Alberto de Blas

May 19th, 2020

Capstone Project

Machine Learning Engineer Nanodegree

Customer Segmentation and Supervised Learning Model for Arvato

Definition

Project Overview

The purpose of the project is to analyze demographics data for customers of a mail-order sales company in Germany, comparing against demographics information for the general population.

This project pretends to be also a PoC/demo of the Sagemaker capabilities.

Problem Statement

Firstly it will be necessary to use unsupervised learning techniques to perform customer segmentation, comparing to the general population to observe which characteristics best describe the customer base of the company and make a customer persona.

Secondly we will use data from a previous campaign and a model to predict which persons are most likely to convert into becoming customers for the company.

Scoring

For the customer segmentation we will not have any score but to choose a good number of groups we will look at the average centroid distance between cluster points and a centroid.

For the customer prediction we will aim for the best ROC/AUC metric and submit the results to the Kaggle competition to confirm if we achieve a good result.

ROC curve is a graph showing the performance of a classification model at all classification thresholds and AUC is the Area Under the Curve. AUC provides an aggregate measure of performance across all possible classification thresholds.

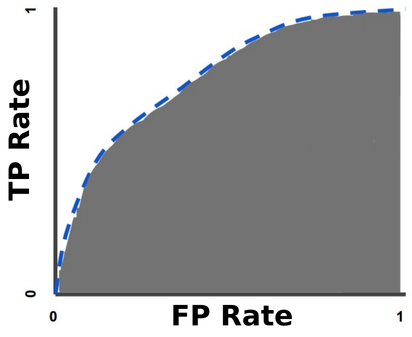


Figure 1: ROC/AUC

As will be shown later, the prediction target has very imbalanced targets and accuracy will not describe well if the model has a good level of prediction.

Analysis

Data Exploration and Visualization

There are four data files, two for the customer segmentation and two for the customer prediction.

- *Udacity_AZDIAS_052018.csv*: Demographics data for the general population of Germany; 891 211 rows x 366 features

- *Udacity_CUSTOMERS_052018.csv*: Demographics data for customers of Arvato; 191 652 rows x 369 features, with customer_group, online_purchase and product group columns

- *Udacity_MAILOUT_052018_TRAIN.csv*: Demographics data for individuals who were tragets of a marketing campaign; 42982 rows x 367 columns, with response column

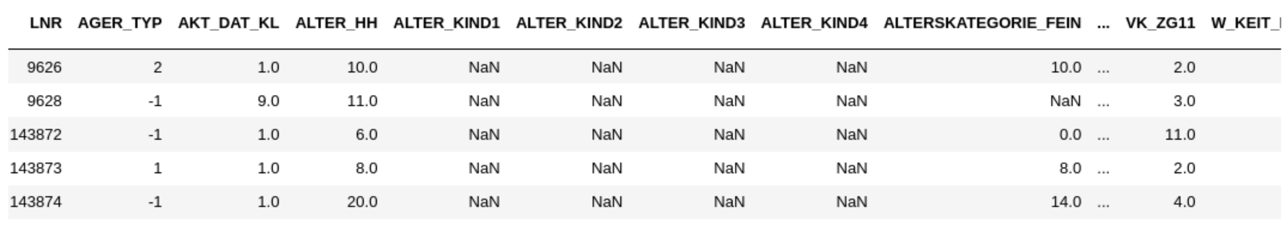
- *Udacity_MAILOUT_052018_TEST.csv*: Demographics data for individuals who were targets of a marketing campaign; 42833 x 366, without response column

There are to additional Excel files that help to understand the domain of the data:

- *DIAS Information Levels - Attributes 2017.xlsx*: Description of the features

- *DIAS Attributes - Values 2017.xlsx*: Possible values of each feature

The majority of the variables are categorical variables with a numeric value and contain ordinal data so can be left in one column and it is not necessary to split them.



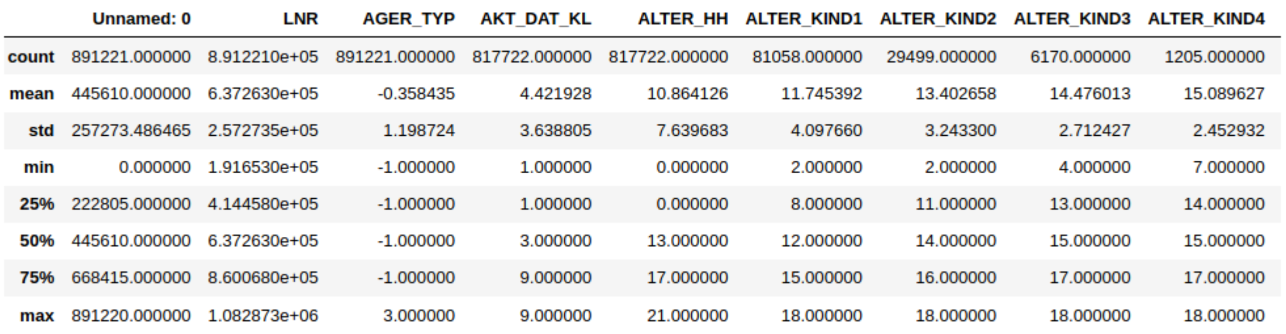
LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	...	VK_ZG11	W_KEIT_
9626	2	1.0	10.0	NaN	NaN	NaN	NaN	10.0	...	2.0	
9628	-1	9.0	11.0	NaN	NaN	NaN	NaN	NaN	...	3.0	
143872	-1	1.0	6.0	NaN	NaN	NaN	NaN	0.0	...	11.0	
143873	1	1.0	8.0	NaN	NaN	NaN	NaN	8.0	...	2.0	
143874	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	...	4.0	

Figure 2: Azdias sample values

Also there are some date values that have to be converted to a numeric value, *EINGEFUEGT_AM*

There is a categorical variable that not contains ordinal data but only has to possible values so it can be left in one column after encoding into a numerical value, *OST_WEST_KZ*

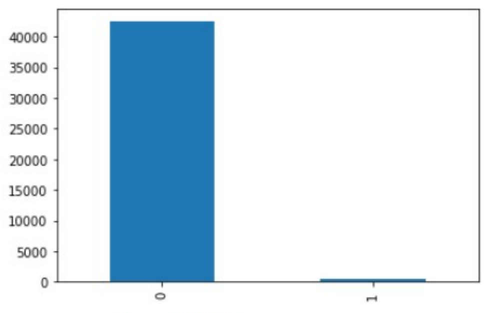
Lastly there are a group of 19 categorical variables that the data is not ordinal and contain more than two different values. It will be necessary to expand these features performing one hot encoding.



