Arvato Customer Segmentation Report

Oliver Farren

August 26, 2020

Abstract

Bertelsmann Arvato Analytics are consulting a Mail Order company in better understanding their customer demographic and building a future targeted marketing campaign. To this end, using datasets supplied by Arvato, key findings that describe the Customer population were ascertained to aid building a report for the Mail Order Company. Finally, a supervised model was built using an XGBoost Classifier to predict the likelihood of customer conversion to aid better targeting. The final model had a Receiver Operating Characteristics Area Under Curve score of 0.848 and at time of publishing, is the winning model in the Udacity-Arvato Kaggle competition, with the next best solution having a score of 0.811.

1 Project Overview

This project was one of the proposed capstone projects for the Udacity Machine Learning Nanodegree. The overall goal was to investigate whether we could understand the customer demographic of a German mail order company in order to inform a better targeted marketing campaign. This is a customer segmentation task consisting of data science methods and machine learning methods to deliver business intelligence on behalf of Bertelsmann Arvato Analytics who are consulting for the mail order company

This project was of personal interest to me due to the widespread application of customer segmentation and the technical application of Data Science methods and use of both unsupervised and supervised machine learning models.

1.1 Project Brief

This is the original project brief supplied by Udacity:

> In this project, you will analyze demographics data for customers of a mail-order sales company in Germany, comparing it against demographics information for the general population. You'll use unsupervised learning techniques to perform customer segmentation, identifying the parts of the population that best describe the core customer base of the company. Then, you'll apply what you've learned on a third dataset with demographics information for targets of a marketing

campaign for the company, and use a model to predict which individuals are most likely to convert into becoming customers for the company. The data that you will use has been provided by our partners at Bertelsmann Arvato Analytics, and represents a real-life data science task.

1.2 Provided Data

4 datasets were provided by Bertelsmann Arvato Analytics through Udacity for this project.

1. Demographics data for the general population of Germany: 891 211 persons (rows) x 366 features (columns)
2. Demographics data for customers of a mail-order company; 191 652 persons (rows) x 369 features (columns)
3. Demographics data for individuals who were targets of a marketing campaign; 42 982 persons (rows) x 367 (columns)
4. Demographics data for individuals who were targets of a marketing campaign; 42 833 persons x 366 (columns)

In addition, two supporting documents were provided that described the encoding and meaning of features across datasets:

1. Dias Information Levels Attributes - Contains descriptive information for each feature
2. DIAS Attribute Values - Contains detailed information about the meaning of the different value levels for each feature in the 4 datasets

1.3 Domain Background

Customer Segmentation is the practice of dividing a customer base into groups that are similar based on a set of features for that customer base. In the case of Bertelsmann Arvato Analytics, the intent is to understand what distinguishes customers of the mail order company from the general population based on a set of features. If they can be segmented from the general population then we can start to understand what characterises customers for better ad targeting and how to better select individual for those targeted ads.

This targeted marketing can save significantly on costs and effort over a general marketing campaign and can inform more tailored marketing products with a higher customer conversion rate.

1.4 Problem Statement

The problem statement is composed of two parts:

1. Can we identify a demographic in our current customers with respect to the general German to inform better marketing products?
2. Can we reduce the number of people targeted and still retain a high proportion of our target demographic?

1.5 Metric

The latter part of the problem statement will be evaluated in a Kaggle competition using a dataset with witheld labels. Area Under Curve (AUC) will be the performance metric used to evaluate the model.

The AUC performance metric is appropriate for a highly imbalanced labeled dataset where the number of positive responses is significantly smaller than the number of negative responses. This metric is able to evaluate the quality of our model when, despite being a good predictor, we would still expect a fairly small conversion rate from our target demographic.

1.6 Project Outline

The project is broken into three major sections:

1. Data Analysis and Cleaning
2. Creation of an unsupervised learning model that can segment customers and help identify key characteristics of the customers
3. Creation of a Supervised learning model that can give a prediction between 0 and 1 of the likelihood of an individual converting to a customer.

All of this took place in Jupyter Notebooks on a local machine using local resources. While cloud resources were explored during the Udacity Nanodegree it was felt the financial cost to utilising these resources did not justify their use.

2 Cleaning the Dataset

This section describes the steps I took in preparing the dataset ready for use in training an unsupervised learning model.

2.1 Initial discussion of my plan

The main question I held in mind when approach data cleaning was: *What am I trying to achieve by cleaning this data?*

Working backwards, I want to understand how customer demographic is distinct from a general population so I'm going to build an unsupervised model to achieve this.

I am going to build this unsupervised model using KMeans clustering which should be able to cluster my dataset and I should be able to see that a number of clusters significantly represents the customer population.

In order to build this unsupervised model I need to reduce the dimensionality of the original dataset. The higher the number of dimensions, the less able the KMeans model is able to cluster features. Known as the 'curse of dimensionality'. In low dimensional datasets, the closest points tend to be much closer than average, but two points are only close if they're close in every dimension. So when there are lots of dimensions, it's more likely that the closest points aren't on average much closer than average.

I can use Principle Component Analysis (PCA) to reduce the dimensionality. PCA generates a new dimensional space where each new dimension represents a principle component that explain the overall variance of the dataset in decreasing amounts. Using this method, a high proportion of the dataset variance could be explained by a subset of it's cumulative principle components and the remaining low variance components can be discarded.

PCA can only be used on a fully populated, ordered, numerical dataset that has been standard scaled: 0 mean and unit variance.

This drew me to conclude my initial data cleaning strategy:

1. Drop features with a high number of missing values
2. Convert categorical features to numerical features
3. Drop sparsely populated rows
4. Impute any remaining missing values
5. Scale all features using Sklearn Standard Scaler
6. Use Principle Component Analysis to reduce the number of dimensions and align features by variance

When first presented with the large population dataset consisting of 900,000 individuals and 366 features. My initial impression was that this many features meant that individual feature inspection was not a good strategy and that a semantic understanding of the features was not needed. Instead, I should move forward in an abstract 'data driven' sense.

I shortly discovered this strategy had some major shortcomings however. By not understanding the data or referencing the supplied DIAS information sheets, I was creating very poor numerical features with high amounts of distortion which would not be useful for building my unsupervised learning model.

The better approach was to first gain an understanding of each feature individually and understand whether the feature would be a clean, numerical feature that I should use in PCA. This is what I ended up doing.

2.2 Analysing the Datasets

2.2.1 Understanding the features

To begin with, I planned on reading through the supplied DIAS sheets in order to understand the features and outlined some key considerations to hold while reading:

- Are numerical features ranked?
- Are categorical features ordinal or low cardinality?
- Does anything else stand out about these features that might be worth keeping in mind?

