Capstone Project - CREATE A CUSTOMER SEGMENTATION REPORT FOR ARVATO FINANCIAL SERVICES

Udacity - MACHINE LEARNING ENGINEER NANODEGREE

Radu L. Enuca
Bucharest, Romania
`https://github.com/raduenuca`
`https://www.linkedin.com/in/raduenuca`

August 2, 2019

ABSTRACT

This project, analyzes demographics data for customers of a mail-order sales company [1] in Germany, comparing it against demographics information for the general population. EDA [2] is performed to understand and clean the data. We'll use unsupervised learning techniques to perform customer segmentation, identifying the parts of the population that best describe the core customer base of the company. Then, we'll apply what we've learned on a third dataset with demographic information for targets of a marketing campaign for the company, and use a model to predict which individuals are most likely to convert into becoming customers for the company.

***K*eywords** Exploratory Data Analysis · Unsupervised Learning · Supervised Learning

1 Problem Statement

A company that performs mail-order sales in Germany is interested in identifying facets of the population that are most likely to be purchasers of its products for a mail-out campaign.

The goal is to identify segments of the population that form the core customer base for the company. These segments can then be used to direct marketing campaigns towards audiences that have the highest expected rate of returns.

2 Datasets and Inputs

There are four data files associated with this project:

- *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).

The "CUSTOMERS" file contains three extra columns ('CUSTOMER_GROUP', 'ONLINE_PURCHASE', and 'PRODUCT_GROUP'), which provide broad information about the customers depicted in the file. Each row of the

[1]The data has been provided by Bertelsmann Arvato Analytics and represents a real-life data science task.
[2]Exploratory Data Analysis

demographics files represents a single person, but also includes information outside of individuals, including information about their household, building, and neighborhood.

3 Solution Statement

The information from the first two files will be used to figure out how customers ("CUSTOMERS") are similar to or differ from the general population at large ("AZDIAS"), then use this analysis to make predictions on the other two files ("MAILOUT"), predicting which recipients are most likely to become a customer for the mail-order company.

4 Benchmark Model

The original "MAILOUT" file included one additional column, "RESPONSE", which indicated whether or not each recipient became a customer of the company. For the "TRAIN" subset, this column has been retained, but in the "TEST" subset it has been removed; it is against that withheld column that the final predictions will be assessed in a Kaggle competition.The higher the score obtained in the Kaggle competition, the better the model is at predicting customers.

5 Evaluation Metrics

The evaluation metric for the Kaggle competion is AUC for the ROC curve, relative to the detection of The evaluation metric for the Kaggle competition is AUC for the ROC curve, relative to the detection of customers from the mail campaign. A ROC, or receiver operating characteristic, is a graphic used to plot the true positive rate (TPR, the proportion of actual customers that are labeled as so) against the false positive rate (FPR, the proportion of non-customers labeled as customers).

The line plotted on these axes depicts the performance of an algorithm as we sweep across the entire output value range. We start by accepting no individuals as customers (thus giving a 0.0 TPR and FPR) then gradually increase the threshold for accepting customers until all individuals are accepted (thus giving a 1.0 TPR and FPR). The AUC, or area under the curve, summarizes the performance of the model. If a model does not discriminate between classes at all, its curve should be approximately a diagonal line from (0, 0) to (1, 1), earning a score of 0.5. A model that identifies most of the customers first, before starting to make errors, will see its curve start with a steep upward slope towards the upper-left corner before making a shallow slope towards the upper-right. The maximum score possible is 1.0 if all customers are perfectly captured by the model first.

6 Project Design

6.1 Preprocessing

6.1.1 Assess Missing Data

In this step the demographics data is assessed in terms of missing data at the column level, row-level and data level (some values in columns could also represent unknown data, and we need to re-encode them to numpy NaN).

6.1.2 Select and Re-Encode Features

Checking for missing data isn't the only way in which one can prepare a dataset for analysis. Since the unsupervised learning techniques that we will use, will only work on data that is encoded numerically, we need to make a few encoding changes or additional assumptions to be able to make progress. Besides, while almost all of the values in the dataset are encoded using numbers, not all of them represent numeric values:

- For numeric and interval data, these features can be kept without changes.
- Most of the variables in the dataset are ordinal. While ordinal values may technically be non-linear in spacing, we can make the simplifying assumption that the ordinal variables can be treated as being an interval (that is, kept without any changes).
- Special handling may be necessary for the remaining two variable types: categorical, and 'mixed.'

6.1.3 Create a Preprocessing Pipeline

All the cleaning steps above need have to be reapplied three more times (for the Customers and Mailout datasets), and in this step, we create a pipeline for preprocessing the data in one go.

6.2 Feature Transformation

6.2.1 Feature Scaling

Before we apply dimensionality reduction techniques [Shl05] to the data, we need to perform feature scaling so that the natural differences in scale for features do not influence the principal component vectors.

