Capstone Project Report

UDACITY MACHINE LEARNING ENGINEER NANODEGREE

Customer Segmentation for Arvato Financial Services

Karina Quibell | July 2020

Contents

Figures

Definition

Project Overview

The project is a real-life data science task with four datasets provided by Arvato Financial Services. We are given general German population data as well customer population data for a mail order company in Germany. Each row in the datasets represent an individual and provides information about them, their household, their building and their neighborhood. The customer population data contains the same information as the general German population data and is therefore a subset of the census data with additional customer-specific information. In particular, we have:

1. Udacity_AZDIAS_052018.csv: Demographics data for general population of Germany - 891 211 persons (rows) x 366 features (columns)

2. Udacity_CUSTOMERS_052018.csv: Demographics data for customers of the mail order company – 191 652 persons (rows) x 369 features (columns). This dataset contains the same columns as the general population as well as the following customer-specific columns: 'CUSTOMER_GROUP', 'ONLINE_PURCHASE', and 'PRODUCT_GROUP'.

Our first task was to use these two datasets and unsupervised learning techniques to describe which people in the general population are most like the customer base of the company. This insight can be used to develop a marketing strategy tailored to the companies' target audience.

We were also provided with training and testing datasets containing demographics for individuals that were targeted in a marketing campaign. We have:

3. Udacity_MAILOUT_052018_TRAIN.csv: Demographics data for individuals who were targeted for the marketing campaign that will be used for training purposes – 42 982 persons (rows) x 367 features (columns). This dataset also includes a 'RESPONSE' column indicating whether the individual responded to the campaign and become a customer of the company.

4. Udacity_MAILOUT_052018_TEST.csv: Demographics data for individuals who were targeted for the marketing campaign that will be used for testing purposes – 42 833 persons (rows) x 366 features (columns).

Our second task was to use these two datasets and supervised learning techniques to predict which individuals are likely to respond to the marketing campaign and become customers. This would ensure that marketing campaigns are only sent to individuals who are most likely to respond in order to improve the ROI of the campaign.

In addition, we are also provided with two description files:

1. DIAS Information Levels - Attributes 2017.xlsx – provides a top-level list of attributes and descriptions organized by information category.

2. DIAS Attributes - Values 2017.xlsx – detailed mapping of data values for each feature in alphabetical order.

Problem Statement

Given general population demographics as well as customer demographics data, use unsupervised learning techniques to identify segments of the population that are most like the company's customer base in order to a marketing strategy tailored to the companies' audience. By reducing the dimensionality of the data through Principal Component Analysis (PCA), we can create clusters and determine segments of the population that are most like the customer base as well as segments that are least like the customer base.

Further, given training and test demographics data, use supervised learning techniques to predict which individuals are likely to respond to the marketing campaign and become customer to ensure future marketing campaigns are only sent to individuals who are most likely to respond in order to improve the ROI of the campaigns. By using classifiers we can predict the probability of a customer responding to the campaign.

Evaluation Metrics

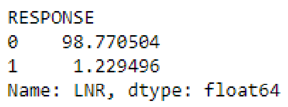
There are two parts to this project, each with it's own set of evaluation metrics that will be discussed below.

Part 1: Customer Segmentation using unsupervised learning

In this part of the project we will use PCA to reduce the dimensionality of the data. We will need to pick the number of principal components in such a way as to capture 80-90% of the variation within the data. Once we have reduced the dimensionality of the data we can cluster the data in component space. For K-Means Clustering, we will need to determine the number of clusters (k) to use. We want to choose a good k by ensuring that the data points are close to the cluster center, but with enough clusters to separate the data efficiently. We can look at how close each data point is to the cluster center and compare it to the number of clusters. When we represent this visually it will create an elbow plot with the sharpest bend indicating where the distances between centroids and data points stops decreasing at a sharp rate, indicating a good choice for the number of clusters (k).

Part 2: Customer Response Prediction using supervised learning

When we look at the 'RESPONSE' variable, we see that only 1.2% of the individuals in the training data set have taken up the offer and we therefore have a severe class imbalance.



```
RESPONSE
0    98.770504
1     1.229496
Name: LNR, dtype: float64
```

Given the nature of the problem and that we have class imbalance, based on this article by Jason Brownlee, the final prediction can be evaluated by the AUC (area under the curve) for the ROC (receiver operating characteristic) curve, where the ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. Further we have,

$$TPR = TP / (TP + FP) \quad \text{and} \quad FPR = FP / (FP + TN)$$

where TP = True Positive, FP = False Positive and TN = True Negative.

