AI2-PRJ1

Violetta Gkika
1115201600222

November 25, 2021

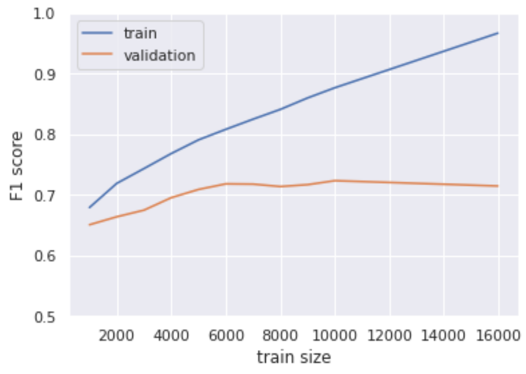
In this report I represent some details about the implementation and a summary of the testing process I did. More details about the implementation can be found on the notebook file.

Preprocessing

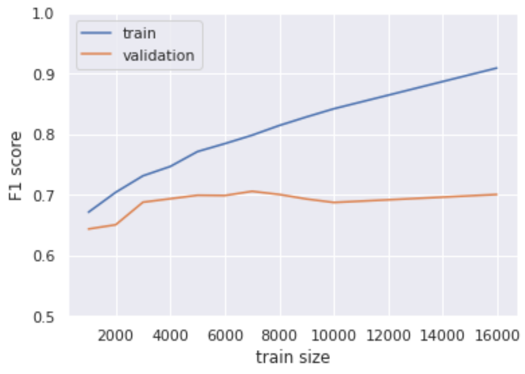
Using nltk library I lowercased all the tweets, removed tags, hashtags and links. Also filtered out useless data by removing stop words using nltk list for english stop words, and finally split the tweets in individual words.

Learning curves

To show how the model performs and show that it avoids underfitting / overfitting I plotted learning curves. For this step I trained the model 10-15 times, each time feeding the model about 1000 more data in each iteration . And what I observed is that the model performed better as the number of data was increased so I concluded that it is better to use all the train data to train the model. F1 score was used to measure the performance of the model.



(a) Before preprocessing

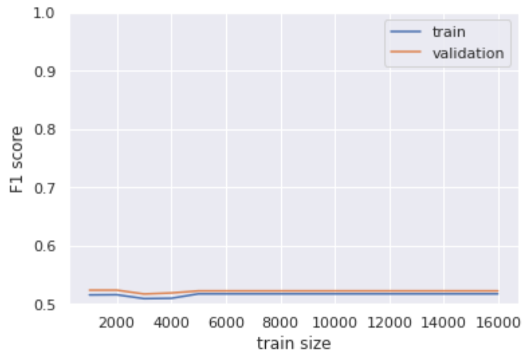


(b) After preproccesing

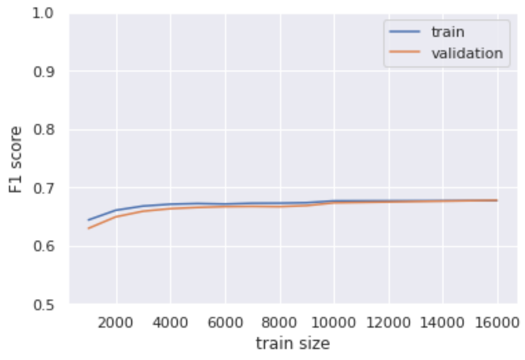
As we can see from the images independently from the preprocessing the model is clearly overfitting since we get a very high f1 score for the training data and a very low f1 score for the validation, compared with the training data. (large gap)
After preprocessing the data:
F1 score for validation is 0.7007011393514461
F1 score for training 0.9094266399599399

A brief “history” of experimenting

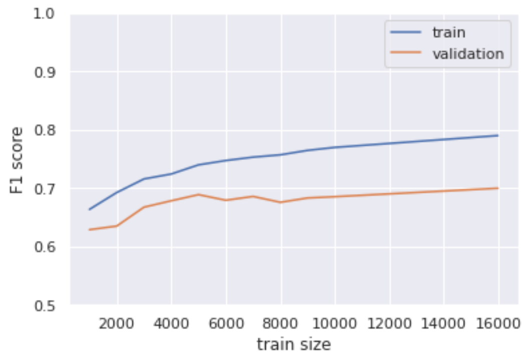
Using the Bag of Words model to vectorize the data and setting ngram-range=(1,2) in order to use both unigrams and bigrams one can see that not much change is reflected in the results, the model still overfits the training data. What had a considerate impact in the results, as one can see from the learning curve shown below is the min-df and max-df parameter.
For min-df = 0.1 training curve and validation curve has no gap but the accuracy is considerably low. Also the curve is very flat, that means the training data size has no impact on the model, and that is because a lot of data is considered as noise and is excluded. As we decrease the value min-df, so less data are ignored the results are better but if we decrease the value too much than not so much data are considered as noise and the model has a tendency to overfit so I concluded that a min-df between 0.005 and 0.01 would be better.



