Final Capstone Report for Udacity Machine Learning Engineering Nanodegree
Chetan Raj Rupakheti

Introduction

Marketing is a critical aspect of any new or existing business. It can potentially help in identifying new niche as well as expand business. Customer interest for a product can depend upon various factors including their socioeconomic condition as well as geographic location. With a huge investment and profit at stakes, it is imperative for a business to identify their potential market as precisely as possible. In fact, business intelligence is a huge sector where business try to make better use of data that they collect to formulate decision support system that enhances their profit and growth[1, 2, 3, 4]. This competition dealt with modeling the dataset graciously provided by the Arvato company.

The first part of the project was to use the unlabeled dataset to perform population segmentation where we can identify demographic features that defines the customers of the Arvato company. The second part of the project was to use the labelled dataset and build models that predict potential customers of the company. Finally, the predicted label on the test data was submitted in the Kaggle competition.

Datasets and Inputs

The following datasets provided by the Arvato company containing their customer and demographic data were used.

- 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).

Evaluation Metrics

The model evaluation was done using accuracy and the area under the curve (AUC) metrics

$$accuracy(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(\hat{y_i} = y_i)$$

The accuracy is average of the correct predictions is obtained between the truth (y_i) and the prediction ($\hat{y_i}$).

For the binary class classification, the AUC is defined as the average of the sensitivity and specificity:

$$AUC = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right)$$

TP, TN, FN, and FP are the true positive, true negative, false negative, and the false positive respectively.

Methods and Results

Part 1. Data Cleaning for Unsupervised Learning using Udacity_AZDIAS_052018.csv

The demographic data contained 891 211 persons (rows) x 366 features (columns). It was a challenging dataset to model. The data had a mix of categorical features containing two or more categories. OST_WEST_KZ feature could had a “O” and “W” categories. It could be easily be transformed using binary encoding. The other features had more than binary categories presenting a challenge for proper feature engineering. Another challenging aspect of the dataset was that lots of rows had missing values. Any columns that contains >30% NaN were removed. The ‘LNR’ column that contains the ID for each row was also removed from any kind of modeling. Some columns that were categorical such as ‘CAMEO_DEU_2015’ and 'CAMEO_DEUG_2015' were one-hot encoded. The column 'OST_WEST_KZ' was converted into binary with 1 for ‘W’ and 0 for ‘O’. Since there were 363 column containing integer label for category, it quickly became infeasible to perform one-hot encoding. The memory and storage requirement quickly grow if one hot encoding is chosen for such a large dataset. Therefore, they were kept as provided integer encoded value. The feature were a mix of binary, integer, and some one-hot encoding. After the feature engineering step, pair-wise feature correlation was computed. One of the features in a pair that correlate by more than 80% was removed. Features that were not described on “DIAS Attributes - Values 2017” were also removed because it was difficult to know what they meant. After these cleaning steps, the NaNs were imputed using the most frequent value. The reason for using the most frequent value instead of mean was because most of the features were categorical. After the features were imputed, the dataset was scaled using both minmax scaler of sklearn that converts the column in a 0 to 1 range. I also tried Standard Scaler in sklearn to convert the data into *N(0,1)* distribution since it is recommended for Linear Discriminant Analysis (LDA) algorithm.

Part 2. Model Generation during Unsupervised Learning

After the feature cleaning step, 293 features remained and used for clustering. These features were used for minibatch KMeans clustering.

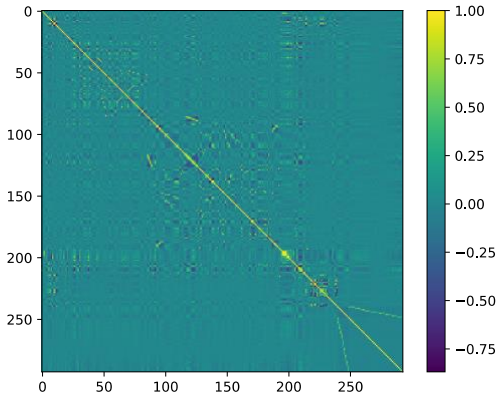

Figure 1: Lots of off-diagonal features had high Pearson correlation >0.8 as seen in the figure. Only one of the features in such a pair were kept for further modeling.

