Project Definition: Customer Acquisition is the process of bringing new customers to a business, a key ingredient of business growth. Ideally, it consists of a strategy that targets true potential customers, rather than wide spread marketing to the general population at large. Machine Learning (ML) provides a unique approach to this problem, in that, instead of relying on instinctive heuristics that have traditionally made marketing an 'art', it leverages on statistical analysis of known customer's data, and the correct extraction of the information therein, making the process more of a science.

The problem that this project addresses is the question of whether the customer acquisition strategy of a Mail-Order company can be made more efficient using ML techniques. A vast amount of demographic data is made available for the target general population — Germany, as well as data from some of the firm's customers. Additionally, the results of a recent mailout campaign are provided. All in all, the givens for this project are three datasets in four files as follows:

- 1. ***Udacity_AZDIAS_052018***.csv: Demographics data for the *general population* of Germany; 891 221 persons (rows) x 366 features (columns).
- 2. ***Udacity_CUSTOMERS_052018***.csv: Demographics data for *customers* of a mail-order company; 191 652 persons (rows) x 369 features (columns).
- 3. ***Udacity_MAILOUT_052018_TRAIN***.csv: Demographics data for individuals who were targets of a marketing campaign; 42 962 persons (rows) x 367 (columns).
- 4. ***Udacity_MAILOUT_052018_TEST***.csv: Demographics data for individuals who were targets of a marketing campaign; 42 833 persons (rows) x 366 (columns).

Additionally, two spreadsheets: ***DIAS Attributes - Values 2017.xlsx***, and ***DIAS Information Levels - Attributes 2017.xlsx*** were also provided with descriptions for most, but unfortunately not all of the features in the datasets. Perhaps they need to be updated from 2017

The first two files were used to establish the differences between customers and the general population at large using a combination of statistical visualizations as well as unsupervised learning ML techniques. The remaining two files were used to train a predictor model to ascertain which individuals would be most likely to become customers. The "TRAIN" data was used to train and tune the predictor, while the "TEST" data was used to participate in a Kaggle Competition.

The essence of this project is didactic in nature; it is the capstone project culminating a series of assignments that aims to provide a well rounded understanding of ML techniques for someone interested in Data Science. First, we'll use visualization tools to gain some insight as to how the features of the general population and the customers differ, i.e., their distributions. Second, we'll implement unsupervised learning ML techniques, namely, Customer Segmentation through Principal Component Analysis and K-Means Clustering to examine what they reveal about the characteristics of the customers. Third, we'll fit a collection of binary classifier models to data from the recent mailout campaign, and fine-tune one of them as a supervised learning ML approach. Finally, we'll test the goodness of the tuned fit model by evaluating how well it predicts which individuals are indeed customers. Our goal is to conceive a targeted advertising campaign based on recommendations from the data itself, as exposed through ML techniques, in order to achieve a higher hit-ratio in marketing than is otherwise possible.

The project relied almost entirely on off-the-shelf Scikit-Learn algorithms and models, together with a few Python 3.0 code snippets and ancillary libraries to massage the results and make them more amenable to presentation and visualization. The project was completed entirely on an Apple Computer iMac 24-inch with an M1 chip (8-core CPU, 8-core GPU) running macOS Version 12.0.1 (Monterey). Data wrangling and modelling took place exclusively on a Jupyter notebook available through the Anaconda Navigator 2.0.4 suite of utilities. The rest of the code editing and debugging was carried out using Atom 1.5.8.

The heavy lifting for this project took place during the Data Exploration and Processing stage. Here we faced not only a significant amount of missing row data, but also a surprisingly structured character of the missing data as a series of jumps — Fig. 1.

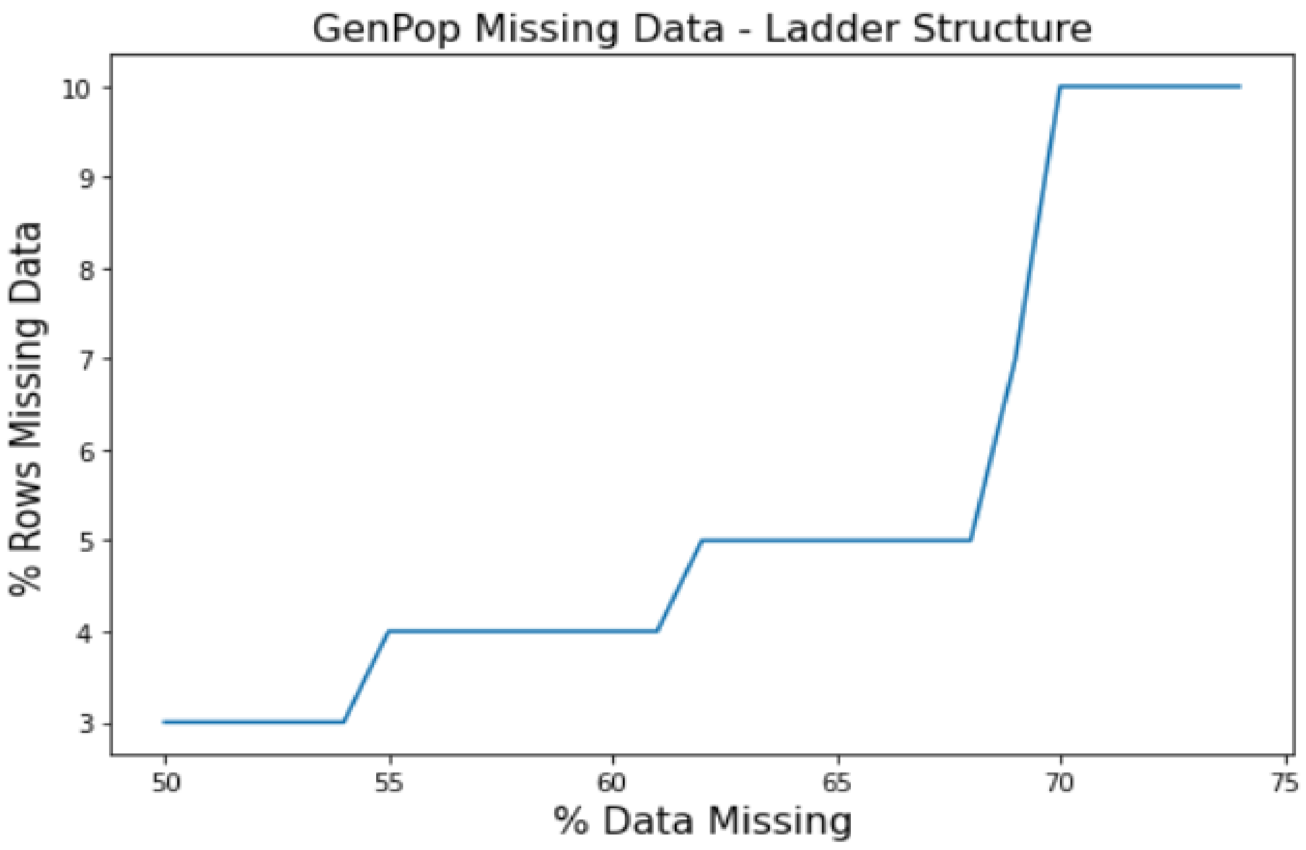


Figure 1.

