Machine Learning Engineer Nanodegree

Capstone Project

Customer Segmentation for Arvato Financial Solutions

Zhaoyun Ma

01/18/2021

Contents

1 Project overview

This project is one of the proposed capstone projects in Udacity Machine Learning Engineer Nanodegree program. This project aims to help a mail-order company to optimize the customer targeted marketing campaign as well as predict the probability of customer conversion for better targeting. The first part of this project focuses on customer segmentation in order to identify the clusters of population that best describe the core customer base of the company compared to the general population. The main task of second part is to identify a supervised learning model that can predict the likelihood of customer conversions with a reasonable accuracy.

1.1 Project data

The data used is provided by Udacity partners at Bertelsmann Arvato Analytics, and represents a real-life data science task. Four demographic data files are provided with some general information, including:

- Udacity_AZDIAS_052018.csv (general population of German, 891 211 rows by 366 columns)
- Udacity_CUSTOMERS_052018.csv (customers of a mail-order company, 191 652 rows by 369 columns)
- Udacity_MAILOUT_052018_TRAIN.csv (targets of a marketing campaign, 42 982 rows by 367 columns)
- Udacity_MAILOUT_052018_TEST.csv (targets of a marketing campaign, 42 833 rows by 367 columns)

The first two data files (azdias and customers) are used for the customer segmentation and figure out the similarity and difference between the customer and general population. The mailout_train data file is used to build up a supervised machine learning model and then the mailout_test data file is then used to test the model accuracy in a Kaggle competition.

In addition, two excel spreadsheets are provided with detailed documentation of the features in the demographic data files, which are:

- DIAS Information Levels — Attributes 2017.xlsx
- DIAS Attributes — Values 2017.xlsx

1.2 Project background

Bertelsmann Arvato are providing consultation for a mail-order company to better understand their customer demographics compared to the general population in German and identify the customer segments for future marketing campaigns. **Customer segmentation** aims to divide **customers** into clusters based on common characteristics so that companies can market to each cluster effectively and appropriately. Customer segmentation is very helpful for companies to understand their customers and the targeted marketing can save companies significant cost compared to marketing towards the whole population.

1.3 Problem statement

- Identify the customer segments that can describe the core customer base of the company compared to the general population for future marketing campaign through principle component analysis (PCA) and unsupervised learning technique
- Identify a supervised machine learning model that can predict the customer conversion probability based on the output a customer segmentation (Will the customer respond to the tailed campaign?) through comparing a few supervised learning classifiers, Bayesian Optimization, and GridSearchCV to find the best model

1.4 Evaluation metrics

In this project, since the **mailout** data is highly imbalanced with more than 98% of the data response is negative and less than 2% of the data response is positive, accuracy is not appropriate for the model evaluation. Since our goal is to predict the customer conversion probability, AUROC(Area under Receiver Operating Curve) is used as the evaluation metric with considering the contributions of both true positive rate and true negative rate. ROC plots the true positive rate (TPR, the proportion of actual customers that are labeled as 1) against the false positive rate (FPR, the proportion of non-customers labeled as 1) as shown in Fig 1.

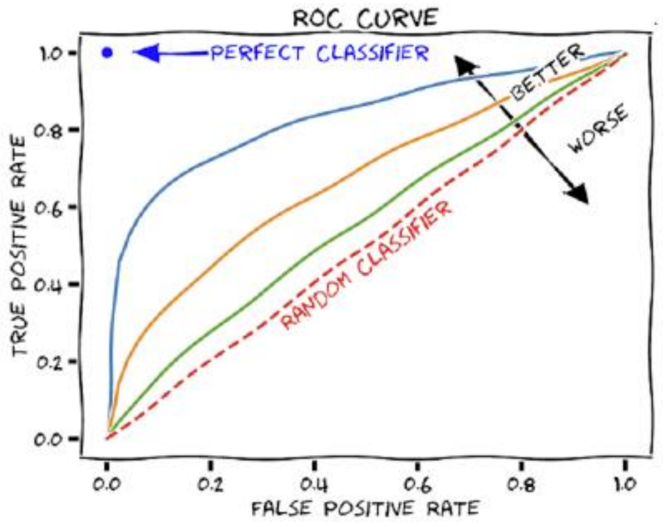


Fig 1 ROC curve [1]

1.4 Project outline

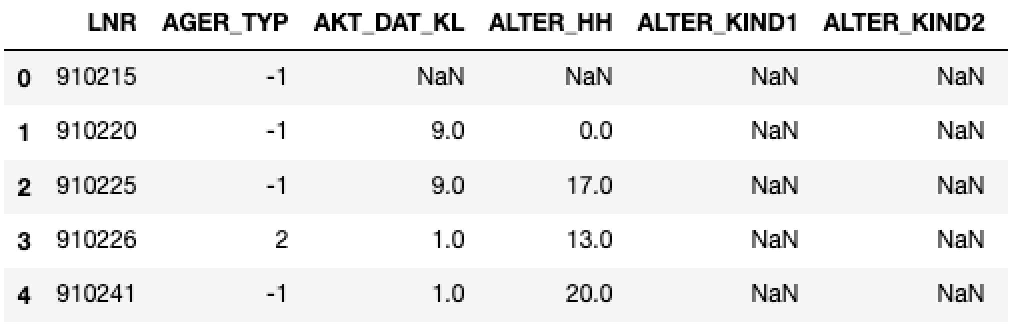
- **Data exploration and cleaning**: the provided dataset is explored and then cleaned through building up an **ETL pipeline** aided with appropriate data visualization.

- **Customer segmentation report**: **PCA** is first conducted on the cleaned dataset to analyze the principal component and feature importance, **171 features** are kept and used for **K-Means** clustering, resulting in **3 clusters** in the customers standing out compared to the general population.
- **Customer conversion probability prediction**: the **mailout** dataset is used for customer conversion probability prediction through supervised learning modeling. A few classifiers are tested with the base model and the best one **XGBoost** is selected for further model improvement. **Bayesian Optimization** is used to fast find the ideal hyperparameters for classification modeling process. GridSearchCV is then used for fine tuning to further improve the model performance.

2. Data exploration and cleaning (ETL pipeline)

2.1 Data precleaning

The **azdias** dataset was first precleaned and studied to understand the data structure. We can see that the row represents each individual and the column represents the features. The features values contains **positive, negative** and **NaN** values.