Minibatch KMeans clustering, available in Sciket-Learn library, was performed using these reduced non-redundant features. Since the data is ~900,000 records and high dimensional, straight up KMeans becomes unsuitable because of the intense memory requirement. Minibatch KMeans has a nice feature that only a subset of the full data is used in a training iteration[5]. This algorithm iteratively draws random sample at a time and updates the cluster centroids from the previous iteration. A batch size of 20,000 was used for clustering. The model selection was done using the elbow method that tries to estimate the number of clusters to balance the model complexity and generalizability. As shows in figure 2, there were a few good solutions to this data set. A cluster size of 20 seemed to be one of a good choice for this data since the sum of squared errors (sse) decays slowly upon increasing the number of clusters.

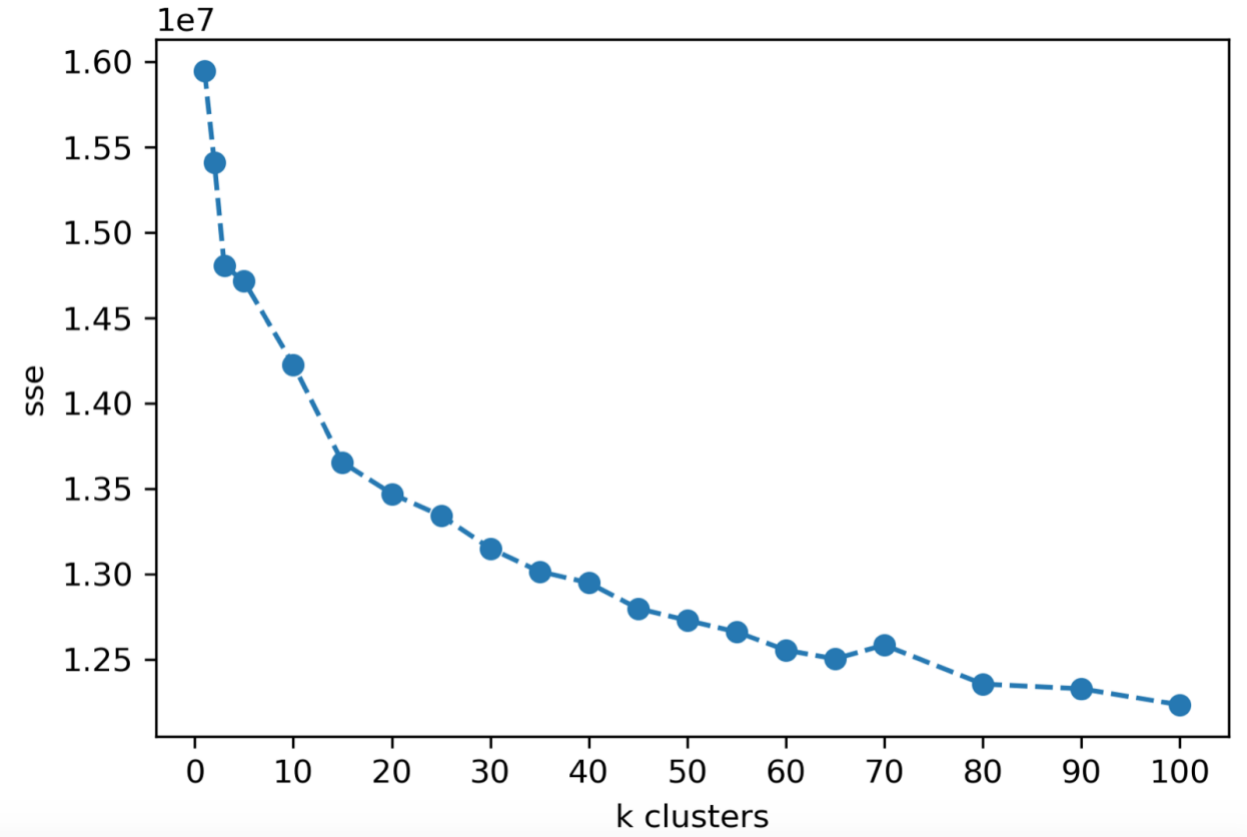

Figure 2: Sum of squared error (sse) starts to saturate after around cluster size of 20. Another clear 'elbow' does occur around 65 clusters. A less complex model of 20 clusters was chosen.

Density of each cluster was also analyzed by projecting the population data on the 20 clusters using the trained model. Clearly, some of the clusters were highly populated compared to others as shows in figure 3.

