1 Data Exploration

The first issue to be faced is to verify the quality of data source by detecting outliers, handling any missing values and providing some statistics in order to better perform the sentiment analysis task.

1.1 Outliers Detection – Missing values

A quite reasonable way to detect outliers is to select all not italian reviews by checking if there is a certain number of foreign words inside a single document. Only 9 out of 28754 reviews are detected, and consequently removed since they cannot help to better predict italian texts.
[See the References section to find out which reviews have been detected as outliers.] -> Bisogna ancora farla

In addition no missing values are present in the development set, appearing to be quite clean.

1.2 Statistics - Imbalanced dataset

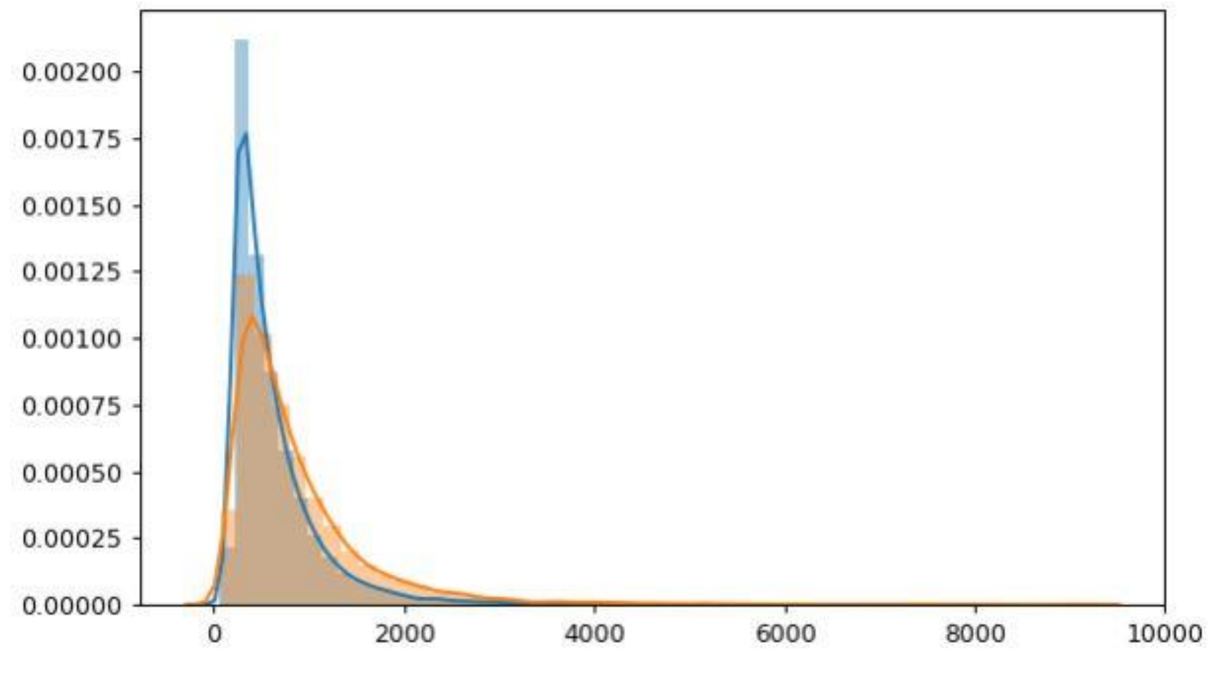
To explore as well as possible data source, some statistics are provided.

Firstly, there are more positive reviews than negatives. Imbalanced classes might impact on the performance of the classifier, therefore some well known technique in literature could be applied such as:

1. ***Over-sampling***: typically applied when few data are present in the training set, hence new syntethic data are generated. It is a critical step because it may be cause of **over-fitting** if the synthetic data will have a data distributon different from one on which it will be tested.
Since the development dataset contains a sufficient number of reviews to train a classifier, over-sampling is not the right choice to pursue.

2. ***Under-sampling***: applied when data source is large enough to train a classifier, so some samples belonging to the majority classes are discarded (randomly or by using some algorithms to remove less significant samples)**.**
Since classes are not extremely imbalanced, no re-sampling techniques have been applied up to now.

Secondly, the length distribution of the reviews is plotted:



The two distributions have very similar characteristics, initially growing exponentially and slowly decreasing on the tails. On average, the negative reviews are longer than the positive ones, the latter however vary more than the former. Apparently, there is no close correlation between the length of the reviews and their target, consequently any assumption (such as to delete reviews longer/shorter than a given threshold for each class) is not strong enough to be pursued.