	Unnamed: 0	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4
count	891221.000000	8.912210e+05	891221.000000	817722.000000	817722.000000	81058.000000	29499.000000	6170.000000	1205.000000
mean	445610.000000	6.372630e+05	-0.358435	4.421928	10.864126	11.745392	13.402658	14.476013	15.089627
std	257273.486465	2.572735e+05	1.198724	3.638805	7.639683	4.097660	3.243300	2.712427	2.452932
min	0.000000	1.916530e+05	-1.000000	1.000000	0.000000	2.000000	2.000000	4.000000	7.000000
25%	222805.000000	4.144580e+05	-1.000000	1.000000	0.000000	8.000000	11.000000	13.000000	14.000000
50%	445610.000000	6.372630e+05	-1.000000	3.000000	13.000000	12.000000	14.000000	15.000000	15.000000
75%	668415.000000	8.600680e+05	-1.000000	9.000000	17.000000	15.000000	16.000000	17.000000	17.000000
max	891220.000000	1.082873e+06	3.000000	9.000000	21.000000	18.000000	18.000000	18.000000	18.000000

Figure 3: Azdias descriptive statistics

In the mailout_train data set the target column is named RESPONSE, we can see that has high imbalanced values, so over the total of people of the campaign only a little fraction has reacted to it.



```
0    42430
1      532
Name: RESPONSE, dtype: int64
```

Figure 4: Target RESPONSE value distribution

Null distribution:

Although null values are spread among the majority of features there are a few features that contains more nulls than the average as can be seen in the figures below:

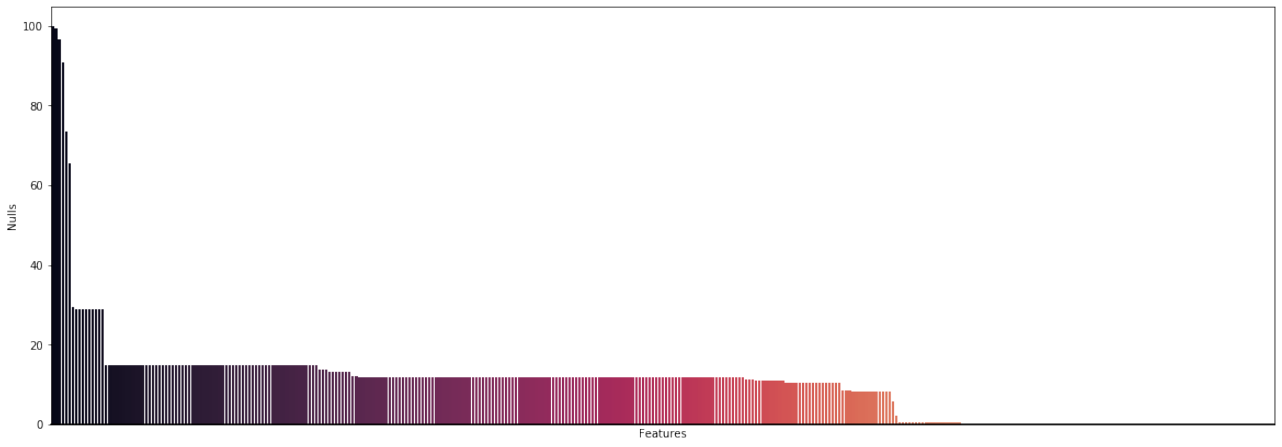


Figure 5: Null distribution for Azdias

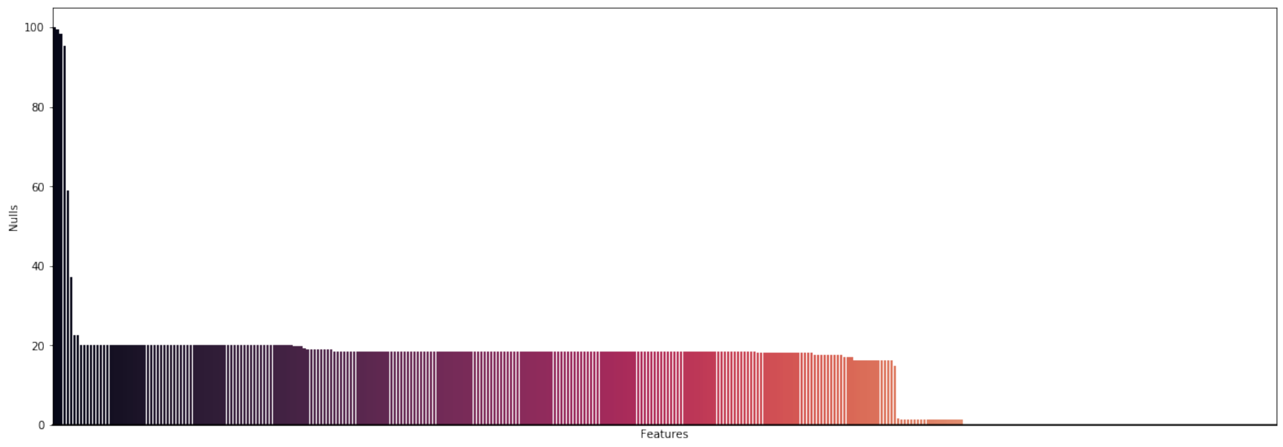


Figure 6: Null distribution for Mailout_Train

Variance distribution

Plotting the variance distribution can be seen that there are features with extreme variance value that can be discarded for the model and it is almost identical in both training data sets.