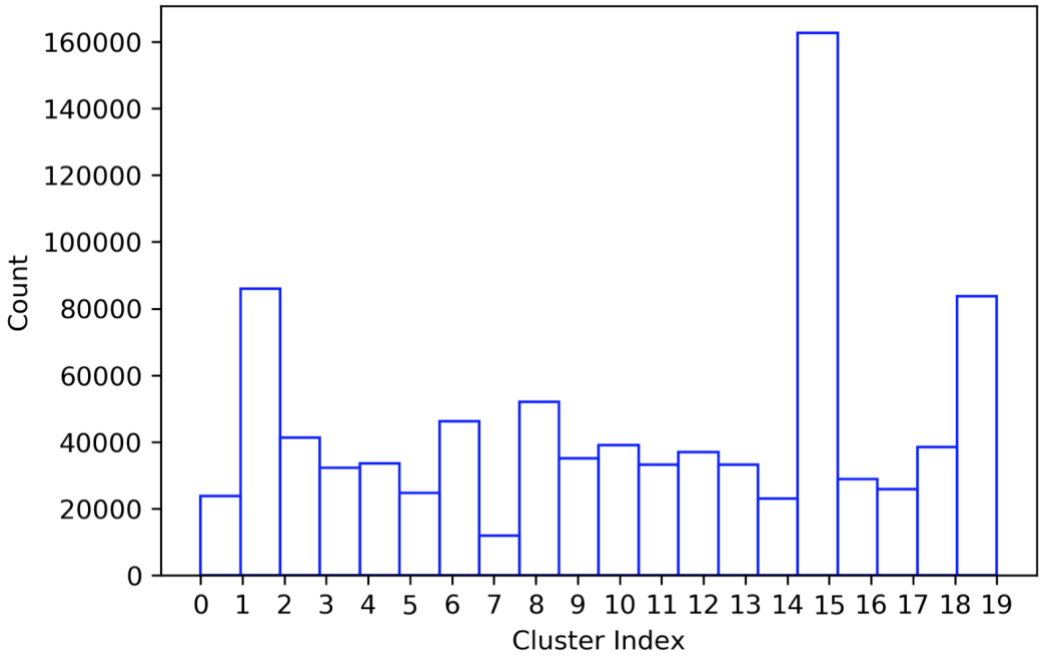

Figure 3: The population dataset was mapped to the clusters. Cluster 15 seem to be a favored one. Membership count for each cluster is shown.

The customer dataset was cleaned and transformed similarly to the population dataset. It was then mapped to the same number of clusters using the obtained model to see which demographic population cluster the customer tends to fall. Interestingly, some population cluster was highly favored compared to other as seen in figure 4.

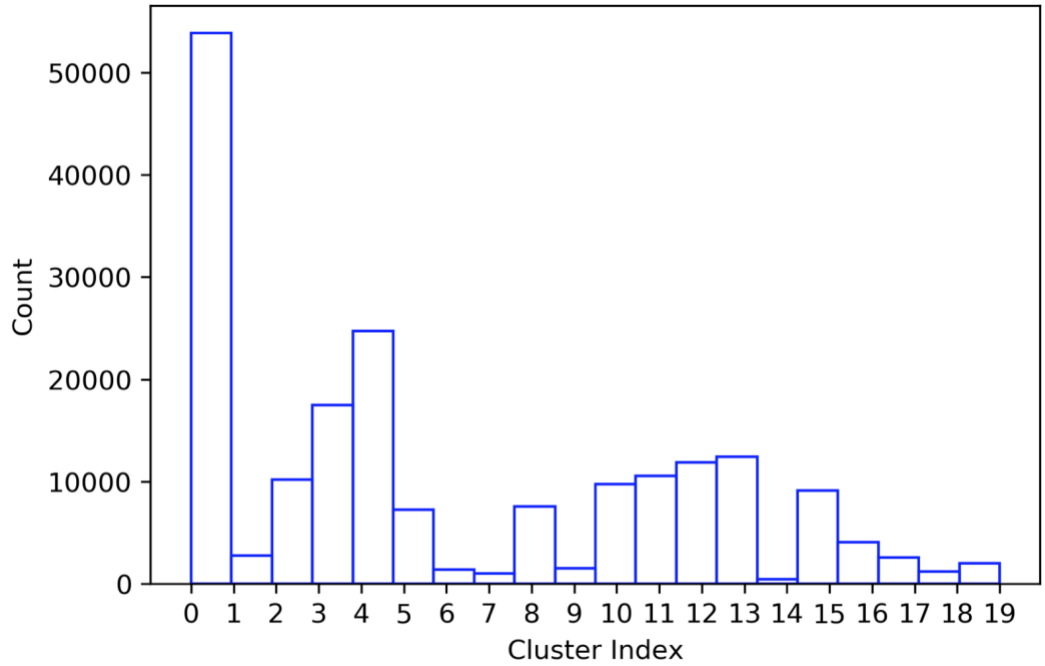

Figure 4: Projection of customer data on to the population clusters. Membership count for each cluster is shown.

