Data Science Lab: Process and methods

Politecnico di Torino

Project report

# 1. Data exploration

The purpose of this assignment is to perform a sentiment analysis task, analysing user's textual review in order to understand if a comment includes a positive or negative mood.
The dataset contains more or less 40k reviews divided in two csv files. I have loaded the two files with pandas.
As we can see, the training data set consists in around 30k textual reviews labelled with 'pos' for positive sentiment or 'neg' for negative sentiment. Instead the data test contains around 10k reviews without label.

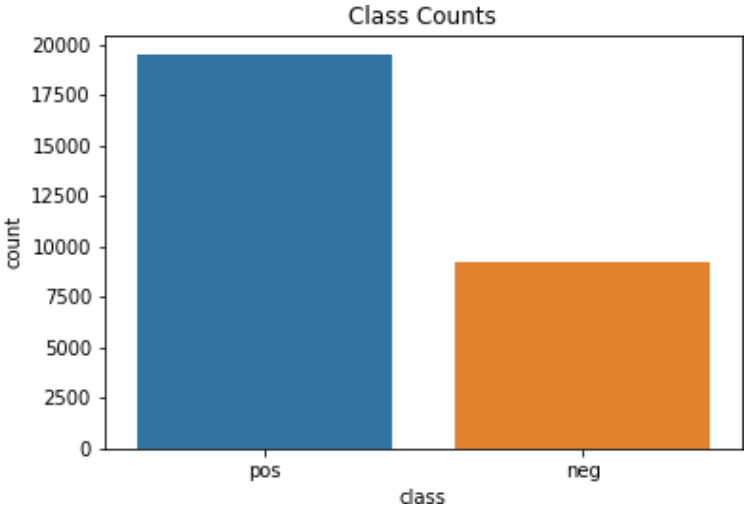


*Figure 1*

After verification on the dataset (if nothing is missing, and if anything was missing on the data) we count the number of elements for each 'class' and we plot using countplot from seaborn library. The Figure 1 tells that the problem is not perfectly balanced (approximately 68% pos and 32% neg). I also calculated the length of the reviews but I think is irrelevant for this analysis.

## 2. Preprocessing

The given data sets are comprised of very much unstructured reviews which should be pre-processed to make an NLP model. Pre-processing of data plays an important role, both in terms of organizing it into usable format, as well as reducing the number of features by removing redundant information. The pre-processing consists in:

- Removal of punctuations
- Removal of commonly used words
- Normalizing word

For this three step I have used spacy because the Italian vocabulary is most completed than other library as nltk.
I created a list of stop words, a list of punctuations and tokenizer function. Tokenizer function create token object, which is used to create documents with linguistic annotations, remove punctuations and return a list of tokens.
The stop words don't make sense in learning because they don't have connections with sentiments but spacy stop words list also contains words like non, male, buono etc. which shouldn't be removed because have an important role in the asses the mood of the reviews; for this reasons I have removed this kind of words from the stop word file (I also upload in the portal the stop words file modified) .
An important role have also the emoji can help to correctly asses the mood of the person when he is writing the comment.
Before to train I want to have numerically representation of the pre-processed data. This is done by applying a tf-idf algorithm to the set of textual reviews.
The tf-idf algorithm is implemented by the sklearn library.
As argument I specified:

- The tokenizer, useful to avoid different inflexions of the same term to be treated singularly. I passed the function spacy_tokenizer function we built as its tokenizer
- Stop_words removes stops words from the list of token returned by the spacy_tokenizer
- The max_tf and min_tf used to ignore words that are too common or too rare. Usually very common terms don't bring much information, while very rare terms are likely to be typos or non-characterizing word
- So ngram_range parameter sets the lower and upper bounds of the ngrams

The matrix obtained consists in 28k rows and a column for each distinct word present in the dataset.

## 3. Algorithm choice

I split the data in training set and test set in order to see how accurately the classification model performs.
I chose to split the data in 80% data training and 20% for the test set.
After the splitting, I have fit the training data with several classifier then in order to choice the best algorithm, I calculated the accuracy and the f1_score of each.
Accuracy refers to the percentage of correctly classified objects instead, the f1_score is a better measure in order to have a balance between precision and recall.

