Investigating Customer Segments and Predicting Response Rate in Marketing Campaign

A Customer Segmentation Report for Arvato Financial Solutions

Zhitao Wang

Jan 1 · 14 min read

Image Source: Evolve Digital Agency

Background

This report is part of the Udacity Data Scientist Capstone Project in partnership with Bertelsmann Arvato Analytics to solve a real-life data science task. As Josh Bernhard and Timo Reis have mentioned in Video1, Video2, there are two major parts in this project:

> *Identifying customer segments among the general population.*

The motivation in the first part is to increase customer acquisition efficiency by targeting those who are more likely to become company's customers, rather than reaching out to all in Germany.

Here, I compared customers' dataset of a mail-order sales company in Germany against demographics dataset for the general population. Through applying unsupervised learning techniques, I created customer segmentation, and identified the parts of the population that best describe the core customer base of the company.

> *Building a supervised-learning model to predict customer behavior*

The second part aims to build a machine learning model that predicts which customers are more likely to respond to the campaign, based on what we've learned from the customer segments on a third dataset with similar attributes from targets of a mail order campaign for the company. Each target in the dataset contains two possible outcomes: either response or no response.

Finally, the model was validated by another independent dataset on Kaggle, to see how the improvement on the model can measure up against the Udacity fellow students.

Part 0: Data Preprocessing

There are four dataset associated with this project:

AZDIAS: 891 211 persons (rows) x 366 features (columns).
Customers: 191 652 persons (rows) x 369 features (columns).
Mailout Trainset: 42 982 persons (rows) x 367 (columns).
Mailout Testset: 42 833 persons (rows) x 366 (columns).

In part 1**,** ***AZDIAS*** and ***Customers*** refer to the demographics data for the general population of Germany and customers of a mail-order company respectively. They were used to compare and identified marketing segments**.** In part 2, we trained the supervised-learning model to predict response rate of customers using ***Mailout Trainset***, and was then used to make predictions on ***Mailout Trainset***. The result was submitted to Kaggle for validation. For ***Mailout Trainset,*** it is similar to the rest of datasets in addition to a labeled ***RESPONSE***, with 0 indicating no response, and 1 with response***.***

Part 0.1 Accessing the Missing Values

Although these datasets on hands are not as nasty as raw data, we still need to apply some trivial cleaning steps, such as converting the code in every data entry that indicates “missing” or “unknown” to missing values (“NaN”) that a machine learning model can differentiate and interpret.

```
Number of feature to be dropped out:  66
```

	Feature	NaN Count	NaN Occupancy Rate
7	ALTER_KIND4	890016	99.864792
349	TITEL_KZ	889061	99.757636
6	ALTER_KIND3	885051	99.307691
76	D19_TELKO_ONLINE_DATUM	883018	99.079577
33	D19_BANKEN_LOKAL	874745	98.151300

Table 1 Top Ranked Column-Missing Features

It is necessary to evaluate the missing values in each column since dropping columns with high NaN helps to reduce feature dimension and simplify the model. The table on the left shows an example of features that have been dropped out due to the high percentage of missing value (“NaN Occupancy Rate”). In my case, a threshold has been set at 30% and 66 features were dropped out.

By accessing the missing data row-wise, we also found that the group of individuals with high row-NaN percentage is distinct from the group with low row-NaN percentage, as shown by the distribution over features in the figure below:

as shown by the distribution over features in the figure below:

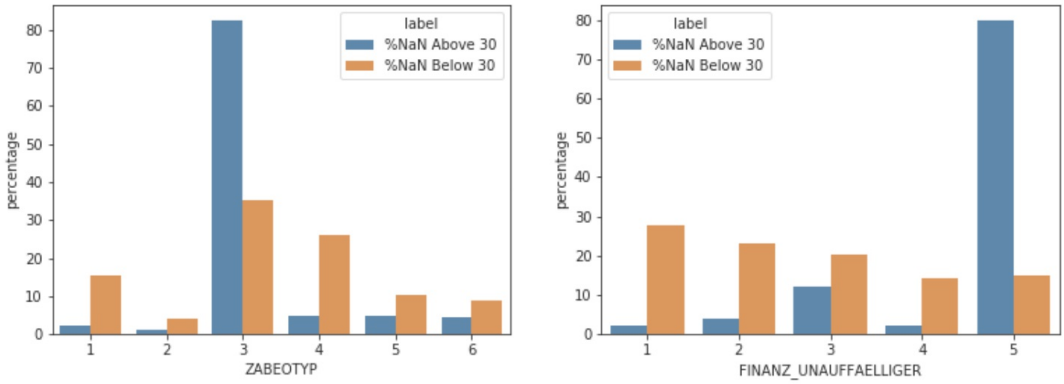


Figure 1. Comparison of High (Blue) and Low (Orange) Row-Missing Groups over ZABEOTYP (related to type of energy consumers) and FINANZE_UNAUFFAELLIGER (related to financial type for individual) Features

The left panel above shows that the high row-NaN group is heavily distributed in the "3" bucket in categorical feature **ZABEOTYP**, compared to relatively smooth distribution of low row-NaN group. Therefore, we should treat the high row-NaN group as a special cluster and apply segmentation only to low row-NaN group.