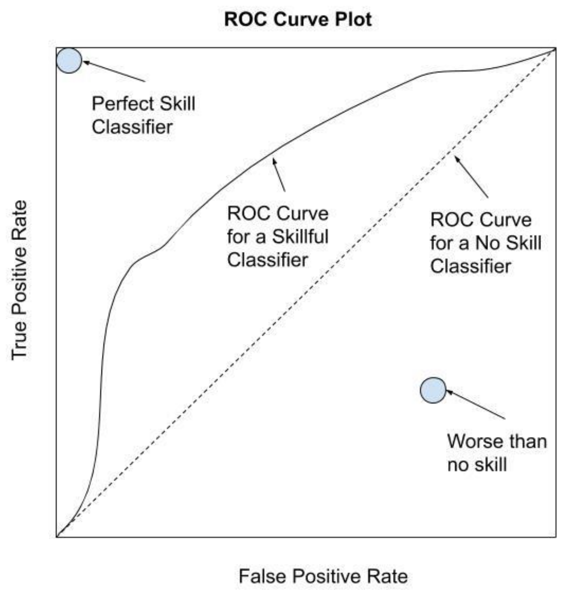


Figure 1: ROC Curve Plot (Jason Brownlee, 2020)

Analysis

Data Exploration & Preprocessing

Due to the size of the datasets, loading them takes some time. By passing an additional 'chunksize' parameter to the starter code provided and concatenating the these chunks into a single DataFrame, I managed to reduce the time it takes to read in the datasets quite substantially.

```
azdias_chunks = pd.read_csv('../../data/Term2/capstone/arvato_data/Udacity_AZDIAS_052018.csv', sep=';', chunksize=100000)
azdias = pd.concat(azdias_chunks)
```

I verified that the number of rows and columns in each DataFrame tie up with the dataset size information we were provided with to make sure all chunks loaded correctly.

The below schematic shows the processing steps that was taken to clean the data.

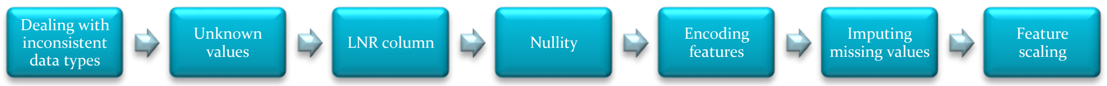
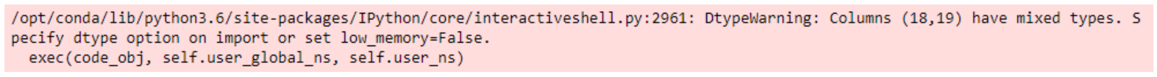


Figure 2: Data Cleaning Process

Each of these steps are discussed below.

