Data Science Lab: Process and methods
Politecnico di Torino

Project report
Student ID: s265386

Exam session: Winter 2020

1 Data exploration (max. 400 words)

One of the first steps is to understand how to read the dataset. While it is immediately clear that, in *development.csv*, the reviews are well separated by ",pos\n" or ",neg\n", so it is straightforward to split them with some regular expressions, it is instead more challenging to understand how it can be done on *evaluation.csv.* Since it doesn't have the labels previously used as separator, it can be seen that there are 3(+1) ways a review can start or end. Starting cases:

1. by double quotes, but not immediately followed by a second one
2. by double quote and immediately go to a new line
3. by triple double quotes, but not immediately followed by a fourth one

(Ending cases are analogous, but the limitations are on what precedes the double quotes)
Those six cases are handled with regular expressions, while another case is when a review doesn't neither start, nor end by double quotes. In that situation it lays on one line and it is easily recognizable.

During this exploratory phase, some interesting aspects of the dataset were noticed:

- Presence of emoticons and smileys
- Some text in foreign languages
- Presence of money related symbols

Using the explore(word) function it can be seen how many times a word is associated to positive and negative reviews, hence the most frequent emoticons and smileys were classified as positive/negative/irrelevant. The positive and negative ones were substituted with the words "ottim" and "pessim" respectively, two words chosen according to their meaning and their unbalanced split on the two labels. This step is important for unifying under two single words many different symbols and their meanings, it's not negligible because the total number of substitutions is around five hundreds.

Some text in Chinese language was detected and translated in Italian in order to have the same kind of words to be classified.

Also some other symbols, as £, $ and €, were found and substituted with the currency name to merge elements with same meanings (after observing that the two versions of each currency had no particular differences in their distribution on classes). More than twenty five hundreds of these kind of symbols were substituted.

A final search for *implicit features* was done observing some punctuation characters or the review length, but nothing useful was found and, without having missing data to handle, the data exploration phase was concluded.

2 Preprocessing (max. 400 words)

In sentiment analysis it is important to reduce the number of words and try to keep only the ones containing some opinion in order to perform feature extraction.

2.1 Tokenization and cleaning

It is necessary to tokenize the reviews at first, so they are not only split by whitespaces, but also with a specific set of separators (punctuation and symbols). All the words with length lower than three are then discarded because almost certainly would not bring any kind of sentiment.

2.2 Stemming

The NLTK Snowball Stemmer[4] is an important tool used to perform stemming, a morphological transformation converting each word in its root form (i.e. stem). Thanks to this transformation many words with different variants become identical, increasing the classification "power" of those terms.

2.3 Stop word generation

Some of the words left are too frequent to be useful in the classification, or still don't bring any subjectivity, that's why a list of stop words to be removed from the reviews is generated. The sources are:

- nltk.corpus.stopword.words("italian")
- a subset of the most frequent words in the development reviews

The custom subset has a great relevance since it is extracted directly from the dataset, therefore it includes some terms that usually might not be stop words (as "hotel"). From the obtained list, some elements are removed (thus are not deleted from reviews) because of their relevance even if they are too frequent (like "non"). Some others are also kept due to their good unbalanced presence in the two classes, but must be kept in mind that the two classes are also unbalanced (70% of reviews positive and 30% negative). The list of stop words is also stemmed in order to match with the stems in the reviews.

2.4 Vectorization

The reviews, now stored as lists of words, are then rebuilt concatenating those words and whitespaces. In this way they can be passed to the Scikit-learn[3] TfidfVectorizer together with a tokenizer and the stop words list. The vectorizer allows setting some important parameters:

- min_df: 2, to remove features appearing once, thus not useful in classifying
- max_df: 0.35 (empirically), to remove too frequent features

- max_features: 20000 (empirically), too few would mean risking not having enough for classification and too many may impact on quality and performances
- ngram_range: (1,2) (empirically), to consider also couple of words and the additional meaning they can carry