2. Preprocessing

The next step is to model a document, representing it by a set of features. A good choice is to consider any word as a point into a vector space; consequently a review will be represented as a point inside an N-dimensional space. The following pipeline has been applied:

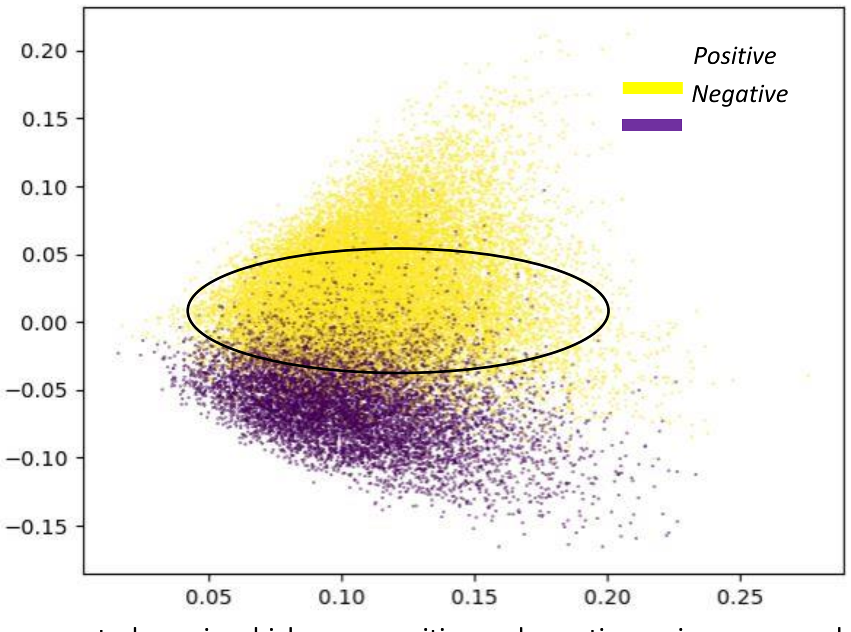
1. **Tokenization**: each text is firstly cleaned by removing any type of punctuation, secondly it is broken into a set of words.
2. **Case-normalization:** a lower case normalization is applied to each word.
3. **Stopwords removal:** in order to reduce as much as possible the cardinality of the N-dimensional space, all words unuseful to discriminate between positive and negative reviews are removed (by using *wordcloud*).
4. **Stemming:** different words with the same root are collapsed into a single dimension by removing prefixes,suffixes and pluralisation. Regarding how to improve its performance will be discussed later on.

Now each document is going to transform into a feature vector consisting of a set of weights, one for each distinct word.

Each weight is given by ***TF-IDF*** schema, in which terms appearing frequently in a single document but rarely in the whole collection are preferred.

2.1 Dealing with noisy data

Subsequently, a reduction into a 2-D dimensional space is performed by using SVD technique in order to visualize how documents are distributed along two highest explained variance features (be aware that ant assumption is done on a simplified model in which documents are distributed along two highest explained variance features!) .



This figure shows a central area in which some positive and negative reviews are overlapped,therefore there is a group of reviews belonging to two different targets that are similar (i.e. reviews containing both good and bad aspects, having an intermediate rate), considering parth of them as noisy data.

In order to try to be affected as less as possible by noise, a re-sampling method based on ***Tomek links*** is applied:

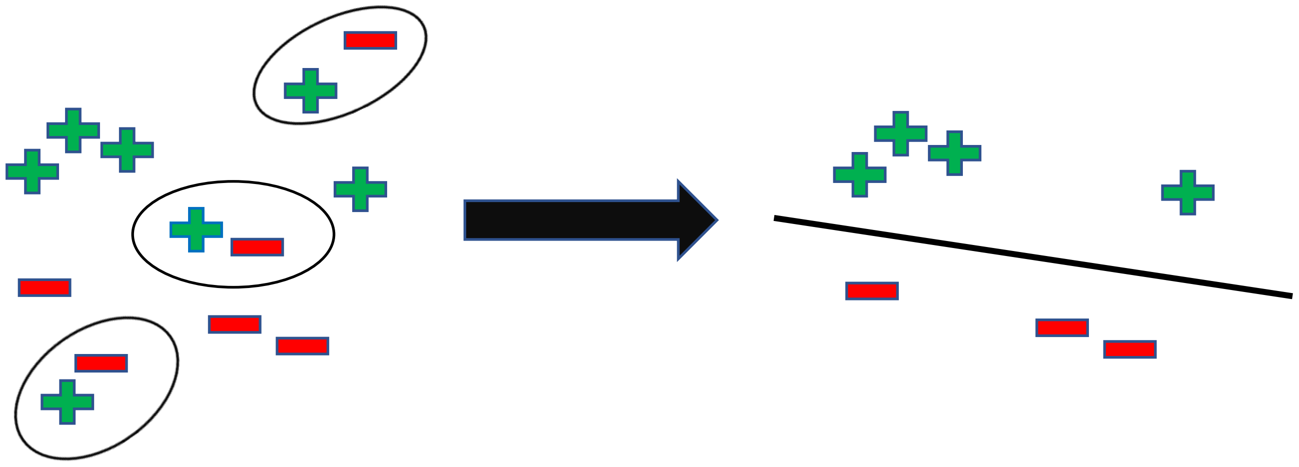
(DEF.) Tomek links:

Let d_p a positive document and d_n a negative one.

Let D(d_p , d_n) be the distance between d_p and d_n.

Then (d_p , d_n) is a *Tomek Link* if:

1. D(d_p , d_n) < D(d_p ,d) $\forall$ d
   or
2. D(d_p , d_n) < D(d, d_n) $\forall$ d



Noisy dataset *Re-sampled dataset – Tomek Links*

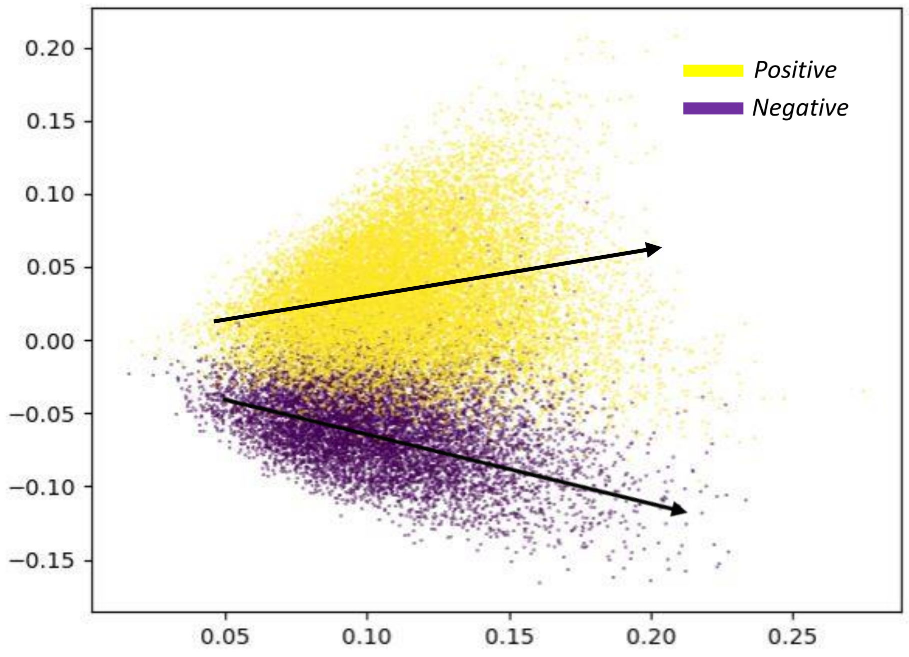
The elements contained in the ellipses form a *Tomek link*, as they belong to different classes and are closer to each other than to any other instance: either one of them is a noisy instance or they are both located near the decision boundary.

All reviews forming a *Tomek link* have been discarded, ending up with a training set with about 1000 out of 28754 less.

[See the References section to discover which reviews have been detected as noise.]

3. Algorithm Choice

The aforementioned re-sampling technique finds a subset of the initial dataset. A 2D-plot obtained after **SVD** technique is plotted:

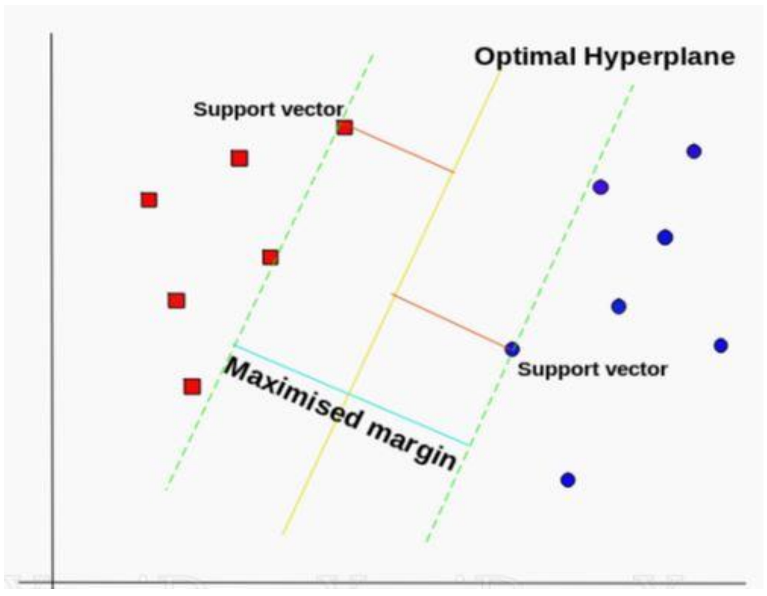


Comparing this chart with respect to the previous one, training set seem to be less affected by noise. In fact reviews are better separated and the number of different targets spreading over the separation line has decreased.

In this context, the first choice of the algorithm to be used has been directed towards ***Support Vector Machine***. A brief recapitulation on how it works is provided.

3.1 Support Vector Machine (SVM)

Support Vectore Machine is a widely used technique for data classification and regression. The original idea is to use a linear separating hyper-plane which maximizes the distance between two classes to create a classifier.



In particular, SVM algorithm finds the points closest to the line from both the classes, calling them *support vectors*. Then it computes the distance between the line and the support vectors, defining it as *margin*. Finally it ends up with an *hyperplane* for which the margin is maximum (i.e. classes are separated as well as possible).

Since it works with distances data must be normalized prior to feeding them to the SVM, scaling the values to lie roughly within range [-1,1]. Otherwise, if e.g. dimension 1 is from 0-1000 and dimension 2 is from 0-1, then dimension 1 becomes much more important than dimension 2, which will skew results.

3.2 Comparing results

Finally, everything is ready to be tested on the development dataset.

To find out if the resampling technique fits well with different types of classifiers, both ***SVM*** and ***Random Forest Classifier*** are used, giving the following results: (Togliere tabella e inserire grafico su tableau anche con n-grams)

	SVM (f1-score weigthed)	RFC (f1-score weighted)
TF-IDF	*96.4%*	*93.8%*
TF-IDF (re-sampled)	*96.9%*	*94.7%*

Regardless of the method used for classifying, re-sampled training set allows to reach higher score than the initial one, thus confirming that it is slightly more robust to noise. Furthermore, since resampling works with Tomek Links, the removal of them involves a better separation of the two targets in the N-dimensional space (a simplified example in 2 dimensions is provided in section 2.1), thus allowing SVM to be able to divide them better. Consequently, classification algorithm chosen is **SVM** which works on the **resampling dataset**.

4. Tuning and Validation

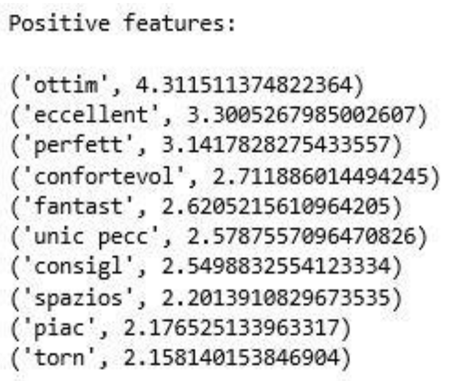
This section will describe how all the parameters are coherently selected for each algorithm.

The first group of parameters to tune concerns the *TfidfVectorizer*, which are the following:

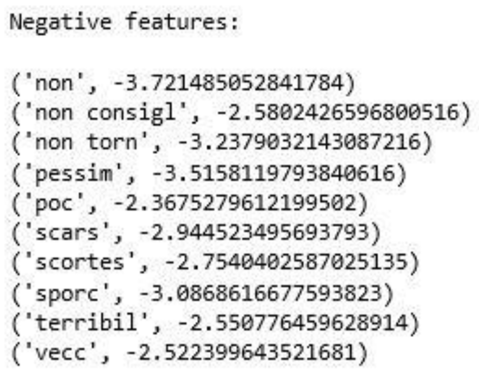
1. **min_df** : keeping it by default (equal to 1), more than 700.000 features are created that is surely an huge number to pursue the given task. Consequently, it is increased by 1 incrementally until the number of features does not change a lot with respect to the step before, choosing it **equal to 5** ending up with about 55000 features.
   Since the content of the reviews is homogeneous (i.e. they talk about the same topic) it is quite reasonable to keep an high number of features. It would have been different in a context in which the content of the texts was heterogeneous and the task was to distinguish the topics. In the latter case it would have been better to choose a few features by applying an SVD (i.e. the words used in different topics are completely different, therefore selecting only the most representative ones performs the distinction better).
2. **ngrams_range:** in many cases the positivity/negativity of a word depends on what is written before (e.g. "buono" becomes negative if it is preceded by "non"). For this reason also combinations of two words are created as a single feature, choosing ngrams_range **equal to [1,2]**
3. **norm:** As mentioned in section 3., SVM classifier needs normalized data , hence a norm **'l2'** is applied.