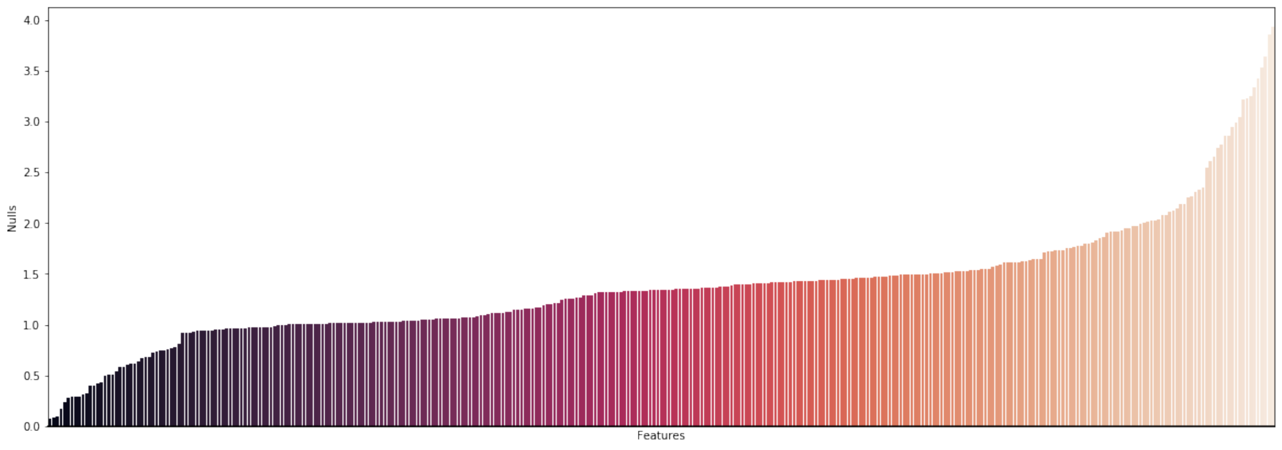


Figure 7: Variance distribution for Azdias

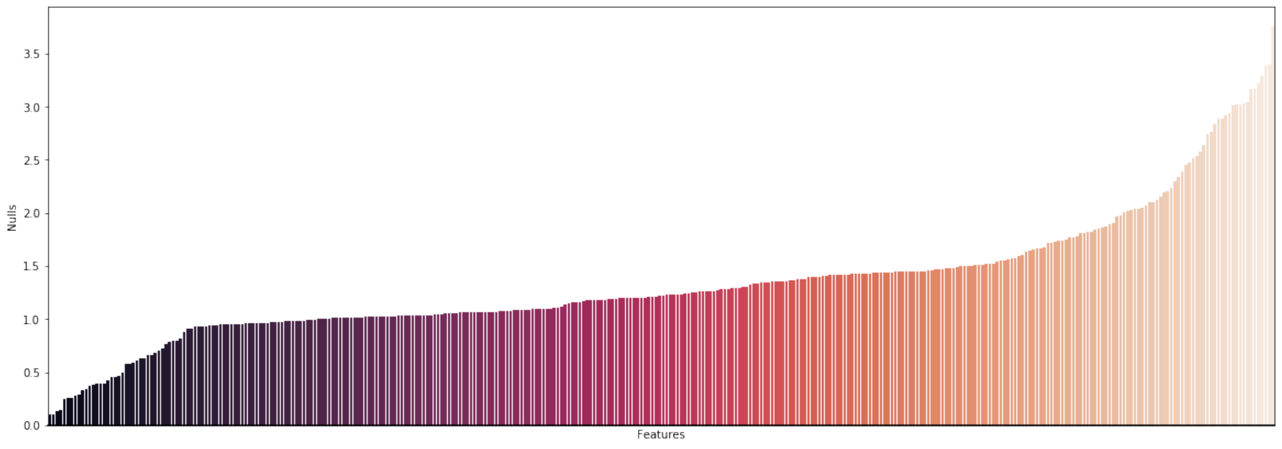


Figure 8: Variance distribution for Mailout_Train

Algorithms and Techniques

PCA

We have a large number of features and it is necessary to reduce the amount of features and reduce dimensionality.

Benefits of dimension reduction:

- It reduces the time and storage space required.
- Removal of multi-col-linearity improves the interpretation of the parameters of the machine learning model.
- It becomes easier to visualize the data when reduced to very low dimensions such as 2D or 3D.
- It avoids the curse of dimensionality.

The algorithm selected is PCA with the AWS Sagemaker implementation. Even though, due to the number of rows of the dataset, would not be necessary to use the Sagemaker PCA estimator and could be processed directly in the notebook, as stated before this project pretends to demo of its functionalities and a way to learn more about them.

PCA helps us compute the Principal components in data. Principal components are basically vectors that are linearly uncorrelated and have a variance with in data. From the principal components top *p* is picked which have the most variance.

The algorithm finds the line that maximizes the variance (the average of the squared distances from the projected points to the origin). The second principal component is calculated in the same way, with the condition that it is uncorrelated with (i.e., perpendicular to) the first principal component and that it accounts for the next highest variance.

We will select a number of components containing a part of the total explained variance based on the explained variance vs number of features graph.

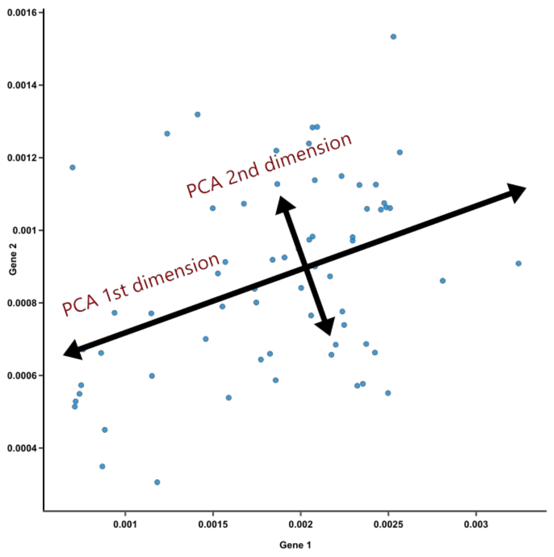


Figure 9: PCA Components

K-Means

For the customers segmentation we will use the K-Means algorithm with the Sagemaker implementation.

K-Means clustering is a method of vector quantization that aims to partition *n* observations into *k* clusters in

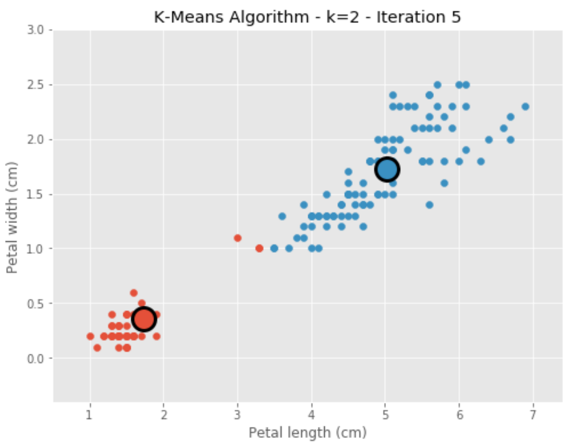


Figure 10: K-Means step

which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.

K-Means starts by randomly defining *k* centroids. From there, it works in iterative (repetitive) steps to perform two tasks:

1. Assign each data point to the closest corresponding centroid, using the standard Euclidean distance. In layman's terms: the straight-line distance between the data point and the centroid.
2. For each centroid, calculate the mean of the values of all the points belonging to it. The mean value becomes the new value of the centroid.