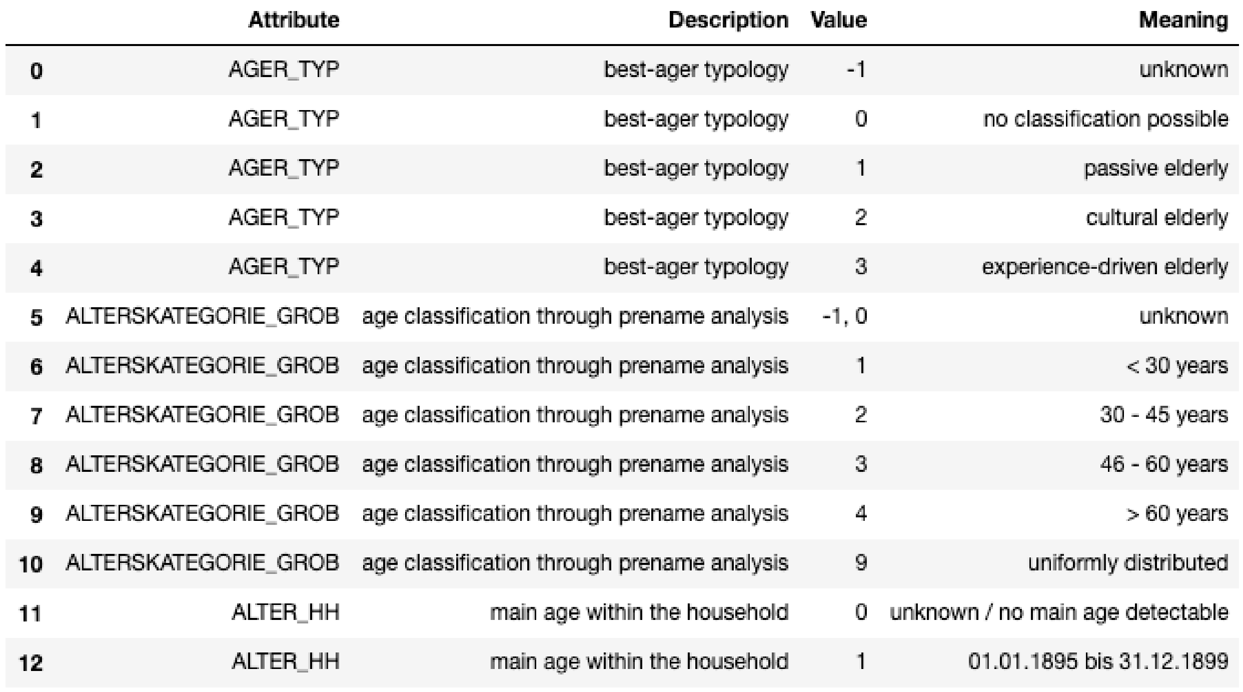


	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2
0	910215	-1	NaN	NaN	NaN	NaN
1	910220	-1	9.0	0.0	NaN	NaN
2	910225	-1	9.0	17.0	NaN	NaN
3	910226	2	1.0	13.0	NaN	NaN
4	910241	-1	1.0	20.0	NaN	NaN

Fig 2 Precleaned azdias dataframe

After browsing the data structure and study the informational spreadsheets, a detailed **feature_df** DataFrame is constructed based on the **DIAS** Attributes spreadsheet as shown in Fig 3. With feature_df, the features as well as their values are studied in detail. The **features contain unknown values** are obtained:

- features_contain_unknowns_n1_only (unknown: -1)
- features_contain_unknowns_0_only (unknown: 0)
- features_contain_unknowns_n1_0_only (unknown: -1, 0)
- features_contain_unknowns_n1_9_only (unknown: -1, 9)
- features_contain_unknowns_n1_0_9_only (unknown: -1, 0, 9)



	Attribute	Description	Value	Meaning
0	AGER_TYP	best-ager typology	-1	unknown
1	AGER_TYP	best-ager typology	0	no classification possible
2	AGER_TYP	best-ager typology	1	passive elderly
3	AGER_TYP	best-ager typology	2	cultural elderly
4	AGER_TYP	best-ager typology	3	experience-driven elderly
5	ALTERSKATEGORIE_GROB	age classification through prename analysis	-1, 0	unknown
6	ALTERSKATEGORIE_GROB	age classification through prename analysis	1	< 30 years
7	ALTERSKATEGORIE_GROB	age classification through prename analysis	2	30 - 45 years
8	ALTERSKATEGORIE_GROB	age classification through prename analysis	3	46 - 60 years
9	ALTERSKATEGORIE_GROB	age classification through prename analysis	4	> 60 years
10	ALTERSKATEGORIE_GROB	age classification through prename analysis	9	uniformly distributed
11	ALTER_HH	main age within the household	0	unknown / no main age detectable
12	ALTER_HH	main age within the household	1	01.01.1895 bis 31.12.1899

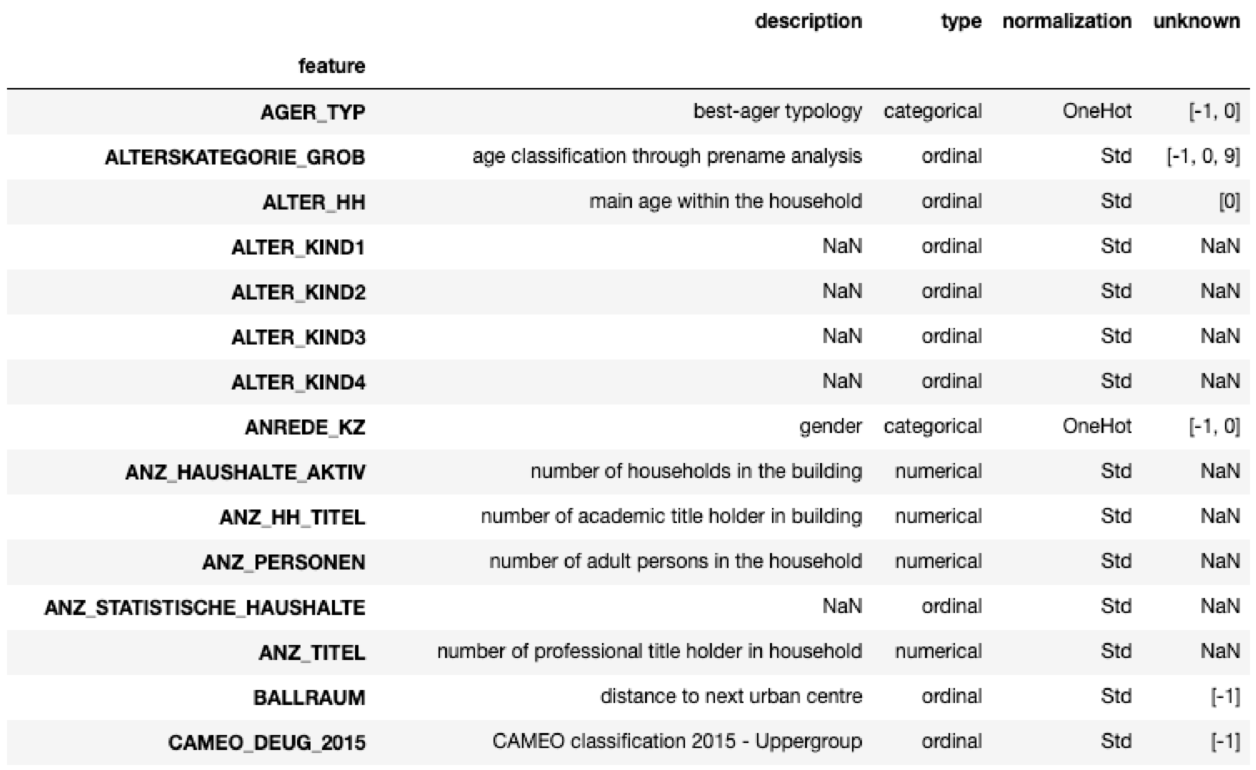
Fig 3 feature_df dataframe extracted from DIAS Attributes

2.2 Drop high dimensional features

By initially study the **azdias**, we can see that there are **366** features, which means that this dataset has already high dimensionality. Thus, some categorical features are initially dropped to reduce the high dimensionality. By comparing the **azdias** and **feature_df**, most of the feature descriptions are identified while there are still **35** features where no descriptions are available. Through some simple statistical analysis, the features which have higher chance to be categorical features or have unknown values exist are dropped to reduce the high dimensionality of the dataset. In summary, total **27** features are dropped.

2.3 Create a DataFrame retained_feature_df

A DataFrame **retained_feature_df** is created with all the retained features as indexes and four columns with the feature description, type, normalization strategy and unknown values as shown in Fig 4.

