Bertelsmann/Arvato Capstone Project Report

Yusong Zhou

October 8, 2020

1 Problem Definition

Bertelsmann is a media, services and education company in Germany. It's one of the largest media conglomerates in the world and have divisions in music, printing and investments as well. Arvato, the service provider division of Bertelsmann, is an internationally active services company that develops and implements innovative solutions for business customers from around the world. The objective of this project is to use labeled and unlabeled datasets provided by them to make predictions of whether a certain general individual will become a customer of Bertelsmann.

Customer segmentation involves identifying certain groups of people with a usually huge demographic dataset. The size of the dataset, especially the number of features contained, makes machine learning models suitable for this job, as other simple statistical models may not do well with such a high dimension of features.

This project will use both unsupervised and supervised machine learning models to first classify a population sample into several groups and then try to make predictions of whether a certain customer will respond to the company, based on another sample with the same set of features.

The unsupervised learning problem involves using two of the datasets, one for the general population and the other for some former customers of the Germany company. A clustering model will be applied to find certain groups in the general population that's most similar to the most of the former customers.

The supervised learning problem involves using the third dataset with a response column indicating whether this person replied the mail to predict the reaction of people from the last dataset, where the response column is empty. A regression model will be applied to generate a possibility for the empty response column.

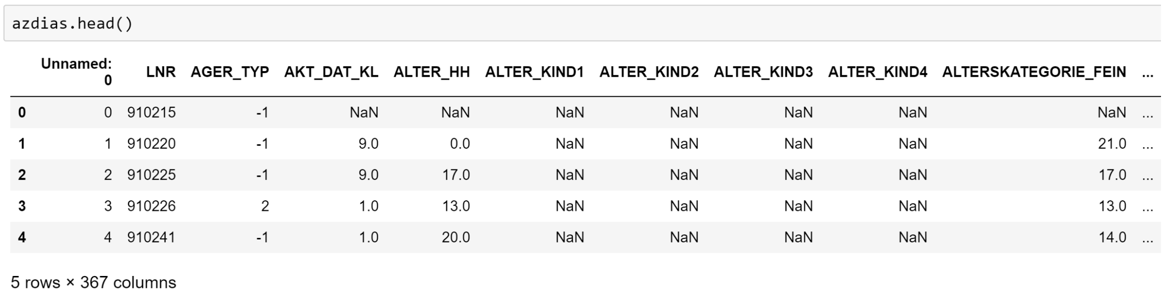
2 Data Exploration and Cleaning

2.1 Datasets

Four main datasets contains the general population and customers datasets for unsupervised learning task and a labeled train set together with an unlabeled test set for supervised learning task.

- *Udacity_AZDIAS_052018.csv*: contains 891211 people × 366 features, denoting general population. Most of the features are numerical types and a few are string;
- *Udacity_CUSTOMERS_052018.csv*: contains 191652 people × 369 features, denoting customers of the company. It has three additional features compared to azdias dataset;
- *Udacity_MAILOUT_052018_TRAIN.csv*: contains 42982 people × 367 features, denoting target people of marketing. It has one additional feature compared to azdias dataset, which is individual's response;
- *Udacity_MAILOUT_052018_TEST.csv*: contains 42833 people × 366 features, denoting test set of marketing without the response record of the last file.

Since these four datasets are mostly similar, in Figure 1 I will present the first several rows of AZDIAS, one of the dataset, only.



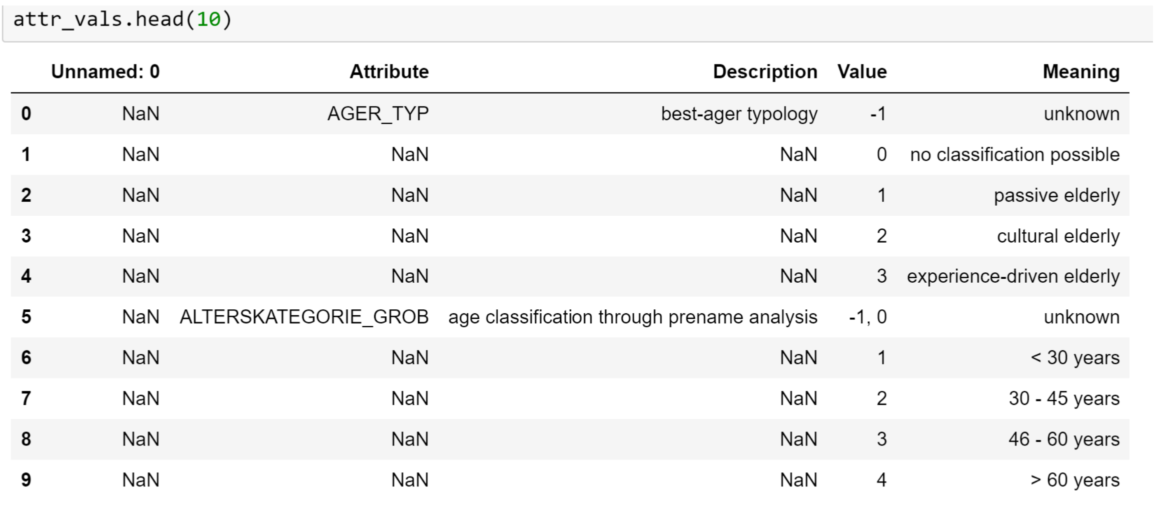
```
azdias.head()
```

	Unnamed: 0	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	...
0	0	910215	-1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
1	1	910220	-1	9.0	0.0	NaN	NaN	NaN	NaN	21.0	...
2	2	910225	-1	9.0	17.0	NaN	NaN	NaN	NaN	17.0	...
3	3	910226	2	1.0	13.0	NaN	NaN	NaN	NaN	13.0	...
4	4	910241	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	...

5 rows × 367 columns

Figure 1: The AZDIAS Dataset

Figure 2 shows the first several rows of *DIAS Attributes - Values 2017.xlsx* dataset. We can see it contains the explanation of all values of every feature. For example, for the attribute *ALTERSKATEGORIE_ GROB*, it represents *age classification through prename analysis*. Values 1, 2, 3, 4 corresponds to different age intervals, -1 and 0 represents unknown data. Note here the missing values are represented as numerical values, this is an issue I will address later in subsection 2.3.

