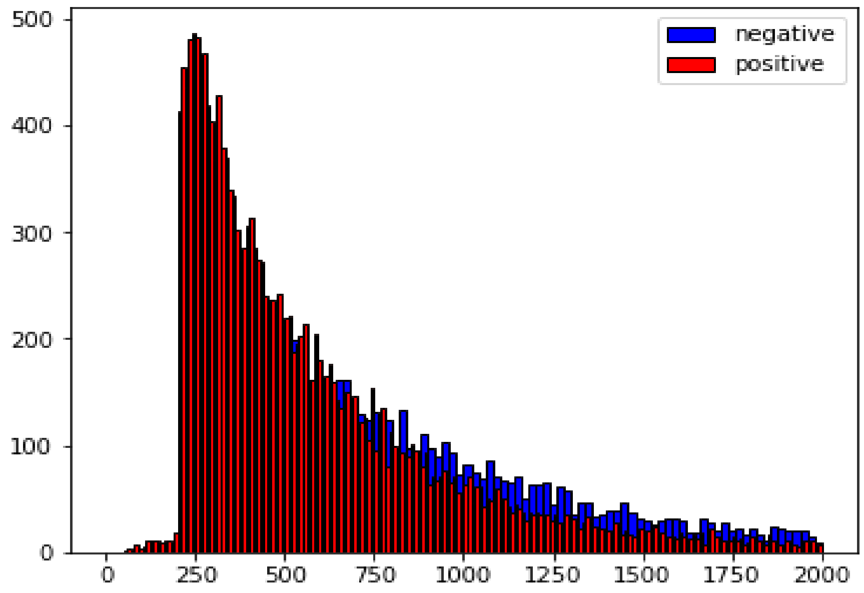
Data Science Lab: Process and methods

Politecnico di Torino
project report
Student ID: s277959
Marco Di Nepi

1. Data Exploration

The dataset provided contains 41077 reviews: 28754 are from the development set, where each data point comes with a label 'pos' and ‘neg’ for positive and negative review respectively, and 12323 are from the evaluation set. The goal is to predict the correct labels for the second dataset.
Once the data is saved in a padas dataframe it is possible to analyze the distribution.

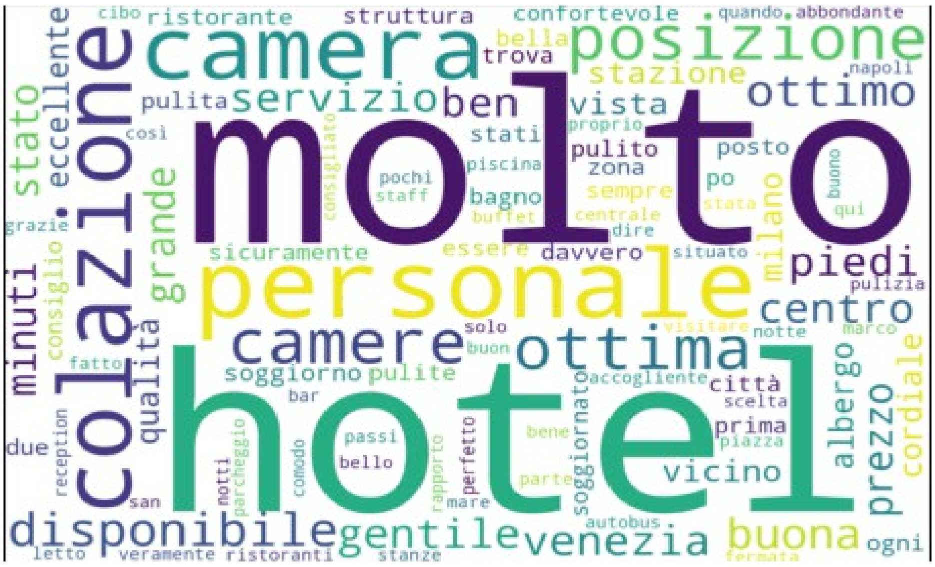
In the development set there are no rows with null values and there are 19532 positive reviews and 9222 negative reviews. Since almost 68% of the reviews are positive, the dataset is unbalanced. By looking at the length of the text for each review it can be also noticed that the negative ones are on average slightly longer than the positive ones: the negative reviews are on average 824.2 characters long while the positive ones 624.5 characters long.