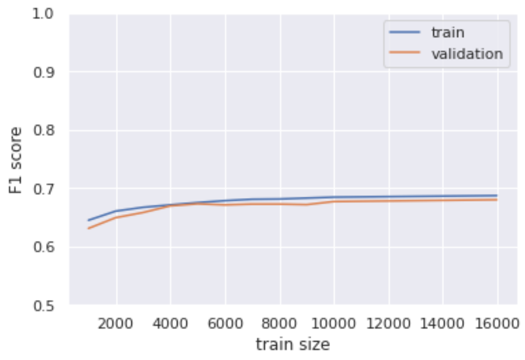
(a) min-df=0.1



(b) min-df=0.01



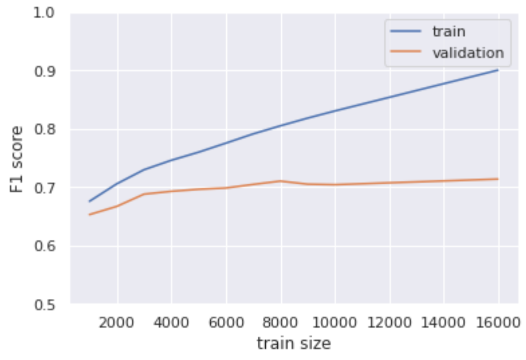
(c) min-df=0.001



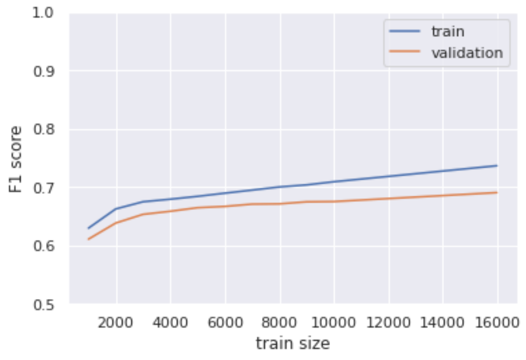
(d) min-df=0.007

Another parameter that had a great impact on the accuracy is the C parameter of Linear Regression function that represents the inverse of regularization strength.
Setting C parameter of logistic regression 0.01 one can see that the accuracy is around 0.7 with a very small gap between training and validation curve.
But setting a very small C or a very large C effects negatively the results.

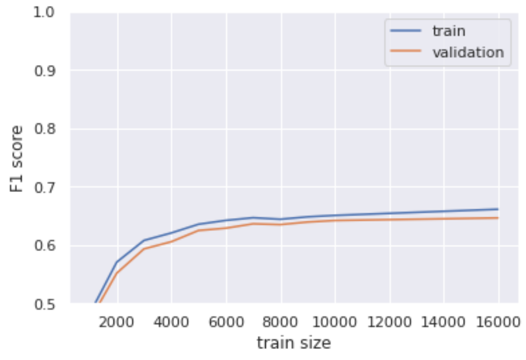
As we know C=1/λ, and λ controls the trade-off between allowing the model to increase it's complexity as much as it wants with trying to keep it simple. So for bigger values of C, we lower the power of regularization that means, the model increase its complexity and there is a higher chance to overfit the data. As we can observe from the learning curve in figure(a) (below), for C=0.1 the model clearly overfits.
For smaller values of C we increase the power of regulization and the model tend to underfit, because now we have a simpler model.
So as I mention earlier a value of C around 0.01 would be better.
Combining this observation with the one made earlier for min-df = 0.007 we can see from the learning curve that the model avoids overfiting and has a considerably "good" accuracy score.



