Descriptive and Predictive Analytics Project
'DSC Re-Activation Campaign Plan 2019'

Documentation

Group	Gutiérrez, Sara Montenegro, Carlos Olivera, Andrés Thomas, Brayan Varghese, Betsy
Date	28/11/2019
Due Date	02/12/2019
Version	Alpha 1.0
Objective	Provide a detailed description about the process involved during the development of the project

Contents

Data preparation

Donor data

1. We cleaned the **gender** variable by assigning "NA" to all the observations with value: "U".
2. We transformed the **language** variable into a binary variable by assigning 1 to the French speakers. We then renamed the variable **language** to **is_french.**

Gift data

1. We converted the **campID** variable into a string variable and replaced the '0' values for 'NA'.
2. We transformed the **date** variable into a date time variable.
3. We created the variables **year** and **months** to extract the year and the month from the **date** variable.
4. We created variables for the total amount of donations by donor in the previous year, the previous 3 years, the previous 5 years and the previous 10 years and 18 years. These variables were named **gifts_1_year, gifts_3_year, gifts_5_year, gifts_10_year**, **gift_18_year** respectively. We renamed these variables **18y, 10y, 5y, 3y, 1y.**
5. We merged these variables into a table named **final_merge_gifts.**
6. We pivoted the **gifts** table two times: one with the year **(gifts_1)** values and one with the month **(gifts_2)** values.
7. We merged both tables into a table named **gifts_final.**
8. We created the variable **frequency_gifts,** which captures how many times the donor has donated over the last 18 years.
9. From the **gifts_final** table we created four variables for the year quarters: **Q1, Q2, Q3, Q4.**
10. We dropped the year variables (1995 until 2014) and renamed this table **gifts_quarter.**
11. We merged the **final_merge_gifts** and the **gifts_quarter.** New table: **gifts_final.**
12. We added a variable **gifts_date** that captures the latest donations before 2014 by donor.
13. We added a variable **recency_number** that captures the time passed since the last donation.
14. The final table was renamed **gifts_final_2.**

Campaign 2013 data

1. We merged **gifts_final_2** with the **c_2013** table. The resulting table was renamed **left_1.**
2. We merged the **left_1** table with the **donor** table. The resulting table was renamed **left_2.**

Final data

1. We applied get_dummies to the **gender** variable and renamed the **left_2** table **final.**
2. We created the variable **amount_missing** as a dummy that takes the value of 1 if the donor had donated more than 35EUR in the 2013 campaign.
3. We transformed the **zipcode** variable by assigning "0" to the observations with the value "SW6 3PS" and the blank values. We then filled the "NA" values with "0". Finally, we transformed the variable into an integer.
4. We created the variable **province** and **region** by applying the function given in class.
5. We applied get_dummies to the **region** and **province** variable.
6. We created the table **final_model** that selects all variables except **donorID, amount, date** and **zip code** because we only wanted to keep the variables that could be used as features.

7. We created the table **table_model** that selects the same as **final_model** but deselects the number of months.
8. We filled all resting "NA" values with "0".
9. We **standardized all variables** in the table_model.