Once step 2 is complete, all of the centroids have new values that correspond to the means of all of their corresponding points. These new points are put through steps one and two producing yet another set of centroid values. This process is repeated over and over until there is no change in the centroid values, meaning that they have been accurately grouped. Or, the process can be stopped when a previously determined maximum number of steps has been met.

The approach will be to apply K-Means with different number of groups and to evaluate with the help of an elbow graph the most suitable number of groups to use.

Finally for the customer segmentation we will apply K-means with the selected group number to the general population data and the customer data and analyze the characteristics of the groups that have a larger number of customers to make a profile based on the features that makeup the components which have more relevance in that groups.

<u>Gradient Boosting</u>

For the customers classification prediction we will be using Gradient Boosting with an image with XGBoost in a docker container ready to use with Sagemaker Estimator.

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

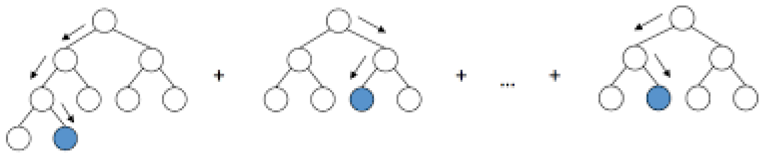


Figure 11: Gradient Boosting sequence

It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model in order to minimize the error. How are the targets calculated? The target outcome for each case in the data depends on how much changing that case's prediction impacts the overall prediction error:

- If a small change in the prediction for a case causes a large drop in error, then next target outcome of the case is a high value. Predictions from the new model that are close to its targets will reduce the error.
- If a small change in the prediction for a case causes no change in error, then next target outcome of the case is zero. Changing this prediction does not decrease the error.

The name *gradient boosting* arises because target outcomes for each case are set based on the gradient of the error with respect to the prediction. Each new model takes a step in the direction that minimizes prediction error, in the space of possible predictions for each training case.

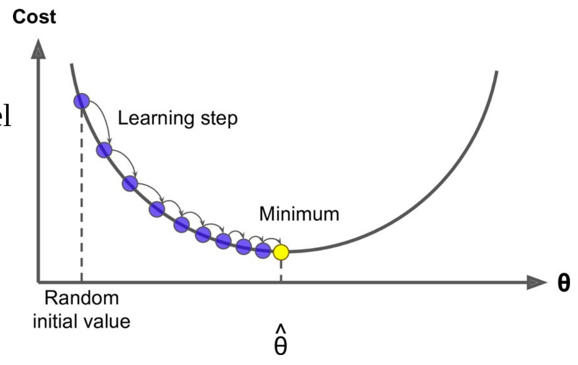


Figure 12: Gradient Boosting steps

Advantages & Disadvantages

Advantages:

- Often provides predictive accuracy that cannot be beat.
- Lots of flexibility - can optimize on different loss functions and provides several hyperparameter tuning options that make the function fit very flexible.
- No data pre-processing required - often works great with categorical and numerical values as is.
- Handles missing data - imputation not required.

- GBMs will continue improving to minimize all errors. This can overemphasize outliers and cause overfitting. Must use cross-validation to neutralize.
- Computationally expensive - GBMs often require many trees (>1000) which can be time and memory exhaustive.
- The high flexibility results in many parameters that interact and influence heavily the behavior of the approach (number of iterations, tree depth, regularization parameters, etc.). This requires a large grid search during tuning.
- Less interpretable although this is easily addressed with various tools (variable importance, partial dependence plots, LIME, etc.).

<u>Benchmark</u>

To have a reference point to prove our that our intuition that XGBoost will have a good performance comparing to other algorithms, and to prove that is worth to use it instead more simpler algorithms we will train a Logistic Regression model.

Also we will the recently released in AWS Ireland availability zone Sagemaker Studio, with the experiments functionality to have another reference and test it.

Engineering techniques

Due to the fact that we will have to separate notebooks and it is important the code re-usability and to have all the functions well organized, the common functionality of both notebooks is implemented in several Python modules that are imported in the notebook.

This modules are:

data_loading.py: Contains functions to load the data into the notebook

data_cleaning.py: Contains functions to clean the data

data_visualization.py: Contains functions to plot graphs

feature_selection.py: Contains functions to find feature importance and to fit the model

csv_transformation.py: Contains functions to convert data from files to dataframes or vice-versa

memory_dataframe.py: Contains functions to get information about memory usage of the objects

model_selection.py: Contains functions to know which is the most convenient model or hyperparameters to use

import_management.py: Utility function to reload custom libraries without the need to restart the kernel

Methodology

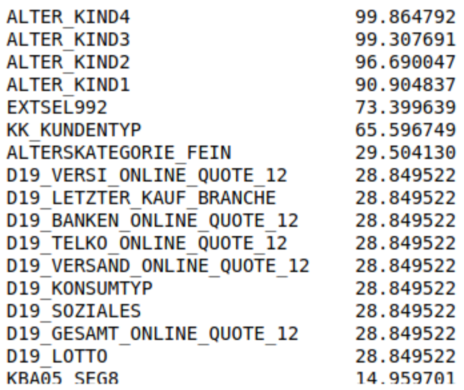
Data pre-processing

Drop columns with nulls

The first step that we have done is to reduce the number of features dropping the ones that have a high number of nulls. It is important in this case to get rid of them because the data occuppies a high amount of memory and the budget for AWS EC2 notebook instances is tight.

As shown in the null distribution figures, there are a few features that have lots of nulls.

We will drop the columns that have more than 25% of null values from the azdias data set and the mailout_train and then



```
ALTER_KIND4                    99.864792
ALTER_KIND3                    99.307691
ALTER_KIND2                    96.690047
ALTER_KIND1                    90.904837
EXTSEL992                      73.399639
KK_KUNDENTYP                   65.596749
ALTERSKATEGORIE_FEIN           29.504130
D19_VERSI_ONLINE_QUOTE_12      28.849522
D19_LETZTER_KAUF_BRANCHE       28.849522
D19_BANKEN_ONLINE_QUOTE_12     28.849522
D19_TELKO_ONLINE_QUOTE_12      28.849522
D19_VERSAND_ONLINE_QUOTE_12    28.849522
D19_KONSUMTYP                  28.849522
D19_SOZIALES                   28.849522
D19_GESAMT_ONLINE_QUOTE_12     28.849522
D19_LOTTO                      28.849522
KBA05_SEG8                     14.959701
```

Figure 13: Nulls % in Azdias features

we will drop the same columns from the customers data set and the mailout_test.

Drop columns with low variance and high (all the columns different) variance

The next step to continue getting rid of features that are not useful and fill the memory space is to drop the columns that have little variance and the ones that have a very high variance, so high that all its values are different. Based on the graph of the previous section, there are some values heading and tailing that can be dropped.

