Capstone Project Report

Create a Customer Segmentation Report for Arvato Financial

1. Project Overview

Correctly predicting the respond to campaigns is important for companies so that they target the most appropriate groups of people, hence saving on cost. It also helps them acquire new customers more efficiently as instead of reaching out to everyone, they would just reach out to people identified as most likely to respond to the marketing campaign. If a company were to market their product without a strategy, they might not be able to reach potential customers as easily.

Arvato is a services company that develops and implements innovative Supply Chain Management (SCM), financial and IT solutions for business customers globally ("Arvato - Bertelsmann SE & Co. KGaA", 2020). In this project, a client of Arvato Financial Services, i.e. a German mail-order sales company is looking into targeted marketing on the population to acquire new customers. Attributes of existing clients of the company will be analysed and matched to the attributes of the general population in Germany to identify new clients for the company.

This project is one of the suggested capstone project options for the Udacity Machine Learning Nanodegree.

The outline of this project includes

- Customer Segmentation. Unsupervised learning techniques will be used to investigate the relationship between demographics of the general population in Germany and the company's existing customers
- Supervised Learning Model to predict if individuals will respond to a marketing campaign
- Submission of results on Kaggle

1.1 Dataset Overview

There are 4 datasets that are provided by Arvato Financial Services for this project:

Udacity_AZDIAS_052018.csv: Demographics data for the general population of Germany; 891 211 persons (rows) x 366 features (columns).
Udacity_CUSTOMERS_052018.csv: Demographics data for customers of a mail-order company; 191 652 persons (rows) x 369 features (columns).
Udacity_MAILOUT_052018_TRAIN.csv: Demographics data for individuals who were targets of a marketing campaign; 42 982 persons (rows) x 367 (columns).
Udacity_MAILOUT_052018_TEST.csv: Demographics data for individuals who were targets of a marketing campaign; 42 833 persons (rows) x 366 (columns).

The first 2 datasets will be used in an unsupervised learning task to divide the population and customers into clusters. The last 2 datasets will be used for training and testing in a supervised learning task to predict which customers are likely to respond to a marketing campaign.

Some helper excel files are also provided to explain the attributes found in the general population and customers dataset.

Attribute	Description	Value	Meaning
AGER_TYP	best-ager typology	-1	unknown
		0	no classification possible
		1	passive elderly
		2	cultural elderly
		3	experience-driven elderly

Information level	Attribute	Description	Additional notes
Person	HEALTH_TYP	health typology	in cooperation with Kantar TNS; the information basis is a consumer survey
	LP_LEBENSPHASE_FEIN	lifestage fine	modelled on different AZ DIAS data
	LP_LEBENSPHASE_GROB	lifestage rough	modelled on different AZ DIAS data

Fig 1. A snapshot of the excel helper files

1.2 Problem Statement

The problem statement of this project is “How can a mail-order company acquire new clients in an efficient manner?”

We can break this down further into, what are the attributes of general population that customers correspond to? And how likely are individuals to respond to marketing campaigns?

To solve this problem, I will break down the project into 3 parts. First, doing some data exploration and cleaning. I will also do some feature engineering to extract more information from the dataset provided.

Next, I will use an unsupervised learning approach to divide the German population into clusters, in which after that I will investigate how the customer population fits in to the previously created clusters. It is in the hope that some clusters will have significantly more customers than others. These over-represented clusters can then be assumed to be the core characteristics of customers for the company. This information can be further used for targeted marketing.

Lastly, using a supervised learning approach, I will predict the likelihood of individuals responding to a marketing campaign by the mail-order company. The response tells us how likely someone from the general population will convert to be a customer of the company.

1.3 Metrics

Since the classes are extremely imbalanced, in which most people do not respond to the marketing campaign, using accuracy will only give a false sense of how good the model is. For instance, the model has to only predict that all individuals will not respond and it can still achieve a reasonably high accuracy. Here, the Area Under the Receiver Operating Characteristic curve (AU ROC, or also known as AUC) which works well with imbalanced classes is used to evaluate the performance of the model. The ROC curve plots the True Positive Rate against the False Positive Rate, and this can be thought of as the proportion of correct predictions for the positive class against the proportion of errors for the negative class. Ideally, the plot should be in the top left of the plot. The AU ROC gives a score of between 0 and 1 as a quantitative measure to assess how the model is performing. The closer the score is to 1, the better the model is.