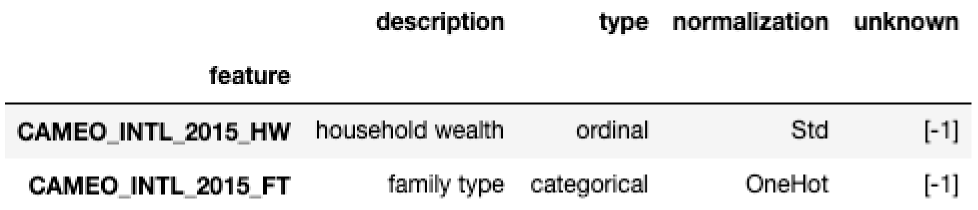


feature	description	type	normalization	unknown
AGER_TYP	best-ager typology	categorical	OneHot	[-1, 0]
ALTERSKATEGORIE_GROB	age classification through prename analysis	ordinal	Std	[-1, 0, 9]
ALTER_HH	main age within the household	ordinal	Std	[0]
ALTER_KIND1	NaN	ordinal	Std	NaN
ALTER_KIND2	NaN	ordinal	Std	NaN
ALTER_KIND3	NaN	ordinal	Std	NaN
ALTER_KIND4	NaN	ordinal	Std	NaN
ANREDE_KZ	gender	categorical	OneHot	[-1, 0]
ANZ_HAUSHALTE_AKTIV	number of households in the building	numerical	Std	NaN
ANZ_HH_TITEL	number of academic title holder in building	numerical	Std	NaN
ANZ_PERSONEN	number of adult persons in the household	numerical	Std	NaN
ANZ_STATISTISCHE_HAUSHALTE	NaN	ordinal	Std	NaN
ANZ_TITEL	number of professional title holder in household	numerical	Std	NaN
BALLRAUM	distance to next urban centre	ordinal	Std	[-1]
CAMEO_DEUG_2015	CAMEO classification 2015 - Uppergroup	ordinal	Std	[-1]

Fig 4 **retained_feature_df DataFrame containing detailed feature information**

2.4 Feature engineering

Feature '**CAMEO_INTL_2015**' combined two types of information: household wealth and family type. This feature is split into two features and the relative features in **retained_feature_df** is also updated with new features: '**CAMEO_INTL_2015_HW**', and '**CAMEO_INTL_2015_FT**'.

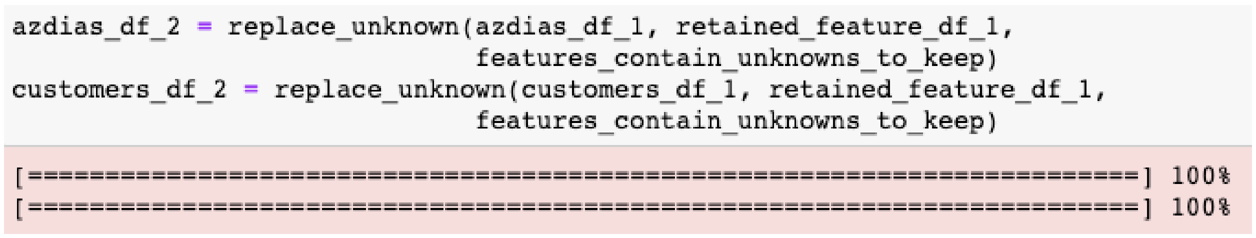


feature	description	type	normalization	unknown
CAMEO_INTL_2015_HW	household wealth	ordinal	Std	[-1]
CAMEO_INTL_2015_FT	family type	categorical	OneHot	[-1]

Fig 5 New features in the **retained_feature_df**

2.5 Fixing the unknowns

The unknown values in the **azdias** dataset are replaced with NaN based on the unknown values for each feature in the **retained_feature_df** .



```
azdias_df_2 = replace_unknown(azdias_df_1, retained_feature_df_1,
                              features_contain_unknowns_to_keep)
customers_df_2 = replace_unknown(customers_df_1, retained_feature_df_1,
                              features_contain_unknowns_to_keep)
```

```
[========================================================================] 100%
[========================================================================] 100%
```

Fig 6 Replacing unknowns

2.6 Drop columns and rows with significant amount of NaN

In this step, columns and rows containing more than 33% of NaN are dropped since it will affect the model accuracy with such significant amount of NaN values.

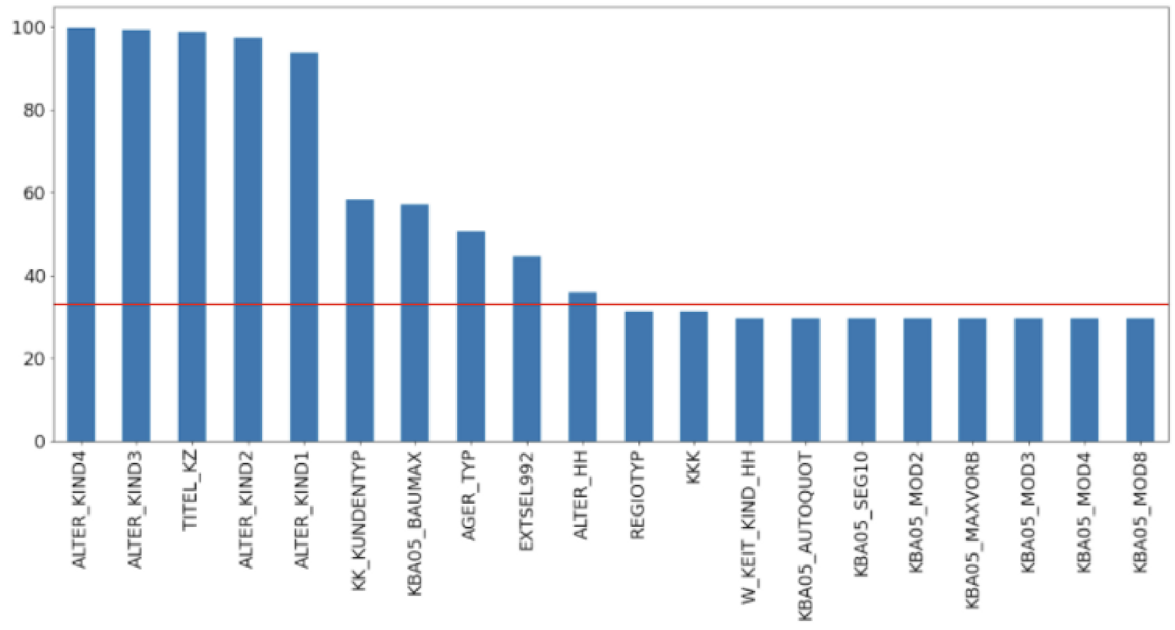
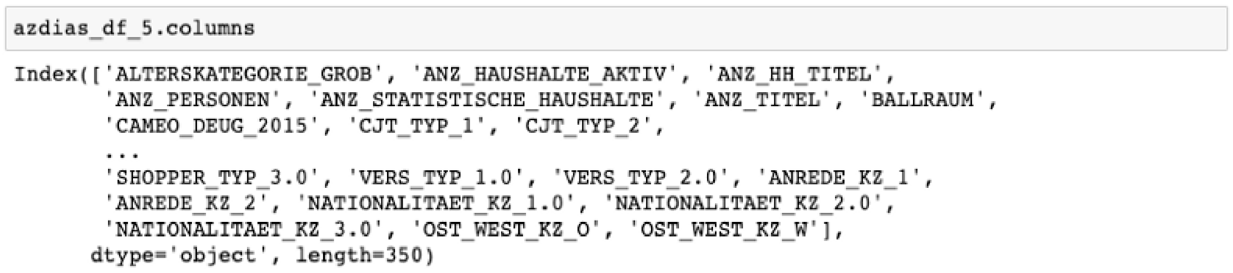


Fig 7 NaN percentage in each columns (red line indicate 33% of NaN)

2.7 One-hot encode the categorical features

The categorical features in **azdias** are one-hot encoded based on the feature types in **retained_feature_df**



```
azdias_df_5.columns
```

```
Index(['ALTERSKATEGORIE_GROB', 'ANZ_HAUSHALTE_AKTIV', 'ANZ_HH_TITEL',
       'ANZ_PERSONEN', 'ANZ_STATISTISCHE_HAUSHALTE', 'ANZ_TITEL', 'BALLRAUM',
       'CAMEO_DEUG_2015', 'CJT_TYP_1', 'CJT_TYP_2',
       ...
       'SHOPPER_TYP_3.0', 'VERS_TYP_1.0', 'VERS_TYP_2.0', 'ANREDE_KZ_1',
       'ANREDE_KZ_2', 'NATIONALITAET_KZ_1.0', 'NATIONALITAET_KZ_2.0',
       'NATIONALITAET_KZ_3.0', 'OST_WEST_KZ_O', 'OST_WEST_KZ_W'],
      dtype='object', length=350)
```