Part 0.2 Feature Engineering

Since the machine learning techniques to be used will only work on data that is encoded numerically, it is necessary to apply a few encoding changes or additional assumptions to be able to make progress. There are five types of features in the dataset: numerical, interval, ordinal, categorical and mixed types.

- *For numeric and interval data, these features can be kept without changes.*
- *Most of the variables in the dataset are ordinal in nature. While ordinal values may technically be non-linear in spacing, make the simplifying assumption that the ordinal variables can be treated as being interval in nature (that is, also kept without any changes).*
- *Apply one-hot encoding to categorical data.*
- *Decompose "mixed" features into combinations of any other feature types above.*

4.3. CAMEO_INTL_2015

German CAMEO: Wealth / Life Stage Typology, mapped to international code

- -1: unknown
- 11: Wealthy Households - Pre-Family Couples & Singles
- 12: Wealthy Households - Young Couples With Children
- 13: Wealthy Households - Families With School Age Children
- 14: Wealthy Households - Older Families & Mature Couples
- 15: Wealthy Households - Elders In Retirement
- 21: Prosperous Households - Pre-Family Couples & Singles
- 22: Prosperous Households - Young Couples With Children
- 23: Prosperous Households - Families With School Age Children

Figure 2. An Example of Mixed Feature From the Data Dictionary

An example of the “mixed” feature “CAMEO_INTL_2015” can be shown on the left figure. According to the dictionary (Link), it contains combinations of wealth and life stage information. We can break up the two-digit codes by their ‘tens’-place and ‘ones’-place digits into two new ordinal variables (which, for the purposes of this project, is equivalent to just treating them as their raw numeric values).

Overall, there is no standard way to re-engineer the “mixed” types features. In fact, steps to apply data cleaning and feature engineering, especially generating new features hidden from current features, can be very trivial and also time consuming. Remind that after generating these new features, the original features shall be excluded from the dataset.

Part 1: Customer Segmentation Report

Part 1.1 Interpreting PCA component

In the next step, we will apply data transformation on the ***AZDIAS*** dataset, including feature scaling and dimension reduction via principle component analysis (PCA). By reducing the data dimension, it provides convenience for us to: 1. understand the relationship among different features on each principle component; 2. and reduce the volume of data to run process-intensive Kmeans algorithm.

Based on the scree plot analysis, I decided to 15 components, consisting of 30% total variance. The figure below shows an example of the first principal component, in which correlated features are related to financial and social status. The positive weights indicate lower income, and social status. For instance, the dummy variable **LP_STATUS_GROB_1.0** = 1 indicates a positive direction towards lower income earners (as shown by the data dictionary). And a higher score of **HH_EINKOMMEN_SCORE** indicates lower household income. A negative weight indicates an opposite trend: a higher value of **FINANZ_MINIMALIST** indicates lower financial interest (meaning better financial credit).

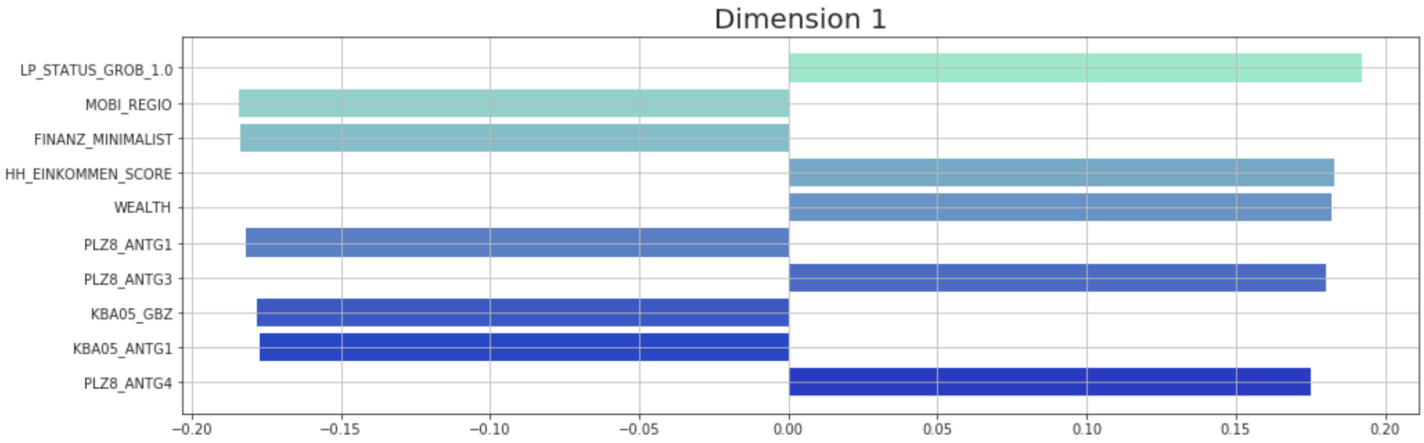


Figure 3. Weights of Feature in Principal Component 1

Part 1.2 Deciding Cluster Number

Following PCA, I applied k-means algorithm to cluster each individual in the transformed data ***AZDIAS***. A total number of 10 clusters has been chosen based on the Figure 4 scree plot on the left panel below, where the x-axis is total cluster and the y-

Figure 4 scree plot on the left panel below, where the x-axis is total cluster and the y-axis the average distance (SSE). The plot on the bottom right panel corresponds to the first derivative of the left scree plot.

