Machine Learning Engineer Nanodegree

Project Report

Customer Segmentation Report for Arvato Financial Solutions

Marcos González Collado

1. Definition

1.1. Project Overview

Arvato is a global services company headquartered in Gütersloh, Germany. Its services include customer support, information technology, logistics, and finance. The history of Arvato goes back to the printing and industry services division of Bertelsmann; the current name was introduced in 1999. Today, Arvato is one of eight divisions of Bertelsmann, the media, services and education group. In 2016, Arvato had about 68,463 employees and an overall turnover of 3.84 billion euros.

In this project, Arvato helps a mail order company to understand the demographics of the population in Germany and match them with their customer data, in order to estimate whether a person is a potential customer or not, based on their demographics.

1.2. Problem Statement

In summary, the problem is: How can you use demographic data to estimate whether or not a person may be a potential customer?

The chosen way to carry out the resolution of this problem, is indicated in the project_design.pdf file.

1.3. Metrics

In the customer segmentation problem, we have been used the weight in each cluster of the main features to assess which features are most influential on our customers.

In the second problem, we have been used the Confusion Matrix, Precision, Recall and Area under the Receiver Operating Curve (AUROC).

As we have seen, the target is highly unbalanced. In training dataset, there are 42430 values 0 and only 532 values 1 (for every value 1, there are almost 80 values 0), because of, we must choose a metric that takes this class imbalance into account. The most common metrics for this are Precision and Recall or Area under Receiver Operating Curve (AUROC).

- Area under the ROC Curve — It provides an aggregate measure of performance across all possible classification thresholds.
- Precision — Also called Positive predictive value. The ratio of correct positive predictions to the total predicted positives.
- Recall — Also called Sensitivity, Probability of Detection, True Positive Rate. The ratio of correct positive predictions to the total positives examples.

2. Analysis

2.1. Data Exploration

The first steps check dataset integrity and dimensions and the extra columns match the description.

We can look at the datatypes of columns 18 and 19 since we got a warning while we loaded the data and fix it. There are 'X' and 'XX' as values in these columns which have not been given in the description, also there are 'nan' values. In these features, we will replace 'X' and 'XX' values to numpy nan.

Explore metadata datasets

Additionally we have been given two excel books.

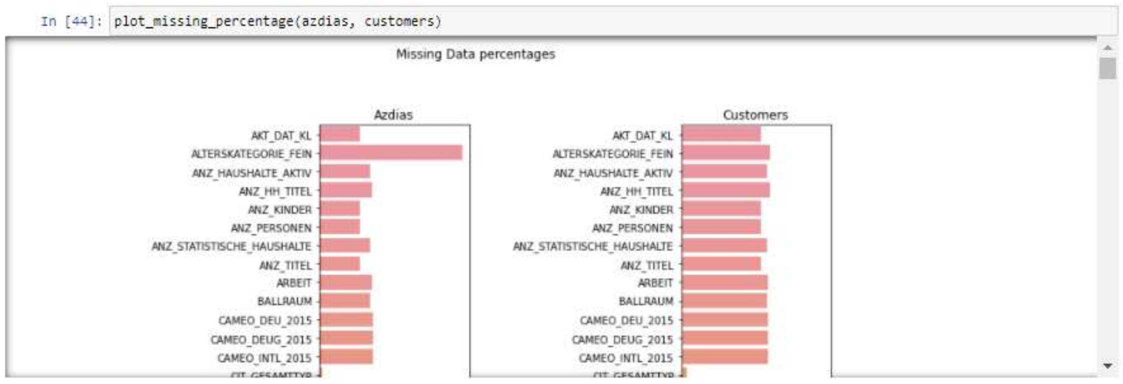
- DIAS Attributes - Values 2017.xlsx - Has a description of demographic columns, information about range of values each column can take and their respective meanings.Loaded in metadata_values.
- DIAS Information Levels - Attributes 2017.xlsx - Has detailed information about each column with additional notes wherever required. Loaded in metadata_info_levels.

