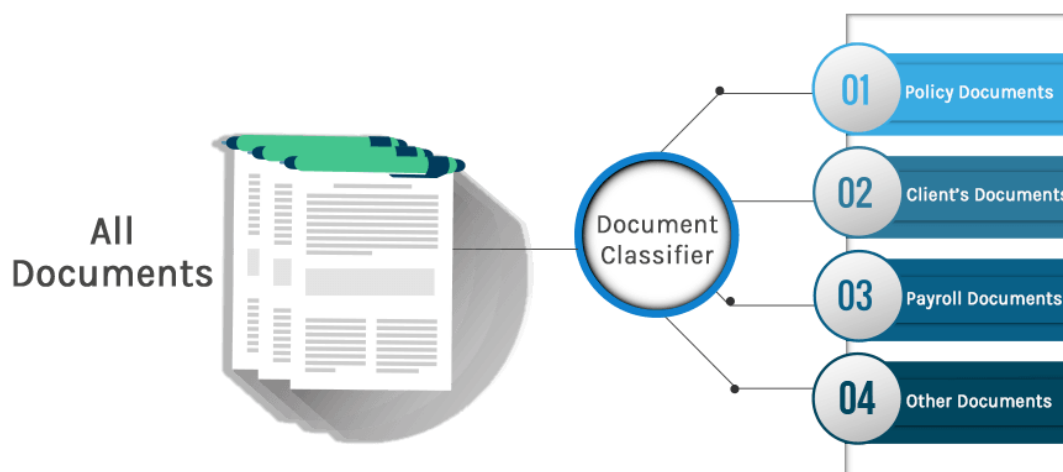




IT LICENSE PROJECT REPORT

Prototype of a document classification system



Submitted by :

KAHILIA Dounia.

ELHEIT Mohamed Wassim.

Supervised by:

MESKALDJI Khouloud.

Academic year: 2019/2020

Acknowledgement

It is with pleasure that we reserve these few lines as a sign of deep gratitude to all those who, directly or indirectly, have contributed to the success of this work. We particularly thank our supervisor Ms MESKALDJI Khouloud for her valuable recommendations and their availability.

Finally, we are also grateful to the jury members for accepting to assess our work.

Dedication

We would like to dedicate the fruit of this work to:

Our beloved parents.

Our lovely sisters and brothers.

And to each person who supported us during the realization of this project .

ELHEIT Mohamed Wassim

Summary

This Project is about allowing our computer to automatically classify a document, giving us his own class in a data base that was already classified (supervised mode). E.g.: a document about sport is going to be recognized directly by the Pc and shows us that it belongs to the << sport >> class.

Le projet a pour but de permettre à l'ordinateur de classifier automatiquement un document et lui donne sa propre classe dans une base de données déjà classée (mode supervisé).ex: un document parle du sport sera identifié automatiquement par le PC qu'il appartient a la classe << sport >>.

هذا المشروع يهدف إلى جعل الحاسوب قادرا على تصنيف الملف أوتوماتيكيا و إعطائه الصنف الذي ينتمي إليه في قاعدة بيانات قد تم تصنيفها . مثال : بإدخالك ملف يتحدث عن الرياضة سيتم التعرف عليه من طرف الحاسوب و يخبرنا أنه ينتمي إلى صنف الرياضة.

Table of contents

| | |
|--|----|
| Prototype of a document classification system | 1 |
| Acknowledgement..... | 2 |
| Dedication | 3 |
| Summary | 4 |
| List of abbreviations | 5 |
| List of figures..... | 6 |
| Table of contents | 7 |
| General Introduction..... | 9 |
| Chapter 1: document classification step by step | 10 |
| 1 The conception of the learning corpus: | 11 |
| 2The corpus representation: | 11 |
| 2.1 Countvectorizer : | 12 |
| 2.2 TF-IDF: | 12 |
| 3 The learning algorithm choice: | 14 |
| 3.1 Multinomial Naive Bayes: | 14 |
| 3.2 Bernouli Naive Bayes:..... | 14 |
| 3.3 Random forest algorithm: | 15 |
| 3.4 Linear svm: | 17 |
| Chapter 2: working steps..... | 19 |

