Data Science Lab: Process and methods

Politecnico di Torino

Project report
Student ID: s277760

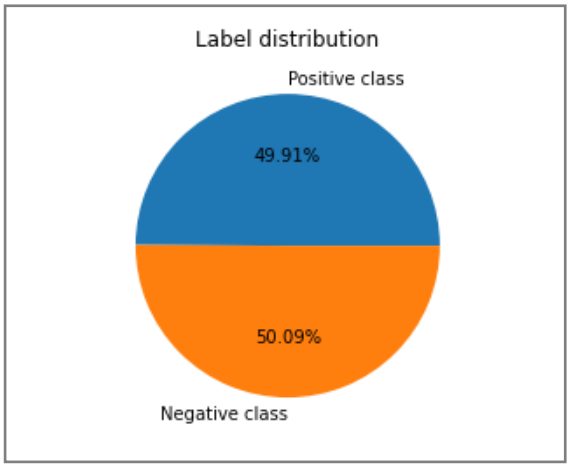
1. Data exploration

The aim of this project is to perform a sentiment analysis on contents from Twitter regarding the Black Lives Matter movement, by building a classification model which is able to predict whether users support the movement or not.

The dataset used consists in a collection of tweets (in English) published between May 27th and June 5th, located in [1]. More specifically, it is divided into two sets: a *development* set, characterized by 80,000 tweets labeled as positive (1) or negative (0), and an *evaluation* set composed by 20,000 unlabeled tweets.
Note that all graphs and observations are related to the *development* set only.
Each tweet can be characterized through a set of attributes, such as user, location, posting time, retweets count and the text. In this report, only the "full_text" attribute has been taken into account. Other attributes (such as "quoted_status_id" or "favorite_count") have been considered but they did not provide any improvement in the performance and thus they have been discarded.

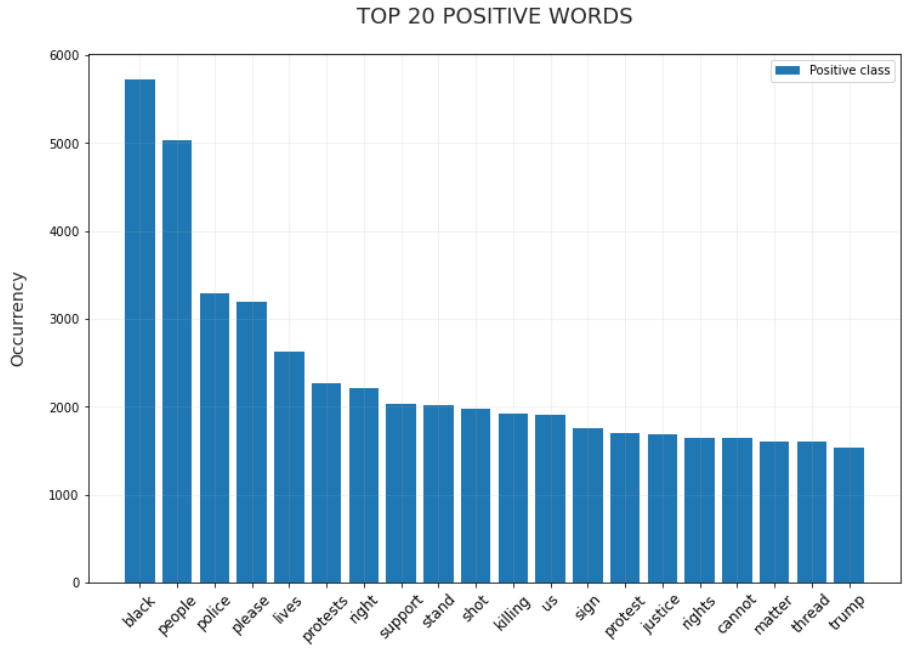
After a first analysis, the dataset presented some duplicates (same ID), which consequently have not been considered for building the model. The label distribution is the following:

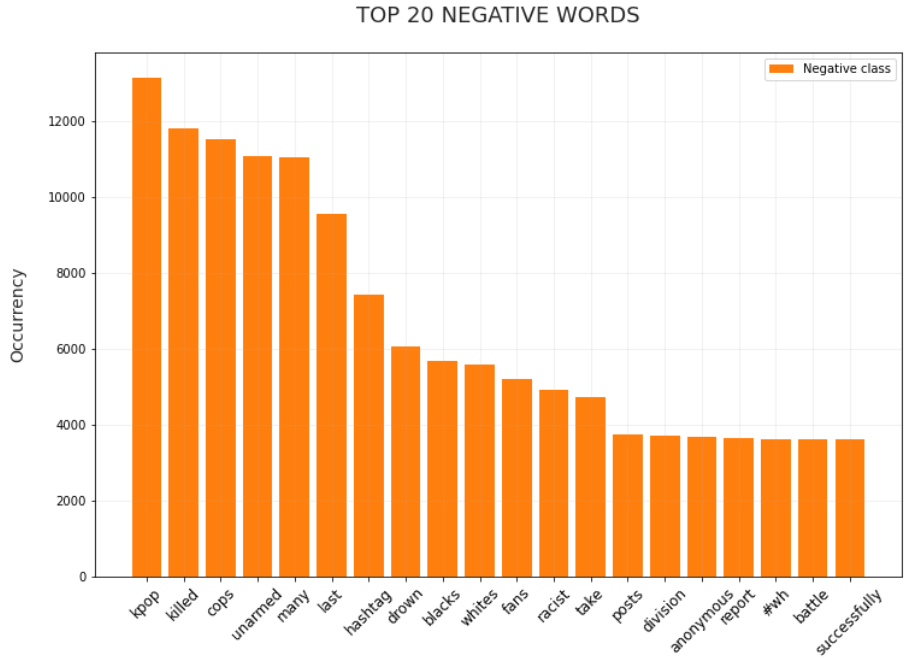
- *Development* set size: 79,960 (after removing duplicates)
- Number of positive tweets: 39,907
- Number of negative tweets: 40,053



Since the dataset is balanced in terms of positive and negative classes, the accuracy measure can be used for evaluating model's goodness. Moreover, one can expect the same distribution of classes among the *evaluation* set, since all tweets have been fetched from the same place.

The following two graphs show the top 20 words for each class (after removing stop-words but not applying stemming):

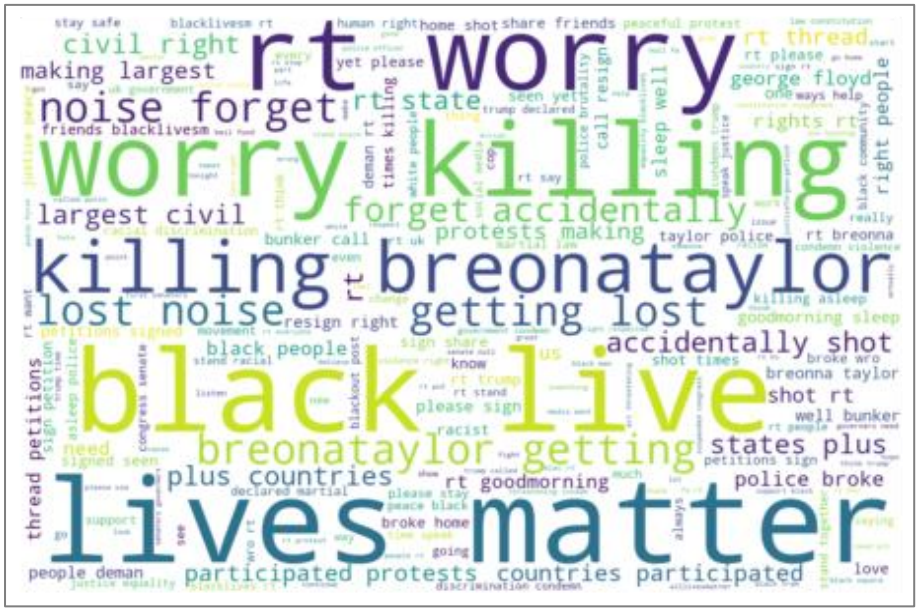




The data exploration has been carried out such that it is possible to infer on whether the tweets are coherent with the concept of “Black Lives Matter”. Therefore, some minor data cleaning adjustments have been made to disregard eventual special symbols which might bias the classification.

The results are broadly in line with expectations, but few inconsistencies can be spotted. For example, words that are semantically affine, as “police” (rated as positive) and “cops” (negative), if placed in specific contexts they assume opposite meanings. Indeed, this does not necessarily imply that whenever a tweet refers to cops is against the movement, nor for police to be a movement supporter.

The next two figures (wordclouds) represent the most frequent bigrams both for the positive and the negative class:



2. Pre-processing

Data preprocessing is a data mining technique used to turn raw input into good quality and significant data, that otherwise would lead to poor results in terms of performance. The text of each tweet has been tokenized word by word by splitting at each blank space. Each token has been subjected to several adjustment steps:

Case normalization
Each token has been converted to full lower-case characters. This has been done because, for example, words at the beginning of the sentence (characterized by the first upper case character) should be treated in no different way than the same word in lower case. This technique has been applied also to full upper-case words (e.g. BLACK → black) delivering better results, in contrast to the common logic that the use of full upper-case words usually relates to a strong tone for supporting arguments.

Data cleaning
Special characters (including punctuation and numbers), time measures (e.g. day of the week, month of the year, …), conjugated verbs (e.g. "I'm", "I've", "I'll", …) and URL links have been filtered out. Moreover, the string "\u2026" has been removed because it represents a Twitter bug that does not allow the platform to correctly show the three suspension dots "…". Additionally, strings starting with @ (indicating tags) have been considered because tags of specific users might be significant for classification, for example known public figures.
Hashtags (i.e. strings starting with #) and emojis have been kept for the analysis because the former are used to highlight key contents while the latter to strengthen personal opinions.

