the description of under and over represented clusters that we obtained in the first part of the project, compared to the other models.

c. Logistic Regression

Before fitting a logistic regression, we had to handle all our categorical variables. We encoded each categorical variable with dummy indicators. The feature 'EINGEFUEGT_AM' had 1600 different values, so for this feature we replaced the timestamp by the year and the month.
Once we performed this encoding of categorical variables we obtained a mailout_train file with 2725 features.
The missing numerical features were inputted with the median and were scaled between 0 and 1.

A first logistic regression was fit with scikit learn library. Default parameters were used and after a 3 folds stratified cross validation the AUC obtained was 0.6496

We then performed a tuning of the hyperparameters with the random search of Hyperopt library. We tried 10 random searches (we did not perform more as it was very slow), changing:

- The penalty: L2 penalty (default), L1 penalty, elastic net penalty or no penalty
- The class weights: none (default) or balanced
- C parameter: between 0 and 1 (default) controlling the strength of the regularization (stronger for <1)

The best model we found after this random search with 3 folds stratified cross validation had L1 penalty, no class weight, a small C of 0.19 and scored an AUC of 0.7538 on the test set and 0.7659 on the Kaggle test set.

d. Random Forest

The same treatment as described above was applied for the mailout_train dataset, at the exception that numerical features were not scaled.

A first random forest was fit with scikit learn library. Default parameters were used and after a 3 folds stratified cross validation the AUC obtained was 0.6187.

We then performed a tuning of the hyperparameters with the random search and Tree Parzen Estimator of Hyperopt library. We tried 100 searches for each approach with a 3 folds stratified cross validation. We changed the following parameters:

- N_estimators: number of decision trees fitted (default 100)
- Criterion: gini (default) or entropy
- Max_depth: max depth of individual decision trees (default none)
- Min_samples_split: the minimum number of samples required to split an internal node (default 2)
- Class_weight: none by default

We obtained the best model with the Tree Parzen Estimator, with the following parameters:

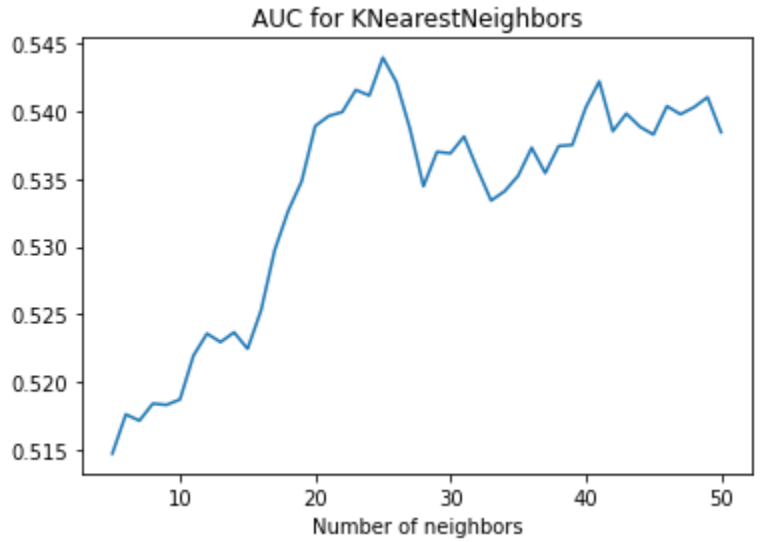
- N_estimators: 300
- Criterion: gini
- Max_depth: 9
- Min_samples_split: 14.7% ie: need a minimum of 14.7% of the samples to split an internal node
- Class_weight: the target was given a weight of 50

This model scored an AUC of 0.7590 on the test set and 0.7856 on the Kaggle test set.

e. KNearest Neighbors

The same treatment as described above was applied for the mailout_train dataset, at the exception that numerical features were not scaled. We used the Euclidian norm to measure the distance between each points. We used the Facebook Artificial Intelligence Similarity Search library which allows very quick similarity searches. It was very useful especially to tune the number of neighbors, as we quickly encountered huge prediction time with the scikit learn library.