Interestingly the above cluster ‘0’ also represents population with high-income category as shown in figure 5. In contrast, cluster 14, which is sparsely populated in the customer data also

fall largely in the low-income category as seen in figure 6. The income category is indicated by the 'HH_EINKOMMEN_SCORE'.

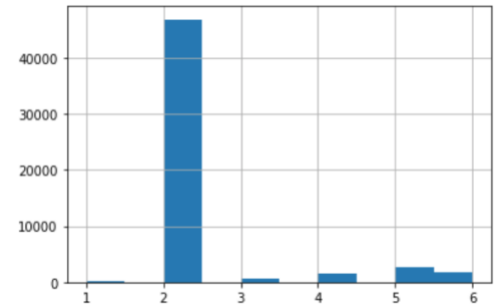

Figure 5: Cluster 0 constitutes demography with high income.

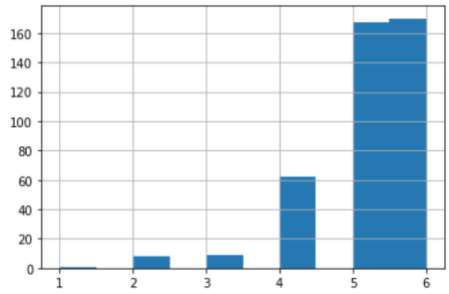

Figure 6: Cluster 14 constitutes of demography with low income.

While the clustering is able to pick out population segment relevant to being customers, there were other criteria such as willingness to buy house and gourmet types where not much difference between the two clusters were noticed. It probably points to the limitation of current feature engineering and selection procedure.

Part 3. Supervised Learning

The provided training data 'Udacity_MAILOUT_052018_TRAIN.csv' was used for this step. The provided dataset is very complex. Apart from lots of missing data, it is also highly imbalanced. Only about 1.2% of the total customer responded to the survey labelled as '1' in this dataset. Imbalanced data, missing data, and imprecise feature engineering makes this supervised learning highly challenging.

a. Linear discriminant analysis Model

As before, the dataset was cleaned, and features engineered using identical procedure resulting in 270 features. To counter the presence of overwhelming number of 'RESPONSE 0' samples, these examples were under sampled. Various ratio of 'RESPONSE 1' to 'RESPONSE 0' from 4:1, 2:1, 1:1, 1:2, and 1:4 were tried (i.e. the 'RESPONSE 1' samples are 4 , 2, 1, 0.5, and 0.25 times the 'RESPONSE 0' samples). For the validation set, 3 times more 'RESPONSE 0' data was sampled compared to 'RESPONSE 1' data to represent the imbalance seen in the actual dataset. Ten percent of the 'RESPONSE 1' data was kept for validation.

Linear discriminant analysis (LDA) algorithm and XGBoost algorithm were chosen for the supervised learning step. LDA is a simple algorithm that tries to project this high dimensional data into a two-dimensional space that maximizes the separation between two classes. Exact weights of the features can also be extracted from this algorithm. Because of its simplicity, it is an initial choice as a supervised classifier.

The best model from 10-fold cross validation was selected based on average AUC from the 10 folds. Figure 7 shows that LDA favors keeping half the amount of response 1 samples compared to response 0 samples since the peak of validation AUC occurs when 'RESPONSE 1' is half the size of 'RESPONSE 0'. One would think that 1:1 ratio is better. This discrepancy could be because the validation had 3 times higher 'RESPONSE 0' samples compared to 'RESPONSE 1' samples. Unfortunately, LDA seems to converge between training and validation set around a

low AUC score of 0.57. For this reason, XGBoost classifer was also tried hoping in increase the accuracy.

