Capstone report for obtaining the Udacity Machine Learning Engineer Nanodegree

Definition of the problem

This capstone project deals with customers data coming from Arvato Financial Solutions, a Bertelsmann subsidiary. Arvato is a global services company including customer support, IT, logistics and finance.

The issue at hand is customer segmentation and classification i.e. being able to derive the traits of a customer of the specific company Arvato is providing services to compared to general population and then be able to state whether or not a person, based on his traits, will convert to a customer?

The following datasets were provided as four different files :

- A general population file containing 891 211 samples (persons) with 366 features for each sample;
- A customer file containing 191652 samples with 369 features i.e. 366 features are identical to the general population file while three additional ones are provided : CUSTOMER_GROUP, ONLINE_PURCHASE and PRODUCT_GROUP identifying the type of customer being described;
- A TRAIN mail campaign file containing 42982 samples and 367 features i.e. 366 features being identical to the general population file and one additional feature identifying whether or not the person became a customer following the campaign;
- A TEST mail campaign file containing 42833 samples being identical to the TRAIN file except that the target (became customer) has been removed.

Additionally, Excel files providing more information about the features of the customers and general population (levels, type of information, …) were also provided.

The problem at hand is therefore a supervised leaning problem. One needs to use the general population and customer files to try devising the features that distinguish a customer from a non-customer. Once these features are identified, one will build a model to infer if a given person will become a customer or the likelihood of this event for that particular person.

To solve this problem, the following techniques have been used :

- Dataset exploration using visualizing techniques (histogram, …);
- Feature-cleaning, engineering and cleaning as well as feature space reduction using Principal Component Analysis;
- Supervised model in the form of XGBoost.

For the supervised model, given the high imbalance of the dataset (only 2% of customers (target response = yes) among the data), using an accuracy scoring metric would be ill-advised. Therefore, one uses the Area Under Curve (AUC) metric. The objective is set to binary logistic given a binary classification is at hand.

Analysis

A specific notebook (Arvato Project Workbook_DataExploration.ipynb) is dedicated to data exploration and analysis. This notebook has been used to determine statistics such as :

- Amount of missing data for each feature and for each dataset,
- Amount of missing data among samples for each dataset

- Type of features and number of levels for categorical features

This Exploratory Data Analysis (EDA) allowed determining some threshold for dropping features containing too many missing data. Threshold was chosen at 60%. In a similar manner, a correlation threshold has been set to 70% for dropping highly-correlated features. Additionally, some features were chosen to be dropped manually because containing, in the analyst opinion, irrelevant or redundant data or containing too many levels for being relevant for the analysis. Eventually, categorical features were treated so that non-frequent categories were binned together.

Also, this EDA allowed analyzing the similarity between customers and population datasets in order to decide whether or not a common cleaning pipeline could be used.

Next, the EDA also identified that missing values could be replaced using the information provided in the Excel files giving information about features and categorical levels.

Eventually, the EDA allowed splitting numerical and categorical features. Indeed, preprocessing operations are different between those two types of features regarding e.g. scaling and imputing.

Methodology

Based on the EDA performed, cleaning steps have been defined. To perform these cleaning steps in an identical manner on each dataset, a scikit-learn customer pipeline has been defined in utils/custom_transformers.py using functions defined in utils/helper.py. Note that custom_transformers.py does not entirely answer scikit requirements e.g. return data is a pandas.DataFrame and not a numpy.ndarray. The pipeline can then be fit on the general population dataset and applied (.transform) on any dataset in a completely similar manner.

Once the cleaning and preprocessing steps (including normalization and one-hot encoding[1]) are completed, the feature space reduction PCA technique has been applied using AWS PCA container and storing the model on AWS S3 for future usage. This allowed reducing the features from 300+ to 100+.

With this reduced feature space, clustering could be applied using K-MEANS container again from AWS. A number of 10 cluster was chosen. Applying the K-MEANS algorithm on both the customers and general population datasets allowed identifying that customers are more represented in two particular clusters and analyzing the features that stand out in those clusters. Please refer to the graphs provide in the notebook for a detailed view on the dominant features. Again, the K-MEANS model was stored on S3 for future usage.

With clustering part done, one then moved to the binary classification task. Re-using the cleaning and PCA steps already performed, one then fed the dimensionality reduced mailing dataset to AWS XGBoost model to predict whether or not a person is likely to become a customer. As already identified, the loss function was set to binary logistic and the metric to Area Under Curve (AUC). The dataset was split between a training and a validation set representing 30% of the data. Also, hyperparameters were derived from an AWS autoML hyperparameters tuning job. However, this tuning job was performed with autoML determining itself the cleaning & preprocessing steps for the features. It could be that it wrongly preprocessed many of the categorical data not recognizing that label-encoding was not appropriate or that one data was actually categorical and not numerical. This shall be further analyzed.

[1] Note that one-hot encoding has been preferred to label encoding for all categorical features. This could be refined with

Once the XGBoost model was trained, the test set was evaluated through it and sent to Kaggle. A very bad score of 0.5134 was obtained meaning that the model performs just slightly better than random guessing (which would give a value of 0.5).

In order to improve the performances, the following steps have already been imagined :

- Studying other students proposals : it appears that people obtaining a score of 0.8 took very similar steps compared to the steps performed in the present work. Therefore, EDA and preprocessing steps performed do not seem to be completely off. Maybe a more detailed label encoding and one-hot encoding would improve the results. Also, maybe not binning categories of categorical features would help;
- Using other imputing methods for missing values than “frequent” or “mean” e.g. using nearest neighbors;
- Using embedding to better translate categorical features compared to one-hot encoding;
- Tuning the hyperparameter based on the cleaned datasets rather than letting autoML use the raw datasets
- Using another algorithm would be possible but the performed review shows XGBoost should provide good results regarding the dataset

Results

As already identified in the results chapter, a metric of 0.5134 was obtained. This metric is just slightly better than random guessing and should be improved. The proposed steps are already identified in the methodology chapter.