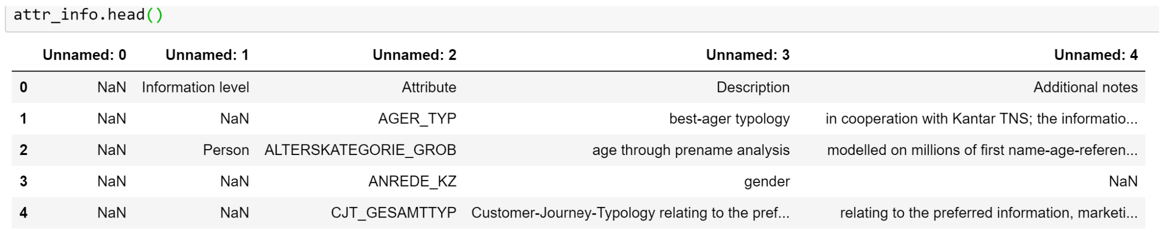


```
attr_vals.head(10)
```

	Unnamed: 0	Attribute	Description	Value	Meaning
0	NaN	AGER_TYP	best-ager typology	-1	unknown
1	NaN	NaN	NaN	0	no classification possible
2	NaN	NaN	NaN	1	passive elderly
3	NaN	NaN	NaN	2	cultural elderly
4	NaN	NaN	NaN	3	experience-driven elderly
5	NaN	ALTERSKATEGORIE_GROB	age classification through prename analysis	-1, 0	unknown
6	NaN	NaN	NaN	1	< 30 years
7	NaN	NaN	NaN	2	30 - 45 years
8	NaN	NaN	NaN	3	46 - 60 years
9	NaN	NaN	NaN	4	> 60 years

Figure 2: Attributes - Values

Figure 3 shows the first several rows of *DIAS Information Levels - Attributes 2017.xlsx* dataset. For every attribute, it gives a brief description and some additinal notes.



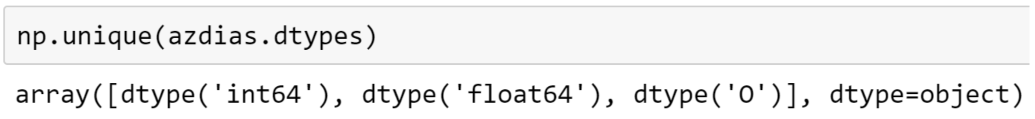
```
attr_info.head()
```

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	NaN	Information level	Attribute	Description	Additional notes
1	NaN	NaN	AGER_TYP	best-ager typology	in cooperation with Kantar TNS; the informatio...
2	NaN	Person	ALTERSKATEGORIE_GROB	age through prename analysis	modelled on millions of first name-age-referen...
3	NaN	NaN	ANREDE_KZ	gender	NaN
4	NaN	NaN	CJT_GESAMTTYP	Customer-Journey-Typology relating to the pref...	relating to the preferred information, marketi...

Figure 3: Attributes Information Levels

2.2 Data Types

First let's take a look at the data types of azdias' columns. Figure 4 indicates there are int, float and object types, the later refers to string or mixed types in Pandas. Columns that contain string need to be transferred to numerical type as input to the model. But first we need to know which columns are of type object.

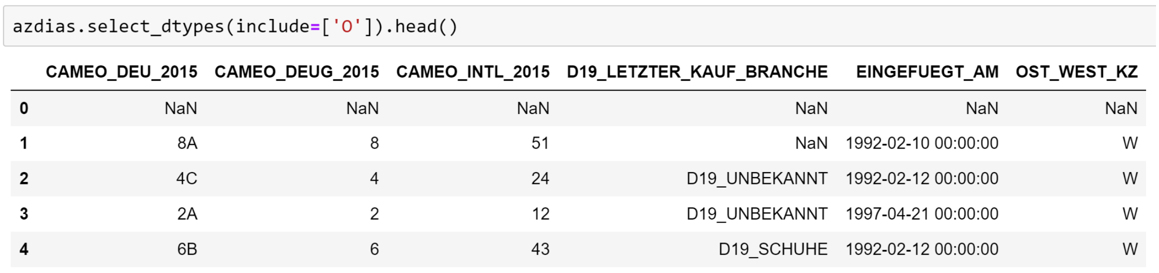


```
np.unique(azdias.dtypes)
```

```
array([dtype('int64'), dtype('float64'), dtype('O')], dtype=object)
```

Figure 4: Data Types

Figure 5 shows the columns with object type. The second and third column actually have numerical entries but also contain string "X" or "XX" values. The fifth column is time, which is not useful for the learning job. Other columns are categorical values that can be replaced by integers.



```
azdias.select_dtypes(include=['O']).head()
```

	CAMEO_DEU_2015	CAMEO_DEUG_2015	CAMEO_INTL_2015	D19_LETZTER_KAUF_BRANCHE	EINGEFUEGT_AM	OST_WEST_KZ
0	NaN	NaN	NaN	NaN	NaN	NaN
1	8A	8	51	NaN	1992-02-10 00:00:00	W
2	4C	4	24	D19_UNBEKANNT	1992-02-12 00:00:00	W
3	2A	2	12	D19_UNBEKANNT	1997-04-21 00:00:00	W
4	6B	6	43	D19_SCHUHE	1992-02-12 00:00:00	W

Figure 5: String Columns

The rule of processing string columns can be summarized as:
• String to original numerical values: CAMEO_DEUG_2015, CAMEO_INTL_2015;
• String to categorical integers: CAMEO_DEU_2015, D19_LETZTER_KAUF_BRANCHE, OST_WEST_KZ;
• Drop: EINGEFUEGT_AM

