**Artificial Intelligence and Knowledge Engineering Laboratory**

**Assignment 4**

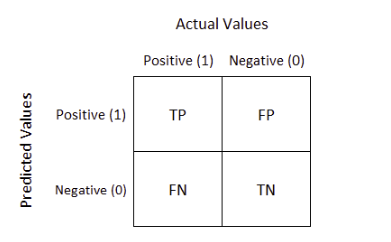
**Document classification**

**Naive Bayesian Classifier(NB)**

**Decision Trees(DT)**

Let’s start with the introduction to those algorithms. NB - It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Regarding the DT, it is the algorithm that aims to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data), where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

The accuracy can be defined as the percentage of correctly classified instances (TP + TN)/(TP + TN + FP + FN). where TP, FN, FP and TN represent the number of true positives, false negatives, false positives and true negatives, respectively. For good classifiers, TPR and TNR both should be nearer to 100%



Accuracy (all correct / all) = TP + TN / TP + TN + FP + FN

Misclassification (all incorrect / all) = FP + FN / TP + TN + FP + FN

Precision (true positives / predicted positives) = TP / TP + FP

Sensitivity aka Recall (true positives / all actual positives) = TP / TP + FN

Specificity (true negatives / all actual negatives) =TN / TN + FP

F-1 score is a measure of the accuracy of the test = 2 \* Recall \* Precision / Recall + Precision

The last one is Support which is the amount of tries for the class.

Recall and precision for each class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NB | alt.atheism | sci.space | comp.graphics | rec.sport.baseball | talk.politics.misc |
| REC\_I | 0.91 | 0.95 | 0.95 | 0.94 | 0.87 |
| PREC\_I | 0.92 | 0.9 | 0.92 | 0.99 | 0.92 |
| F1\_i | 0.914 | 0.924 | 0.934 | 0.964 | 0.894 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DT | alt.atheism | sci.space | comp.graphics | rec.sport.baseball | talk.politics.misc |
| REC\_I | 0.49 | 0.69 | 0.71 | 0.72 | 0.58 |
| PREC\_I | 0.55 | 0.66 | 0.67 | 0.82 | 0.51 |
| F1\_i | 0.518 | 0.674 | 0.689 | 0.761 | 0.542 |

Recognition algorithm selection:

|  |  |  |  |
| --- | --- | --- | --- |
| Fold | DT | NB | SGD |
| 1 |  |  |  |
| 2 | 0.63 | 0.895 | 0.871 |
| 3 | 0.63 | 0.895 | 0.877 |
| 5 | 0.65 | 0.896 | 0.887 |
| 10 | 0.64 | 0.897 | 0.889 |
| Average Acc | 0.64 | 0.896 | 0.88 |

With feature selection / without

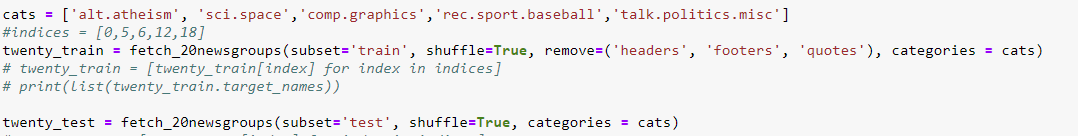
|  |  |  |  |
| --- | --- | --- | --- |
|  | DT | NB | SGD |
| With feature selection | 0.64 | 0.89 | 0.87 |
| Without f.s | 0.54 | 0.86 | 0.66 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | DT | NB | SGD |
| Gain | 78.2% | 78.5% | 38.2% |

|  |  |
| --- | --- |
| %train data | NB |
| 20 |  |
| 40 |  |
| 60 |  |
| 80 |  |
| 100 |  |

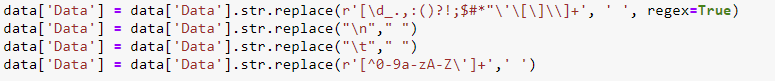
To determine the extraction of features, the TfidfVectorizer library is used, thanks to which we can not only determine how many words will appear in a given file, but also the frequency of a given word in a given document compared to other documents, i.e. using tf-idf (term frequency-inverse document frequency). To determine the extraction of features, the TfidfVectorizer library is used, thanks to which we can not only determine how many words will appear in a given file, but also the frequency of a given word in a given document compared to other documents, i.e. using tf-idf (term frequency-inverse document frequency).

Moreover, the one of import things is to read properly data.

