Data Science Lab: Process and methods

Politecnico di Torino

Project report
Student ID: s278936

Exam session: Winter 2020

1. Data exploration

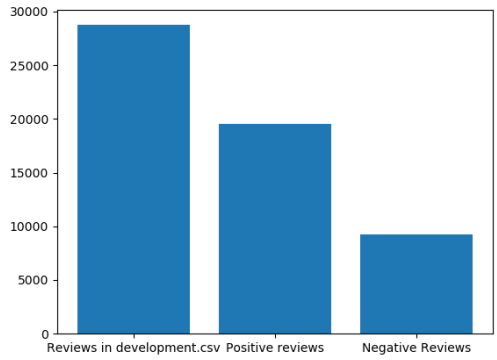
This report describes the pipeline used to carry out a text sentiment analysis on reviews collected in the Italian TripAdvisor website [1].
The first step of the analysis consisted of importing data from `development.csv` and `evaluation.csv` file.

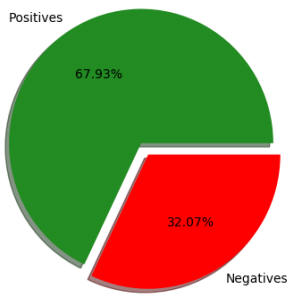
By reading the files, it was possible to notice that the **41077** textual reviews contained into the dataset were so split:

- **28754** in `development.csv`
- **12323** in `evaluation.csv`

In order to build a good classification model, the focus was directed toward the content of `development.csv`. Analyzing better this file, it was possible to extract other important statistics like:

- The number of positive reviews: **19532** (67.93%)
- The number of negative reviews: **9222** (32.07%)





As shown by plots, classes are not represented equally, so this dataset is quite imbalanced, but in this preliminary phase neither *undersample* to majority class nor *oversample* technique to minority class have been applied. In this sense, an overall

measure weighted (*f1_weighted*) by class support will be mainly considered during the evaluation of the model, assuming that both classes have the same importance.

Some other statistics have been computed by processing the content of each review in `development.csv`: it was observed that the overall number of words is over **20** million, **153276** of them are different. Further information about the words in the reviews are collected in the table below.

	Overall	Positive	Negative
Mean	701	624	864
Standard Deviation	606	513	740
Median	515	465	644
Minimum	58	58	67
Maximum	9153	7800	9153

From these results we can see that the starting amount of data is very huge, then it would be necessary effecting an efficient preprocessing phase before proceeding to evaluate a classification model.

2. Preprocessing

Document processing

It consisted of sequential operations that have allowed us to generate a structured data representation.

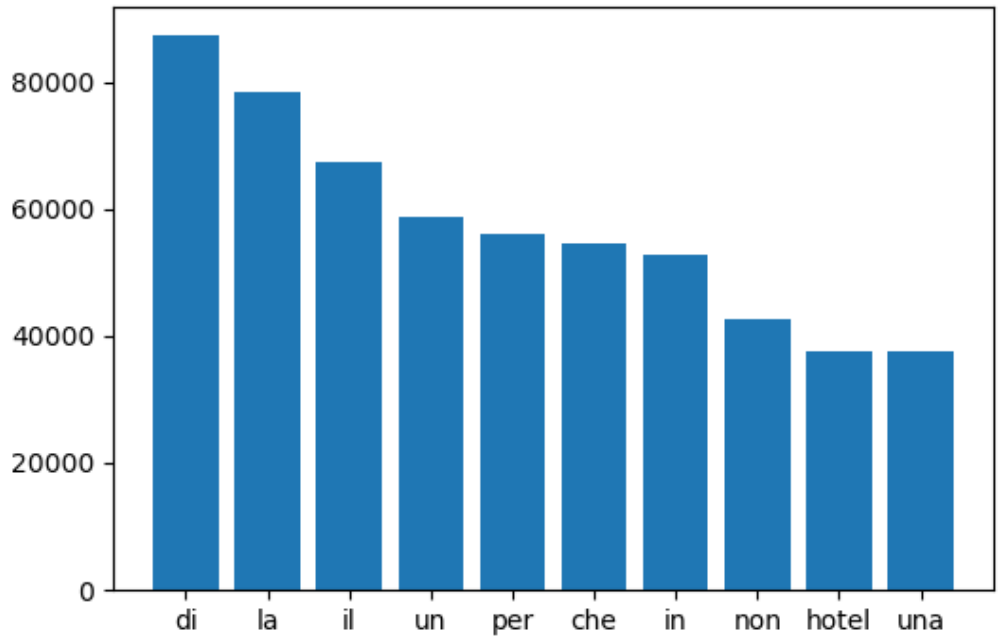
Firstly, the following operations are performed on each textual review:

- Removal of all the numbers;
- Removal of all the foreign characters and words;
- Removal of all the special characters;
- Removal of punctuation and newlines;
- Removal of all single characters;
- Removal of whitespaces;
- Conversion of all words to lowercase (Case normalization).