2. Analysis

2.1 Data Exploration

There are 891211 people in general population dataset and 191652 existing customers with the mail-order company. Each row in the dataset represents an individual, and includes information about their neighbourhood, spending, household and so on. 366 attributes of these individuals are given in the population dataset while 369 are given in the customers dataset. The customer dataset has 3 extra columns in it, namely 'CUSTOMER_GROUP', 'ONLINE_PURCHASE', and 'PRODUCT_GROUP'. The figure below shows an example of the first few rows of the population dataset.

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PERSONEN
0	910215	-1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	910220	-1	9.0	0.0	NaN	NaN	NaN	NaN	21.0	11.0	0.0	0.0	2.0
2	910225	-1	9.0	17.0	NaN	NaN	NaN	NaN	17.0	10.0	0.0	0.0	1.0
3	910226	2	1.0	13.0	NaN	NaN	NaN	NaN	13.0	1.0	0.0	0.0	0.0
4	910241	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	3.0	0.0	0.0	4.0

Fig 2. An example of the general population dataset

There are 3 different types of data in the dataset, which is numerical, categorical and ordinal. Numerical variables are variables where the number has a numerical meaning, categorical variables can be grouped into types or categories, while ordinal variables are variables that show a clear ordering of variables. For example, 'ANZ_HAUSHALTE_AKTIV' is a numerical column that represents the number of households in the building. 'ANREDE_KZ is a categorical column that represents the gender of the individual. 'ALTERSKATEGORIE_GROB' is an ordinal column that represents the age group of people.

A separate train and test set are provided which consists of individuals who were targets of the marketing campaign. This will be used in the latter portion of the project.

Information regarding the columns in the dataset can be found in 2 accompanying datasets. However, not all variables in the dataset are explained in these 2 files.

2.2 Data Exploration

- Missingness

I first did some exploration on the missingness of data. The figure below shows the histogram of the number of values missing per column. Most columns have less than 200000 missing values, which is less than a quarter of the dataset. Some columns even have more than 80000 values missing, implying that it is missing for more than 90% of the population.

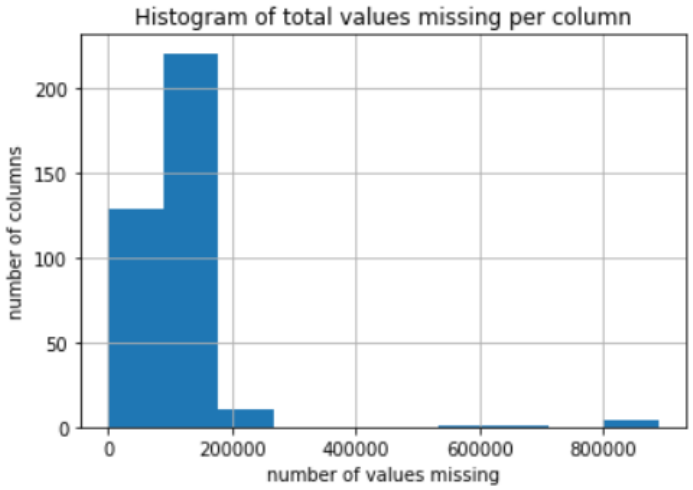

Fig 3. Histogram of total values missing per column

After checking the number of values missing per column, I will now have a look at the number of values missing per row. Most rows have less than about 25 values missing, while some rows that are on the more extreme end have more than 225 values missing.

Fig 4. Histogram of total values missing per row

- Correlation

Since there are 366 columns, the columns might be correlated to each other and therefore some might be redundant. Upon checking, 'KBA13_HERST_SONST' and 'KBA13_FAB_SONSTIGE' have a correlation coefficient of 1, implying a prefect positive correlation of the two. There are also several other columns that are quite strongly correlated with each other and they are shown below.