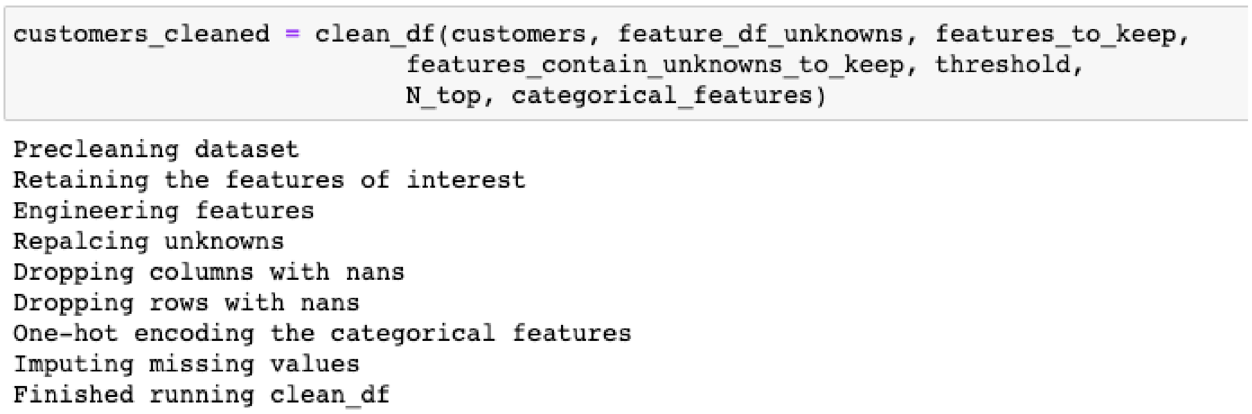
Fig 8 One-hot encoded categorical features

2.8 Imputing missing values

Ideally, in order to better impute the missing values, we can use KNN imputer, however, the computation for this large dataset will be time consuming. Thus, I choose to use a sklearn SimpleImputer to fill the missing values with the mode for each feature, which will be the 'most_frequent' strategy.

2.9 Crate a one-step clean function

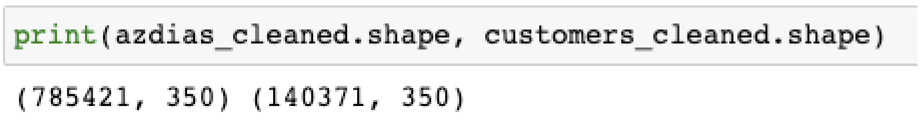
For convenience, a one-step clean function is created by combing step 2.1 to 2.8. The azdias and customers dataset are both cleaned with this function. After cleaning, the azdias and customers datasets are ready for cluster study. Up to this point, there are



```
customers_cleaned = clean_df(customers, feature_df_unknowns, features_to_keep,
                             features_contain_unknowns_to_keep, threshold,
                             N_top, categorical_features)
```

```
Precleaning dataset
Retaining the features of interest
Engineering features
Repalcing unknowns
Dropping columns with nans
Dropping rows with nans
One-hot encoding the categorical features
Imputing missing values
Finished running clean_df
```

Fig 9 **customers** data beingcleaned



```
print(azdias_cleaned.shape, customers_cleaned.shape)
```

```
(785421, 350) (140371, 350)
```

Fig 10 Dimensions of cleaned **azdias** and **customers**

After combing through DIAS Attributes description file, a DataFrame retained_feature_df is created with all the retained features, including their description, type, encode/normalize strategy, and unknown values. The original azdias dataset has been cleaned based on the retained_feature_df by setting the relavant unknown values to NaN, dropping rows and cols with more than **33%** nan, and one-hot encoding the categorical features. The cleaned azdias and customers dataset are saved as **azdias_cleaned** and **customers_cleaned** with **350** features. ETL Pipeline so far:

- preclean() to deal with datatype inconsitency
- retain_features() to drop the high dimensional features
- replace_unknown() to replace unknown to NaN
- drop_cols() to drop columns with more than 33% nan
- drop_rows() to drop rows with more than 33% nan
- one_hot_cat_cols() to one-hot encode categorical features
- imputing_missing_values() to impute missing values

3 Customer segmentation (unsupervised learning)

In this part, I used unsupervised learning techniques to describe the relationship between the demographics of the company's existing customers and the general population of Germany. By the end of this part, the clusters of the general population that are more likely to convert to part of the mail-order company's main customer base are identified.

3.1 Feature scaling

A standard scaler from sklearn is used to normalize both the **azdias_cleaned** and **customers_cleaned** dataset for principal component analysis. The scaled data are saved as **azdias_df_scaled** and **customers_df_scaled**.

3.2 Principal component analysis (PCA)

PCA is often used to reduce data dimension when the data dimension is high. After conducting PCA on the **azdias** dataset, We can see that in Fig 11, 90% of the variance is explained when the number of components is 171. Thus, the number of components can be set as 171 and we can perform the PCA to see the feature importance.

3.3 Feature importance

The feature importance of a few components are discussed. Component 0 as an example,

- Observed from feature 'PLZ8_ANTG*' and 'KBA13_ANTG*', high positive weight for 'number of 1–2 family houses' in the neighborhood and high negative weight to number of 6–10 or > 10 family houses.
- Observed from feature 'LP_STATUS_FEIN' and 'LP_STATUS_GROB', high postive weight for the social status
- Observed from feature 'MOBI_REGIO', high positive weight for moving patterns

```
Number of dimensions:8         that explain variance of 30%
Number of dimensions:16         that explain variance of 40%
Number of dimensions:30         that explain variance of 50%
Number of dimensions:50         that explain variance of 60%
Number of dimensions:79         that explain variance of 70%
Number of dimensions:118         that explain variance of 80%
Number of dimensions:171         that explain variance of 90%
```

