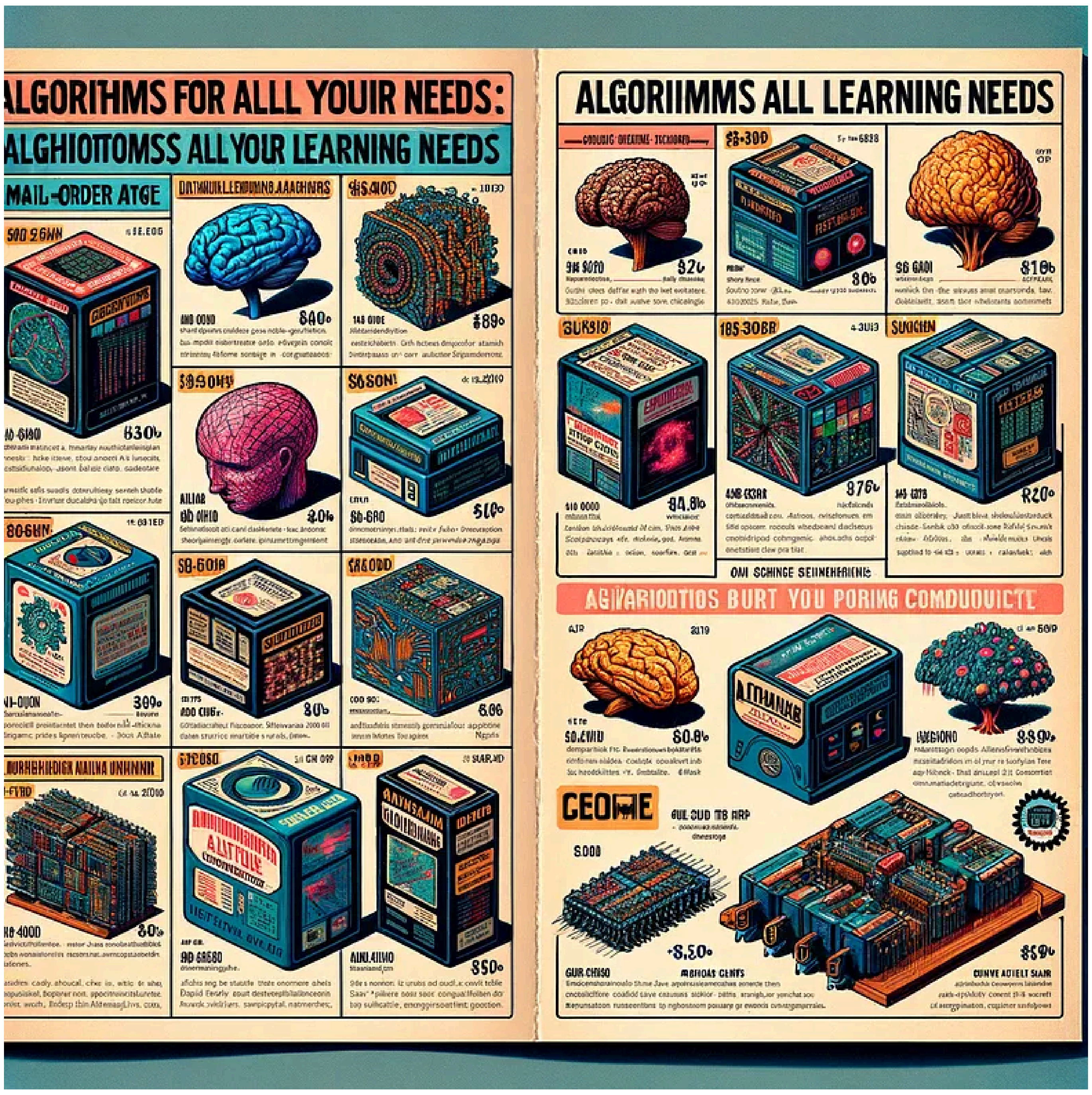
Data Science approach to Marketing Strategies: customer segmentation and campaign conversion

Jakub Below · Follow

11 min read · Just now


ALGORITHMS FOR ALLL YOUIR NEEDS:
ALGHIOTTOMSS ALL YOUR LEARNING NEEDS
MAIL-ORDER ATGE
ALGORIIMMS ALL LEARNING NEEDS
AGIVARIODTIOS BURT YOU PORIMG COMDUDUICTE
CEOME

Introduction

This project was driven by two primary questions: How does the company's customer base compare to the general population of Germany, and what is the conversion level for the target audience of the given marketing campaign? Addressing these questions required a comprehensive analysis of demographic data, with the ultimate goal of enhancing the company's marketing efforts through targeted customer segmentation and predictive modeling.

The data has been provided by Bertelsmann Arvato Analytics.
This project has been created as a part of the Udacity Nanodegree Data Scientist program.

Data Explanation

We've been provided with four files:

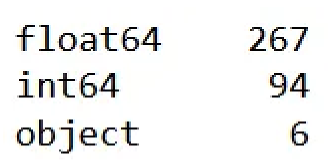
1. **Azdias** file containing demographic data for the general population of Germany, containing almost 400 features and 900 K entries.
2. **Customers** file containing data for existing customers and as such includes 3 additional columns:
   - customer group
   - online purchase flag
   - product group
3. **Mailout train** file containing largely the same information but including targets of a marketing campaign with another label indicating whether the recipient became a customer.
4. **Mailout test** is virtually same as above but for testing purposes only.

Most of the features are numerical (either continuous or ordinal), though several are categorical. It's an extensive source covering a broad variety of factors including demographic data like age and gender, living conditions, political convictions, economic standing, and purchase behaviors.

