Machine Learning Engineer Nanodegree Capstone Project

Konstantinos Rotas

April 27th, 2022

Customer Segmentation Report for Arvato Financial Services

I. Definition

Project Overview

Arvato Bertelsmann, a Germany based financial services company, needs a data scientist to analyze demographic information and attributes from a mail-order company's existing clients and compare them to a larger dataset of people in Germany. The goal of the company is to acquire new clients more efficiently. The new advertising campaign will be targeted towards people that will be identified as new potential customers. We will build a model to predict which individuals are the most likely to be potential customers.

This project is about customer segmentation; the process of diving consumers into groups based on common traits so companies target each group suitably. Different methodologies that can be used for customer segmentation include parameters such as demographic, geographic, behavioral, and psychological. A machine learning approach to identify the underlying patterns would include K-means clustering, as we have unlabeled data. This way the algorithm discovers groups or clusters in the data and the number of clusters is represented by the K value. Each input data is assigned to one of K clusters, meaning that each set of data points is grouped by feature similarity (Kumar, 2021).

I chose this project as it a modern, real-life problem that requires different approaches and strategies. There is no one size fits all customers, and it is important to uncover underlying traits that will help a company to better serve each new and existing customer.

Problem Statement

The main question that we are trying to answer is how can Arvato's client, the mail-order company acquire new clients more efficiently? By analyzing the attributes of existing clients and those of the general population of Germany we would like to identify which people in Germany are most likely to become customers for the mail-order company selling organic products, using unsupervised learning techniques. We can improve the customer acquisition process by targeting specific people with a marketing campaign rather than the general population of Germany. To solve the

problem, we will also train supervised machine learning models to predict the response of individual customers to the marketing campaign and then assess our model against a test set.

Metrics

To evaluate the performance of our classification models we may use the AUC - ROC curve, which is also useful in the case of class imbalance. Most people didn't respond to the mail-out, which is why classes have very different value counts. ROC - Receiver Operating Characteristics is a probability curve and AUC - Area Under the Curve shows the degree of separability between classes. When the AUC is higher, the better the model is at correctly predicting the classes. To plot the ROC curve, we need the TPR (True Positive Rate) on the y-axis and FPR (False Positive Rate) on the x-axis. A good model has AUC near to 1, meaning that it has a good measure of separability (Narkhede, 2018).

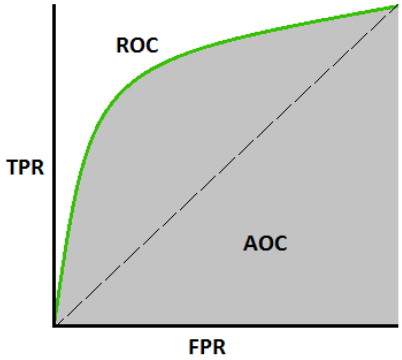

Figure 1: AUC - ROC Curve

$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{FPR} = 1 - \text{Specificity}$$

$$= \frac{FP}{TN + FP}$$

Figure 2: TPR and FPR definitions

II. Analysis

Data Exploration

There are four datasets and two description files to utilize in this project:

- Udacity_AZDIAS_052018: Demographic data for the general population of Germany. 891.211 persons
- Udacity_CUSTOMERS_052018: Demographic data for existing customers of a mail-order company. 191.652 persons
- Udacity_MAILOUT_052018_TRAIN: Demographic data for individuals who were targeted by a marketing campaign. 42.982 persons
- Udacity_MAILOUT_052018_TEST: Demographic data for individuals who were targeted by a marketing campaign. 42.833 persons
- DIAS Attributes — Values 2017: Detailed mapping of data values for each feature (alphabetical order).
- DIAS Information Levels — Attributes 2017: List of attributes and descriptions, organized by informational category.

The “AZDIAS” and “CUSTOMERS” datasets consist of more than 365 columns of data, with each column representing an attribute or feature. Each row of both datasets represents a person, including information about their household, building, and neighborhood. The "CUSTOMERS" file contains three extra columns ('CUSTOMER_GROUP', 'ONLINE_PURCHASE', and 'PRODUCT_GROUP'), which provide more information about the customers. The original "MAILOUT" file included one additional column, "RESPONSE", which indicated whether each recipient became a customer of the company.

```
1  print(azdias.info())
2  print(customers.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891221 entries, 0 to 891220
Columns: 366 entries, LNR to ALTERSKATEGORIE_GROB
dtypes: float64(267), int64(93), object(6)
memory usage: 2.4+ GB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 191652 entries, 0 to 191651
Columns: 369 entries, LNR to ALTERSKATEGORIE_GROB
dtypes: float64(267), int64(94), object(8)
memory usage: 539.5+ MB
None
```

Figure 3: Data types (AZDIAS, CUSTOMERS)

As we can see from Figure 3 the datatype for each dataset is mixed; Most of the data contain float64 values, followed by int64 and a few objects. We can also notice that the memory usage for the azdias dataset is much larger which might become a problem in computation if we don't make a good choice of a Notebook instance.

When we print the first few values of the data (Figure 4) we can easily notice that there are many NaN values which will have to be considered when we process the data. A closer look at the "Attributes" file reveals some unknown values represented as 'X', 'XX', '-1', '0', etc. The dataset is also comprised of many missing values in both rows and columns.