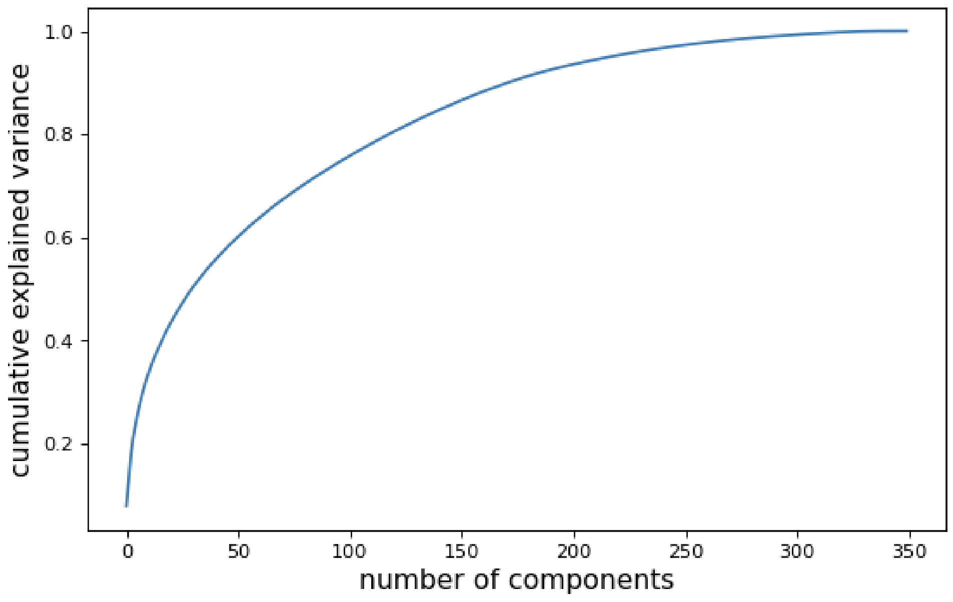


Fig 11 Cumulative explained variance vs. number of components

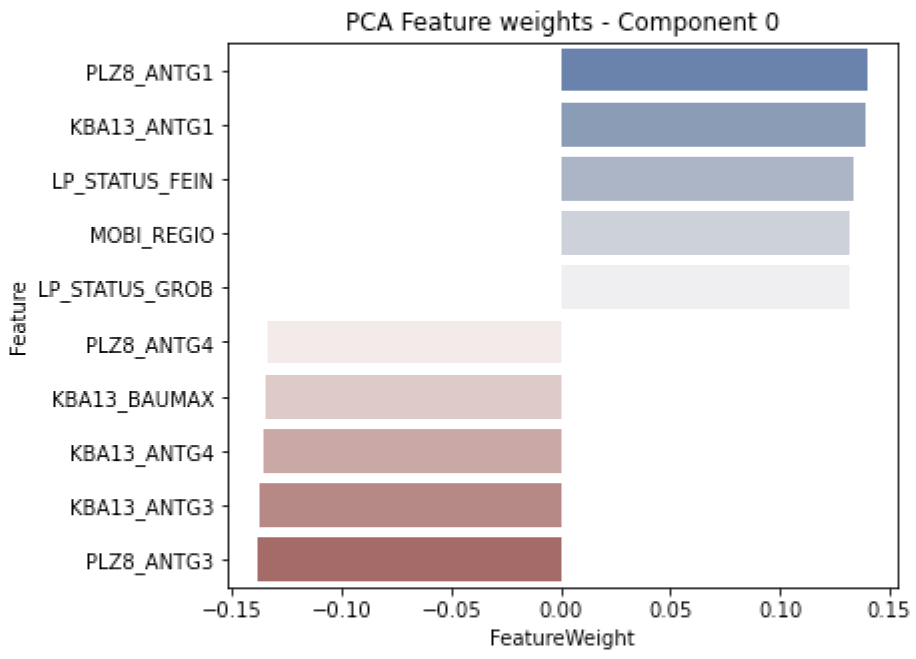


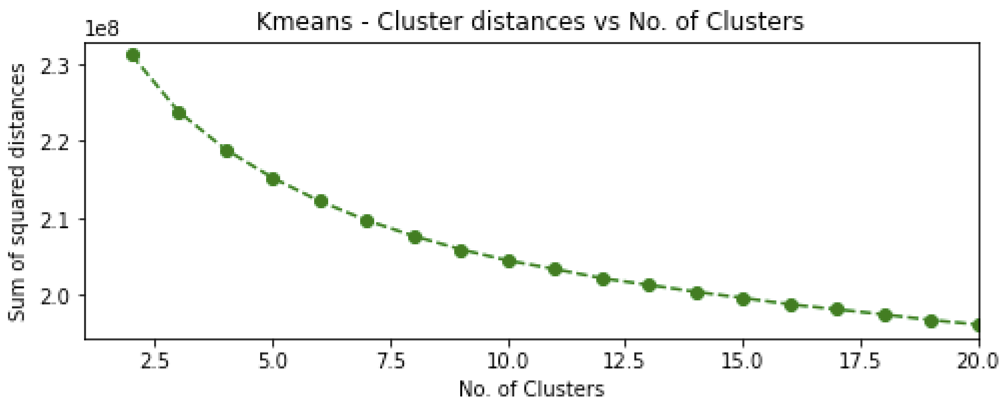
Fig 12 Feature weights for component 0

3.4 K-Means Clustering

After reducing the number of dimensions, we will now use the K-Means Clustering algorithm to cluster the general population into different segments. In this project, the **Elbow method** is used to select the appropriate number of clusters in order to minimize the intra-cluster variation. The total intra-cluster variation measures the compactness of the clustering and we want it to be as small as possible.

The Elbow method [2] looks at the total intra-cluster variation as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't improve the total intra-cluster variation significantly.

In the plotted figure inertia vs number of clusters (Fig 13), the inertia decreasing speed does not change significantly at around **11 clusters** and the slope is almost flat. Thus, 11 clusters will be used for the customer segmentation.



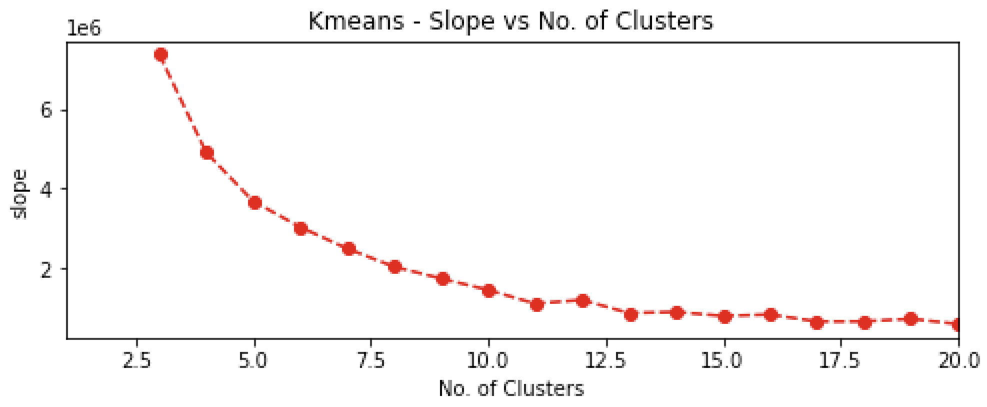


Fig 13 Inertia vs number of clusters

We can observe in Fig 14 that cluster 1, 5 and 8 in the customers dataset have significantly higher percentage compared to the general population. Thus, people have similar demographic properties compared to cluster 1, 5 and 8 will have higher chance as potential customers and can be selected as future marketing target.

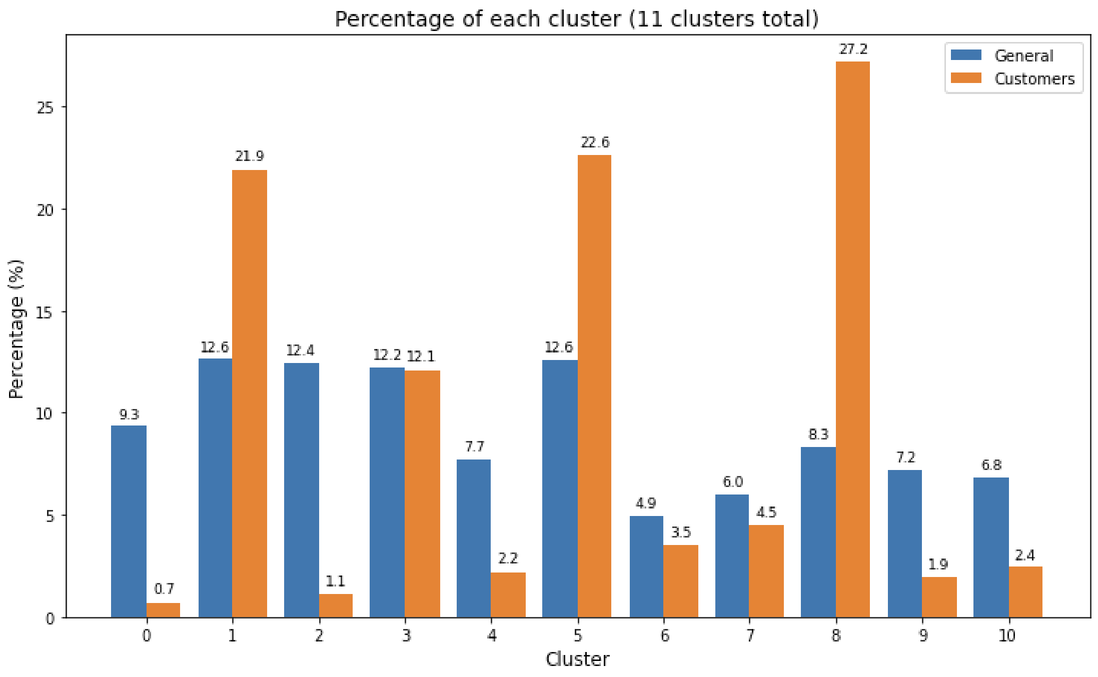


Fig 14 Percentage comparison of each cluster in the general population and customers of the mail-order company

3.5 Cluster interpretation

In this following section, the three clusters 1, 5 and 8 will be studied. The PCA component as well as feature weights that affect the cluster the most will be discussed. For convenience, only the top 2 components (highest weight) and the relative top 5 positive features and top 5 negative features are discussed.