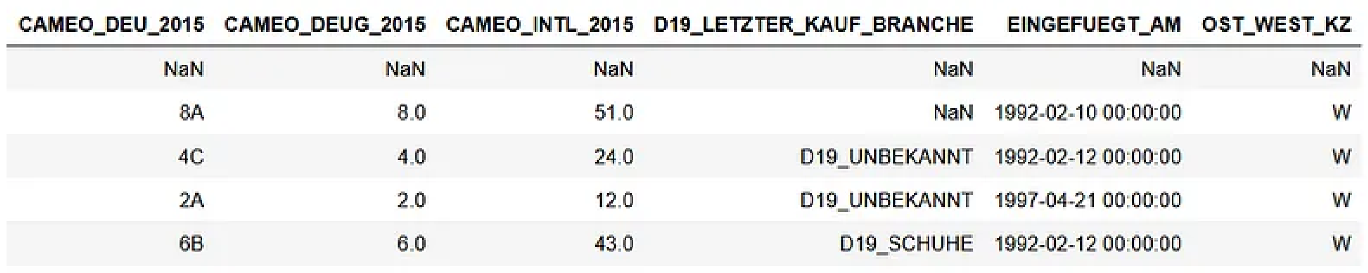
All the data cleaning has been done using the azdias dataset since it's the most comprehensive one. Later, the same steps were reapplied to the remaining files.

Part 1. Data

There are several things we need to tackle before clustering the population, starting with handling data types and redundant columns. We can start this task by identifying features with mixed data types. There are 6 such columns which can be spotted right away:



```
float64    267
int64       94
object       6
```



CAMEO_DEU_2015	CAMEO_DEUG_2015	CAMEO_INTL_2015	D19_LETZTER_KAUF_BRANCHE	EINGEFUEGT_AM	OST_WEST_KZ
NaN	NaN	NaN	NaN	NaN	NaN
8A	8.0	51.0	NaN	1992-02-10 00:00:00	W
4C	4.0	24.0	D19_UNBEKANNT	1992-02-12 00:00:00	W
2A	2.0	12.0	D19_UNBEKANNT	1997-04-21 00:00:00	W
6B	6.0	43.0	D19_SCHUHE	1992-02-12 00:00:00	W

The three **CAMEO** columns represent an already established stratification of the population by social and economic standing. Since they are partially overlapping, I chose to keep **CAMEO_INTL_2015** as it combines family status with economic class which seems to impact purchasing decisions the most. Don't get fooled, this is not a numerical feature and we will have to create dummy variables for each unique value.

D19_LETZTER_KAUF_BRANCHE was not explained anywhere in the provided files but "letzter kauf branche" means "last purchase industry" in German. We will create dummy binary variables for this feature as well to capture the public's preferences.

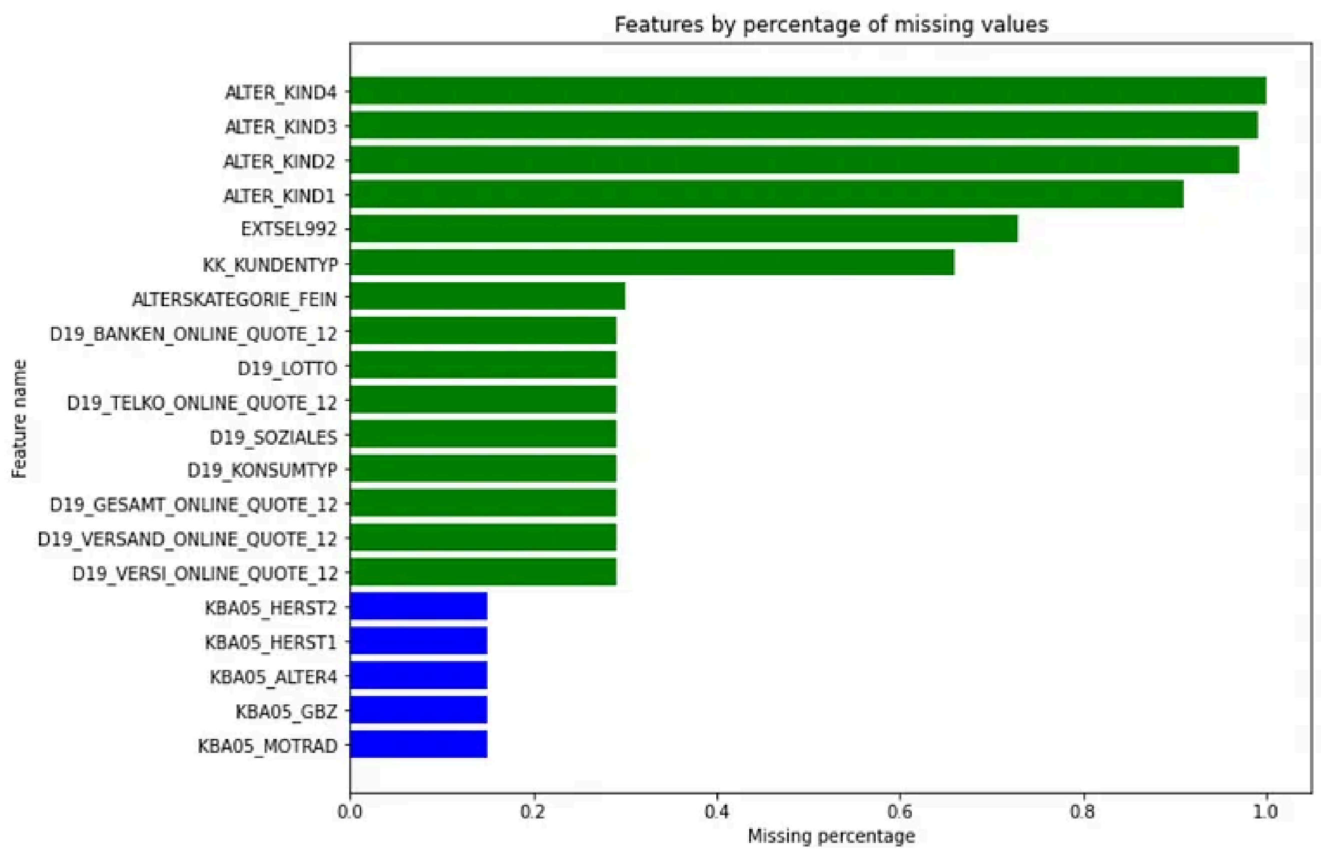
PRAEGENDE_JUGENDJAHRE despite appearing to be ordinal is actually categorical. At first glance, corresponding values seem to represent generations by decade but after further investigation, it turns out they also mix different political orientations.

