Mini-project review - Track 1

Guglielmo Menchetti

December 7, 2018

1 Problem description

The task of this project is to build a classification model that is able to predict how empathic is a person. More in particular, if a person can be associated to one of the following categories: *Very Empathetic, Not Very Empathetic.*

2 Data and preprocessing

This project is based on the *Young People Survey* data, a collection of 1100 samples with 150 variables that can be split into 8 different categories. Particularly, the features are 146 *categorical* and 4 *numerical.* Moreover, the subset of categorical features also contains the label, that is the *Empathy* column. After removing this feature from the input dataset, the following preprocessing techniques has been applied:

- **Non labeled samples removed**: the examples in which the label was not defined have been removed.
- **Null values imputation**: for each feature I decided to impute the null values as the median.
- **One-hot-encoding**: in one of the approaches that I tried, I applied the one-hot-encoding for the categorical variables.
- **Remove outliers**: for the columns *Weight* and *Height* I used quantiles to determine a lower and upper bounds, such that values below and over this values were moved to this bounds.

After the preprocessing step I noticed that the distribution of the labels was not equal: in particular the class *Very Empathetic* appears around twice the time the other labels (665 vs 340). For this reason I tried to apply downsampling to the majority class, reducing the number of samples from 665 to 340, using a random selection of the examples. However, the results obtained with the entire dataset outperform the one obtained with the donwsampled dataset.

The last technique that I tried is feature selection, in order to reduce the dimensionality of the data (715 features after one-hot-encoding). I applied different approaches that are available in the *scikit-learn*[1] library, such as *PCA*, an approach based on the variance and the selection of the best features using a model (*Linear SVM*, *Extra Tree*, *Lasso*). The last one is the one that produced the best results, with *Extra Tree* as base model.

3 Models, results and baseline

In order to produce good results, I tried different classifiers. The ones that resulted in the best performance are *SVM* (both with *Linear* and *RBF* kernel), *Ridge Regression* and *Naive Bayes* (both *Multinomial* and *Gaussian*). To evaluate the performance of the classifiers and to perform hyperparameters tuning, I used the *scikit-learn* library *GridSearchCV*, that applies a k-fold crossvalidation for the model evaluation. Particularly, I set the number of folds to 5.

The baseline classifier that I used to compare my results is the *Majority Class* classifier. As evaluation metrics, I used the *accuracy* metric, but since this metric can not be representative with a non balanced dataset, I also used *precision*, *recall* and also *f1-score*. In the following table I show the results obtained, along with the hyperparameters used, for the complete dataset. The results concerning the downsampled dataset can be seen running the *test file.*

Classifier	Parameters	Acc	Prec	Rec	f1
Majority Class	-	0.64	0.64	1	0.78
SVM	C=1, linear	0.73	0.74	0.89	0.80
Ridge Classifier	$\alpha = 0.1$	0.76	0.76	0.91	0.83
Multinomial NB	-	0.75	0.76	0.88	0.82
Gaussian NB	-	0.74	0.76	0.91	0.83

(a) Result obtained using One-Hot-Encoding on the complete dataset

Classifier	Parameters	Acc	Prec	Rec	f1
Majority Class	-	0.64	0.64	1	0.78
SVM	C=1, rbf	0.69	0.69	0.94	0.80
Ridge Classifier	$\alpha = 10$	0.75	0.77	0.92	0.84
Multinomial NB	-	0.69	0.75	0.79	0.76
Gaussian NB	-	0.75	0.80	0.81	0.80

(b) Result obtained without One-Hot-Encoding on the complete dataset

[1] scikit-learn documentation