In general, SSE decreases with the number of clusters monotonically. At the elbow near 10 clusters, the slope of SSE starts to increase much slower as shown in the right panel.

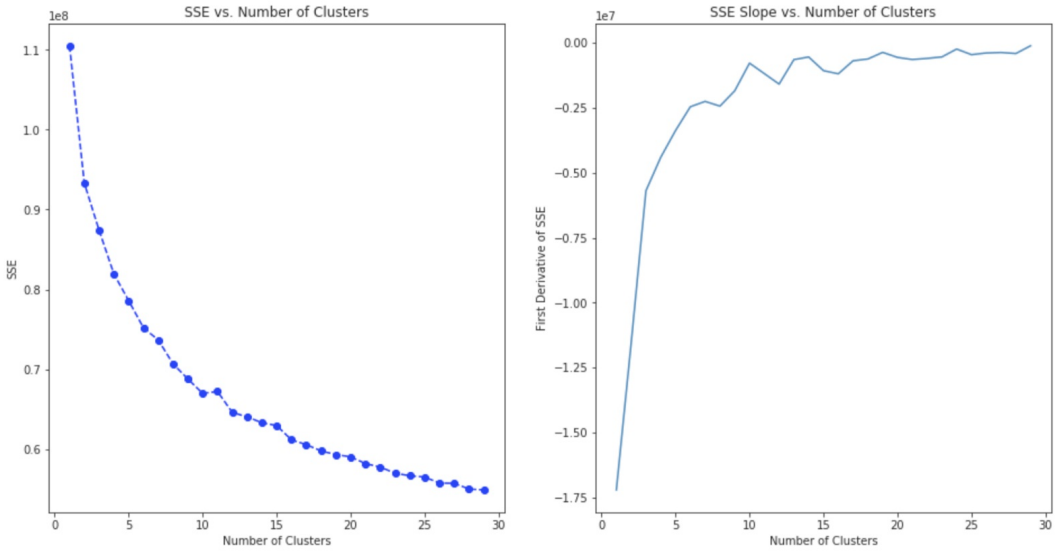


Figure 4. Scree plot of SSE v.s. Cluster Number

Part 1.3 Comparing the Customer Data to Demographics Data

At this point, we have clustered the ***AZDIAS*** data (demographics of the general population of Germany), and can now apply clustering to the ***Customers*** data for a mail-order sales company. Figure 5 compares the two distributions of ***AZDIAS*** and ***Customers*** from Cluster 0 ~10. Note that Cluster 10 refers to the group with high row-missing percentage in the features.

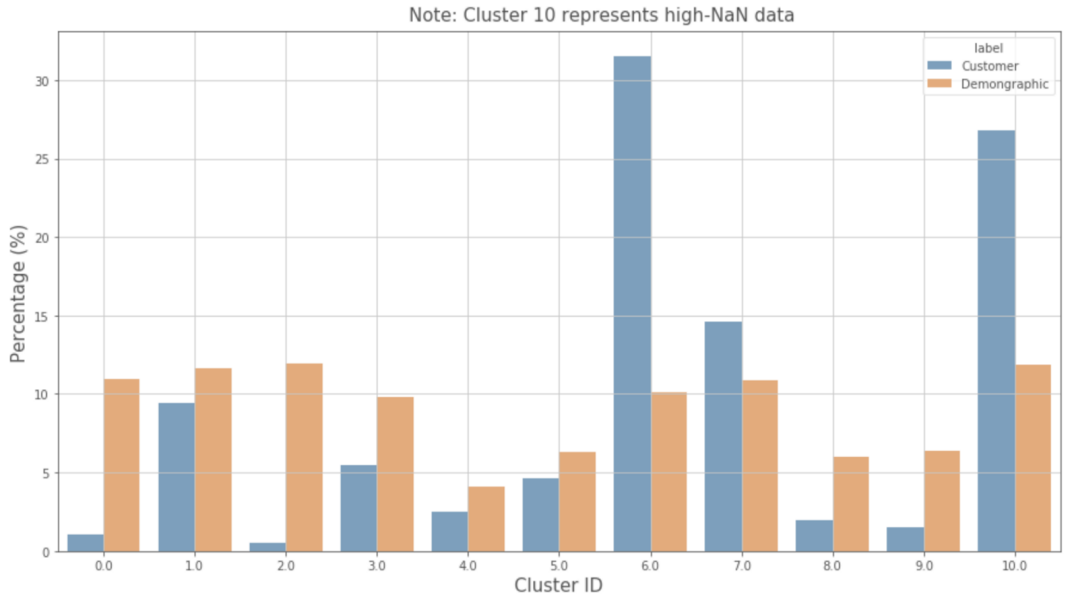


Figure 5. A comparison of normalized distribution between customer and general demographics among all clusters

In principle, if the company's customer base is universal, the cluster assignment proportions should be fairly similar between the two datasets, ***AZDIAS*** and ***Customers***.

