

Data Science Lab: Process and methods

Politecnico di Torino

Project report
Student ID: s267566

Exam session: Winter 2020

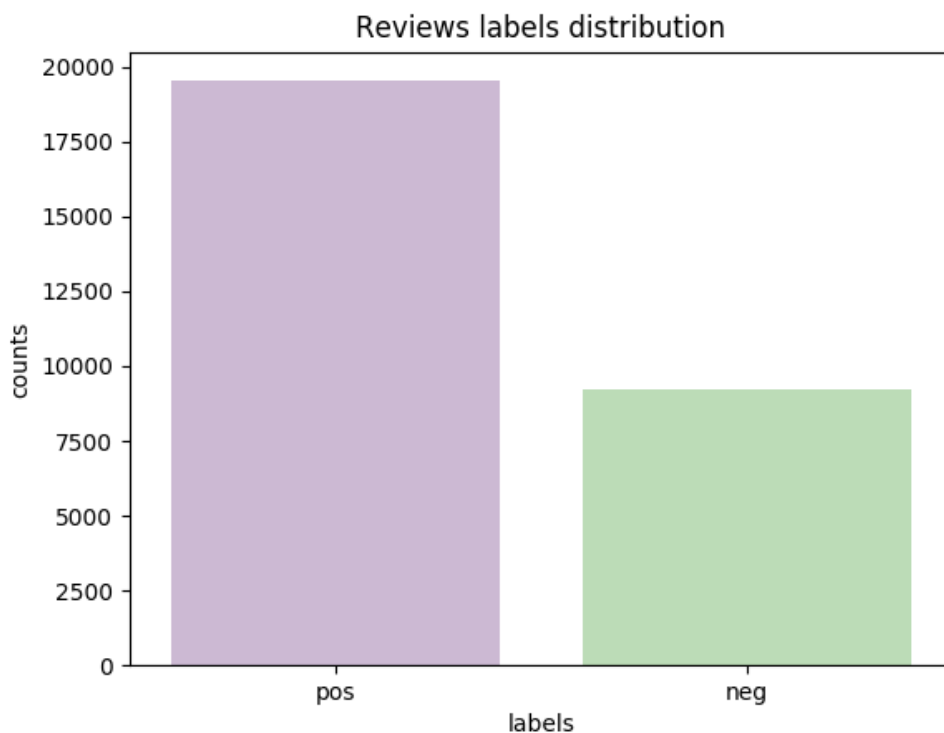
1. Data Exploration

The dataset used for this analysis contains 41.077 textual reviews taken from the TripAdvisor Italian site. The dataset is divided into two collections:

- 28.755 labeled reviews used for training
- 12.323 unlabeled reviews used for evaluation

Each review is labeled with the values “pos” or “neg”, which are encoded into 1 and 0 respectively.

An initial analysis allows understanding the distribution of the two labels within the dataset.