There are 6 categorical columns in "azdias", one of which is a date column. "Customers" has two more columns – Product Group and Customer Group which are not in the general population dataset (Figure 5). More specifically, CAMEO_DEU_2015, CAMEO_DEUG_2015, CAMEO_INTL_2015 depict the sociodemographic background of a household based on the CAMEO classification system. D19_LETZTER_KAUF_BRANCHE is probably the category of the latest buy of a household. EINGEFUEGT_AM is the date when a household was added to the database and OST_WEST_KZ shows in which area the building was located - former East Germany (O) or former West Germany (W).

```
1 customers.head()
```

Python

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ
0	9626	2	1.0	10.0	NaN	NaN	NaN	NaN	10.0	
1	9628	-1	9.0	11.0	NaN	NaN	NaN	NaN	NaN	
2	143872	-1	1.0	6.0	NaN	NaN	NaN	NaN	0.0	
3	143873	1	1.0	8.0	NaN	NaN	NaN	NaN	8.0	
4	143874	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	

5 rows × 369 columns

```
1 azdias.head()
```

Python

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ
0	910215	-1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	910220	-1	9.0	0.0	NaN	NaN	NaN	NaN	21.0	
2	910225	-1	9.0	17.0	NaN	NaN	NaN	NaN	17.0	
3	910226	2	1.0	13.0	NaN	NaN	NaN	NaN	13.0	
4	910241	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	

5 rows × 366 columns

Figure 4: Data Sample – Customers, AZDIAS

```
1 customers.select_dtypes(['object']).head(5)
```

	CAMEO_DEU_2015	CAMEO_DEUG_2015	CAMEO_INTL_2015	D19_LETZTER_KAUF_BRANCHE	EINGEFUEGT_AM	OST_WEST_KZ	PRODUCT_GROUP	CUSTOMER_GROUP
0	1A	1	13	D19_UNBEKANNT	1992-02-12 00:00:00	W	COSMETIC_AND_FOOD	MULTI_BUYER
1	NaN	NaN	NaN	D19_BANKEN_GROSS	NaN	NaN	FOOD	SINGLE_BUYER
2	5D	5	34	D19_UNBEKANNT	1992-02-10 00:00:00	W	COSMETIC_AND_FOOD	MULTI_BUYER
3	4C	4	24	D19_NAHRUNGSERGAENZUNG	1992-02-10 00:00:00	W	COSMETIC	MULTI_BUYER
4	7B	7	41	D19_SCHUHE	1992-02-12 00:00:00	W	FOOD	MULTI_BUYER

```
1 azdias.select_dtypes(['object']).head(5)
```

	CAMEO_DEU_2015	CAMEO_DEUG_2015	CAMEO_INTL_2015	D19_LETZTER_KAUF_BRANCHE	EINGEFUEGT_AM	OST_WEST_KZ
0	NaN	NaN	NaN	NaN	NaN	NaN
1	8A	8	51	NaN	1992-02-10 00:00:00	W
2	4C	4	24	D19_UNBEKANNT	1992-02-12 00:00:00	W
3	2A	2	12	D19_UNBEKANNT	1997-04-21 00:00:00	W
4	6B	6	43	D19_SCHUHE	1992-02-12 00:00:00	W

Figure 5: Categorical data sample

Exploratory Visualization

A very informative visualization for the missing values in azdias is a histogram plot, which depicts the distribution of missing value percentage per column. On average, columns are missing 11.49% of their total values; around 130 columns. Approximately 94% of columns have a missing segment of 24.77% or less. Utilizing this chart, we can set a threshold of missing values in order to drop uninformative columns.