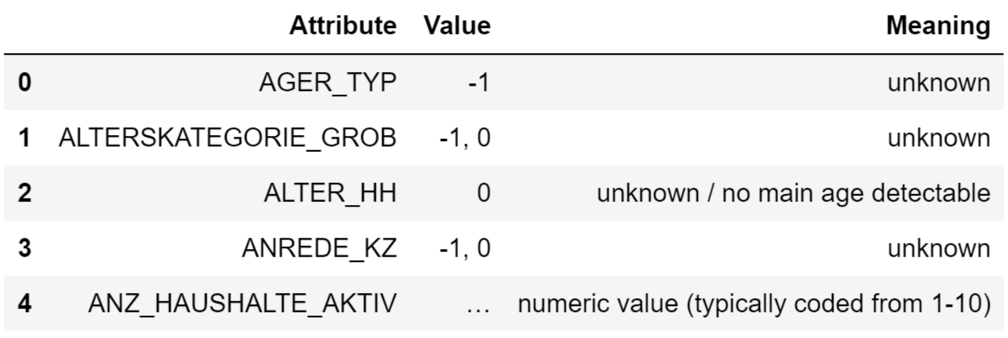
As the columns of azdias dataset are the common columns of four main datasets, the processing procedure can be applied to all other datasets. Thus I wrote a cleaning function for dealing with string columns, and I will do the same for all other processing procedures.

2.3 Numerical Missing Data

From Figure 2 we can know that there're many unkown values recorded as numerical values: -1, 0 or 9. These are actually missing values but not represented as NaNs. My job here is to identify these values from the original dataset and replace them with NaNs, mainly because numerical values will have an effect on the final result but they do not in fact contain any useful information.

To make use of the information provided in Figure 2, I select only the first rows of each attribute, for they may be the numerical value of the unknown type, if there is unknown type denoted in this attribute. The result table is Figure 6. Another question arises as not all first rows are about missing or unknown values, see the fifth row of Figure 6.

My solution is choosing only rows with keyword "unknown" in the "Meaning" column and classify the attributes according to the numerical values of unknown values. It turns out there are three types for representing unknown values: [-1, 0], [-1, 9] and [-1]. The processing method is to save attribute names for these three types and replace -1, 9 or 0 with NaNs, in relevant datasets.



	Attribute	Value	Meaning
0	AGER_TYP	-1	unknown
1	ALTERSKATEGORIE_GROB	-1, 0	unknown
2	ALTER_HH	0	unknown / no main age detectable
3	ANREDE_KZ	-1, 0	unknown
4	ANZ_HAUSHALTE_AKTIV	…	numeric value (typically coded from 1-10)

Figure 6: Missing Values as Numerical Types

2.4 Missing Data

Missing data is an inevitable part of data cleaning, as not all datasets are perfect with all values available, and these missing values cannot be directly fed into the model. My strategy is to use a *SimpleImputer* from *sklearn library*, which fills NaNs with the average values in corresponding columns. But first let's plot the distribution of the ratio of missing values across all columns and rows to get an overview.

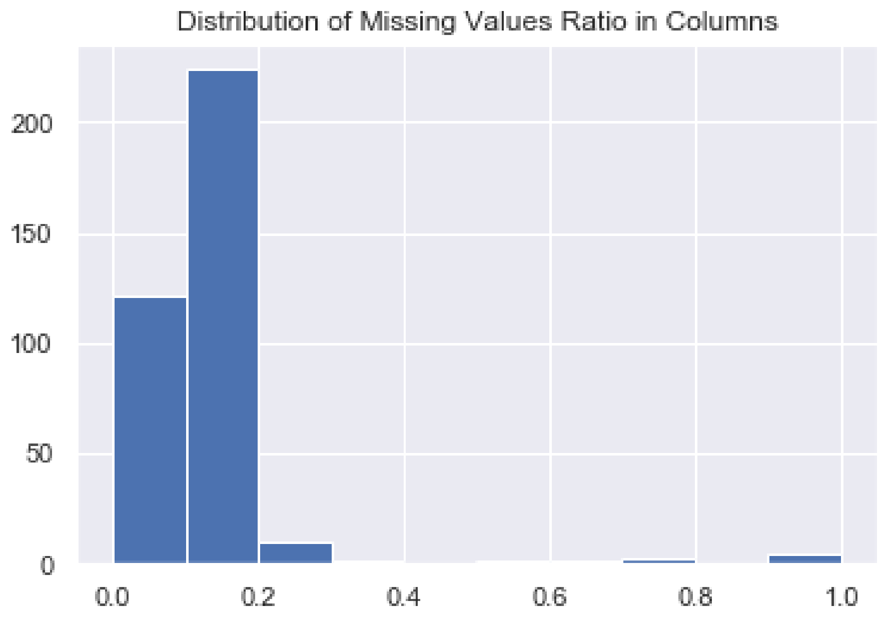


Figure 7: Distribution of Missing Values in Columns

Figure 7 shows the distribution across all columns after preliminary cleaning steps. Most columns, approximately 300 of them, have less missing values than 30 percent of all values. And several columns are almost empty with less than 10 percent of the values are not missing. Dropping columns needs careful consideration, as all records or rows share the same set of features. Dropping one column means reducing one feature for all 891211 records, in the case of azdias dataset. Fortunately here the distribution is quite polarized, and choosing a middle threshold value will get rid of columns with too many missing values without discarding useful information in other columns. The threshold missing value ratio is 0.5, a fair choice between 0 and 1.

Figure 8 shows the distribution for rows. We can see that the distribution is a little different from the columns as it's even more polarized to two endpoints, 0 and 0.7. Most of the records tend to be completed, and more records contain too many missing values, compared to the above one. But here the right endpoint is 0.7 rather than 1. Dropping rows is less serious than dropping columns as this would not affect the whole dataset as columns does. So I also choose the threshold to be 0.5, regardless of the right endpoint of 0.7.