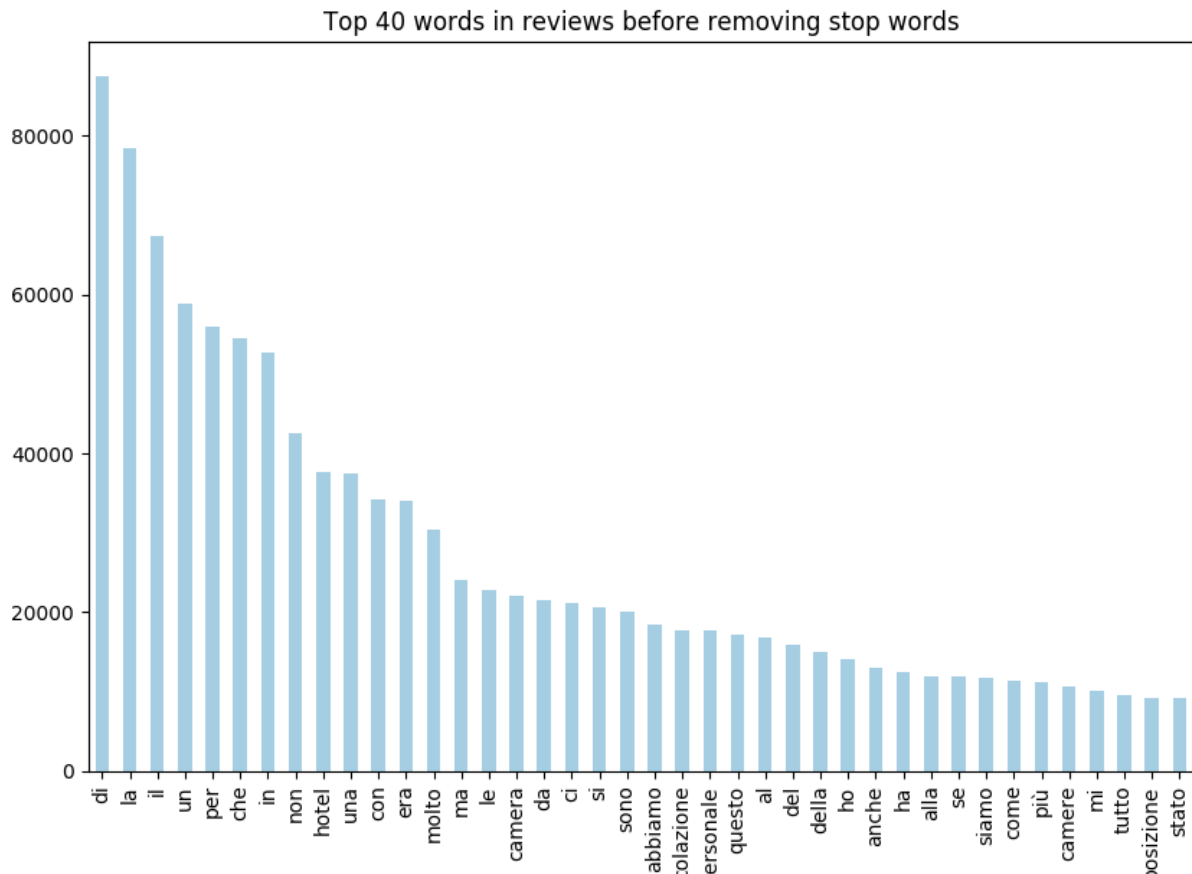


The above graph tells that the given dataset is an imbalanced one with very less amount of “neg” labels.

There are no missing values or wrong labels, all the data are textual, although some reviews contain particular characters - such as ideograms - that cannot be interpreted. Numbers are written both in digits and words and some of them are particularly important for the polarity of the sentence: for instance, numbers referring to distances, which express an evaluation on the quality of the accommodation or on the satisfaction/disappointment.

