Data science lab: Process and methods

Politecnico di Torino

Project report

Giulio Alfarano – S267589

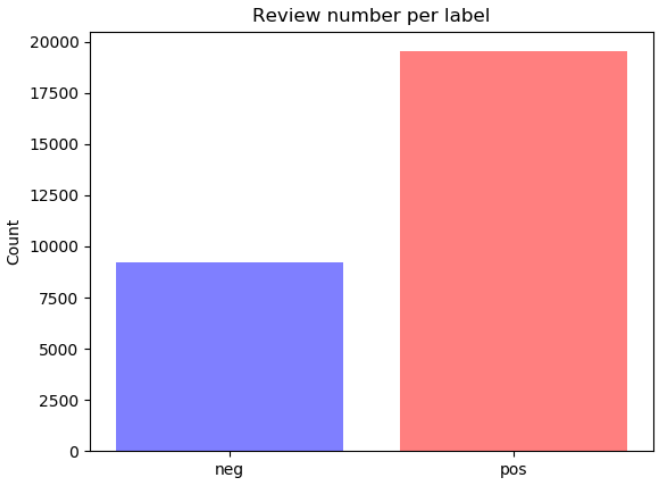
Exam session: winter 2020

The task of the competition was to perform a sentiment analysis based on a database of tripadvisor textual reviews. The model should perform a classification of every textual data in order to identify if it is a positive or negative review based on the words in them and the information that could be extracted.

1. Data exploration

A strong model for classification should be able to load and explore the data properly. The software performs these actions through the function load data, which memorize through a pandas DataFrame text, labels and indices. Except for the first line, which fields are saved as column names, for every row of the dataset I substitute all the missing values with the NaN value, in order to discard them through the dropna() function. I adopted the rejecting technique of the missing values because for this kind of data it would be impossible to fill them with other substitute values. Furthermore, the label is converted instantly in a numerical value: the “neg” value will be 0 and the “pos” value will be 1 (a dictionary will be used to re-convert the numbers into real labels).

It could be useful to visualize some information about the development part in order to understand and explore the data. First, I decided to print the relationship between the number of reviews and the labels, the chart below describes the dataset trend through bars, it was produced through the matplotlib.pyplot.bar.



It can be noticed that the labels of the reviews are more positive than negative, so for this asymmetric behaviour the expectation is to obtain more positive words than negative ones, especially when the reviews will be computed in the classification part.

Also, it could be useful to know which words are the most common in the reviews for analysis purposes. For this reason, printing word clouds of the development part could help us to study the data. We can clearly observe that the most common words are bigger (usually nouns), but there are also important smaller words which characterize the cloud as negative or positive.

We expect that the word clouds about the evaluation part will be similar.

Word cloud for negative reviews

Word cloud for positive reviews

2. Preprocessing

First, I begin initializing an object of StemTokenizer: this is a callable class that acts as tokenizer during the process. Indeed there is a Stemmer among the object attributes, this attribute uses the ItalianStemmer function from the ntlk.stem.Snowball library, which is properly intended for computing various languages, included the Italian one. The other attributes are regex expressions for digits, punctations and various emoji, The callable function tokenizes every review and for each word in it, it strips it and it deletes from it every emoji, punctation or number; words compounded by two or more punctation are accepted because the points of suspension and similar elements could be useful. Then the word is stemmed and memorized in a list that will be returned. This tokenizer is passed at the TfIdfVectorizer function, which has the task to extract

the features from the list of documents and transform them into a matrix of tfidf values; the tf-idf algorithm is implemented by the sklearn library.

The arguments, in addition to the tokenizer object, are:

- The max_tf and min_tf, which are used to ignore some too frequent or too rare words, both useless.
- The ngram parameter to search also the compositions of words, compounded by one, two or three different words.
- Other parameters which indicates that all the document must be converted in lower case without accents.