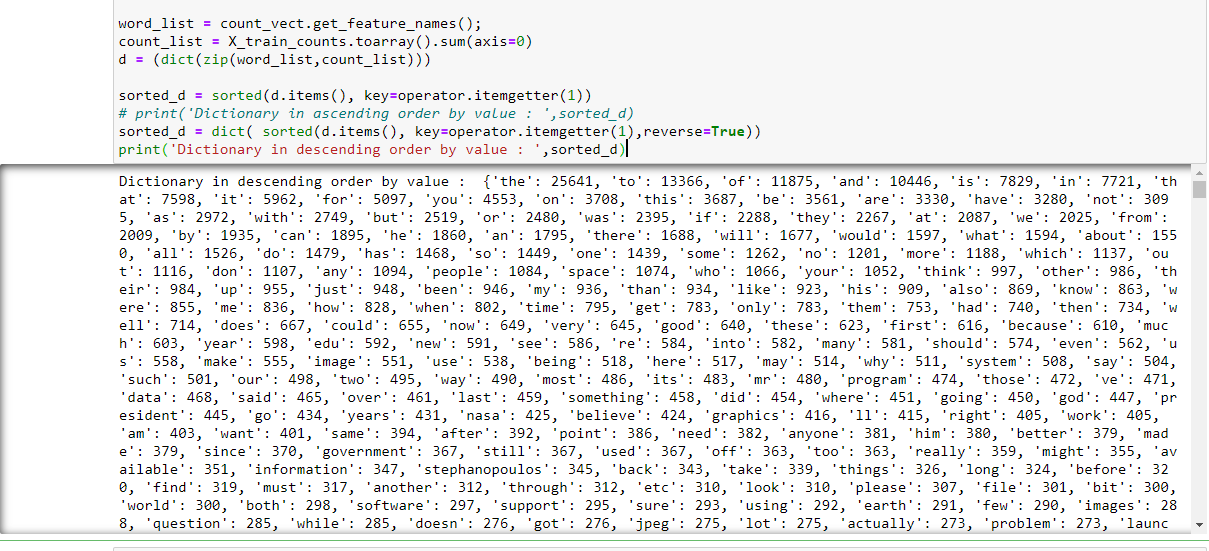


So, here we have categories which will be used and removed parts from the documents, but as the data after reading is the bunch, I would like to change it to the Dataframe to simplify our life and in easier way manipulate data. As well the goal is to get rid of useless symbols, so the regex is needed.

Furthermore, let’s add the new columns which will store the most frequent words, the name is “words”.

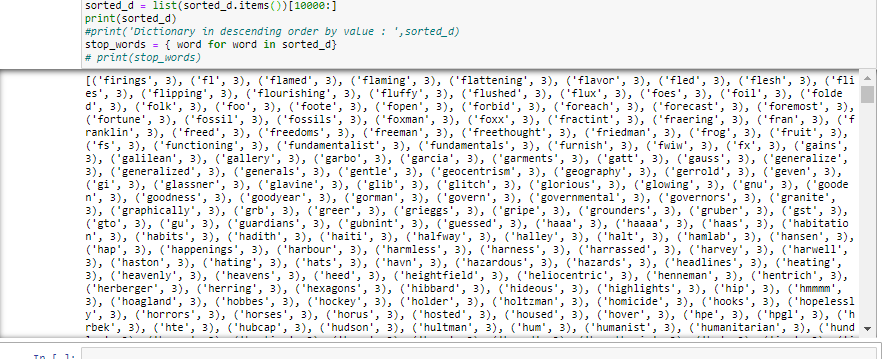


Count the frequency of words :

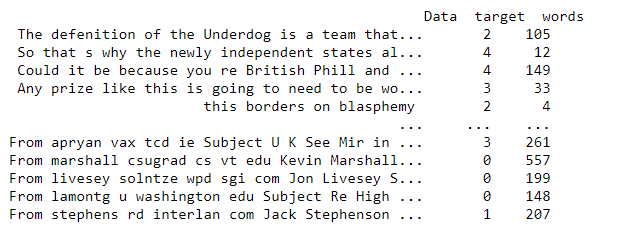


The length is 26357, so we have to select only 10000 words with the most frequency.

The words which have to be avoided.