Column 1	Column 2	correlation coefficient
KBA13_HERST_SONST	KBA13_FAB_SONSTIGE	1.000000
LP_LEBENSPHASE_GROB	LP_LEBENSPHASE_FEIN	0.989961
LP_FAMILIE_GROB	LP_FAMILIE_FEIN	0.984100
LP_STATUS_GROB	LP_STATUS_FEIN	0.982411
PLZ8_GBZ	KBA13_GBZ	0.979854
		...
KBA13_SITZE_5	KBA13_SITZE_4	-0.867961
D19_VERSAND_ONLINE_QUOTE_12	D19_VERSAND_ONLINE_DATUM	-0.868449
D19_GESAMT_DATUM	D19_GESAMT_ANZ_24	-0.868989
D19_GESAMT_ONLINE_QUOTE_12	D19_GESAMT_ONLINE_DATUM	-0.870384
ORTSGR_KLS9	GEMEINDETYP	-0.934515

Fig 5. Table showing correlation coefficient of columns

2.3 Algorithms and Techniques

- KMeans Clustering

In the unsupervised learning part of the project, I will be using the KMeans clustering algorithm to divide the population (and customer) into clusters. Clustering is the process of dividing individuals into several groups based on patterns found in the data (Sharma, 2019). It is an unsupervised learning task as there is no target for us to predict. Data points found in a cluster should be like one another, while data points from different clusters should be different. Some common application of clustering includes for recommendation engines and for customer segmentation. To determine the number of clusters that is optimal, we measure the inertia of the data, which gives the sum of squared distance of points to their closest centre.

- Principal Component Analysis (PCA)

Since there are many columns in the dataset, we can transform data into a lower dimensional space with features that independent (orthogonal) to each other. Principal Component Analysis is a dimension reduction technique to reduce the dimensions of the data. Each principal component obtained is a unit vector explaining the maximal amount of variance.

- Decision Tree

Decision Trees are easily implemented and interpreted. Decision trees make decision by splitting nodes into sub-nodes. The splitting process is repeated until only nodes that are homogenous are left. When deciding on which features to split on, the tree tries to split the data in such a way that the resulting groups are as different from each other as possible. There are a few methods in which decision trees use to split their nodes, such as by calculating the information gain and the Gini impurity.

- Logistic Regression

Logistic Regression uses the sigmoid (or logistic) function that is used to map any value to a range between 0 and 1. Input values are combined linearly, producing a numeric value. The s-shaped sigmoid function then takes the real-valued number and maps it to between 0 and 1 It is fast and easy to implement, but doesn't perform well with too many features.

- Random Forest

The Random Forest Classifier is an ensemble of decision trees which are assembled independently. Multiple decision trees give a class prediction and the class with most votes becomes the model's final prediction. Each tree in the random forest chooses from a subset among all features when splitting their nodes, instead of considering all possible features like in a normal decision tree. This results in more variation between trees and helps prevent overfitting. Moreover, each individual tree randomly samples N data from the dataset (of size N) with replacement in a process called bagging. Hence, the random forest ends up with trees trained on slightly different dataset and features, making it less prone to overfitting and more robust.

- XGBoost

The XGBoost (Extreme Gradient Boosting) Classifier works by gradient boosting decision trees. The term gradient boosting is used because the gradient descent algorithm is used to minimise loss when new models are added. Subsequently added trees learn from its predecessors with the aim of reducing the errors of the previous tree, resulting in an improved performance. The process is repeated until there is no room for further improvement. Each tree is a weak learner which contributes information for prediction, resulting in a final strong learner through a combination of the weak learners.

- Light GBM

Light GBM works by splitting the tree leaf-wise, rather than depth-wise which is done by other boosting algorithms. This leaf-wise splitting algorithm can reduce more loss than the depth-wise algorithm, hence improving model performance. Light GBM can handle large datasets and is also very quick; hence the prefix 'light'.

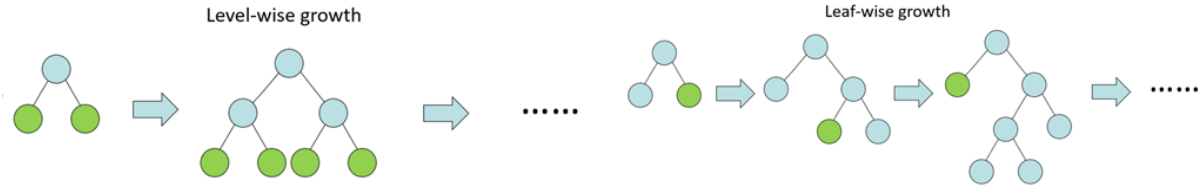

Fig 5. Diagram showing level-wise and leaf-wise growth (*LightGBM and XGBoost Explained,* 2018)

2.4 Benchmark Model

The benchmark model is a decision tree model. A decision tree is used because it is simple to implement. The decision-making process in a tree can also be easily interpreted by drawing the tree itself.