This seems to indicate that several data collection efforts were mixed to produce the datasets. The idea of analyzing mixed data presented us with a dilemma; On the one hand, discarding roughly 10% of the general population data, and correspondingly 20% of customer rows would be wasteful. On the other hand, if indeed the data was collected through disparate efforts, then this discarded 20% may well be the most meaningful piece of the entire dataset. Further, with a hodgepodge of maybe five separate data collection procedures, the datasets feature distributions and overall characteristics are not likely to be homogeneous, but mixed. That being the case, it makes little sense to proceed casually with such data and ignore this mixing of statistics. Such steps as imputing, scaling and the later modeling algorithms do not expect to deal with these disparities.

For the sake of expediency, it was decided to let go of the worst offenders, i.e., the mostly empty rows missing 60% or more of their entries, while keeping the great majority of hollow observations for later handling through feature imputations. This, in my opinion, was preferable to the alternative excessive imputation of values, leaving columns exhibiting not only a mixture of statistics — the different rungs of the ladder, but also compounding the problem by averaging out these potentially incompatible distributions. By keeping barren rows with fewer missing entries, a compromise was struck that allowed conserving mixed data with good enough quality to conduct a somewhat more meaningful analysis.

Subsequent Data Exploration and Preprocessing steps were straightforward. Most of the features found in the datasets were categorical in nature with numbers for labels, and with only a handful requiring encoding.

The categorical nature of the data made the imputation process simple. Typical univariate imputation algorithms usually rely on some central tendency measure to fill the missing entries of a feature. This works well to the extent that the imputed features are numerical and as long as the amount of missing entries remains relatively small. This was not the case with our data; we took on significant amounts of missing entries, as explained previously. Furthermore, our datasets' features consisted almost exclusively of categorical data, albeit numerically encoded.

For our purposes, given the categorical nature of the features in the data, we implemented an imputing routine of our own. Rather than relying on a ready-made algorithm to impute based on some central tendency of the values, such as mean, median, most frequent or some constant, which would be 'categorically' devoid of meaning, we used the frequency of the feature bins themselves. Distributing the imputed values according to the relative frequency of the slots, allowed us to maintain the shape of the column histograms almost intact. Our imputation technique was also useful in dealing with outliers, resulting in a smoother handling through imputation, rather than discarding — Fig. 2.

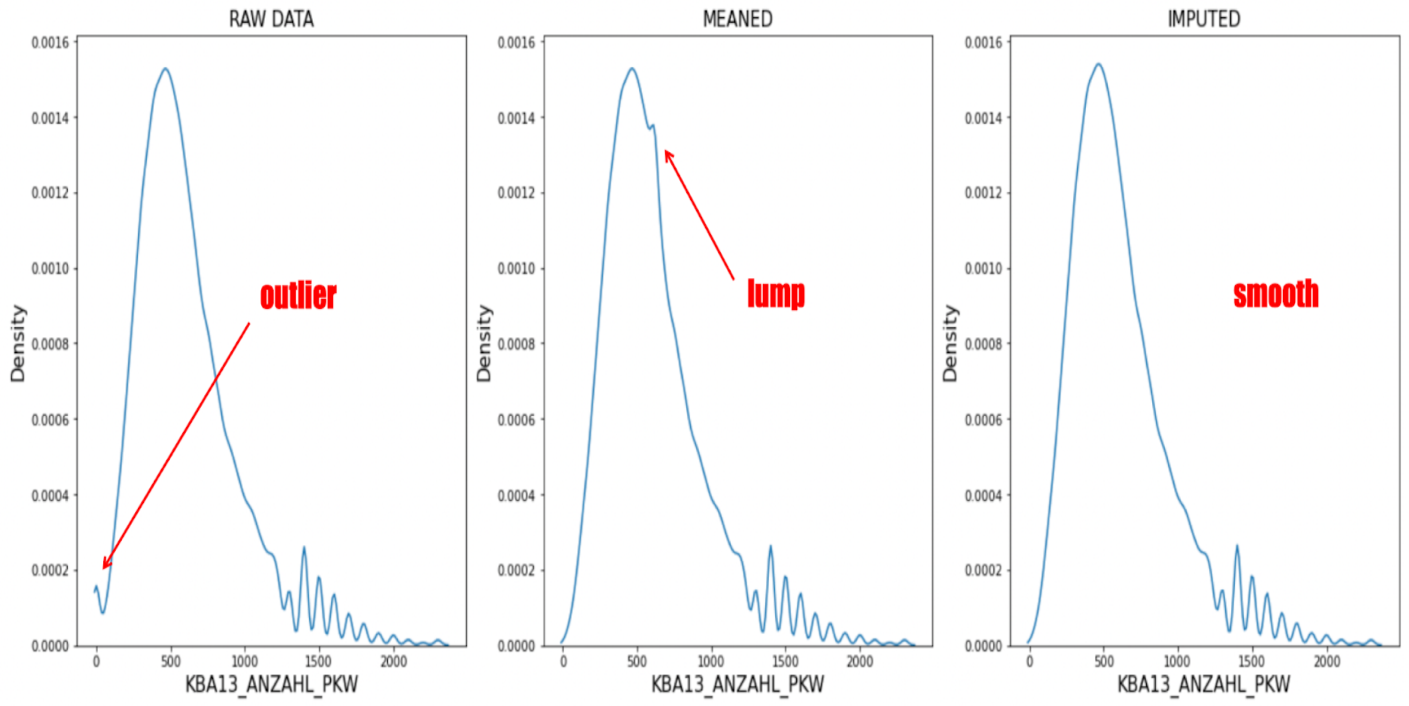


Figure 2.

After cleaning and preprocessing both the general population and customer datasets, we proceeded to compare them side by side on a feature by feature basis using histogram visualizations — Fig. 3.

The accompanying EXCEL spreadsheets were useful to decode most of the feature names, and after enough acquaintance with the data through these visualizations, we were able to ascertain that customers tended to be older, sedentary, of versatile consumption type, heavily into saving or investing money, suburban, with multiple cars in the household, mostly top earners of advanced age, and with a slightly heavier male presence.