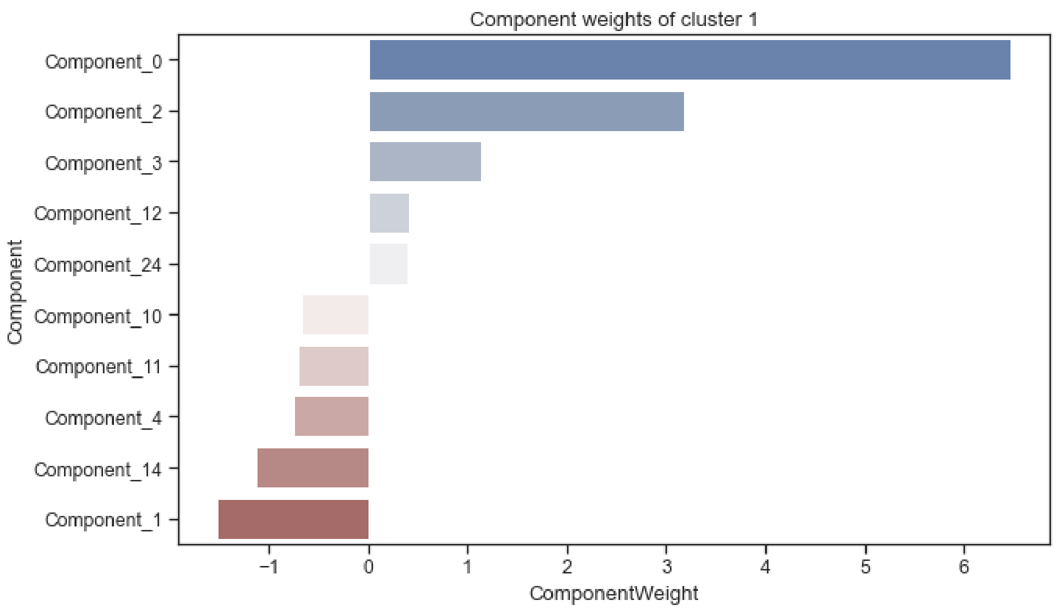


Fig 15 Component weights of cluster 1

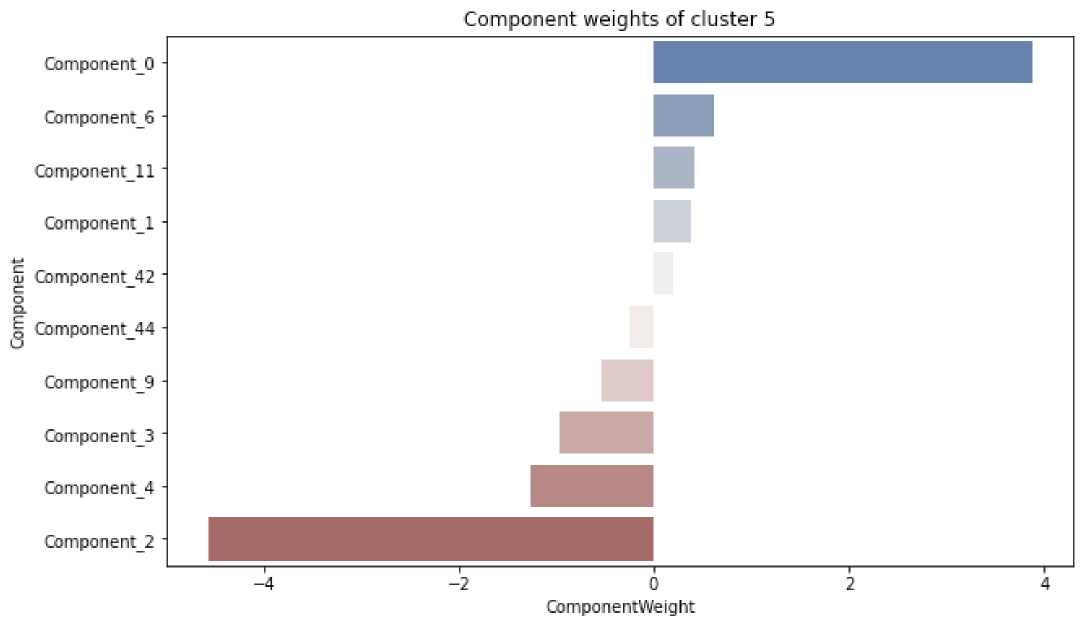


Fig 16 Component weights of cluster 5

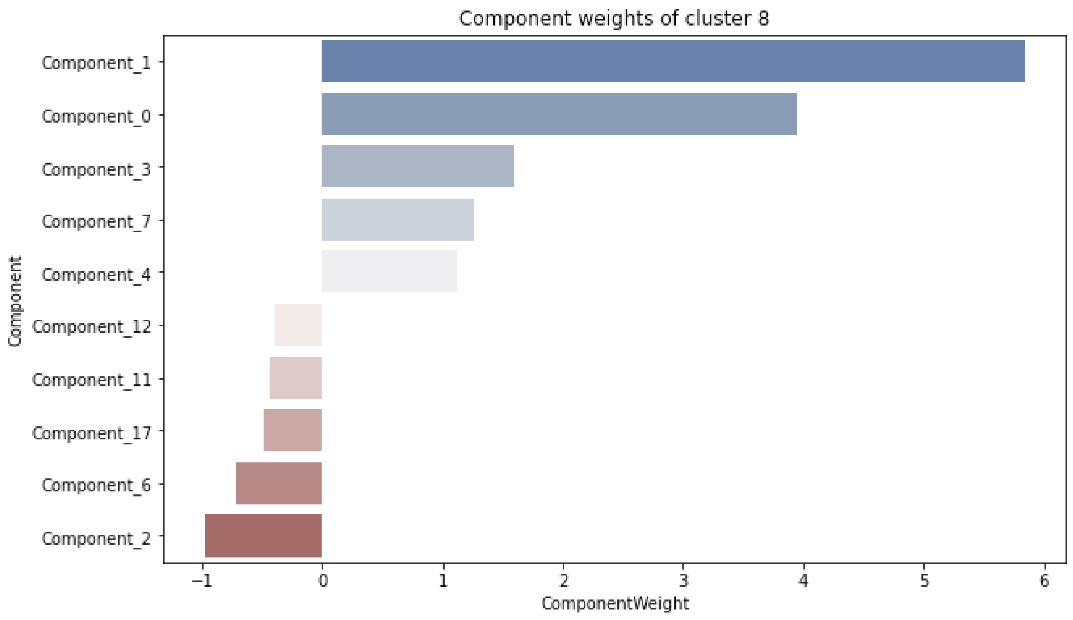


Fig 17 Component weight of cluster 8

From the figures above (Fig 15 to Fig 17), we can observe that component 0 and 2 affect cluster 1 and 5 the most, and component 0 and 1 affect cluster 8 the most. Component 0 and 2 both have positive effect on cluster 1, while component has positive effect and component 2 has negative effect on cluster 5. Component 0 and 1 affect cluster the most and they both have positive effect.