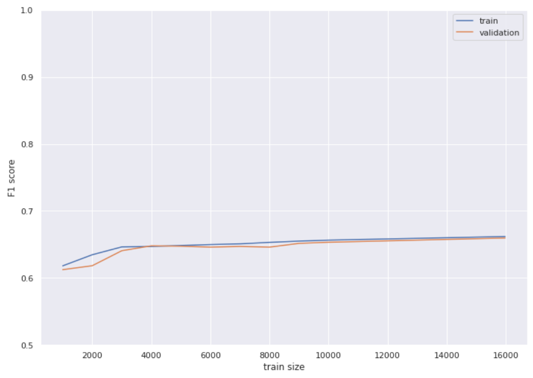
(a) C=0.1



(b) C=0.01

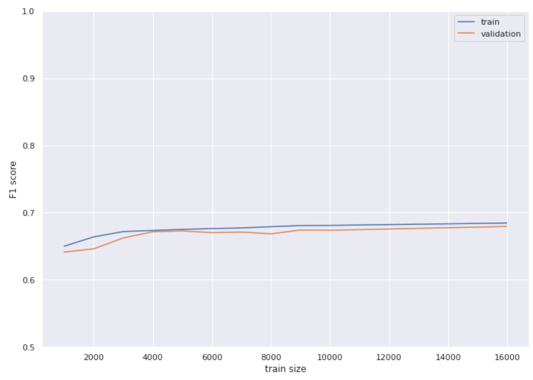


(c) C=0.001

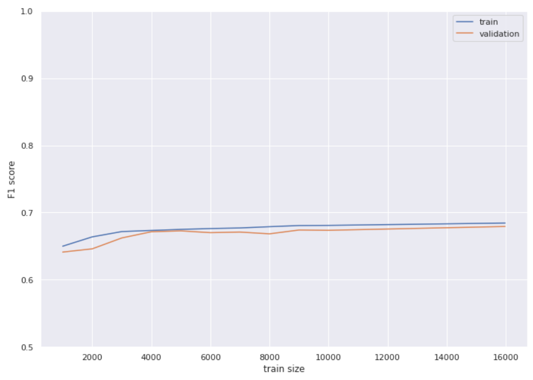


(d) C=0.05, min-df=0.007

I experimented also with some other parameters that I chose not to analyse in this report because they do not contribute so much on the training of the model. For example I chose to train the model with the default value of max-df parameter because generally ignoring data with a high frequency results in a very low accuracy.What is interesting in our dataset is that the max-df parameter did not have so much impact on the accuracy and that is probably because our tweets do not have so many common words.
Also penalty l1 gave very low accuracy because as it is stated in the documentation "As the value of coefficients increases from 0 this term penalizes, cause model, to decrease the value of coefficients in order to reduce loss" and also has some limitations.