This approach did not give good results, with the best AUC after a 3 folds stratified cross-validation reaching less than 0.545 as can be seen below



We did not try to score this model on the Kaggle test set given the poor performance.

f. Soft Voting Classifier

This last model combined the best catboost classifier, the best logistic regression, the best random forest and averaged the predicted probability on the Kaggle test set. It achieved an AUC of 0.8011.
I found this result surprisingly good. It could have been interesting to try a weighted average rather than a simple average. However, it would have required a lot of computation time as each combination of weights would have required to train at least 9 models (the three models for each of the 3 folds and potentially more if we were tuning the parameters of each model for this ensemble), and hence we did not try a weighted average.

g. Summary of results

The best models in their category have been highlighted in yellow in the below table:

Model	Features	Parameters	AUC after 3 folds cross validation	AUC on Kaggle test set
Catboost Classifier	All features	Default	0.7737	0.7923
Catboost Classifier	All features 'ANZ_STATISTISCHE_HAUSHALTE' as numerical	Default	0.7767	NA
Catboost Classifier	All features 'EINGEFUEGT_AM' replaced by year and month	Default	0.7735	NA
Catboost Classifier	Reduced number of features to 6	Default	0.7756	0.7922
Catboost Classifier	Reduced number of features to 6	Tuned after 100 searches with TPE algorithm	0.7767	0.7843
Catboost Classifier	Reduced number of features to 6	Tuned after 100 random searches	0.7756	0.7844
Catboost Classifier	Reduced number of features to 6	Tuned after 100 random searches	0.7756	0.7844
Catboost Classifier	All features 'ANZ_STATISTISCHE_HAUSHALTE' as numerical	Tuned after 50 random searches	0.7814	0.8061
Logistic Regression	All features 'EINGEFUEGT_AM' replaced by year and month 'ANZ_STATISTISCHE_HAUSHALTE' as numerical	Default	0.6496	NA
Logistic Regression	All features 'EINGEFUEGT_AM' replaced by year and month 'ANZ_STATISTISCHE_HAUSHALTE' as numerical	Tuned after 10 random searches	0.7538	0.7659
Random Forest	All features 'EINGEFUEGT_AM' replaced by year and month 'ANZ_STATISTISCHE_HA	Default	0.6187	NA

	USHALTE' as numerical			
Random Forest	All features 'EINGEFUEGT_AM' replaced by year and month 'ANZ_STATISTISCHE_HA USHALTE' as numerical	Tuned after 100 random searches	0.7585	NA
Random Forest	All features 'EINGEFUEGT_AM' replaced by year and month 'ANZ_STATISTISCHE_HA USHALTE' as numerical	Tuned after 100 searches with TPE algorithm	0.7590	0.7856
KNearest Neighbors	All features 'EINGEFUEGT_AM' replaced by year and month 'ANZ_STATISTISCHE_HA USHALTE' as numerical	Euclidian metric From 5 to 50 nearest neighbors tried	Below 0.545	NA
Ensemble Soft Voting Classifier	All features 'ANZ_STATISTISCHE_HA USHALTE' as numerical	Simple average across the best 3 models in yellow	NA	0.8011

5. Conclusion

I found that the 2 biggest difficulties for this project were the size of the datasets (as it made every computation very slow and hence restraining the number of experimentation) and the lack of deeper understanding of the features.

This last point was the most difficult one. I initially thought of implementing some dimension reduction techniques such as PCA or factor analysis, but the difficulty to identify which variables were numerical, categorical, ordinal...as well as the numerous missing data (and hence potential loss of information) made me take a different path.

Lot of improvements could be made:

- The treatment of features: as explained above a deeper understanding would enable to more precisely separate ordinal features which could potentially be treated as numerical from numerical and categorical one.
- The tuning of hyperparameters. It is very likely that by running more iterations and by using the TPE algorithm based on Bayesian optimization a better set of hyperparameters could be found. This is especially true for the Catboost classifier model but also for the soft voting classifier which gave promising results just performing a simple average.