One key observation I made was that many of the numerical features encoded an 'unknown' as -1 or 0. Having prematurely explored missing values in the dataset in the form of no entry or NaN this seemed to be very confusing. Why would a feature have 'uknown'=0 and NaN? Was this significant? No information on this point was supplied by Bertelsmann Arvato Analytics. My current working theory is that the provided dataset was an outer join of SQL databases supplied by different partners. This theory is supported by the apparent naming structure that seems to indicate that groups of features stem from different sources.

Because I wanted my features to be ranked, categorising 'unknown' as a value would distort the scaling and conversion to principle components so all of these known 'unknown' and 'missing' values would need to be converted to NaN.

I also noted a few perculiar features while going through the information sheets. CAMEO_DEUINTIL_2015 was a perculiar ranked, numerical feature that seems to combine two categories:

- Wealth i.e. Wealthy/Prosperous/Comfortable
- Life stage i.e. Pre-Family/Young Couples/Families

The problem with this feature was that Life stage is conflated with wealth in the same rank. I concluded that this feature should be split into two features and that I should be careful to verify there were no other features like this.

An additionally perculiar feature was TITEL_KZ which had the following numerical categories:

- -1. unknown
- Dr.
- Dr. Dr.
- Prof.
- Prof. Dr.
- other

The categories struck me as unclear. Perhaps this is more meaningful to a German company but in the UK there is no meaning behind Dr. Dr. and no meaning could be ascertained from the information sheets.

Later analysis showed that 98% of samples for this feature were NaN. In context of an academic title I induce this to indicate that 98% of the dataset do not have an academic title with 2% known to hold one. Out of context, my inclination would be to drop this feature, however with this context I can re-engineer the feature as a binary:

- 0: No title
- 1: Has title

Since only 2% of individuals have a title, it did not seem to make sense to have more granularity than this and my expectation would be that this could form value information when it would later come to building my clusters.

Identifying this method I applied the same principle to several other features, converting them to binary features.

2.2.2 Verifying no features are missing

Having finished reading through the information sheets I next wanted to verify that at all features in the dataset were accounted for in the information sheet and remove any that looked uninterpretable. This was because later when I had formed my clusters I would need to interpret the feature values in each cluster and derive characteristics that described the customer cluster.

I converted the DIAS excel spreadsheets to a .csv and read this into my Jupyter notebook. This was not entirely straightforward since excel sheets do not translate seamlessly into .csv files but I was able to isolate a feature list and cross-reference this with the dataset.

This highlighted an initial point raised in the project brief: This is a messy dataset and as part of this the features listed in the sheet did not match the features in the database. A lot of time was spent matching up features with slight name variations and further time was spent isolating the 'definitely missing' features.

Each missing feature was analysed with a key concern being whether it would have an undocumented missing value assignment.

At the end of this analysis I had identified 26 features that were either undocumented and considered risky or high cardinality unranked categorical features that I felt would harm my clustering attempts.

2.2.3 Reclassifying 'unknown' values as NaNs

At this stage, I had a set of features I largely understood and was confident I could normalise. My next step was to handle the unknown values that I had identified.My intent was to convert these to NaN values and later, impute new values for the so to aid in this, I constructed an analysis .csv table manually and input the unknown values for each feature. I considered writing a script to do this but considering the discrepancies between the information sheet and dataset feature names I decided the quickest way was to do this by hand. Finally I developed a Python function to handle this task.

2.2.4 Dropping sparse features

Now that all unknown values had been reclassified as NaN I wanted to evaluate whether any features or rows should be dropped. Figure 1 shows a chart of the features with the highest % of missing rows with a cutoff line at 33% to indicate which features I should analyse further.

ALTER_KIND1-4 was not present in the information sheets but from analysis I noticed that it ranged from 1-18. I inferred this to mean whether the individual had a child, and if so the age. If they had 2 kids, the age of the second child would be ALTER_KIND2 and so on. Using this information I decided I could also convert this feature to a binary: 'Has Child' = 1, 'Does not have child'=-1, preserving information that could improve the prediction.

After analysing all these features, my feature_drop count was 30, leaving 336 features.

2.2.5 Engineering new features

I One Hot encoded two features:

- ARBEIT
- NATIONALITAET_KZ

```
<matplotlib.axes._subplots.AxesSubplot at 0x21117a030c8>
```

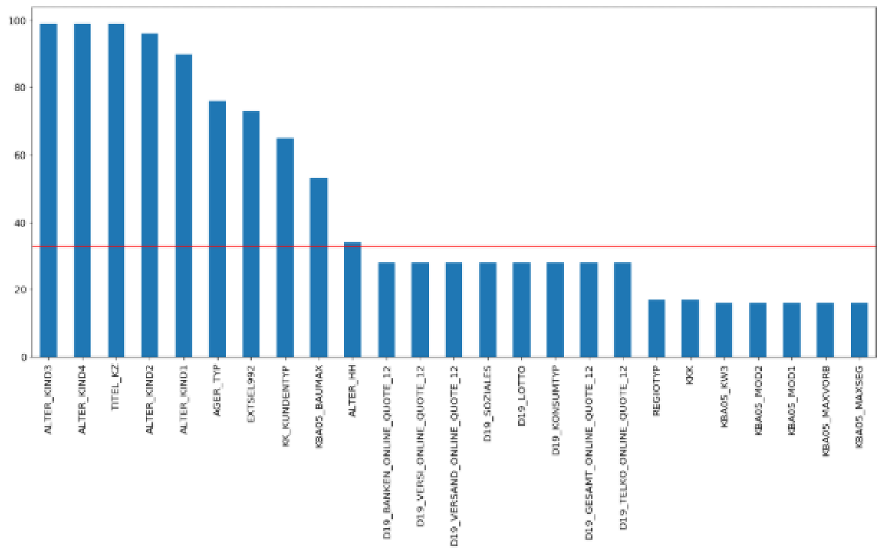

Figure 1: % Missing rows in Azdias Dataset features

In hindsight, there was no need to encode ARBEIT. I mistook this feature for meaning the type of work an individual did but re-referencing the DIAS information sheets I see that this actually means share of unemployment. NATIONALITAET_KZ refers to nationality.

CAMEO_INTL_2015 wa s split into two features representing Wealth and Life-Stage.

2.2.6 Remove sparse rows

I wanted to remove any individuals that were sparse since for the clustering and PCA there need to be no missing values. I felt that for very sparse individuals, imputing values would lead to greater distortion.