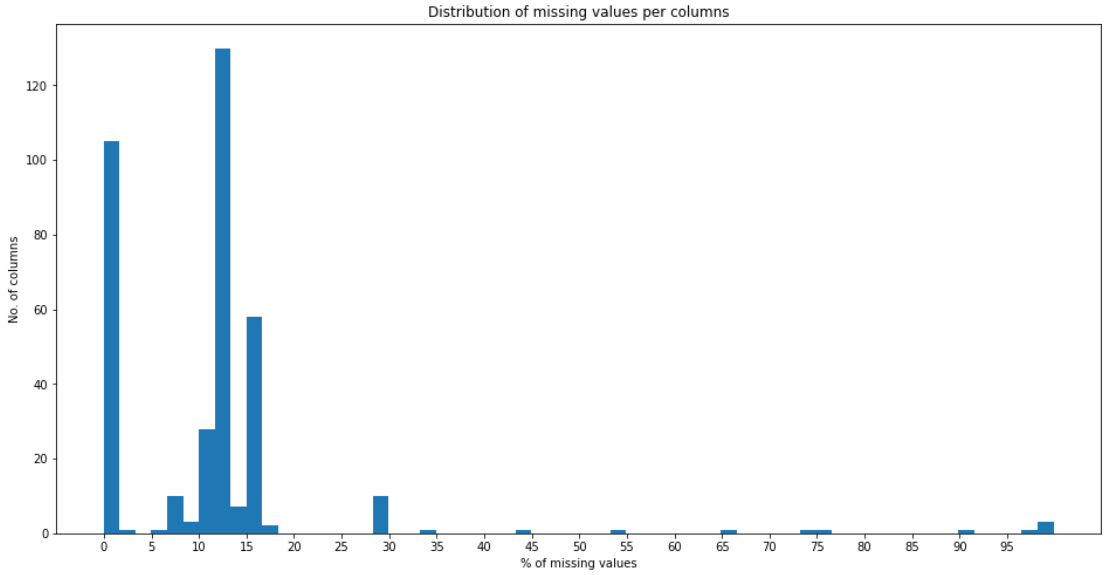

Figure 6: Missing values per column

Another informative visualization is the percentage of missing values of the first 50 columns, sorted in descending order Figure 7. An interesting observation is the fact that the top 5 columns are missing more than 90% of their values. Among those columns is the attribute of whether a person holds an academic title and the age of children within a household. Some columns are missing between 30 and 80% of their values, while most columns are missing less than 20% of their values. This plot is helpful in deciding which columns to drop when cleaning the dataset.

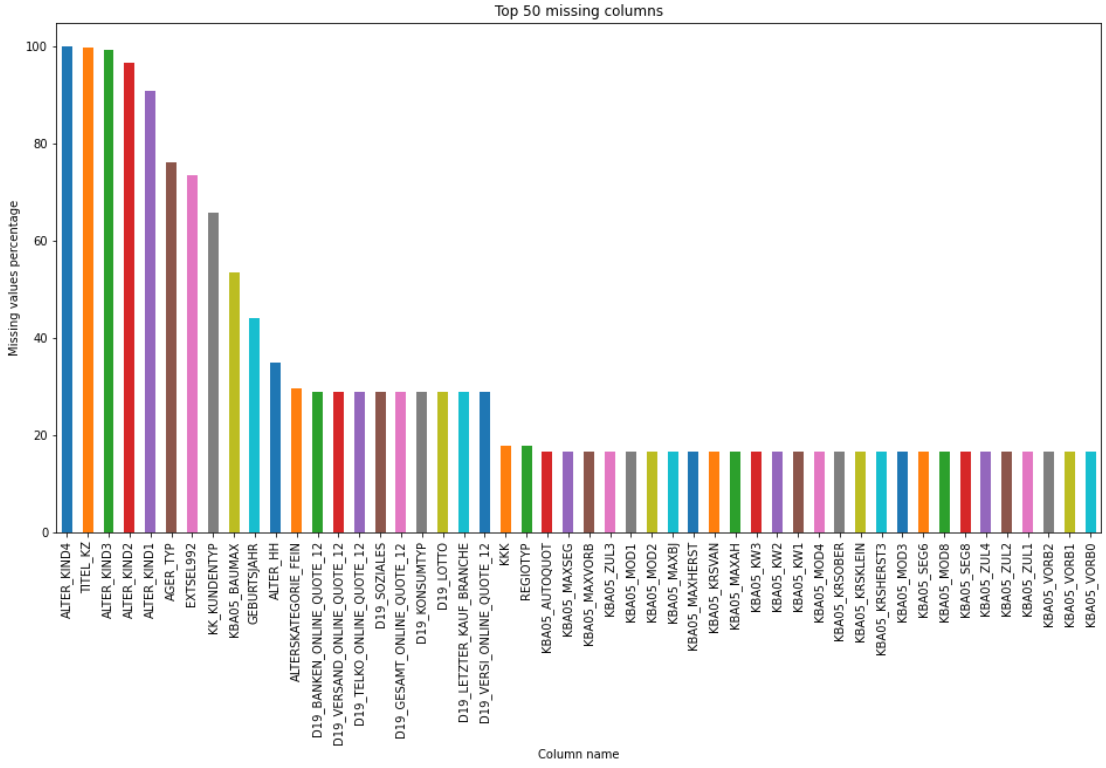

Figure 7: Top 50 columns with most missing values

The same process is followed with missing values in every row of the dataset. In the following plot (Figure 8) we can observe how many values are missing. Most rows are missing between 1-30 values with only a few missing almost 260 values. The cumulative sum of missing values is 37.480.954. Again, we will define a threshold to drop rows and clean the dataset.