We can see that there are only 313 (in metadata_info_levels dataset) and 314 (in metadata_values dataset) different columns described, but in the AZDIAS dataset, there are 366 demographic columns. We need to check which features have no description in metadata datasets. To do this, we explore and use these two dataframes to understand the data and check how many features are described in the metadata datasets.

```
Number of described demographic columns:  272
Number of undescribed demographic columns:  42
```

2.2. Exploratory Visualization

We performed an exploratory visualization using matplotlib to check that the features have more or less the same percentage of null values.



```
In [44]: plot_missing_percentage(azdias, customers)
```

2.3. Algorithms and Techniques

Algorithms used throughout the project were as follows:

- LogisticRegression
- DecisionTreeClassifier
- RandomForestClassifier
- AdaBoostClassifier
- GradientBoostingClassifier

Techniques used throughout the project were as follows:

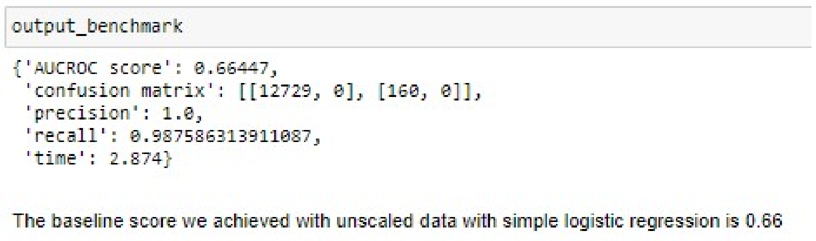
- SimpleImputer: Imputation transformer for completing missing values. In this project, we use default 'most_frequent' strategy that replaces missing using the most frequent value along each column.
- StandardScaler: Standardize features by removing the mean and scaling to unit variance. The standard score of a sample x is calculated as:

  $$z = (x - u) / s$$

  Where *u* is the mean of the training samples or zero if *with_mean=False*, and *s* is the standard deviation of the training samples or one if *with_std=False*.
- PCA: Principal component analysis. Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space. The input data is centered but not scaled for each feature before applying the SVD.
- train_test_split: Split arrays or matrices into random train and test subsets.
- KFold: Provides train/test indices to split data in train/test sets. Split dataset into k consecutive folds. It is used for cross validation step.
- Hyperopt: Python library to provide algorithms and parallelization infrastructure for performing hyperparameter optimization.

2.4. Benchmark

To obtain a first starting point, we will fit a simple unbalanced estimator with unscaled data. In this case we have chosen a Logistic Regression estimator.



```
output_benchmark
```

```
{'AUCROC score': 0.66447,
 'confusion matrix': [[12729, 0], [160, 0]],
 'precision': 1.0,
 'recall': 0.987586313911087,
 'time': 2.874}
```

The baseline score we achieved with unscaled data with simple logistic regression is 0.66

3. Methodology

3.1. Data Preprocessing

The steps followed for data preprocessing has been as follows:

1. **Preprocessing**

- **Clean warning columns.** Replace 'X' and 'XX' values with numpy nan in 'CAMEO_DEUG_2015' and 'CAMEO_INTL_2015' columns.

- **Nulls normalization.** Looking at the metadata of the columns in the provided excel workbooks, it can be seen that some columns have their own category to indicate a null value (for example categories like 'unknown', 'unknown / no main age detectable', 'numeric value', etc..). We have unified the null values and replace these categories with np.nan values. It has only been done for the 272 columns that are described in the metadata datasets, as for the remaining 42 columns, we do not know the meaning of the values.
  The replaced values have been: ['unknown', 'unknown / no main age detectable', 'numeric value', 'none']
  Function used: *replace_unknown_values()*

2. Univariate Analysis

- **Remove missing columns.** Get and remove columns exceeding the null values threshold (default=0.3).
  Function used: *get_missing_columns(), drop_columns()*