| | |
|--|----|
| 1 Introduction | 20 |
| 2 Data Profiling..... | 20 |
| 3 Hunting Missing Values: | 20 |
| 4 Handling duplicate values in the dataframe | 21 |
| 5 Finding out the Value Distribution profile of the Category Variable..... | 21 |
| 6 Lexical Analysis of the Text Data..... | 21 |
| 7 Extracting features from text files..... | 22 |
| 8 Learning Classifiers, Making Predictions and Validating Results | 22 |
| Chapter3: The evaluation and deployment of the model..... | 27 |
| 1 Evaluation of the model :..... | 28 |
| 2 deployment of the model :..... | 30 |
| Conclusion..... | 37 |
| Refrences | 38 |

List of abbreviations

SVM : Support Vector Machine.

TF : Term Frequency.

IDF : Inverse Document Frequency .

ML : Machine Learning.

PC : Personal computer.

NLTK : Natural language Toolkit.

NLP : Neuro_linguistic programming.

NAN : Not A Number.

List of figures

| | |
|---|----|
| Figure 1 : Simple decision tree example | 16 |
| Figure 2 : Simple labeled data | 17 |
| Figure 3 : Random hyperplane | 18 |
| Figure 4: Best hyperplane..... | 18 |
| Figure 5 : Result of example's 1 classification | 28 |
| Figure 6 : Graphic interface | 30 |
| Figure 7 : Open a document on the interface | 31 |
| Figure 8 : Text shown in graphic interface | 32 |
| Figure 9 : Radio Buttons | 33 |
| Figure 10 : Classification with Linear SVM | 34 |
| Figure 11 : Classification with Random Forest | 35 |
| Figure 12 : Clearing text from text areas | 36 |

General Introduction

Document classification or document categorization is an example of Machine Learning (ML) in the form of Natural Language Processing (NLP). By classifying text, we are aiming to assign one or more classes or categories to a document, making it easier to manage and sort. This is especially useful for publishers, news sites, blogs or anyone who deals with a lot of content.

There are two classes of ML techniques: supervised and unsupervised.

In unsupervised method, the learning phase will do the training of classifier and its subsequent performance.

In supervised method, a model is created based on a training set. Categories are predefined and documents within the training dataset are manually tagged with one or more category labels. A classifier is then trained on the dataset which means it can predict a new document's category from then on.

In this project we are aiming to initialize with document classification and build a prototype of a document classification system using supervised method.

This project report is divided on three parts:

Chapter 1, talking about the document classification steps, adding examples from our project.

Chapter 2, creating our classification model. We chose the work environment, in our case Python which is rich with text processing libraries like NLTK. Then, we chose the Database to work on it. After some operations we got our model.

Chapter 3, creating our graphic interface in which we added our model. The interface can do the classification and show the result to the user.

Chapter 1: document classification step by step

Introduction:

The Automatic document classification is divided in general in many steps:

1. The conception of the learning corpus.
2. The corpus representation.
3. The learning algorithm choice.
4. The evaluation and deployment of the model.

We are going to present the three first points in this chapter. However we are going to talk about the evaluation and deployment of the model in the third chapter.

The conception of the learning corpus:

Many websites offer a free learning and test corpuses that are well structured for the realization of projects of automatic documents classification, written on Latin character.

Unfortunately, it's not the case for Arabic because we can't find any available free corpus, this is because the number of projects of automatic classification of document written in Arabic are very few comparing to those in Latin character.

The corpus representation:

Nowadays, there is no artificial learning method able to use the documents of learning corpus in their original aspect. So, we have to represent them into another form that allows the learning algorithms to use them.

The representation approach adopted in this these is to transform the corpus into a numerical array using Countvectorizer then into a numerical feature matrix using TFIDF vectorizer.