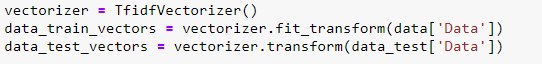
So, the data looks right now as :



CountVectorizer



TfidVerctorizer



Create and fit the algorithm : NB



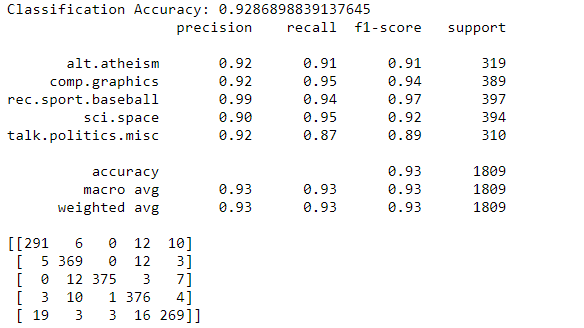
DT



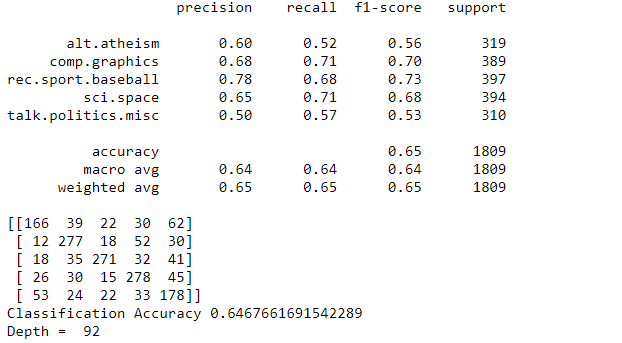
In further experiments, it would be compared with empty body and feature parameters, which has to improve the outcome, but as well may deteriorate it.

***Results***

**NB**

****

**DT**

****

**Table with algorithms without any parameters (empty body)**

|  |  |  |
| --- | --- | --- |
|  | NB | DT |
| Classification Accurancy | 0.92 | 0.65 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Tfidf** | **Tfidf with stop words** | **Count Vectorized** | **Count Vectorized with stop words** |
| **NB** | **0.9271** | **0.9287** | **0.9137** | **0.9223** |
| **DT** | **0.647** | **0.666** | **0.654** | **0.658** |

**Now, let’s use the cross validation**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cross validation / test data | 3 / 1 | 3 / 0.5 | 5 / 1 | 5 / 0.5 | 7 / 1 | 7 / 0.5 | 10 / 1 | 10 / 0.5 |
| NB | 0.87 | 0.88 | 0.87 | 0.89 | 0.88 | 0.91 | 0.9 | 0.92 |
| DT | 0.52 | 0.54 | 0.54 | 0.58 | 0.57 | 0.63 | 0.64 | 0.68 |

To get the best accuracy, best option it to decrease amount of test data and increase cross validation.

**Influence of Alpha for NB algorithm**

From 0.0001 up to 0.01 it increases, as the alpha = 0.01 is the peek of the max accuracy, as the alpha increases from this value the accuracy decreases.

**Influence of Entropy for DT**

|  |  |  |
| --- | --- | --- |
| Type | Gini | Entropy |
| Accuracy | 0.67 | 0.54 |

