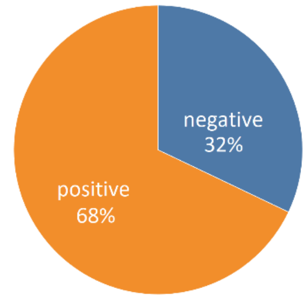
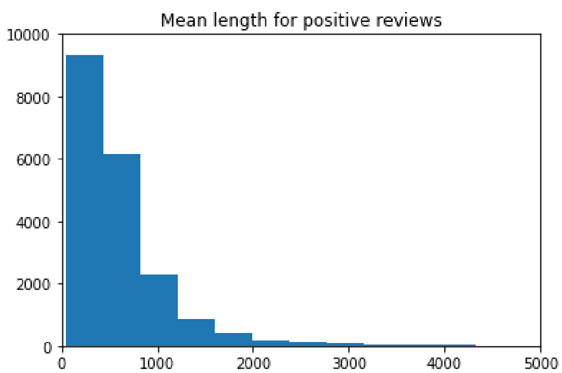
Sentiment Analysis

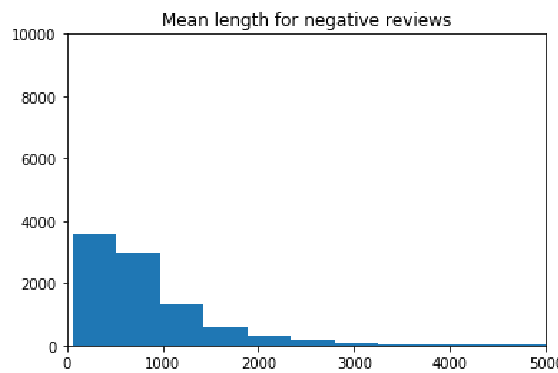
1. Data Exploration



Our development dataset contains 28754 reviews, while the evaluation dataset has 12323 reviews (42% of the dev).
In the development dataset we have 19532 positive reviews (68%) and 9222 negative reviews (32%) → this is an imbalanced dataset.
Concerning outliers, in the development we have 10 reviews of foreign language[1], while we have 7 in the evaluation.
The following table contains some features with a different distribution among the two classes, so any of these features is a possible candidate.

Features that may have had an impact	Positive	Negative
Text length: mean, stddev	625, 513	864, 740
First name[2]: reviews with at least one first name	26%	54%
#Exclamation marks: mean, stddev	0.94, 2.24	1.63, 3.63
#Question marks: mean, stddev	0.04, 0.32	0.27, 0.98
#Sentences: mean, stddev	6.4, 4.55	7.5, 6.36





2. Data preprocessing

Stopwords

Starting from the `nltk` dataset, I added some words to the list. However, the most important step was to remove negations from the stop-words. The reason is that, given an n-gram, it is possible to have a negation of a certain token. If we leave negations to stop-words, "non è bello" ("it's not great") could wrongly become "è bello" ("it's great").

String replacing

I have replaced extremely common words which usually are splitted by a character. For instance, «wi fi» will become «wifi».

[1] Reviews with a number of Italian words < 90% (according to `spaCy`)
[2] Italian names from: https://data.world/axtscz/italian-first-names

Tokenization
Possible choises: simple tokenization, lemmatization, stemming. All these features are available in the `spaCy` library. While tokenization simply splits the text into words, lemmatization and stemming usually provide a better pre-processing. Despite some issues (the lemmatization of “la fermata del bus” is “la fermare del bus”), I have decided to use `spaCy` lemmatizer as tokenizer of the `TfIDFVectorizer`.

Outliers elimination
For each review, `spaCy` is able to detect the main language and to assign a score (percentage of words of that language). I have used this feature to detect all the non-Italian reviews and the reviews with a score < $t_{threshold}$. I have translated the reviews without a single sentence in Italian, while I have simply deleted the non-Italian part on the reviews with a score below $t_{threshold}$.