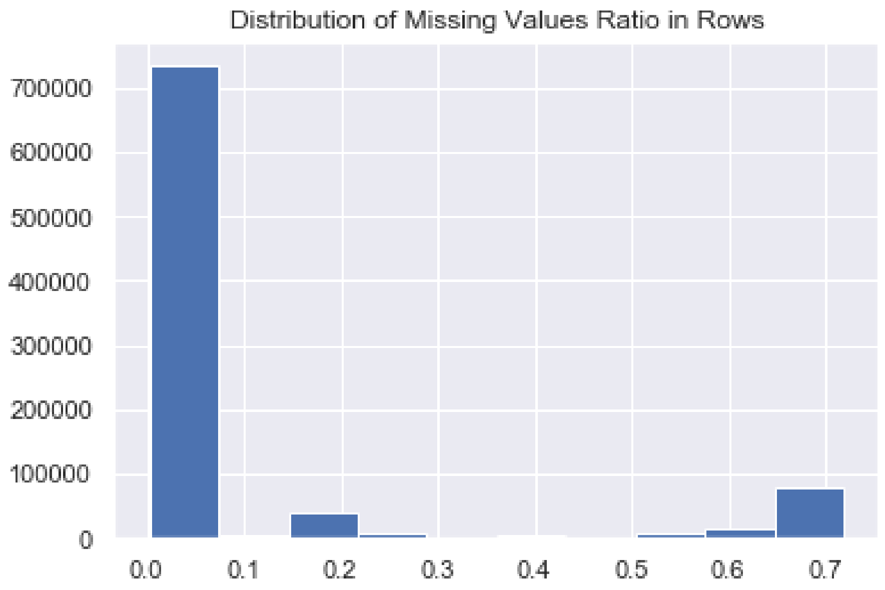
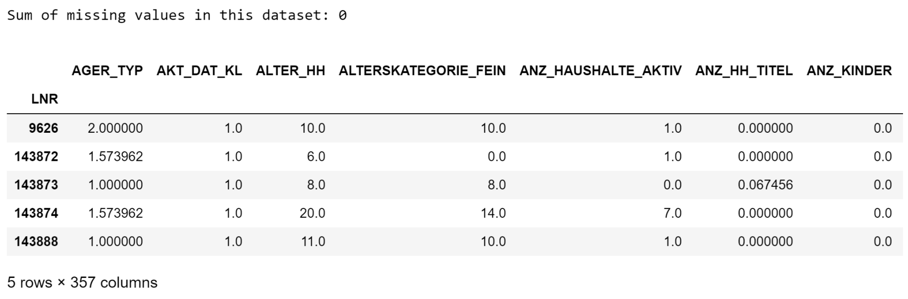


Figure 8: Distribution of Missing Values in Rows

Another point to make is that dropping rows may not be an action that applies to all datasets. For the test set in the supervised learning task, it would be unwise or even ridiculous to do so, as predictions have to be made for all rows. So I set a Boolean parameter in the cleaning function to indicate whether rows with too many missing values will be dropped. For the mentioned test set, it's value should be *False*.

2.5 Integrated Cleaning

All three steps of data cleaning are integrated into a single cleaning function for the ease of calling. And the function includes a step of resetting the index to be the *LNR* column, which is exactly the original individual index. Figure 9 shows the head of the processed cutomer dataset as a test.



```
Sum of missing values in this dataset: 0
```

	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER
LNR							
9626	2.000000	1.0	10.0	10.0	1.0	0.000000	0.0
143872	1.573962	1.0	6.0	0.0	1.0	0.000000	0.0
143873	1.000000	1.0	8.0	8.0	0.0	0.067456	0.0
143874	1.573962	1.0	20.0	14.0	7.0	0.000000	0.0
143888	1.000000	1.0	11.0	10.0	1.0	0.000000	0.0

5 rows × 357 columns

Figure 9: Head of Processed Customer Dataset

3 The Unsupervised Learning Model

The customer segmentation analysis will come in this section. Here, I will use 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, I will use the result to describe parts of the general population that are more likely to be part of the mail-order company's main customer base, and which parts of the general population are less so.

3.1 Correlation Filtering

To begin with, I'll delete columns with high correlation to another, for two reasons: First, the following PCA algorithm will reduce the dimension of the feature space, and features with high correlations will be reduced anyway; second the azdias dataset is of huge size and simplify the dataset in advance will improve the running time, especially for possible many loops (and there will be).

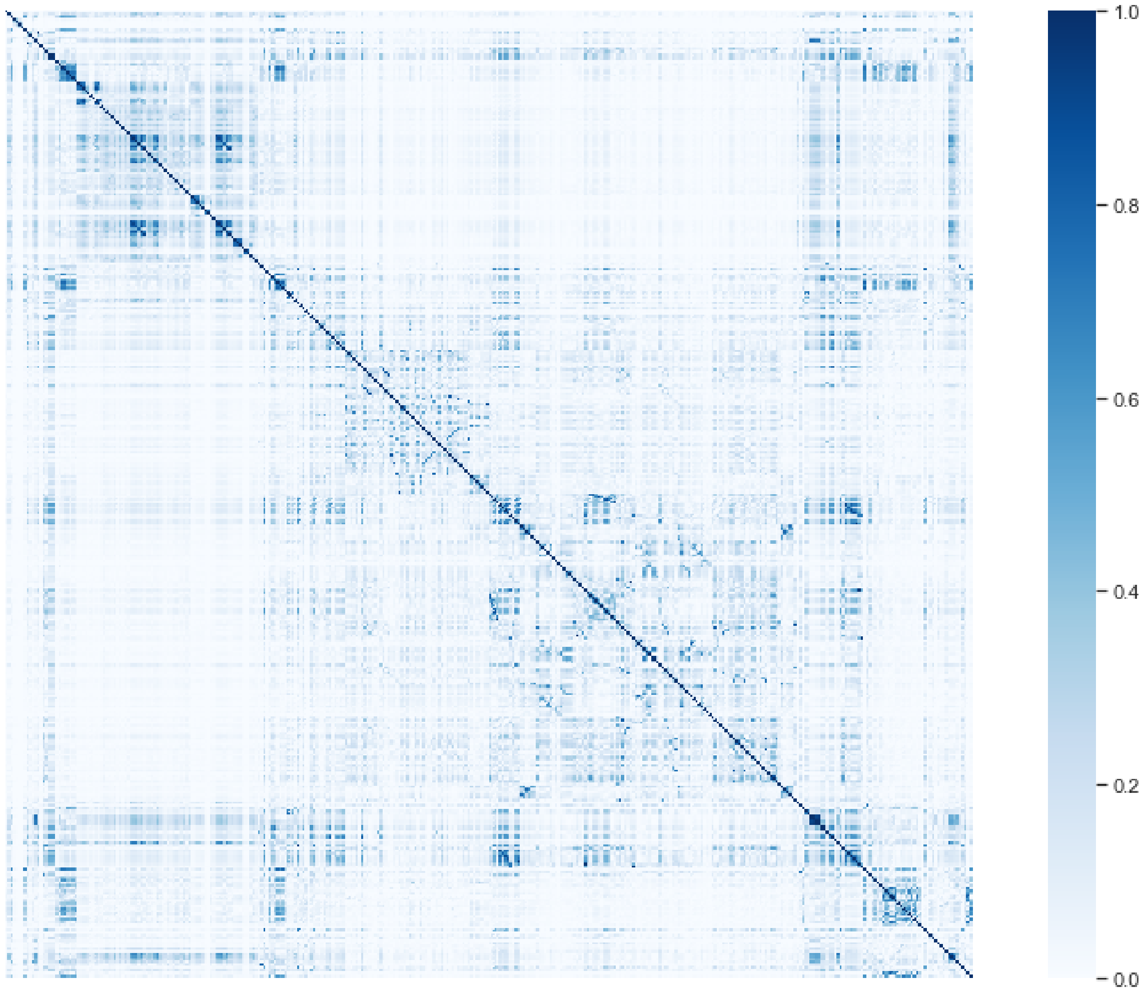
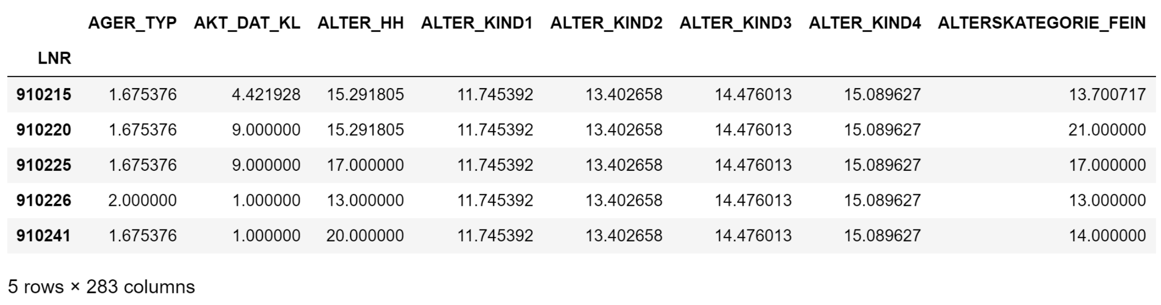