EINGEFUEGT_AM seems to be metadata info on when the record has been inserted and thus is redundant.

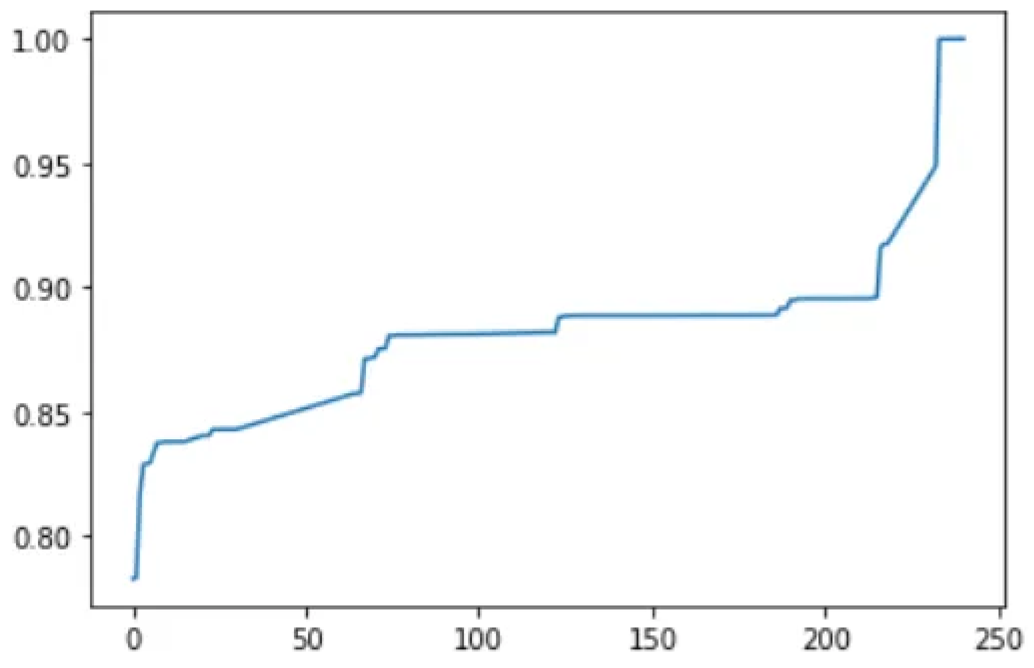
Finally, **OST_WEST_KZ** denotes the republic of origin, ost being short for osten (east), so either DDR or FDR. It has to be mapped into binary values.

We will also drop the **LNR** column as it's a unique identifier field and **ANREDE_KZ** denoting gender.

Another important decision to make is which features are carrying enough information, i.e. what is the percentage of missing values. Considering the large number of features and relatively low number of nulls, we will drop any feature that has less than 85% of values. There are 15 such columns (in green).



Next, we need to look at entries with missing data to see if they contain enough information for our purpose. To that end, we plotted a cumulative sum of missing values per feature to visually check for any breaking point.



Around 83% of instances have 6 or less missing values. The remaining 17% have between 7 and 240 missing values. The cutoff of 6 missing features seems to be reasonable so we will drop any rows exceeding this threshold.

We're almost ready to handle the remaining missing values but before deciding on how to resolve this issue, we need to distinguish between binary and numerical features.

A binary feature would be any column that only contains two unique values. Apart from dummy variables we've just created there are some features with 'flag' in their names and a couple of other columns that appear to be binary. These cannot be treated like other numerical ones when it comes to scaling and filling in null values. Unfortunately, two of these columns contain some nulls.

```
HH_DELTA_FLAG    31186
KONSUMZELLE          4
```