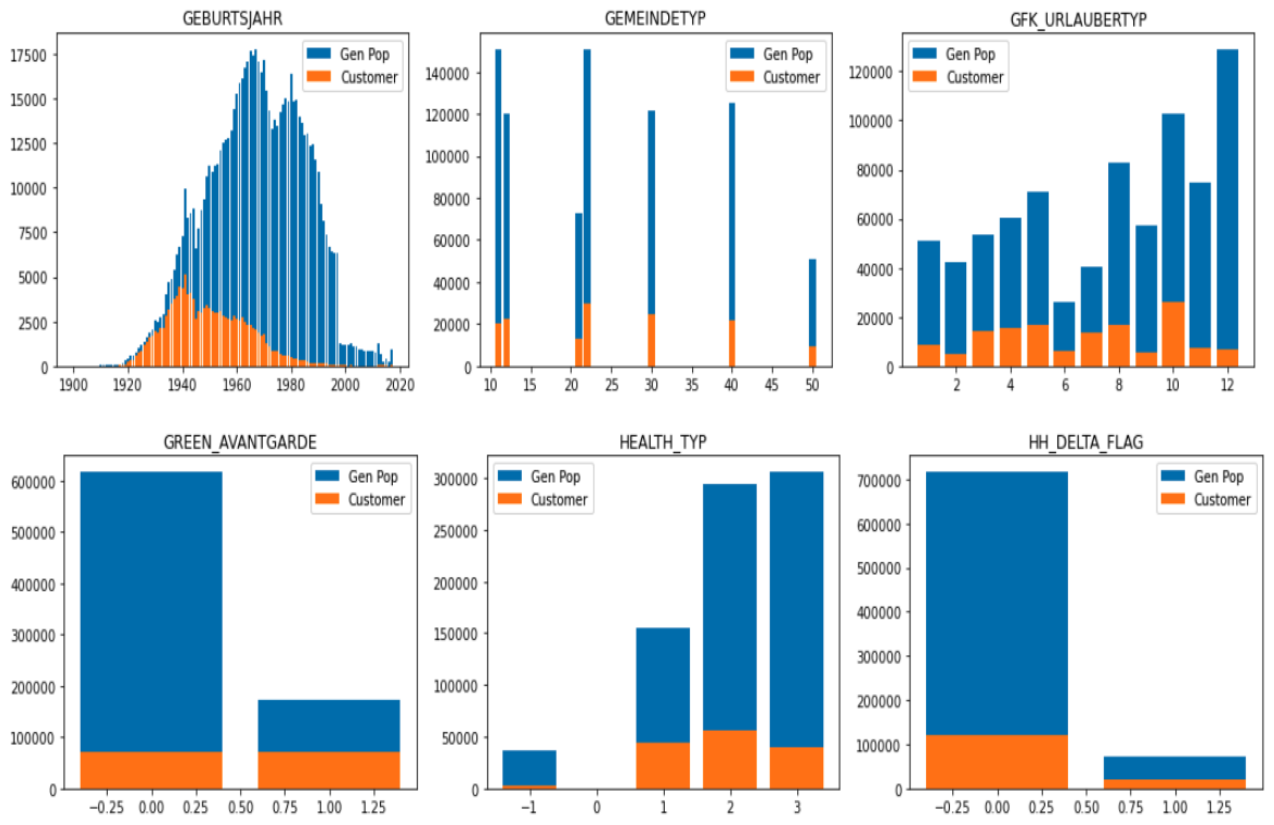


Figure 3.

The next part of the project consisted of actually using Machine Learning algorithms, more specifically, unsupervised ML techniques. From an outsider's perspective, unsupervised Machine Learning appears to be almost magical, in that it "lets the data speak for itself". It is essentially a collection of powerful algorithms that search for and reveal latent patterns in the data, patterns that may be inherent to the data, but may otherwise go undetected. Once demystified, however, it is easy to understand the motivation and inner workings of many of these algorithms. Such obvious procedures as eliminating multicollinearity, or exposing natural groupings underlying the observations are simply the self-evident steps to follow in order to make the information contained in the data intelligible. The following steps do precisely this.

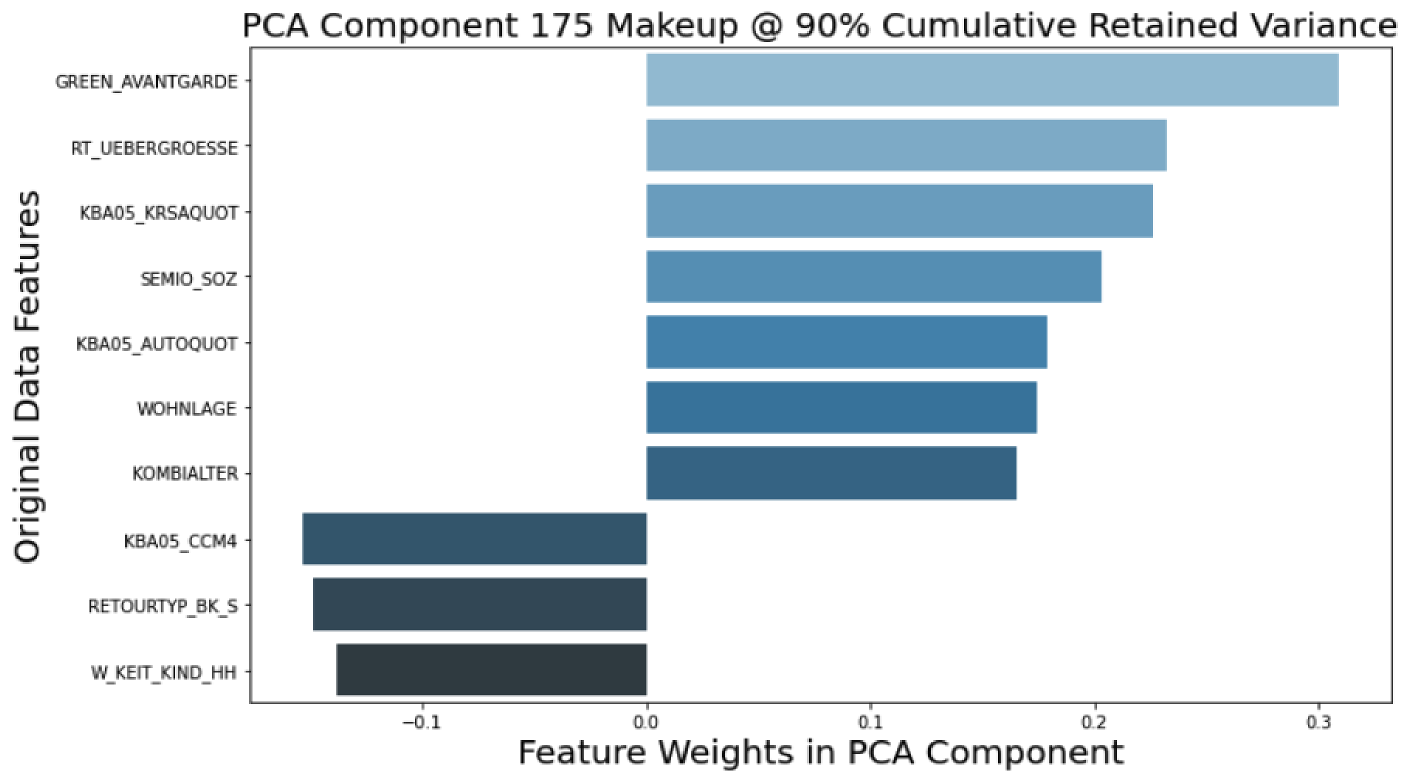
The customer segmentation part of the project consisted first of a dimensionality reduction step through Principal Component Analysis (PCA). With PCA we managed to project the datasets onto a 90% retained variance space, using only 175 principal components instead of the original 366 features of the data. The linear combination of features that made up each principal component as well as the cumulative amount of variance retained by each, were made visible interactively. This interactive tool was then used to decide on a cutoff point for the trade-off between features retained and variance retained. In the end, we opted to let go of about 50% of the columns at the modest cost of 10% of the information contained in the data — Fig. 4.