Convert to NaN not valid values

Inspecting the values taken of the remaining features there are some of them that contain invalid values that should be treated as missing values.

This is the case of CAMEO_INTL_2015 with value XX

CAMEO_DEUG_2015 with value X

EINGEFUEGT_AM with value NaT, that is a timestamp and Pandas isna() function do not recognize as a NaN

Discard the rows which have less than 80% values (at least 290 non null values)

Even with a careful imputing, chances that samples with more than 20% nulls reflect real cases are low so we drop them.

Convert to categorical values

This step is very important to reduce the amount of memory used, specially by azdias dataset. Reviewing the data description and possible values from the features we can make a list of all the categorical values going from 2.5gb to 600mb.

Data encoding

It is necessary to convert into a numerical value timestamp column EINGEFUEGT_AM, in this case parsing YEARMONTHDAY. Another approach could be to calculate the timedelta in years between each value and the current time.

Impute nulls

In the first part the approach is to impute median to each null of the numerical features and to impute mode to each null of the categorical features.

For the second part, prediction, the approach has been to impute median with an Iterative Imputer, that performs MICE to impute nulls for numerical features and impute mode to each null of the categorical features.

One thing that I will want to do is to try both approaches and compare the results to see which one is better in each case.

One hot encoding

With the categorical variables that do not have an order between the values we perform one hot encoding with Pandas → getDummies function. It generates a new binary feature for each value.

Drop low variance columns second iteration

After performing one hot encoding we have a new set of features that potentially could have low statistical relevance because they are created from a value of a features that rarely appears. So we perform another drop of features with low variance.

Data scaling

With a MinMaxScaler from Sci-kit we scale all the features between 0 and 1. It is very important for PCA and K-Means to have the features with the same scale because the computation of the distance from the feature to the component or to the centroid would be different and biased with values in different scales.

Model training

PCA

We perform a training of the PCA model with a high number of features, in this case 300, to know how many components would be necessary to maintain to have most of the information but with less features.

From the graph we can see that more or less to the 125th feature the curve begins to flatten more and more.

We take the number of components that have the 90% of the explained variance, and these are 134 components and refit the PCA model.

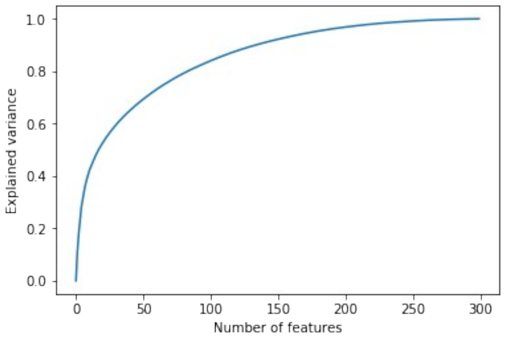


Figure 14: PCA Explained Variance vs Number of Features

K-means

The next step will be to perform K-means multiple times, between 2 and 12 groups to show in a graph how many groups should be the best number to train K-means. Plotting Sum of Squared Errors (distortion) vs number of groups and selecting the group number when the curve begins to flatten.

In our case with 7 groups we observe that the curve begins to lose steepness.

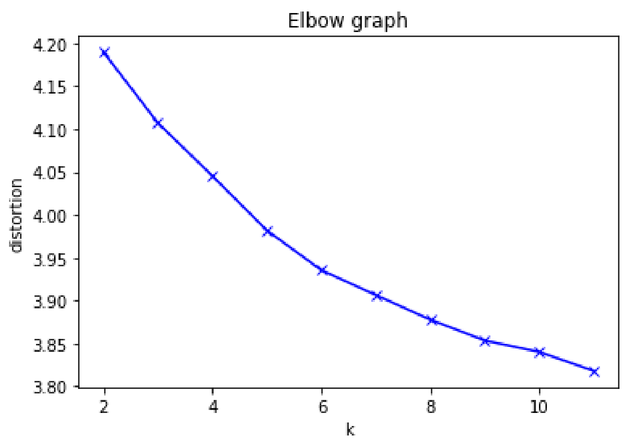


Figure 15: Elbow graph

We use the same fitted model to group data from azdias and customers dataset and we can plot the number of samples in each group.

It can be seen that most of the customers have been classified in the group 1 and 5.

Now we need to know which are the features that influence this grouping to have a better understanding about the customers.

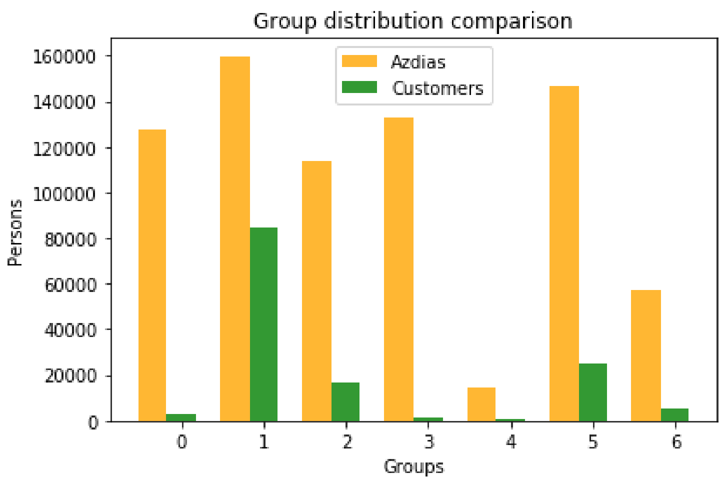


Figure 16: General Population vs Customers

Plotting a heat map of the 10 most important components and the clusters we can see that component 132 is important for the classification in group 1 and components 132 and 133 are important for group 5 classification.

Next action we can do is to plot the feature composition and we will show the results in the results.

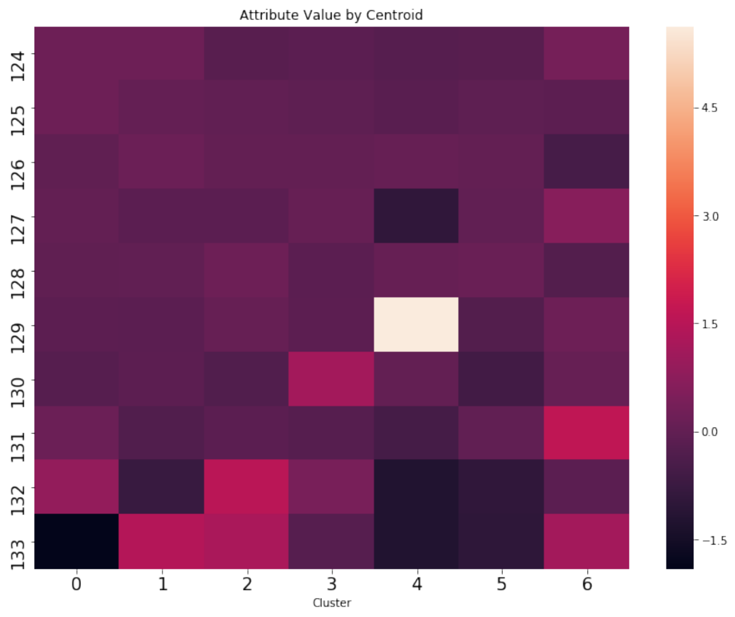


Figure 17: Components relevance for cluster

Logistic Regression/Gradient Boosting

Before training the model we can do an uni-variate analysis about which are the features that are most correlated with the target.