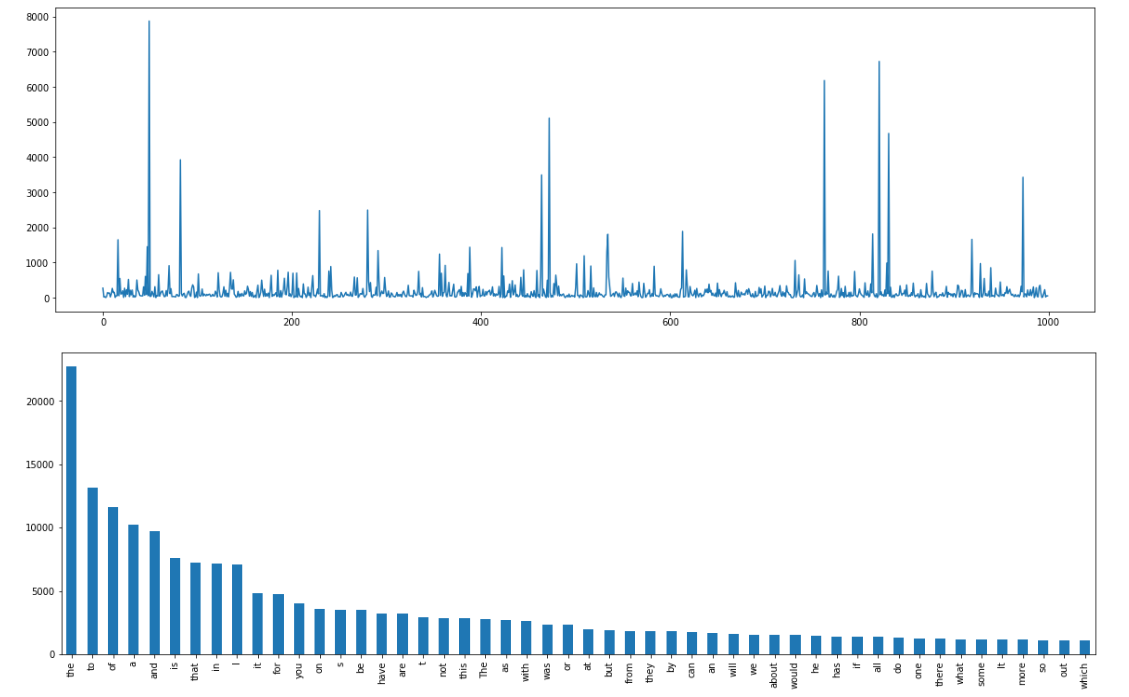
**Influence of max\_depth for DT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Max\_depth | 25 | 75 | 150 | 200 |
| Accuracy | 0.55 | 0.59 | 0.65 | 0.67 |

**Influence of min\_samples\_split for DT**

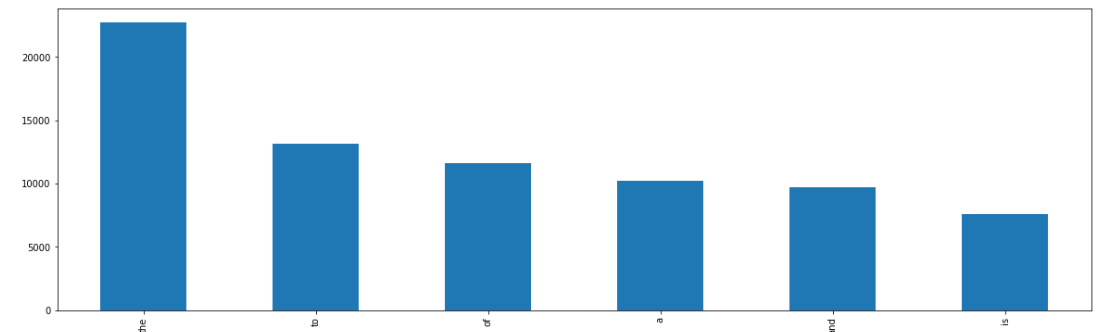
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **min\_samples\_split** | 2 | 3 | 7 | 15 |
| Accuracy | 0.655 | 0.651 | 0.623 | 0.689 |

***Plot***

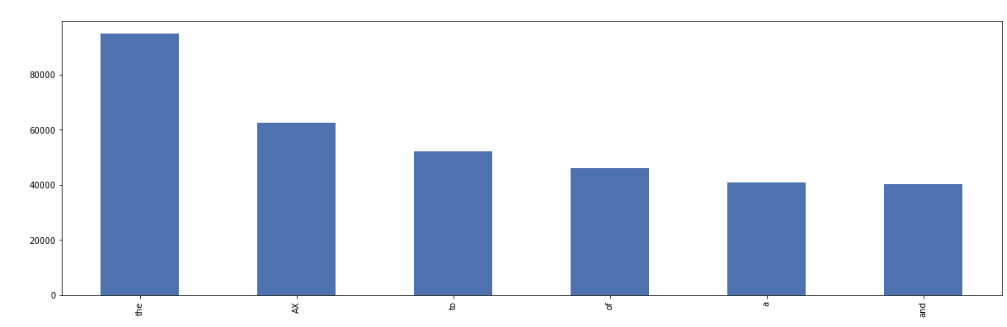
****

**The plots describes amount of words, starting counts from the 1st read file, so that’s why in 1st graph they are not in order way.**