.

After these operations, the number of different words in `development.csv` was **52890** (about a third of the original set).

10 most common words in `development.csv`



As it can be seen from graph, there were still useless terms: indeed, most common words in `development.csv` are mostly articles and prepositions that certainly were not be able to provide useful indications about a negative or positive feeling that could direct correctly the sentiment analysis.

A preprocessing phase improvement was also performed by ***document splitting***, ***tokenization*** and ***stemming***.
In order to implement these, some *Natural Language Toolkit* (*NLTK*) libraries were imported:

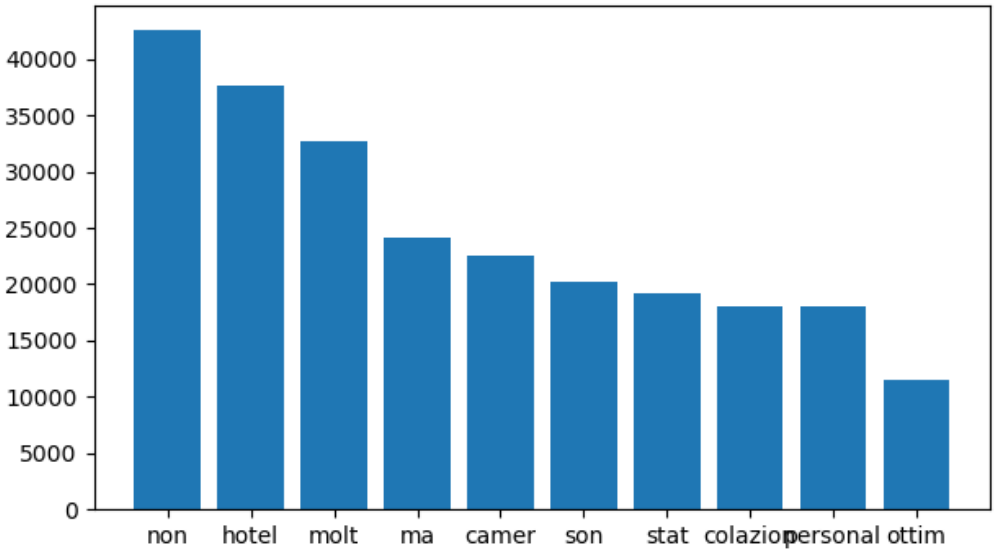
- **Nltk.tokenize [2]**: this suite provides *word_tokenize* method, that returns a tokenized copy of text, using *NLTK's* recommended word tokenizer:

- **Nltk.stem.snowball [3]**: this suite provides us the *ItalianStemmer* method, which reduces Italian words to their root form, removing prefixes, suffixes and pluralization

- **Nltk.corpus [4]**: this library contains a method that provides us a list of stopwords from many languages. From this default list of words, I removed some items that would be useful for the data mining model:

- "*non*", an adverb with a negative meaning (indeed it is the 5th word most used in negative reviews);
- "ma", an adversative conjunction;
- "sono" an Italian form of the verb "to be";
- "contro", a negative preposition.

10 most common token



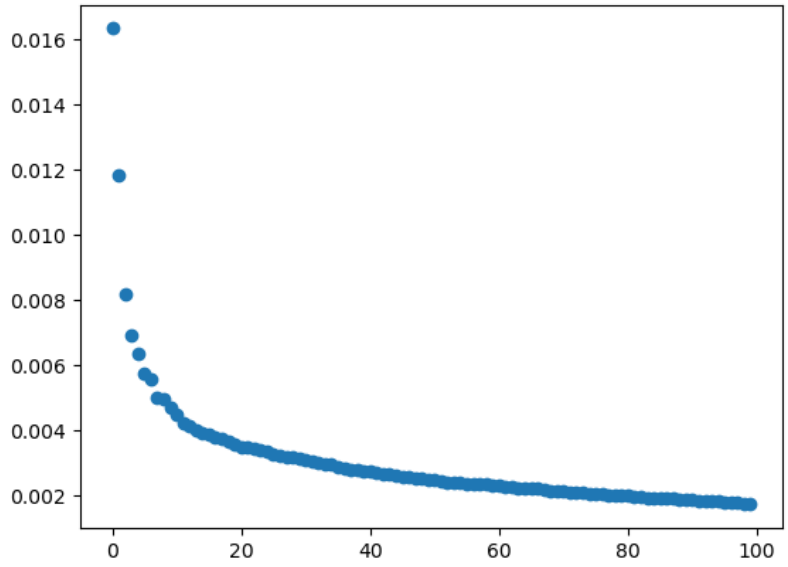
After the described operations, the total number of different tokens was equal to **27207** (less than a fifth of the original set).

