Project Overview

Arvato Financial Services, a subsidiary of the Bertelsmann Group seeks to optimize its mail order sales to attract new customers. For this purpose it has a dataset with information on these current customers and another dataset on the general population of Germany. we have deployed in this project a model of supervised learning and a model of unsupervised learning. The unsupervised learning model aims to find the characteristics of typical customers of the targeted mail order company. With the supervised learning model the goal and predict which person would respond to a mailing campaign. Like any machine learning project, I start with :

- The laborious step of cleaning, preparing and selecting the relevant variables: this step is known as features engineering in machine learning language.
- Then I move on to the unsupervised learning stage where I first perform PCA and then perform clustering with the Kmeans method and make comparisons between the current customers and the individuals in the global population in the different clusters.
- I finish by the supervised learning stage where I train several models (regression logging, random forest, XGboost...) to choose the best one for prediction.

  > In this post I present only the outlines and the results. The code and the different details can be freely consulted in the notebook that can be found in github

Project data

The data provided by Arvato consists of four datasets:

- Demographics data for the general population of Germany which contains 891 211 persons (rows) and 366 features (columns).
- Demographics data for customers of a mail-order company with 191 652 persons (rows) and 369 features (columns).
- Demographics data for individuals who were targets of a marketing campaign; 42 982 persons (rows) x 367 (columns). In the project I use this dataset to train the supervised models.
- Demographics data for individuals who were targets of a marketing campaign; 42 833 persons (rows) x 366 (columns). This dataset is use to make prediction after training models with the training dataset.

In addition to the four datasets, two excel files were also provided for descriptive purposes. These two files are very useful in that they allow for an understanding of the data. These two files are:

- `DIAS Information Levels - Attributes 2017.xlsx`:a top-level list of attributes and descriptions, organized by informational category
- `DIAS Attributes - Values 2017.xlsx:` a detailed mapping of data values for each feature in alphabetical order

Data cleaning and features engineering

In the data cleaning step, the following steps were performed:

- Encode missing values as NaNs
- Drop all categorical variables with more than 10 modalities
- Re-encode categorical variables by creating dummy variables
- Base on the distribution of the percentage of missing values per column and per row I take the decision as follow:
  - Drop all columns with more than 20 % of missing values. The figure (1) indicates a preponderance of the number of variables with less than 20 % missing values.
  - For the raw I drop those which have more than 18% of missing value figure (2)

Generate a cleaning function that performs all the above steps so that it can be run on the customers dataset as well Impute missing values using the mean. I have tried we the median but I got a `memoryError`.

Perform feature scaling using StandardScaler

Figure 1: Distribution of missing values per column

Figure 2: Distribution of missing values per row

Unsupervised learning for customers segmentation

PCA

After the data cleaning and features engineering, I created a function to perform PCA. Below is the code of the function to make the PCA easier

```python
def make_pca(n_components, dataset):
    '''
    Transforms data using PCA to create n_components, and provides back the results of the
    transformation.

    INPUT: n_components - int - the number of principal components to create
           dataset - the dataset you would like to transform. This dataset must be cleaned and standardize before
performing
           the PCA
    OUTPUT: pca - the pca object created after fitting the data
            fit_pca - the transformed  matrix with new number of components
    '''
    pca = PCA(n_components, svd_solver='full')
    fit_pca = pca.fit_transform(dataset)
    return pca, fit_pca
```

Figure 3: PCA scree plot of the demographics data

I could not decide on an optimal number of main components to retain after the PCA. I therefore decided to retain the p best principal components that would achieve at least 90% of the total variance of the dataset. To do that, I created a function to facilitate the selection.

Below is the code of the function that selects the optimal number of principal components based on the minimum variance that we want to keep.

```python
def optimal_n_componemt(pca, min_variance):

    ''' return the minimal number of the first most important components that
        able to capture at least nin_variance

        INPUT: pca - the result of instantian of PCA in scikit learn , min_variance: the minimum of cumulative variance we
want to target

        OUTPUT: The optimal number of component
   '''
    cum_var=np.cumsum(pca.explained_variance_ratio_)
    L1=len(pca.explained_variance_ratio_)
    L2=len(pca.explained_variance_ratio_[cum_var > min_variance])
    return L1-L2 + 1
```

Applying the function `optimal_n_componemt`, the optimal number of principal components to capture **90 %** of the total variance of the data set is **155**.

For the remainder of this project, I choose 155 principal components. It is on these components that I realize after clustering with the Kmeans method.

Kmeans

I perform the Kmeans on the 155 main components and using the elbow figure (4) graph I choose 6 clusters to perform the segmentation. Looking at the elbow graph, I see that the optimal number of clusters is between 4 and 10. My intuition led me to choose 6 clusters.

Figure 4: Elbow plot for k-means clustering

After the Kmeans, I plotted the population individuals and clients on the same graph to get an overview of the over-represented cluster. I found that cluster 3 figure (5) is the most over-represented in the customer data. This segment should be the company's target market in the general population.

Figure 5: Elbow plot for k-means clustering

Prediction of potential customers using Supervised learning

The last step of the project is devoted to the construction of a model to predict the next the conversion of customers.

In this part I have built several models and compared their performances on the basis of the ROC AUC to choose the best one. The training dataset used consists of 42,982 lines with 367 features so the target variable coded in 0 and 1 (1 if the person became a customer of the company and 0 otherwise).

The test dataset consists of 42,833 persons. The choice of the ROC AUC as is interesting because we are facing a situation of imbalanced data (1.2% conversion versus 98.8 non-conversion).

Before training the models, I subject the data set to the same pre-treatments as before. I trained 7 models whose results are recorded in the table below.

Among these models the `xgboost` model has the highest ROC AUC(0.717) followed by the Gradient boosting classifier(0.70) and the Adaboost classifier(0.689). The ensemble models perform better than the others models here.

I tuned the hyperparameters of the xgboost and used the optimal parameters to train the final xgboost model and the used this model to make prediction with the test data.

After the prediction with the test dataset, I saved the results in an excel file which I submitted on kaggle.

Conclusion and areas of improvement

The following project, which is a real-world project, was a very exciting challenge. From data processing to prediction to the buiding of supervised and unsupervised models, all of its steps require a lot of ingenuity and attention. The initial dataset contains a lot of information (more than 360 features for global population demographics dataset and the same for the customer dataset). This large number of variables implies to make often very radical choices (for example I removed all columns with more than 20% missing values and rows with more than 18% missing values). In spite of this, I was able to bring the project to its end with satisfactory results. Indeed, with unsupervised learning I further segmented the population and customers into 6 clusters and identified the cluster that would potentially match Arvato's target. In addition, I was able to set up 7 models and choose the one that was the best for predicting Arvato's potential customers.

However, the work I've just done is still something that can be greatly improved.

For example, when imputing missing values, I wanted to try more powerful imputation methods than average imputation, but I was confronted with a memory problem. I wanted to try the method of imputation by the median, the imputation by KNN etc but it did not work. Also in the choice of the supervised learning model another axis of improvement would be the use of artificial neural