Data Science Lab: Process and methods

Politecnico di Torino

Project report

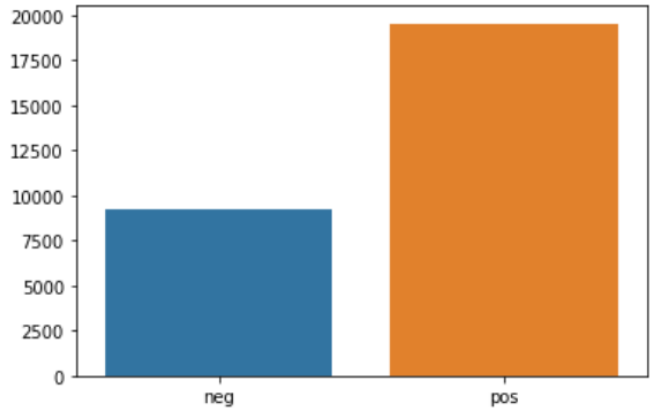
Student ID: s274563

Exam session: Winter 2020

1. Data exploration + Preprocessing

This is a classification problem with two classes ('positive' and 'negative') and one feature: a text data. First of all, I have analysed the number of reviews I had in the development and how many of them were positive or negative.

class	text
neg	9222
pos	19532



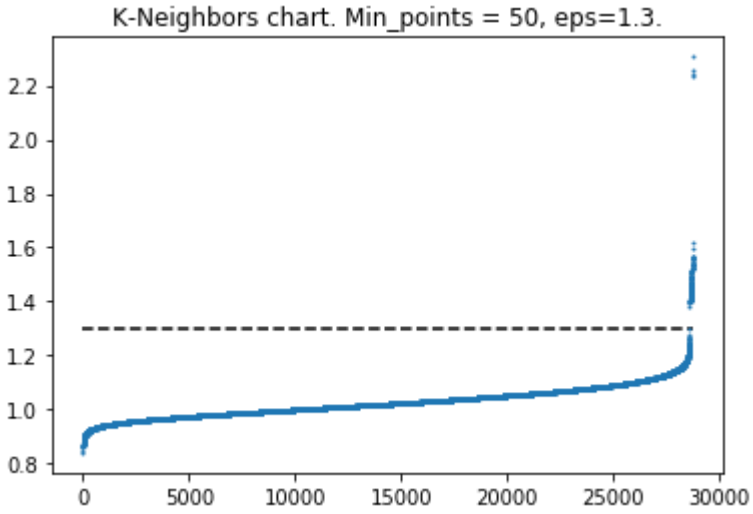
I had around 30.000 reviews, of which 66% were positive, and the remaining (34%) negative. Therefore, my dataset is not so balanced (and this could be a problem for my classifier, but fortunately I have got neither missing or repeated data). In order to have a better understanding of the language used in this dataset, I decided to produce a wordcloud of all the dataset.

The picture shows that the original dataset is in Italian, but after some analyses I discovered that there are further English and French versions of the comments, and that part of the reviews are simply translations. I have also tried to plot separately the words found only in positive or negative reviews but the result was not so different.

Although all the reviews were short documents (only a few words in many cases), I decided to analyze every document as a whole, and not to split them in phrases or sections. As a first step , I broke the text into tokens (each token represents a word), I removed the punctuation and I transformed all the chars from upper to lower cases. Then I removed the suffix of all words and maintained only the root, operation known as "stemming". I have also tried to remove stopwords and numbers but this choice didn't work.

Next step was to transform the tokens into a feature vector using a weighting scheme. Even if I had a collection of homogeneous contents, a tf-idf approach fits better the problem. When this method builds its vocabulary, it ignores all terms in the document that are strictly more frequent than a threshold value(I chose this value around 0.5). In the same way, the algorithm ignores all the terms which don't appear at least in three document. In this way was possible to remove all the grammar errors and the most part of conjunctions, articles, pronouns and so on… To have a better understanding of the text, I decided to sample the documents not only in single words but also in groups from two to four words, to better understand the context of the phrase.