Countvectorizer:

Countvectorizer assign a unique number to each word as also know as(Tokenize) then counts the occurrence of each word , then it stores it as an array of shape (i,j)
i number of samples, j Number of features.

Example:

- We have a text : “The quick brown fox jumped over the lazy dog”.
- Assign a unique number to each word (Tokenize){ ‘the’ :7 , ‘lazy’:4, ‘jumped’:3 , ‘brown’:0 , ‘over’:5 , ‘quick’:6 , ‘dog’:1, ‘fox’:2}
- Features are(Vocabulary) [8 features] [‘Brown’ , ‘dog’ , ‘fox’ , ‘jumped’ , ‘lazy’ , ‘over’ , ‘quick’ , ‘the’]
- Counts the occurrence of each word[[11111112]]
- Stores it as an array and its shape is : (1,8) 1 number of samples and 8 number of features.

TF-IDF:

TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document and the inverse document frequency of the word across a set of documents.

TF-IDF was invented for document search and information retrieval. It works by increasing proportionally to the number of times a word appears in a document, but is offset by the number of documents that contain the word. So, words that are common in every document, such as this, what, and if, rank low even though they may appear many times, since they don't mean much to that document in particular.

However, if the word *Bug* appears many times in a document, while not appearing many times in others, it probably means that it's very relevant.

How is TF-IDF calculated?

TF-IDF for a word in a document is calculated by multiplying two different metrics:

- The term frequency of a word in a document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by length of a document, or by the raw frequency of the most frequent word in a document.
- The inverse document frequency of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.
- So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1. Multiplying these two numbers results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document.

To put it in more formal mathematical terms, the TF-IDF score for the word t in the document d from the document set D is calculated as follows:

- $tfidf(t, d, D) = tf(t, d) * idf(t, D)$

- Where: $tf(t, d) = \log(1 + freq(t, d))$
- And: $idf(t, D) = \log\left(\frac{N}{count(d \in D: t \in d)}\right)$

The learning algorithm choice:

The next step is to choose some learning algorithms and apply them on our corpus comparing the result of each one.

Naive Bayes Algorithm:

Naive Bayes is one of the most widely used classification algorithm in text mining applications. Based on Bayes theorem, this model makes the assumption that all the features are independent of each other and uses the probabilities of each attribute belonging to each class to make a prediction, There are two forms of Naive Bayes:

Multinomial Naive Bayes:

The multinomial model is designed to determine term frequency i.e. the number of times a term occurs in a document. Considering the fact that a term may be pivotal in deciding the sentiment of the document, this property of this model makes it a decent choice for document Classification. Also, term frequency is also helpful in deciding that whether the term is useful in our analysis or not.

Sometimes, a term may be present in a document many times which increases its term frequency in this model but at the same time, it may also be a stopword which potentially adds no meaning to the document but possesses a high term frequency so, such words must be removed first to gain better accuracy from this algorithm.

Bernoulli Naive Bayes:

In the multivariate Bernoulli Naïve Bayes Classifier algorithm, features are independent binary variables which represents that whether a term is present in the document under consideration or not . Being slightly similar to the multinomial model in the classification process, this algorithm is also a popular approach for text classification tasks but it differs from the multinomial approach in the Aspect that multinomial approach takes into account the term frequencies where as Bernoulli approach is only interested in devising that whether a term is present or absent in the document under consideration.

Random forest algorithm:

Random Forests belong to the ensemble learning methods for classification and regression tasks. In Random Forest, a subset of the training data is fit on a number of decision trees. Random Forests have the characteristic to minimize variance if it's there in the data-set. Their training time is generally quite higher which is one of their drawbacks to be used in production environment. However, the operations can be parallelized to reduce the training time. Also, greater the number of trees in the random forests, better will be the result however it comes with a tradeoff: the training time tends to increase as well. Also, if the number of trees tends to increase, the rate of improvement of accuracy may also tend to slow as well.