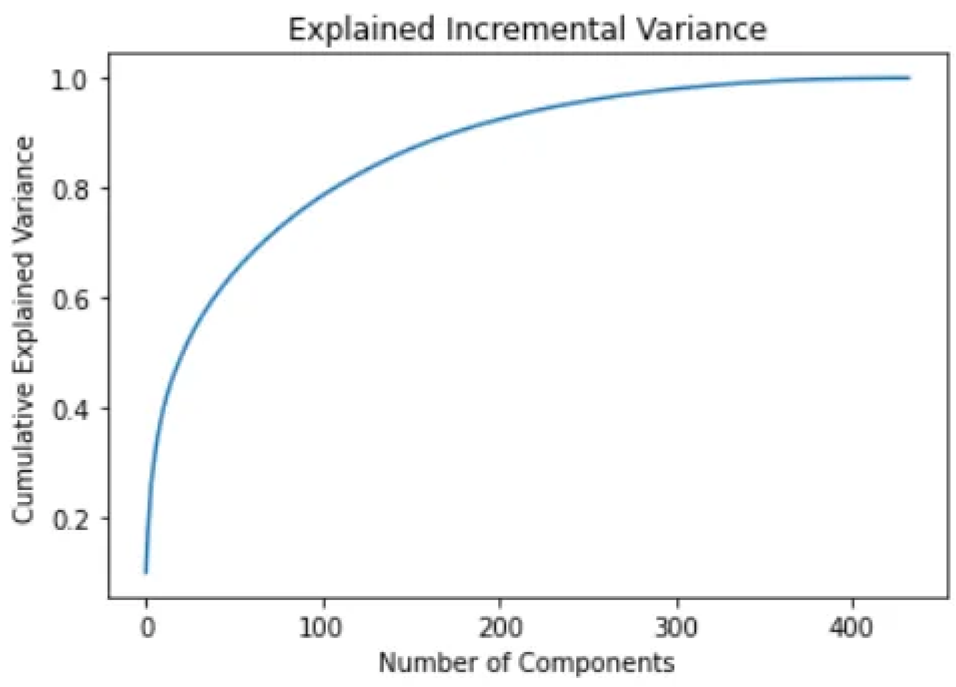
We will fill them with modes as the most prevalent outcome should be typical for demographic data. The rest of the columns with missing data can be filled with means, so we did just that.

The final step will be scaling data using the Standard Scaling method. It may be worth experimenting with other methods like Min-Max scaler in the future though.

Part 2. Customer segmentation

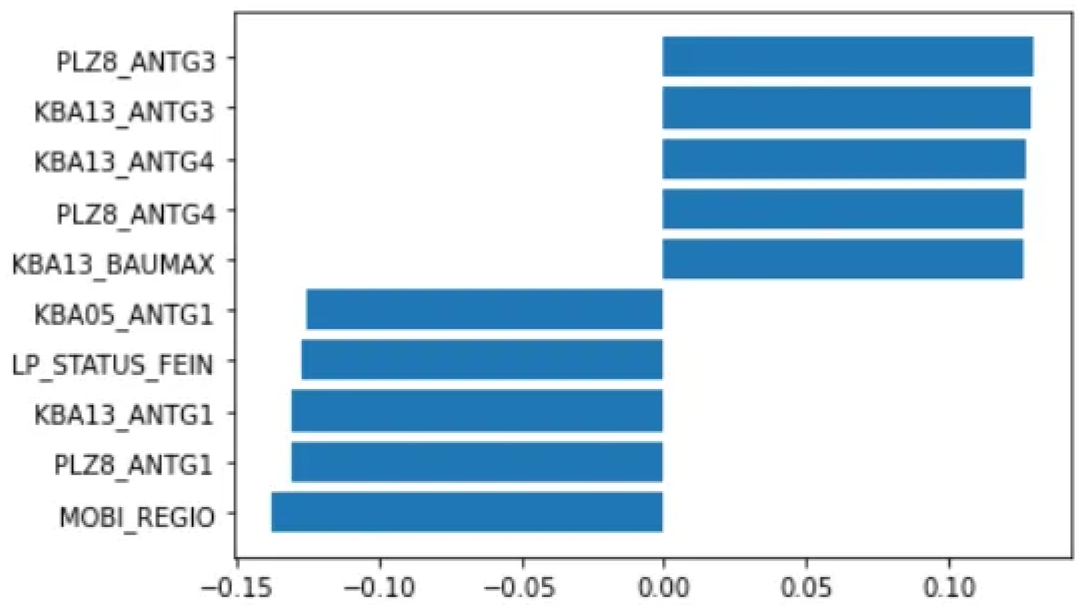
Often, when constructing a customer segmentation framework, it's imperative to ascertain a particular level of clarity so that business stakeholders can easily understand the division and marketing teams can create personas. In this instance, however, the ultimate goal of segmentation is the task at hand — predictions for the mailing campaign. This, combined with the multi-dimension dataset convinced us to perform dimensionality reduction using the PCA method.

After fitting PCA on the dataset we plotted the cumulative variance explained by including more components to decide how many of them we need.



Out of 433 features, 235 components explain 95% of the variance in the dataset and this is the threshold we're going to assume.

Looking at the output we can get a better understanding of what differentiates our clusters. The first component (explaining 10% of variance) is mostly determined by the car the person owns (KBA) and social status (LP_STATUS_FEIN). The car owned is an interesting addition but we may assume that the car owned by a person is correlated with income. After all, socioeconomic factors are usually the leading factors for market segmentation. Similarly, MOBI_REGIO, which indicates mobility, should be correlated with car usage.