Tf-idf Vectorizer

Another important step consisted of calculating ***tf-idf*** (*term frequency–inverse document frequency*) parameter through *TfidfVectorizer* function from *sklearn* library. This weighting scheme measured the importance of each term with respect to the dataset.

Data reduction

In order to speed up the research of best Classification model it was fundamental using *Principal Component Analysis* (***PCA***) to reduce the size of attributes. In this solution is adopted the specific case of *Incremental principal component analysis* (***IPCA***) [5], a technique used when the dataset to be decomposed is too large to fit in memory.



From this picture, it is possible to see that the suitable number of kept components after applying ***IPCA*** w 10.

3. Algorithm choice

Starting from results obtained from the previous steps, it was possible to build a robust model able to predict the belonging class of data coming from an evaluation set.

In order to find a solution, it is compulsory choosing one of classification techniques able to handle a linear text processing problem like this.

Basing on this premise, classification algorithm considered were:

- Decision Tree Classifier
- Random Forest Classifier
- Stochastic Gradient Descent (*SGDClassifier* [6])
- Support Vector Machine (*LinearSVC Classifier* [7])

The decision of including SGDClassifier is dictated from its simple and efficient approach to solve linear text classification problems like this.

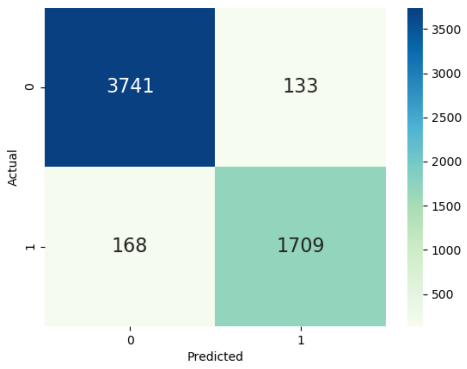
Criterion that was used to choose the model was the *accuracy* and the *f1_weighted* score: indeed, these parameters are the index of the quality of the prediction.

For the computation of this value, the dataset was split into training set and evaluation set, limiting the effect of the unbalanced dataset through the cross-validation approach: more specifically, it was used the *KFold* method that consists of dividing data into *k* disjoint subsets. Here, *k* was set equal to 10.

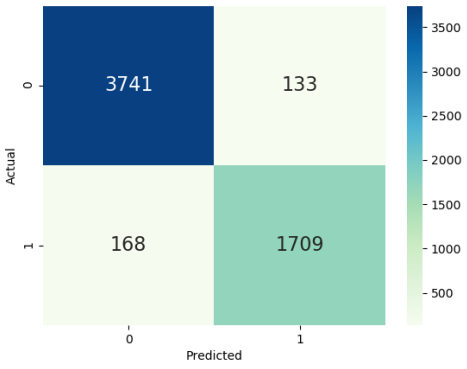
The selection of the best model was done experimentally by using the different parameters. Results obtained are summarized in the table below:

Classifier	**Accuracy**	**F1_weighted**
Decision Tree	0.8382	0.8603
Random Forest	0.9287	0.9412
SGD Classifier	0.9460	0.9596
Linear SVC	0.9476	0.9681

According to these results, it was possible to notice that most accurate models were ***LinearSVC*** and ***SGDClassifier***, that provided the following confusion matrixes:



LinearSVC confusion matrix



SGDClassifier confusion matrix

4. Tuning and validation

After the previous analysis, the *LinearSVC* classification has been chosen for building the model and making predictions from the evaluation set, since it showed the highest accuracy and the highest robustness to handle this high-dimensional problem.

To improve the chosen algorithm is used the *GridSearchCV* function, that found the best configuration of parameters testing all possible combinations from a dictionary of parameters called *param_grid*.
Discovery of best model setting was made using the dataset reduced by ***IPCA***, in order to speed up the process and preserving a good quality of the process.

At the end of the procedure, the best configuration for *LinearSVC* model has been obtained by tuning parameters as shown below:

- *C*=5;
- *class_weight*=None
- *dual*=True
- *fit_intercept*=True
- *intercept_scaling*=1
- *loss*='squared_hinge'
- *max_iter*=1000,
- *multi_class*='ovr'
- *penalty*='l2'
- *random_state*=None
- *tol*=1e-11
- *verbose*=0

This model setting has allowed to obtain a **0.9681** *f1_weighted* score on test set.

5. References

[1] Dataset from tripadvisor.it Italian web site (https://www.tripadvisor.it/)
[2] NLTK.tokenizer (https://www.nltk.org/api/nltk.tokenize.html)
[3] NLTK.stem.snowball Italian Stemmer (https://www.nltk.org/api/nltk.stem.html#nltk.stem.snowball.ItalianStemmer)
[4] Nltk.corpus stopwords (https://www.nltk.org/api/nltk.corpus.html)
[5] Incremental PCA (https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.IncrementalPCA.html)
[6] sklearn.linear_model.SGDClassifier(https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html)
[7] LinearSVC (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html)