DSL ASSIGNMENT REPORT

Sentiment analysis classification on TripAdvisor reviews

Frattin Fabio
Politecnico di Torino
272785

1. DATA EXPLORATION

First, a null-value check is performed and since there isn't any null-value nor in the *text* field, neither in the *class* one, we can conclude that our development dataset is quite clean. A further analysis was done separately on each field to confirm this assumption:

- ***class**:* since this is our target value, we are interested in its distribution in our development set. The analysis shows that there is roughly 68% of positive labelled reviews and 32% of negative, thus revealing a **1:2** imbalanced proportion.
  Even if the dataset is not balanced, since I wasn't able to find an external source of compatible data to increase the number of negative reviews, I decided not to *under-sample* on the positive neither to *over-sample* (by generating duplicates) on the negative: both this techniques are quite extreme and are actually useful when the proportion is much more imbalanced than in our case.
  Taking the right precautions while developing the classification algorithm could be enough to produce a not too biased classifier.

- ***text**:* even if no null-value has been detected, there may be some reviews whose content isn't suitable for our analysis, for instance if it is composed of a few characters. Moreover, it may be interesting to understand if the length of the reviews is somehow correlated to the sentiment they encode: it might make sense to suppose that usually positive reviews aren't as large as the negative ones.
  The minimum length turned to be 58, thus showing that there isn't any empty review.
  The distribution of the logarithm of the length turned to be almost normal, but unfortunately the comparison based on the sentiment wasn't successful: the negative ones have a slightly bigger mean, but the two "normal bells" are way too much overlapped for this feature to have a discriminatory power.

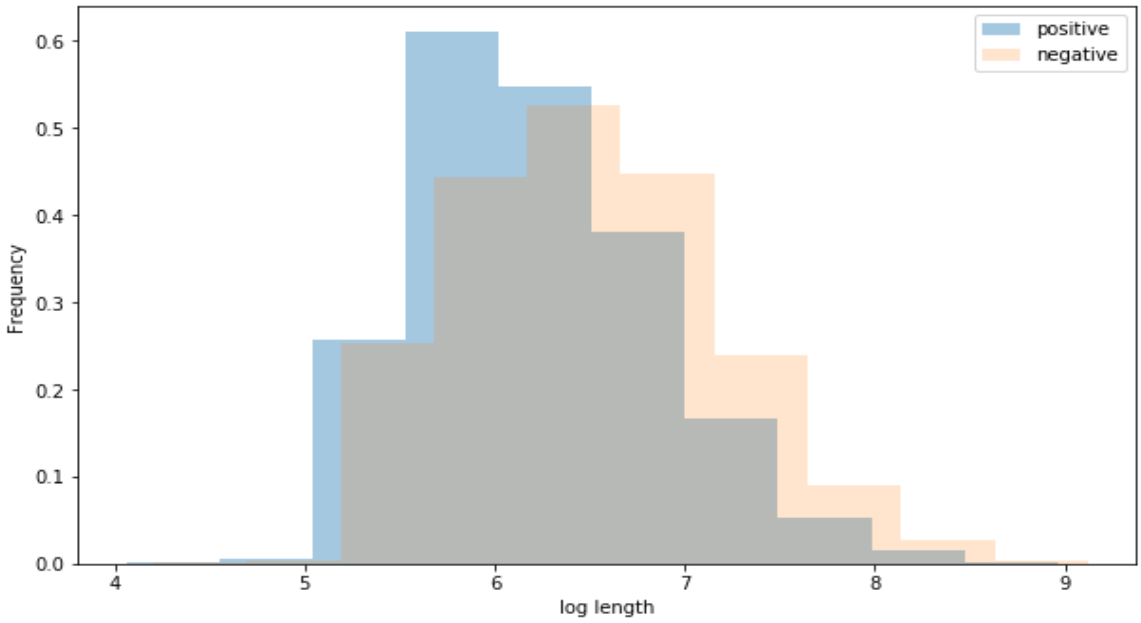

Fig.1 – Log-length Distribution for positive and negative reviews

2. PREPROCESSING

Dealing with raw text data, we need to operate some transformation to extract the features that will later be used in the classification process. This purpose is achieved by performing two consecutive steps: stem-tokenization and vectorization.