Dealing with inconsistent types

When loading the data, we are confronted with an error:



```
/opt/conda/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2961: DtypeWarning: Columns (18,19) have mixed types. S
pecify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Figure 3: Error message when loading the data

'CAMEO_DEU_2015' has null values, string values as well as 'X' value. No warning message was printed for this column as all values are all string values.

'CAMEO_DEUG_2015' and 'CAMEO_INTL_2015' has null values, string values, float values as well as values of 'X' and 'XX' which explains the mix type error message.

Helper functions were created to:

- replace the odd values in each of the columns with null
- convert the mixed types to numeric values

Unknown values

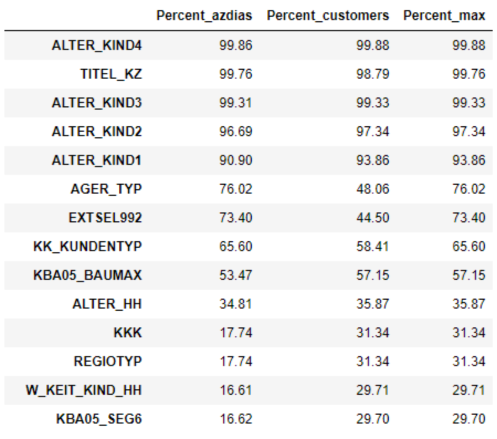
The attribute values file was used to check which values correspond to missing / unknown values. Unknown values were replaced with null (NaN) values.

LNR column

The LNR column seems to represent a record number and since it is unique for each row, it will not help with our analysis so it was dropped.

Nullity

Column nullity: The percentage of rows with null values for each row was analyzed.



	Percent_azdias	Percent_customers	Percent_max
ALTER_KIND4	99.86	99.88	99.88
TITEL_KZ	99.76	98.79	99.76
ALTER_KIND3	99.31	99.33	99.33
ALTER_KIND2	96.69	97.34	97.34
ALTER_KIND1	90.90	93.86	93.86
AGER_TYP	76.02	48.06	76.02
EXTSEL992	73.40	44.50	73.40
KK_KUNDENTYP	65.60	58.41	65.60
KBA05_BAUMAX	53.47	57.15	57.15
ALTER_HH	34.81	35.87	35.87
KKK	17.74	31.34	31.34
REGIOTYP	17.74	31.34	31.34
W_KEIT_KIND_HH	16.61	29.71	29.71
KBA05_SEG6	16.62	29.70	29.70

Figure 4: Column nullity

Columns that had null values for more than 30% of its' rows in either the population or customer data were dropped. This resulted in 12 columns dropped from both datasets.

Row nullity: For each row, the percentage of columns for which it had null values was analyzed.

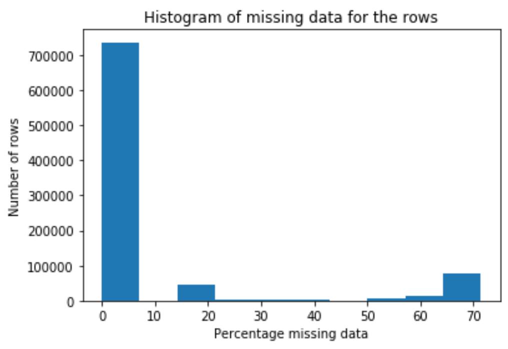


Figure 5: Row nullity

Rows that had null values for more than 10% of its' columns were dropped.

Encoding features

In order to perform PCA, we need to make sure that the values can be scaled. In order to scale values we need to:

- Make sure all values are numeric
- Impute missing values

We had the following object columns:

'CAMEO_DEU_2015', 'D19_LETZTER_KAUF_BRANCHE', 'EINGEFUEGT_AM', 'OST_WEST_KZ'

We didn't have information about the 'D19_LETZTER_KAUF_BRANCHE' and 'EINGEFUEGT_AM' columns, so instead of guessing how to impute them they were dropped.

