Capstone Report: Bertelsmann/Arvato Project, A segmentation approach

John Zhao

yzh2013@yahoo.com

Introduction

This is the report for Udacity Machine Learning Engineer Capstone Projects. We got our data from AZ Direct GMBH. The end goal is to help them identify new customers for a mail-order company based on existing customer data and whole population of the Germany. The idea is to train an unsupervised learning model to identify the similarity and difference between the general population and the mail-in company's customers based on geographical features provided. And then build a supervised learning model to identify potential new customers with the training data.

For this project I used Principle Component Analysis for feature deduction, KMeans for clustering and Xgboost for classification. I will talk about the details about Preprocessing and EDA, model selection and training, hyperparameter tuning in the following part.

Keywords: Customer Segmentation, PCA, KMeans, Unsupervised Learning

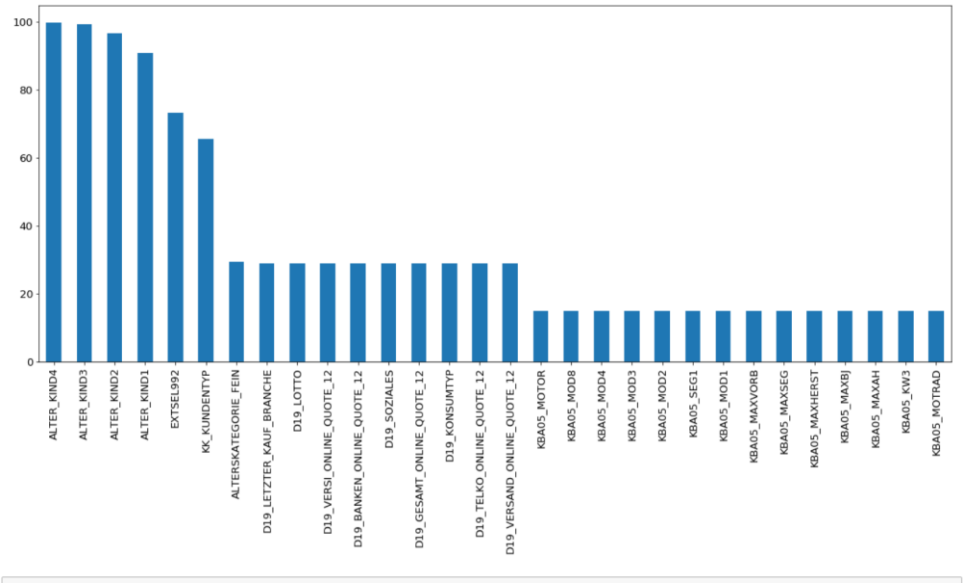
1. Preprocessing and Exploratory Data Analysis

The data provided by AZ Direct GMBH is really complex and messy with a lot of unknown values. Before model training, I will need to clean the data and process some analysis on the raw data for some insights of the project.

1.1 Dealing with NA

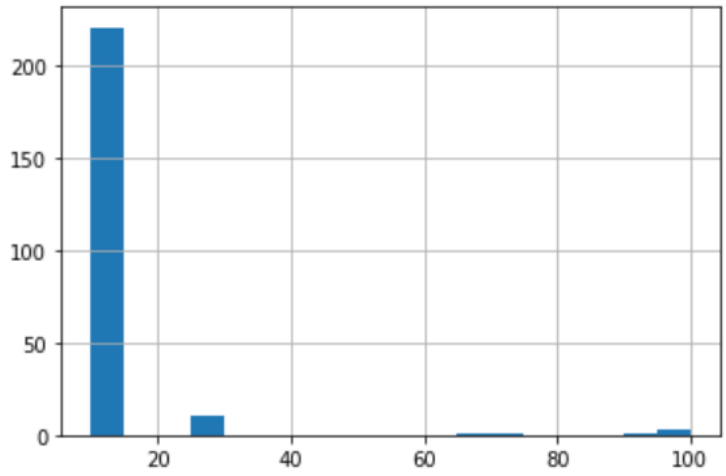
As mentioned earlier, we got the real customer data from AZ Direct GMBH. The data is very messy. With a lot of NA values.

I calculated the missing data's percentage of all the features. The result is as below.



Most of the data is below 30% missing, so we will choose to keep them. And only delete the top 6 columns that are missing.

Another plot to show the histogram of unknown values shows that most of the features are missing lest then 30% values.



There is another problem when reading the data into python.