In fact, the histogram shows an overrepresentation of Customer in Cluster 6, 7 and 10, suggesting a strong customer base in these clusters. On the other hand, the underrepresentation in Cluster 0 and 2 suggest that individuals in these clusters are less likely to become company's customers.

			Cluster 6 (overrepresenting)	Cluster 2 (underrepresenting)
Income	HH_EINKOMMEN_SCORE	Estimated household net income	2.499271	5.001463
	LP_STATUS_GROB_1.0	low-income earners	-0.005925	0.545940
	LP_STATUS_GROB_2.0	average earners	0.183684	0.320937
	LP_STATUS_GROB_3.0	independents	0.086453	0.002391
	LP_STATUS_GROB_4.0	houseowners	0.133851	0.112071
	LP_STATUS_GROB_5.0	top earners	0.601937	0.018660
	FINANZ_MINIMALIST	low financial interest	4.443390	1.955474
	WEALTH	Wealth Scale	1.949191	3.334989
Age	ALTERSKATEGORIE_GROB	Estimated age	3.296425	1.940437
	DECADE	Borned Decades	3.494692	5.635243

Table 2. Averaged Feature Attributes of Over-Representing and Under-Representing Clusters

Table 2 shows the average income and age between two typical over-representing and under-representing clusters, obtained by projecting the k-means centroids back to the original feature dimension.

For income attribute**, HH_EINKOMMEN_SCORE** scales the estimated household net income from 1 to 6, with higher scores indicating lower income. Cluster 6 corresponds to "very high income" and "high income" according to the data dictionary, while Cluster 2 corresponds to "lower income". (To see complete list of attributes, please follow the table in my Jupyter Notebook.)

Overall, we can concluded that a typical over-representing cluster is the group of people, with high-income in their middle-age born in 40–50s, involved in green avantgarde, and mostly living in the area of former West Germany. Individuals in one of the under-representing clusters trend to be on the opposite side: they are lower income earners, mainly younger female born in the 80–90s, and not a member of green avantgarde.

Part 2: Predictive Analytics of Customer Response Rate in a Mail-Out Campaign

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

label	Cluster_ID	Count	Frequency	Response_Rate

Trainset: Responded	6.0	139	26.127820	1.495589
Trainset: Responded	7.0	98	18.421053	1.380282
Trainset: Responded	10.0	97	18.233083	1.217828
Trainset: Responded	1.0	80	15.037594	1.205001
Trainset: Responded	8.0	35	6.578947	1.069029
Trainset: Responded	5.0	25	4.699248	1.245640
Trainset: Responded	9.0	20	3.759398	0.776699
Trainset: Responded	3.0	18	3.383459	0.697404
Trainset: Responded	4.0	11	2.067669	1.726845
Trainset: Responded	0.0	5	0.939850	0.822368
Trainset: Responded	2.0	4	0.751880	1.418440

Table 3. Percentage and the Average Response Rate of Each Cluster in the Mailout Trainset.

Table 3 shows the statistics about individuals who responded to the mail-out campaign in various clusters. The “frequency” column indicates the cluster percentage over all population. The “Response Rate” is derived from the amount of customer who responded over all customers in each cluster. Overall, the dataset is highly unbalance, with only about 1% of response rate. Based on this finding, I have applied stratified splitting based on the “**RESPONSE**” label to maintain relative similar ratio of response/no-response individuals in training set (75%) and testing set (25%). For each cluster, I bootstrapped the samples with response (y=1) to the same amount of no response (y=0).

Using the oversampled dataset, I have trained different machine learning models, including KNN, Logistic Regression, and ensemble methods (such as Random Forest Decision Tree, XGBoost, LGBM). By tuning the hyperparameters via GridSearchCV with 5-fold cross-validation, I have obtained the optimal classifier for each model.

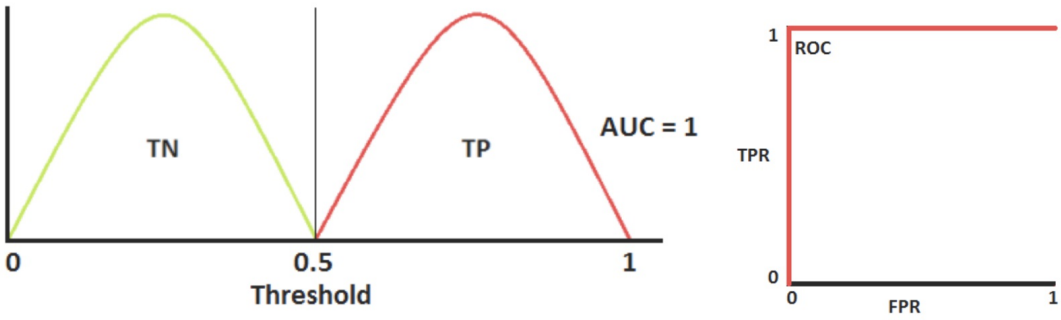
Part 2.1 Metrics

Area under the receiver operating characteristic curve (ROC_AUC) from predicted probabilities will be used to evaluate performances of the models. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random target person more highly than a random non-target person. Thus, ROC analysis provides tools to select possibly optimal models for customer prediction task.