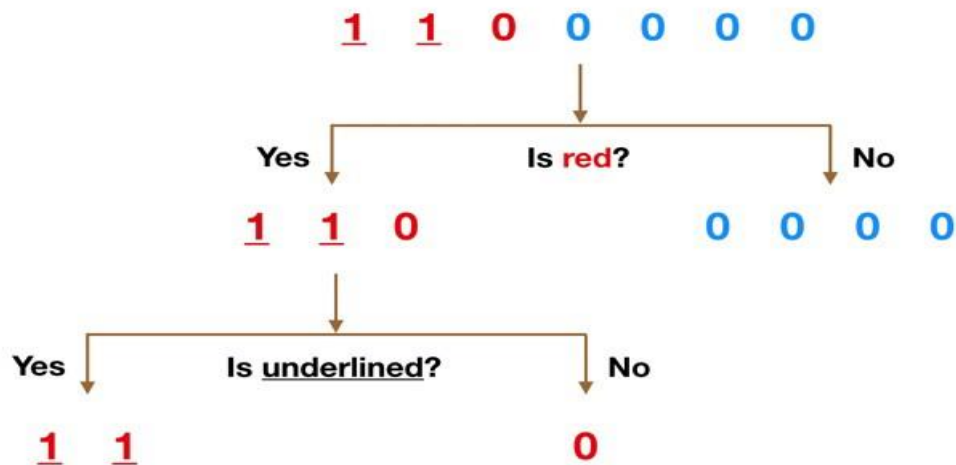


Figure 1 : Simple decision tree example

Support Vector Machine algorithm:

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model set of labeled training data for each category, they're able to categorize new text.

The basics of Support Vector Machines and how it works are best understood with a simple example. Let's imagine we have two tags: red and blue, and our data has two features: x and y . We want a classifier that, given a pair of (x,y) coordinates, outputs if it's either red or blue. We plot our already labeled training data on a plane:

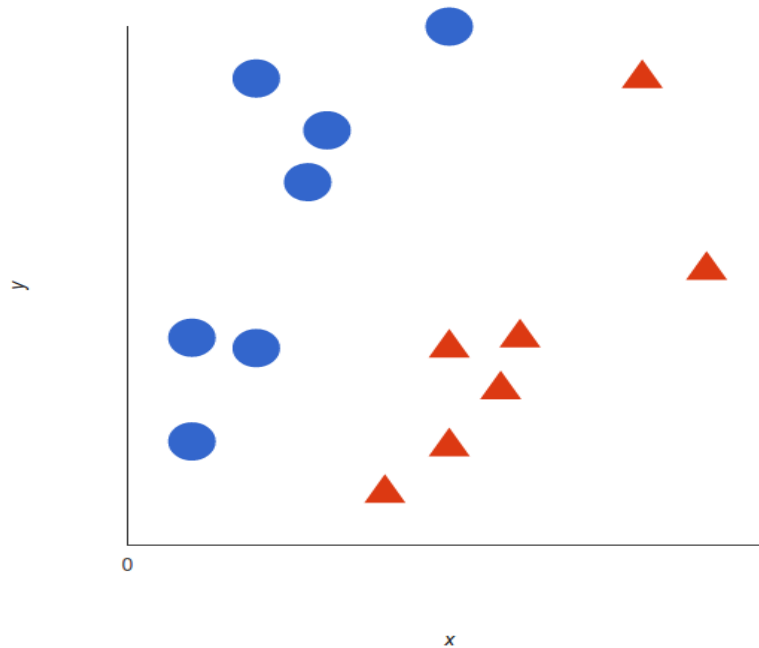


Figure 2 : simple labeled data

Linear svm:

A support vector machine takes these data points and outputs the hyper plane (which in two dimensions it's simply a line) that best separates the tags. This line is the decision boundary: anything that falls to one side of it we will classify as *blue*, and anything that falls to the other as *red*.