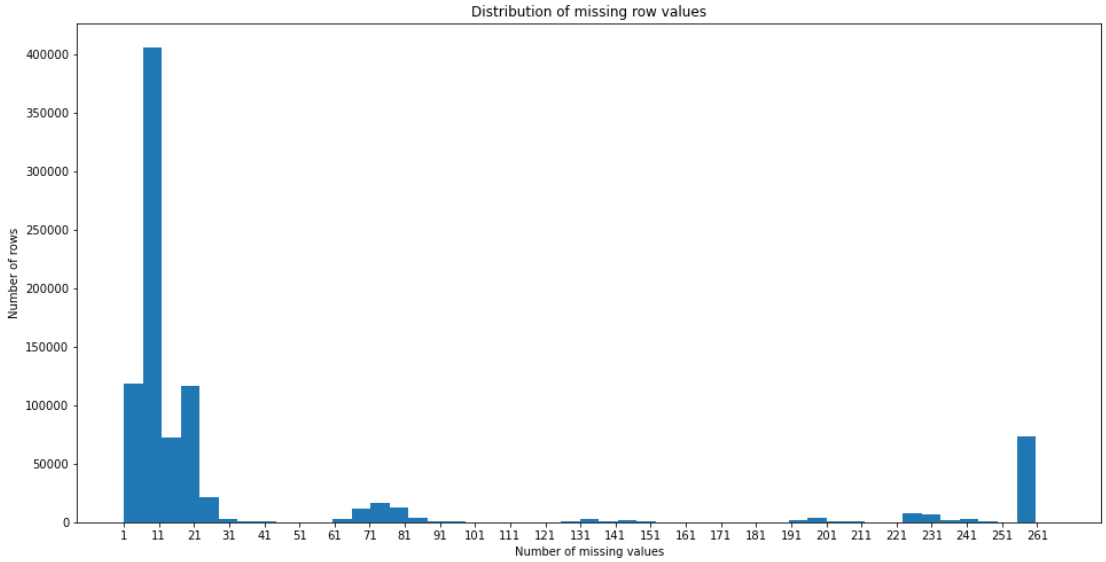

Figure 8: Distribution of missing row values

Algorithms and Techniques

For the unsupervised part of the project demographic information was used to determine groups of people with similar traits. To achieve this, we employed Principal Component Analysis (**PCA**), a technique that reduces the number of features using Singular Value Decomposition. The data is projected to a lower dimensional space by retaining most of the data variance so very little information is lost. Principal components represent the information as a combination of features.

The second algorithm that was utilized is **K-means**, which categorizes the data set into k groups of similarity. To calculate similarity, the Euclidean distance is utilized as measurement. Firstly, k points are randomly initialized and then every data point is assigned to the nearest mean. The mean's coordinates are updated after each point is assigned and the final clusters are formed after a given number of iterations.

The supervised part of the project required to predict whether a person should be included to the marketing campaign or not, using the demographic data. The following algorithms were considered:

Random Forest Classifier creates a set of decision trees from a randomly selected subset of the training set. The votes from different decision trees are collected and the algorithm makes a final prediction.

Boosting is an ensemble technique that attempts to create a strong classifier from several weak classifiers.

AdaBoost (Adaptive Boosting) is such an ensemble algorithm which builds a model from the training data and then creates a second model which tries to correct errors from the first model.

Gradient boosting is also an ensemble algorithm that combines many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting.

XGBoost (eXtreme Gradient Boosting) is an implementation of gradient boosting trees algorithm and a popular supervised model that offers good performance and computational speed. It also supports parallelization of tree construction, sparse aware implementation, etc.

Benchmark

A simple model that can be used as a benchmark is Logistic Regression. We can establish a baseline by using this model to later compare with other models. To evaluate the model, we used AUC-Score as a metric. The default parameters were used for this model with iter=500 to obtain the baseline score and compare results.

III. Methodology

Data Preprocessing

Preprocessing steps were needed in order to prepare the data for machine learning use. Initially, unknown values were converted to NaN for each attribute based on the "DIAS Attributes – Values" by creating a dictionary of missing keys. Subsequently, data was cleaned from missing columns and rows that contained more NaNs than specific thresholds. Columns that had more than 25% missing values were removed (Figure 9). Also, the column "LNR" contained unique identifiers for each person and was also removed. Similarly, the process for removing rows with many NaN values was followed, using a threshold of 20 values per row. This resulted in 158315 rows being removed for the azdias dataset (Figure 10). For the customers dataset, the three extra columns were also removed.

```
25 columns will be dropped from the dataset: ['AGER_TYP', 'ALTER_HH', 'ALTER_KIND1',
'ALTER_KIND2', 'ALTER_KIND3', 'ALTER_KIND4', 'ALTERSKATEGORIE_FEIN',
'D19_BANKEN_ONLINE_QUOTE_12', 'D19_GESAMT_ONLINE_QUOTE_12', 'D19_KONSUMTYP',
'D19_LETZTER_KAUF_BRANCHE', 'D19_LOTTO', 'D19_SOZIALES', 'D19_TELKO_ONLINE_QUOTE_12',
'D19_VERSAND_ONLINE_QUOTE_12', 'D19_VERSI_ONLINE_QUOTE_12', 'EXTSEL992', 'GEBURTSJAHR',
'KBA05_BAUMAX', 'KK_KUNDENTYP', 'TITEL_KZ', 'CAMEO_DEUG_2015', 'CAMEO_INTL_2015',
'D19_LETZTER_KAUF_BRANCHE', 'LNR']
```

Figure 9: Columns to be removed

```
1  azdias_new = df_clean(azdias,drop_cols)

Original datasest shape: (891221, 366) ---> New dataset shape: (732906, 342)
Rows dropped: 158315 , Columns dropped: 25
float64    254
int64       87
object       1
dtype: int64
```

Figure 10: Cleaned AZDIAS dataset info

The next step was to employ different imputation strategies according to each data type. For the numerical attributes the missing values were replaced by the median value of the respective feature columns. Missing values for categorical data were substituted with the most common value after they were one-hot encoded. For categorical data to be processed, all input variables and output variables need to be numeric. Finally, there are some attributes that are numerical, but comprise continuous values rather than discrete such as the number of people in a household. This case presents more skewed distributions and need to be log-transformed before being imputed by the median value of each attribute. All features were transformed, scaled, and cleaned in order to produce the final dataset which was suitable for PCA treatment. The features were scaled onto unit scale with mean=0 and variance=1 using standard scaler, which assists in having optimal performance when used with algorithms.