- **Remove missing rows.** Get and remove rows exceeding the null values threshold per row (default = 50). If threshold is None, the threshold will be the maximum.
  Function used: *get_missing_rows(), drop_rows()*

- **Fix datetime columns.** Convert datetime columns to int, replacing datetime value by year. This function applies to 'EINGEFUEGT_AM' column.
  Function used: *fix_datetime_columns()*

- **Translate column categories.** Translate column values from translations dictionary. This function applies to binarize 'OST_WEST_KZ' column and translate categorical to numerical values.
  Function used: *translate_columns()*

- **Categorical to numerical transformation.** Translate categorical to numerical values from translations dictionary. The first n_categories will be translated by the order of their ranking by number of occurrences, the rest will go to the category

'0'. This function applies to 'CAMEO_DEU_2015' and 'D19_LETZTER_KAUF_BRANCHE' columns.
Function used: *categorical_to_numerical_transformation ()*

- **Check id columns.** Checks that all values in columns labelled as id are unique. This function applies to 'LNR' column.
  Function used: *check_id_columns()*

- **Imputing missing values.** Imputing missing values according to strategy (default='most_frequent').
  Function used: *imputing_missing_values()*

3. Feature Scaling

Standardize all dataset features by removing the mean and scaling to unit variance.

Function used: *feature_scaling()*

3.2. Implementation

A number of functions and classes have been defined as utils. These functions and classes will be used throughout the project and are as follows:

Functions for managing logger.

- get_logger(). Creates logger common object with formatting and WARNING level.
- set_log_level(). Set log level to verbose or debug.

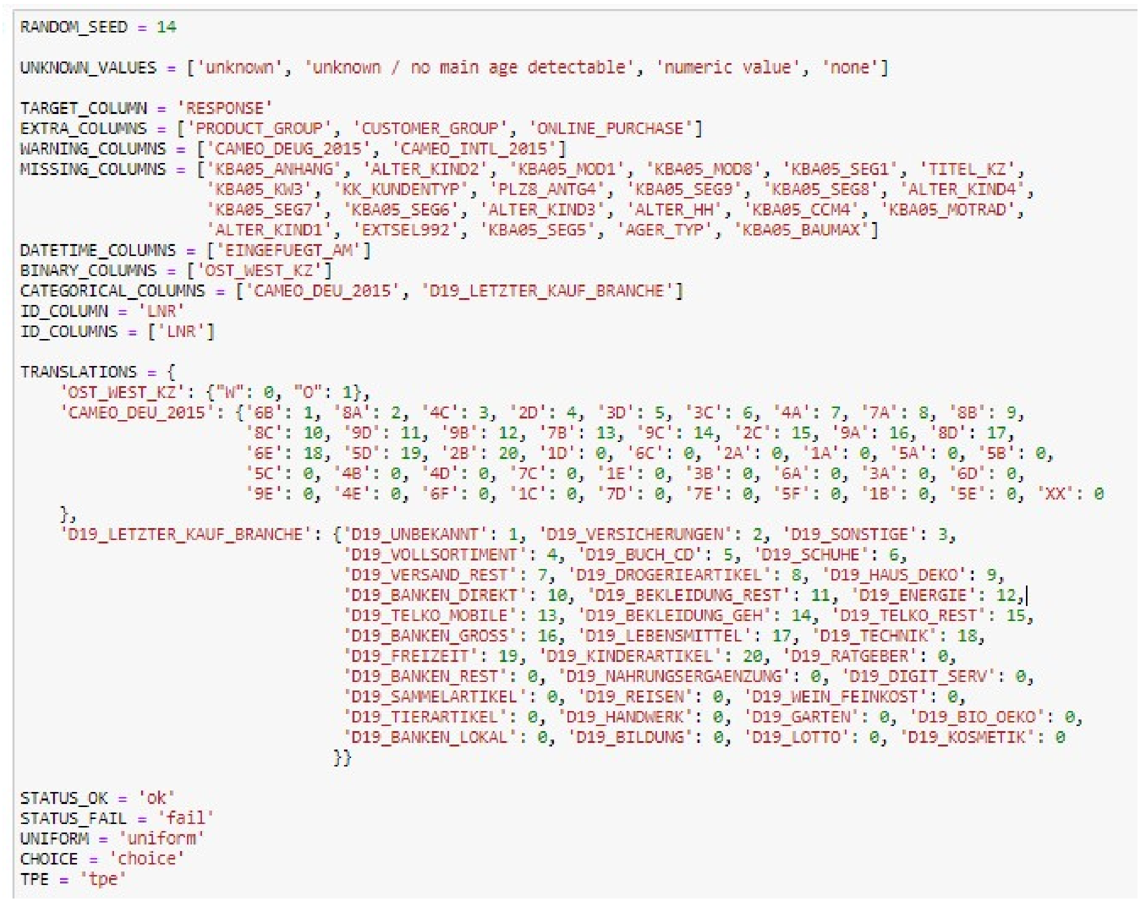
Timer: Class for managing runtimes.