I plotted a histogram of the individual counts against number of features missing as shown in Figure 2

2.2.7 Summary

The following steps have been cleaning processes have been taken:

1. Dropping selected features
2. Replacing uknown values with NaN
3. Engineering new features
4. Dropping sparse rows

The Azdias dataset had been reduced from (891221, 366) to (784111, 345).

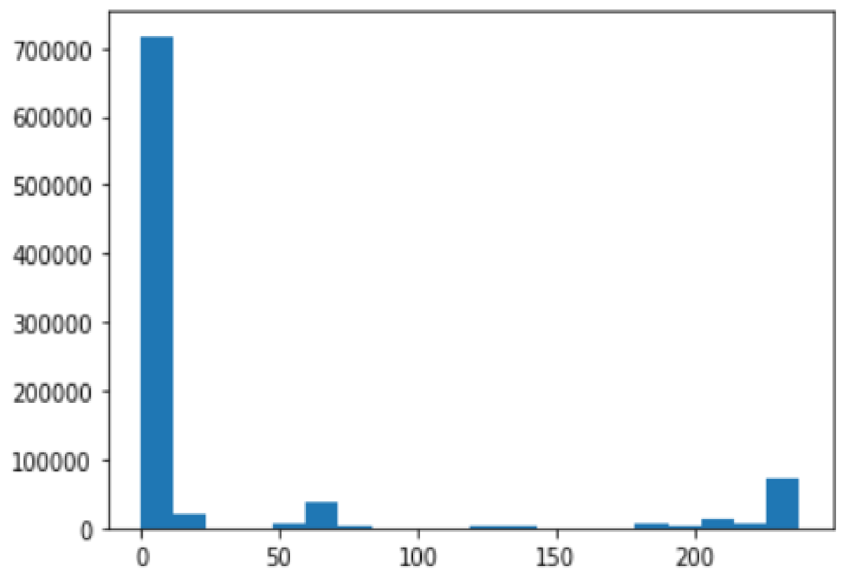

Figure 2: % Missing feature counts in Azdias Dataset features

2.3 Exploring noisy features

Once the dataset was cleaned I did some exploratory data analysis. I wanted to verify the features I had cleaned wouldn't be noisy as this would affect the PCA transform.

Initially I tried identifying noisy features through outliers and features with a high proportion of single value counts but later I found plotting histograms of the unique value counts for each feature was much more effective at identifying noisy patterns and I found a few features with more missing values that weren't noted in the DIAS sheets. My key findings were:

1. GERBURTSJAHR - Year of Birth - had lots of values set to 0 which presumably mean missing. This was not documented in the information sheets and was not identified as an outlier since the median value was 1990 but was clearly seen in the histogram. I dropped this feature since over a third of the values were now seen to be missing.

2. ANZ_HAUSHALTE_AKTIV - number of households in this building - max value of 595, 75% value point was 10. This long tail would distort the normalisation. One option would have been to window the higher values into one bin which is a cleaning method I learned about much later. I chose to drop any rows that had a value of 28 - 5% of the rows.

3. ANZ_PERSONEN - number of persons in this household - max value of 45, 75% value point was 2. I dropped all rows that had a value greater than 6 - 0.9% of rows.

4. ANZ_STATISTISCHE_HAUSHALTE - undocumented feature. max value of 449, 75% value point was 9. I dropped all rows that had a value greater than 20 - 6.9% of rows

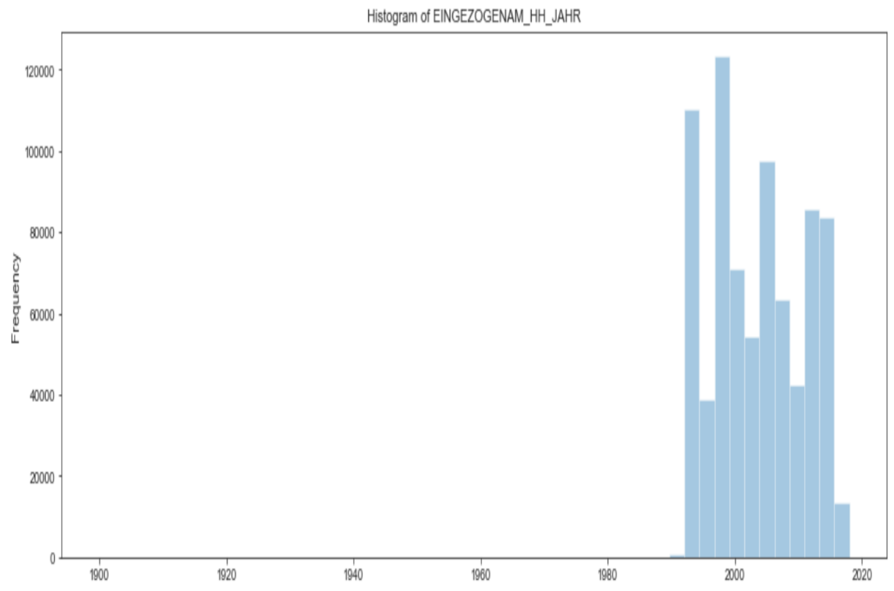

Figure 3: Histogram of EIGENZOGENAM_H

5. EINGEZOGENAM_HH_JAHR - undocumented feature. This feature had 1 value at 1900 with 99% of values above 1993. Without more context I couldn't ascertain the value of this feature. I dropped all rows that had a value less than 1993.

After analysing each histogram I chose to convert a large number of D19 features into binary features alongside other features that had a very high proportion of values at 0. The D19 features were largely identifying transaction behaviour with 0 being no known transactions. By converting these into binary features it could better indicate whether an individual was either 'active' or 'inactive'.

2.3.1 Summary

After exploring the dataset for noisy features, the Azdias dataset had been changed from (891221, 366) to (718446, 341). I did not want to use the binary in PCA and Kmeans since my understanding is they do not translate well into the euclidean space used for both those processes. The number of non-binary, numerical features I would use for PCA was 248 which was already a significant reduction.

2.4 Preprocessing the Clean Dataset

At this stage I had a set of numerical features I was confident would transform well into principle components and I just needed to impute missing values and perform standard scaling.

Initially I tried using a knn imputer to impute missing values. The drawback of using a simple imputer that just replaces with the rows median,mean or modal value is that this can distort PCA by ignoring the correlation between features. Unfortunately due to the size of the dataset the computational requirements for this were prohibitive. So instead I further dropped rows with greater than 20 features missing and chose a modal value for imputation. Towards the very end of the project I learned of a technique where I could use correlation matrices between features to construct an improved imputation method which may be worth using in future projects. In this instance, I was able to perform clustering for the purposes I required so did not revisit this method.