Figure 10: Correlation Matrix Heatmap of Azdias Dataset

Figure 10 is the correlation matrix heatmap of Azdias dataset. The heatmap is quite large due to the amounts of features. We can see that outside the diagonal, which has values of 1, there are other small squares with deep blue colored, indicating a high correlation.

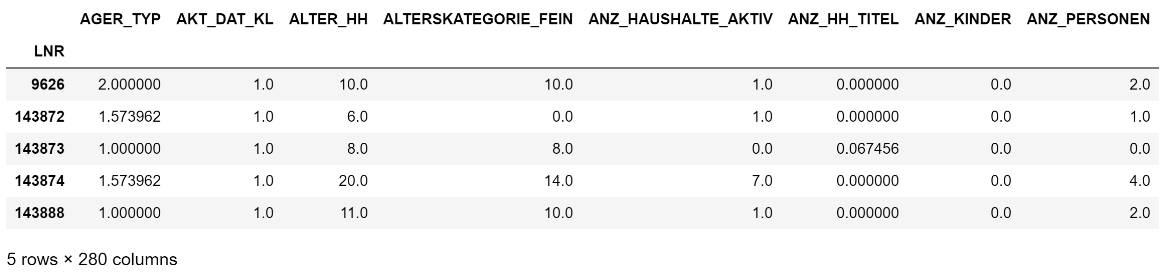
My strategy is to select these high correlation pairs and drop the corresponding columns. The threshold correlation value is set to be 0.75. As the correlation matrix is symmetrical by the diagonal line, I select only from the upper triangular area only to avoid repeating operations.



LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN
910215	1.675376	4.421928	15.291805	11.745392	13.402658	14.476013	15.089627	13.700717
910220	1.675376	9.000000	15.291805	11.745392	13.402658	14.476013	15.089627	21.000000
910225	1.675376	9.000000	17.000000	11.745392	13.402658	14.476013	15.089627	17.000000
910226	2.000000	1.000000	13.000000	11.745392	13.402658	14.476013	15.089627	13.000000
910241	1.675376	1.000000	20.000000	11.745392	13.402658	14.476013	15.089627	14.000000

5 rows × 283 columns

Figure 11: Azdias Dataset after Correlation Filtering



LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PERSONEN
9626	2.000000	1.0	10.0	10.0	1.0	0.000000	0.0	2.0
143872	1.573962	1.0	6.0	0.0	1.0	0.000000	0.0	1.0
143873	1.000000	1.0	8.0	8.0	0.0	0.067456	0.0	0.0
143874	1.573962	1.0	20.0	14.0	7.0	0.000000	0.0	4.0
143888	1.000000	1.0	11.0	10.0	1.0	0.000000	0.0	2.0

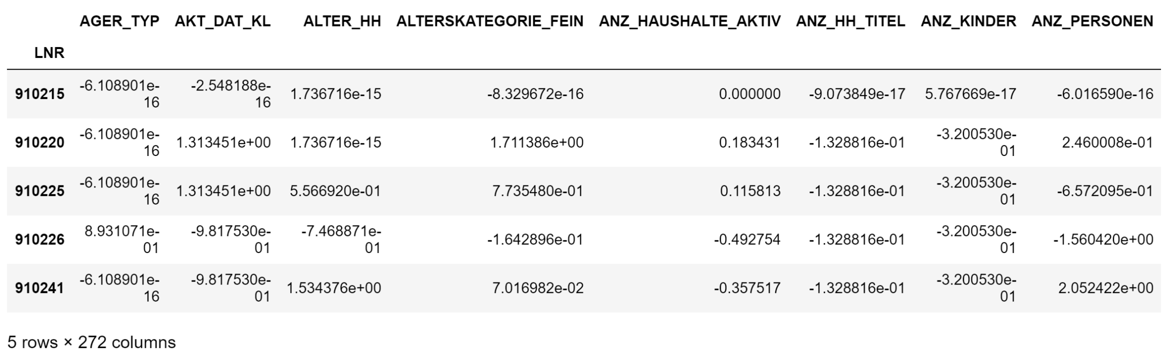
5 rows × 280 columns

Figure 12: Customers Dataset after Correlation Filtering

Figure 11 and 12 are azdias and customers datasets after correlation filtering, we can see that only 283 and 280 features are left, separately. This filtering reduces a lot more columns than earlier processing, compared to Figure 9.