- start(). Start a time account.
- lapse(). Get the elapsed time since the last Timer.start() or Timer.lapse()
- total(). Get the elapsed time since the last Timer.start().

Other utils:

- print_df_full(). Print full dataframe.
- convert_to numeric(). If possible converts string to float, else returns a string.

A number of constants are defined after Preprocessing and Univariate Analysis stages that will be used throughout the project:

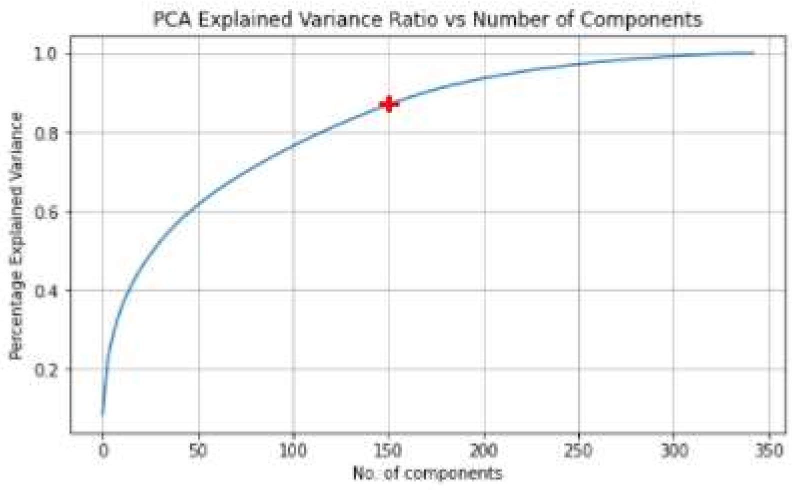


```
RANDOM_SEED = 14

UNKNOWN_VALUES = ['unknown', 'unknown / no main age detectable', 'numeric value', 'none']

TARGET_COLUMN = 'RESPONSE'
EXTRA_COLUMNS = ['PRODUCT_GROUP', 'CUSTOMER_GROUP', 'ONLINE_PURCHASE']
WARNING_COLUMNS = ['CAMEO_DEUG_2015', 'CAMEO_INTL_2015']
MISSING_COLUMNS = ['KBA05_ANHANG', 'ALTER_KIND2', 'KBA05_MOD1', 'KBA05_MOD8', 'KBA05_SEG1', 'TITEL_KZ',
                   'KBA05_KW3', 'KK_KUNDENTYP', 'PLZ8_ANTG4', 'KBA05_SEG9', 'KBA05_SEG8', 'ALTER_KIND4',
                   'KBA05_SEG7', 'KBA05_SEG6', 'ALTER_KIND3', 'ALTER_HH', 'KBA05_CCM4', 'KBA05_MOTRAD',
                   'ALTER_KIND1', 'EXTSEL992', 'KBA05_SEG5', 'AGER_TYP', 'KBA05_BAUMAX']
DATETIME_COLUMNS = ['EINGEFUEGT_AM']
BINARY_COLUMNS = ['OST_WEST_KZ']
CATEGORICAL_COLUMNS = ['CAMEO_DEU_2015', 'D19_LETZTER_KAUF_BRANCHE']
ID_COLUMN = 'LNR'
ID_COLUMNS = ['LNR']

TRANSLATIONS = {
    'OST_WEST_KZ': {"W": 0, "O": 1},
    'CAMEO_DEU_2015': {'6B': 1, '8A': 2, '4C': 3, '2D': 4, '3D': 5, '3C': 6, '4A': 7, '7A': 8, '8B': 9,
                       '8C': 10, '9D': 11, '9B': 12, '7B': 13, '9C': 14, '2C': 15, '9A': 16, '8D': 17,