The benchmark model had a AUC score of 0.757 during training and validation and a slightly lower score of 0.732 on the Kaggle leaderboard. Subsequent models that I trained will attempt to perform better than this score.

3. Methodology

3.1 Data Pre-processing

Since the dataset is quite large, it sometimes resulted in memory issues which causes the notebook to crash. Hence, I created a function to downcast the datatype of columns where possible. For example, columns of datatype int64 being down casted to uint8. This reduced the size of dataset from more than 2GB to about 1GB.

Since there was a warning right after I loaded the Azdias dataframe, where columns 18 and 19 had mixed datatype, I had a closer look. These columns had numerical values (e.g. 4, 5), as well as string representative of numerical values (e.g. '4', '5') and 'X', 'XX' for missing values. The string values were converted to integers and 'X' and 'XX' were converted to np.nan. Some other columns that had similar issues were dealt with in a similar fashion.

```
/usr/local/lib/python3.6/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWarning: Columns (18,19) have mixed types.
```

Fig 6. Mixed datatype warning when loading data

I then dropped rows and columns that contain too many missing values. Columns with more than 50% of the values missing and rows with more than 50 values missing were removed.

I also created a function to check if two dataframes contains the same columns so that extra columns can be removed. This function was used several times in the project to ensure that dataframes that I was working with had the same columns.

3.2 Feature Engineering and Encoding

Multi categorical variables ('AGER_TYP', 'CAMEO_DEU_2015' and so on) were then one-hot encoded. I however dropped the 'D19_LETZER_KAUD_BRANCHE' as there were quite a few categories.

For binary categorical variables, such as 'OST_WEST_KZ', I encoded the values with 0 and 1.

Next, I did some feature engineering to extract more information from the data. The 'PRAEGENDE_JUGENDJAHRE' column contains 3 types of information: the decade in which the movement happened (40ies, 50ies), whether it was avantgarde or mainstream and whether it was O or W. It is quite hard to separate the 'O's and 'W's as some contain either while others contains both, I will only create 2 new variables with capture the decade of movement and whether it was avantgarde or mainstream.

The column 'CAMEO_INTL_2015' combines information on the wealth stage and family type in its tens and ones place. I separated the two-digits code into 2 one-digit codes to create new ordinal variables.

'LP_LEBENSPHASE_FEIN' contains information on both income and age. There are many categories which look similar to each other in this column, which will massively increase the number of columns in the dataset if they are one-hot encoded, and having too many columns takes up a lot of RAM causing my kernel to keep crashing. Hence, I extracted the age from 'LP_LEBENSPHASE_FEIN' and the drop the column, the income information can subsequently be obtained from 'LP_LEBENSHPASE_GROB'.

The column 'EINGEFUEGT_AM' is a timestamp and having timestamps with a lot of different values doesn't give us much information. Hence, I extracted the year from it.

Several columns such as 'LP_FAMILIE_GROB' and 'LP_STATUS_GROB' have different categorical values that carry the same meaning. For instance, in 'LP_FAMILIE_GROB', values 3, 4 and 5 all mean 'single parent'. These values were replaced such that they become the same category.

Since there are still missing values in the dataset, I used a simple imputer to fill in the missing values with the most frequent value. The imputer was fit on the population dataset and used for imputing both the general population and customer dataset.

3.3 Scaling

Before applying dimensionality reduction on the data, I scaled the data so that principal component vectors are not affected by the difference in magnitude of features. I first tried scaling using MinMaxScaler and StandardScaler but the process was quite memory intensive, causing the kernel to keep crashing. Hence, I ended up using RobustScaler to scale the data, which gives similar results to the previous scaler. The Robust Scaler is a scaler that is robust to outliers in the data. It uses the median and interquartile range in transforming data.

3.4 Dimensionality Reduction

After scaling, I used sklearn's PCA to obtain the principal components in the data. I first tried with a large number of principal components. The percentage cumulative variance explained was plotted against the number of principal components. Based on the plot, I selected the first 100 principal components as they explained more than 85% of the variance in the data. I then refit a PCA instance on the general population and used that to transform both the population and customer dataset.