Stemming
Each token has been reduced to its base root form (e.g. killed → kill).

Stopwords elimination
The list of stopwords to be removed has been downloaded from the Natural Language Toolkit [2].
Stopwords refers to the most common words used in a language, meaning that they are much frequent in both positive and negative tweets, and consequently they are not able to identify a unique class. However, it has been noticed that words such as "not", "no" and "all" were meaningful to identify positive tweets, so they have been removed from the stopwords list.

Vectorization
The CountVectorizer approach [3] has been chosen for features engineering. It converts a collection of text documents into a sparse matrix of token counts. The ngram_range has been set to (1,3), allowing to tokenize the text document as unigrams, bigrams and trigrams.

3. Algorithm choice

Different classifiers (with default parameters) have been used for building the model and the one with the highest accuracy has been selected as best model.

Random Forest Classifier
Random Forest classifier is fast during training phase and provides a reasonable accuracy. Interpretability of results is not always easy due to high number of trees, but it provides global feature importance.

Logistic Regression Classifier
Logistic Regression classifier [4] is easy to implement, interpret and very efficient to train. It provides good performance in terms of accuracy.

K-Nearest Neighbors Classifier
K-Nearest Neighbors classifier is based on Euclidean distance. The training phase is very fast, but the model is slow during label assignment. It is not interpretable and suffers from curse of dimensionality.

SVC Classifier
SVC classifier (with linear kernel) [5] is not interpretable but it is suited for high dimensional problems and provides good performance in terms of accuracy.

Naïve Bayes Classifier
Naïve Bayes classifier is based on probability distributions, but it is not relevant for interpretability, speed or quality; it is useful when new data is available because probabilities can be easily updated without re-training the model.

The following table shows the average accuracy provided by the models and computed through Cross Validation Score with number of folders equal to 10 on the *development* dataset.

	Random Forest	Logistic Regression	KNN	SVC (linear)	Multinomial NB
Average accuracy	92.21%	**93.05%**	79.39%	92.63%	91.51%

The results obtained rank Logistic Regression as the best method, while the worst performing model is the KNN.

4. Tuning and validation

The hyperparameters that have been tuned for the Logistic Regression model are the following:

- *C*, which determines the strength of the regularization,
- *penalty*, that indicates the type of regularization used in the cost function,
- *solver*, which determines the algorithm used in the optimization problem.

In particular, the following values have been tested through a grid search:

- *C* = [0.5, 1, 1.5, 2, 2.5]
- *penalty* = ["l1", "l2", "elasticnet", "none"]
- *solver* = ["newton-cg", "lbfgs", "liblinear", "sag", "saga"]

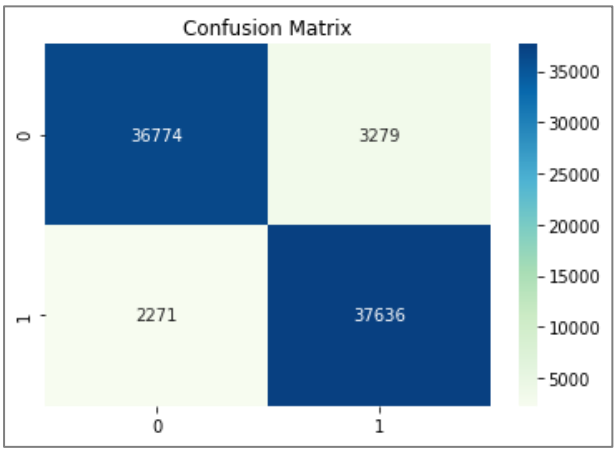
The tuning process has been performed by using the entire *development* dataset, by grid-searching over all the hyperparameters mentioned and evaluating each configuration on a 10-fold cross validation using accuracy as score. This approach takes into account the risk of overfitting because the model is trained and tested by using all available data.

The best configuration of hyperparameters that has been obtained is the following:

- *C* = 1.5
- *penalty* = "l2"
- *solver* = "lbfgs"

The average accuracy obtained using the optimal hyperparameters is 93.06%, approximately equal to the one achieved by training the classifier with hyperparameters at default values.

The graph below shows the corresponding confusion matrix, summarizing the distribution of correct and erroneous predictions:



Afterwards, the model algorithm has been applied on the *evaluation* dataset (unlabeled) and the resulting predictions have been submitted to the platform, in order to test whether the classifier is able to generalise well on new *unseen* data points.

The final accuracy score is 93.16%. Since the model displayed analogous results to the predictions on the *development* set, it proved to be well suited for generalisation on new non-processed data.

5. References

[1] Dataset. URL: https://bit.ly/DSL1920_exam_summer_dataset
[2] NLTK stopwords: http://www.nltk.org/nltk_data/
[3] CountVectorizer: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
[4] Logistic Regression classifier https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[5] SVC classifier https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html