Implementation

Firstly, the PCA algorithm was employed to perform dimensionality reduction and reduce the features to almost half, while preserving most of the information. The minimum number of principal components were extracted such that 95% of the variance was retained. Another interesting fact is that the first 5 principal components were able to explain almost 25% of the variance of the data. The first component (8.5% of variance) is mainly comprised of the social status of a family, moving patterns, number of households known in a building and number of buildings in the area.

	weight
MOBI_REGIO	0.14
PLZ8_ANTG1	0.13
KBA13_ANTG1	0.13
LP_STATUS_FEIN	0.13
KBA05_ANTG1	0.13
LP_STATUS_GROB	0.13
MOBI_RASTER	0.13
KBA05_AUTOQUOT	0.13
KBA05_GBZ	0.12
KBA13_AUTOQUOTE	0.12

	weight
ORTSGR_KLS9	-0.11
EWDICHTE	-0.11
HH_EINKOMMEN_SCORE	-0.13
PLZ8_BAUMAX	-0.13
PLZ8_ANTG4	-0.13
KBA13_BAUMAX	-0.13
KBA13_ANTG4	-0.13
KBA13_ANTG3	-0.13
PLZ8_ANTG3	-0.13
ANZ_HAUSHALTE_AKTIV	-0.13

Figure 11: First Principal Component Weights

The second principal component (5.2% of variance) is comprised of the car that a person owns. More specifically, the most positively weighted features included expensive cars (upper middle class/ upper class), Mercedes and BMW owners and owners of cars with a greater top speed. The negatively weighted features showed a relationship with more affordable cars and cars with lower top speed limits.

The third principal component (4.9% of variance) is comprised of online affinity, dominating movement in a person's youth (avantgarde or mainstream) and the financial behavior of a person, transactions, etc.

The next step of the implementation is to employ the K-means algorithm and determine the optimum number of k based on the elbow method. The model was trained using 40000 samples of data for a ranging number of k. Values ranging from 1 to 15 were used. The point where the distortion declines the most is the elbow point and, in this case, k was chosen as 10 (Figure 12). Distortion is defined as the average of the squared distances from the cluster centers of the respective clusters. The number 10 represents the different categories of customers that can be deduced from the data.

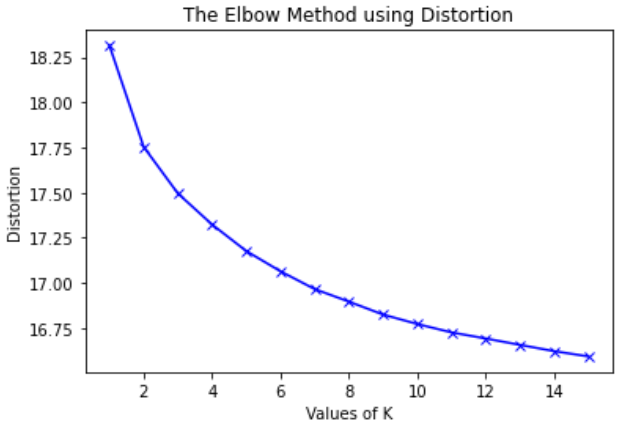

Figure 12: K-Means Elbow Plot (Distortion)

We then trained and fitted a model with parameters k=10 and pca_components = 219 using the complete azdias dataset to observe the differences in cluster distribution between the general population and the customers (Figure 13).

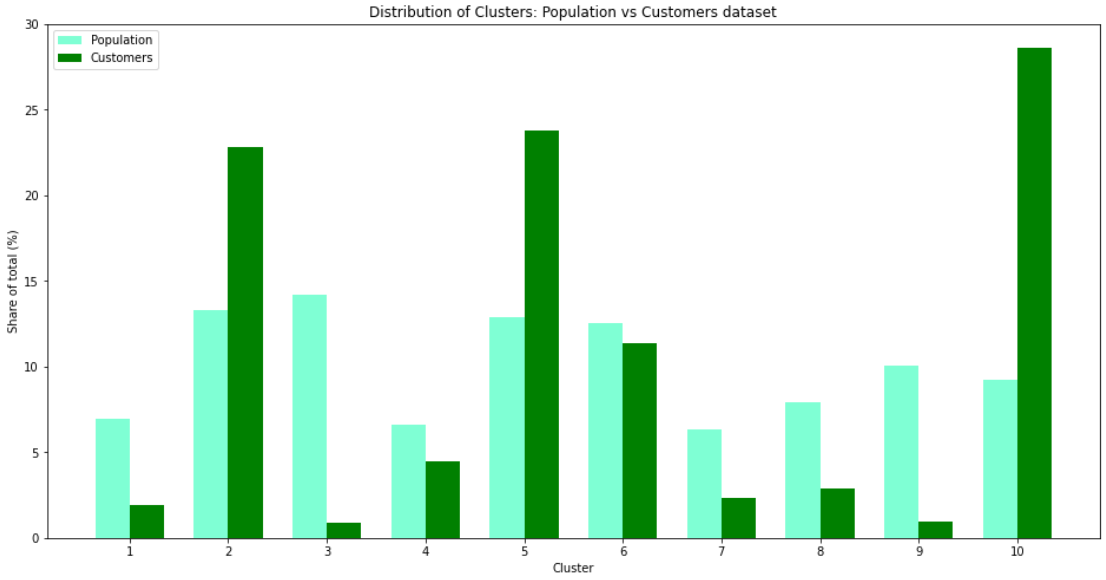

Figure 13: Distribution of general population and customers between clusters.