Once all missing values were imputed, the dataset was scaled using a standard scaler, with unit variance and a centred mean value at 0.

2.5 Dimension Reduction

Now the dataset was ready for PCA, it was important to consider a target number of dimensions I would need for clustering. The general rule of thumb I found was around 10 dimensions or greater in a dataset was considered high and should require some form of dimension reduction before clustering. However, I was unclear how this related to the number of samples in a dataset. To understand this further I considered the mathematical relationship.

Supposing each of my features on average has 5 discrete levels, in 1 dimension I could have minimum of 5 evenly spaced sample points. In 2 dimensions this would be 25. The relationship that can be derived is therefore an exponential one:

$$N^d \approx S_{min}$$

Where N is the number of levels, d the number of dimensions and *S_min* the approximate minimum number of samples I would expect to evenly populate that dimension space. The caveat with this formula is that I am expecting there to be clusters where there are close groupings separated by space. But for want of a better resource I felt it served as a useful approximation to give a sense of how much of a reduction I should require in order to be able to perform clustering.

The number of samples I have is approximately 1 million.

$$\log_N S_{min} = d$$

$$\log_5 1,000,000 = 9\, dimensions$$

$$\log_4 1,000,000 = 10\, dimensions$$

$$\log_3 1,000,000 = 13\, dimensions$$

$$\log_2 1,000,000 = 20\, dimensions$$

Explained Variance %	Number of Principle Components
30	5
40	10
50	18
60	30
70	48
80	74
90	116

Table 1: Table of Principle Components representing Explained Variance - Initial attempt

So depending on the groupings I should at least want to reduce my dimensions down from 248 to at least 20 or less.

Performing PCA yielded results shown in Table 1

From this I concluded that PCA was not sufficient to reduce my dimensions and I would need to explore further feature selection prior to clustering.

2.5.1 Feature Selection

I wanted to find the features that varied most between the general population and customers. In other words, which features were the best predictors of being a customer?

I concatenated the Azdias and customer datasets and labeled the general population having a RESPONSE==0 and the customers having a RESPONSE==1. In other words, if this were a mail order campaign, all current customers would have a conversion of 1 and all non-customers would have a conversion of 0. This dataset was then fed into an XGB Classifier to leverage the feature_importance_ instance method. XGB is an Extreme Gradient Boosting ensemble method. The basis for which, is a decision tree algorithm using gini impurity to distinguish how much each feature contributes towards the performance of the model.

Using this wrapper method of feature selection, I was able to quickly reduce the dataset of 248 features down to 53 that my model considered most important to being a customer.

2.5.2 PCA with feature selection

Table 2 shows a table of the explained variance from PCA. From this I concluded I could successfully use PCA to further reduce the number of dimensions and selected 15 Principle Components which represents 90% of the dataset variance.

Explained Variance %	Number of Principle Components
30	0
40	1
50	2
60	3
70	5
80	9
90	15

Table 2: Table of Principle Components representing Explained Variance - after Feature Selection

3 Customer Segmentation Report

3.1 Unsupervised Learning with KMeans

To find an optimal number of clusters I used the inertia criterion and Elbow method. The inertia decreases as a function of the number of clusters and plotting the inertia against number of clusters shows the rate of change of inertia. The optimal cluster can be found at an inflexion point where the rate of change sharply decreases.

The Elbow method result can be seen in Figure 4. There wasn't a clear inflexion point 4 but the the rate of change seemed to decrease most between K value of 7 and 9. Figures 5 and 6 show comparison of the clusters for the general population and customer population. In each plot it can be seen that the customer population falls largely into 3 clusters and the change from 9 clusters to 7 clusters doesn't noticeably alter the proportions. This indicates that the additional 2 clusters are only further segmenting the non-customer cluster groups. Based on this analysis it seemed that 7 clusters would be an optimal based on the supplied data.

Now that I have identifiably segmented the customer population into clusters I could start building a report of features. I labelled the customer clusters: A,B,C with cluster A making up the largest % of the customer population and cluster C making up the third largest %.

I selected features to analyse by comparing the mean,median and mode values for between Clusters A+B and Non-Customer clusters for each feature in the original Azdias and Customer datasets and building a set of features which had non-zero differences. I then plotted histograms for A,B,C and Non-Customer clusters for each of those selected features. Overall 33 features were analysed across 165 histograms. I considered other ways of combining plots or presenting the information in a way that would be more impactful to a non-technical audience but decided these histograms were the easiest way to draw comparison and build conclusions about the customers.