3.2 Feature Normalization

Feature normalization is another standard procedure of clustering analysis and here I adopt mean-standard deviation normalization to both datasets. Figure 13 shows the result of azdias dataset, the values become a little peculiar, but statistically more synchronized as they now have mean and standard deviation of 0 and 1 across columns.



LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PERSONEN
910215	-6.108901e-16	-2.548188e-16	1.736716e-15	-8.329672e-16	0.000000	-9.073849e-17	5.767669e-17	-6.016590e-16
910220	-6.108901e-16	1.313451e+00	1.736716e-15	1.711386e+00	0.183431	-1.328816e-01	-3.200530e-01	2.460008e-01
910225	-6.108901e-16	1.313451e+00	5.566920e-01	7.735480e-01	0.115813	-1.328816e-01	-3.200530e-01	-6.572095e-01
910226	8.931071e-01	-9.817530e-01	-7.468871e-01	-1.642896e-01	-0.492754	-1.328816e-01	-3.200530e-01	-1.560420e+00
910241	-6.108901e-16	-9.817530e-01	1.534376e+00	7.016982e-02	-0.357517	-1.328816e-01	-3.200530e-01	2.052422e+00

5 rows × 272 columns

Figure 13: Azdias Dataset after Normalization

3.3 PCA Analysis

The next step is Principle Component Analysis, or PCA. The *sklearn library* also has a nicely encapsulated class for this. I first fit the azdias with PCA alogirthm and plot the cumulative variance curve. The result is given in Figure 14.

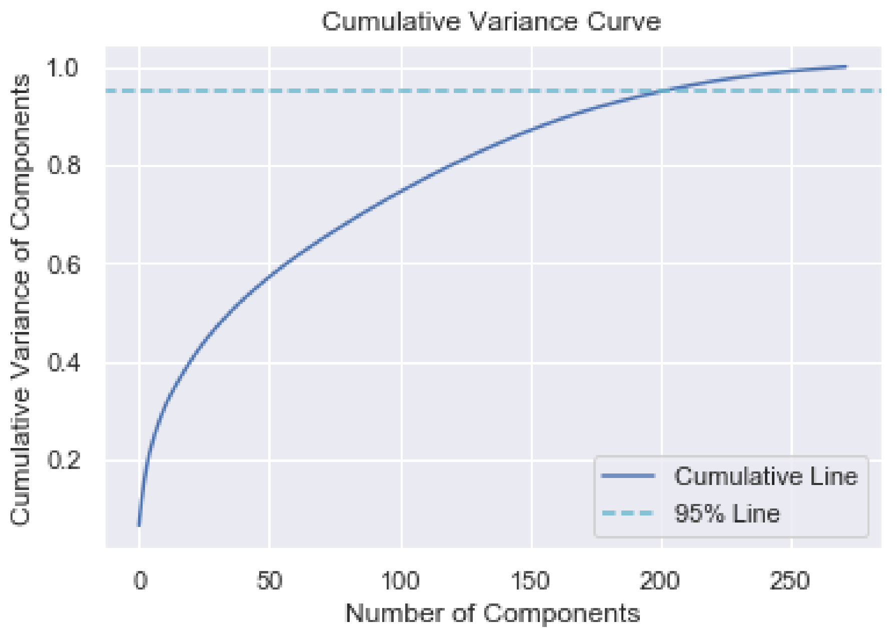
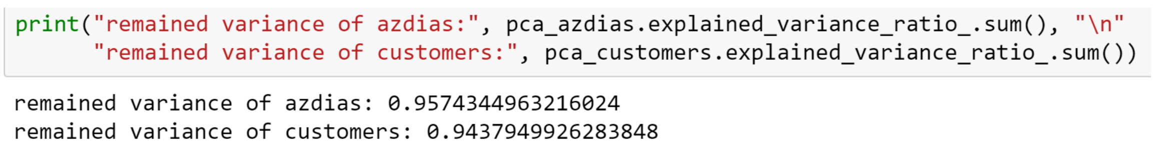


Figure 14: The Curve of Cumulative Explained Variance of Azdias Dataset

In Figure 14 I also plot a 95 percent line, from which we could see that approximately 200 components will make up for 95 percent of the variance. Now fit both datasets with componets number of 200 and calculate the remaining variance in Figure 15. Not far from the rough estimation, nearly 95 percent of the variances are kept for both datasets.



```
print("remained variance of azdias:", pca_azdias.explained_variance_ratio_.sum(), "\n"
      "remained variance of customers:", pca_customers.explained_variance_ratio_.sum())
```

```
remained variance of azdias: 0.9574344963216024
remained variance of customers: 0.9437949926283848
```

Figure 15: The Remaining Variance of Two Datasets

3.4 K-Means Clustering

Now it's time to perform the final K-Means clustering. To determine the number of clusters to use, first plot a curve of different numbers of clusters on the pca_azdias datasets (azdais after PCA fitting) to corresponding relative distance scores. Figure 16 gives the result.

Though there's not an explcit "elbow" pattern in Figure 16, we can see at cluster number of 13, the trend of the curve changed. At numbers less than 13, the slope declines gradually but after 13 the slope seems to decline at a larger rate. This is quite similar to the "elbow" pattern so I choose cluster number of 13 as the final number of clusters.

Fit both after-PCA-fitting datasets with K-Means model of cluster number 13 to form a distribution of 13 clusters, the result is shown in Figure 17.