Top features in component 0 (Fig 18):

- Observed from feature 'PLZ8_ANTG*' and 'KBA13_ANTG*', high positive weight for 'number of 1–2 family houses' in the neighborhood and high negative weight to number of 6–10 or > 10 family houses.
- Observed from feature 'LP_STATUS_FEIN' and 'LP_STATUS_GROB', high positive weight for the social status
- Observed from feature 'MOBI_REGIO', high positive weight for moving patterns

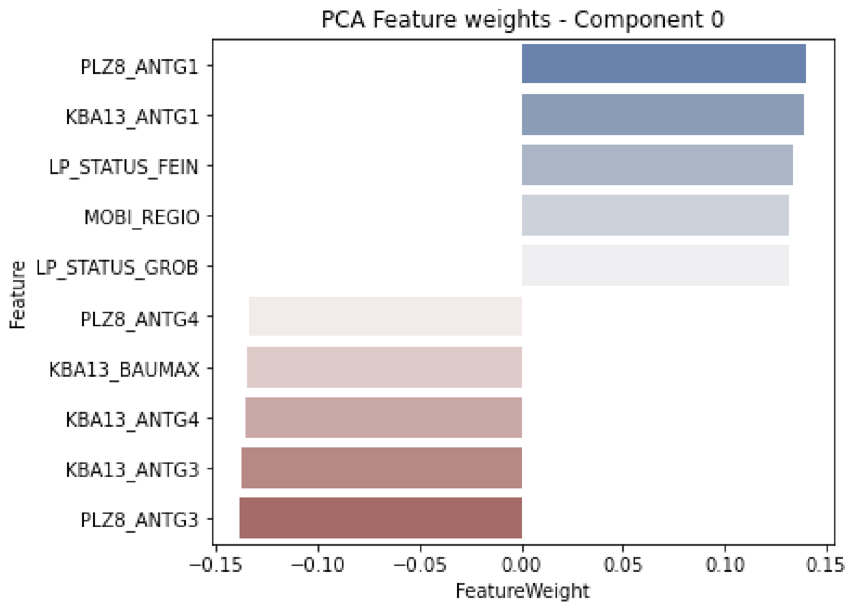


Fig 18 Feature weights of component 0

Top features in component 1 (Fig 19):

- Observed from feature 'KBA13_HERST_BMW_BENZ', 'KBA13_MERCEDES', 'KBA13_BMW', and 'KBA13_SEG_OBEREMITTELKLASSE', the higher the share of BMW & Mercedes, and/or share of upper middle class cars and upper class cars, the higher chance the personal is a potential customer
- Observed from feature 'KBA13_SITZE_4' and 'KBA13_SITZE_5', number of cars with less that 5 seats (sports car or sedan) has positive effect while that more than 5 seats (SUV, van) has negative effects

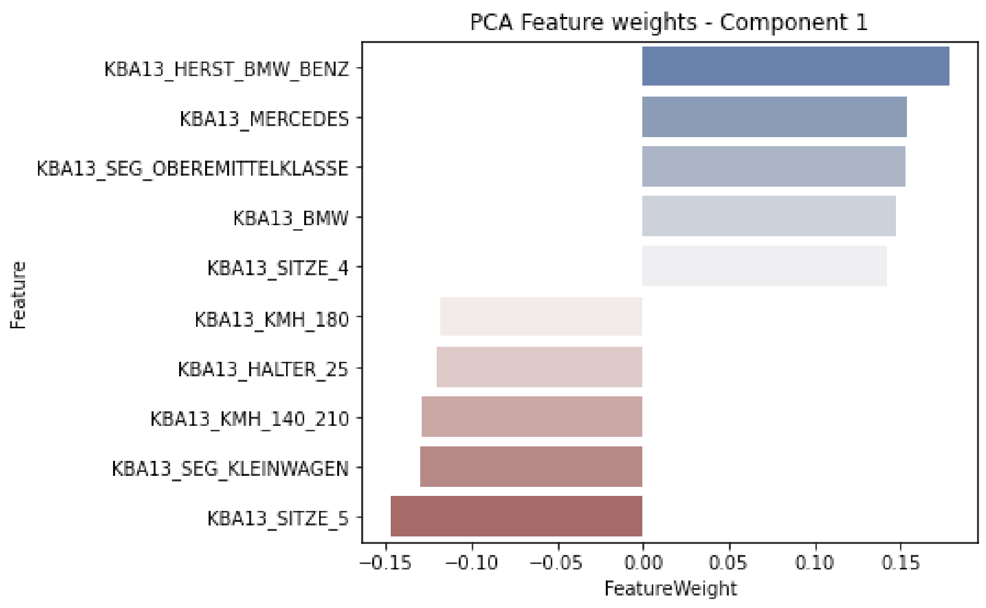


Fig 19 Feature weights of component 1

<u>Top features in component 2</u>: (Fig 20, Reminder: CJT refers to customer journey typology)

- Observed from feature 'ONLINE_AFFINITAET', the higher the online affinity and, the higher chance the personal is a potential customer
- Observed from feature 'CJT_TYP*', type 1 and 2 has positive weight while type 3, 4, and 5 have negative weight
- Observed from feature 'D19_GESAMT_ONLINE_DATUM', high negative weight indicates that the higher the transaction activity, the higher chance the personal is a potential customer

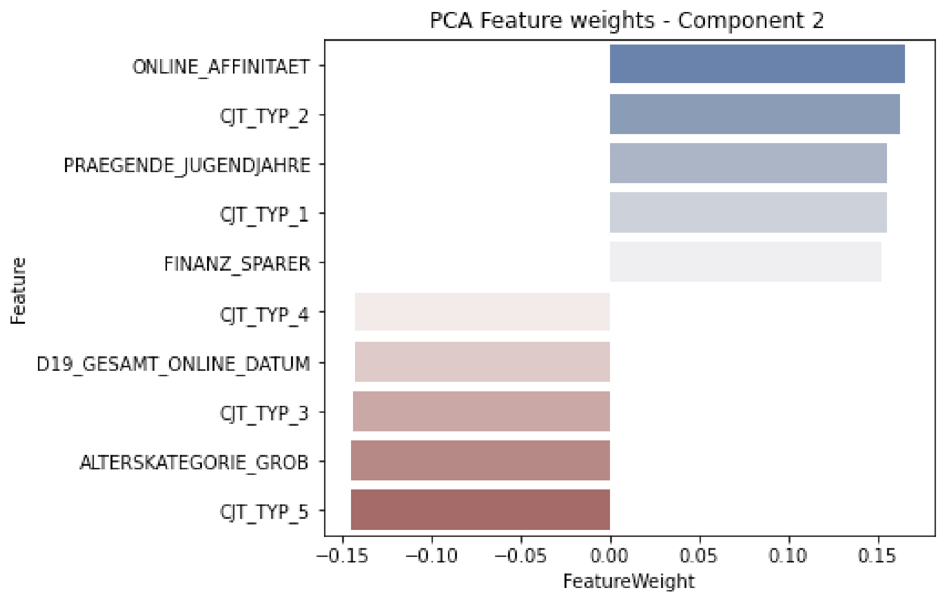


Fig 20 Feature weights of component 2

4 Customer conversion probability prediction (supervised learning model)

Each of the rows in the **mailout** data files represents an individual that was targeted for a mailout campaign. Ideally, we should be able to use the demographic information from each individual to decide whether or not it will be worth it to include that person in the campaign.

The **mailout** data has been split into two approximately equal parts: **mailout_train** and mailout_test, each with almost 43 000 data rows. In this part, my model is first trained and verified with the "TRAIN" partition, which includes a column, "RESPONSE", that states whether or not a person became a customer of the company following the campaign. Then the model is used to create predictions on the "TEST" partition, where the "RESPONSE" column has been withheld.

4.1 Data cleaning and preparation

The dataset mailout_train was cleaned using the same process as in section 3 except no rows were removed since we are building a prediction model. The cleaned data was scaled using StandardScaler from sklearn. For verification purpose, the dataset was split into two datasets: 70% for train and 30% for verification.

4.2 Benchmark

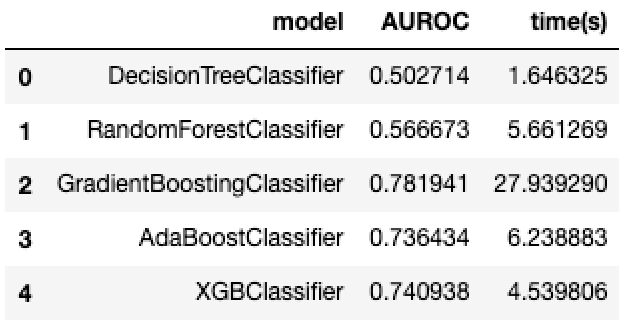
The benchmark I selected is to use XGBClassifier base model. XGBoost can provide a high-performance implementation of gradient boosted decision trees, and it becomes many Kaggle competition winning model. I trained the base model on the 70% train data and got 0.74(AUROC score) on the verification data. I will compare the future results with this bench mark and further improve the model. Considering the high dimensionality of the data, I would try to achieve 0.8 AUROC score at the end (top score on Kaggle leader board is 0.84).