‘DtypeWarning: Columns (18,19) have mixed types. Specify dtype option on import or set low_memory=False.

interactivity=interactivity, compiler=compiler, result=result)’

I print out the columns values to check what’s wrong. The features is an object type, with a mixed of string, integer and floats. There are ‘x’ and ‘xx’ in the columns as well, I interpret them as unknown class to impute them as -1.

For other NA values, I refer to the 'DIAS Attributes - Values 2017.xlsx', where there are definitions of unknown values. I build a dictionary called unknown_dic and map those values to numerical values.

For features that have multiple values for unknowns, I refer to the original file see which value exists in the columns.

After this impute, there are still unknown values, since not all the values show in the original file. In this case, I used sklean's Imputer method and use the mode the impute the NAs.

1.2 Categorical Data

Now it is time to move to Categorical Data. Those are the features with type as object. There is another special case here. The column EINGEFUEGT_AM is time stamp saved as string. The way I transferred it is to get the minimum value and calculate the difference between every row.

After that transfer, I used one hot encoding to change categorical data to numerical data.

I could improve performance with binary encoding as the feature size is quite large with the training data. This is something for further analysis.

1.3 Multicollinearity Problem

I am trying to calculate VIF score to eliminate features correlated with other features. It took a long time to finish and there are memory issues with my laptop.

With some more research, I realized this will not be a problem for the end result. I could affect the confident interval but the dependent variable will not be affected.

1.4 Scaling features

Since I am using PCA method to reduce dimensionality, I need standardize or normalize the features. Otherwise it will not have the same level of variance to represent the importance.

Plus, it's also good practice to Xgboost, RF models and maybe a linear model as bench mark. At minimum, it will increase the computation speed.

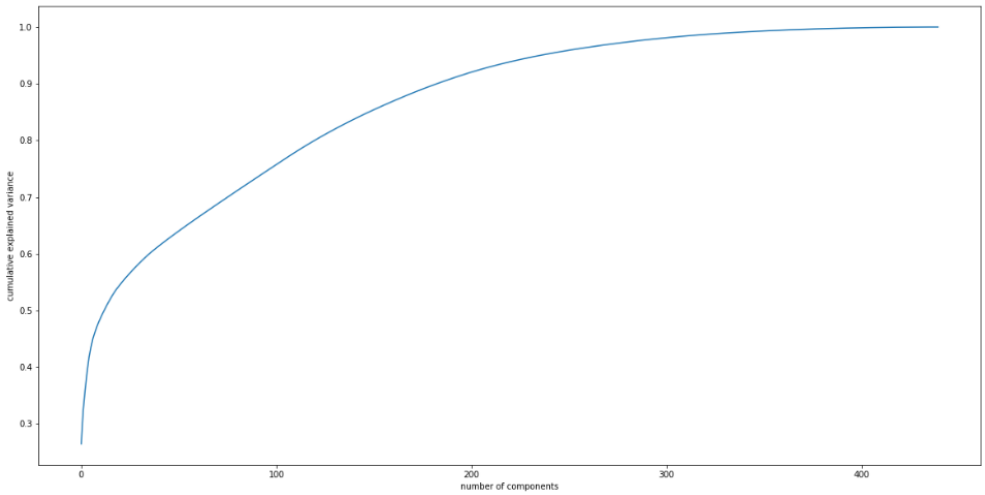
2. Dimension Reduction

In this part, I used PCA to reduce the dimension and prepare data for KMeans analysis.

Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance

2.1 Deciding the number of components

I used same method as instructed during the class. I used sklearn's PCA model and plot the number of components against the proportion of the explained variance to decide the best value for the number of components.