Then, I looked for the presence of outliers in the dataset. To do that, I chose an Unsupervised learning technique called "clustering" and I adopted the DB scan algorithm in the case.

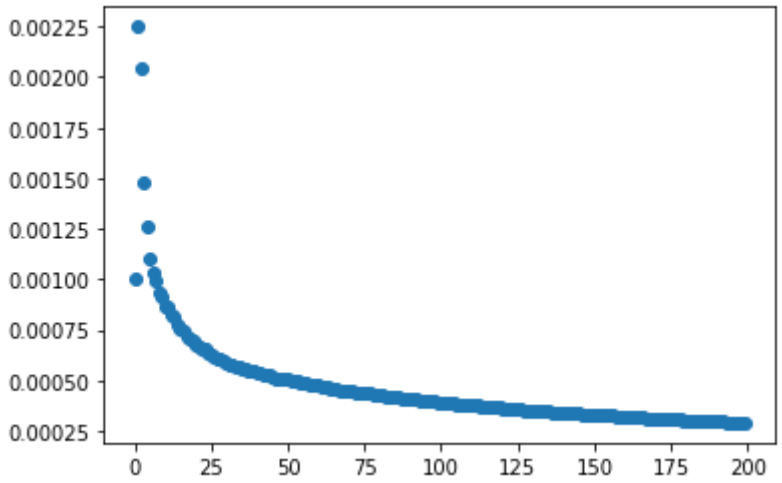


Even if, with the choice of parameters shown in the image above, there were some outlayers, I didn't have any improvement by removing them.

Before going on, I also tried to add three more features, to give further informations about the text to the classifier. First of all, I started to look for “good words”, or words that express positive contents (such as “suite", "bellissimo", "indimeticabile”…) and “bad words”, that are words that express negative contents (like “decadente", "orribile", "caserma”). Then I counted the number of times the words were repeated in the text and I added these two information to the features created by the td-idf. Moreover, I considered the length of every review because I noticed that the bad words were usually shorter than the good ones.

After doing that, I also noticed that the td-idf produced a huge number of features, and this could be a problem for my classifier for two reasons. First of all, if I had too many features my classifier would be extremely slow and it would take a lot of time to find good hyperparameter. The second - perhaps more crucial - reason was that with a high number of features (which are all correlated, maybe) it would be harder to find a good and appropriate solution. So this was a perfect condition to apply a dimensionality reduction. The first approach I tried was the “Principal Component Analysis” (PCA), but unfortunately this algorithm didn’t work with sparse matrix; so I tried to use the truncated SVD method, which is optimized to work with sparse objects.

To select the right number of dimensions to reduce them down to, it is necessary to choose those with a portion of sufficiently large variance .



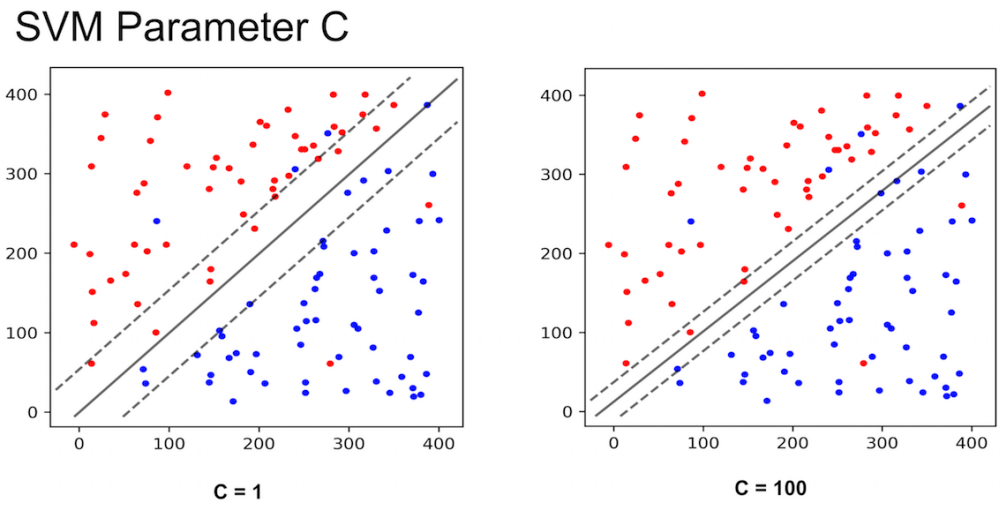
Looking at this image, it seems that a good number of dimensions could be around 100, the so-known "elbow point", but in that particular point I had only the 0.061% of all the variance. Going on, with 200 dimensions I had only the 1% of all the variance. I also tried to increase the number of dimensions, but the algorithm became extremely slow, too slow to work with it.