4.3 Classifier comparison

I selected a few often-used classifiers and used the base model on the train data and then verify with the verification data:

- GradientBoostingClassifier
- RandomForestClassifier
- GradientBoostingClassifier
- AdaBoostClassifier
- XGBClassifier

In this project, since the data is highly unbalanced with more than 98% of the data response is zero, accuracy is not appropriate for the model evaluation. Since our goal is to predict accurately for response 0 and 1, AUROC(Area under Receiver Operating Curve) is used as the evaluation metric with considering the contributions of both true positive rate and true negative rate.



	model	AUROC	time(s)
0	DecisionTreeClassifier	0.502714	1.646325
1	RandomForestClassifier	0.566673	5.661269
2	GradientBoostingClassifier	0.781941	27.939290
3	AdaBoostClassifier	0.736434	6.238883
4	XGBClassifier	0.740938	4.539806

Fig 21 Classifier comparison

The AUROC and calculation time is shown in Fig 21. We can see that GradientBoostingClassifier results in the best AUROC compared to the other classifiers, however, it takes much longer computation time. XGBClassifier results in the second best AUROC and it takes much shorter time to calculate compared with GradientBoostingClassifier. In this project, XGBClassifier is used for further hyperparameter tuning considering both the time and AUROC performance. This model was tested on the test data in the Kaggle competition with a

4.4 Bayesian Optimization

Before hyperparameter tuning, a **Bayesian Optimization** [3] is performed to fast identify the parameter combinations using a XGBClassifier that will result in high AUROC on the verification data in the preselected bounds for those parameters. The following hyperparameter values results in 0.9996 for the train data and 0.7893 for the verification data.

- colsample_bytree = 0.8
- gamma = 0.1
- learning_rate = 0.03
- max_depth = 5
- min_child_weight = 2
- reg_alpha = 0.02
- n_estimators = 500
- subsample = 0.5

After fitting this model on the test data in the Kaggle competition, the score is **0.78918**, which is a fairly good score. I notice that the score on the training data is almost 1, which indicates overfitting. Thus, I tried another parameter combination which gives fairly high verification score 0.7842 and a little lower training score 0.9853. And this model yields the test score of **0.79599** on the Kaggle competition platform.

4.5 Hyperparameter tuning

To further improve the model, **GridSearchCV** is used to find the best combinations of hyperparameters by fine tuning the parameters identified using the **Bayesian Optimization.** The best parameters are:

- colsample_bytree = 0.7
- gamma = 0.2
- learning_rate = 0.01
- max_depth = 5
- min_child_weight = 2
- reg_alpha = 0.05
- n_estimators = 500
- subsample = 0.5

with a training score 0.9532, and a verification score 0.7677. This model is then evaluated on the Kaggle competition platform, resulting in a score **0.80184**.

4.6 Feature importance

In the figure below, the top 30 important features are plotted. By referring to the features_df DataFrame, we can summarize some important features that affect the model the most: family type, features about share of cars, most common building type, and health typology et al.

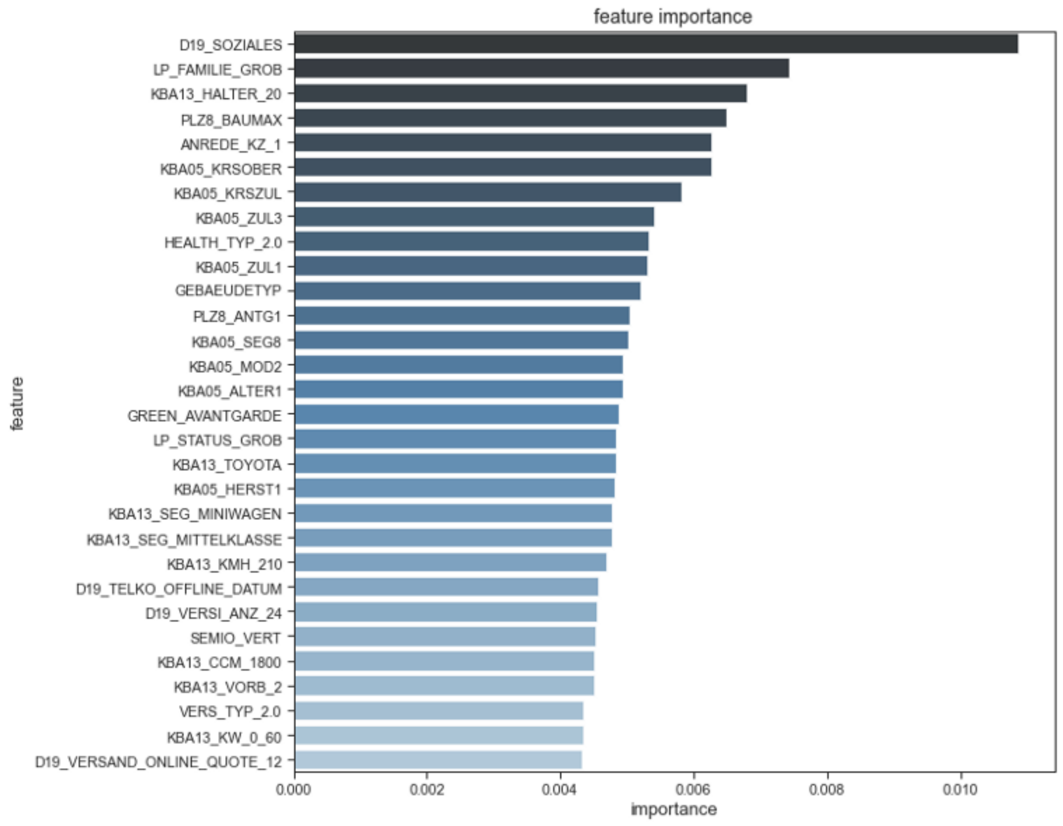
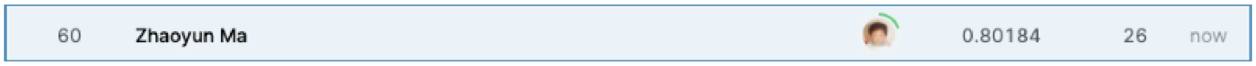


Fig 22 Feature importance of the fine tuned XGBClassifier model

5 Kaggle competition

The final model used for the Kaggle competition is the fine-tuned model with the parameters listed in section 3.4 and the final test score is **0.80184**, which is 0.0456 lower than the top score 0.84739.



60	Zhaoyun Ma	0.80184	26	now

Fig 23 Kaggle competition leaderboard

The model can be further improved by trying out more iterations using **Bayesian Optimization** and **GridSearchCV** with a larger parameter grid can be conducted to find better hyperparameters.

6 Summary and conclusions

In this project, I analyzed demographics data for customers of a mail-order sales company in Germany, comparing it against demographics information for the general population. I first used **PCA** and unsupervised learning techniques (**K-Means**) to perform customer segmentation. Then, I applied what I've learned on a third dataset with demographics information for targets of a marketing campaign for the company and used a supervised learning model (**XGBoost**) to predict which individuals are most likely to convert into becoming customers for the company. The final model results in a score **0.80184** on the Kaggle platform**.** The important findings are summarized as following:

- Data understanding and cleaning are the foundation of any data science project, and they are critical for the modeling process to get more accurate classification and prediction.
- PCA is performed for the unsupervised learning model, 171 features (90% of cumulative variance are explained) are used for the K-Means cluster.
- **Three distinguish clusters** are identified for the customers compared to the general population in German. **Component 0, 1 and 2** have the highest weights for those clusters. Important features in those components are **numbers of different size of families**, **share of certain type of cars**, and **transaction activeness** et al.
- **Bayesian Optimization** is a fast way to find the ideal hyperparameters for classification modeling process. **GridSearchCV** can be used for fine tuning to further improve the model.
- The fine-tuned model yields a **0.80184** on the Kaggle platform, which is 0.046 lower than the top score 0.84739 in the Kaggle competition leaderboard.

References

[1] https://commons.wikimedia.org/wiki/File:Roc-draft-xkcd-style.svg
[2] https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/
[3] https://github.com/fmfn/BayesianOptimization