We can see that types of consumer are the variables most positively and negatively correlated.

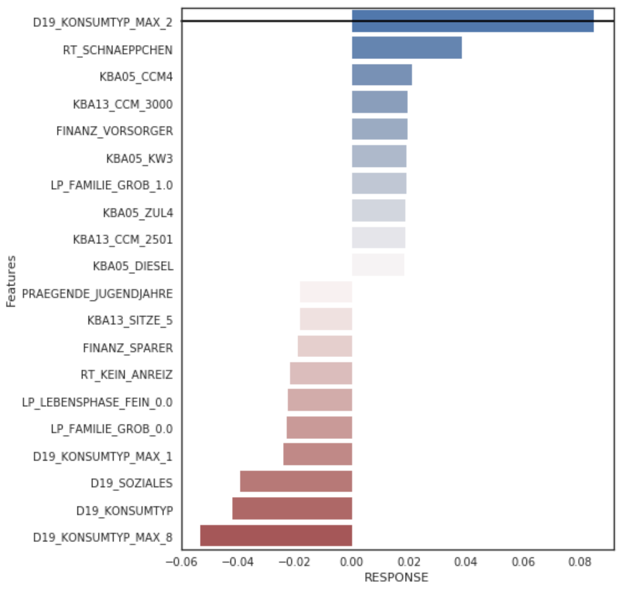


Figure 18: Feature correlation with target

To perform a binary classification to predict which persons will replay to the mailing action we will train a Linear Learner and a XGBoost model with the cleaned data.

Before the training we will use HyperparameterTuner to look for the best parameters to use. This utility applies Bayesian Optimization to find a local maximum for binary classification in our case fitting the model several times, each time adjusting the hyperparameters in the direction that points the Bayesian sequence.

- Linear Learner hyperparameters
  - predictor_type: binary_classifier, due to the only two possible values
  - normalize_data: True, the data is already normalized so really this hyperparameter does not affect the result
  - wd: L2 regularization parameter, fixed to 0.1 because when given a range tuning was overfitting due to the imbalanced classes.
  - L1: L1 regularizatoin parameter, fixed to 0.1 because when given a range occurred the same as with wd
  - learning_rate: size of the steps of each iteration
  - mini_batch_size: Divide transformation to avoid reach the MaxPayload.
  - positive_example_weight_mult: Weight assigned to positive examples, specially useful in cases as this with imbalanced classes.
- XGBoost hyperparameters
  - objective: binary:logistic, due to the two only possible values
  - early_stopping_rounds: number of rounds without improving to stop, fixed to 10
  - num_round: number of rounds to run the training, range given between 6 and 12 to avoid too much computing
  - max_depth: maximum depth of the trees, range given between 2 and 8 to tune

  - eta: step size shrinkage to avoid overfitting, range given between 0.1 and 0.5
  - gamma: minimum loss reduction required to make a further partition on a leaf node, range given between 2 and 6.
  - min_child_weight: minimum weight to partition a leaf node, a low value can lead to overfitting and a high one to underfitting, given range between 3 and 9
  - subsample: subsample ratio of the training instance, how many data is randomly collected, given range between 0.5 and 1.0

After that we execute a batch prediction for the mailout_test data set and submit the results to Kaggle.

Observations during the implementation

The implementation of the capstone has faced several difficulties that it is interesting to note:

- Encoding problems with Sagemaker library: `session.upload_string_as_file_body` uploads to s3 the data in a text format that is not text/csv, or at least, Sagemaker transformer function does not take it as a valid format. The aim to use this function was to store the data directly from notebook memory to S3 without the need to store to the notebook space and then copy to S3, to avoid the assignation of more space to the notebook instance. To notice that the encoding problem was here has taken a considerable amount of time and I will like to give special thanks to Shibin M mentor for his help.
- Wall time for transforming and tuning jobs:

  Once the job is prepared is quite quick but the preparation of each job takes a high amount of time reducing productivity and lose of focus because I had to change of activity while waiting and then return to that task and considering that I was learning to use it and I needed to readjust several times the job preparation this factor extended my dedication time to the project. Would be very useful an option to test if the job is correctly configured to receive quick feedback for calling errors.
- Memory management

  With this project I have realized the importance of having control of the memory used by the notebook to keep it correctly running specially when exist limitations of the computing instance to use.

RESULTS

Results for Customer Segmentation

With the feature composition of components 132 and 133 we can make a persona of the type of customer that buys products from Arvato.