Before passing everything to the classifier, I computed the chi2 statistic between each feature and the labels, and I only took the ones that produced a large value, because this means that these features are non-randomly related to the labels. So, an important piece of information is provided: in this way, I was able to halve back the number of the features.

2. Algorithm choice

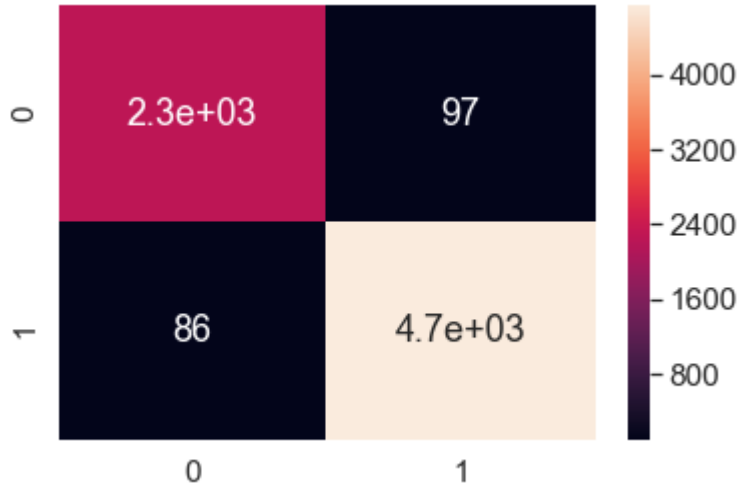
Since the KNeighborsClassifier (which I usually use) is not suggested for text classification I have tried a great number of different classifiers. I started with the “random forest classifier”, which generally works and in fact I immediately got a “F1 score“ which is higher than 0.9 but, after doing some research, I discovered that there were classifiers more recommended for the managemet of the problem. In fact, in sentimental analysis a “Linear Support Vector Machine”(LSVC) model works really well and can easily determinate the class of a review after tuning up the parameters. Moreover LSVC is particularly suggested for problems which had less than 100k samples and it’s optimized for working with sparse matrix.