Perusing the top features in component space revealed some curious characteristics of the German population at large. Certainly, the German general population was not the focus of this project, but nevertheless, it was shown to be rather sedentary, mostly living in 1-2 family homes, driving mostly new upper class cars (BMWs and the like), with above average presence of vans and trailers, as well as being money savers with high online affinity.

given_PCA 175

Features 10

Use the sliders to change Component and Number of Features



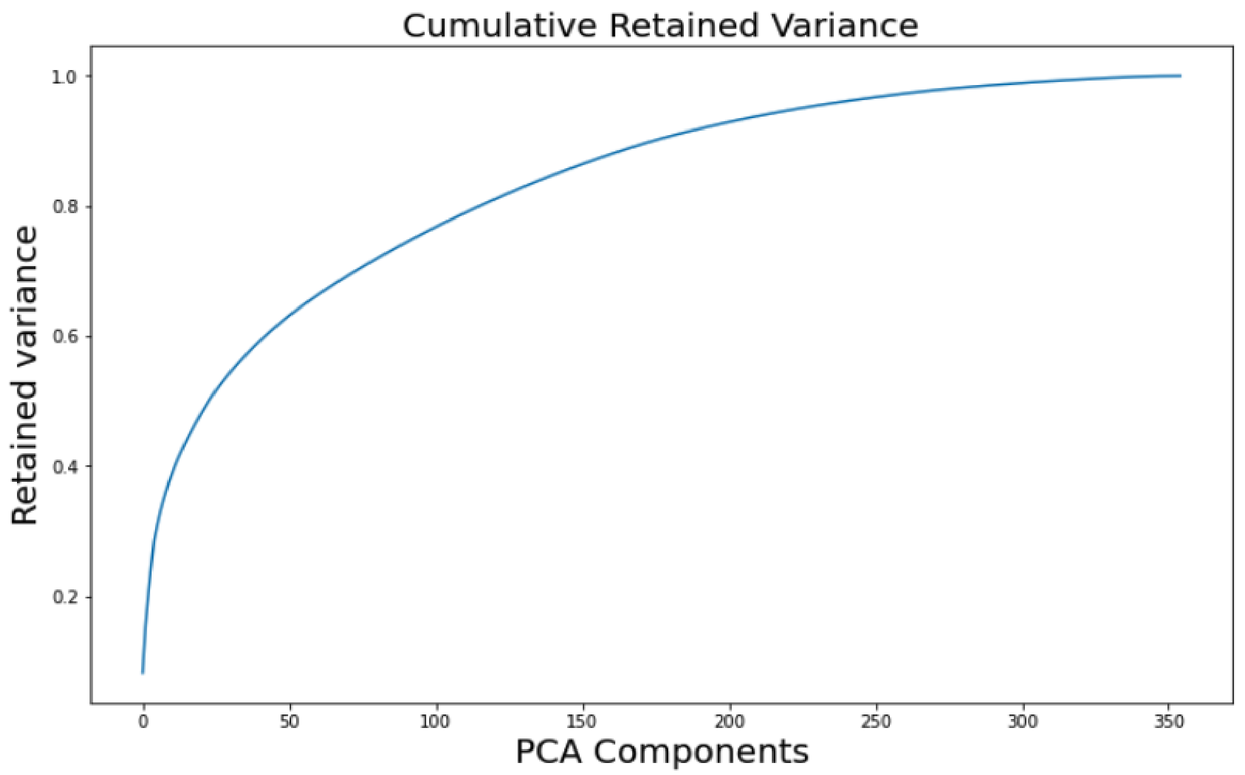


Figure 4.

The next step was a further reduction in the complexity of the data by abstracting it into a segmented representation using a K-Means clustering algorithm. Several segmentation attempts were made by varying the number of clusters. The results were plotted against their average centroid distances resulting in the graph shown below — Fig. 5.

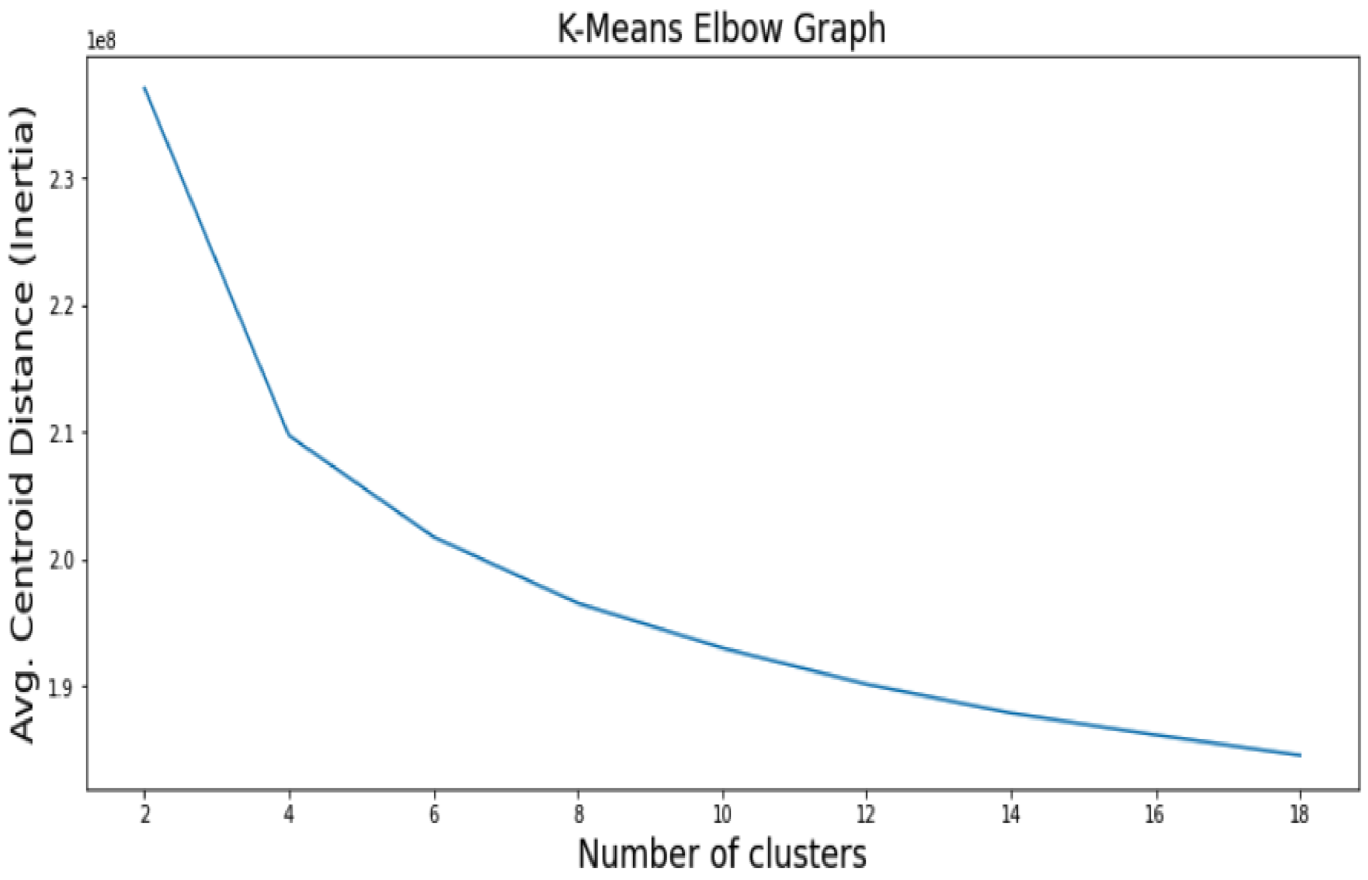


Figure 5.

After a fruitless search for some kind of elbow or kink in this somewhat asymptotically decreasing curve, we opted to choose a value of ***k = 10*** clusters, so that the average centroid distance was low, but not so low that we'd end up with clusters that were too thinly populated. The rate at which the average centroid distance decreases beyond this value for k is only marginally lower, and hence, this arbitrary value was judged appropriate for our purposes.

The PCA and K-Means models that were fit using the general population dataset, were then used to transform the customers dataset. This allowed us to visually compare them side by side by virtue of the differences in the level of overrepresentation or underrepresentation in each of the clusters — Fig. 6.

Code was implemented to dissect the clusters and inspect the main features making up the PCA components with the most impact on each cluster; an excerpt of this dissection is shown below the cluster visuals. This segmentation exercise revealed similar conclusions about the customers to what was already gleaned from the histogram visualizations. But now the degree to which some features were over or underrepresented in the customer dataset could be measured. Among other things, customers were found to be single, high income earners, top earners of advanced age, interested in investing, low in mobility, mostly male, dominant and feisty.