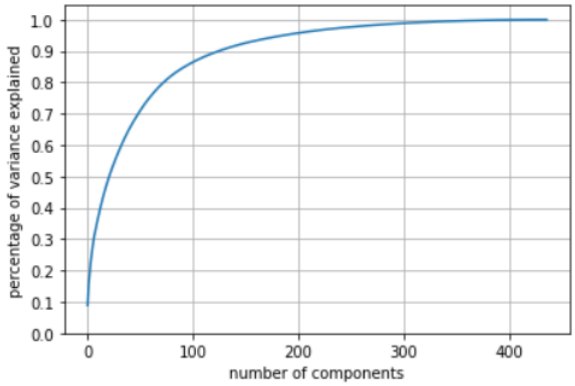

Fig 7. PCA plot showing cumulative percentage of variance explained against the number of components

3.5 Clustering

Next, KMeans clustering was applied on the PCA- transformed data. To obtain a suitable number for the clusters, the Elbow Method was used. I plotted the inertia (the sum of squared distance of points to their closest cluster centers) against the number of clusters. It can be seen that the inertia decreases as the number of clusters increase, although there is smaller decrease with each additional cluster. The optimal value of k is at the "elbow", which is the point where inertia starts decreasing linearly (Gupta, 2019). To help in determining the "elbow", I fitted the last few points on the plot with a linear line to get the number of clusters where the plot and line started diverging. I ended picking 8 clusters. I re-fit a KMeans instance with 8 clusters on the population data and transformed both datasets.

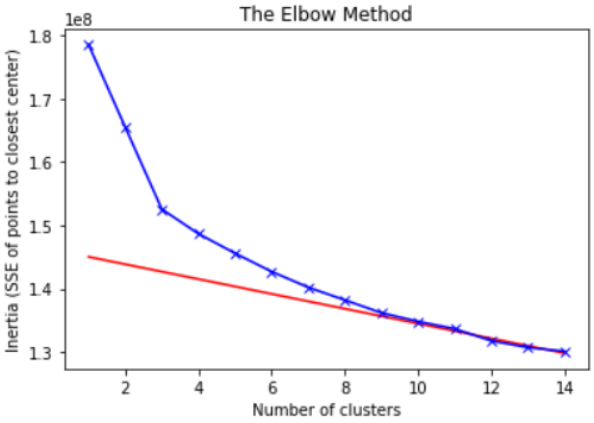

Fig 8. Plot of Sum of Squared error against number of clusters to get optimal number of clusters

4. Customers Segmentation

I now have clustered data based on demographics of the general population and customer data that fits into those cluster. If the company's customers are fairly universal, then the percentage of people from the population in each cluster should be somewhat similar to the percentage of people from the customers data. If there is a specific cluster that is extremely interested or extremely not interested in the company's product, there should be a large different in the proportion of people. For example, if there is a significantly larger percentage of people from the customer dataset than from the general population in a cluster, this suggest that the cluster might be a target audience of the company. I plotted the percentage of people from the general population and from the customers dataset for each cluster side by side to ease comparison.

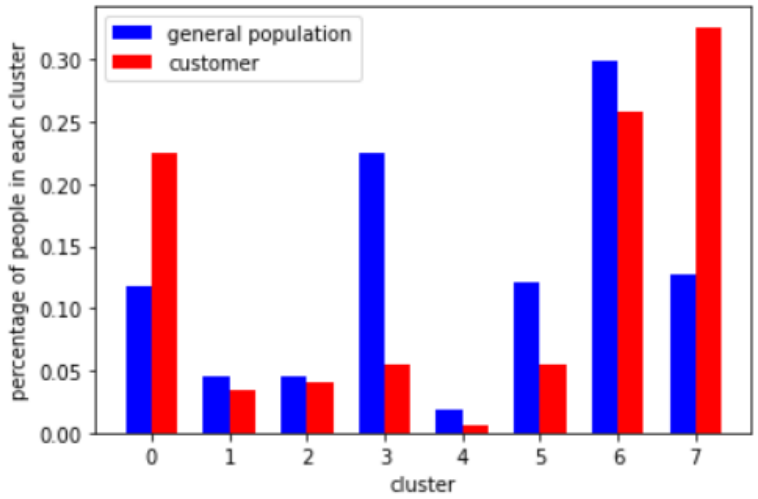

Fig 9. Plot of distribution of the general population and customers across different clusters

Clusters 7 and 0 are found to be likely target customers of the company while clusters 3 and 5 are outside of the targeted group. I then used the inverse_transform() method of the PCA on the clusters specified above to retrieve the important features that contribute to individuals being allocated into the specified clusters.