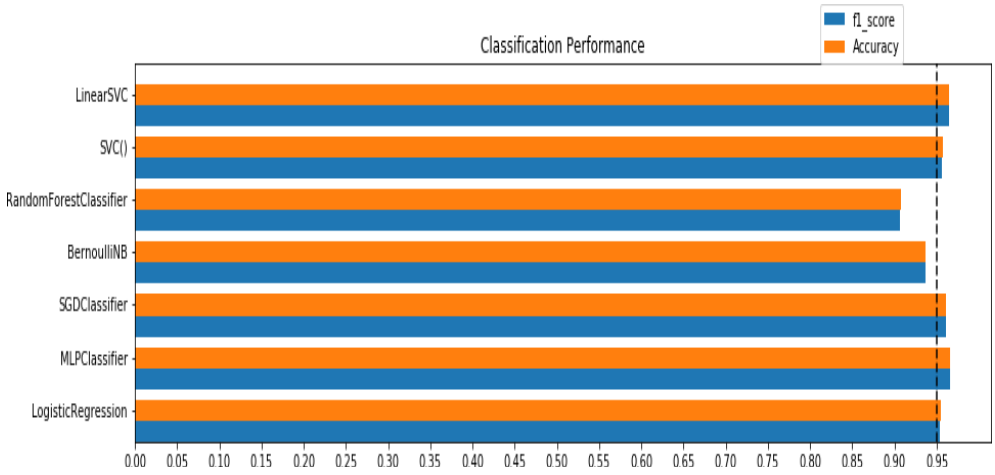


*Figure 2*

As we can see, from the Figure2, the best accuracy and the best f1_score are obtained by the Linear Support Vector Classification and by the Multi-layer Perceptron classifier. Instead, Linear Svc tends to be faster to converge respect Svc because Svc doesn't use linear kernel. I chose the Linear Support Vector Classification because is optimal for binary classification problem like this assignment.
After the choice of the classification algorithm I have create a pipeline with two component: a vectorizer and a classifier.
The vectorizer uses tidif objects to create the bag of words matrix for the data text. The classifier is an object that performs the by the Linear Support Vector to classify the sentiments. After all this step I can fit the model.

## 4. Tuning and validation

First of all, I have fit the model with the default parameter of the LinearSvc obtaining a good result.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| neg | 0.9548 | 0.9364 | 0.9455 | 1760 |
| pos | 0.9722 | 0.9805 | 0.9763 | 3991 |
| accuracy | | | 0.9670 | 5751 |
| macro avg | 0.9635 | 0.9584 | 0.9609 | 5751 |
| weighted avg | 0.9669 | 0.9670 | 0.9669 | 5751 |

The accuracy score is: 0.966962267431751

*Figure 3*

| | Negatives | Positives |
|---|---|---|
| Negatives | 1648 | 112 |
| Positives | 78 | 3913 |

The figure above shows that the accuracy is 96% and the f1_score is 96.6%
Then from the confusion matrix we see that there are a total of 1760 negatives reviews.
Of this 1760 negatives sentences 1648 were classified correctly while 112 (about 6%) were classified as positive sentence, creating some false positives. On other hand there is s 3913 positives reviews classified correctly and 78(about 2%) false negatives.
For the improvement of this scores and for avoid overfitting I used GridesearchCV that performing hyper parameter tuning in order to determine the optimal values for a given mode.
As argument I specified:

- param_grid Dictionary with parameters names as keys and lists of parameter settings to try as values (in my case I try with different value of C and tolerance)
- as scoring f1_score in order to improve the f1_score
- as estimator the LinearSVC
- cv determines the cross-validation splitting strategy in order to avoid overfitting. Specify the number of folds in a stratified KFold

I obtained the best cross-validation score of 0.98 with c=10 and toll=0.1.
The C parameter tells how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly.
At this point I set the parameter of the LinearSVC and I fit and predict respectively with the data training and the test set.
I obtained a numpy array that I wrote in a CSV file for the submission.