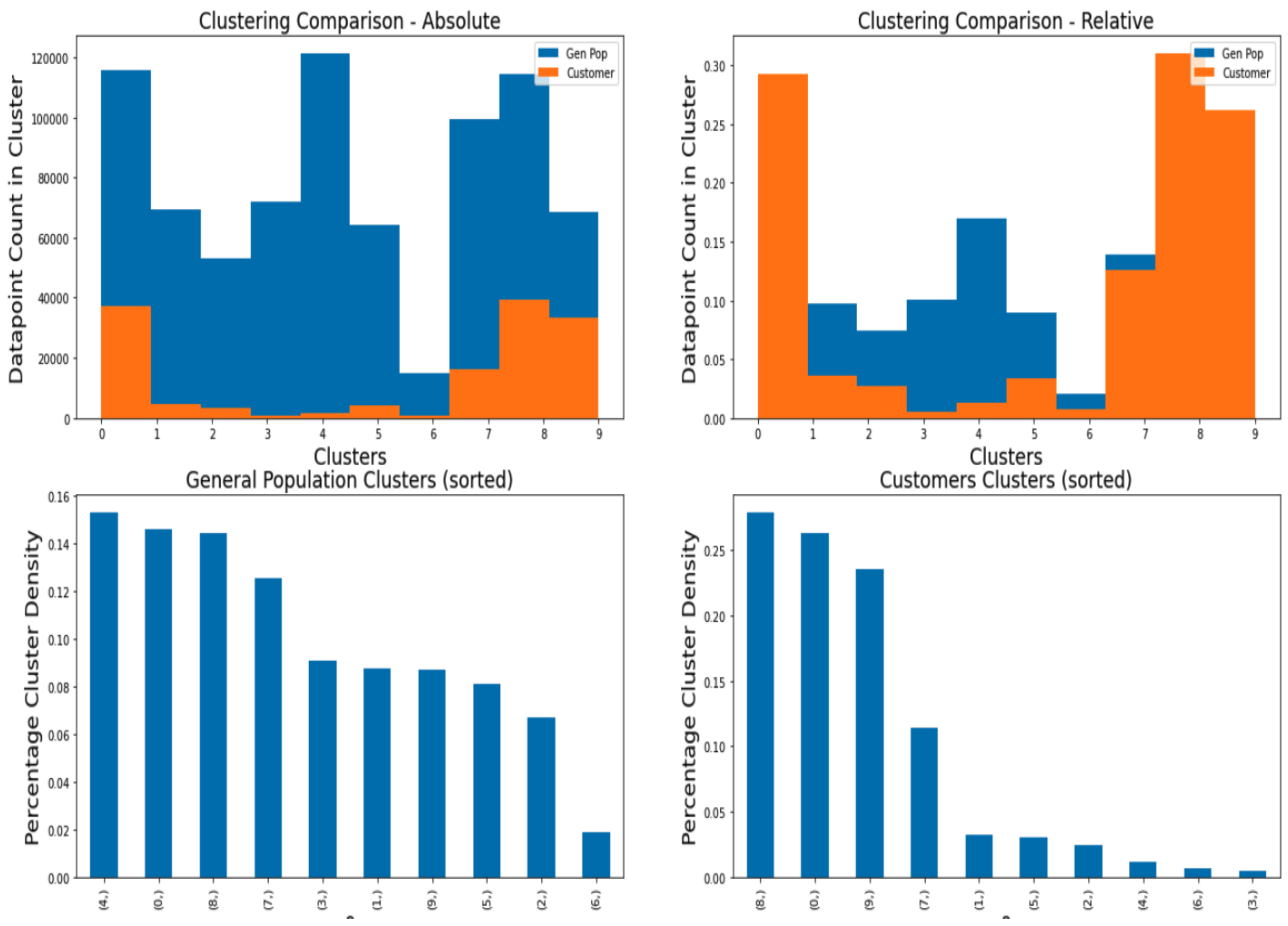


Figure 6.

```
Cluster_9 overrepresents the Customer data by 14.84%
KM_PCA['Cluster_9'] Components :  ------------ Features ------------
                    PCA_3 : ['KBA13_HERST_BMW_BENZ', 'KBA13_MERCEDES', 'KBA13_SEG_OBEREMITTELKLASSE', ...]
                    PCA_0 : ['MOBI_REGIO', 'KBA13_ANTG1', 'PLZ8_ANTG1', 'KBA13_ANTG3', 'PLZ8_ANTG3']
                    PCA_7 : ['KBA13_ALTERHALTER_61', 'KBA13_HALTER_66', 'KBA13_ALTERHALTER_45', ...]
                    PCA_1 : ['KBA05_SEG6', 'KBA05_KRSOBER', 'KBA05_KRSVAN', 'KBA05_KRSZUL', 'KBA05_ANHANG']
                    PCA_9 : ['LP_LEBENSPHASE_FEIN', 'LP_LEBENSPHASE_GROB', 'D19_BANKEN_ANZ_24', ...]

Cluster_8 overrepresents the Customer data by 13.45%
KM_PCA['Cluster_8'] Components :  ------------ Features ------------
                    PCA_0 : ['MOBI_REGIO', 'KBA13_ANTG1', 'PLZ8_ANTG1', 'KBA13_ANTG3', 'PLZ8_ANTG3']
                    PCA_1 : ['KBA05_SEG6', 'KBA05_KRSOBER', 'KBA05_KRSVAN', 'KBA05_KRSZUL', 'KBA05_ANHANG']
                    PCA_5 : ['KBA13_KW_0_60', 'KBA13_KW_61_120', 'KBA13_KMH_210', 'KBA13_BJ_2000', 'OST_WEST_KZ']
                    PCA_4 : ['FINANZ_ANLEGER', 'FINANZ_SPARER', 'FINANZ_UNAUFFAELLIGER', 'CJT_TYP_1', 'RT_KEIN_ANREIZ']
                    PCA_8 : ['KBA13_KMH_140_210', 'KBA13_CCM_1401_2500', 'KBA13_KRSZUL_NEU', 'KBA13_KW_30',
'KBA13_KMH_0_140']

Cluster_4 underrepresents the Customer data by -14.17%
KM_PCA['Cluster_4'] Components :  ------------ Features ------------
                    PCA_4 : ['FINANZ_ANLEGER', 'FINANZ_SPARER', 'FINANZ_UNAUFFAELLIGER', 'CJT_TYP_1', 'RT_KEIN_ANREIZ']
                    PCA_3 : ['KBA13_HERST_BMW_BENZ', 'KBA13_MERCEDES', 'KBA13_SEG_OBEREMITTELKLASSE', ...]
                    PCA_7 : ['KBA13_ALTERHALTER_61', 'KBA13_HALTER_66', 'KBA13_ALTERHALTER_45', ...]
                    PCA_11: ['KBA13_ALTERHALTER_45', 'KBA13_HHZ', 'PLZ8_HHZ', 'KBA13_HALTER_40', 'KBA13_SEG_VAN']
                    PCA_6 : ['ANREDE_KZ', 'SEMIO_KAEM', 'SEMIO_VERT', 'SEMIO_DOM', 'SEMIO_FAM']
Cluster_3 underrepresents the Customer data by -8.61%
KM_PCA['Cluster_3'] Components :  ------------ Features ------------
                    PCA_2 : ['CJT_TYP_2', 'PRAEGENDE_JUGENDJAHRE', 'ONLINE_AFFINITAET', 'CJT_TYP_1', 'FINANZ_SPARER']
                    PCA_4 : ['FINANZ_ANLEGER', 'FINANZ_SPARER', 'FINANZ_UNAUFFAELLIGER', 'CJT_TYP_1', 'RT_KEIN_ANREIZ']
                    PCA_7 : ['KBA13_ALTERHALTER_61', 'KBA13_HALTER_66', 'KBA13_ALTERHALTER_45', ...]
                    PCA_10: ['LP_FAMILIE_FEIN', 'LP_FAMILIE_GROB', 'LP_LEBENSPHASE_GROB', 'ANZ_PERSONEN', ...]
                    PCA_8 : ['KBA13_KMH_140_210', 'KBA13_CCM_1401_2500', 'KBA13_KRSZUL_NEU', 'KBA13_KW_30', ...]
```