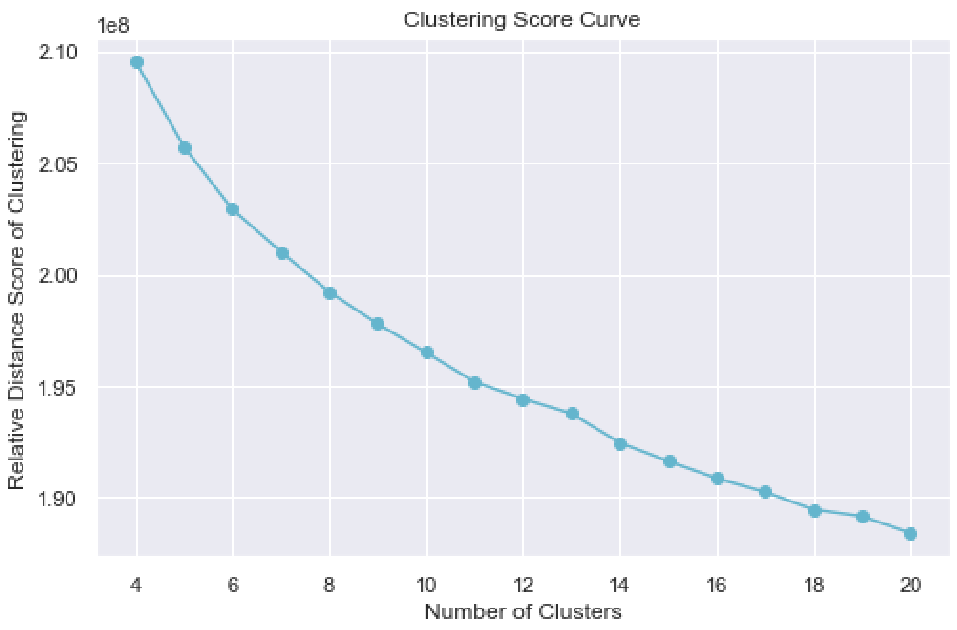


Figure 16: The Score Curve of Different Clusters of AZDIAS Dataset

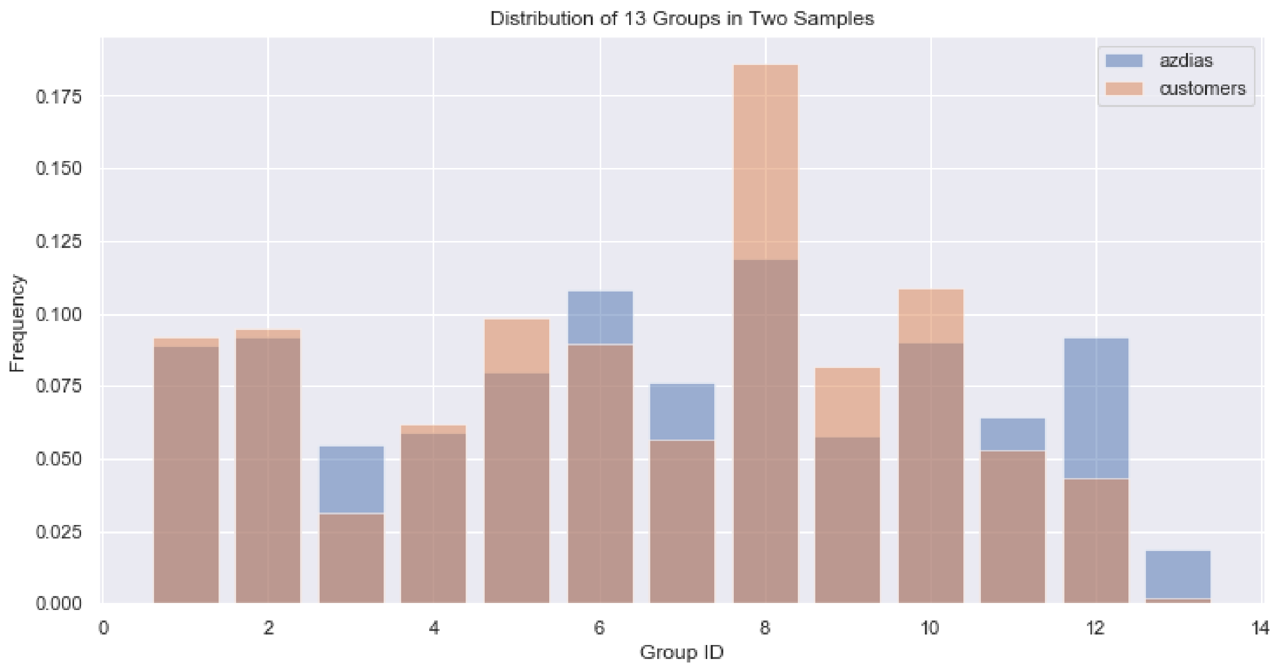


Figure 17: Distribution of 13 Groups in Two Samples

Among all clusters, the distributions of azdias and customers have differences and also similarities. In general azdias has a more "even" distribution of all clusters except for the last cluster, and customers clearly has extreme values. The similarity is that both have very few ratios of the last cluster. Maybe the last type of people is just especially rare in the general population.

The conclusion can now be drawn from the difference of two distributions: type 8, 9 and 10 have the most potential to become customers and type 3, 7, 12 and 13 are less likely to be.

4 The Supervised Learning Model

Now that the conclusion regarding which parts of the population are more likely to be customers of the mail-order company is made, it's time to build a prediction 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, each with almost 43 000 data rows. I will select and optimizing the model 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. After that, predictions will be created based on the "TEST" partition, where the "RESPONSE" column has been withheld. And the result shall be evaluated by Kaggle to get the exact performance on the test set.

• **Evaluation Metric** The Kaggle competition uses Area Under Curve as the score of a model. AUC will be applied here to be compared to Kaggle's ranked scores. Another reason to use AUC is that there's class imbalance in the dataset, as records with responses are a lot less than records without responses. Using classification accuracy would not be a good idea.

• **Benchmark Model** Logistic Regression will be the benchmark model here as it's basically the simplest probability regression model and the final model should have a better performance than the simple one.

4.1 Model Selection

Figure 18 plots the ROC Curve of four candidate models: Logistic Regression, Random Forest, Gradient Boosting and XGBoost. All four models uses default parameters to compare the general performance, except for *class weights* used in Logistic Regression and *objective* in XGBoost.