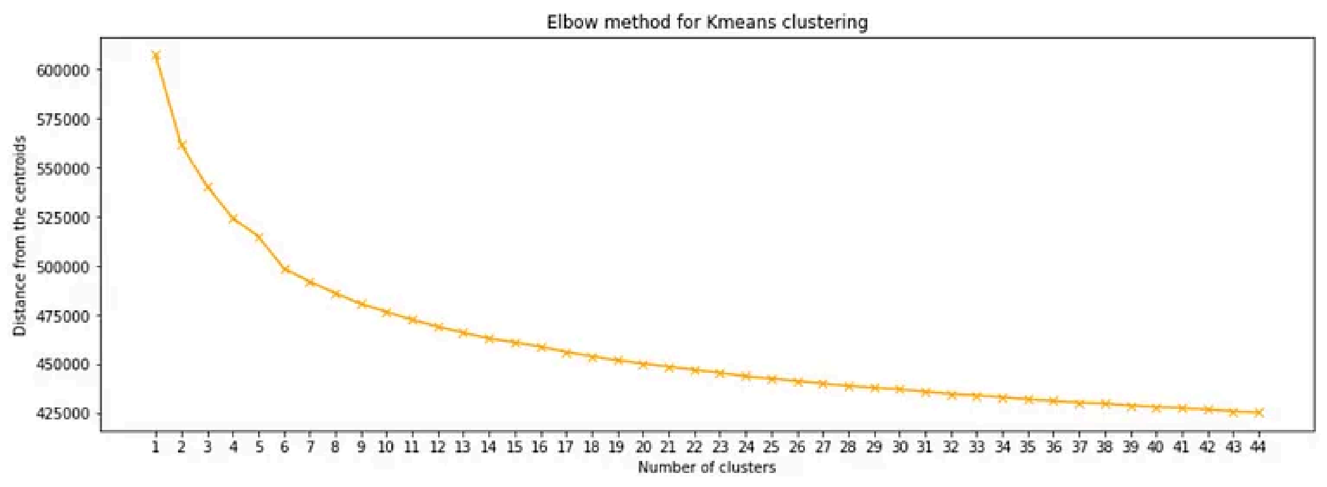
Let's try segmenting the population.

K-means clustering is an unsupervised ML algorithm which makes it perfect when we don't exactly know what clusters data points should end up in. It's by far the most popular algorithm for unsupervised clustering problems. It's also proper when following the MECE rule (mutually exclusive — collectively exhaustive) which means that each customer needs to belong to exactly one segment. If, on the other hand, we would like to try to make customers fit into many different segments, we could use the Hidden Markov Model (HMM).

Alternatively, we could also use Agglomerative Hierarchical Clustering which iteratively merges clusters by their similarity, starting with each data point being its cluster. Its advantages are that it provides granular information on any data point and it is not sensitive to outliers, which gives it a great descriptive utility. We might not want to do it here though cause it's slow and not optimal for predicting new instances.

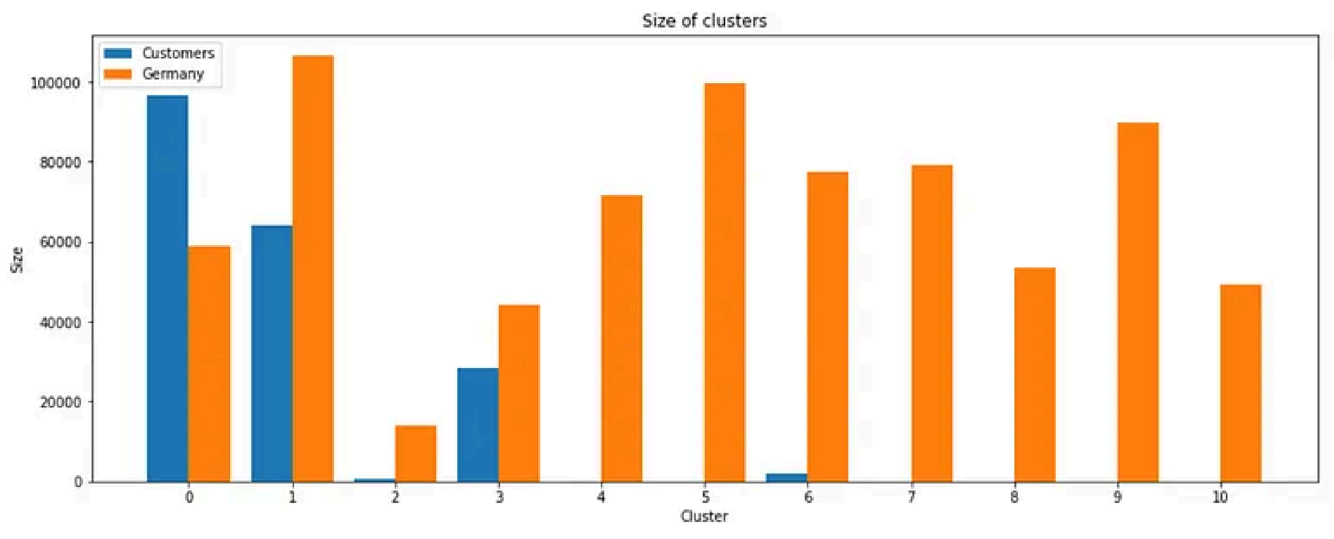
Let's stick to K-means clustering.

To decide on how many clusters we even want, we need to employ the elbow method. It involves plotting the explained variation as a function of the number of clusters and picking the elbow of the curve to spot the sweet spot between relatively low numbers of clusters and possibly the biggest impact. The function measures a drop in inertia, being a measure of how internally coherent clusters are. The inertia formula quantifies how close the data points are to their centroid. The lower the value, the denser the cluster.