3. Algorithm choice
A **Naïve Bayes** classifier, despite the dependence among the tokens [1], offers efficient performances with a reasonable accuracy [2].
A **Random Forest** is a type of classifier which, due to its randomness, is extremely robust to noise and outliers [3]. According to [4], RF could be highly competitive in sentiment analysis with a fine tuning of hyperparameters. Given the structure of a decision tree, it is possible to easily add features without normalize the data.
An **Artificial Neural Network** can simulate the structure of the human brain. As stated by [5], it can be more accurate and precise as compared to NB and SVM algorithms.
A **Support Vector Machine** construct a set of hyperplanes into a high-dimensional space which split data into classes. It is possible to obtain results comparable to the other algorithms [6]. This classifier allows us to have a specific weight for each class. This could improve our solution given our imbalanced development dataset.

4. Tuning and validation
Following scores are performed using `train_test_split` with a test set size = 30% of dataset size.
To tune hyperparameters of vectorizer and classifier I have used `hyperopt`[3]. Once we have defined a distribution, the TPE algorithm [7] will find the optimum by minimizing the objective function, which I have set to $1-\left(p_{pos}\cdot F_{pos}+p_{neg}\cdot F_{neg}\right)$, where p_{class} is the class percentage and F_{class} is the F1 score of *class*.

4.1 Data balancing
External dataset used: “Scraping-TripAdvisor-with-Python-2019”[4].

(pos-neg)	Final distribution (pos-neg)	F1 score variation[5]
Original data	70-30	
Original data – 20%pos → 50-50	50-50	-5%
Original data + (0-100) $_{external}$ → 50-50	50-50	-10%
Original data + (50-50) $_{external}$	65-35[6]	-2.5%
Original data + (70-30) $_{external}$	70-30	**+5%**

[3] https://github.com/hyperopt/hyperopt
[4] https://github.com/giuseppegambino/Scraping-TripAdvisor-with-Python-2019 > reviewALL.csv
[5] F1 score = $F1_{pos}$ * $\%_{pos}$ + $F1_{neg}$ * $\%_{neg}$
[6] Depends on the external dataset size

4.2 Vectorizer (TF-IDF)

The values obtained with `hyperopt`, having a space of [0, 0.5] for min_df and [0.5, 1] for max_df, are those in the following table. The best values have a max_df of about 0.3, and a very low min_df. The worst results have a max_df which varies from 0.5 to 0.95 (so all the available space), but the common factor is the high values of the min_df. We can say that we need a low min_df (low values could also be zeros) and a max_df of about 0.3.

I have used the same approach for the two solutions I have sent. I have obtained opposite results with local trials compared to the leaderboard, so I have uploaded both due to the possibility of having overfitted the test set.

Best values

max_df	min_df	loss
0.346	0.0013	0.0699
0.275	0.0070	0.0733
0.317	0.0009	0.0747
0.292	0.0003	0.0768
0.383	0.0047	0.0776
0.441	0.0002	0.0796
0.314	0.0057	0.0802

max_df ≈ 0.3
min_df: low

Worst values

max_df	min_df	loss
0.503	0.4763	0.4512
0.557	0.2439	0.3378
0.515	0.3939	0.3298
0.524	0.2720	0.3291
0.822	0.4390	0.3098
0.965	0.4975	0.3041
0.958	0.4634	0.2943

max_df: any
min_df > 0.2

4.3 Decomposition

Having n_1 of features from the TF-IDF matrix, and reducing the feature size with SVD to $n_2 \ll n_1$ features, turned out to be worse: we gain 20% of avg F1-score by vectorizing directly using $max\_features = n_2$.