Here, area under the “receiver operating characteristic” (ROC) curve (AUC) of predicted probabilities has been used to measure the classifier performance. The ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. Usually, it is plotted with true positive rate (TPR) on the y-axis against false positive rate (FPR) on the x-axis. The corresponding AUC score provides quantitative description about the curve characteristics, as shown in the following examples (see more explanation by this reference).

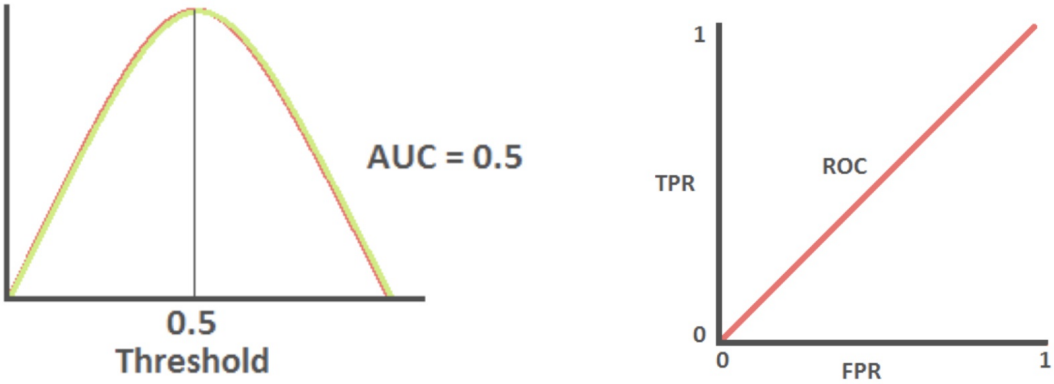
The figure below shows an example of a perfect classifier, in which all True Negatives

The figure below shows an example of a perfect classifier, in which all True Negatives (TN) and True Positives (TP) can be perfectly divided at 0.5 predicted probability. Tuning the threshold can only change either FPR or TPR. Therefore, the ROC corresponds to the red lines in the right panel, with an AUC score of 1.



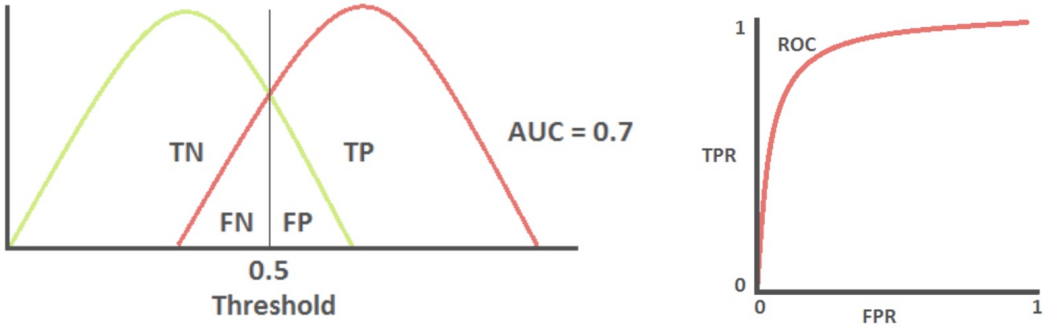
An Example of the Best Classifier (Image Source: Link)

To another extreme, the figure below shows the worse classifier, where the distribution of TPs and TNs completely overlaps each other. In this case, tuning the threshold scales the TPR and FPR linearly, as shown by the diagonal red line in the right panel, with an AUC score of 0.5.



An Example of the Worse Classifier (Image Source: Link)

In reality, the classifier often partially separate TNs and TPs, corresponding to the following case.



An Example of A Real-Life Classifier (Image Source: Link)

Part 2.2 Results

Now that we have a better understanding of AUC metric, We can compare AUC scores

on different classifiers. By just using the default hyperparameters, Random Forest, XGBoost and LGBM models obtains similar scores around 0.75–0.78, while KNN only scores about 0.67 on the training set. Therefore, KNN can be immediately ruled out.

GridSearchCV was used to fine tune the hyperparameters in Random Forest, XGBoost and LGBM models, including but not limited to: the number of estimators, tree depth, and learning rate. The three candidates obtained optimal AUC scores around 0.82. However, LGBM and Random Forest Algorithms are the most efficient in training (about 3 minutes), compared to XGBoost (41 minutes).

Figure 6 shows an example of KNN classifier (top panels) and a XGBoost classifier (bottom panels). A poor classifier (KNN) with an AUC score close to 0.5 corresponds to the scenario that all true positives and true negatives overlap each other, as shown by Panels (a)&(b). A decent classifier (XGBoost) with a AUC of about 0.8 corresponds to the case that most true negatives (no response) and true positives (response) were separated at a large margin. The XGBoost has done a good job in classifying true positives. However, there are still some true negatives misclassified: the small bump near 0.7 in panel (c), leading to an increase in the false positive rate in the ROC.