'OST_WEST_KZ' had values of 'O' and 'W' that were replaced with 0 denoting East and 1 denoting West respectively.

Each alphanumeric value in 'CAMEO_DEU_2015' was mapped to an integer value.

Imputing missing values

The data that were not removed by the column and row nullity thresholds were imputed using the most frequently observed observation for each feature.

Feature scaling

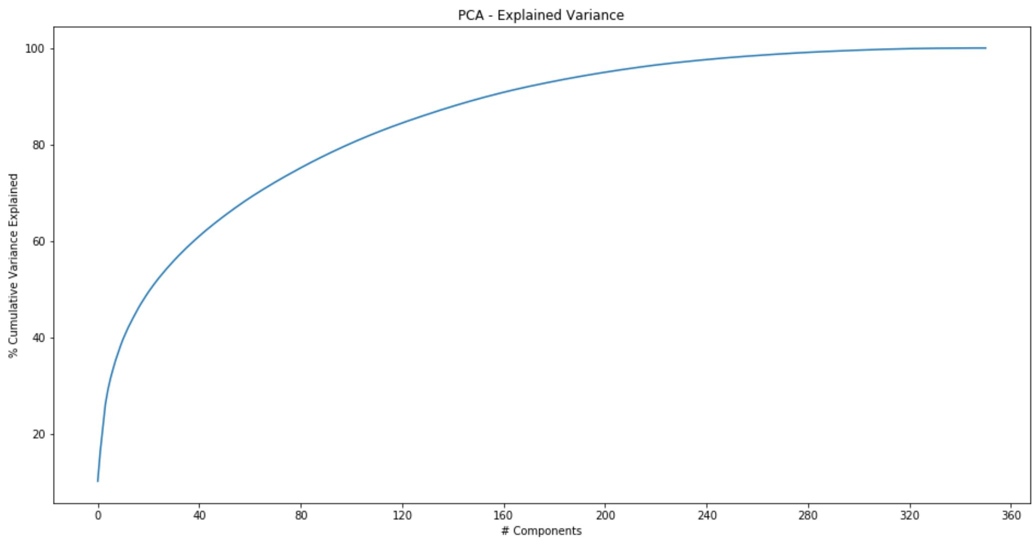
The last data processing step is to scale the features in preparation of PCA. Based on this article I have used a MinMaxScaler to transform the values so they fall in the range (0,1).

Algorithms, Techniques & Methodology

Part 1: Customer Segmentation using unsupervised learning

Dimensionality reduction

After the data preprocessing steps, we are left with 351 columns. To try extract meaningful results from all of these features would be a very time-consuming and tedious process and would not give us much insight. As such we will reduce the dimensionality of the data using a technique called Principal Component Analysis (PCA). PCA creates weighted independent combinations of features (called components) to reduce the number of features in the data in such a way that each component explains more of the variability than the subsequent component.

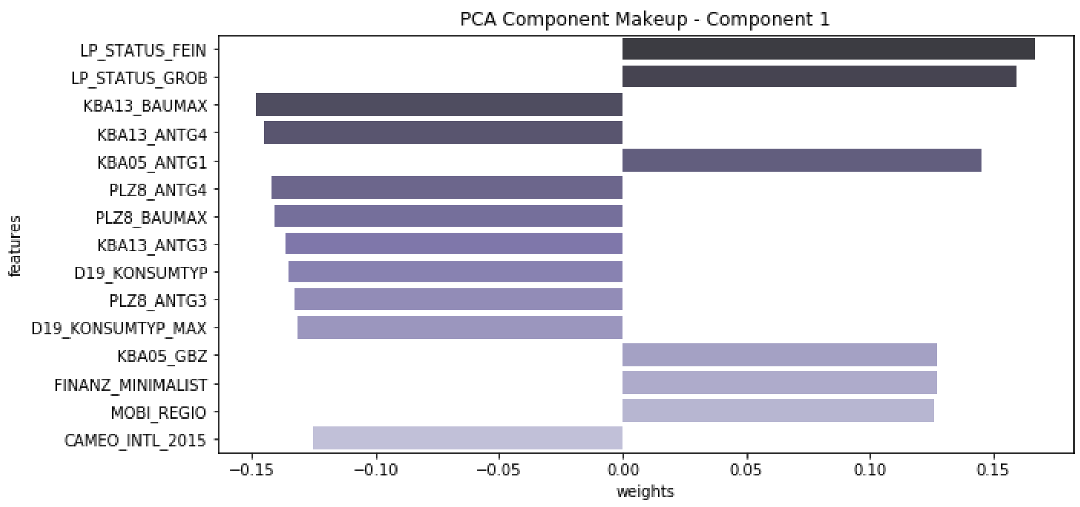


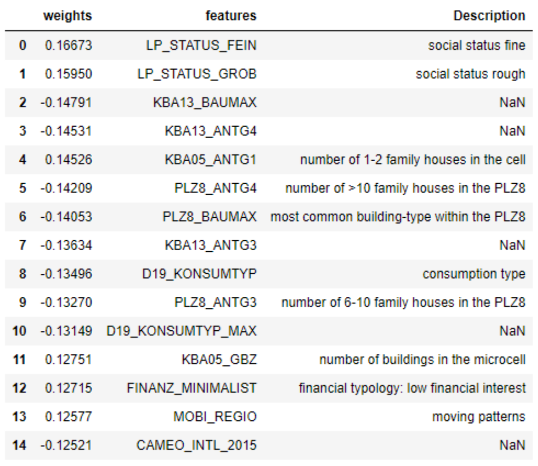
```
75.15% variance -> 81 components
80.05% variance -> 100 components
85.01% variance -> 124 components
90.02% variance -> 155 components
```

Figure 6: PCA - variance explained by number of components

The above output showed that **100** components are needed to explain at least **80%** of the variance. Since 100 components is already quite a lot and the idea is to reduce the number of features to reduce the complexity and computational time and resources, I've settled with 80% variance explained.

Each principal component gives a weighted linear combination of each of the original features. The larger the weight of a feature, the more important that feature is in that component.





	weights	features	Description
0	0.16673	LP_STATUS_FEIN	social status fine
1	0.15950	LP_STATUS_GROB	social status rough
2	-0.14791	KBA13_BAUMAX	NaN
3	-0.14531	KBA13_ANTG4	NaN
4	0.14526	KBA05_ANTG1	number of 1-2 family houses in the cell
5	-0.14209	PLZ8_ANTG4	number of >10 family houses in the PLZ8
6	-0.14053	PLZ8_BAUMAX	most common building-type within the PLZ8
7	-0.13634	KBA13_ANTG3	NaN
8	-0.13496	D19_KONSUMTYP	consumption type
9	-0.13270	PLZ8_ANTG3	number of 6-10 family houses in the PLZ8
10	-0.13149	D19_KONSUMTYP_MAX	NaN
11	0.12751	KBA05_GBZ	number of buildings in the microcell
