Arvato Bartelsman Customer Segmentation Report

Sebastian Koenig - Udacity Capstone - January 2020

Summary

This report includes both customer segmentation using Kmeans clustering and the use of a supervised XGBClassifier to identify those individuals who are most likely to be responsive to an advertisement campaign.

Data exploration revealed that some steps had to be taken to reduce noise. This included ensuring uniform encoding of missing data, removal of sections with unusually large amounts of missing values, and the selection of useful features. Kmeans clustering pointed towards a target segment roughly 30% of the general population with similar attributes as 60% of the clients customer base.

The training set for the supervised learning task proved highly imbalanced, I therefore resampled data with SMOTEENN (automatic over and undersampling). The XGBClassifier delivered an AUC of 0.996+ on the resampled training data but only about 0.627 on the unaltered development set. Hyperparameter tuning seemed the next sensible step. After testing some values manually I found an XGBClassifier that performed only marginally better which revealed that feature engineering would be more crucial. I first discovered that our oversampling actually caused the algorithm to overfit, XGB is remarkably resilient to imbalanced data and my attempt to train it on balanced data meant that it produced false positives.

I then recleaned the Training data, leaving a larger selection of remaining features and achieved significantly better results. I finally decided to run a randomized hyperparameter search to see if a better selection of hyperparameters than those intuitively selected could be found.

Data Exploration and Feature Selection

The data set consisted of just over 800,000 entries with a variety of features. Looking at the explanation of the features provided in an excel sheet, I realized that a significant portion of those features dealt primarily with the prevalence of makes and models of cars owned by the consumer or prevalent in their local area. This seemed of little relevance as our client is a mail order company. Obviously this data could give some indirect insight into the socio economic conditions of the household, but I do not believe it to be significant enough given the inclusion of far more direct metrics in the dataset. Data concerned with cars was thus removed from consideration. I also removed any other entries which was not explained in the excel sheet and thus not useful for the segmentation report.

Customer Segmentation

Data Cleaning

Data exploration revealed that quite a few of the values in our data actually encode missing information. Ensuring that missing information is uniformly encoded was vital to the project. Therefore I created a checkbook identifying which values encode missing information for each feature, then iterated over the whole dataset to uniformly encode all missing information as NaN.

In a second step we analyzed which features had large gaps and removed them from consideration. We then performed the same elimination for those individual entries which were highly incomplete. To ensure that both the general population and customer data would be processed the same way we used the remaining features of the general population data as a filter for the customer data.

Finally we looked at the remaining features to figure out which ones represented non-sequential categorical data. Sequential categorical data or binary data that is numerically encoded can be used in learning algorithms, but non-sequential categorical data, even when encoded numerically, needs to be reencoded because the algorithm will treat numbers close in value as related even if the underlying categories have no relation to one another. A list of these features was created and one-hot encoding applied, generating a new binary feature for each category. Finally we defined a cleaning function to automate the cleaning process for all other datasets.

Scaling and Dimensionality Reduction

I used a simple imputer to replace all remaining missing values in the data set with the median value for the feature then re-scaled it to ensure that all values fall into the same range. As a final step in the preparation I used principal component analysis to reduce the dimensionality of the dataset without losing significant variance.
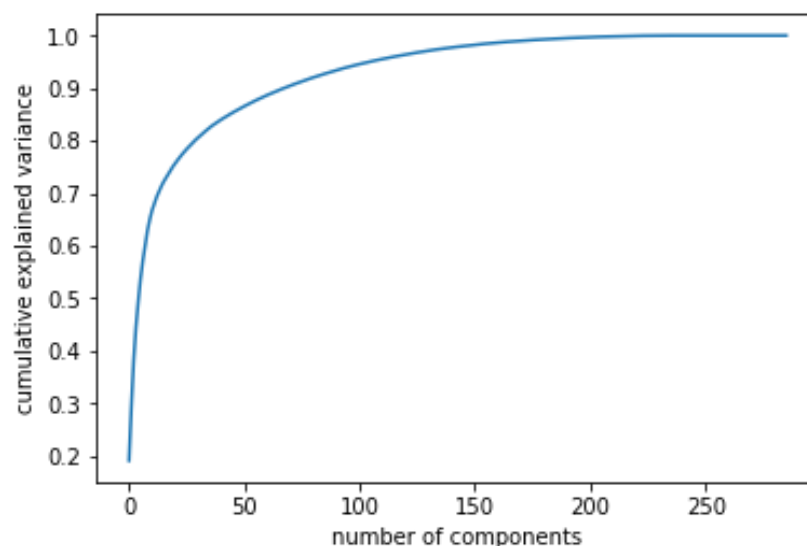
As visible in the graph above about 200 features explain close to 100% of the variance in the data, thus applied PCA dimensionality reduction to include only the most significant 100 features.