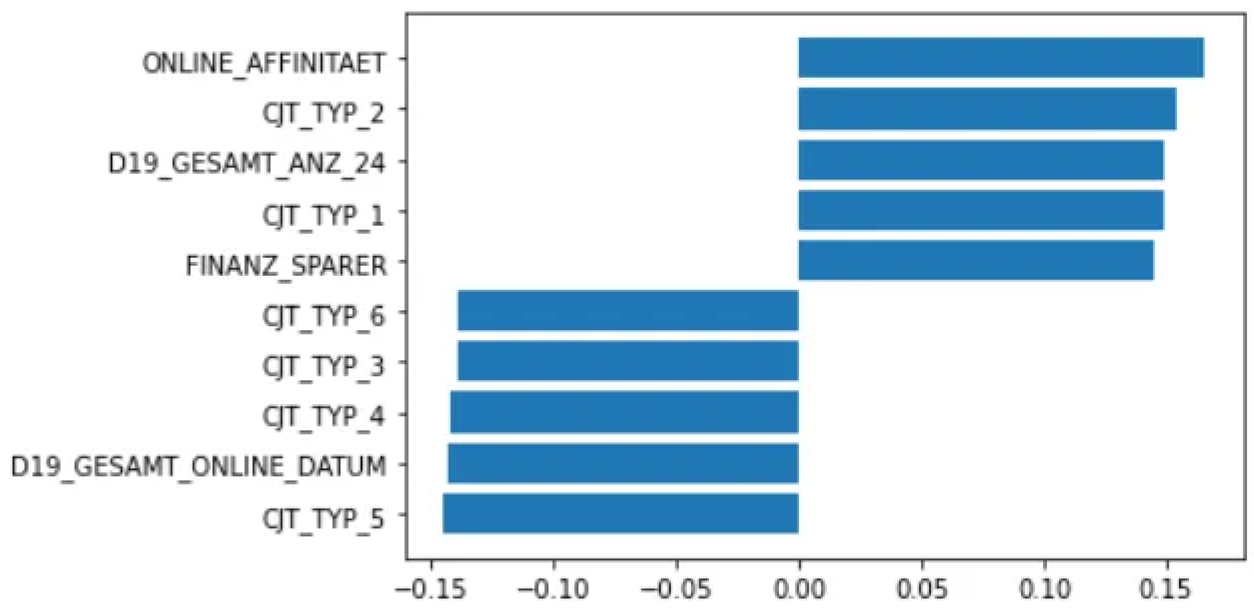


We see a sharp decline around 6 clusters and then a steady downward slope. In my opinion, selecting the proper number of clusters tends to be more of an art than science depending on the business needs. After careful consideration, I've decided to go with 11 clusters.

As mentioned before, we delve into unsupervised ML techniques because we don't know what segments we want to have and how to name them. That's why we can only call them by their ids. As it turns out, the company is mostly focused on three of these segments as its traditional customers.



We can look into cluster centroids to analyze the leading components that influenced them.



Looking at the most overrepresented cluster (the most prevalent customer group), we see a positive impact of customer journey types 1 and 2 — traditionalists and minimalists. This is in line with the order-mail company being somewhat of a traditional business model. These customers also tend to be less interested in saving money (indicated by the FRANZ_SPARER feature).

Part 3. Predicting response (conversion) rate

We are almost ready to choose a training model. But before we jump into it, let's talk about a serious issue with many business applications — **the dataset imbalance.**

It so happens that the most interesting class — being the converted customer — is the severely underrepresented one. Any business would love to have major market coverage but the reality is that we need to fight for customers. In our instance, the class that is most relevant accounts for less than 2% of all data entries. This makes it difficult to train a model as simply hard-coding the majority class for every prediction would result in an accuracy of over 98%!

```
0    0.987617
1    0.012383
Name: RESPONSE, dtype: float64
```

There are three ways in which we're going to mitigate this issue.

- **resampling techniques** (by undersampling/oversampling the majority class)
- **evaluation metrics** (selecting scoring metrics that are not overly sensitive to this problem)
- **cost-sensitive training** (using balanced class weights where possible)

The most crucial part is to choose between resampling methods. To make it easier, the **undersampling** approach has the following effects (converse to oversampling):

1. (advantage) we will make the training process faster and less resource-intensive
2. (advantage) we will reduce the risk of overfitting
3. (advantage) we might be reducing noise from irrelevant instances of the majority class
4. (disadvantage) we might be losing some important information since the imbalance is major and we will discard many instances

Unfortunately, undersampling results in a small dataset that is not sufficient to train the model (only 1064 entries). That's why I've decided to oversample the minority class instead (84860 records at the end).

Scoring metric

To establish the best model, it's important to select a **proper scoring metric**. It's always a matter of some trade-offs as no single metric can tell us anything about the problem. In this instance, we could consider several of them, for instance:

1. **Precision-recall curve** plots the proportion of correctly predicted positives (precision) against the proportion of all positives predicted correctly (recall). Since it focuses on the performance of an imbalanced class, it would be suitable for our case. The scoring metrics in the average precision that summarizes the precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight.
2. **The ROC curve** plots the proportion of all positives predicted correctly (recall) against the false positive rate at various threshold settings. The final metric is the **area under the ROC curve**. It might not be the best for highly imbalanced datasets though.
3. **The F1 score** measures the balance between precision and recall with the following formula: 2 * (precision * recall) / (precision + recall)
   It's an improvement from simple accuracy but treats precision and recall equally.

We chose the ROC AUC metric since we've already addressed the imbalance issue and it's a good measure for classification problems.