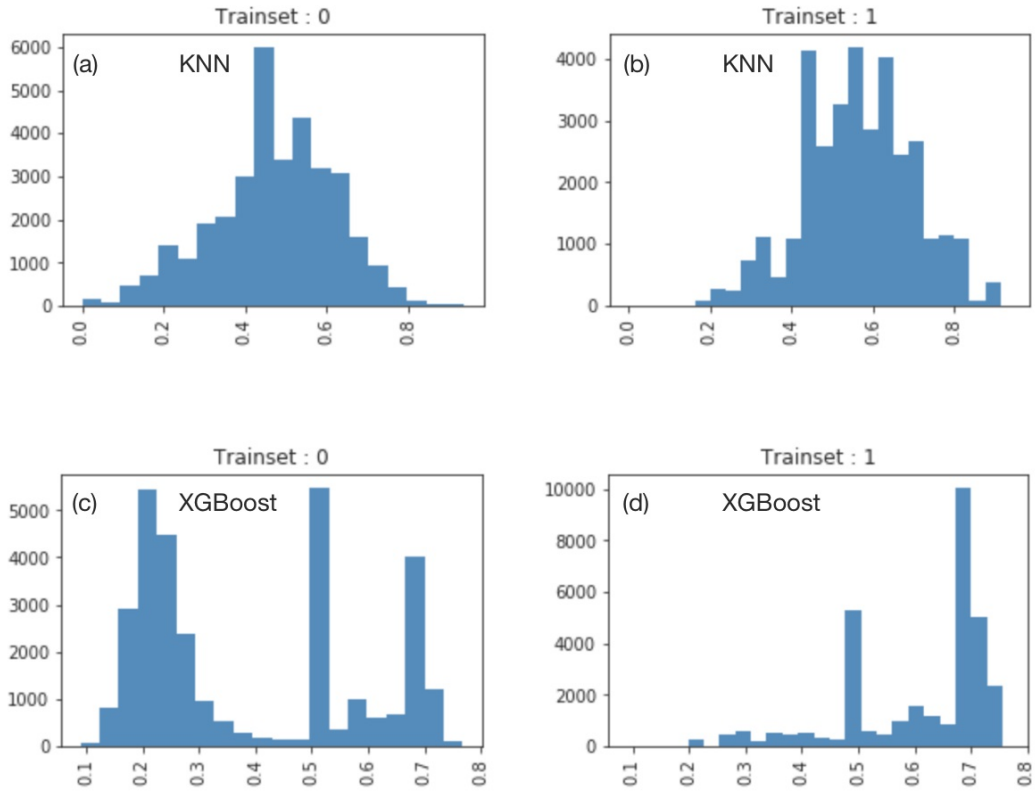


Figure 6. Distribution of true positives (panels a and c) and true negatives (panels b and d) distributed based on their predicted probabilities on the x-axis

	Cluster_ID	AUC Trainset	AUC Testset
0	0	0.966144	0.953642
1	1	0.894661	0.720762
2	2	0.850569	0.928571
3	3	0.857131	0.865835
4	4	0.810223	0.679406

5	5	0.839439	0.831149
6	6	0.817779	0.719466
7	7	0.819196	0.857143
8	8	0.880496	0.665706
9	9	0.920863	0.827074
10	10	0.577828	0.484767

Table 4. Comparison of AUC scores on each cluster.

In Figure 6 (c)&(d), it is also interesting to notice that a spike is present in both train set and test set at about 0.5 on the x-axis. In more details, I found that this part is contributed mainly by Cluster 10, which only scores slightly above 0.5 as shown by Table 4 and Figure 7. Recall that Cluster 10 represents the high row-NaN group. The low AUC on this cluster is very likely due to the lack of informative features.

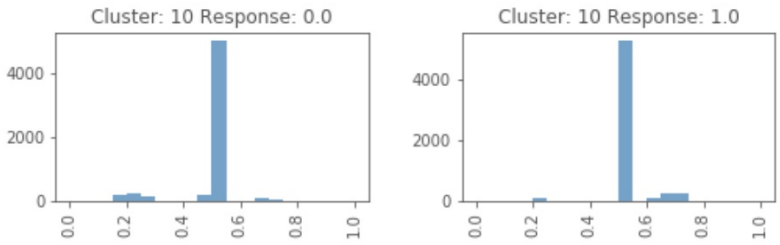


Figure 7

Figure 8 shows an example of top 20 most important features in the XGBoost model. In other tree-based models, the attribute of "**D19_SOZIALES**" has also been identified as the most importance feature in predicting the customer's response.