1. ***Text stem-tokenization:*** our aim is to generate, from each review, a series of tokens that we think may be useful to characterize its sentiment. We first need to filter out everything that could create some noise, such as punctuation, typos and all those words that are very likely to be repeated in many documents, thus giving no additional information. The *nltk library* already provide us with a list of Italian stopwords, basically containing prepositions and auxiliary verbs, which I extended step by step by adding words that, during the steps of the classification, turned to have an high frequency both in positive and in negative reviews (such as *"albergo", "hotel", "camera"*). After this filtering process, each token is stemmed in order to group all the words having the same stem and thus basically giving the same inflection to the sentence.

2. ***Text vectorization:*** this process creates the features and assigns, for each of them, a numerical value based on the content of the review. By using a CountVectorizer, the value assigned to each feature is the frequency of that term in the review. This method was chosen over the TF-IDF because the collection of documents under analysis is quite homogeneous and we are not really interested in giving low values to common terms since we only have two classes and thus the DF of many important word is likely to be high (e.g. most of the positive reviews are likely to contain the word *"buono"*). It was also exploited the ability of the vectorizer to generate *bi grams* by combining the input tokens: this is fundamental for a correct sentiment detection since an adverb or a negative preposition in front of an adjective can completely turn or enhance its inflection.

Fig.2 – Wordcloud for positive reviews

From the above word cloud, representing the most important words for the positive reviews, we can easily spot stems coming from words like *“ottimo”, “buono”, “disponibile”, “perfetto”,* hence concluding that the preprocessing phase has been correctly performed.
The outcome of the preprocessing is a matrix with roughly 15000 features: even if this number seems quite high, a dimensionality reduction only resulted in worse scoring and slower execution, thus wasn’t performed.

3. ALGORITHM CHOICE

The classification method chosen is the **Gradient Boosting** based on decision trees**.** This method can be considered an enhanced version of a **Random Forest**, where the trees composing the forest aren't independent each other but they are indeed built consecutively, each depending on the previous ones.

The idea at the base of the boosting framework is that each base classifier (the single *weak learner*, in our case the decision tree) is trained on a weighted form of the training set (either the whole set or a subsample). Those weights depend on the performance that the previous base classifier has achieved, measured in terms of distance from the true value. This process aims at making a sort of corrections of the wrong decisions taken by the previous tree. Once all base classifiers have been trained, they are combined to create the final classifier (the *strong learner*).

The things that mostly drove my decision towards this type of classifier are:

- expressiveness and interpretation, being based on decision trees, from which we can get the feature importance, and some relatively easy to understand math behind their construction;
- my familiarity with the idea of decision tree and forest;
- comparable performances in terms of *f1-score* for binary classification with other more complex, but less explanatory, classifiers such as NNs;