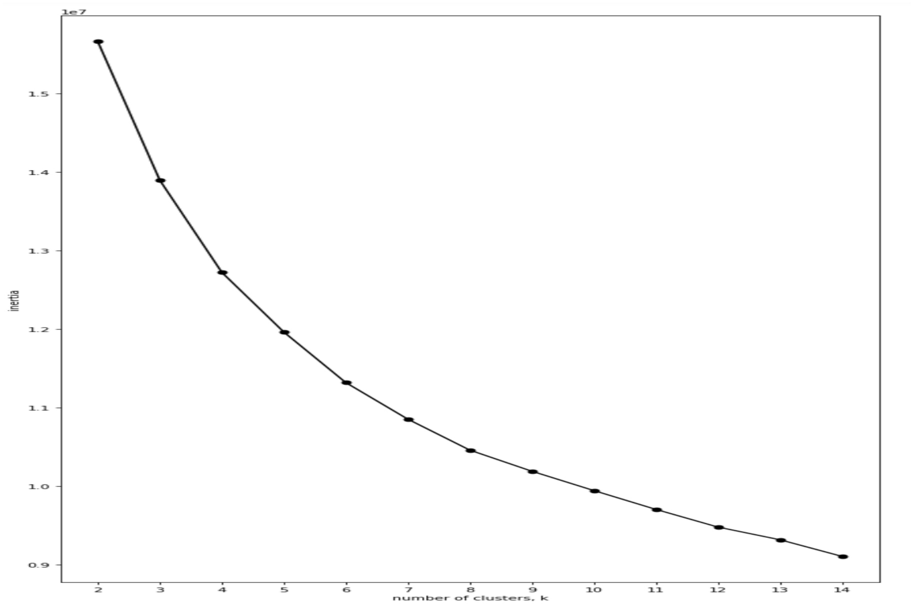

Figure 4: Elbow Method to find Optimal cluster K

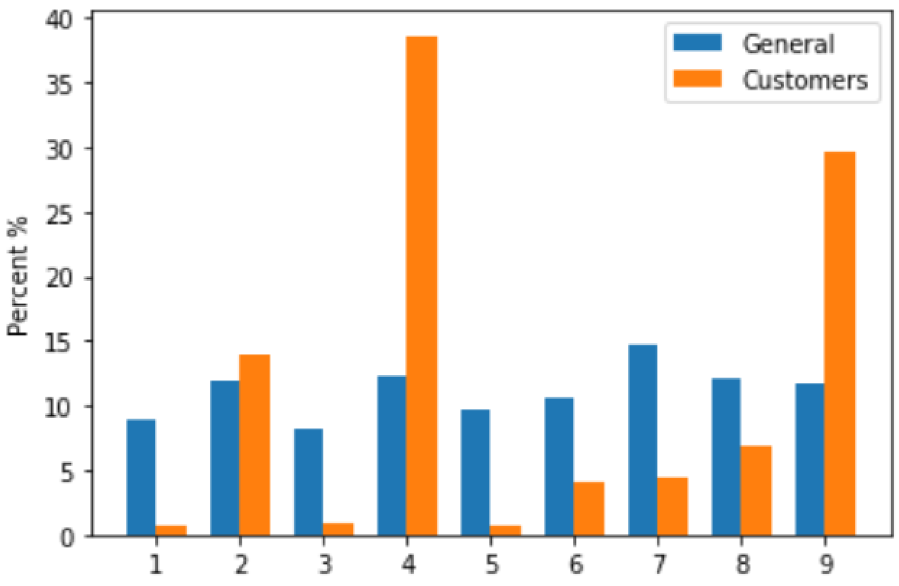

Figure 5: General Population and Customer population cluster comparison for K=9

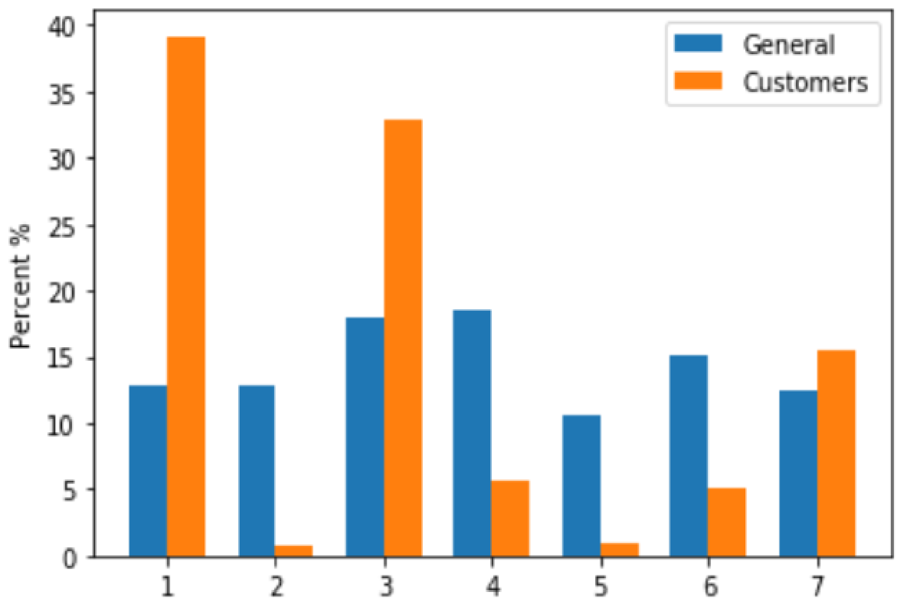

Figure 6: General Population and Customer population cluster comparison for K=7

3.2 Key Findings

This section outlines the key extracts I identified from the histogram comparisons between Customer cluster's A,B,C and the general population.

3.2.1 Person

Features used: ANREDED_KZ, GREEN_AVANTEGARDE

- BCustomer demographics are a balance of gender with slightly more Men than Women: 55% - 45%
- A significant proportion of demographic A customers: 45% compared with 18% outside those demographics, belong to the Green Avantegarde Movement.

3.2.2 Affinity

Features used: SEMIO_PFLICHT, SEMIO_LUST, SEMIO_RAT, SEMIO_TRADV, SEMIO_ERL

- Customers score very high for being dutyfully minded
- Customers score low for being sensually minded
- Customers score very high for being rationally minded
- Customers score very high for being traditionally minded
- 50% of Customers score very low for being eventfully orientated and only 20% score above average

3.2.3 Finance Typology

Features used: FINANZ_SPARER, FINANZ_MINIMALIST, FINANZ_VORSORGER, FINANZ_ANLEGER

- Customers score the highest for money saving
- Customers A & B score the lowest for financial interest with Customers C scoring average. Non customer demographics generally scored high for financial interest
- Customers score the lowest for 'be prepared'
- Customers score are highly likely to be investors

3.2.4 Customer Journey Type

Features used: CJT_TYP_1, CJT_TYP_2, CJT_TYP_4, CJT_TYP_5, CJT_TYP_6, RETOURTYP_BK_S

- Customers score high for Advertising and Comsumption Minimalists
- Customers score high for Advertising and Comsumption Traditionalists
- Customers score low for Advertising interested Online Shoppers
- Customers score low for Advertising and Cross-Channel Enthusiasts
- Customers score low for Advertising Enthusiast with restricted Cross-Channel Behaviour
- Customers are generally determined Minimal-Returners with A and B demographics being more incentive-receptive

N.B. There is no supplied scale so this information would need to be fedback to Arvato to confirm. I am assuming a low value indicates a high interest and a high value indicates a low interest.

3.2.5 Financial Activity

Features used: D19_SONSTIGE, D19_GESAMT_OFFLINE_DATUM, D19_GESAMT_ANZ_24

- Demographic B customers are active transactors in ALL OTHER CATEGORIES compared to non customer demographics
- Demographic A customers are active transactors in the OFFLINE group compared to non customer demographics
- Demographic A customers are active transactors in the TOTAL_POOL group compared to non customer demographics

3.2.6 Age & Earnings

Features used: AGER_TYP,ALTERSKATEGORIE_GROB,PRAEGENDE_JUGENDJAHRE, CAMEO_INTL_2015_LS,LP_LEBENSPHASE_GROB,LP_LEBENSPHASE_FEIN

- Customers from A are predominantly high-income earners of higher age from older families or Mature Couples
- Customers from B are predominantly high-income earners of higher age from single person households
- Customers from C are predominantly single low-income and average earners of higher age
- Over 50% of the customer demographic belong to the elderly typology, in stark contrast to around 5% of the non-customer demographics
- Over 60% of the customer demographic is over 60 years of age, with less than 5% being under 45.