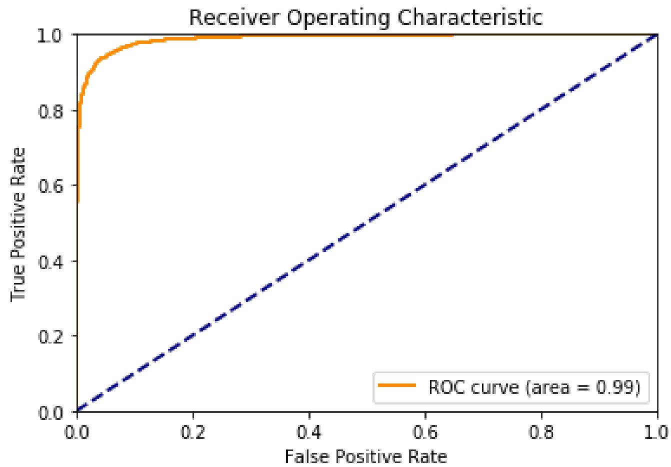
n_1	n_2	F1 [neg; pos]
20 000	5 000	0.8425; 0.8710
5 000	None	**0.9253; 0.9502**

4.4 Classifier

	BernoulliNB	GaussianNB	MultinomNB	ComplNB	RandFor(100)	ArtNN	LinSVC
F1 (neg, pos)	0.9279; 0.9518	0.9172; 0.9446	0.9346; 0.9565	0.9385; 0.9581	0.9158; 0.9464	0.9164; 0.9448	**0.9379; 0.9612**
Time (mm:ss)	00:01	00:05	00:00	00:00	01:50	15:00	00:05

ngram_range	F1 [neg; pos]	Time (sec)
(1,1)	0.9106; 0.9498	10
(1,2)	0.9071; 0.9481	15
(1,3)	**0.9102; 0.9501**	30
(1,4)	0.9066; 0.9478	45
(1,5)	0.9084; 0.9491	60

According to the following ROC curve, if we choose a probabilistic algorithm, the best threshold (the value which maximized the Youden's J statistic[7]) for a review to be positive is 0.681.



5. Issues and possible improvements

By comparing the predicted labels of the development dataset with the real ones, it is possible to analyze the wrong predictions. I have noticed that there are two main groups of wrong predictions:

- Approximately neutral: reviews which contains both positive and negative sentences. Possible solution: assign a weight to each sentence (according to adjectives, punctuation, uppercases…). The final prediction would be positive if the summation of positive weights over negatives is above a threshold;
- Too short: reviews containing one or two sentences. Possible solution: use a different classifier with different hyperparameters.

To enhance the algorithm, it could also be possible to remove 1-grams from the tokens if that token will be included in another n-gram which includes a negation.

References

[1] P. Gamallo and M. Garcia, "A Naive-Bayes Strategy for Sentiment Analysis on English Tweets," Association for Computational Linguistics, 2014.

[2] C. Manning, P. Raghadvan and H. Schütze, "Introduction to Information Retrieval," Cambridge University Press, Cambridge, 2008.

[3] H. H. Parmar and G. Shah, "Experimental and Comparative Analysis of Machine Learning Classifiers," International Journal of Software Engineering and Knowledge Engineering, 2013.

[4] H. H. Parmar, S. Bhanderi and G. Shah, "Sentiment Mining of Movie Reviews using Random Forest with Tuned Hyperparameters," 2014.

[5] S. J. Livingston, B. S. Selvi, M. Thabeetha, C. Grena and C. S. Jenifer, "A Neural NetworkBased Approach for Sentimental Analysis on Amazon Product Reviews," International Journal of Innovative Technology and Exploring Engineering, 2019.

[6] A. Muscolino and S. Pagano, "Sentiment analysis, a Support Vector Machine Model based on Social Network data," nternational Journal of Research in Engineering and Technology, Catania, 2018.

[7] J. Bergstra, R. Bardenet, Y. Bengio and B. Kégl, "Algorithms for Hyper-Parameter Optimization," 2011.

[7] https://en.wikipedia.org/wiki/Youden%27s_J_statistic