Most customers are mapped in clusters 2, 5 and 10, while the general population is spread more widely between clusters. The potential target audience is revealed where customer proportion is greater than the population share. To make this relationship evident we can use another plot which presents the differences by cluster (Figure 14).

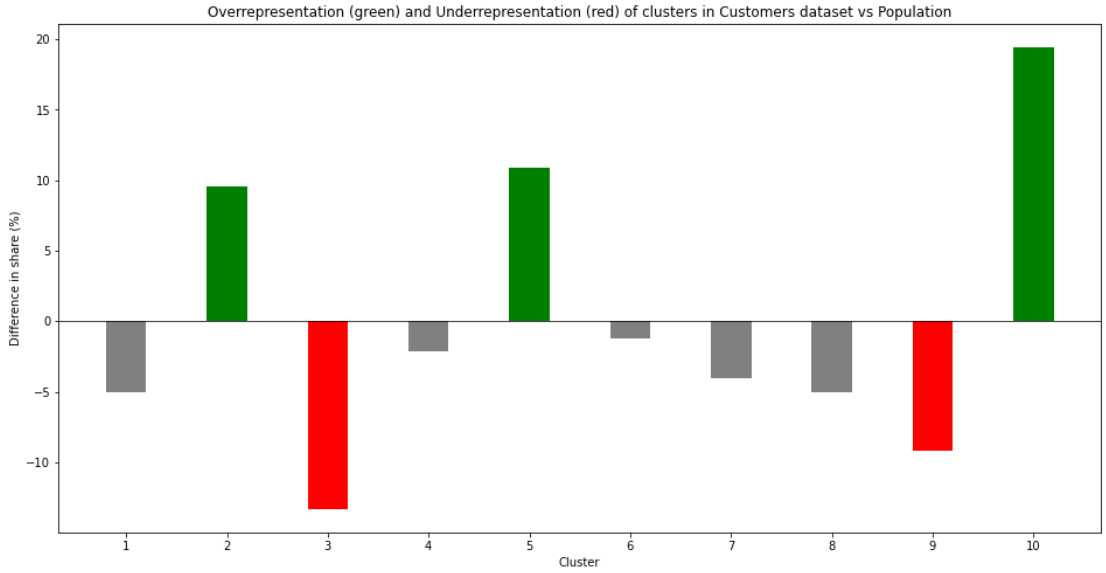

Figure 14: Customer - Population differences by Cluster

Customer clusters 2, 5 and 10 have a higher proportion than their population counterparts, meaning that people that fit in these groups comprise the target audience. On the contrary, clusters 3 and 9 have a smaller proportion than their population counterparts, meaning that the people that belong in this group are not included the target audience.

To interpret the results of the clustering we calculated the cluster centroids and performed inverse transformations to the scaled features to reveal the original features.

	Target_cluster_2	Target_cluster_5	Target_cluster_10	Non-Target_cluster_3	Non-Target_cluster_9
D19_KONSUMTYP_MAX	6.136881	2.303843	4.451249	7.318042	7.337636
AKT_DAT_KL	3.370823	2.460624	3.206871	6.618513	6.371015
VK_DISTANZ	7.854193	3.980604	6.243343	9.525835	9.493877
LP_LEBENSPHASE_FEIN	17.078793	29.855299	20.064214	9.583810	7.117255
LP_STATUS_FEIN	7.803849	7.908741	7.263306	3.895805	1.715301
VK_DHT4A	6.755057	2.631049	5.284133	7.170402	7.377971
D19_GESAMT_ONLINE_DATUM	8.675864	3.841470	7.080302	9.037774	8.863520
VERDICHTUNGSRAUM	2.454237	2.913530	5.759242	3.617463	7.063893
D19_VERSAND_ONLINE_DATUM	8.977611	4.220924	7.429300	9.219214	9.053129
EINGEZOGENAM_HH_JAHR	2000.868381	2000.933155	2001.652572	2005.742488	2005.837932
LP_FAMILIE_FEIN	3.157575	8.022417	4.378277	2.654929	2.246080
VK_ZG11	5.692816	3.530433	4.623958	7.883442	7.700179
ANZ_STATISTISCHE_HAUSHALTE	2.187761	1.603764	3.177824	5.365858	15.330113
D19_VERSAND_DATUM	8.642593	3.784146	7.029752	9.134018	8.977512
GEMEINDETYP	33.033681	31.674203	20.345508	27.820899	13.355545
LP_LEBENSPHASE_GROB	4.921009	9.028567	5.842731	3.030058	2.477268
PRAEGENDE_JUGENDJAHRE	6.150962	10.333012	7.831630	12.426742	11.624202
D19_GESAMT_DATUM	7.835267	2.946249	6.164217	8.645937	8.471231
ORTSGR_KLS9	3.538483	3.796022	5.980035	4.453316	7.531124

Figure 15: Feature Centroids

The clusters that comprise the target groups show characteristics such as high-income families that have been shopping online, have life stage, social status and family type marked as fine. On the contrary, the non-target clusters are comprised of people that

tend to have low-income, live in less densely populated areas (rural) and don't usually shop online.