3. Tuning and validation

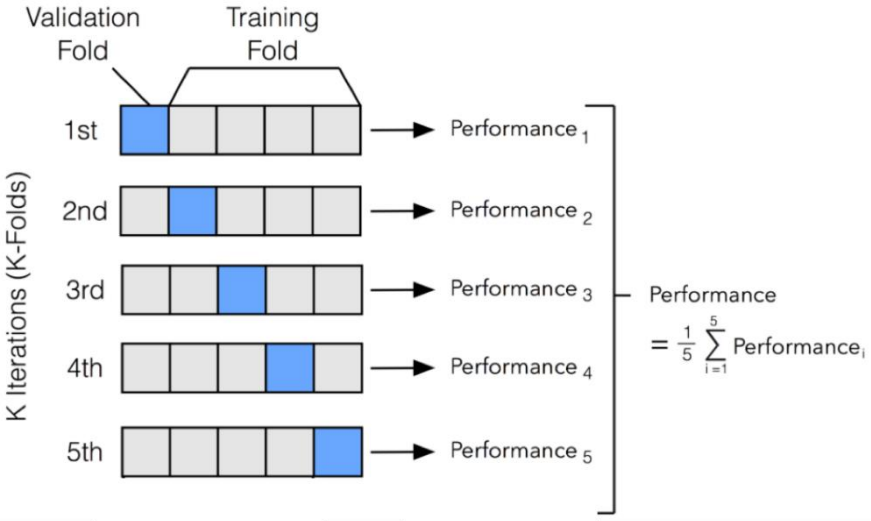


The basic idea underlying this algorithm is simple. It’s clear from the image, that SVM separates the two class with a straight line. The most important parameter is C. As you can see from the image, with a high value, the margin violation is not so bad but this can cause an overfitting problem. Using a lower value of C, the margins are even worst, but it generalizes better. First I tried with a GridSearch approach and it underlined that that algorithm performed better with low values of C. So I tried to use the lowest possible value, and with the values in the range between [2,5] the algorithm performed excellently.

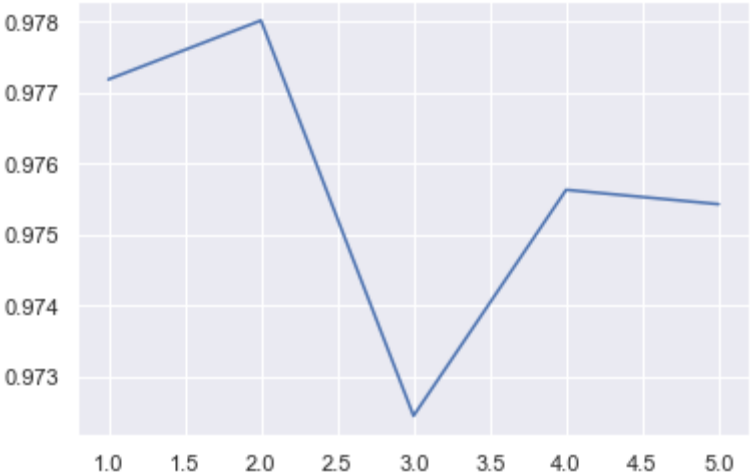
So, to train my model until obtaining a large dataset, I split my dataset into two parts: I assigned ¾ for training and ¼ for testing. After the training and the predicting phase, I plotted the F1_score, which was very high (about 0.978). To get a better understanding of the result, I decided to plot the confusion matrix, and to my surprise the rate of false positive and false negative was quite the same even though the dataset was not balanced.



To validate the stability of the machine learning model and to understand how much it was possible to generalize in the evaluation of the dataset, I chose to opt for a cross validation approach. I split the dataset into five parts and I trained my model on four of them, predicting the labels of the remaining part as it's shown in the picture below.



The result (as you can see in the picture below) was good, not so good as it was before, but it allowed me to understand the validity of the steps that I had taken and that could have led me to obtain a great result if I had pursued that path.



4. Conclusions

That was a very difficult competition in my opinion, mainly because everything started with an Italian dataset. Considered this, it was really hard to find a library that could work with the Italian language and also the research of an additional external dataset that worked well was nearly impossible. I have also tried to translate all the dataset into English, but unfortunately I didn't obtain any further improvement. So I decided to do my job with the initial dataset, without any kind of modification.
I uploaded two different results, but they are basically the same file which differ from each other only in a few parameters. All the steps I have presented until now can be found in both solutions.

Finally, in managing this solution I came across a lot of approaches which involve neural network, Word2vect, TextBlob and so on, which can surely help in this kind of task, but I have decided to choose none of them, even though they were allowed and some of them were also introduced into the class during the Academic Year. They are very complex and even if they work in an amazing way, I can't really show what I have learned in the annual course because, a deep approach is considerably far from what we have been taught during the class and lab activities.