Likely customers

For clusters classified as likely targets for the mail-order company, the features 'D19_HAUS_DEKO', 'D19_KOSMETIK', 'D19_REISEN' and 'D19_LOTTO', which are the transactional activity based on the house decoration, cosmetic products, travel related products and LOTTO product groups contributes positively. 'D19_LOTTO' seems to have an extremely strong influence on classifying an individual to be in cluster 0. 'D19_GESAMT_OFFLINE_DATUM' and 'D19_VERSAND_OFFLINE_DATUM', i.e. doing the last transaction with the complete file offline and for the segment mail-order offline contributes negatively.

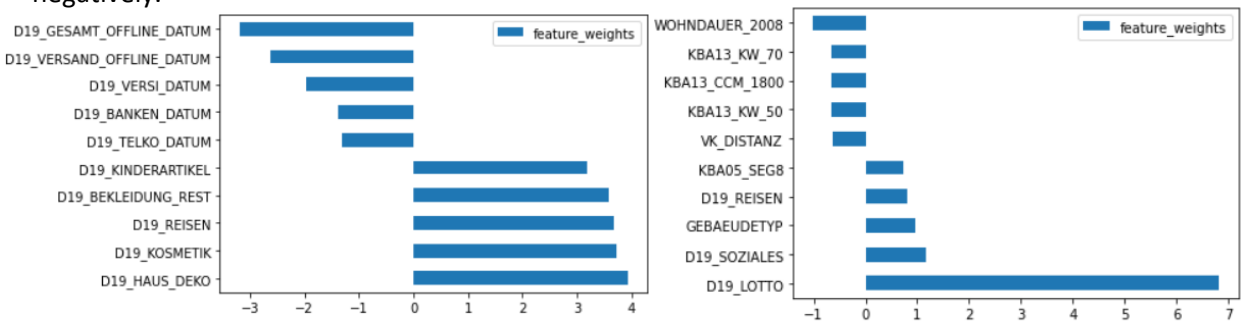

Fig 10. Important features for cluster 7 and 0 respectively

Unlikely customers

For clusters that are relatively unpopular with the company, 'ANZ_HAUSHALTE_AKTIV' and 'D19_BANKEN_DIREKT', which are the number of households in the building and the transactional activity for the product group direct bank contributes positively to individuals being in that cluster. Features that are negatively correlated with the group includes 'KBA13_KMH_211' (share of card with maximum speed greater than 210km/h within the PLZ8) and 'D19_BANKEN_DATUM' (the last transaction for the bank total segment).

5. Supervised Learning model

5.1 Model training and hyperparameter tuning

Now, I will move on to the supervised learning part of the project, which is to train several classification models to predict whether individuals will respond to a marketing campaign. The train and test set were first pre-processed the same way the general population and customers dataset was pre-processed, i.e. dropping rows/ columns with a lot of NAs, feature engineering and imputing. Several types of algorithms were used such as the decision tree as the baseline model, logistic regression, random forest, XGBoost and Light GBM.

The logistic regression classifier failed to converge after the default number of iterations had been reached. Hence, I tried scaling the data and increasing the number of iterations. It however performed much worse than the decision tree classifier, even after scaling was applied so I did not proceed in fine tuning or trying to improve this model.

I then tried using random forest, XGBoost and Light GBM. The untuned random forest preformed surprisingly poorly. It might be because the default hyperparameters of the random forest does not work well with the training data I had. Nevertheless, I tried tuning its hyperparameters using RandomizedSearchCV and it ended up performing slight better than the decision tree. The untuned and tuned XGBoost and Light GBM performed quite well.

5.2 Results

In evaluating the models during training and hyperparameter tuning, the cross-validation score was used, to prevent an overly or underly optimistic score of the model. The training set is split into k smaller sets and the model is trained using only the k-1 sets of data. The model is then validated using

the remaining set of data which was withheld. The performance of model is then average of the scores computed. The scoring parameter used is the Area under the ROC curve as explained above.

The results of various models that had their hyperparameters tuned are summarized in the table below.