For the supervised part of the project, we used different models and chose the best one to perform grid search and achieve better results, using the mailout_train dataset. The dataset was pre-processed through the previously defined pipeline. There is a vast difference between the classes of the "RESPONSE" target, since only 1.2% of people responded to the campaign. Therefore, the AUC metric was set as appropriate metric for this project.

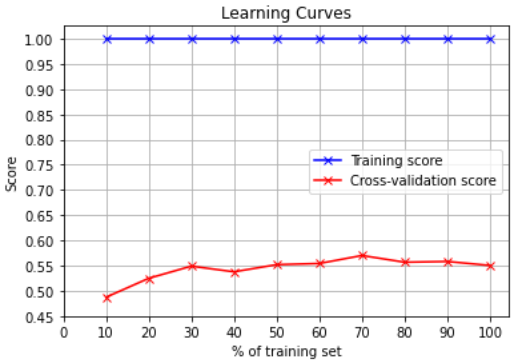

Figure 16: RandomForestClassifier

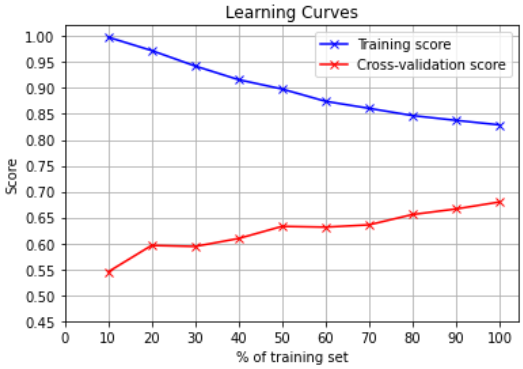

Figure 17: AdaBoostClassifier

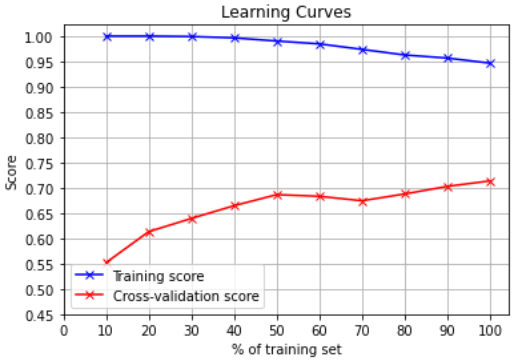

Figure 18: GradientBoostingClassifier

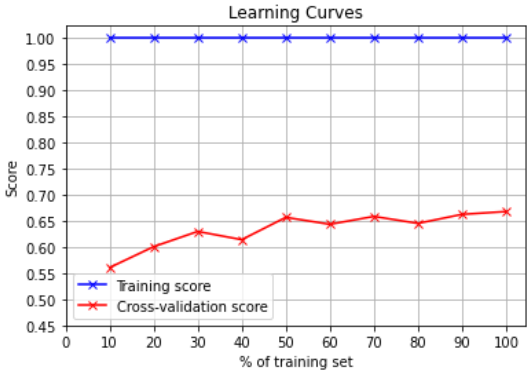

Figure 19: XGBClassifier

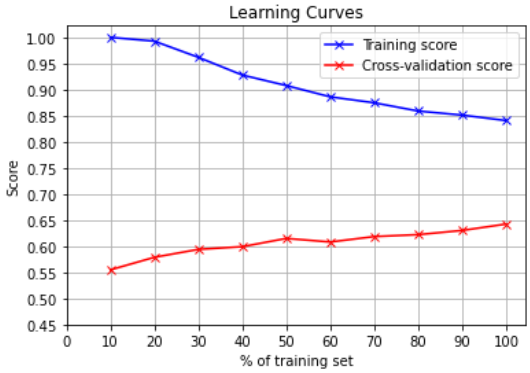

Figure 20: Logistic Regression

By observing the learning curves, we notice that the Random Forest Classifier and the XGBoost Classifier have a perfect training score, which means that there is high bias and at the same time the cross-validation score is very low. AdaBoostClassifier and

GradientBoosting Classifier demonstrate better performance while the training and cross-validation scores are converging. The AUC validation score is 0.71 for the GradientBoosting Classifier and is chosen as our final model for further improvements.

Refinement

The GradientBoosting Classifier was chosen for further improvements using Grid Search which is an extensive search over specified parameter values for an estimator.

```
#GradientBoostingClassifier
gbc_pipe = Pipeline([
    ('transform', column_transformer),
    ('gbc', GradientBoostingClassifier(random_state=42))
])

parameters = {'gbc__learning_rate': [0.1, 0.2],
              'gbc__n_estimators': [100],
              'gbc__max_depth': [3, 5],
              'gbc__min_samples_split': [2,4]
             }
# Fit the grid search object to the training data to find optimal parameters

grid_obj = GridSearchCV(gbc_pipe, parameters, scoring = 'roc_auc', verbose=2)
grid_obj.fit(X, y)
```

Figure 21: Estimator Parameters for Grid Search

Every iteration of the search was fitted for 1-2 minutes with a total of 40 iterations. The best parameters of the classifier were:

{'gbc__learning_rate': 0.1,

'gbc__max_depth': 3,

'gbc__min_samples_split': 4,

'gbc__n_estimators': 100}

These parameters produced a final ROC-AUC score of 0.9276 from an initial ROC-AUC score of 0.9255 with default parameters.