Kmeans Clustering

To determine how many clusters our data should be grouped into we used MiniBatchKmeans to iterate over each possibility between two and twenty clusters, looking at the sum of squared error as a proxy for the relative intra-cluster distance between data points.
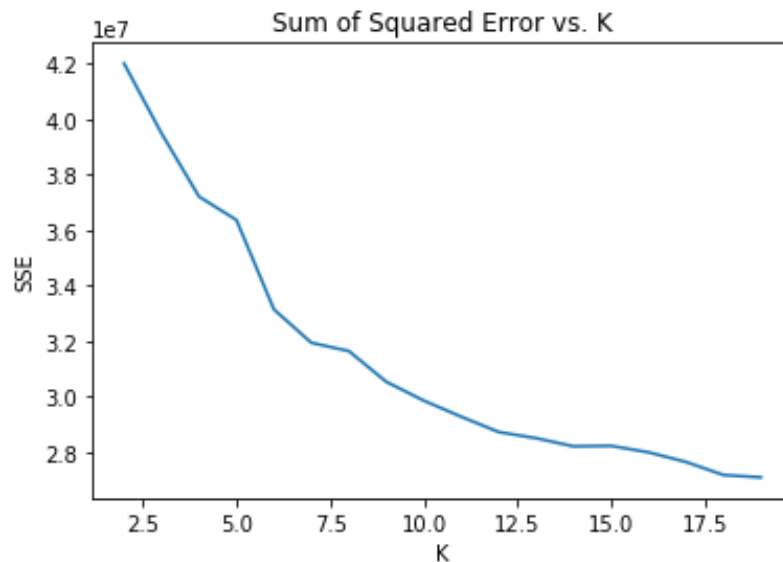
It appeared sensible to try both 7 and 14 clusters as reduction in intra-cluster distance dropped off significantly after those two values. Ultimately I elected to use only 7 clusters as we found a cluster that accounted for the majority of the customer data while providing us a target segment of the general population, and the more granular 14 clusters did not provide a smaller window target window resembling a similar or larger portion of the customer data.

Target Segment Analysis

After clustering I extracted the portion of the general population data belonging to the target cluster (3) and saved it indexed by LNR which serves as a unique identifier for every data point. Using that data in combination with the value explanation spreadsheet we can get some general insight into what type of individual would be most likely to become a new customer.
The following is a table giving the average values for our target group along a selected list of features:

Affluence	Established Middle Class
Age	45-60
Household size	2
Movement	Mainstream
Location	40-50km outside of urban centers, West Germany

This appears to be people around age 50, usually two person households living on average 40 to 50 km outside of the nearest urban center. They tend to have slightly above average purchasing power.

Supervised Learning Model

Knowing that Extreme Gradient Boosting (XGB) algorithms have performed well on similar problems in the past, I decided to use a basic XGB as the baseline and see how much performance can be improved by a combination of intuitive manual and randomized algorithmic hyperparameter tuning.

Addressing Data Imbalance

The primary difficulty of this task is the extreme imbalance of the dataset. Of the almost 43,000 data points only 532 are positive responses. Training data this imbalanced incentivises many algorithms to label everything uniformly as the dominant class. To address this I resampled the data using a combination of synthetic over- and undersampling to create a more balanced data set.

Baseline XGBClassifier

To establish a baseline I trained an XGBClassifier on the resampled data and evaluated it on a development set roughly 10% of the original training data provided, which was not resampled. XGB fit the training data well (AUC 0.996+) but performed less well on the evaluation set (AUC 0.627). This algorithm was clearly overfitting the training data. I decided to first tune hyperparameters manually to see if the issue was one of tuning or feature engeneering.

Intuitive Hyperparameter Tuning

I have trained some XGBClassifiers for past projects and decided to use a few combinations of hyperparameters that served me well on similar datasets in the past.

Tuned XGB Performance

Our tuned XGB performed a lot better on the un-resampled training data (AUC ~0.681 to ~0.876), but didn't improve on the test data (AUC ~0.62). We were able to improve fit without further over fitting but didn't manage to reduce overfitting either. Training the optimized model on imbalanced data reduced overfitting a little, but not significantly enough (validation set AUC ~0.64).
At this point additional feature engineering appeared to be the only option for further progress. We likely removed features important to model performance in cleaning. While our cleaning approach served us well for the Kmeans clustering it is possible that some of the features not explained in the DIAS Attributes Values 2017 excel file are important to model performance.

Reexamined Features

I decided to reclean the data manually, leaving a lot more features, specifically those not explained in the DIAS Attributes Values 2017 excel sheet. They were not useful for segmentation as we would not have been able to explain what they signified even if they had produced better defined clusters, but they might improve our ability to predict customer responses. Indeed even our model optimized for the original feature selection performed significantly better (Validation Set AUC of 0.8+). I decided to test sklearn's RandomSearchCV algorithm to tune hyperparameters hoping to produce a final optimized model for our new feature set.

Kaggle Competition

Finally we cleaned the test data to conform to our revised feature engineering, imputed missing values and applied the same scaling as we did for the training data, then had our final XGBClassifier predict label probabilities, rather than actual labels as required by the Kaggle

competition. Unfortunately we did not achieve the AUC we hoped and also failed to beat Google AutoML which achieves AUCs of 0.802+. We only got up to AUC 0.78791 with our XGBClassifier. In other words, the most sensible workflow for this project appears to be using a state of the art pre-build classifier like Google AutoML rather than to create one from scratch. I can see that maybe with a lot of time or luck it could be possible to engineer features further to improve performance, but this would represent a lot of effort for marginal gains.

Model	XGBClassifier	TunedXGB	FinalXGB	Google AutoML
Kaggle Competition AUC	0.598	0.621	0.788	0.802