After trying out several models, we can print the report with their outcomes.

```
----------------------------------------------------------------------------
----------------------  Training LogisticRegression  ----------------------
----------------------------------------------------------------------------

Area under curve: 0.8159730179593467
best params: {}
Function train_clf took 1198.4 seconds to execute
LogisticRegression(class_weight='balanced', max_iter=1000, solver='liblinear')

----------------------------------------------------------------------------
----------------------------------------------------------------------------
--------------------  Training RandomForestClassifier  --------------------
----------------------------------------------------------------------------

Area under curve: 0.9936583839215235
best params: {}
Function train_clf took 174.5 seconds to execute
RandomForestClassifier(class_weight='balanced')

----------------------------------------------------------------------------
----------------------------------------------------------------------------
------------------  Training GradientBoostingClassifier  ------------------
----------------------------------------------------------------------------

Area under curve: 0.9046647723393967
best params: {}
Function train_clf took 671.8 seconds to execute
GradientBoostingClassifier()

----------------------------------------------------------------------------
----------------------------------------------------------------------------
-------------------------  Training XGBClassifier  -------------------------
----------------------------------------------------------------------------

Area under curve: 0.9930608351396477
best params: {}
Function train_clf took 30.2 seconds to execute
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)

----------------------------------------------------------------------------
```

Random Forest classifier and XGB classifier provide us with shockingly high ROC AUC scores strongly suggesting overfitting on the dataset. These algorithms will not generalize well on a new dataset. This is why we picked the Gradient Boosting classifier.

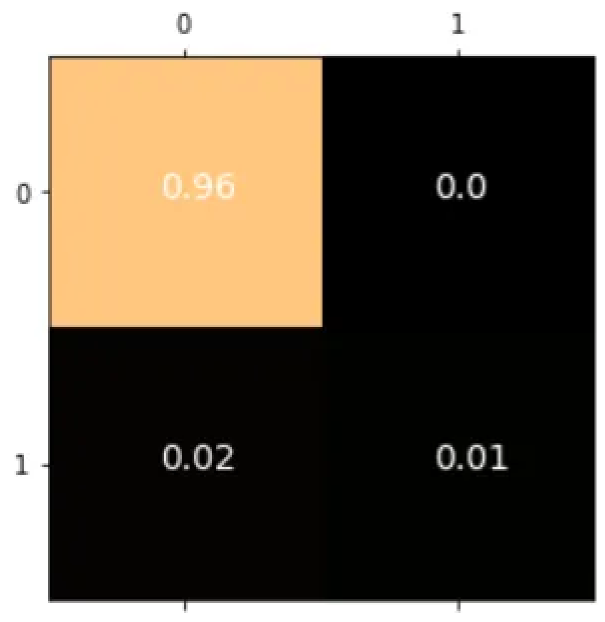
It's time to tune the model's parameters using GridSearchCV to improve the performance. The final model will include the following parameters.

```
Area under curve: 0.993609497757852
best params: {'learning_rate': 0.2, 'max_depth': 7, 'n_estimators': 200}
Function train_clf took 16003.2 seconds to execute
GradientBoostingClassifier(learning_rate=0.2, max_depth=7, n_estimators=200)
```

Hypertuning might've caused the model to overfit though, so let's try our model on the original training set, without any resampling done.

```
array([[41457,    44],
       [  973,   488]],
```

Thankfully, the model did not predict all instances for the majority class. While 96% of predictions were assigned to it, it's pretty close to the actual dataset.



Interestingly, the F1 score is pretty low due to imbalance, since recall might not be the best.

```
[illegible]
```

An important takeaway is that for future usage, we might want to pay more attention to recall as we can afford to address more customers with the marketing campaign (which should generate relatively low costs), whereas we don't want to miss potential customers.

Summary

The project addresses the critical questions of comparing the company's customer base with the general population of Germany and evaluating the conversion level for a specific marketing campaign. Through data cleaning, handling of mixed data types, and thoughtful feature selection, the groundwork was laid for in-depth analysis. The application of PCA for dimensionality reduction and K-means clustering for customer segmentation provided valuable insights into the company's customer demographics and their preferences.

The challenge of dataset imbalance, particularly prevalent in predicting conversion rates, was tackled using a combination of resampling techniques, appropriate evaluation metrics, and cost-sensitive training. Following a process of model comparison and parameter tuning, the Gradient Boosting classifier has been selected.

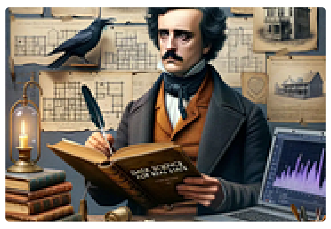
For future improvements, several avenues can be explored. First, experimenting with alternative machine learning models and ensemble methods could potentially enhance predictive accuracy and generalizability, especially favoring recall. Additionally, a thorough review of the demographic and behavioral data might reveal deeper insights into customer preferences and motivations and ultimately might result in discarding features introducing noise into data. A more granular tuning process could be employed with more processing resources available. Finally, implementing a continuous feedback loop where the model's predictions are regularly updated with new campaign data could also refine its predictive capabilities over time.

Data Science · Marketing · Cluster Analysis · Pca Analysis · Imbalanced Data

Written by Jakub Below

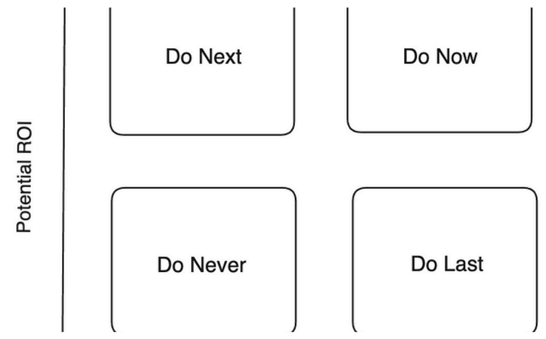
0 Followers

Follow



See all from Jakub Below

Recommended from Medium



The Learning Project

Customer Data Platform 101: Segments

· 7 min read · Feb 23, 2024

92

OWOX

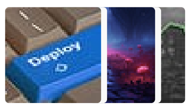
RFM Analysis: Learn more about your customers and RFM...

Get expert insights into RFM analysis and discover how to leverage this powerful tool t...