12	0.12715	FINANZ_MINIMALIST	financial typology: low financial interest
13	0.12577	MOBI_REGIO	moving patterns
14	-0.12521	CAMEO_INTL_2015	NaN

Figure 7: PCA Component 1 Makeup

Component 1 corresponds to people who have a greater number of 1-2 family households in their neighborhood, have low financial interest and high movement patterns.

A function was created to explore the weights, features and description of any of the other 100 components.

Clustering

Now that we have reduced the dimensionality of the data, the data was clustered in component space. For K-Means Clustering, we needed to determine the number of clusters (k) to use. We wanted to choose a good k by ensuring that the data points are close to the cluster center, but with enough clusters to separate the data efficiently. We did this by looking at how close each data point is to the cluster center and comparing it to the number of clusters. When represented visually using an elbow plot we see the sharpest bend indicating where the distances between centroids and data points stops decreasing at a sharp rate, indicating a good choice for the number of clusters (k).

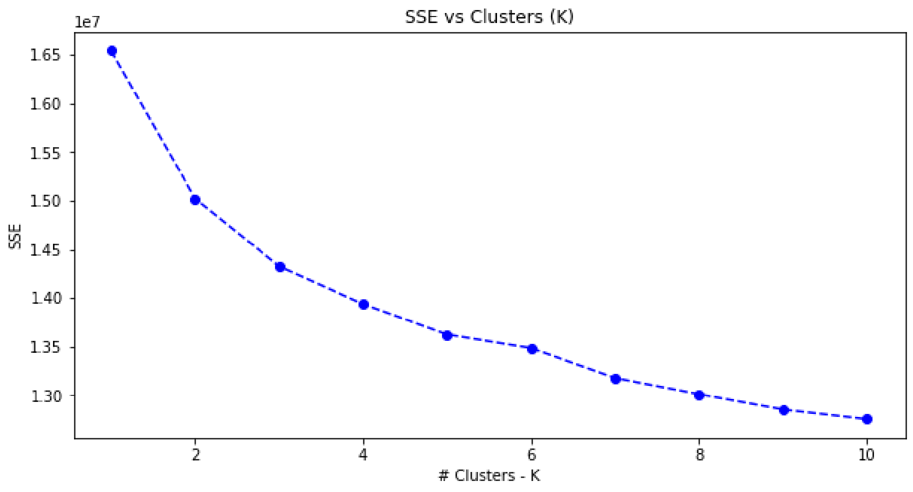


Figure 8: SSE vs Clusters Elbow Plot

By playing around and plotting the above graph for different end values of k, 6 clusters seemed to be a good choice for k.

Compare Customer Clusters to Demographics Clusters

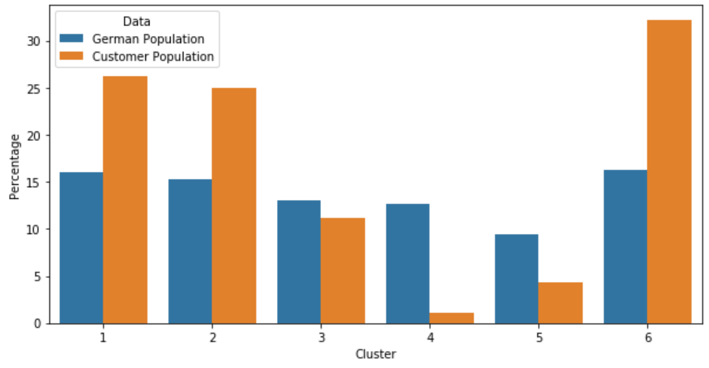


Figure 9: General population to customer population cluster comparison

Clusters 1, 2 and 6 are overrepresented in customers compared to the general German population, with cluster 6 being the most overrepresented. Clusters 3, 4 and 5 are underrepresented, with cluster 4 being the most underrepresented. We can therefore conclude that people belonging to cluster 4 are least likely to become customers, while customers belonging to cluster 6 are most likely.

It would be quite difficult for us to visualize 100 component centroids, but we can create a heatmap to show which characteristics define each cluster. Only the first 10 components are shown.

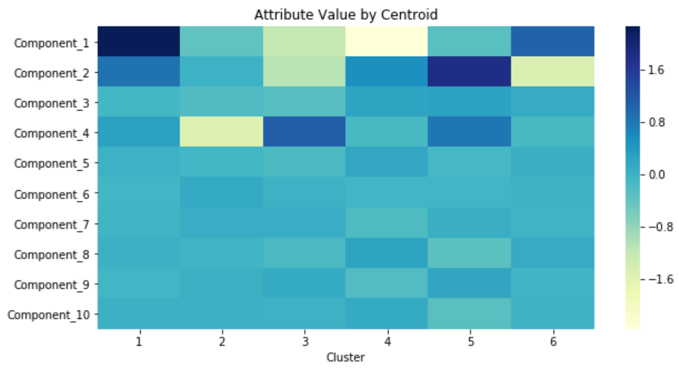
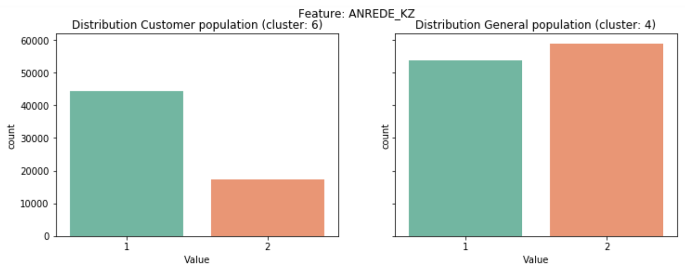
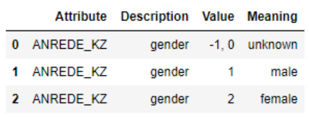


Figure 10: Cluster Characteristics Heatmap

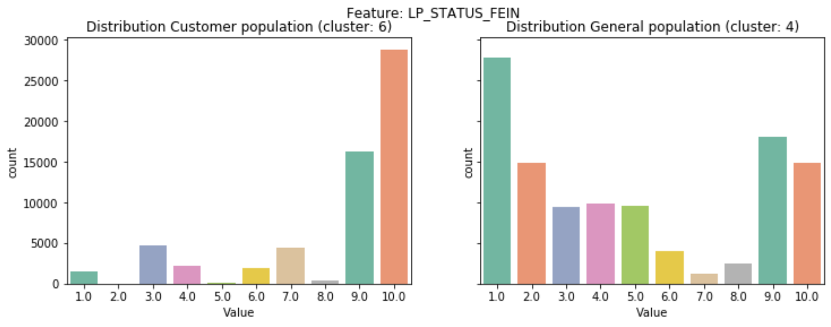
Cluster 6 (which are likely to be customers) is most influenced by component 2 and cluster 4 (which are least likely to be customers) is most influenced by component 1.

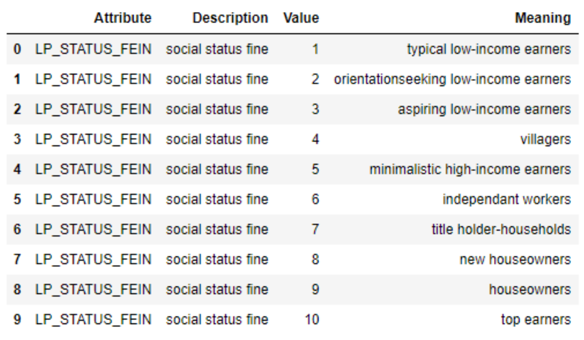