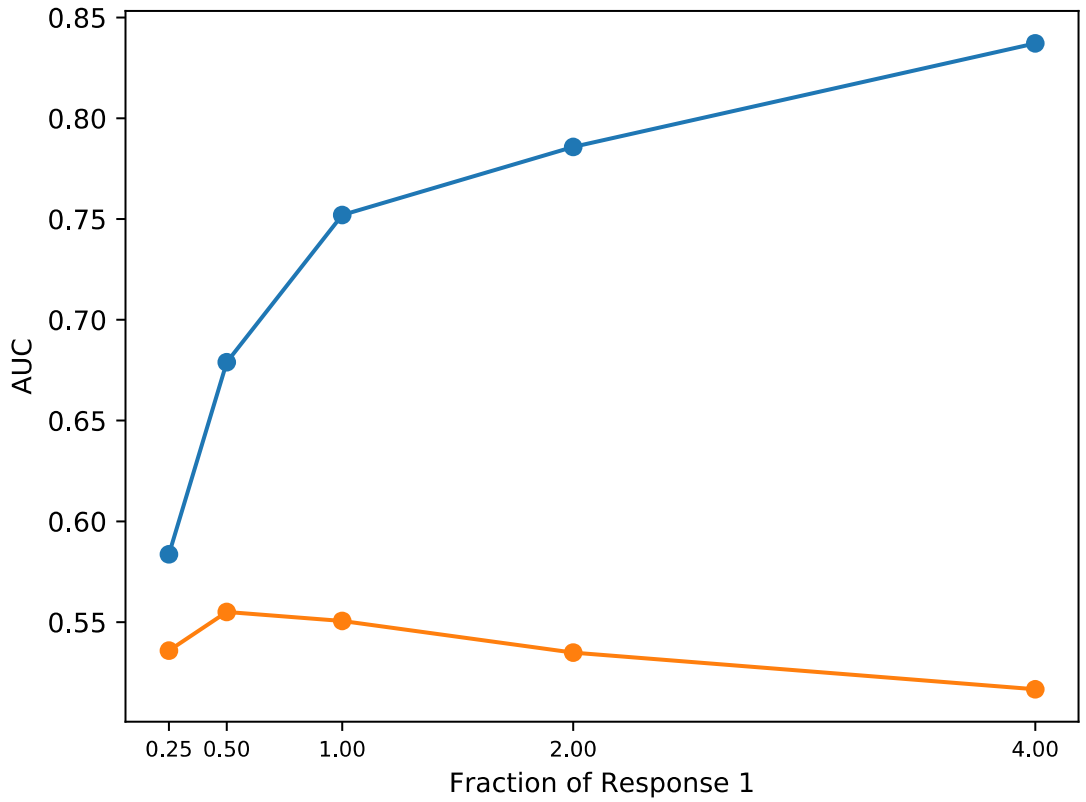

Figure 7: Comparison of training and validation set and various under sampling ratio of negative samples.

b. XGBoost Model

Because of the messy dataset, XGBoost algorithm was also tried to compare against the simple LDA algorithm. XGBoost applies ensemble of decision trees to learn complicated decision functions. The container of the XGBoost algorithm is available for a AWS Sagemaker instance. This container was used for model building purpose. Since the dataset is imbalanced, 'scale_pos_weight' parameter was treated as a hyper parameter and a simple grid search was done to find the suitable choice of this parameter as shown in figure 8. The usual recommended value of this parameter is $len(class\ 0)\ /\ len(class\ 1)$. This ratio comes around 90 for the training dataset. Values around this parameter was searched. As with the LDA modeling, ten percent of the 'RESPONSE 1' sample were kept for validation and 3 times that size was kept for 'RESPONSE 0' in the validation set. As with the LDA based analysis, increasing the weight of 'RESPONSE 1' sample improves the training score but not necessarily the validation score. Based on the best accuracy score on the validation set, a model was chosen to predict the test data 'Udacity_MAILOUT_052018_TEST.csv'.