Model	Validation (mean CV AUC)	Kaggle Leaderboard
Decision Tree (benchmark)	0.757	0.732
Logistic Regression (with scaled data)	0.653	0.623
Random Forest (untuned)	0.604	0.616
Random Forest (tuned hyperparameters)	0.753	0.751
XGBoost (untuned)	0.778	0.791
XGBoost (tuned hyperparameters)	0.783	0.805
Light GBM (untuned)	0.761	0.753
Light GBM (tuned hyperparameters)	0.774	0.778

Fig 11. Results of various supervised learning models

Of the untuned models, the XGBoost model performs the best. We can see here why the XGBoost is said to be a popular choice in machine learning competitions. The tuned XGBoost model worked quite well if the validation data and even better with the test set. Hence, this model is selected as the final model.

The tuned XGBoost model performed best of all models and gave an AUC score of 0.805, which in on the top 15% of the leaderboard. This is much better than the benchmark model that I had initially, with an AUC score of only 0.732. At the time of writing, aside from the top scorer having an AUC score of 0.847, all the other top scorers have AUC scores of around 0.80-0.81. I would say that the performance of my XGBoost model is quite good, perhaps with more time spent on hyperparameter tuning, the model might perform even better.

From the XGBoost model, I tried plotting the important features that contributed to decisions made by the model. 'D19_SOZIALES', 'KBA05_MAXVORB' (most common preowner structure in the microcell) and 'D19_SAMMELARTIKEL' (transactional activity based on the product group collectable items) are among the top features that are important in helping the model reach its decision. The most important feature 'D19_SOZIALES' however is one of the features in the dataset with no description given.

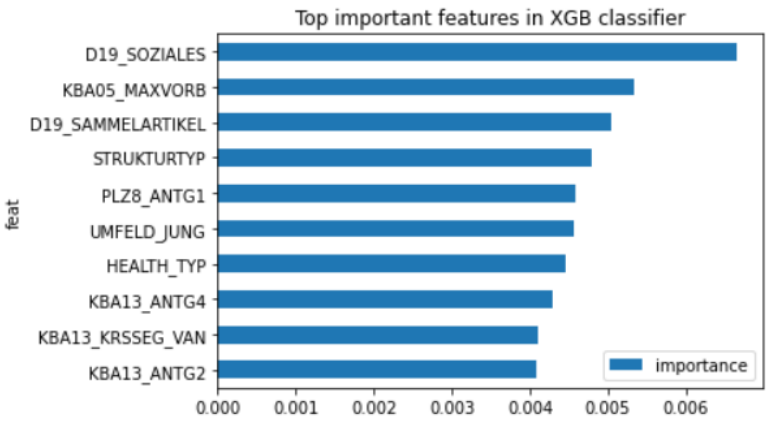

Fig 12. Features with greatest impact on the model

6. Discussions and Conclusion

In this project, I looked at unsupervised and supervised learning techniques in helping a mail order company acquire new clients. It is interesting to see how clustering the general population into groups and seeing where customers fit in those groups gives us an idea on the characteristics of individuals that form the company's customer base. It is also intriguing to see how demographic data can be used to predict individuals' responses to marketing campaign through supervised learning. From the supervised learning model, we can also see what are the top features that helps the model in making its decision.

We can see that the top contributing features in the clustering and prediction tasks are different. An interesting point might be to use the same dataset in the unsupervised and supervised learning task, and to investigate individuals who were clustered in the 'likely to be customers' group yet had a prediction of 'no' in responding in the marketing campaign or vice versa to find out the reasons they differ.

The project was an interesting experience as it provided me the opportunity to work with a real-life dataset that have a lot of columns and rows, as well as missing data. The large number of columns and rows also made it quite memory intensive to do some operations on, giving me the chance to explore different options of achieving similar results.

7. Improvements

Some further improvements could be made to improve the results. For instance,

- Changing the threshold for dropping rows and columns.
- Using an iterative imputer to fill in missing values instead of merely filling in with the mode.
- Since PCA was applied to the general population and customers dataset, performing PCA on the training set before training it to predict responses might improve the performances of models, such as the logistic regression model.
- More models can also be trained in the supervised learning task, such as by using support vector machines, naïve bayes or deep learning methods.

References

Gupta, A. (2019, June 06). Elbow Method for optimal value of k in KMeans. Retrieved September 30, 2020, from https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/

Sharma, P. (2019, August 19). The Most Comprehensive Guide to K-Means Clustering You'll Ever Need. Retrieved September 28, 2020, from https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

K. (2018, January 5). LightGBM and XGBoost Explained. Retrieved September 30, 2020, from https://mlexplained.com/2018/01/05/lightgbm-and-xgboost-explained/