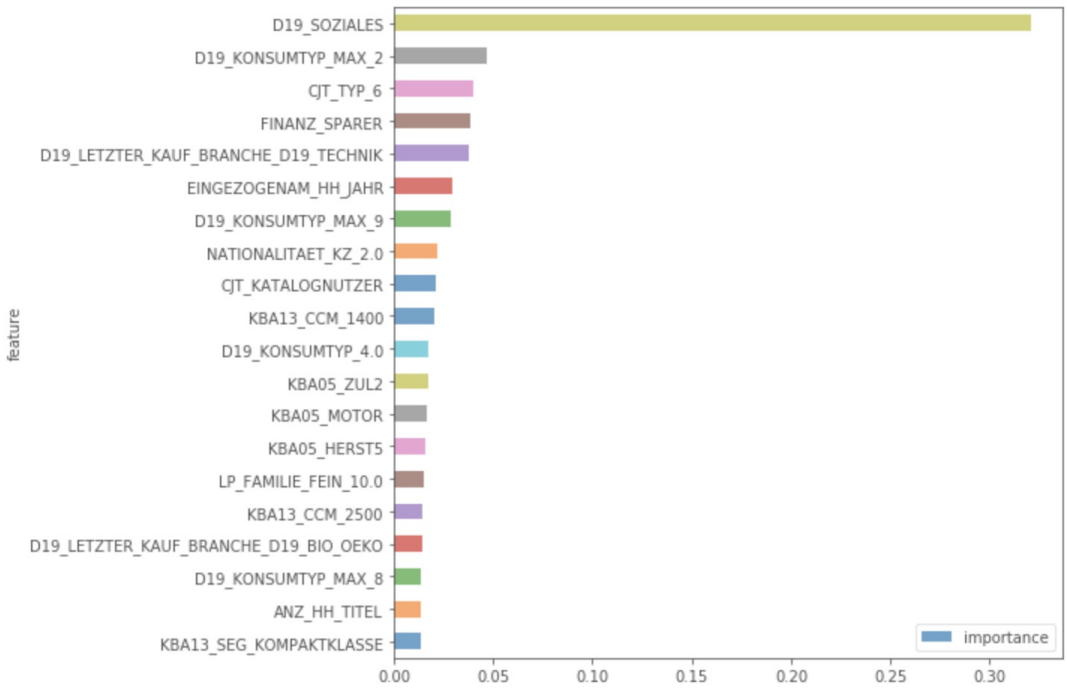


Figure 8. Features of Importance in XGBoost

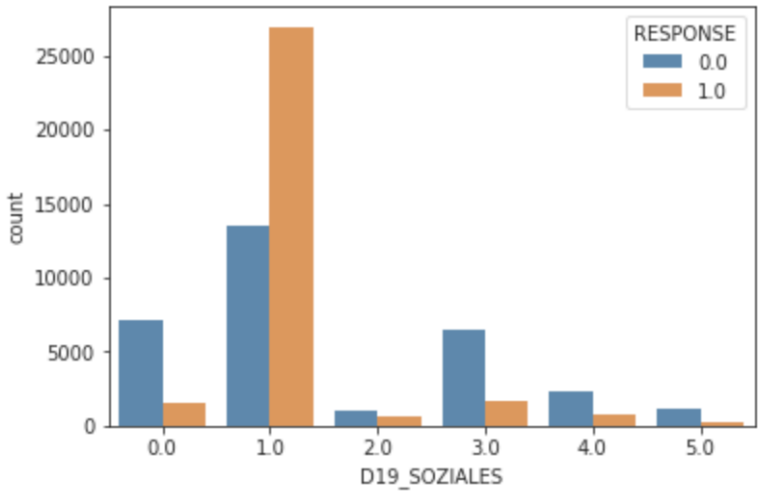


Figure 9. Distribution in '**D19_SOZIALES"**

Figure 9 shows that the majority of customers who responded to the mail-out campaign falls into the "1" bucket in the attribute of "**D19_SOZIALES",** while the "no response" customers spreads more evenly on the rest buckets. This is an example to show the power of a machine learning algorithm, in that it will be able to search for the most critical features among the many others.

Lastly, I decided to use LGBM model to predict the response rate probabilities on the **Mailout Testset.** After submission to Kaggle, I have achieved a AUC score of 0.79981.

Part 2.3 Improving AUC

To improve my model accuracy, I have attempted two approaches:

1. **Combining Results From Joint Classifiers**

This approach is just a simple average from the three classifiers: random forest (RF), XGBoost (XGB) and LGBM. The result shows that there is no improvement of AUC on both the train set and test set, as shown by Table 5.

	AUC on Trainset	AUC on Testset
RF Classifier	0.8383	0.7666
XGB Classifier	0.8101	0.7657
LGBM Classifier	0.8217	0.7625
RF + XGB + LGBM Classifiers	0.8307	0.7667
LGBM + Cluster-6 Classifiers	0.8261	0.7665

Table 5. List of AUC Scores Based on Different Models

2. **Improving AUC Score on Each Cluster**

As before, we have not utilized any clustering information to make a prediction. Now that we have clustered customers into segments of different attributes, can we improve the AUC score based on each cluster?

Here, I have tested on Cluster 6, since it is the majority group consisting of 26% of total

population (Table 3). Figure 10 below shows that aside from **D19_SOZIALES**, the ranking in other features is completely different from the previous model (Figure 8). This indicates that each segment may have different important attributes that determine whether or not the customer will respond to the campaign.