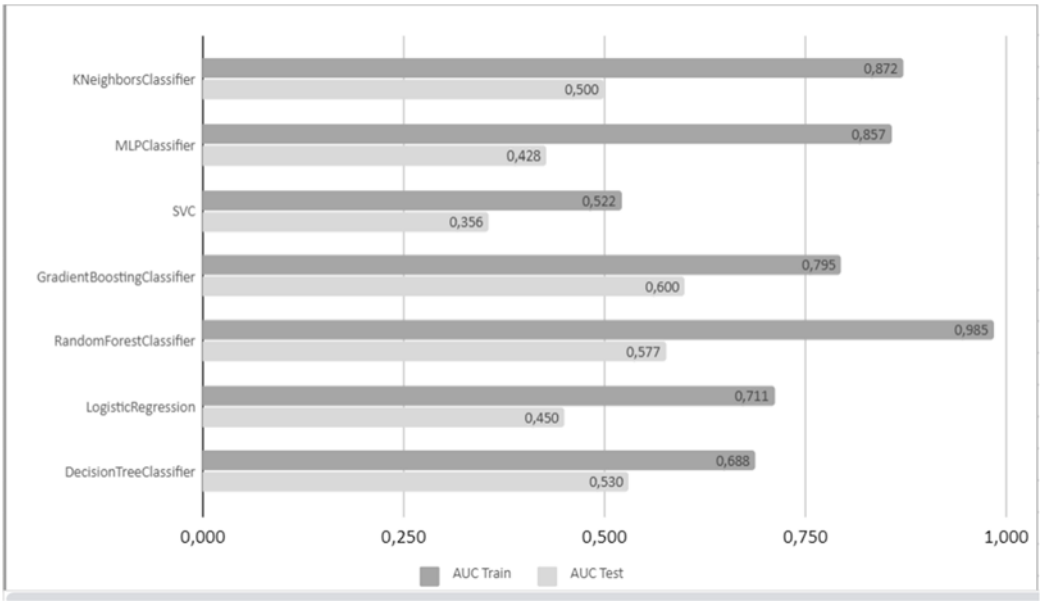
Segmentation models

1. We winsorized our continuous variables **10y, 5y, 3y, 1y, Q1, Q2, Q3, Q4.**
2. We build 5 clusters using a **K-means** clustering algorithm.
3. We printed the **statistical description** of each cluster using the describe method.

Model building

1. We renamed the **table_model** as **data**.
2. We settled the partitions and the **trainingSet** and **validationSet.** (Initially, we build the models by oversampling the training data. However, it resulted it in overfitting which consequently led to poor performance of the models on the test data. Stratification of the data also resulted in poor model performance. Below are a few results from when oversampling was incorporated.)

Model Algorithm	AUC Train	AUC Test
SVC	0,522	0,356
Logistic Regression	0,711	0,450
Random Forest Classifier	0,985	0,577
Gradient Boosting Classifier	0,795	0,600
Decision Tree Classifier	0,688	0,530
MLP Classifier	0,857	0,428
KNN Classifier	0,872	0,500



3. We selected as features all variables in the **data.**

4. We defined **amount_missing** as our target.

Predictive models

1. Since the **stepwiseRegression** was a custom defined function, we ran this stepwise regression on each of the models (using training and validation data) to find the best set of features for each of them.
2. We ran all the following models on the test data using the corresponding customized feature set derived from stepwise regression: decision trees, logistic regression, random forests, gradient boosting, support vector machines (SVM), multi-layer perceptron classifier (MLP classifier), and K nearest neighbor classifier.
3. We computed the **performance metric** of each model (accuracy and AUC score).
4. We plotted the **lift and accumulated gains** of the best model (which was based on the Gradient Boosting for Classification.)

Test set

1. We performed the same data preparation described above on the test set.
2. We choose the **Gradient Boosting Classifier** as the best model.
3. We ran the model with the selected features and the target and we then calculated the **performance metric** (prediction, probability, accuracy and AUC score).
4. We plotted the **ROC** graph for the test set.

Technical aspects

Libraries imported

- **pandas:** we used this library to convert variables to date time format, to merge tables, to apply get_dummies, among others.
- **numpy:** we used this library to calculate a time difference, to find NaN values, to load a data file, to fix a random state, to obtain absolute values, to arrange bins, among others.
- **sklearn:** we used this library to import all the models tested as well as some of the elements of the performance metric.
- **matplotlib.pyplot:** we used this library to plot stems, to plot subplots, to plot figure, to plot axes
- **scipy:** we used this library to import the Pearson correlation function.
- **ieseg:** we used this library to import the partition, roc, lift, cumulative response and cumulative gains that were used during the class.

Functions created

- **Explore:**
    - Logic: to make data exploration simpler, we created a function that prints the main information about a data set.
    - Input: data set.
    - Output: prints the head, tail, info, columns, dtypes, shape and describes methods of the inputted data.

- **Province:**
    - Logic: to create a variable that gives information about the provinces we applied a control flow to the zipcode variable.
    - Input: zip code variable.
    - Output: a variable with the province for each observation.
- **Region:**
    - Logic: to create a variable that gives information about the regions we applied a control flow to the zipcode variable.
    - Input: zip code.
    - Output: a variable with the region for each observation.
- **Setpwise Regresion:**
    - Logic: to find the best performing features for a specific model based on the AUC values for the train and test set.
    - Input: trainingSet, validationSet, selectedFeatures and target.
    - Output: returns a list of combinations of features sorted in descending order from the highest to the lowest AUC values.