6.2.2 Perform Dimensionality Reduction

In this step, we apply dimensionality reduction techniques on the scaled data:

- Use sklearn's PCA [Bui+13b] class to apply principal component analysis on the data, thus finding the vectors of maximal variance in the data. For a start, no parameters are set (so all components are computed).
- Check out the ratio of variance explained by each principal component as well as the cumulative variance explained.
- We select a value for the number of transformed features we'll retain for the clustering part of the project.
- Once we've chosen the number of components to keep, we re-fit a PCA instance to perform the decided-on transformation.

6.2.3 Interpret Principal Components

Now that we have our transformed principal components, it's an excellent idea to check out the weight of each variable on the first few components to see if they can be interpreted in some fashion.

Each principal component is a unit vector that points in the direction of highest variance (after accounting for the variance captured by earlier principal components). The further a weight is from zero, the more the principal component is in the direction of the corresponding feature. If two features have large weights of the same sign (both positive or both negative), then increases in one tends to be associated with increases in the other. To contrast, features with different signs can be expected to show a negative correlation: increases in one variable should result in a decrease in the other.

To investigate the features, we map each weight to their corresponding feature name, then sort the features according to weight. The most interesting features for each principal component, then, are those at the beginning and end of the sorted list.

We investigate and interpret feature associations from the first three principal components in this substep.

6.3 Clustering

6.3.1 Apply Clustering to General Population

We've assessed and cleaned the demographics data, then scaled and transformed them. Now, it's time to see how the data clusters in the principal components space. In this substep, we apply k-means clustering to the dataset and use the average within-cluster distances from each point to their assigned cluster's centroid to decide on a number of clusters to keep.

The following substeps are taken:

- Use sklearn's KMeans [Bui+13a] class to perform k-means clustering on the PCA-transformed data.
- Compute the average difference from each point to its assigned cluster's center.
- Perform the above two steps for several different cluster counts. You can then see how the average distance decreases with an increasing number of clusters. However, each additional cluster provides a smaller net benefit. Use this fact to select a final number of clusters in which to group the data.
- Once we've selected a final number of clusters to use [PDN04], we re-fit a KMeans instance to perform the clustering operation and also obtain the cluster assignments for the general demographics data.

6.3.2 Apply All Steps to the Customer Data

Now that we have clusters and cluster centers for the general population, it's time to see how the customer data maps on to those clusters. We're going to use the fits from the general population to clean, transform, and cluster the customer data.

6.3.3 Compare Customer Data to Demographics Data

At this point, we have clustered data based on demographics of the general population of Germany and seen how the customer data for a mail-order sales company maps onto those demographic clusters. In this substep, we compare the two cluster distributions to see where the strongest customer base for the company is.

6.4 Supervized Learning Model

Now that we've found which parts of the population are more likely to be customers of the mail-order company, 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.

Before training a model, we split the training data into train and validation subsets. We also need to pay special attention to the fact that we are dealing with an unbalanced dataset. After applying the preprocessing pipeline, we are left with:

- Customers: 433: 1.24%
- Non-customers: 34539: 98.76%

There are several techniques to deal with an imbalanced dataset:

- Under-sampling of the majority class
- Oversampling of the minority class (most common is SMOTE [Cha+02])
- Ensemble methods (Ensemble of Sampler)

After dealing with the imbalance problem, we apply the dimensionality reduction and compute the K-means clusters for each sample. We concatenate the resulting cluster labels to the dataset and train a classifier on it.

For the classifier, a choice between XGBoost, LightGBM, and CatBoost [Swa18] will be made after we tune the hyper-parameters of both models using Bayesian optimization [Koe18].

After we've created a model to predict which individuals are most likely to respond to a mailout campaign, it's time to test that model in a competition through Kaggle. The entry to the competition is a CSV file with two columns. The first column is the copy of "LNR," which acts as an ID number for each individual in the "TEST" partition. The second column, "RESPONSE," is a measure of how likely each individual became a customer – this might not be a straightforward probability.

References

Buitinck, Lars et al. *sklearn.cluster.KMeans*. 2013. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`.

– *sklearn.decomposition.PCA*. 2013. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`.

Chawla, Nitesh V. et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (June 2002). https://arxiv.org/pdf/1106.1813.pdf, pp. 321–357.

Koehrsen, Will. *Automated Machine Learning Hyperparameter Tuning in Python*. July 2018. URL: `https://towardsdatascience.com/automated-machine-learning-hyperparameter-tuning-in-python-dfda59b72f8a`.

Pham, D T, S S Dimov, and C D Nguyen. "Selection of K in K-means clustering". In: *Columbia Engineering - Eletrical Engineering* (May 2004). https://www.ee.columbia.edu/ dpwe/papers/PhamDN05-kmeans.pdf, pp. 1–17.

Shlens, Jonathon. "A Tutorial on Principal Component Analysis". In: *Carnegie Mello University - School of Computer Science* (Dec. 2005). https://www.cs.cmu.edu/ elaw/papers/pca.pdf, pp. 1–13.

Swalin, Alvira. *CatBoost vs. Light GBM vs. XGBoost*. Mar. 2018. URL: `https://www.kdnuggets.com/2018/03/catboost-vs-light-gbm-vs-xgboost.html`.