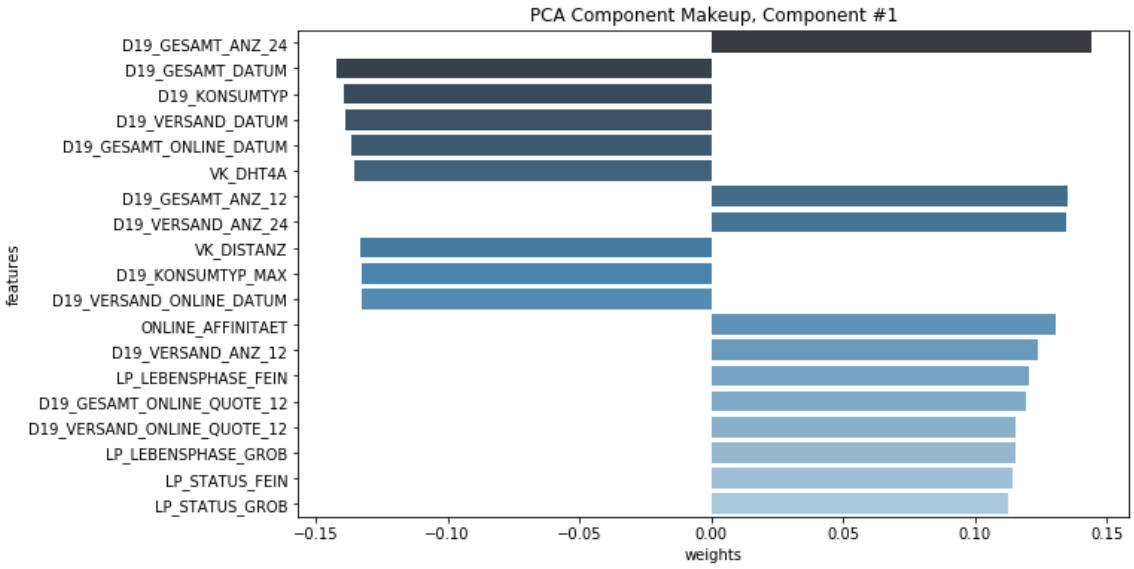
The curve flattens around 250. I end up using 220 as the n. To confirm I have enough variance, I check ethe result.

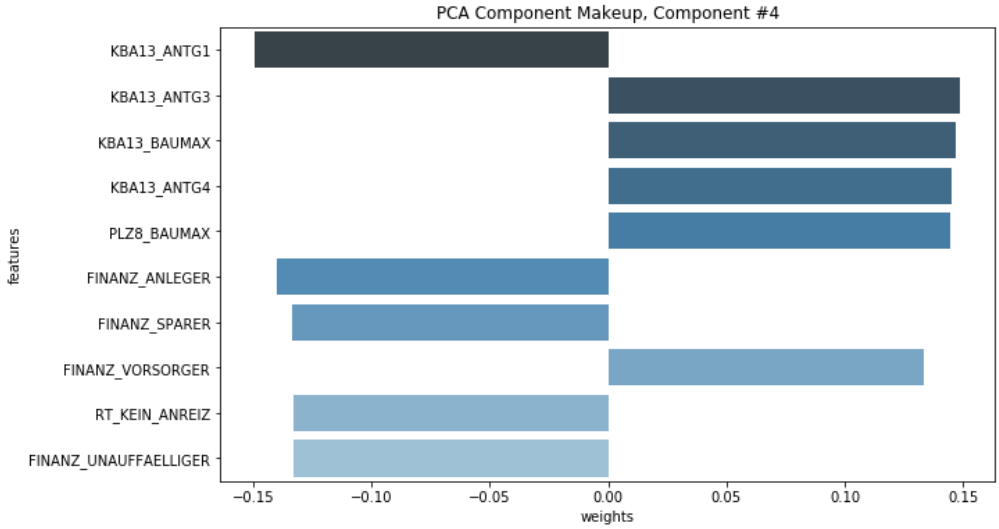
```
reduced_azdias = reduce_data(azdias_scaled)
reduced_customers = reduce_data(customers_scaled)
```

```
0.9374140915725537
0.9567021895261978
```

2.2 Visualizing the principle components

The eigen vector shows how the original features are constructed to build ne new dimension. I built a model to visualize the result. Below shows the first and fourth component.





You can see that the main factors are the financial transaction records and what kind of automobiles th person owns.

3. Unsupervised Learning

Now we are finally ready to train out unsupervised model, a KMeans model.

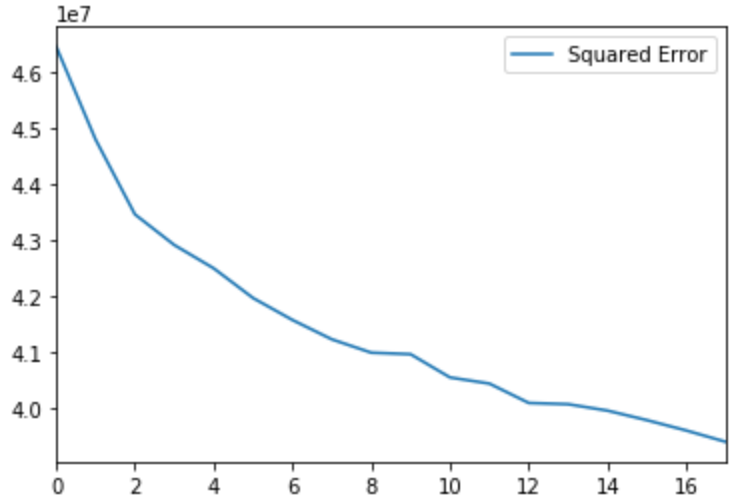
k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.