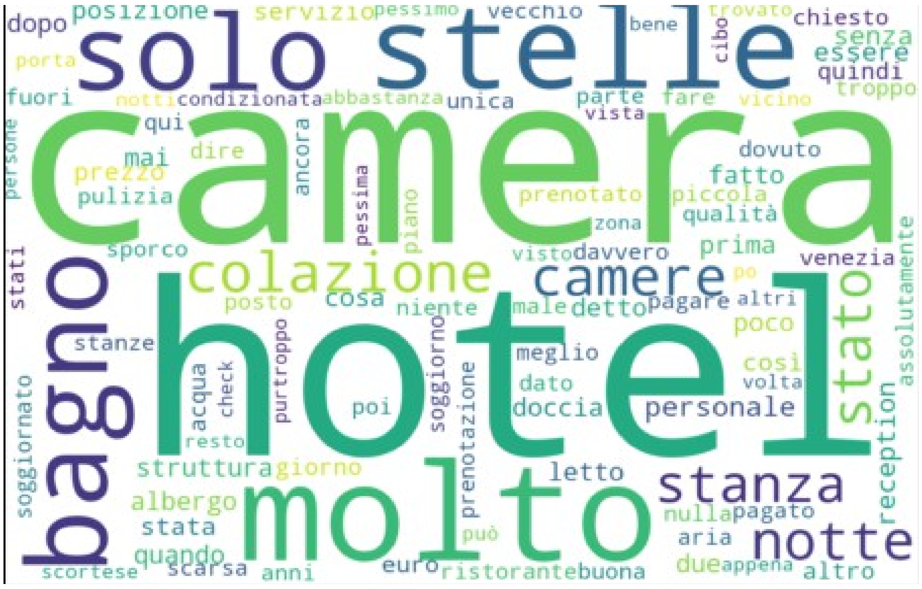
Furthermore, it can be noticed that the reviews, notwithstanding them coming from the Italian tripadvisor database, are most likely written by foreigners and translated using an automatic translation service. Many sentences in fact are not composed in correct Italian, and English (and even chinese) words can be found every now and then. However, as we will see in a short while, this will not affect the accuracy of the model considerably.

By exploiting the “wordcloud” library it is possible to get a hang of the most frequent words for each class. These clouds are generated by using the

TFIDF (*term frequency–inverse document frequency*) weight of each word as metric. These values are obtained by passing the documents to the Tfidfvectorizer of Scikit Learn.
In this case, the only considered parameter is the Italian list of stopwords provided by Nltk.
As expected, words as “camera” or “hotel” are very frequent in both the positive and negative class, but there are other words, for example “colazione”,”personale”, or ”bagno”, which seem to be more frequent in a class compared to the other.



Positive Class



Negative Class

2. Preprocessing

To achieve the goal of building a robust classifier model, the following sequence of operations was performed.

First of all, words are transformed into their base form through lemmatization One public available library able to do this is the Italian lemmatizer from "spaCy" (https://spacy.io/models/it); which although not very simple to use and a little slow, does well in its task. The reason to use lemmatization is to reduce the noise due to the inflectional forms of words.

The final representation of the documents will be a sequence of feature vectors, each feature corresponding to one of the base forms of the words in the dataset with each value being the tf-idf of the word in that particular document. In this way, more weight is given to terms that frequently appear in individual documents but are more rare in the rest of the collection. The next step is choosing the best parameters of the scikit learn's tfidfvectorizer.

The list of stopwords, as before, comes from nltk. This list contains common terms, like pronouns or verbs, and is extended both by other words recommended by the algorithm itself and others inferred by the context: Words like 'hotel', 'camera', 'molto', 'avere', 'essere' are not very meaningful and need to be removed.

All the tokens are converted in lowercase, because there is no reason to differentiate the same word.

In order to consider word pairs as well, the ngram_range parameter is set to (1,2). This comes in handy because sets of words like "camera pulita" or "pessima accoglienza" are good indicators for a good or a bad review.