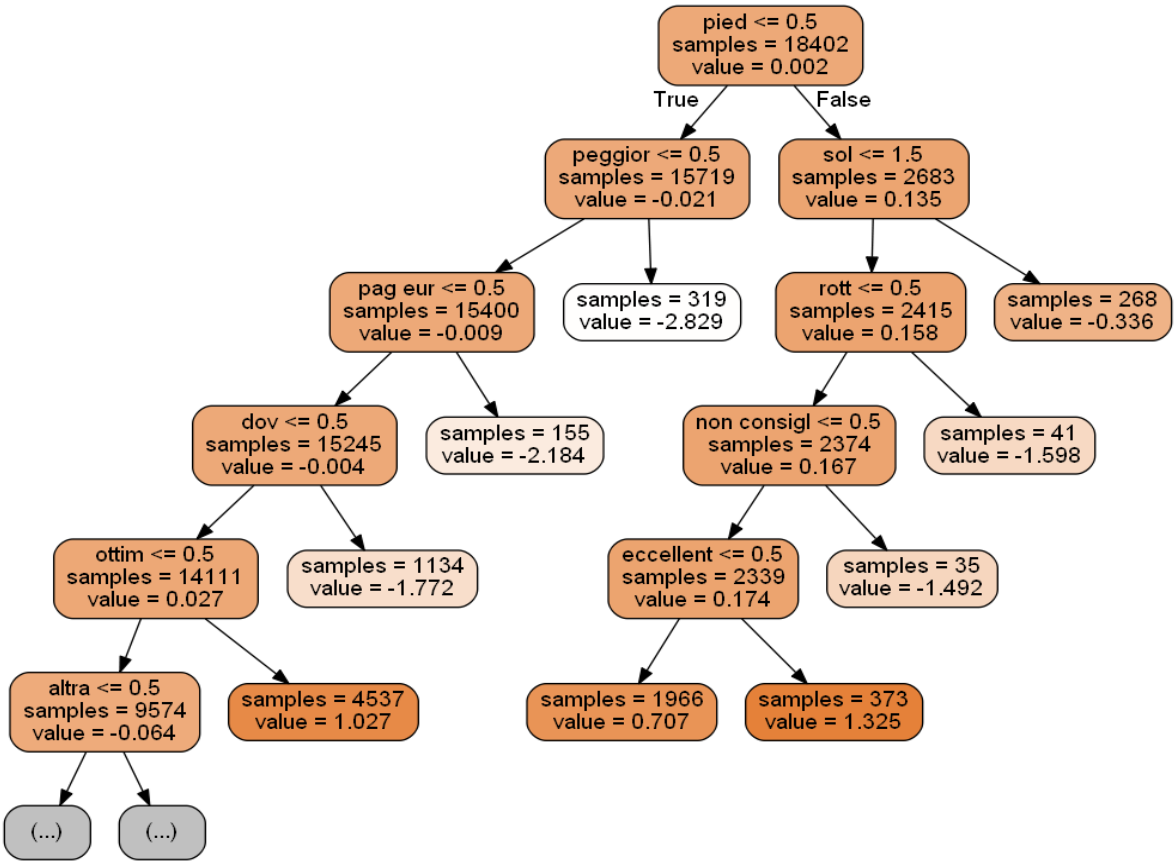

Fig.3 – Example of decision tree extracted from the trained classifier

The above represented tree shows how splits are made, revealing that most of them are based on the presence or absence of a single word in the review (the splitting value is **0.5**), suggesting that the count vectorizer previously used could be replaced by a bag-of-words representation with comparable outcomes. We can immediately spot how splits based on words like *"ottimo"* and *"non consigliato"* are very effective. The criterion used to operate the splits is based on the minimization of a function similar to the MSE, the *Friedman MSE*.

As already mentioned, the Gradient Boosting Classifier allows to understand which are the main reasons that will drive a future prediction. For example, we can inspect the 20 most important features (i.e. the ones operating the most effective splits), along with their importance.