For each feature, we can compare the distributions between cluster 6 (customers) and cluster 4 (general population) to extract characteristics of a target audience.





	Attribute	Description	Value	Meaning
0	ANREDE_KZ	gender	-1, 0	unknown
1	ANREDE_KZ	gender	1	male
2	ANREDE_KZ	gender	2	female





	Attribute	Description	Value	Meaning
0	LP_STATUS_FEIN	social status fine	1	typical low-income earners
1	LP_STATUS_FEIN	social status fine	2	orientationseeking low-income earners
2	LP_STATUS_FEIN	social status fine	3	aspiring low-income earners
3	LP_STATUS_FEIN	social status fine	4	villagers
4	LP_STATUS_FEIN	social status fine	5	minimalistic high-income earners
5	LP_STATUS_FEIN	social status fine	6	independant workers
6	LP_STATUS_FEIN	social status fine	7	title holder-households
7	LP_STATUS_FEIN	social status fine	8	new houseowners
8	LP_STATUS_FEIN	social status fine	9	houseowners
9	LP_STATUS_FEIN	social status fine	10	top earners

Figure 11: Cluster characteristics

By comparing different features for the clusters, customers of the company are more likely to be higher-income single males with a high affinity to saving money. We can go deeper into other variables to refine target customers.

Part 2: Customer Response Prediction using supervised learning

This part of the project aimed at predicting which individuals are more likely to respond to a marketing campaign given a labelled set of data. Since we have labelled data, we could implement supervised learning.

The training data was cleaned using the same steps as detailed in Figure 2.

Benchmark

Based on our evaluation metric, an unskilled classifier or random guess will have a ROC AUC of 0.5. A perfect classifier will have a ROC AUC of 1 with near-perfect classifiers close (but not equal) to 1. My initial benchmark was for the model should to be at the very least be 20% better than a random guess and therefore we should aim to beat a ROC AUC score of 0.6.

I used a logistic regression as the benchmark model. Unless a model significantly outperformed logistic regression, I would choose simplicity over marginal accuracy as logistic regression is much easier to explain, especially in a business context. I've used GridSearch to find the best logistic regression model and obtained a ROC AUC score of **0.692**.

Although this was better than my minimum score of 0.6, I wanted to see if there were any other models that would perform better. I tried different classifiers as documented in the scikit-learn classifier comparison (ScikitLearn, 2019).

Base Models

Based on the classifier comparison, I used GridSearch to find the best base model for each of the following:

- Decision Tree Classifier