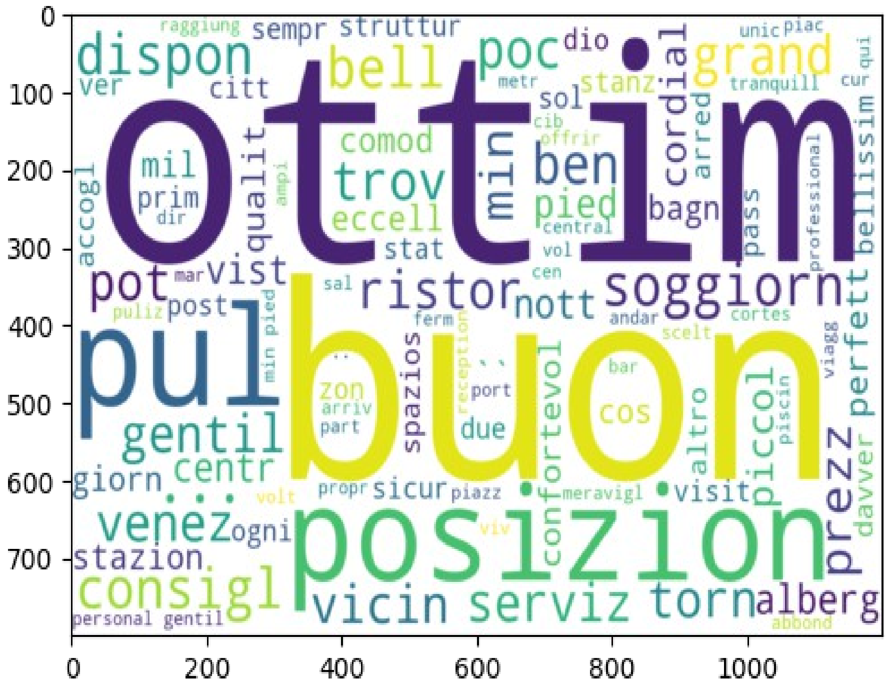
Regarding the max_df parameter, several attempts were made and the best result was obtained with a value of 0.5; by setting this, only the terms that appear at most in 50% of the documents are kept.

In the end, a custom tokenizer is passed to the function. This object divides the document into tokens, removes the punctuation, checks the maximal length (in this case are removed token of length 1) and performs the stemming through the ItalianStemmer, from the nltk library. This last operation was needed in order to achieve great scores.

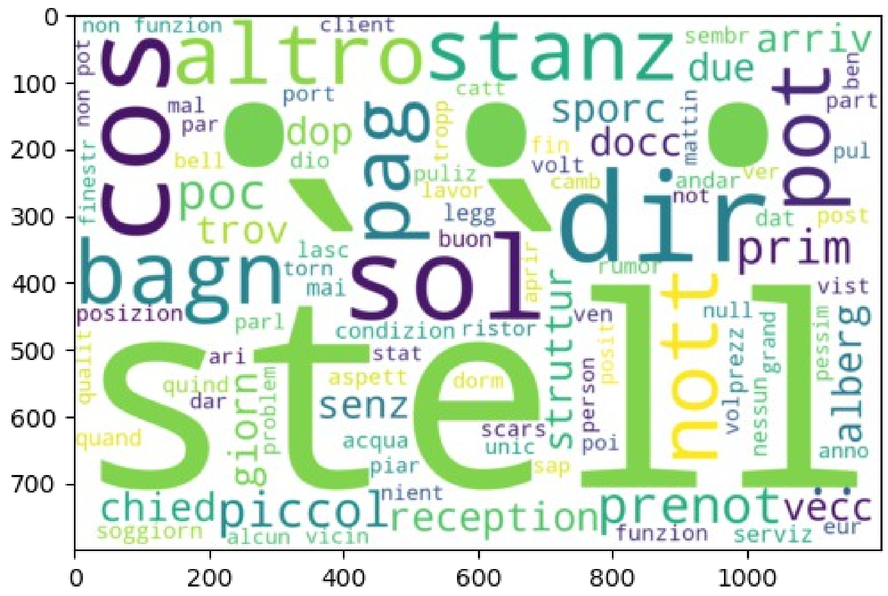
Tfidfvectorizer returns a sparse matrix with 85869 features.

The information regarding the length of the reviews,which at first sight could seem intresting was not taken into account since it greatly worsened the performances.

The result can be visualized exploiting once again the wordclouds.