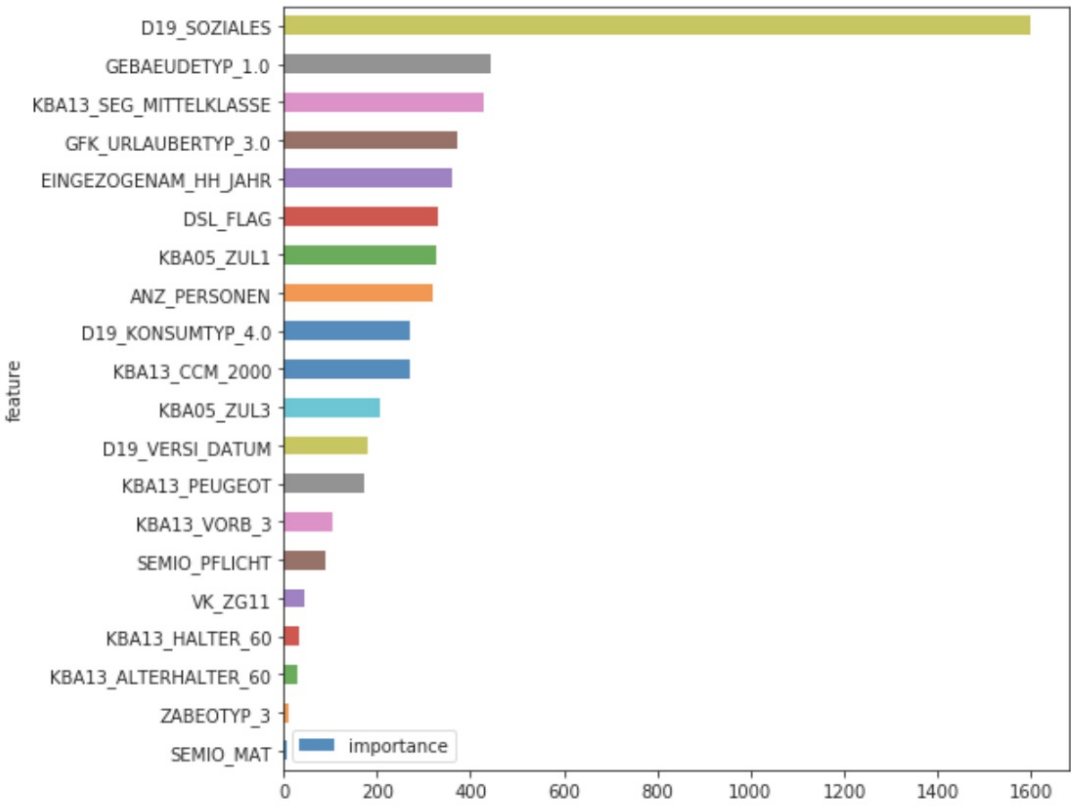


Figure 10. Feature of Importance For Cluster-6 Classifier

For Cluster-6 segment, I obtained AUC scores of 0.8192 and 0.7705 on the training and testing sets, respectively. Compared to Cluster-6 in Table 4 with AUC scores of 0.817779 (AUC Trainset) and 0.719466 (AUC Testset), there is a significant improvement on the testing set.

The next step is to average the predicted probabilities of the previous model and Cluster-6 model. The corresponding overall AUC scores are given in Table 5 (LGBM + Cluster-6 Classifiers). Compared to LGBM, both AUC scores on the training set and testing set have been slightly improved by about 0.4%.

Conclusions

In this report, I have attempted to solve real-life data science problems posed by Arvato Financial Solutions. Below is a summary of my major work:

1. I have conducted an exploratory analysis to gain better understanding about the datasets. Via accessing the missing values, I have found that individuals with a large portion of missing feature attributes are different from the majority. This cluster has been treated as a special group in throughout the analysis. Other feature re-encoding steps have been performed, including converting categorical features to dummy variables, and decomposing “mixed” features to “atomic” types.
2. To investigate customer segmentation, I applied PCA transformation to gain a better understanding about the relationship of correlated features. Using k-means, I have

understanding about the relationship of correlated features. Using K-means, I have segmented the general population and customers into 11 clusters. By comparing their distribution, I found the core customer base of the company. Some typical personal attributes of such clusters include a high income and a mature age.

3. Finally, I have applied several machine learning models to predict the probability of response rate of customers during a mail-out campaign. Using the LGBM model, I have achieved an AUC score of 0.79981 on Kaggle. I have also found a poor prediction in particular on the segment of individuals lacking too many informative features.

Future Work

There are several other aspects of improvement that have not been attempted yet by the time of writing this report. Below are some of the ideas:

1. In the data preprocessing step, we may try different NaN percentage thresholds for the row and column.
2. Think of engineering more features. There are a total of 90 attributes in the dataset not documented in the data dictionary, including **D19_SOZIALES**. More information about these attributes definitely help us decide which features to keep, drop and transform.
3. In the second part of building a supervised-learning model to predict customer response, we have already observed a slight improvement by combining averaged result of the general classifier based on no clustering, and a classifier specifically based on one major cluster. We can attempt to combine the general classifier with more of these cluster-base classifiers.
4. We could also perform exploratory analysis on all attributes, and build supervised-learning model only using those attributes with the most difference between overrepresented and underrepresented clusters.

In the end, I would like to express my sincere gratitude to all anonymous Udacity Reviewers who provided many in-depth suggestions on different projects.

This project is hosted in Github through this link.

Machine Learning

Zhitao Wang
PhD in Applied Physics, NJIT. “Knowledge as action.”