- Random Forest Classifier
- Gradient Boosting Classifier
- Ada Boosting Classifier
- Extreme Gradient Boosting (XGB) Classifier

The ROC AUC scores as well as the runtime for each of these models are tabled below.

Model	Runtime (seconds)	ROC AUC score
Logistic Regression	32.9	0.692
Decision Tree Classifier	8.1	0.761
Random Forest Classifier	3.9	0.699
Gradient Boosting Classifier	239.3	0.768
Ada Boosting Classifier	84.7	0.758
Extreme Gradient Boosting (XGB) Classifier	251.4	0.751

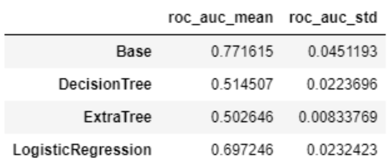
Table 1: Base Model Comparison

Surprisingly, XGBoost didn't perform as well as Ada Boost and took almost three times as long to execute. Gradient Boosting was marginally better than AdaBoost but also takes about three times longer to execute. Ada Boost therefore gives good performance for the time it took to execute. What also makes it a good option is that it is suitable to severely imbalanced data. It starts by fitting a weak classifier with weights that are initially equal and iterates by giving higher weightings to misclassified observations in order to obtain a strong classifier. Therefore, I chose Ada Boost and tuned it further.

Parameter Tuning

Tuning the Base Estimator for Ada Boosting Classifier

I used GridSearch to test which base estimator for Ada Boost would yield the best result and used cross validation to score the models.



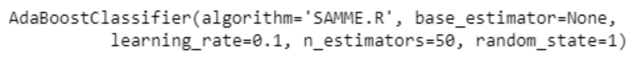
	roc_auc_mean	roc_auc_std
Base	0.771615	0.0451193
DecisionTree	0.514507	0.0223696
ExtraTree	0.502646	0.00833769
LogisticRegression	0.697246	0.0232423

Figure 12: AdaBoost base estimator tuning results

Based on the above results, I used the default Ada Boosting Classifier and tuned the other parameters.

Tuning number of estimators and learning rate for Ada Boosting Classifier

I then proceeded to use GridSearch with the number of folds for StratifiedKFold set to 10 folds to tune the parameters of the Ada Boosting Classifier to find the best estimator using the default base estimator.



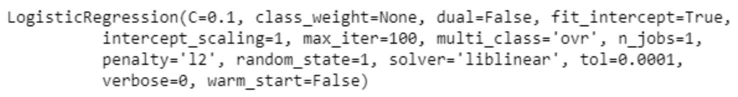
```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=0.1, n_estimators=50, random_state=1)
```

Figure 13: Best AdaBoost Estimator

The best estimator uses the default Decision Tree base estimator, to train 50 weak learners (n_estimators) and contributes a weight of 0.1 to each of the weak learners (learning rate).

Tuning the strength of the regularization for benchmark Logistic Regression benchmark model

I also tuned our benchmark Logistic Regression using a similar approach to that used for AdaBoost tuning.

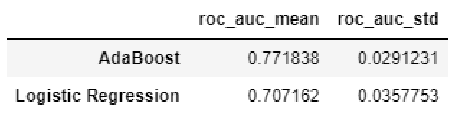


