# Evaluation system for a formality detection model

In this document, I will briefly describe my mini-project of creating NLP solutions for the classification of formal and non-formal texts.

**Dataset**

The first step was to collect the dataset that l will use for training and testing the model. It was a big challenge because I couldn't find ready-made datasets that suited me on the Internet. I decided to create a dataset of texts myself, which contains 30 formal and 30 non-formal texts (N = 60). I know that the dataset is not too big, which is why the model had good results. The entire text dataset is in the file "*dataset\_training.tsv*". I also had problems with the csv file so I switched to tsv.

**Extensions and Libraries**

* import pandas as pd
* from sklearn.model\_selection import train\_test\_split
* from sklearn.feature\_extraction.text import TfidfVectorizer
* from sklearn.pipeline import Pipeline
* from sklearn.svm import LinearSVC

"pandas" for tabular display of my dataset.

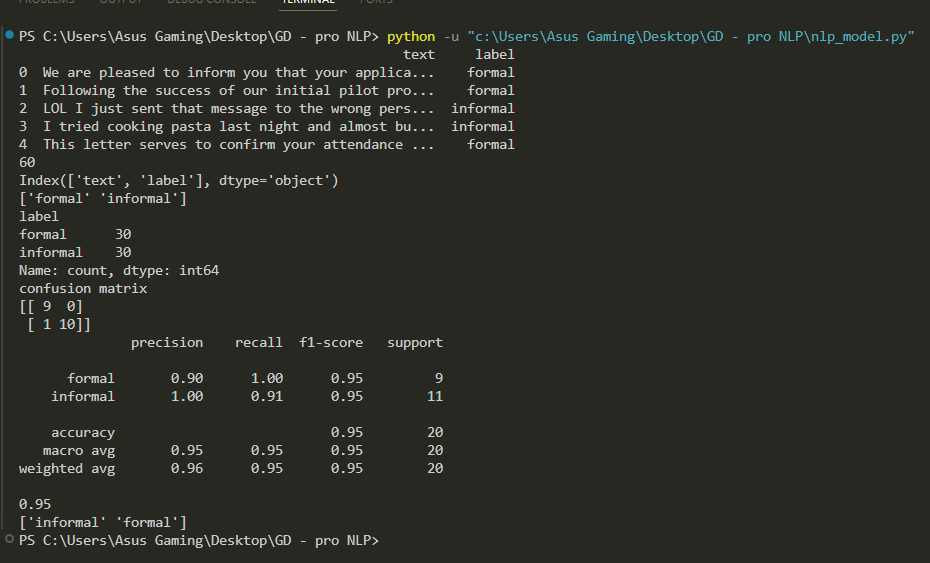
"skl - train\_test\_split " to separate the train and test data groups.

"skl - TfidfVectorizer " is vectorization of words.

" LinearSVC " is my model.

" sklearn.pipeline " is a structure that contains TfidfVectorizer and LinearSVC.

**Results**



**Model explanation**

We use *pipeline* to be able to vectorize the text and also create a good model that can study it. Text can't go directly into a machine model — you have to represent it numerically.

*TfidfVectorizer* takes text and for each word calculates how important that word is in that document relative to all the others.

TF (Term Frequency): how many times the word appears in the text.

IDF (Inverse Document Frequency): reduces the importance of words that appear in all texts (like "the", "is", "and", etc.)

The result is a vector of numbers describing each text, so the model can learn from it. ln this way, we also perform text normalization, as we would do with the help of the “*SpaCy*” library. We do tokenization, lowercase conversion and character stripping at the same time. We don't do lemmatization and we don't remove every stop-word.

*LinearSVC* is a model from the Support Vector Machines (SVM) family, it does classification by trying to find the best "boundary" (hyper-plane) between classes.

Why LinearSVC works well here:

* Works great with a large number of features (words)
* Generalizes well even when there are few examples
* It is fast and efficient for text classifications
* It is robust and does not get crowded easily
* The linear model is often good enough to separate formal and informal language (because the stylistic differences are obvious)

**Conclusion**

We got nice model performance parameters. Precision is 95%, accuracy 95% and recall 90%. It has only one mistake in which it predicted an informal text as formal, which is why it is not a higher score. Keep in mind that the model used about 40 texts for training and 20 texts for testing. Confusion Matrix is [9 0][1 10] which means that he made very few mistakes.

|  |  |  |
| --- | --- | --- |
| n = 21 | P/formal | P/informal |
| formal | TP = 9 | FN = 1 |
| informal | FP = 1 | TN = 10 |

I manually tested a model asking him various texts and he answered them all beautifully and accurately.