                       '6E': 18, '5D': 19, '2B': 20, '1D': 0, '6C': 0, '2A': 0, '1A': 0, '5A': 0, '5B': 0,
                       '5C': 0, '4B': 0, '4D': 0, '7C': 0, '1E': 0, '3B': 0, '6A': 0, '3A': 0, '6D': 0,
                       '9E': 0, '4E': 0, '6F': 0, '1C': 0, '7D': 0, '7E': 0, '5F': 0, '1B': 0, '5E': 0, 'XX': 0
    },
    'D19_LETZTER_KAUF_BRANCHE': {'D19_UNBEKANNT': 1, 'D19_VERSICHERUNGEN': 2, 'D19_SONSTIGE': 3,
                                 'D19_VOLLSORTIMENT': 4, 'D19_BUCH_CD': 5, 'D19_SCHUHE': 6,
                                 'D19_VERSAND_REST': 7, 'D19_DROGERIEARTIKEL': 8, 'D19_HAUS_DEKO': 9,
                                 'D19_BANKEN_DIREKT': 10, 'D19_BEKLEIDUNG_REST': 11, 'D19_ENERGIE': 12,
                                 'D19_TELKO_MOBILE': 13, 'D19_BEKLEIDUNG_GEH': 14, 'D19_TELKO_REST': 15,
                                 'D19_BANKEN_GROSS': 16, 'D19_LEBENSMITTEL': 17, 'D19_TECHNIK': 18,
                                 'D19_FREIZEIT': 19, 'D19_KINDERARTIKEL': 20, 'D19_RATGEBER': 0,
                                 'D19_BANKEN_REST': 0, 'D19_NAHRUNGSERGAENZUNG': 0, 'D19_DIGIT_SERV': 0,
                                 'D19_SAMMELARTIKEL': 0, 'D19_REISEN': 0, 'D19_WEIN_FEINKOST': 0,
                                 'D19_TIERARTIKEL': 0, 'D19_HANDWERK': 0, 'D19_GARTEN': 0, 'D19_BIO_OEKO': 0,
                                 'D19_BANKEN_LOKAL': 0, 'D19_BILDUNG': 0, 'D19_LOTTO': 0, 'D19_KOSMETIK': 0
                                }}

STATUS_OK = 'ok'
STATUS_FAIL = 'fail'
UNIFORM = 'uniform'
CHOICE = 'choice'
TPE = 'tpe'
```