(a) C=0.06,min-df=0.007,max-df=dafault



(b) C=0.06, min-df=0.007, max-df=0.5

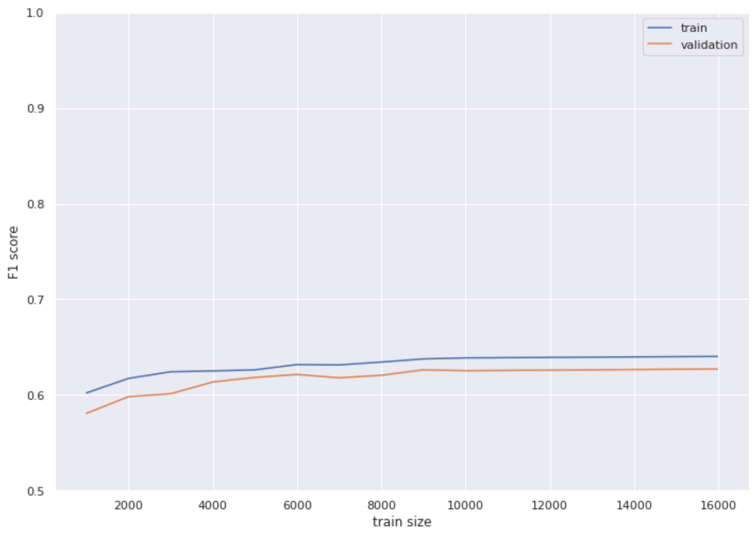


Figure 5: max-df=0.05

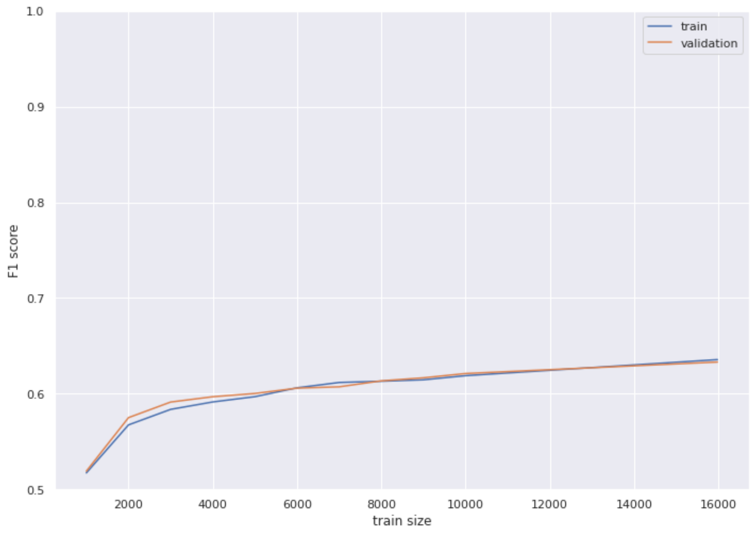
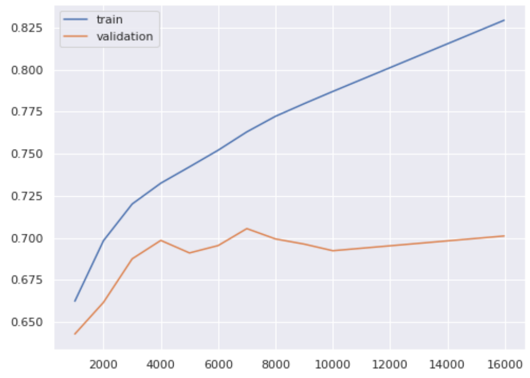
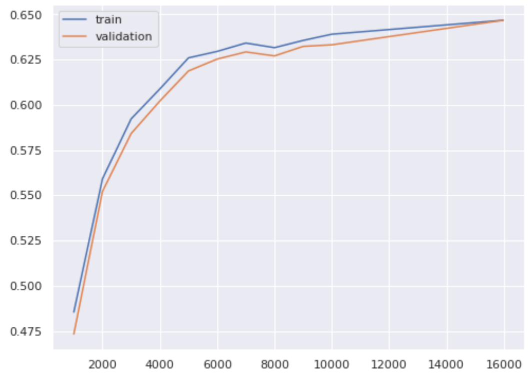


Figure 6: solver=liblinear, penalty=l1

Except from BOW vectorizer I experimented with TF-IDF vectorizes but the accuracy was lower compared to BOW.



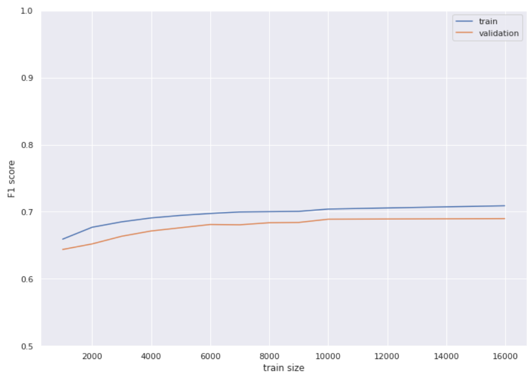
(a) TF-IDF with preprocessed data and no regulization



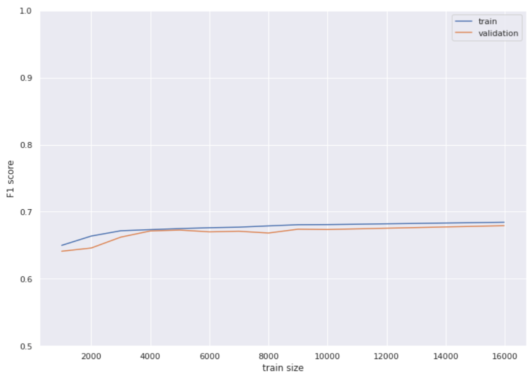
(b) TF-IDF with preprocessed data, C=0.06, min-df=0.007

Keep in mind that y axes has different values than the other graphics in the report.

If we try to train the model without applying any preprocess and setting the parameters C=0.06 and min-df=0.007 that worked best, one can see that there is no huge difference but still the results are slightly better after the preprocess.



(a) Before preprocess



(b) After preprocess