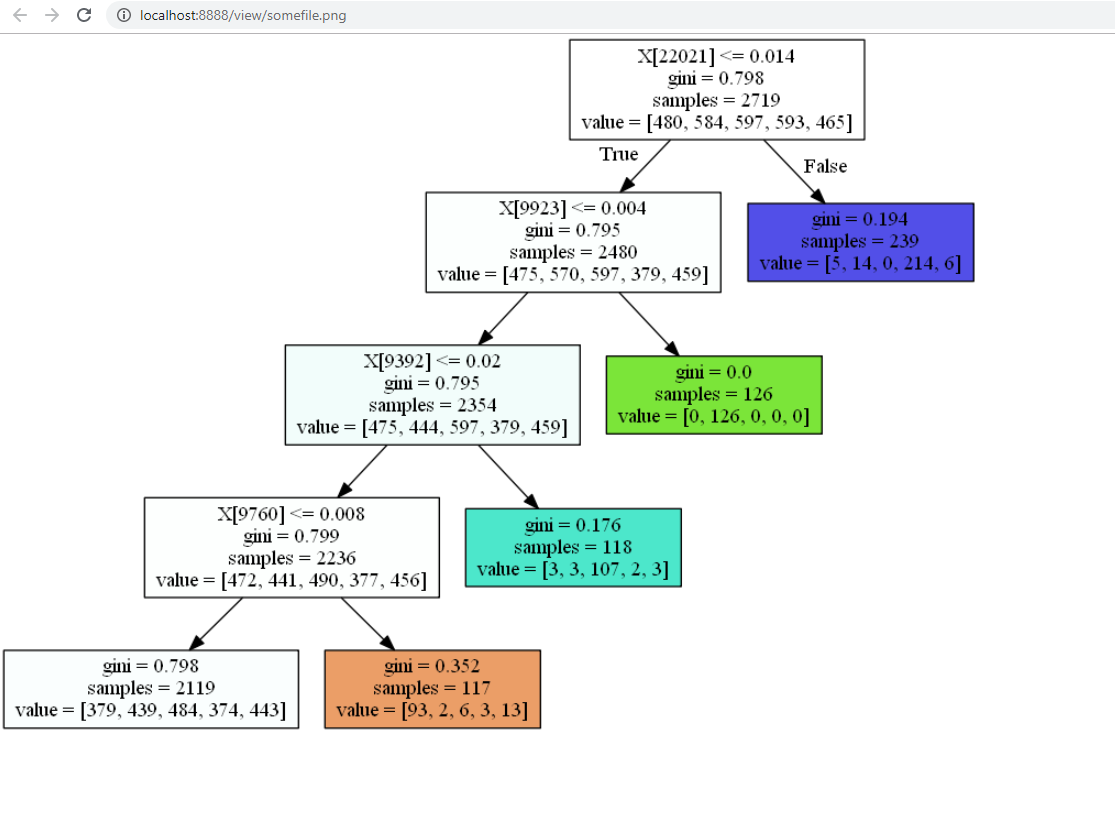
***The most frequent words***

****

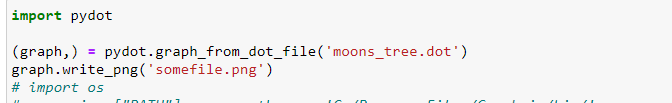
**Plot of the frequencies of the words for all documents (all categories)**

****

**The plot was created to clarify that for the selected categories the frequency of words is the same as it would be for another ones.**

****

**This graph was generated by those lines:**

****

I would like to mention, that documents which are selected does big influence on the final result of accuracy, cause it could as the data from 1 section , as from 5 different, so they communicate in differ way.

To observe the best parameters for the any algorithm I was using the gridsearchCV, and by executing command “grid.best\_estimator\_” the parameters were returned, so it helped me in the researches in further.

***Conclusion***

Both algorithms depends on the parameters while they initialized, so by researching and manipulating, I’ve found probably the best ones. By comparing those 2 algorithms, I can say that NB is better than DT, but this conclusion only for the current dataset, cause for the another one, DT could be much better, as it has own advantages, such as flexible or easy to debug, while NB Lesser overfitting and what was observed during experiments, that NB is much faster, and the last thing with extracting the data DT will be improving while NB gonna be worse.

During the writing the code, I’ve read many states, so I put the references to them.

<https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html>

<https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html>

<https://scikit-learn.org/0.16/datasets/index.html>

<https://medium.com/analytics-vidhya/decisiontree-classifier-working-on-moons-dataset-using-gridsearchcv-to-find-best-hyperparameters-ede24a06b489>

<https://raoumer.github.io/blog_posts/docs_classification_ML.html>

***With best wishes,***

***Vladyslav Gavryliuk.***