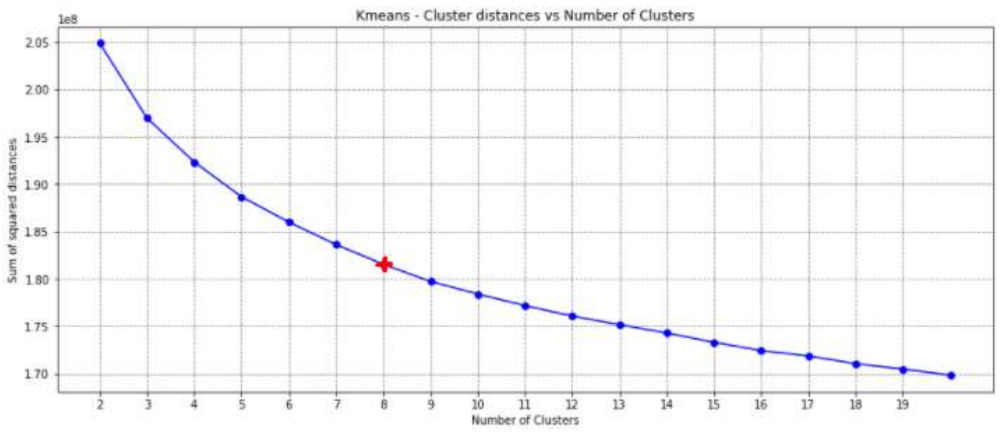
3.3. Refinement

Customer Segmentation Report

- **Principal Component Analysis.** Is a technique for reducing the dimensionality of datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance. In this project we first run a PCA analysis on all variables, and then choose only those that explain around 90% of variance in the dataset, in our case, 150 variables.



- **Plot elbow:** The elbow method helps to choose the optimum value of *'k'* (number of clusters) by fitting the model with a range of values of *'k'*. Here we would be using a 2-dimensional data set but the elbow method holds for any multivariate dataset.
  To determine the optimal number of clusters, we have to select the value of k at the "elbow", that is, the point after which the distortion/inertia start decreasing in a linear fashion. We must also take into account that higher the value of 'k', more is the number of iterations the algorithm takes to converge. In this case, we have considered that k=8 may be a good choice.



Supervised Learning Model

Once the benchmark has been defined, we have carried out a series of tests to try to refine the model and improve the results. These tests have been:

- **Evaluate different binary classification estimators** with default parameters and unscaled data, the estimators evaluated have been:
  - LogisticRegression
  - DecisionTreeClassifier
  - RandomForestClassifier
  - GradientBoostingClassifier
  - AdaBoostClassifier

- **Balancing class weights.** The target is very unbalanced, there are three estimators that have a parameter to account for class balancing.
  - LogisticRegression(class_weight='balanced')
  - DecisionTreeClassifier(class_weight='balanced')
  - RandomForestClassifier(class_weight='balanced')

- **Scaling data without balanced class weight parameter.** Test these binary classification estimators, taking into account the scaling of the data.

- **Scaling data with balanced class weight parameter.** Test these binary classification estimators, taking into account the scaling of the data and class balancing.

- **Hyperparameter optimization step.** Hyperparameters are crucial as they control the overall behaviour of our model. The ultimate goal is to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results. In this project, we use Hyperopt library that has been designed to accommodate Bayesian optimization algorithms based on Gaussian processes and we performs the optimization trainings with a cross-validation process.

4. Results

4.1. Model Evaluation and Validation

Once we have trained the chosen model with the optimal hyperparameters, the last step is to train it using the whole training dataset and use the test dataset to make a prediction. To do this, we carry out the following steps:

- Apply Preprocessing and Univariate Analysis setps to test dataset.
- Check id column and number of features in test dataset.
- Build a model with best hyperparameters.
- Scale test dataset.
- Get prediction and prediction probability for each id.

4.2. Justification

I consider this estimator to be optimal for solving a binary classification problem. Taking into account all evaluations, we see that the best AUCROC score is achieved by the GradientBoostingClassifier, however we will choose to select the AdaBoostClassifier, as it has a similar AUCROC score, takes about 4 times less time to train and the confusion matrix has 0 FP.

The best AdaBoostClassifier hyperparameters chosen are:

- 'algorithm': 'SAMME',
- 'learning_rate': 0.4170011267581649,
- 'n_estimators': 60

With this hyperparameters we have the score: 0.7659

It looks like a good result.