Data Mining and Text Mining – Course Project

Group: SBAM
Alessandro Terragni, Filippo Pedrazzini, Riccardo Lo Bianco
Eugenio Lomurno, Niklas Renstrom

Politecnico di Milano
Prof. Pier Luca Lanzi

1

1. Data Preprocessing

1.1. Data Visualization

We started by analyzing the dataset with Tableau, a visualization tool, trying to find some interesting correlations between variables, discovering that the categorical variables were not really meaningful. This result was also confirmed by applying a univariate feature selection of the dataset with the dummy variables obtained by one hot encoding, so we decided to drop the categorical variables.

1.2. Univariate Feature Selection

We performed a univariate feature selection in order to discover the best features based on univariate statistical tests. In particular, we used the function SelectKBest from the scikit learn library, that selects features according to the k highest scores.
As we expected the first features were exactly the features that we thought being the best after the visual analysis of the dataset.

1.3. Feature Scaling

We used the StandardScaler from ScikitLearn to standardize the features by removing the mean and scaling to unit variance, in order to be able to assess the performances of each classifier with respect to the others.

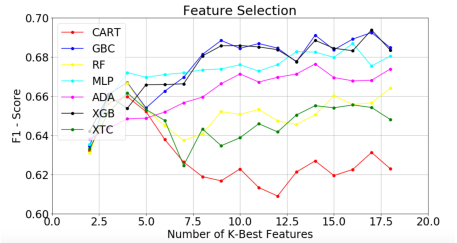
2. Model Selection

We applied several algorithms with default parameters to have an idea of the overall performances. The maximum result we reached was an F1 = 0.51 with Gaussian Naive Bayes, which has really low margins of improvement, so we tried to discover why the overall performances were not so good.

2.1. Imbalanced DataSet

Using the visualization tool and after the computation of the baseline, we discovered that the dataset is completely imbalanced and due to this fact the classifiers were not able to do predictions with a high F1 measure.
To deal with this kind of datasets we tried two different approaches.

- **Changing the Threshold**: We changed the probability threshold in order to understand if F1 could improve, plotting the precision-recall curve and the relative F1 measure of each classifier. We selected 0.3 as the best threshold and then we compared once more all the algorithms.



- **Resampling the Dataset**: we resampled the dataset in order to balance it, using a library known as imblearn. We tried different kind of resampling methods (Over-Sampling, BalanceCascade and others), but we decided to use EasyEnsemble, which creates an ensemble set by iteratively applying random under-sampling. By default, it creates 10 different samples, we used just one that is enough to perform the testing phase without losing generalization.

2.2. Algorithm Comparison with Resampled Dataset

We noticed that the performances of almost all the classifiers increased with the resampled dataset. The performances were much better compared to threshold method and so we decided to use the resampling method to deal with the problems related to the imbalancement of the dataset.
We evaluated the models using crossvalidation on the reshaped dataset, so we generated performances that are higher than those registered on a simple holdout, which are more realistic, but we followed this path to have the possiblity of comparing the algorithms on a solid base, which is not provided by the classical holdout.

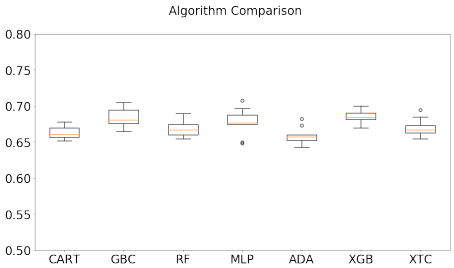
2.3. Features Learning Curve

We chose the best number of features for each classifier using the function introduced above and plotting the performances over the number of features.

2.4. Parameters Tuning

To boost the performances of each single classifier we computed the best parameters using GridSearch, which applies cross validation to a set of estimators tuned with different parameters and returns the one that has the best mean performances.

2.5. Algorithm Comparison with Reshaped Dataset after Tuning Parameters



2.6. Choice of the Model

We selected XGBoost and MLP as best classifiers for our problem and we run an unpaired statistical test but we discovered that there is not a statistical difference between the two best models. So we decided to select XGBoost since it is one of the most robust classifiers that can be used according to the literature.