I decided to not remove any kind of stopword because they negatively affect the classification methods (Source: http://oro.open.ac.uk/40666/1/292_Paper.pdf). After the transformation I didn't reduced my matrix in order to build the model as precise as possible.

Finally, the matrix can be normalized through the Normalizer function from the sklearn.preprocessing library.

3. Algorithm choice

About the classification problem there are many algorithms to find a solution, these are the ones I considered:

- Decision Tree algorithm
- Random Forest algorithm
- XGB classifier
- Naïve Bayes algorithm
- SVM algorithm
- SGD Classifier

Decision Tree is the first algorithm I tried, primarily for its simplicity: indeed, it is the most inexpensive to construct and it can classify new unknown records and it shouldn't be affected by missing values because of the precautions in the exploring data part. However, its outcome is considerably lower than the other algorithms, probably it is due to the largeness of the dataset: the accuracy turns out to be about 0.8583.

Random Forest algorithm, as we know, is a classification algorithm derived by the decision tree one, since it is composed by a set of trees which are built together during the training phase. The accuracy turns out to be more precise than decision tree, since it is more reliable for large datasets, but its outcome remains still too low. Its number of trees is set through a grid search algorithm and it is 200 for the best accuracy of 0.9317 on the train set.

Since Random Forest works well with documents thanks to its highest and faster accuracy, I considered also the XGB classifier, which is a function from the XGBoost library, it is a sequential algorithm derived by the random forest one, where every tree is boosted. It returns a labelled result with an accuracy of 0.9282.

Naive Bayes algorithm is based on Bayes' theorem with the assumption of independence between every pair of features; even though it works good with documents and it is usually faster than the other algorithm, it turns out to be a bad estimator in this case, with an accuracy of 0.7860.

Support Vector Machine is one of the classifiers that works well, especially when the parameters make the SV Classifier non-linear and more flexible, giving an accuracy of 0.9639, but it takes a lot of time to return a solution.

SGD Classifier is finally the one that works better for the sentiment analysis, it implements regularized linear models with stochastic gradient descent learning. The tuning of the algorithm will be discussed in the next paragraph.

The accuracies are calculated in local with the F1 weighted method and the tests are performed with the 80% of the development set.

4. Tuning and validation

After choosing the most appropriate algorithm for the task, I performed some tuning in order to gain the maximum accuracy from the chosen model.

For what concerns the TfIdfVectorizer, after some tests, I figured out that with a max of 0.6 and a minimum of 5 presences I could extract enough features to reach an acceptable accuracy.

For the chosen classifier instead I used the GridSearchCV provided by the sklearn.model_selection library. This grid search allows me to try different parameters doing concurrently with a k-fold validation with 5 different folds for every running session. It returns that the best parameters are modified huber as loss, l2 penalty, alpha equal to 0.0001 and fit intercept equals to True, giving an accuracy of 0.973 (on the uploading site).

Finally, the function dump_to_file has the task to write a csv file with the indices and the predicted labels of the evaluation data. For a deep understanding of the predicted labels, it could be useful to print the same information I extract for the exploration of the development dataset.

Here is the word cloud obtained by the predicted data of the evaluation reviews. In this step a list of stopwords is used only to make the clouds clearer and more readable. We can notice the strict similarity with the previous word cloud I computed: the biggest words are obviously the most used, instead the smaller ones are more significant for our task.

In the negative word cloud it can be noticed that, as the development word cloud, most of the bigger words are nouns, but looking further there are many common Italian words used in negative context like “solo”, “male”, “troppo”, “vecchio” (probably it is used to describe the furniture), “nulla”, “peccato”. As we expected there are also an asymmetry due to the positive words.

Looking to the positive cloud instead we can simply identify the words or construction of them that leads to that label: words like “eccellente”, “ottimo”, “gentile”, “cordiale”, “molto bella”, “gentile”, “disponibile” and others appears in the cloud and they are the reason of the positive characterization of those reviews.