Our so-called "unsupervised learning" algorithms were not entirely without supervision; after all, the number of PCA components to retain (175), and the number of clusters to use (10), were decided with human intervention, trade-offs mostly between information and pragmatism. The methods did reveal interesting insights from the data, and the *Scikit-learn* implementations were easy to apply and well documented; but running them was incredibly time consuming.

The next part of the project — supervised learning, consisted of training a classifier model with the mailout data, and fine-tuning some of its parameters in order to make predictions that would eventually be submitted into a Kaggle Competition. At this late stage of the project, it became apparent that the prior decision to discard the rows with excessive missing data was suboptimal. It resulted in different treatment of the datasets used for unsupervised and supervised ML techniques. This is because the competition expected all the rows extant in the original data files to be submitted for scoring, even those missing beyond 65% of their entries. Apart from conserving all the rows in the training and test mailout datasets, all prior data wrangling steps were carried out exactly as was done for the first part of the project, albeit with a much greater number of imputations.

Inspecting the mailout labels we see an imbalance (Fig. 7); there are hardly any customers in it. This renders some of the usual metrics used in gauging the usefulness of models, such as accuracy, rather useless. Indeed, with such high imbalance, any model predicting "non-customer" for every observation would still be highly accurate, but of little use.

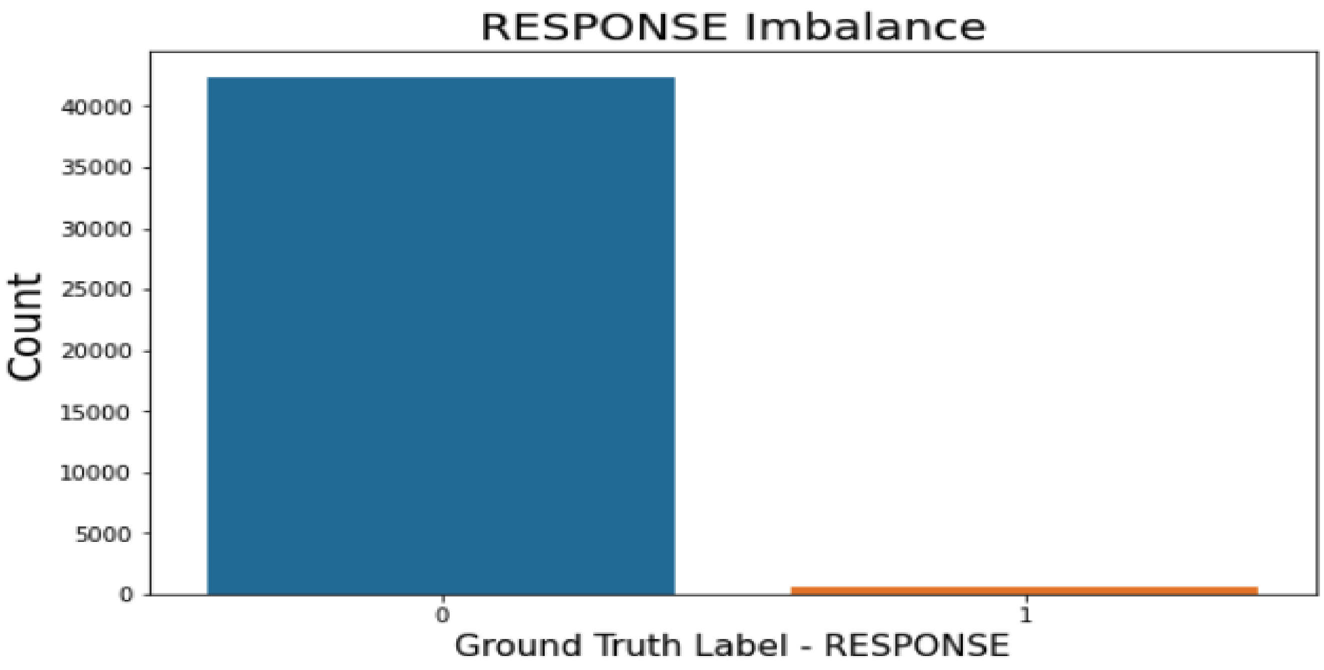


Figure 7.

Rather than accuracy, we want a metric that accounts for false positives and false negatives. Since we are dealing with a marketing problem, we are more inclined to tolerate false positives (marketing to non customers at the risk of annoying them), rather than accept false negatives (missing out on the potential revenue from misidentified true customers). This means that we care more about Recall, than we do about Precision, where:

***Recall** (True Positive Rate or **TPR**) = true positive/(true positive + false negative)*, and

***Precision** = true positive/(true positive + false positive)*.

The ROC (Receiver Operating Characteristic) Curve is a plot of Recall or True Positive Rate vs False Positive Rate, where:

*False Positive Rate (**FPR**) = false positive/(false positive + true negative)*.

Recall captures the fraction of correctly predicted customers to the total pool of customers. Similarly, FPR captures the fraction of incorrect predictions to the total pool of non-customers. By plotting TPR vs FPR with ROC, we can visualize a classifier's ability to tell customers from non-customers. By measuring the Area Under Curve (ROC AUC), we then have a metric that gauges the probability that the scores given by the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. ROC AUC provides us with a scoring mechanism to tell how well a model distinguishes customers from non-customers; therefore, ROC AUC was picked as our metric of choice for model selection and parameter tuning.

Several classifiers were tried, their ROC curves plotted (Fig. 8), and the resulting ROC AUC score computed in order to discern the most promising model and subsequently tune its parameters. In considering which binary classifier to use, we explored *Logistic Regression*, as well as ensemble techniques that generate multiple hypotheses using the same base learner like *Random Forest*, *Adaptive Boosting*, *Gradient Boosting* and the *XGBoost* variant. Boosting refers to the general idea of producing accurate predictions by combining rough and inaccurate rules-of-thumb. It is a sequential process that in some cases tends to over-fit the training data. In order to avoid overfitting, we constrained the tuning of parameters to ranges where the models' learning curves gap between the train and test results weren't too wide (Fig. 9).