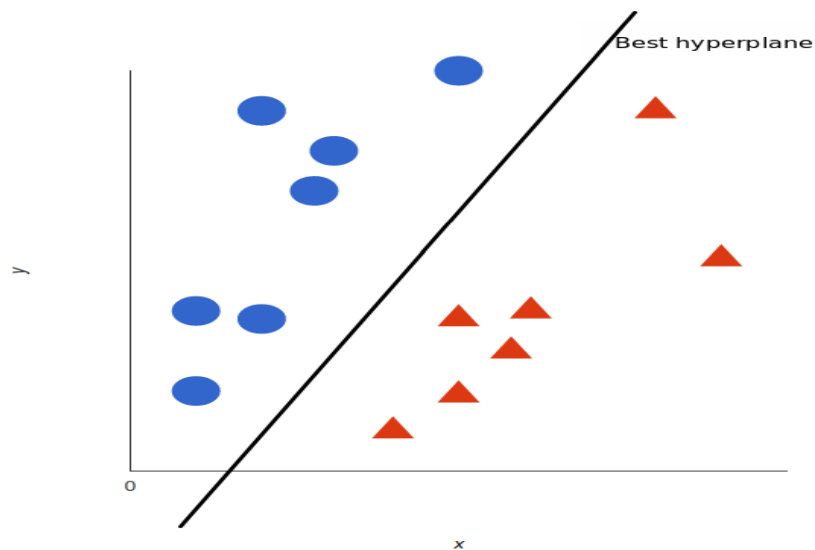


Figure 3 : random hyperplane

In 2D, the best hyperplane is simply a line

But, what exactly is *the best* hyperplane? For SVM, it's the one that maximizes the margins from both tags. In other words: the hyperplane (remember it's a line in this case) whose distance to the nearest element of each tag is the largest.

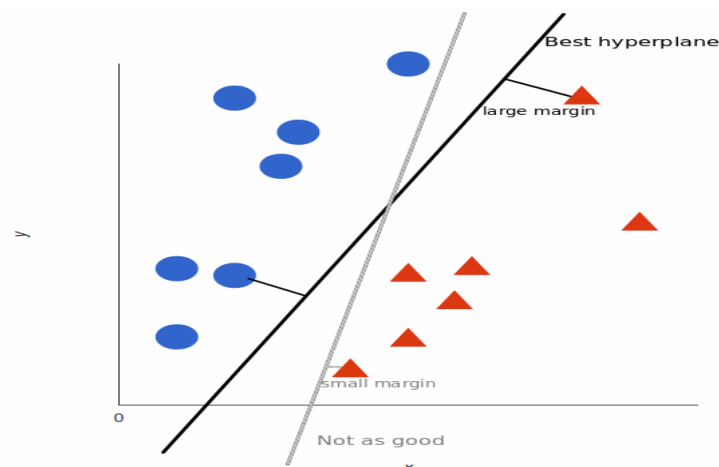


Figure 4: Best hyperplane

Chapter 2: working steps

Introduction:

The 20 Newsgroups data set is a collection of approximately 19997 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. To the best of my knowledge, it was originally collected by Ken Lang, probably for his Newsweeder: Learning to filter netnews paper, though he does not explicitly mention this collection. The 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering.. As the nature of the data was "text" so this project also involved extensive usage of text mining techniques as well. Text in its basic form is unstructured and to develop predictive models, the data needs to be thoroughly pre-processed. So the pipeline of developing models that I followed was.

Data Profiling:

The preliminary step in any data science project should be to develop familiarity with the data at hand. Transitioning right away to model development without knowing the basic characteristics of data can cause issues. So in this step, we computed a series of profiles so as to identify issues, perform quality checks and to get an idea of how different features of the data are distributed.

Hunting Missing Values:

Data-sets usually have missing values in them for a variety of reasons. In Numpy, missing values are represented as NaN and in our data we didn't find any missing values, in the event of finding missing values, we can drill down further and can adjust out.

Handling duplicate values in the dataframe:

Its also imperative to know that whether our data-set is properly balanced w.r.t class distribution or not. For classification, if the data points have somewhat uniform class memberships, then chances of bias is greatly reduced, where duplicate values can also appear in the dataframe and its also imperative to do this quality check before proceeding to the model development.