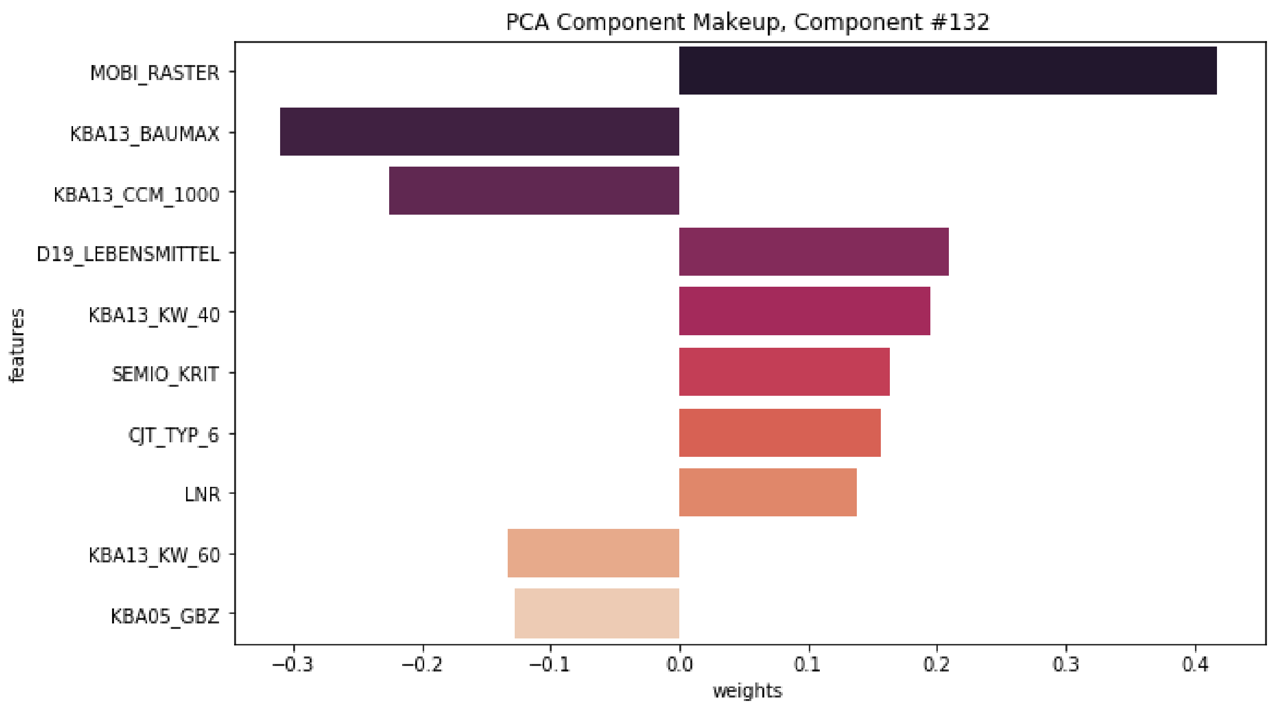


Figure 19: PCA Component 132

From the plot above we can see that KBA13_BAUMAX and KBA13_CCM_1000 are strongly negative correlated and MOBI_RASTER is positive correlated.

From the csv description of these features:

```
- KBA13_BAUMAX: Number of buildings in the area (1 low - 5 high)
- KBA13_CCM_1000: Car with less than 1000cc
- MOBI_RASTER: Industrial area (1 industry - 5 residential)
```

The profile we get from this component confirms the trend that the buyers have a relatively high income (KBA13_BAUMAX negative correlation as the previous component), have some car with less than 1000cc (KBA13_CCM_1000, possibly as a second car) and live in residential cells (MOBI_RASTER positive)

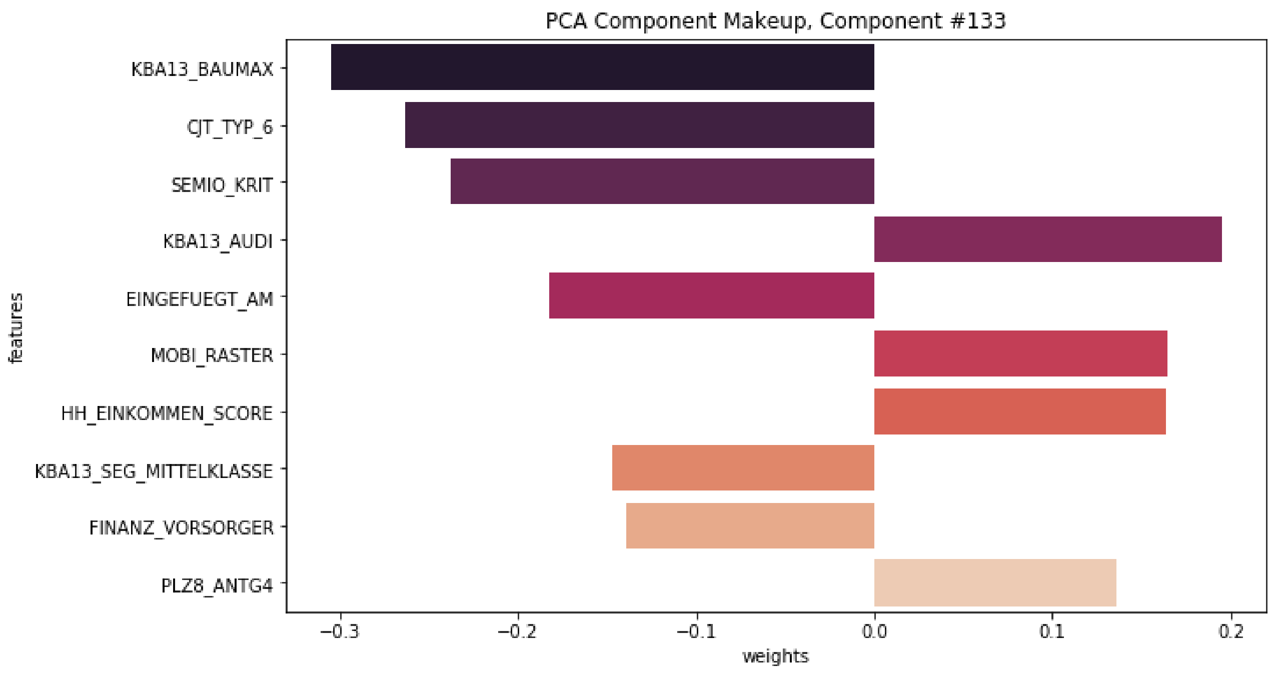


Figure 20: PCA Component 133

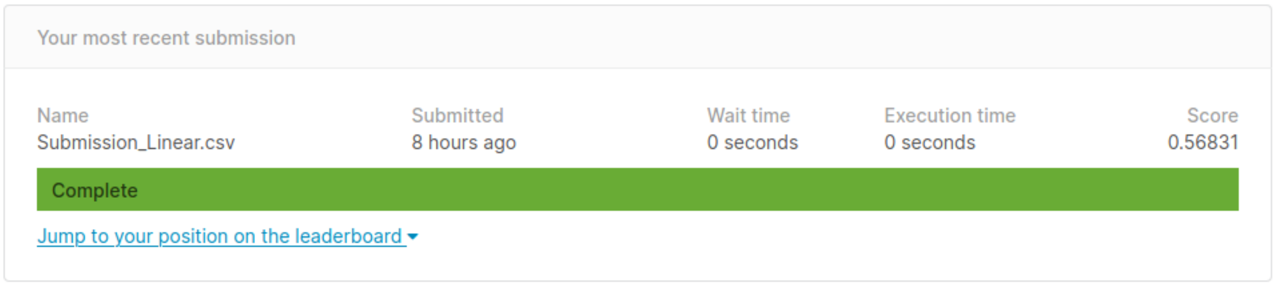
From the plot above we can see that KBA13_BAUMAX, CJT_TYP_6 (from CJT_GESAMTYP, 6 Advertising enthusiast with restricted crosselling) and SEMIO_KRIT strongly negative correlated and KBA13_AUDI is positive correlated.