14 min read · 5 days ago

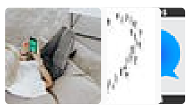
114 1

Lists



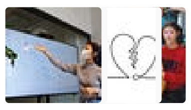
Predictive Modeling w/ Python

20 stories · 1001 saves



Branding

34 stories · 174 saves



Modern Marketing

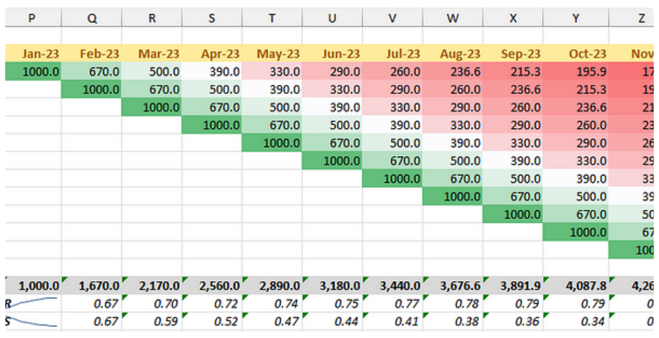
94 stories · 483 saves

Practical Guides to Machine Learning

10 stories · 1199 saves

Malar Raju

Factors need to be analyzed in Customer Segmentation and...



P	Q	R	S	T	U	V	W	X	Y	Z
Jan-23	Feb-23	Mar-23	Apr-23	May-23	Jun-23	Jul-23	Aug-23	Sep-23	Oct-23	Nov
1000.0	670.0	500.0	390.0	330.0	290.0	260.0	236.6	215.3	195.9	17
	1000.0	670.0	500.0	390.0	330.0	290.0	260.0	236.6	215.3	19
		1000.0	670.0	500.0	390.0	330.0	290.0	260.0	236.6	21
			1000.0	670.0	500.0	390.0	330.0	290.0	260.0	23
				1000.0	670.0	500.0	390.0	330.0	290.0	26
					1000.0	670.0	500.0	390.0	330.0	29
						1000.0	670.0	500.0	390.0	33
							1000.0	670.0	500.0	39
								1000.0	670.0	50
									1000.0	67
										100
1,000.0	1,670.0	2,170.0	2,560.0	2,890.0	3,180.0	3,440.0	3,676.6	3,891.9	4,087.8	4,26
R	0.67	0.70	0.72	0.74	0.75	0.77	0.78	0.79	0.79	0
S	0.67	0.59	0.52	0.47	0.44	0.41	0.38	0.36	0.34	0

Paul Levchuk

Blended retention vs Cohort retention

Customer Segmentation and Churn Analysis involve examining various factors to...

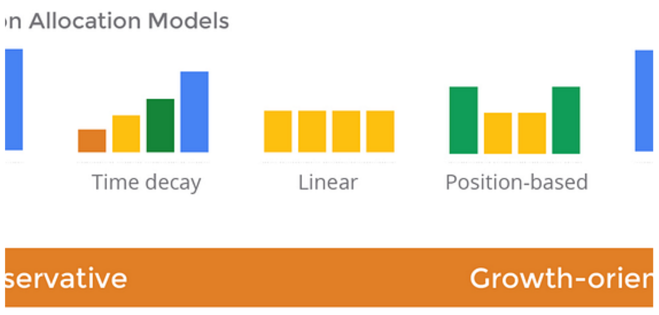
✦ · 13 min read · Jan 29, 2024

51

Never use the blended calculation to figure out user retention

4 min read · 5 days ago

29



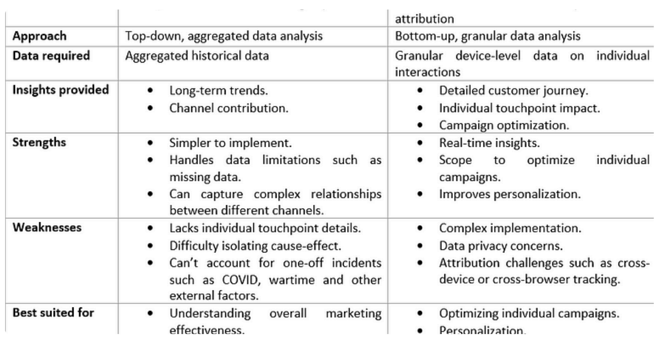
bhaskar ammu

A Machine Learning Approach to Marketing Attribution

LIMITLESS INFORMATION!

10 min read · Jan 29, 2024

75



		attribution
Approach	Top-down, aggregated data analysis	Bottom-up, granular data analysis
Data required	Aggregated historical data	Granular device-level data on individual interactions
Insights provided	• Long-term trends. • Channel contribution.	• Detailed customer journey. • Individual touchpoint impact. • Campaign optimization.
Strengths	• Simpler to implement. • Handles data limitations such as missing data. • Can capture complex relationships between different channels.	• Real-time insights. • Scope to optimize individual campaigns. • Improves personalization.
Weaknesses	• Lacks individual touchpoint details. • Difficulty isolating cause-effect. • Can't account for one-off incidents such as COVID, wartime and other external factors.	• Complex implementation. • Data privacy concerns. • Attribution challenges such as cross-device or cross-browser tracking.
Best suited for	• Understanding overall marketing effectiveness.	• Optimizing individual campaigns. • Personalization.

Nanduru Santosh

Decoding Marketing Analytics: Media Mix Model vs. Multi-Touch...

We come to the culmination of my series on Media mix modeling(MMM). We began this...

3 min read · Feb 4, 2024

132

See more recommendations

Help Status About Careers Blog Privacy Terms Text to speech Teams