Positive class



Negative Class

3. Algorithm Choice

Once obtained the result vectors, the right model for the task must be chosen.

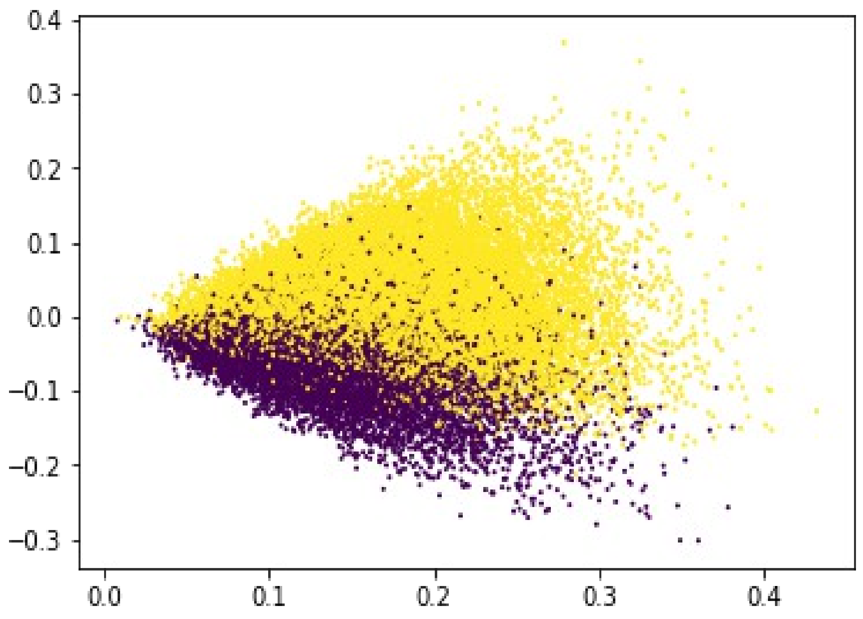
Exploiting the sklearn function cross_val_score, the performances of different classifiers are compared by doing cross validation on the development set. To achieve this, the dataset have been divided in three splits (cv=3) and the f1-macro score has been used as a metric. Classifiers, on the other hand, are evaluated with their default configurations, that is, leaving their constructor empty.

Decision Tree	0,83
Random Forest	0,88

These first attemps have given good results. However, it is worth investigating whether better classifiers can be found.

In order to visualize the general trend of the data in a two-dimensional space, the sparse matrix can be transformed through the Singular Value Decomposition (SVD) and a number of components equals to two. The following figure shows a scatterplot of the two new features. The yellow points are the positive reviews while the purple points are the negative ones. It is easy to observe that the data have a linear trend.

By exploiting this newly discovered feature, it can be deduced that linear models could get better scores.



LinearSVC	0,96
Stochastic Gradient Descent	0,95

As expected, these models perform way better than the previous ones, since decision trees are only able to cut the plane with horizontal or vertical splits. They are also very fast in classification even with such a high number of features.

Since the LinearSVC seems to give a slightly higher score, it has been chosen to classify the reviews in the evaluation set. This classifier implements a support vector machine with a linear kernel. These models are very effective in high dimensional spaces and scale well with dimension, but results cannot be interpreted easily and their actual modus operandi is often obscure.However, some hints are given by the coef_ attribute, which contains the weights of the features in the optimization problem.

4. Tuning and Validation

Now that the final model has been chosen, it is time to set the best hyperparameters for it.

Some interesting hypermarameters of a linearSVC are the following:

Loss function: hinge or squared hinge.

Regularization parameter C: In the official documentation, it is advised to decrease it if there are a lot of noisy observations.

Class weight: since our data are unbalanced, it appears natural to set it to 'balanced'.

Through a GridSearchCV, it is possible to obtain the best combination of these parameters. In our case:

loss = 'squared hinge' class_weight = 'balanced' C=1

Finally, validation is performed on the development set, using 20% of the reviews as test. The *f1_macro score* is **0.962** and the confusion matrix is the following:

Correct/Predicted	Positive	Negative
Positive	3753	109
Negative	73	1816

Since the results are good, the LinearSVC with the right hyperparameters can now be used for the whole development set, and the class of the evaluation set can finally be predicted.