3.1 Deciding the K

Similar to Principle Components Analysis, one hyper parameter I have to provide is the K.

Again, I followed the procedure from class the plot the metric against. Here I used the squared error as my metric to measure the model performance.

The result is as below.

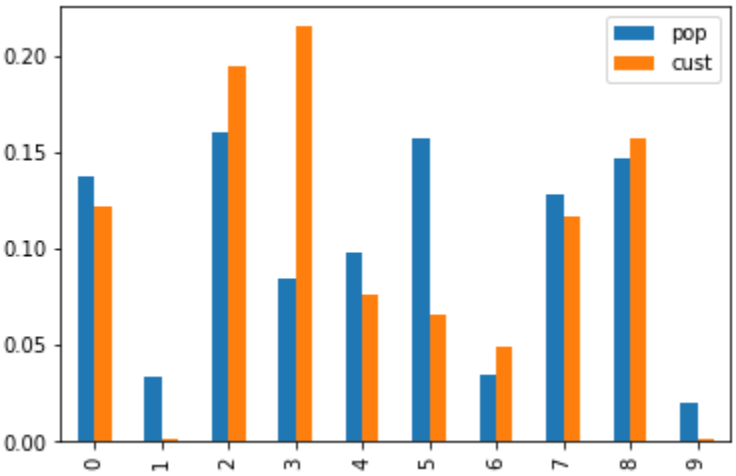


The flatten point seem to be at 9, which means we will need 10 clusters for a good result.

3.2 Model fitting and prediction

I used sklean's Kmeans method to train the model and make predictions. I use the same techniques to preprocess the customer data and make predictions.

I then calculate the percentage of customers and populations from each category.



We can see that while many clusters share similar percentage, cluster 3 has a lot more percentage than general public. This is a significant sign that the main-in company should focus on to bring in more new customers.

They should focus on marketing effort the cluster 3 while limit resources on cluster 1 and 9.

4. Supervised Learning Model Selection

Part 2 is building a supervise learning to forecast the probability of a person to become a customer of the mail-in company.

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

XGBoost is an open-source software library which provides a gradient boosting framework for C++, Java, Python, R, Julia, Perl, and Scala. It works on Linux, Windows, and macOS. From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting Library".

4.1 Defining the metrics

After preprocessing data the same way for clustering, I need to define a metric for supervised learning. After checking the label of the problem, I realized we are dealing with a highly imbalanced training set.

```
y.value_counts()
```

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

So regular metrics like accuracy is not a good metric for this problem. I end up using AUC-ROC score as my metrics.

A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The method was developed for operators of military radar receivers, which is why it is so named.

For high imbalanced labels, there are few methods to use. There two types of techniques, one is to up-sample the fewer class, the other is to down-sample the more class.

I could try SMOTE method to create more synthetic data in the future to improve the model performance.

4.2 Benchmark Selection

Historically, financial company will use heuristic analytical techniques to analysis certain customers and make product promotion. It is a combination of art and science.
With the modern compute power increase, we can now use machine learning to process data and use statistical method to build machine learning models to make inference, whether for classification or regression.

In this project, there are not much benchmark models to compare with. If available, I might use the historical data from the company to compare my result with.
Alternatively, I am thinking to build some simple models with default parameters and compare with a fine-tuned and more complexed models to see the difference and improvement.

After previous section of analysis and data exploration, I do not think linear model is a good choice for this problem, event as a baseline, as there is no linearity between dependent and independent variables.

As a result, I am using a simple random forest as my baseline.

4.3 Model Selection

I used 5 folds cross validation for model selections. The model I used is from sklearn StratifiedKFold method. The random forest is also from sklearn. Gradient boosting model is from xgboost.