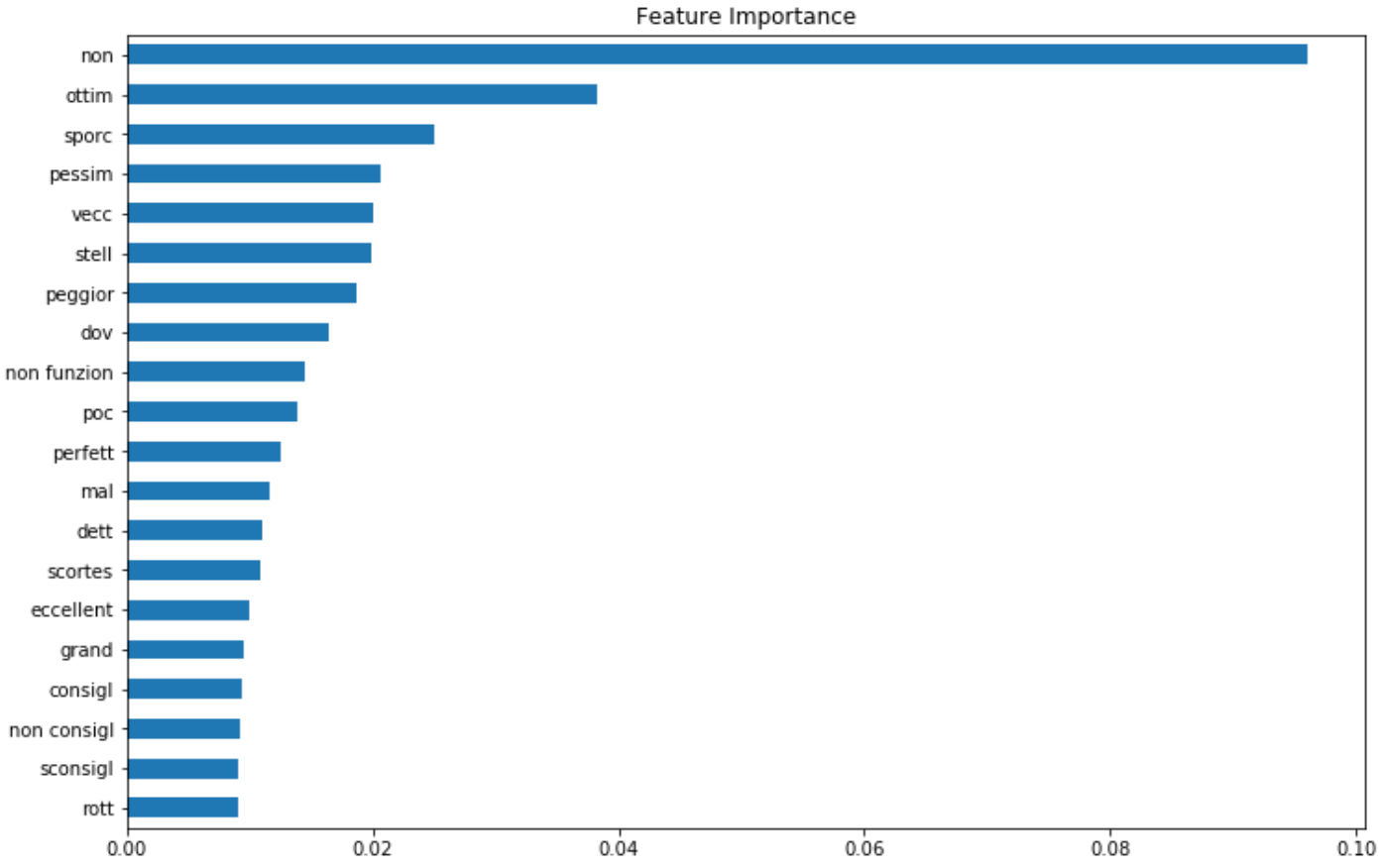

Fig.4 – Bar chart of the feature importance

As we would expect, among the most important words we find the absolute superlatives of the word *"buono"* and *"cattivo",* accompanied by other very polarizing words for what concerns the sentiment of a sentence.

4. TUNING AND VALIDATION

Even though there are several parameters that could be tuned in a Gradient Boosting Classifier, tuning them all would have meant waiting for ages for the *Grid Search* to run. Therefore, I decided to keep some of them fixed to values supposed to perform better on average, focusing only on those whose setting can really affect the nature of the classifier.

- ***learning rate:*** shrinks the contribution of each tree to the final decision;
- ***n_estimators***: the number of trees to be built, each one corresponding to a boosting stage. Unlike Random Forest classifier, the Gradient Boosting is quite resistant to overfitting, thus allowing much higher number of trees;
- ***subsample***: the proportion of samples that each base classifier is fitted on;

	Candidates	Best
learning rate	0.025, 0.05, 0.1	0.1
n_estimators	600, 800, 1000	800
subsample	0.8, 0.9, 0.95	0.8

Tab.1 – Results of the grid search

To better understand the results such a classifier achieved on the validation set, a 2d and 3d scatterplots, obtained by reducing the number of dimensions through the SVD, can come in handy:

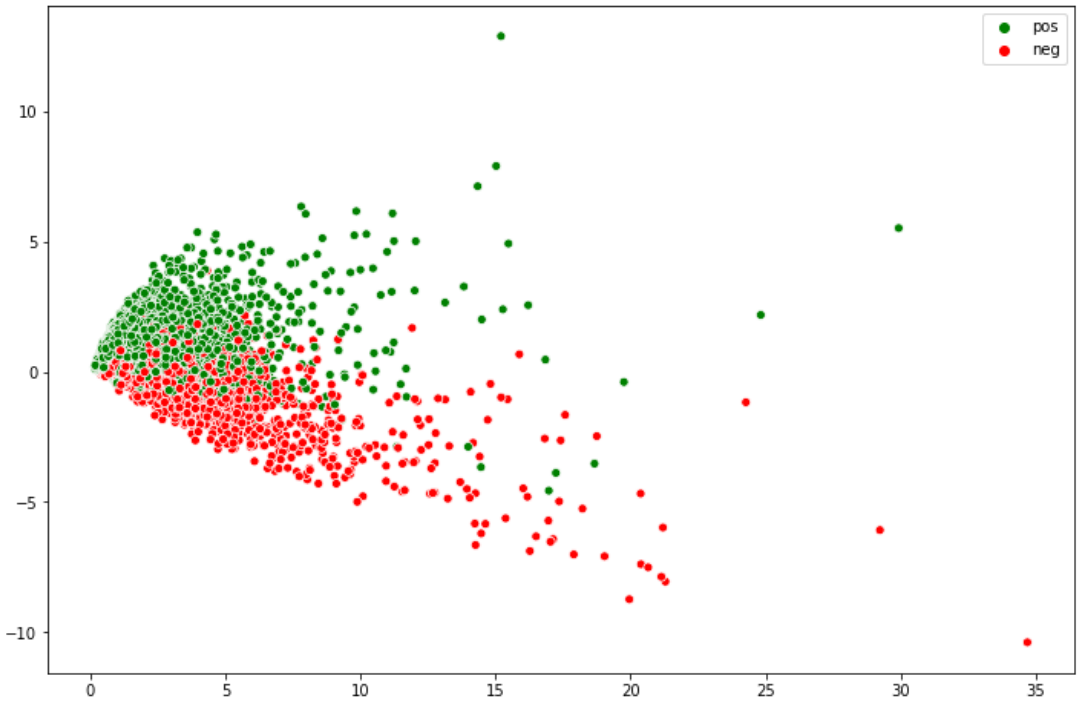

Fig.5 – 2d scatter plot

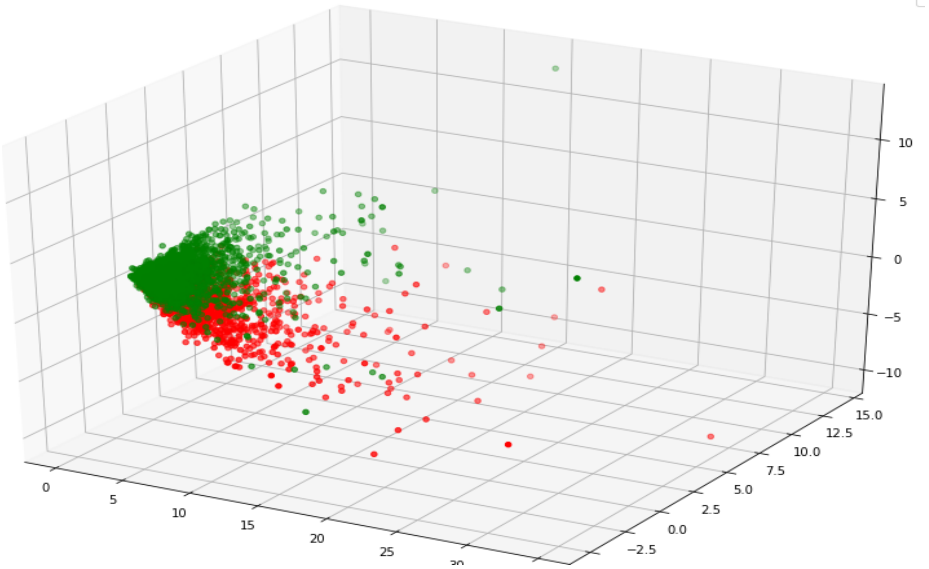

Fig.6 – 3d scatter plot

From a visual point of view, it looks like the classifier has correctly predicted most of the review's class, since all the negative and all the positive reviews are quite well "clustered". Yet, it is not easy to find a precise line (or plane) that clearly divides the two areas: this may be due to the fact that a binary sentiment classification can sometimes be simplistic, and some reviews can be hard to classify even for a human.
Since we want a numerical metric to evaluate the goodness and the weak points of the classification process, let's inspect the confusion matrix and the classification report:

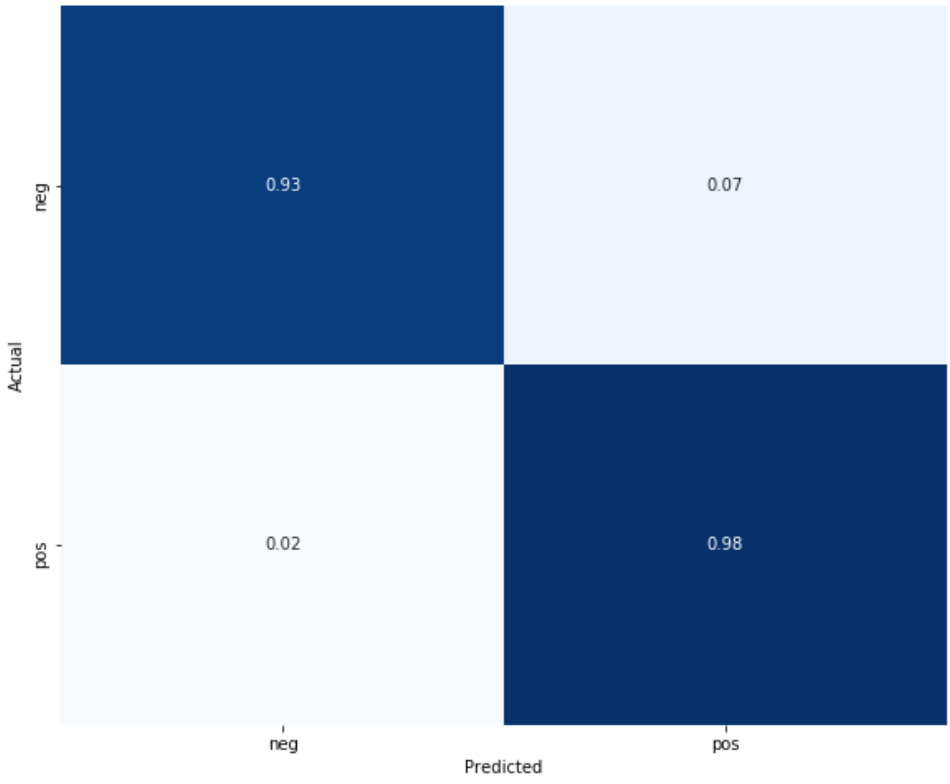

Fig.7 – confusion matrix

	precision	recall	f1-score	support
neg	0.95	0.93	0.94	1844
pos	0.97	0.97	0.97	3907
accuracy			0.96	5751
macro_avg	0.96	0.95	0.96	5721
weightred_avg	0.96	0.96	0.96	5721

Tab.2 – classification report

The metric we are interested in, the *f1-weighted*, has a value of **0.96,** very high thanks to the cleanness of the data and the simplicity of the problem (a positive and a negative review are supposed to be quite different, it would have been much harder to classify, for example, positive from neutral).

The difference between the values obtained on each class separately reflects what was previously discussed about the displacement of the dataset in favor of the positive class: both the precision and the recall are slightly higher than their corresponding of the negative one, making the classifier a little bit biased.