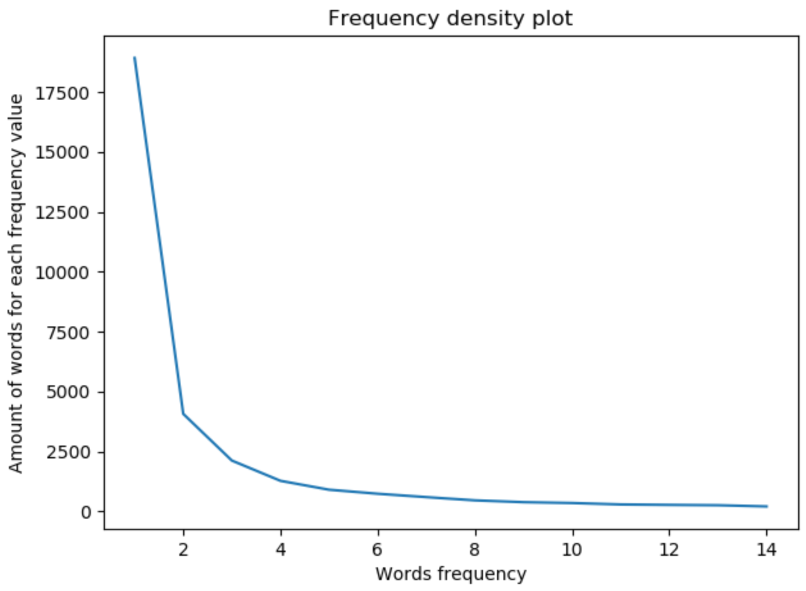

Figure 1: Could set min_df>2 for further reducing #words

3 Algorithm choice (max. 400 words)

The nature of the problem clearly sets some boundaries (interpretability in not interesting, for example), but also some requirements for the algorithm choice, that is the algorithm must be able to work with an high number of features and samples.

Decision tree classifiers are not taken too much in consideration mostly because they are a quick and dirty method, even if they are fast, inexpensive and interpretable. An analogous description can fit also Rule-based classifiers, while instead the *Random Forest* is a much more valid candidate, as it has ”impressive accuracy in classification”[1] for Opinion Mining and is also robust to outliers. Another possibility could be Associative classifiers, but have a reduced scalability in the number of attributes, that is also true for the K-Nearest Neighbor since it suffers from ”curse of dimensionality” having to deal with distance measures. Neural networks instead require a very long training and a deep knowledge of the subject to be effectively used for this purpose. On the contrary, *Naive Bayes* might be another valid choice even if it is known from the literature not to be particularly accurate due to the naive hypothesis. Its main features are: being incremental (thus useful for applications as spam filters) and requiring limited resources and it still shows good performances since ”there are some theoretical reasons for this apparent unreasonable efficiency”[1] in the *Sentiment Analysis* problem. Although Random Forest and Naive Bayes bring good results, the best ones are given by *Support Vector Machines*. In general they show good accuracy and performances, but more importantly they are effective in high dimensional spaces (that is the case of this analysis). This is the reason why Scikit-learn[3] Support Vector Classification with linear kernel (LinearSVC) has been used. From the documentation, LinearSVC is know to scale better in case of high number of samples and it is much more efficient than standard SVC.

A comparison between the performances of most of the considered methods can be seen in Fig.2, where the plot (obtained thanks to Matplotlib[2]) of the Receiver Operating Characteristic (ROC) curves and the Area Under Curve (AUC) values show that LinearSVC, Naive Bayes and Random Forest are in fact the most suitable classification algorithms for this problem.

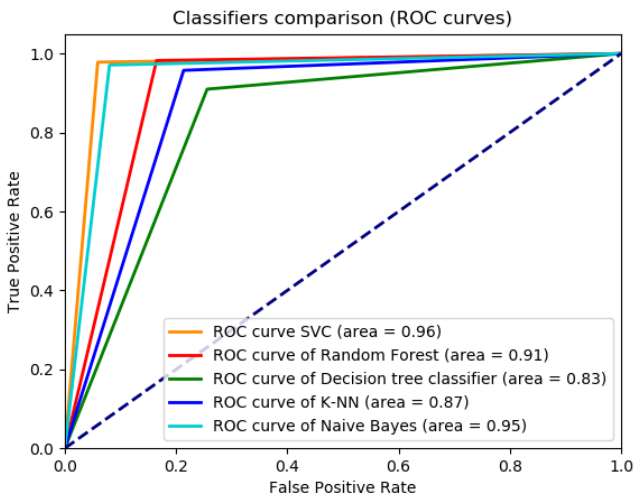

Figure 2: ROC and AUC comparison for different classification algorithms

4 Tuning and validation (max. 400 words)

4.1 Tuning

The LinearSVC algorithm has a main parameter to tune, the *regularization parameter C* that serves as a degree of importance that is given to miss-classifications. SVM pose a quadratic optimization problem that looks for maximizing the margin between both classes and minimizing the amount of miss-classifications. As C grows larger, the less the wrongly classified samples are allowed, that means a too large C causes overfitting, while a too low C causes underfitting. In order to find its optimum, a given range of values was scanned performing, at each iteration, a cross-validation with 20 folds and extracting the F1 (weighted) score that was then plotted (min/average/max) in Fig.3 and 4.