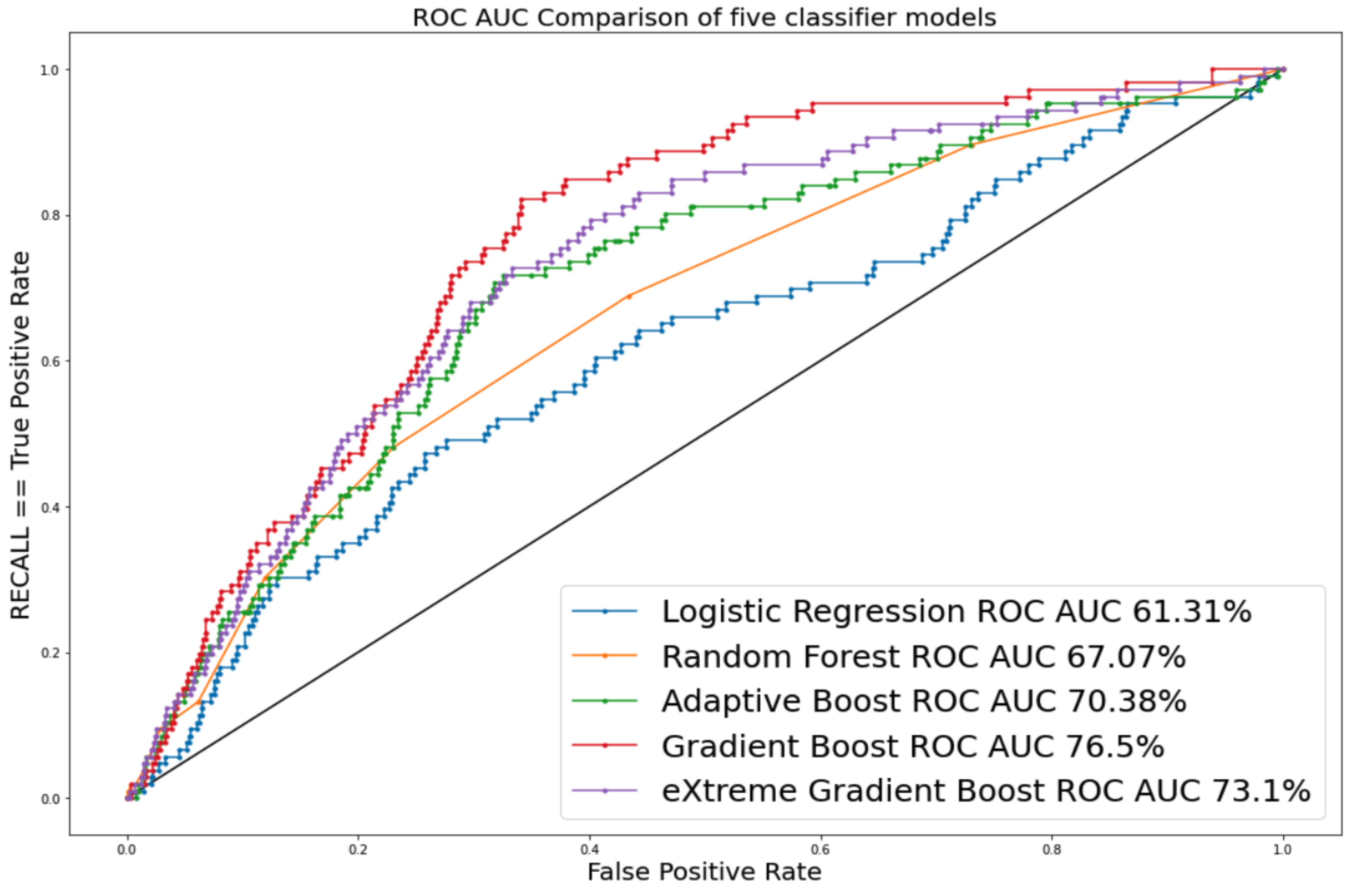


Figure 8.

Figure 8 depicts the ROC curves and ROC AUC scores of the five baseline models considered for supervised learning. The Gradient Boosting algorithm emerged as the clear choice for tuning and predicting; its ROC curve is markedly above the rest, and the score is several percentage points ahead of the next best model. The next step consisted of tuning the parameters of this model.

The parameters of the Gradient Boosting model can be divided into 3 categories: Tree-Specific Parameters, Boosting Parameters, and other miscellaneous parameters for overall functioning. We focused first on a few of the most important tree-specific, and boosting parameters, namely, the number of trees (*n_estimators*), tree depth (*max_depth*), learning rate (*learning_rate*), and the number of randomly selected features to consider while searching for a split (*max_features*). After this tuning, with these four parameters fixed at optimal levels, we tuned three further parameters, namely, *subsample*, *min_samples_leaf*, and *min_samples_split*.

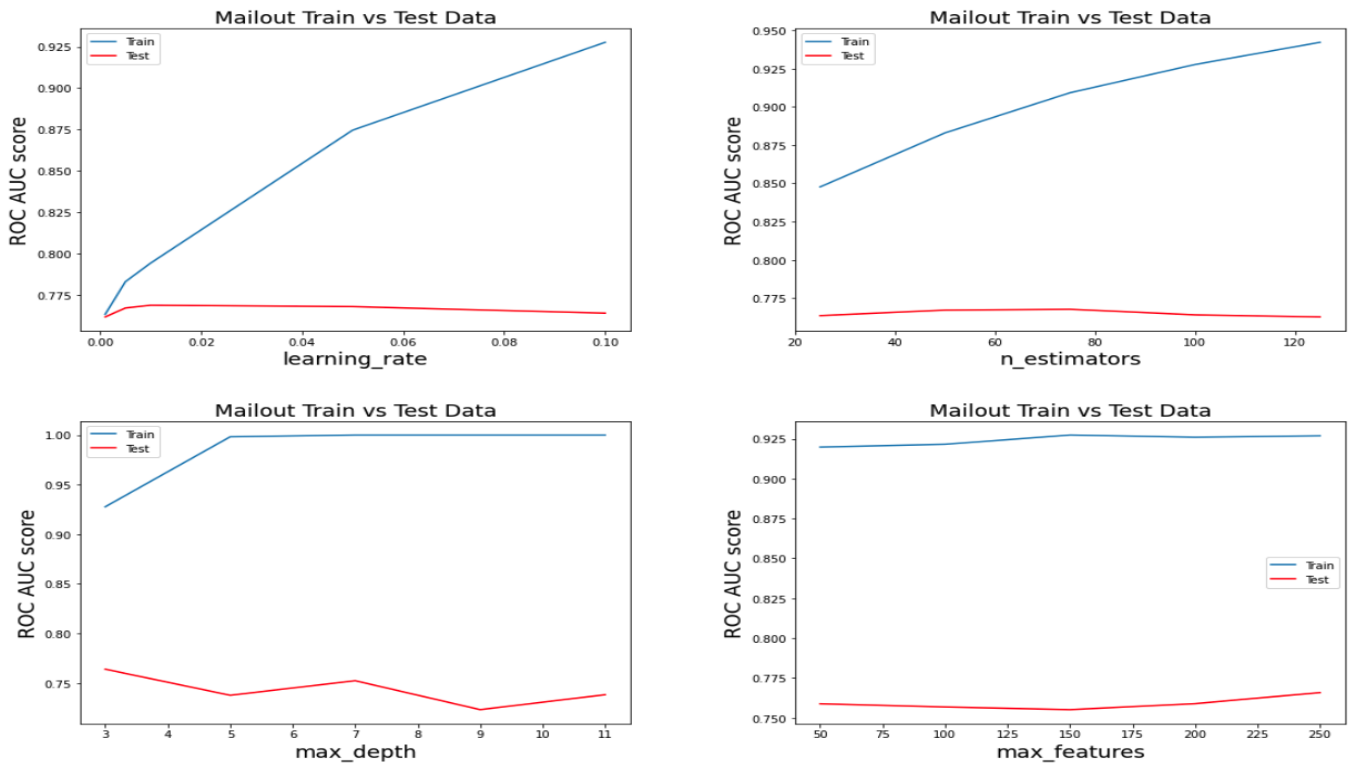
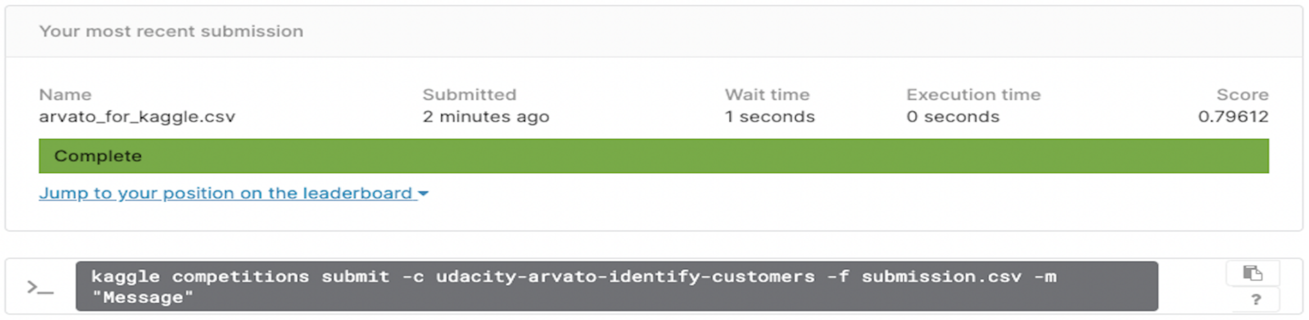