Key dominating movements:

- Customer A - 70ies Family orientation and 70ies Peace movement
- Customer B - 70ies Family orientation and 60ies Economic Miracle
- Customer C - 50ies Economic miracle, 60ies Economic Miracle and 70ies Family orientation
- Non customers were more likely to be of the 80ies and 90ies movements

3.2.7 Class

Features used: CAMEO_INTL_2015_W, CAMEO_DEUG_2015, HH_EINKOMMEN_SCORE, LP_STATUS_FEIN,LP_FAMILIE_FEIN

- Customers from A can be considered Wealthy and Prosperous Households and are generally upper class, upper middleclass, from two-generational and multi-generation households
- Customers from B can be considered Wealthy and Prosperous Households and are generally upper class, upper middleclass and single
- Customers from C can be considered Less Affluent and Poorer Households and are generally lower middle class and working class and single

3.2.8 Location

Features used: EWDICHTE, KONSUMNAEHE

- Customers from A and B are located across all categories of 'density of inhabitants per square kilometer' which we means can expect them to be both in Cities, suburbs and more rural areas.
- Customers from C are more likely located in higher density areas like Cities.

3.3 Next Steps

Having identified the key findings the next step would be building this into a polished Customer Report that could be provided by Arvato to the Mail-Order Company and used to inform better marketing products. Currently the language used in my key findings reflects that used in the supplied Dias information sheets and is currently quite clunky and technical. Working with Arvato, this would need to be re-written and presented in a way that would best communicate to the Mail-Order company the insights I have found. Ideally this would need corroboration with the team that put together the Dias sheets and a marketing team or copywrighters to build an impactful report that reflects well on Arvato.

4 Building a Supervised Model

Now that I have finished the customer segmentation part of the project. The final stage is to build a supervised model using the mailout datasets. The intention of this model is to predict whether an individual is likely to convert to being a customer and can be used to better select individuals to target in a future marketing campaign.

For the supervised model I have chosen to use XGBoost. XGBoost is a competition winning model that stands for Extreme Gradient Boosting. I chose not to test a range of different models for this project. I had spent a considerable amount of time understanding XGBoost in preparation and my concern with evaluating different models was not only the time cost in building understanding for each but also tailoring the datasets to best fit those models. While the same dataset could be used for different models, I believe evaluating models in this way is not a fair comparison; knowing that XGBoost is a strong contendor generally, I thought my best strategy would be to focus on building a strong XGBoost model and corresponding dataset.

In preparation for using a Kmeans clustering to identify clusters, a lot of work went into preparing the Dataset. Dimensions were reduced down to 20, binary features were removed, missing values imputed. For the XGB model there was not the same requirement to alter the data to fit euclidean space and the set of features that would best form clusters wouldn't necessarily be the best predictors of customer response. Therefore, I needed to rebuild the dataset.

The mailout train dataset provided by Arvato is significantly smaller than the Azdias and Customer dataset: 42,962 with only 532 converting i.e. 'RESPONSE'=1. The customer dataset in contrast contains 191,652 individuals. Assuming that the 532 reflect similar qualities to the customer dataset, it strikes me that this is too small a number to be able to create a general model that can predict the RESPONSE on unseen data. For this reason, I decided to once again combine the Azdias and Customer data into one dataset that I would use to train my XGB Classifier model and use the Mailout Train dataset as a validation dataset. I felt this would improve my training algorithms ability to make accurate predictions on unseen data.

4.1 Feature Selection

Before training my model I wanted to determine a subset of features to use. I was a bit concerned at just using the XGBoost feature_importance_ method since I was also using XGBoost for training the model and from my research this method can be susceptible to flaw. To maintain a robust feature selection I cross compared the feature_importance_ with SHAP:

> SHAP (SHapley Additive exPlanations) is a game theoretic approach to explain the output of any machine learning mod[2].

The SHAP library is especially developed for working with ensemble tree methods and can explain which features are most important to the model. Generally the outputs between the XGBoost feature_importance_ and SHAP output agreed as showing in Figures 7 and 8.

I also tried another feature importance technique called Boruta but found it was not able to successfully distinguish features.

Using this process, I reduced the features down to 30 and engineered 7 new One Hot encoded cluster features using the unsupervised learning model in the last section. I left missing values in the dataset since XGBoost can handle these in building the decision tree.

4.2 Training First Model

The final shape of my training data was: (1082873, 37) and my validation data set was: (42962, 37) which I fed into an XGBBoost Classfier with default parameters for my base model.

The Gradient Boosting method looks to minimise subsequent prediction error on each iteration. Using the validation set containing the mailorder data and 10 early_stopping_rounds to regulate the improvement, the model can train whilst limiting overfitting.

Using this method, the final ROC_AUC score of the base model on the validation set was 0.766 or 76.6%. Anything above 75% for an ROC_AU metric is generally considered pretty good so i'm happy with this baseline. Before Hyperparameter Optimization I wanted to experiment with a few alterations to my dataset.