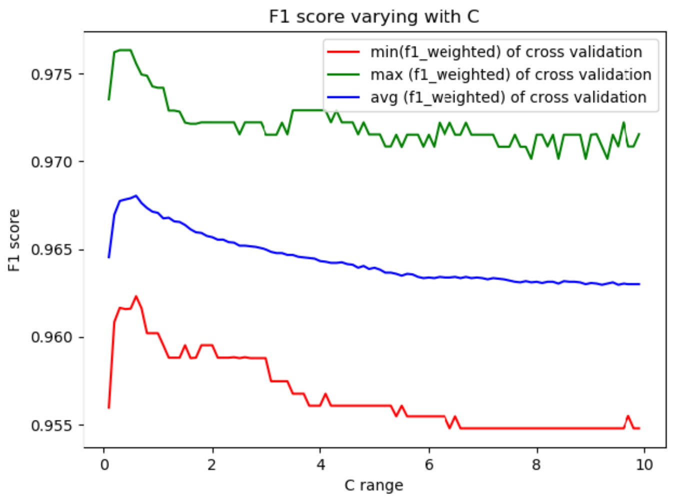

Figure 3: Observe overfitting with too high values of C

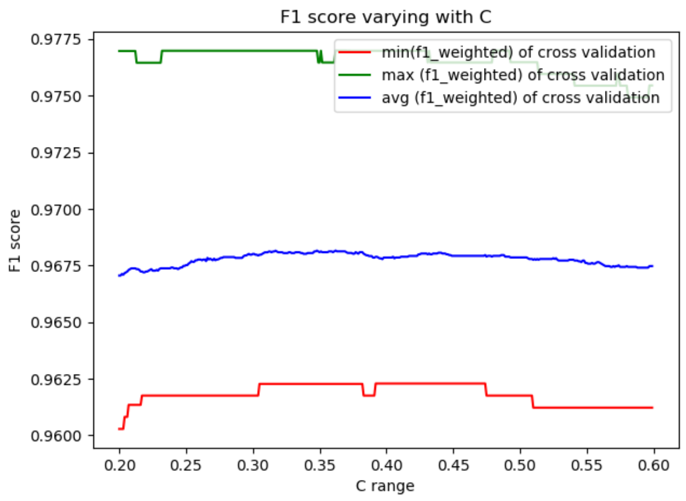

Figure 4: Finding best value of C at 0.3125

Furthermore also the max_iterations parameter could be set, but its behaviour (Fig.5) is not very interesting. It simply shows that with max_iteration<6 the algorithm doesn't converge, any value above that threshold doesn't affect the result.

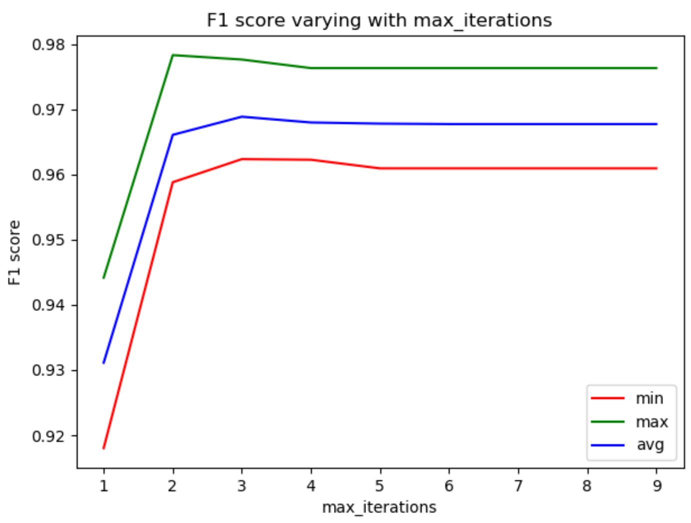

Figure 5: F1_score stabilizes after a given number of minimum iterations

4.2 Validation

The validation of the obtained model can be carried out computing a *confusion matrix* and some specific measures as *accuracy, precision, recall and F-measure.* From the confusion matrix, shown in Fig.6, it can be immediately seen that the amount of true positives (TP) and true negatives (TN) are extremely high with respect to false positives (FP) and false negatives (FN), therefore the result is overall good. To be more specific, some measures must be computed, but accuracy is not a reliable metric due to the unbalancing of the two classes. It is better indeed to compute:

- Precision_pos = TP/(TP+FP) = 950/(950+23) = 97,64%
- Recall_pos = TP/(TP+FN) = 950(950+20) = 97,94%
- F-measure_pos = 2TP/(2TP+FN+FP) = 2*950/(950+20+20) = 97,79%
- Precision_neg = TN/(TN+FN) = 445/(445+20) = 95,70%
- Recall_neg = TN/(TN+FP) = 445/(445+23) = 95,09%
- F-measure_neg = 2TN/(2TN+FN+FP) = 95,39%

The obtained result is very good especially because the precision on the smaller class ('neg') is very high even if it's harder to be classified.

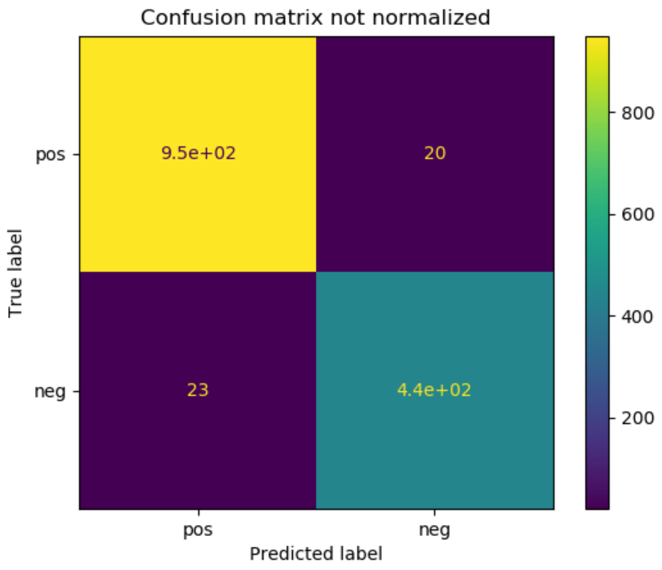

Figure 6: Confusion matrix: holdout with test_size=5%

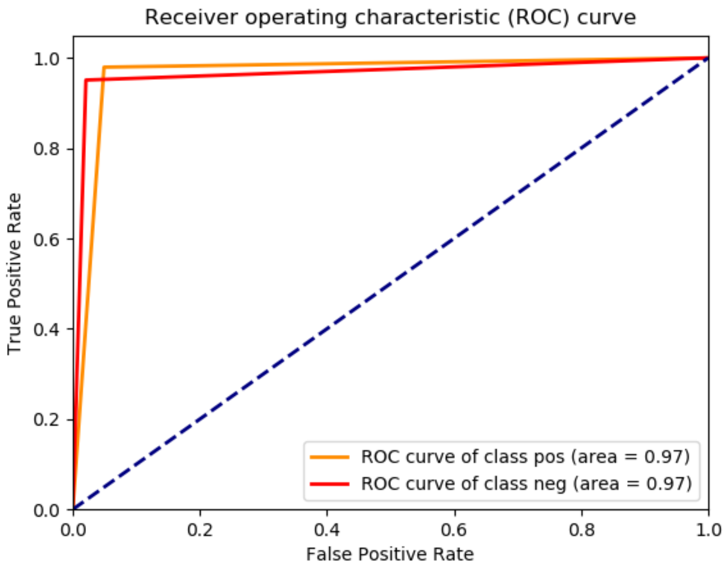

Figure 7: ROC curves for the two classes

References

[1] Pratik Gadgul Akshay Kadam Amit Gupte, Sourabh Joshi. Comparative study of classification algorithms used in sentiment analysis. *Amit Gupte et al, / (IJCSIT) International Journal of Computer Science and Information Technologies*, 5:6261–6264, 2014.

[2] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] Ewan Klein Steven Bird and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.