Figure 9.

With our Gradient Booster tuned, we measured only marginal improvement in predicting power over our baseline model, 78.55% vs 76.5%. Correspondingly, our best submission to the kaggle competition was scored at merely 79.61%. Details of the tuned parameter values, and the output from cross validations are available in the notebook *Capstone Arvato.ipynb*.


Your most recent submission

Name	Submitted	Wait time	Execution time	Score
arvato_for_kaggle.csv	2 minutes ago	1 seconds	0 seconds	0.79612

Complete

Jump to your position on the leaderboard

```
kaggle competitions submit -c udacity-arvato-identify-customers -f submission.csv -m
"Message"
```

Conclusions: In this project we examined unsupervised and supervised ML techniques to identify potential customers for a targeted advertising campaign. There were a number of challenges concerning the data: First, there was a significant number of barren rows with more than 50% blank entries. More concerning, however, was the fact that the percentage of missing data was strangely structured as a ladder. Without a clear understanding of this phenomenon we are left to wonder whether any analysis makes sense for purposes beyond the purely academic. Another unexpected characteristic of the datasets was the exorbitant amount of highly correlated and perhaps not so relevant features. Features such as *"the share of cars with less than 59 KW engine power"*, and *"the share of car owners between 46 and 60 within the Zip Code"* abound. Perhaps it is wise to guide a marketing campaign based on such information about individuals; we must assume so for this project. Also, the EXCEL spreadsheets detailing the meaning behind the column names left a large portion of the columns unexplained; this was disappointing.

After proceeding with the data preprocessing steps, namely, removing rows with missing data, dropping some columns, encoding non-numerical features, imputing missing values in columns, handling outliers and normalizing, several visualizations were produced to inspect and compare the features' histograms of both the general population and customer datasets. What transpired was a first glance at the information we were looking for, i.e., the characterization of customers by demographic data. During this Data Exploration and Preprocessing step, we developed a maverick approach to imputation, taking advantage of the categorical nature of the features. Details of the technique, along with the code used to implement it, are to be found in the files included in the GitHub Repository for this project. The unsupervised learning part of the project achieved a dimensionality reduction of 50% of the columns by sacrificing only 10% of the information, and subsequent segmentation by K-Means clustering using 10 clusters. This segmentation revealed similar conclusions about the customers. But now the degree to which features were overrepresented or underrepresented in the customer dataset could be measured.

The last part of the project consisted of selecting a classifier and tuning some of its parameters using a metric appropriate for the highly imbalanced mailout campaign train and test datasets. Using ROC AUC, the Gradient Boosting Classifier was selected and a number of its boosting and tree-specific parameters were tuned. A score of 79.6% was attained in the Kaggle Competition. As a final step, a last ditch effort was made to improve this score by considering the effect of a dimensionality reduction step on the mailout data before classification. The motivation was a belief that although some information would be lost, the PCA procedure would get rid of data noise in the form of superfluous features. This was a wasted effort that made matters worse. Be it as it may, the idea had didactic value, if only to learn the consequences of stepping out of the bounds of ML orthodoxy. The project was fun; a lot was learned not only with regards to ML techniques, but also about the tools of the trade as applied to a real-life problem. I would recommend this course, and this project to anyone genuinely curious about Machine Learning.

As far as improvement, there were a number of ideas that emerged as after-thoughts. With hindsight they appear promising as possible ways to up the score a bit. First, rather than obsess about the ladder structure of the missing data, and instead of discarding any missing rows, one could attempt to split the datasets into as many data frames as there are rungs, one with rows missing more than 70% of the entries, another with those missing more than 60%, and so on, as dictated by the ladder shown in Figure 1. Then, a simple comparison of the statistics across subsets of data would immediately reveal if the conjecture that the data was sourced with mixed origins was right or wrong. This knowledge would then guide the project to proceed with several separate unsupervised as well as supervised ML techniques, or one large one as we did. But at least this way we would have the certainty that the data could be analyzed without splitting it. Second, there were a number of features, such as *CAMEO_DEUINTL_2015*, with bins showing inner structure, i.e., features containing features. It is probably best to split these features to end up with a higher number of columns, each one possessing truly independent information.

Third, the columns *CUSTOMER_GROUP*, *ONLINE_PURCHASE*, and *PRODUCT_GROUP* in the customer dataset were entirely ignored in our analysis. Here too, it is sensible to consider performing three separate analyses split along these columns. All these suggestions for improvement would mean multiple separate analyses which in reality would be extremely time consuming, but thorough. In the end, it would probably be another trade-off between information and pragmatism.

References

Jonathan Kropko, Ben Goodrich, Andrew Gelman & Jennifer Hill, (2013). Multiple Imputation for Continuous and Categorical Data:Comparing Joint and Conditional Approaches, page 2., http://www.stat.columbia.edu/~gelman/research/published/MI_manuscript_RR.pdf

Samantha Miranda, (2020). Investigation of Multiple Imputation Methods for Categorical Variables., https://dc.etsu.edu/cgi/viewcontent.cgi?article=5204&context=etd

Ian R White, Rhian Daniel, and Patrick Royston, (2010). Avoiding bias due to perfect prediction in multiple imputation of incomplete categorical variables. Computational Statistics & Data Analysis., https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3990447/

Kenneth Leung, (2021). Assumptions of Logistic Regression, Clearly Explained. https://towardsdatascience.com/assumptions-of-logistic-regression-clearly-explained-44d85a22b290

Niklas Donges, (2021). A Complete Guide to the Random Forest Algorithm. https://builtin.com/data-science/random-forest-algorithm

Avinash Naviani, (2018). AdaBoost Classifier in Python. https://www.datacamp.com/community/tutorials/adaboost-classifier-python

Akash Desarda, (2019). Understanding AdaBoost. https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe

Jason Brownlee, (2016). A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning. https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

Manish Pathak, (2019). Using XGBoost in Python. https://www.datacamp.com/community/tutorials/xgboost-in-python

Aarshay Jain, (2016). Complete Machine Learning Guide to Parameter Tuning in Gradient Boosting (GBM) in Python. https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/