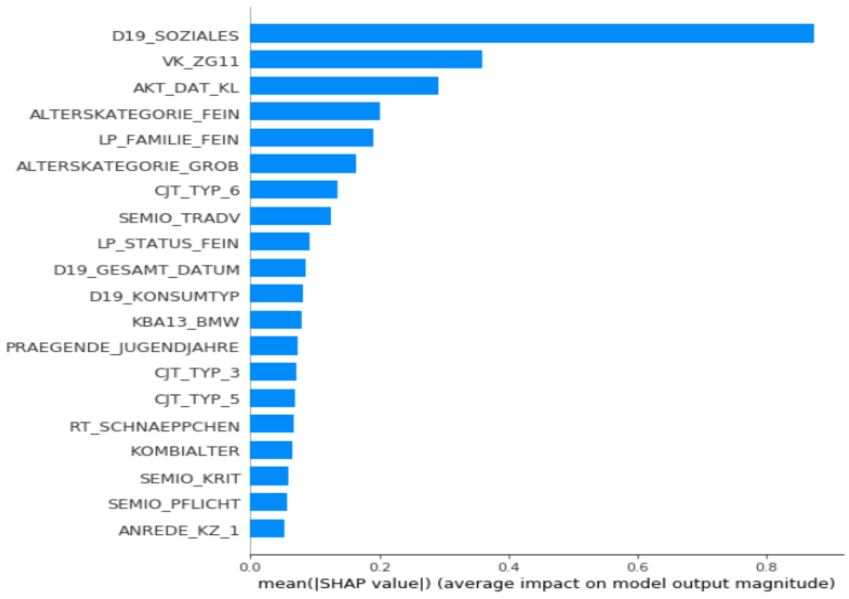

Figure 7: SHAP assessment of Feature Importance

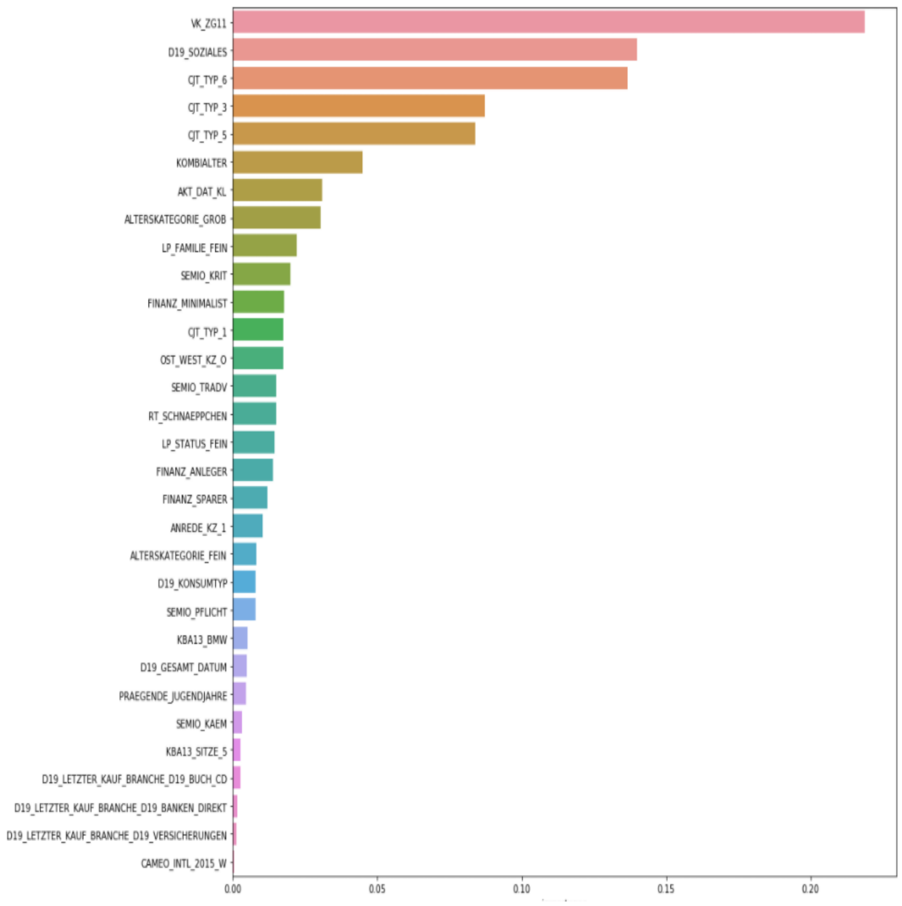

Figure 8: XGB assessment of Feature Importance

I had read that standard scaling the data can help with prediction. In theory this shouldn't make a difference since the underlying decision tree should still branch at the same point regardless of the transformation. After standard scaling I found there as no change to the final score.

I wanted to try imputing the missing values to see whether this could improve the score. I used an iterative imputer from sklearn for this. I was not previously able to make use of this imputer since it is computationally expensive, but with only 37 features this was now feasible. The iterative imputer is a more sophisticated approach of imputation that imputes missing values as a function of other features. Using this iterative imputation I found the resultant model score was slightly worse than the base model.

4.3 Hyperparameter Tuning

The XGBoost model comes with a range of hyperparameters that can be altered:

- max_depth
- n_estimators
- learning_rate
- gamma
- subsample
- colsample_bytree
- reg_alpha
- reg_lambda
- scale_pos_weight

From reading the supplied documentation and reading various articles I settled on some upper and lower bound limits for each of these parameters and used a Bayesian Optimiser to explore this parameter space to attempt to find an optimal set of values. Bayesian Optimisation is a fast alternative to an exhaustive grid search of parameter combinations that aims to quickly find an optimal parameter space. From the Python package docs:

> Bayesian optimization works by constructing a posterior distribution of functions (gaussian process) that best describes the function you want to optimize. As the number of observations grows, the posterior distribution improves, and the algorithm becomes more certain of which regions in parameter space are worth exploring and which are not [1]

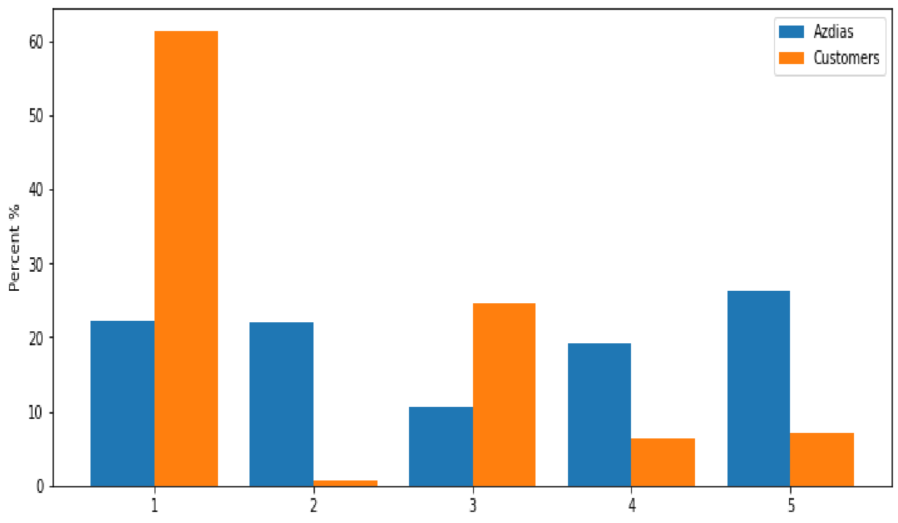

Figure 9: Comparison of Azdias and Customer Clusters

I setup a blackbox function that trained my model on a supplied parameter set and returned the ROC score which the bayesian optimiser would look to maximise. The bayesian optimiser was setup to run for 25 iterations using 3 initial points and at the end of this process the final optimised model score was about the same as the base model.

4.4 Kaggle Prediction - 1

Using my optimised model I made predictions on the mailout test data which had a resultant score of 0.786. Placing me 120th on the Leaderboard. I felt this was a reasonable score but before finishing wanted to explore a few more avenues.