```
LogisticRegression(C=0.1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=1, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

Figure 14: Best Logistic Regression Estimator

Scoring tuned AdaBoost and Logistic Regression models

Using the same scoring method when testing the results of the untuned models, I checked the results for the tuned models.



	roc_auc_mean	roc_auc_std
AdaBoost	0.771838	0.0291231
Logistic Regression	0.707162	0.0357753

Figure 15: Tuned AdaBoost model compared to tuned Logistic Regression benchmark model

Results

Model Validation

Now that we have tuned both the benchmark model as well as the chosen model, let's see how well AdaBoost and our benchmark Logistic Regression models perform on unseen data. To do this I split the data into training and testing sets using a ratio of 80:20.

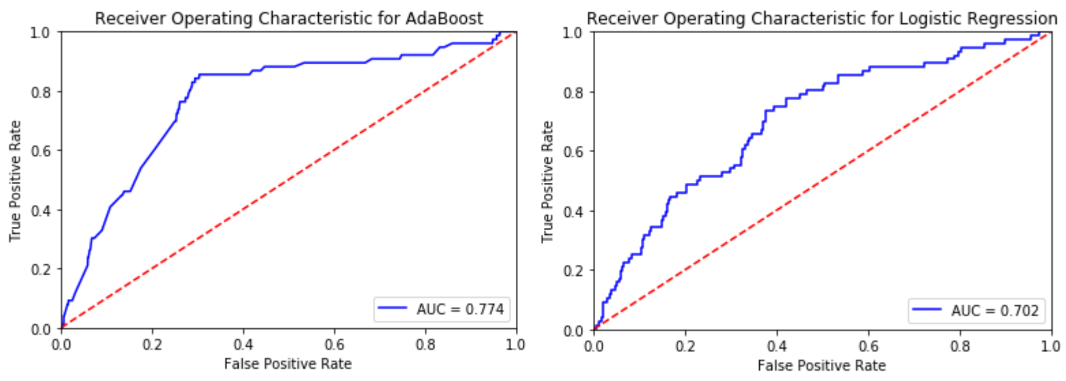


Figure 16: ROC Curve for tuned AdaBoost model compared to tuned Logistic Regression benchmark model

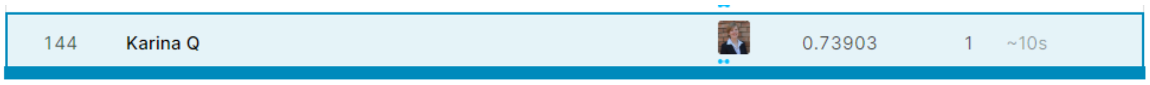
We get an average ROC AUC of 0.707 for the tuned Logistic Regression model and 0.772 for the tuned AdaBoost model. The best ROC AUC we could get from the AdaBoost model was 0.774 and got 0.702 when we have tested this on unseen (validation) data.

Based on these results, it seems that our tuned AdaBoost model does perform significantly better compared to our Logistic Regression benchmark model.

Therefore, the tuned AdaBoost model will be used on the test set as seems to perform well on unseen data.

Submission to Kaggle

Submitting the predictions of this classifier to Kaggle, resulting in a score of 0.739 which I am happy with, but there is some room for improvement, given the top score is 0.81.



144	Karina Q		0.73903	1	~10s

Figure 17: Kaggle Submission Result

Further Improvements

Further improvements include:

- Re-evaluating choices for nullity and feature engineering including encoding and imputation decisions
- Choosing algorithms with longer computation time to improve performance and tweak parameters at a larger scale.

References

Brownlee, J 2020, *Tour of Evaluation Metrics for Imbalanced Classification,* Machine Learning Mastery, viewed 11 June 2020, < https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/>.

Hale, J 2019, *Scale, Standardize, or Normalize with Sci-kit Learn,* Towards Data Science, viewed 4 July 2020, < https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02>.

SciKit Learn Documentation, 2019, *Classifier Comparison*, viewed 10 July 2020, < https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html>.

Waseem, M 2019, *How To Implement Classification in Machine Learning,* edureka!, viewed 11 June 2020, < https://www.edureka.co/blog/classification-in-machine-learning/>.