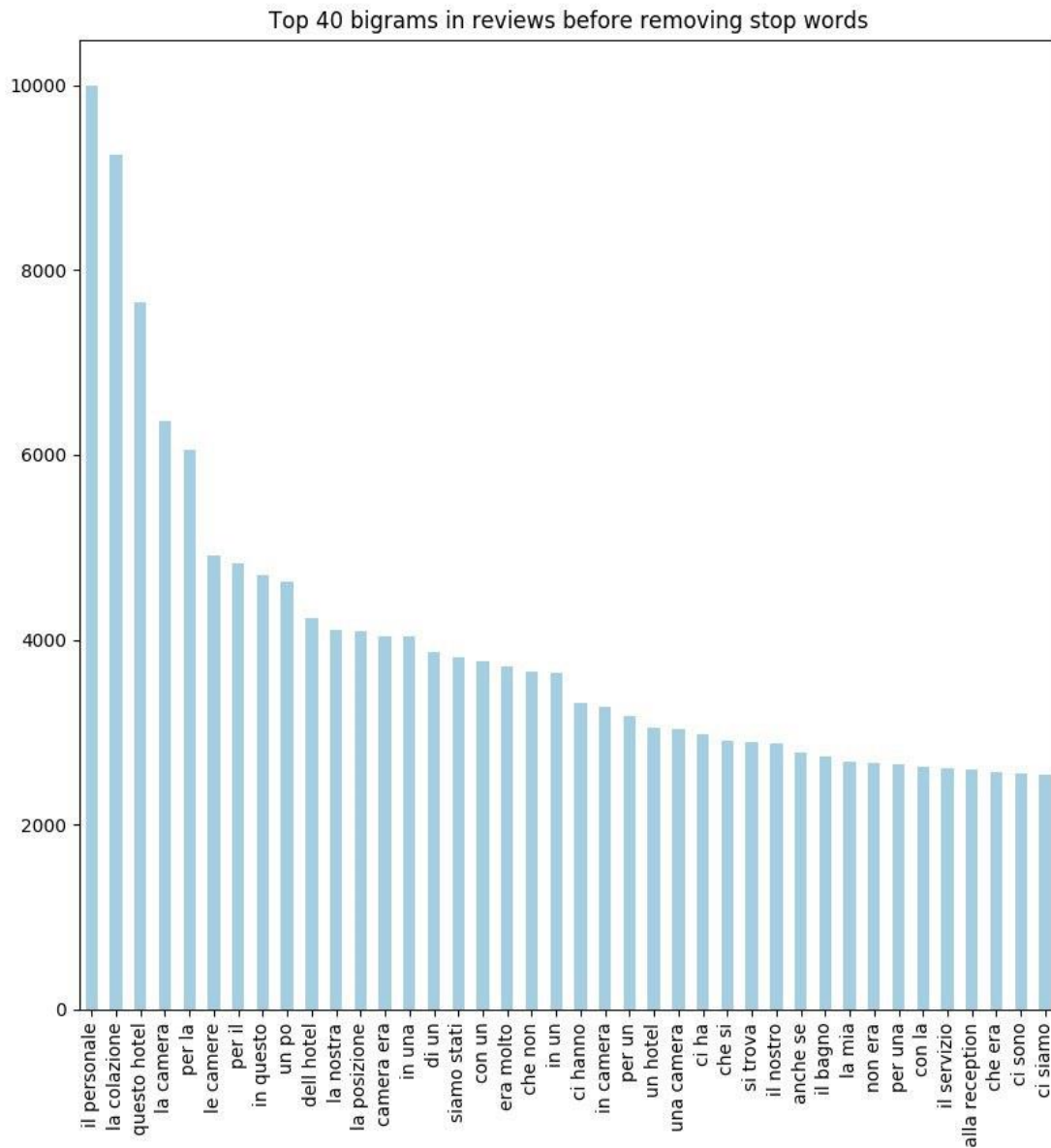
It is easy to understand that not all reviews have been written by native speakers. Many of these have grammatical or typing errors or excessive repetition of vowels and punctuation. Since some of them are poorly written translations, they are sometimes difficult to understand even for a human being. Also some reviews can be considered neutral rather than completely positive or negative.

Further informations can be derived from more detailed textual data analysis. The graph below shows the 40 most frequent words in the reviews and it seems quite obvious that they are completely irrelevant since they don't have connections with sentiments.



Although single words are useful, they provide limited information for the given task. Couple of words can capture contextual information and eliminate ambiguity in the sentiment analysis of the review – for instance, *clean* is different from *not clean*.

However, even in the case of word pairs, the most frequent ones detected before the preprocessing phase are useless and irrelevant, as can be seen in the graph below.



The graphs show the need to perform a thorough data cleaning to extract the most useful and important features from the dataset.

2. Preprocessing

Before training the model, some preliminary steps must be performed to clean the data:

- Up-sampling
- Dataset split (80% training, 20% validation)
- Removal of punctuation
- Tokenization
- Numbers to words conversion
- Lemmatization
- Uni-gram and bi-gram vectorization
- Removal of stop words
- TF-IDF transformation

Since dealing with imbalanced data may be tricky, first of all an up-sample of the minority class is made. As noted in the initial graph, the negative class has less than half the samples of the positive one. Due to this, all the negative reviews are considered twice to rebalance the total distribution.

Each review is converted in a numeric representation by means of the **TfidfVectorizer** class from **scikit-learn**, with a custom **Tokenizer** object. All the punctuation is removed and reviews are split into smaller parts called *tokens* – each of which is cleaned separately – through the **word_tokenize** function.

Since numbers are important, a dictionary is created to convert the important ones written in digits into words. The conversion is not performed out on all numbers, but only on those that fall within a range that could be of particular interest.

After the tokenization, words are lemmatized through the **treetaggerwrapper** class, which supports Italian language. The lemmatization process returns the *lemma* of the word – its dictionary form. This operation is slower than stemming, but it should perform better since it operates on words with knowledge of the context. However, no major changes are noted in the choice of one of the two types of normalization. The library **treetaggerwrapper** also gives the part of speech (POS) for each word as a result. This tool can be very useful for sentiment analysis, however accuracy worsened a lot by eliminating some POSs such as verbs. For this reason, it was decided to consider every possible part of speech.