4.5 Dataset optimisation - 1

One area I had neglected to explore was the cluster distribution on the mailout training set and in comparing datasets I came across an important discovery. The cluster comparisons are shown in Figures 9 and 10. As can be seen in the 5 cluster comparison, the customers fall largely into cluster's 1 and 3. Interestingly looking at the Mailout Train dataset, the distribution between R=0 and R=1 is fairly matched in each cluster and most of the mailout training set falls into cluster 1. From this I can draw some important conclusions:

- The Mail order company are already targeting their customer demographic OR the dataset has been curated such that there is an even stratification between demographics.
- Cluster is not a good predictor of customer conversion

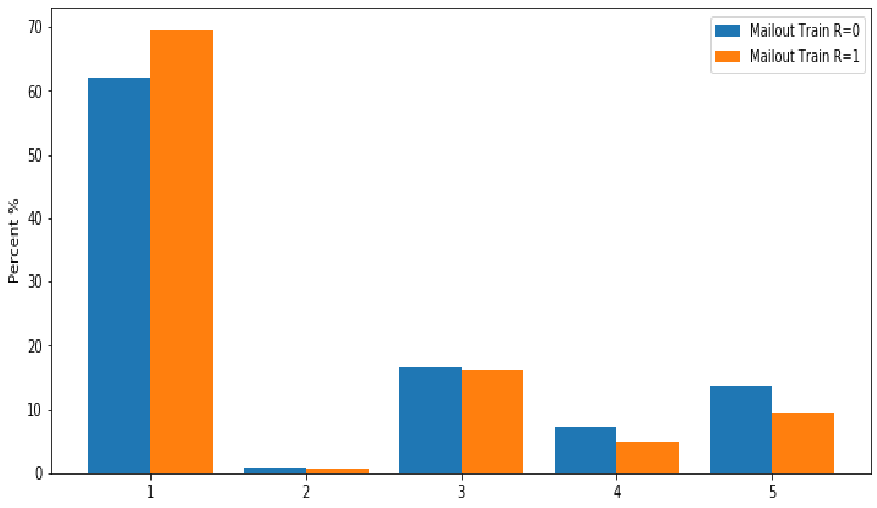

Figure 10: Comparison of Mailout Train dataset Clusters

If cluster is not a good predictor of customer conversion, then the features that distinguish customers from the general population won't necessarily be strong predictors of conversion either. Therefore, in order to determine which features are strong predictors I shouldn't be looking at customers relative to the Azdias population but rather looking at just the mailout train dataset and what distinguishes those with a RESPONSE=1.

Since the XGBBoost Feature Importance instance method appeared to be robust, I used this on the mailout dataset and reduced it down to 284 features. I was no longer certain that using the Azdias dataset and Customer dataset would not be distorting my results so I chose to use a 5 fold stratified cross validation strategy using just the mailout dataset to train a model and Bayesian optimisation to optimise the parameter space. After this, the final ROC score was 0.668. A significant degradation in performance. I was confident the set of features I was using were better predictors than previously, so my best working theory for this performance drop is that there is not sufficient target label data to predict the outcome accurately.

4.6 Kaggle Prediction - 2

One theory I had not tried was improving the Bayesian optimiser. It occurred to me that perhaps 25 iterations was not sufficient despite the output appearing to have found an optimal point early on. I tried running the Bayesian optimiser for 400 iterations at 5 random initial points on the same training and validation datasets I had used the original prediction. Interestingly, this boosted my training set score from 0.763 up to 0.781. Using this new model, I made predictions on the Mailout test data which had a resultant score of 0.8126. Placing me 1st on the Leaderboard!

Despite this producing the best model. I had not used my new features to inform this prediction. Therefore before finishing this project I wanted to explore a few more avenues.

4.7 Dataset optimisation - 2

One concern I had with labelling all the customers with 1 and the Azdias dataset with 0 was that this isn't quite accurate. It creates noise by ignoring the fact that individuals in the Azdias dataset could convert to customers and have the exact same characteristics. Conversely, I know that the Mailout dataset looks quite similar with respect to the clustering, to the customer dataset already so while it's true to say that customers have a some point converted, it's not true to say they would convert based on the historic mail order campaign. I think part of the reason combining the Azdias and customer datasets as the training set and using the Mailout train dataset as a validation set with the XGBoost model is that in each sequence of error improvement it's using the true response of the Mailout data to improve on the noise of the response of the artificial response.

Therefore, one avenue i wanted to try was rather than artificially setting the ylabel in the Azdias and customer datasets, was using a model trained on the Mailout train dataset to predict the y label for Azdias and customers and then using this to train the final model using the Mailout train dataset as the validation dataset. On one hand, this could reduce some of the noise introduced, on the other creating a feedback loop in this way could degrade the performance. After testing this out I found the final score was 0.71 which is significantly worse than my original base model. I would be interesting in doing more research into whether there is any credibility or discussion of this method since I couldn't find much in my extensive reading for this project.

Finally, I tried using just the Mailout training set to determine the 100 most important features for determining customer conversion. I then combined Azdias with a label of 0 and Customers with a label of 1 and selected these 100 features. Training this on a base model my final score was 0.816. My current best score was 0.781 so this was a significant improvement. After Bayesian optimisation to find optimal parametes this score had been raised to 0.8306.

4.8 Kaggle Prediction - 3

Using the final trained model I made a prediction which had a resultant score of 0.847. A significant jump from my last prediction of 0.8126.

At this stage I was content I had explored all the avenues I had wanted in order to produce a strong model and considered a score of 0.847 to be very good in absolute terms and even relative to other submitted models.

5 Conclusion

The problem statement set out at the start of this project had two questions:

1. Can we identify a demographic in our current customers with respect to the general German to inform better marketing products?

2. Can we reduce the number of people targeted and still retain a high proportion of our target demographic?

I have successfully been able to segment the current customer population. I have produced a set of key characteristics that can be built into a customer segmentation report for the Mail Order company that can help build better future marketing products.

A strong performing model has been built and verified. Using this model and by tuning the sensitivity and specificity, a much smaller proportion of a general population can be identified and targeted whilst still maintaining the historic customer conversion rate. This can significantly save on cost of any future marketing campaigns.

6 Closing Remarks

This project explored a wide range of Data Science and Machine Learning methods and challenged me to think deeply about how to translate theory and methods into good practices. Whilst I can't validate my own performance, I am very happy with everything I have learned and achieved through this project and look forward to building these skills further and being able to take part in the discussion on how to building strong Data Science and Machine Learning practices.

References

[1] Baysian Optimisation. https://github.com/fmfn/bayesianoptimization.

[2] SHAP. https://github.com/slundberg/shap.