The validation result is as below.

```
Fold 1...
{'model_name': 'RandomForestClassifier', 'fold': 1, 'roc': 0.4995286353994815}
Fold 2...
{'model_name': 'RandomForestClassifier', 'fold': 2, 'roc': 0.4993518736742871}
Fold 3...
{'model_name': 'RandomForestClassifier', 'fold': 3, 'roc': 0.4985859061984445}
Fold 4...
{'model_name': 'RandomForestClassifier', 'fold': 4, 'roc': 0.4992929530992222}
Fold 5...
{'model_name': 'RandomForestClassifier', 'fold': 5, 'roc': 0.4993518736742871}
Fold 1...
{'model_name': 'XGBRegressor', 'fold': 1, 'roc': 0.5380291012574863}
Fold 2...
{'model_name': 'XGBRegressor', 'fold': 2, 'roc': 0.6232552351206275}
Fold 3...
{'model_name': 'XGBRegressor', 'fold': 3, 'roc': 0.5818429021829519}
Fold 4...
{'model_name': 'XGBRegressor', 'fold': 4, 'roc': 0.5707936267948541}
Fold 5...
{'model_name': 'XGBRegressor', 'fold': 5, 'roc': 0.5925492153558136}
result_list:  [{'model_name': 'RandomForestClassifier', 'fold': 1, 'roc': 0.4995286353994815}, {'model_name': 'RandomForestC
lassifier', 'fold': 2, 'roc': 0.4993518736742871}, {'model_name': 'RandomForestClassifier', 'fold': 3, 'roc': 0.498585906198
4445}, {'model_name': 'RandomForestClassifier', 'fold': 4, 'roc': 0.4992929530992222}, {'model_name': 'RandomForestClassifie
r', 'fold': 5, 'roc': 0.4993518736742871}, {'model_name': 'XGBRegressor', 'fold': 1, 'roc': 0.5380291012574863}, {'model_nam
e': 'XGBRegressor', 'fold': 2, 'roc': 0.6232552351206275}, {'model_name': 'XGBRegressor', 'fold': 3, 'roc': 0.58184290218295
19}, {'model_name': 'XGBRegressor', 'fold': 4, 'roc': 0.5707936267948541}, {'model_name': 'XGBRegressor', 'fold': 5, 'roc':
0.5925492153558136}]
```

Xgboosting has an average of 10% improvement than the random forest model.

5. Hyper Parameter Tuning

There are quite a few hyper parameters from xgboost models. Hyper parameters are the parameters you sent before model train and won't be changed or updated as training process goes on. A wrong set of hyper parameter could cause high bias or high variance.

The parameter I chose are below.

```
parameters = {'max_depth': [3, 5, 7],
              'learning_rate':[0.01, 0.1],
              'min_child_weight': [1,3],
              'colsample_bytree':[0.5, 0.7],
              'reg_alpha':[0.01, 0.05],
              'n_estimators': [500,600],
              'objective' : ['binary:logistic']
             }
```

I used grid search from sklearn model_selection to train the parameters. Behind the scene, it's similar idea to Kfolds validation. The model use a combination of the hyper parameters and running recursively to find the best metric. Here I used AUC-ROC score.

The best model could be retrieved with the feature best_params.

The beset parameters are shown as below

```
{'colsample_bytree': 0.7,
 'learning_rate': 0.01,
 'max_depth': 5,
 'min_child_weight': 3,
 'n_estimators': 600,
 'objective': 'binary:logistic',
 'reg_alpha': 0.01}
```

Since I am using GridSearCV method, the training and validation happens the same time. I could try to use all the training data to train the model again with the hyper parameter to improve the model with 20% more data.

My AUC-ROC score for training set is about 76% and for validation set is about 77% which is higher than a naive random forest model, which has about 50%.

6. Submission file

Part 3 and the last part of the capstone is to predict a testing data.

I went through the same procedure to preprocess the data and make predictions.

My final submission file is like below.

```
final_submit.head()
```

	Response
LNR	
1754	0.037558
1770	0.030495
1465	0.003967
1470	0.004608
1478	0.008679

Conclusions

In this capstone project, I processed raw geographic data and make exploratory analysis on different features and impute null values with meaningful value. I then implement PCA to reduce feature

dimension and explore different number of components to balance training time and variance explanation.

The result of PCA fits to KMeans model for unsupervised learning. I discovered certain clusters have higher percentage of people than general population. This is a great found for the client to work on for better marketing practice.

In the next part, I trained several supervised machine learning models to make predictions on whether a person would become a customer. The label has an extremely high imbalance problem. I end up choosing AUC-ROC score as the metric and xgboost as the final model to make prediction. My model metric for validation dataset is about 80%.

I really enjoyed working on this project. One thing I would recommend is to move the project to the cloud so that we can really use the techniques from AWS class for better modeling and compute power.

I encounter a lot of problem with AWS, include quote issue and memory issue. This really frustrate me in the beginning.

Thank you for the help from mentors, reviewers and classmates. This really has been fun.

Reference

https://royalsocietypublishing.org/

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

https://matplotlib.org/

https://xgboost.readthedocs.io/en/latest/parameter.html