From the csv description of these features:

```
- KBA13_BAUMAX: Number of buildings in the area (1 low - 5 high)
- CJT_TYP_6: Customer journey type (1 Advertising & consumption minimalist
- 6 Advertising enthusiast)
- SEMIO_KRIT: Critical mind ( 1 high - 7 low)
- KBA13_AUDI: Owner of an Audi
```

The profile we get from this info is that the buyers have a relatively high income (If they can spare to buy an Audi), live in low crowded neighborhoods (KBA13_BAUMAX negative correlation), are Advertising and consumption minimalist (CJT_TYP_6 negative correlation) and have studies or interests (SEMIO_KRIT negative correlated)

Results for Customer Prediction



Your most recent submission

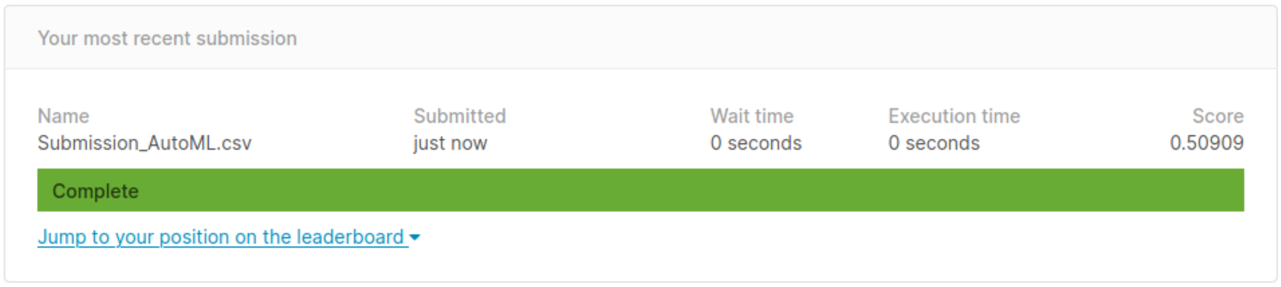
Name	Submitted	Wait time	Execution time	Score
Submission_Linear.csv	8 hours ago	0 seconds	0 seconds	0.56831

Complete

Jump to your position on the leaderboard

Figure 21: Kaggle result for Linear Learner

The result of the Logistic Regression is 0.568, that can be taken as a baseline to see if Gradient Descent has a significant better performance.



Your most recent submission

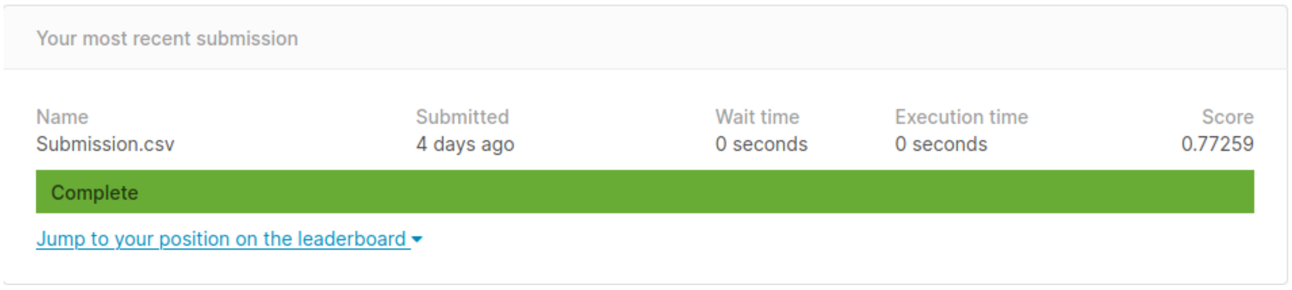
Name	Submitted	Wait time	Execution time	Score
Submission_AutoML.csv	just now	0 seconds	0 seconds	0.50909

Complete

Jump to your position on the leaderboard

Figure 22: Kaggle result for AutoML generated model

The result for the AutoML experiment is 0.509, I think it can be improved running more tuning jobs because the recommended value were 250 and we ran 14 due to time and budget limitations.



Your most recent submission

Name	Submitted	Wait time	Execution time	Score
Submission.csv	4 days ago	0 seconds	0 seconds	0.77259

Complete

Jump to your position on the leaderboard

Figure 23: Kaggle result for XGBoost

The result on Kaggle is 0.77 AUC it is a huge leap comparing to Logistic Regression and the result shows that is a better approach. On the other hand I feel it could be improved with more feature engineering.

Justification

Comparing the three results it is clear that Gradient Descent is a good algorithm to use for the prediction problem. It improves in 18 points the Linear Regression result used as a benchmark baseline.

Conclusions

Reflection

The process for this project with data from Arvato-Bertelsmann has been:

- Data loading, cleaning, reduce the amount of memory used by the data structures of the notebook and PCA for dimensionality reduction.
- Fit a model with a K-Means algorithm to make groups based on general population data and then group the customer samples into this groups, compare them and show which are the main features of the population in the groups that are located most of the customers.
- Fit a Linear, a XGBoost model and an AutoML experiment with data from a mailing action to predict which persons would react to an another mailing.

Improvement

There are some improvements that can be done to this project:

- A more accurate look to all the data features values to extract more features such as rural vs city.
- Perform both imputation approaches for segmentation and prediction to evaluate which is more suitable in each case
- Perform the K-Means separating the customers types and see if each customer type belongs to a specific group or they are distributed among the groups, that would show that each customer type belongs to a general population group.
- Another method that I would like to apply to compare with customer segmentation is to use the first K-Means to label each sample, then train a model with that label as a target and finally use this model to predict the class (group) of the customer data set.
- It is not really an improvement point but I had the sensation that for the amount of data of the project It was too many effort to use the Sagemaker transformations and could be done more quick with standard Sci-kit learn with the computation power of the notebook but the point was to showcase and train this technology
- I would like to launch an experiment in Sagemaker Studio with more jobs (200 instead of 15) to see if the performance would improve in a significant way.

References:

https://medium.com/@raghavan99o/principal-component-analysis-pca-explained-and-implemented-eeab7cb73b72

https://en.wikipedia.org/wiki/Dimensionality_reduction

https://blog.bioturing.com/2018/06/14/principal-component-analysis-explained-simply/

https://en.wikipedia.org/wiki/K-means_clustering

https://blog.easysol.net/machine-learning-algorithms-3/

https://en.wikipedia.org/wiki/Gradient_boosting

http://uc-r.github.io/gbm_regression