The second parameters regards *SVM* , given by a grid search based on **cross-validation** technique. The best performing estimator found provides a **linear kernel** and a low value of **C**, thus finding an optimal smooth hyperplane which is a linear combination of the features (the lower is the value of **C**, the less is the risk to incur in over-fitting).

For each class, the most important features combined with its coefficients are the following:

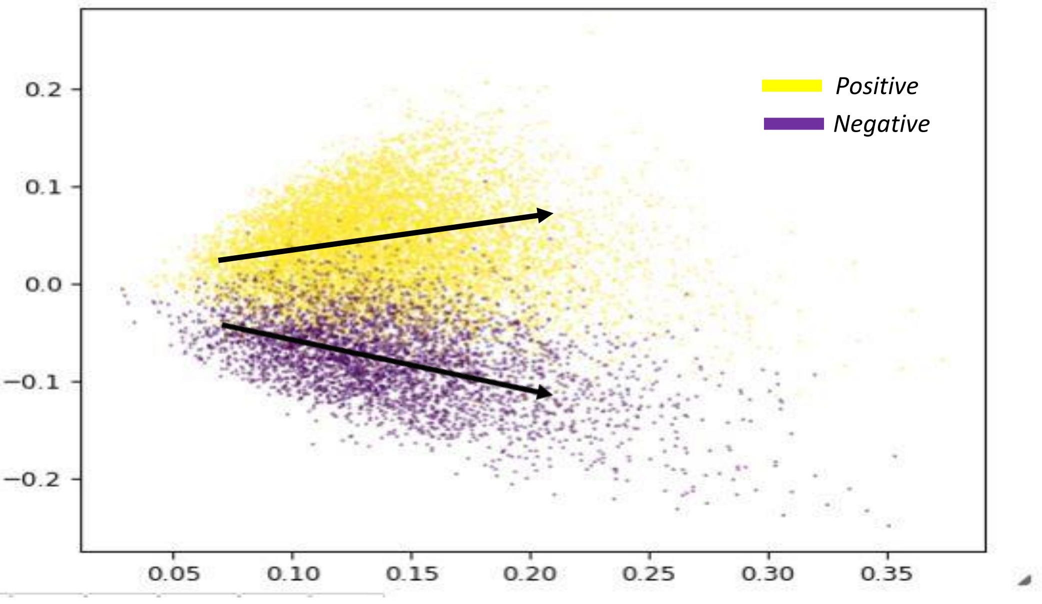


```
Positive features:

('ottim', 4.311511374822364)
('eccellent', 3.3005267985002607)
('perfett', 3.1417828275433557)
('confortevol', 2.711886014494245)
('fantast', 2.6205215610964205)
('unic pecc', 2.5787557096470826)
('consigl', 2.5498832554123334)
('spazios', 2.2013910829673535)
('piac', 2.176525133963317)
('torn', 2.158140153846904)
```



```
Negative features:

('non', -3.721485052841784)
('non consigl', -2.5802426596800516)
('non torn', -3.2379032143087216)
('pessim', -3.5158119793840616)
('poc', -2.3675279612199502)
('scars', -2.944523495693793)
('scortes', -2.7540402587025135)
('sporc', -3.0868616677593823)
('terribil', -2.550776459628914)
('vecc', -2.522399643521681)
```

In order to find out how the two targets are distributed on the evaluation dataset, a bi-dimensional plot applying SVD is shown:



Positive and negative reviews seem to follow the same directions as those in the training set, thus demonstrating the goodness of the previous hires.

Be careful that the shown labels are the predicted ones and not the “ground truth” but, since the submission platform gives a *weighted f1-score* of **97.4%,** the chart shown above can be considered quite reliable.