Stopwords are taken from the **nltk** library. This list is extended with other stopwords found in the tuning phase of the algorithm. The adverb “*non*” is removed from the list since it is important to better understand the real meaning of some sentences.

The last step is the representation of words in numeric form. For this purpose, the Tf-Idf representation is used. Features within a range defined by *min_df* and *max_df* are chosen and the optimal values for these parameter are tuned through the sparse matrix analysis and the IDF weights associated to each feature.

3. Algorithm choice

After the feature selection steps, the text is represented in a form that can be used to train an algorithm. There are many classifiers tested and recommended for sentiment analysis purposes. The ones I have chosen as potential ones are Multinomial Naïve Bayes, Random Forest, Stochastic Gradient Descent and Support-Vector Machine.

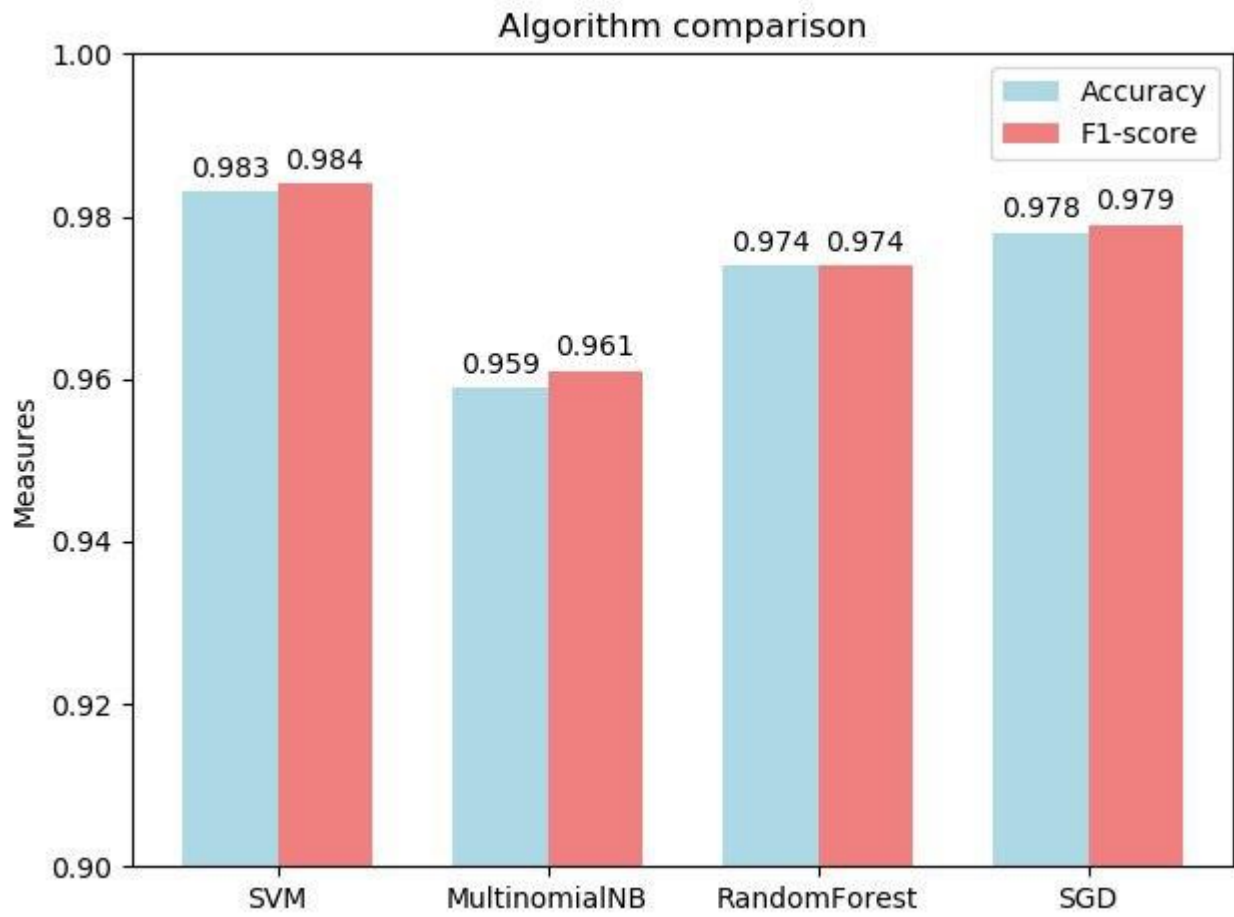
Multinomial Naïve Bayes is a baseline for sentiment analysis and it is the easiest and fastest classifier among those mentioned above. It cares only about the appearance of words and it does not consider their frequency, which is a double-edged sword. For instance, if words like “beautiful” or “amazing” appear in a document, their frequency is not completely relevant, since the polarity of the review is pretty clear. On the other hand, if some words in the training set appear only in one class, the classifier will classify text to that particular class, even if those words are not related to polarity. Moreover, it is based on the strong assumption that features are independent, which is not always true.