Finally, the top 10 features used in the prediction of customer response were recovered. D19_KONSUMTYP_MAX (consumption type) was the most important feature (Figure 22).

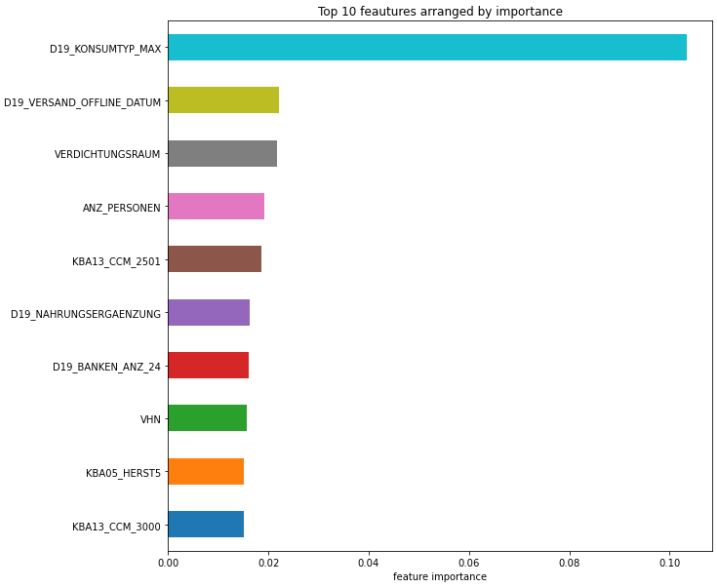

Figure 22: Feature importance

IV. Results

Model Evaluation and Validation

The predictions of the model on the response of individuals to a mailout campaign were tested on data from Kaggle to produce a final score of 0.68583. This means that the model is an adequate solution to the problem since it generalizes on other datasets and is better than random guessing (score 0.5). Gradient boosting is robust to over-fitting which is also a good indication that this solution performs well.

We can also notice that our baseline model, Logistic Regression in Figure 20 provides good performance and doesn't overfit like the other models. The optimized GradientBoosting Classifier in comparison provides stronger results than the baseline even before using the best parameters.

The parameters that were tested with grid search include the learning rate which shrinks the contribution of each tree, the maximum depth of the individual regression estimators, the minimum number of samples required to split an internal node.

V. Conclusion

Reflection

The Bertelsmann-Arvato Project posed some questions that we are trying to answer through supervised and unsupervised models. The main problem was "How can Arvato's client, the mail-order company acquire new clients more efficiently?" The biggest challenge was to handle and preprocess the datasets that consisted of many missing values, different datatypes and required transformations such as dimensionality reduction. We identified the customer and general population clusters and segmented

them into 10 groups. From these clusters 3 groups were identified as potential customers that might respond positively to a marketing campaign. Through analysis of the feature space, we were able to describe the customer base as well-earning with a habit of making online purchases. The Gradient Boosting Classifier performed better against the baseline and was optimized to produce better results.

Improvement

The best results were submitted to Kaggle and indicate that results can certainly further improve. Potential changes that would change the outcome of the project would be to set different thresholds when handling missing values or using different imputation strategies, scalers, etc. Another change would be to use some techniques to counter for the imbalance of the classes in the supervised model. When training the algorithms other parameters could be also tested with grid search and more classifiers could be tested in the pipeline. Finally, a different number of clusters could be decided to create customers groups.

References

1. Brownlee, J., 2016. *A Gentle Introduction to XGBoost for Applied Machine Learning*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/> [Accessed 17 April 2022].
2. Brownlee, J., 2016. *Boosting and AdaBoost for Machine Learning*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/> [Accessed 20 April 2022].
3. Buitinck, L. Et al, 2013. API design for machine learning software: experiences from the scikit-learn project. [online] Available at: <https://arxiv.org/abs/1309.0238> [Accessed 4 April 2022].
4. GeeksforGeeks. 2021. *K means Clustering - Introduction*. [online] Available at: <https://www.geeksforgeeks.org/k-means-clustering-introduction/> [Accessed 12 April 2022].
5. GeeksforGeeks. 2021. *Random Forest Classifier using Scikit-learn*. [online] Available at: <https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/> [Accessed 21 April 2022].
6. Kumar, D., 2021. Implementing Customer Segmentation Using Machine Learning [Beginners Guide] - neptune.ai. [online] neptune.ai. Available at: <https://neptune.ai/blog/customer-segmentation-using-machine-learning> [Accessed 3 April 2022].
7. Narkhede, S., 2018. *Understanding AUC - ROC Curve*. [online] Medium. Available at: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5> [Accessed 2 April 2022].

8. Nelson, D., 2019. Gradient Boosting Classifiers in Python with Scikit-Learn. [online] Stack Abuse. Available at: <https://stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn/> [Accessed 20 April 2022].