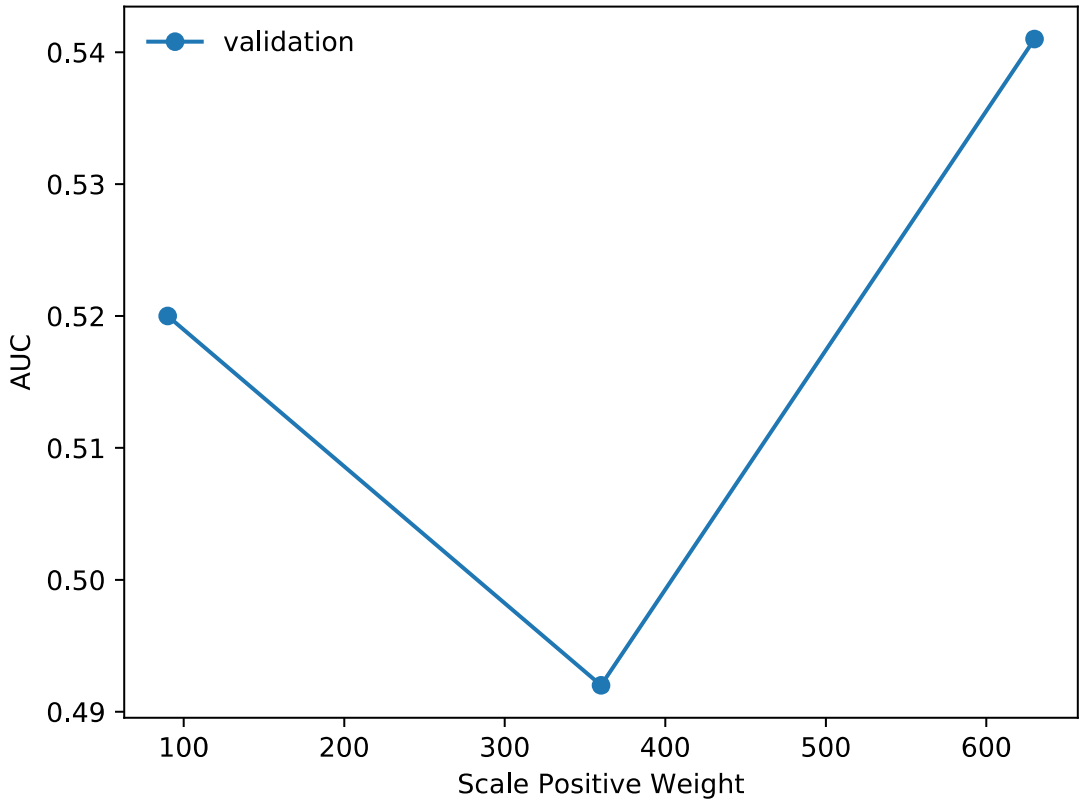

Figure 8: AUC score on the validation set varies with the weight assigned to response 1 sample. The recommended value of 90 appears to be a good choice during grid search.

The XGBoost model was chosen to predict the label of the test data. The test data was cleaned and transformed using the same procedure as described before. The deployed XGBoost model was used to predict the label in the AWS Sagemaker instance.

Part 4. Kaggle Competition

The predicted labels were converted in the required csv format containing 'LNR' and 'RESPONSE' columns. The predictions were submitted in the open competition. For this first try, a modest score of 0.51735 was obtained. Feature engineering and model building improvement will be applied to improve this score further in the next Kaggle submissions.

Part 5. Code and Notebooks

The code and notebooks are deposited and available in the github:
https://github.com/chetanrrk/Arvato_DataSet_Modeling

Discussion and Improvements

The main issue seems to come from the unbalanced dataset. In the subsequent modeling, under-sampling of the 'RESPONSE 0' data will be done and supplied to the XGBoost algorithm as in the case of LDA. The other avenue is to try multiple different algorithm and generate a consensus label for submission.

The other major issue is with the nature of datasets and features. Ideally, one-hot encoding has to be used for all of these categorical features (expect may be binary ones). This dataset has over 300 features making this task memory intensive. A better understanding of the feature and

perhaps discussion with the Arvato's team might lead to 'expert' selection of features to be one-hot encoded. At the very least, more careful analysis of the features will be done to improve the performance of the model. This is a work in progress and the competition open for another year. Improvement on the Kaggle score and ranking will be attempted leading to learning more techniques and building insight to model messy datasets.

Reference:

1. M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. Science, 349(6245):255–260, 2015. ISSN 0036-8075. doi: 10.1126/ science.aaa8415. URL http://science.sciencemag.org/content/349/6245/ 255.
2. S. Moro, P. Cortez, P. Rita. A data-driven approach to predict the success of bank telemarketing. Decision Support Systems, 62 (2014), pp. 22-31, 10.1016/j.dss.2014.03.001
3. David Arnott, Graham Pervan, Eight key issues for the decision support systems discipline, Decision Support Systems 44 (3) (2008) 657–672.
4. EfraimTurban, Ramesh Sharda, Dursun Delen, Decision Support and Business Intelligence Systems, 9th edition Pearson, 2011.
5. "Web Scale K-Means clustering" D. Sculley, Proceedings of the 19th international conference on World Wide (2010)