Stochastic Gradient Descent is a Linear classifier with SGD training. Its major advantage is the efficiency and its simplicity of implementation. On the other hand, it is sensitive to feature scaling and it is difficult to find the best hyperparameter for this model.

Random Forest Classifier is one of the most known machine learning algorithm since it is easy to interpret, fast and scalable. However, although it is used in text classification and NLP, it does not perform well on this specific task as the results provided are not among the best.

Support Vector Machine identifies the optimal hyperplane which maximizes the margin and separates the classes in the best way. It works well on high dimensional datasets, even though its training time can be high. It gives better performances than the other classifiers in terms of accuracy, f-score and recall. Unlike the Naïve Bayes classifier, it looks at the interaction between features and captures it to perform the classification task better. It is chosen since it outperforms the other algorithms considered and it gives good results for both classes.

The graph below shows a comparison between the scores of the classifiers. These results are obtained after a tuning phase, in which the optimal hyperparameters – among those specified – are selected for each algorithm.



4. Tuning and validation

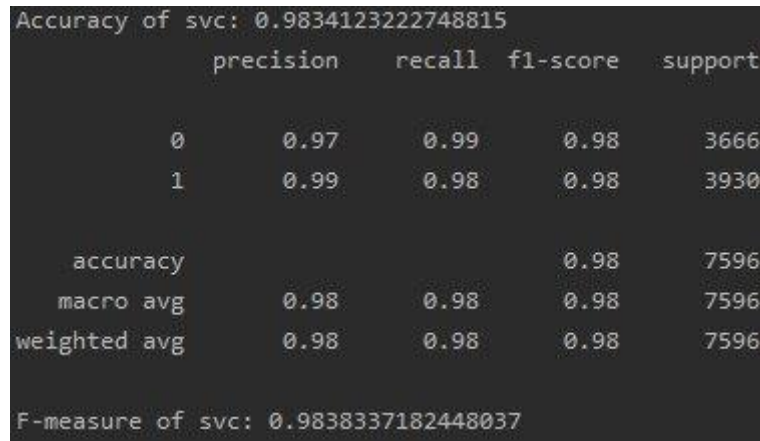
Each algorithm is tuned using `GridSearchCV` in order to search for the best hyperparameters and maximize the performances of these models.

`GridSearch` performs a 5-fold Cross-Validation, so the training set is divided into 5 sets and the algorithm is trained and tested 5 times. The cross-validation procedure can prevent the overfitting problem and improve the accuracy of the model since it uses multiple and different subsets of data.

A different parameter grid is set for each classifier specified above. In particular, for the SVM algorithm two types of kernel are considered: linear and RBF. Results show that RBF kernel performs better than the linear one, since it can handle the case when the relation between class labels and features is nonlinear. There are two parameters to set for an RBF kernel: C and γ . The best values found for this model are $C = 10$ and $\gamma = 1$. After the best hyperparameters are found, the final classifier is generated.

In the end, it can be seen that the accuracy of different versions of SVM is always higher than the accuracy of the different versions of other algorithms.

The following are the results obtained with these hyperparameters:



```
Accuracy of svc: 0.9834123222748815
      precision    recall  f1-score   support

     0       0.97       0.99       0.98       3666
     1       0.99       0.98       0.98       3930

 accuracy                   0.98       7596
 macro avg       0.98       0.98       0.98       7596
weighted avg       0.98       0.98       0.98       7596

F-measure of svc: 0.9838337182448037
```

An effective tool to validate the model is the WordCloud. This is used to better understand the predominant features for each class and to tune the `min_df` and `max_df` parameters in a better way. These graphs are also useful to find new elements to add to the stopwords' list, since they help to recognize frequent but pointless words.

Wordcloud: 0 dataset evaluation



Wordcloud: 1 dataset evaluation