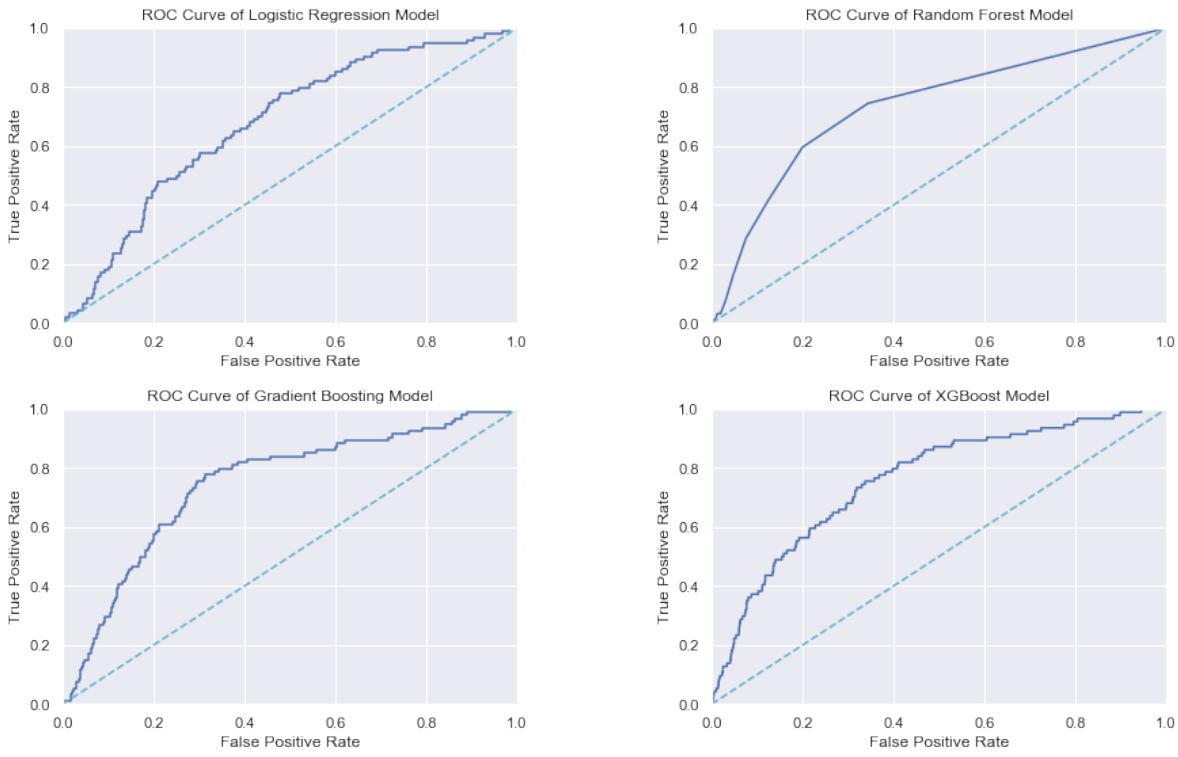


Figure 18: ROC Curves of Different Basic Models

Table 1 presents the AUC-ROC scores for the above four pictures, and XGBoost seems to outperform others so it will be the model I use to make predictions. To make results repeatable and comparable, I set the same random states for train-validation set split.

Models	Logistic Regression	Random Forest	Gradient Boosting	XGBoost
ROC Score	0.678	0.734	0.747	0.760

Table 1: ROC Scores of Different Basic Models

4.2 Hyper-Parameter Tuning

For hyper-parameter tuning, however, I shall use a different random state to avoid overfitting. Random states are set to make repeatable results, but using the same random seed through model selection and hyper-parameter tuning may cause overfitting, as all performances are evaluated on the same validation set. Using different random states for different stages could reduce overfitting and make results repeatable as well.

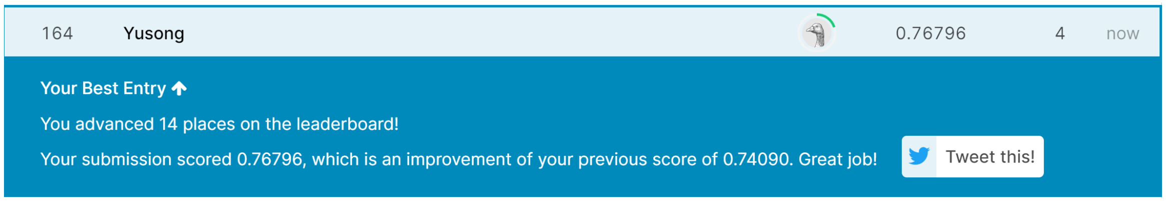
The first grid search covers 7 parameters: *colsample_bytree, gamma, learning_rate", max_depth, min_child_weight, n_estimators, reg_lambda*. And the second search covers four parameters: *gamma, eta, colsample_bynode, reg_alpha*. I split the search to two stages for two reasons: one is running time efficiency, searching more than 10 parameters can cost a lot of time, even they only have two candidate values, respectively; another reason is again overftting, as two searches with same validation sets tend to have worse performance improvement than with different ones. The resulting scores are summarized in the following table.

Stage	Basic Model	First Search	Second Search
ROC Score	0.760	0.763	0.789

Table 2: ROC Scores of Different Tuning Stages

5 Results and Conclusion

Using the final model-XGBoost with two searches of hyper-paramters tuning, the prediction on test set obtain a score of 0.76796 on Kaggle. The result is worse than the final score obtained from training, indicating the potential overfitting. But depending on the several submits with models from different stages, the final score is always increasing, which means the hyper-parameter tuning works.



164 Yusong 0.76796 4 now

Your Best Entry

You advanced 14 places on the leaderboard!

Your submission scored 0.76796, which is an improvement of your previous score of 0.74090. Great job!

Figure 19: Kaggle Rank Record

To conclude, the result is kind of mediocre compared to the leading scores, but better than the benchmark model of Logistic Regression. During the process of model selection and hyper-parameter tuning, the overfitting issue is addressed and the final model has a better performance on the test set than basic model on the training set. I believe All steps are performed as scheduled.