Finding out the Value Distribution profile of the Category Variable:

Its crucial to know in advance the basic statistics of the target/label variable so as to ensure that there is no issue of class imbalance there. For that, we need to calculate the count of each of the class label instances in our dataframe.

Lexical Analysis of the Text Data:

As text data's preprocessing transforms the words/terms into respective columns (though in much improved approach e.g. only frequent terms and also the rare ones (TFIDF) are transformed into columns), so its a good step to analyze the vocabulary richness of our data. If we just analyze the lexical richness of the text description data, it will still give us much idea about how rich our data is in terms of unique vocabulary words.

Data Pre-Processing:

The data in its raw form isn't always suitable for developing analytical models. To make the raw data compliant for analysis, pre-processing steps have to be performed. The pre-processing steps depend largely on the type of analysis that one intends to perform for instance regression, classification or clustering. In our particular case, we want to perform multi-class classification and the data at hand is text. So to perform pre-processing, the following steps are performed:

- Converting all of the data into lower case.
- Stemming the words so as to further reduce the feature size.

Extracting features from text files:

Text files are actually series of words (ordered). In order to run machine learning algorithms we need to convert the text files into numerical feature vectors. We will be using bag of words model for our example. Briefly, we segment each text file into words (for English splitting by space), and count number of times each word occurs in each document and finally assign each word an integer id. Each unique word in our dictionary will correspond to a feature Here we apply CountVectorizer algorithm, Then we create TFIDF features sparse matrix by fitting it on the specified corpus.

Learning Classifiers, Making Predictions and Validating Results:

With the data preprocessed, now is the time to develop the models. When it comes to developing machine learning models (and in our particular case, classifiers), we need to firstly train them on the labeled training data to learn from and then use the test data-set to make predictions. So to do that, we will proceed with splitting our existing data-set into training 70% and test 30% .

We are going to compare the following set of machine learning algorithms:

- Bernoulli Naive Bayes
- Multinomial Naive Bayes.
- Random Forests.
- Linear SVM.

This table shows values Accuracy Score for every algorithm That we have applied

| algorithms | Accuracy Score |
|-------------------------|----------------|
| Bernoulli Naive Bayes | 0.857 |
| Multinomial Naive Bayes | 0.921 |
| Random Forests | 0.939 |
| Linear SVM | 0.976 |

This table shows values Training Time for every algorithm that we have applied

| algorithms | Training Time |
|-------------------------|---------------|
| Bernoulli Naive Bayes | 0.884s |
| Multinomial Naive Bayes | 0.394s |
| Random Forests | 223.015s |
| Linear SVM | 25.253s |

This table shows values Accuracy Score for every algorithm That we have applied

| algorithms | Prediction Time |
|-------------------------|-----------------|
| Bernoulli Naive Bayes | 0.566s |
| Multinomial Naive Bayes | 0.069s |
| Random Forests | 6.724s |
| Linear SVM | 0.929s |

Comparing the tables, we find that each of Random Forests and Linear SVM have the best Accuracy Score value, where Random Forests belongs to the set of learning methods for classification and regression tasks. In Random Forest, a subset of the training data is fit on a number of decision trees. Random Forests have the characteristic to minimize variance in the data-set. Their training time is generally quite higher, which is one of their drawbacks to be used in the production environment.

However, operations can be done to reduce the training time. Also, greater the number of trees in the random forests, better will be the result; however it comes with a tradeoff: the training time tends to increase as well.

Also, if the number of trees tends to increase, the rate of improvement of accuracy may also tend to slow as well. We've used random forests with 100 trees. However, we can implement a routine to check the accuracy of the random forest by using different number of trees, and if we talk about Linear SVM, it has proved to perform well in large-scale and sparse machine learning problems.

Such problems are also encountered in text classification and natural language processing tasks and that motivates the fact to use this in our problem, So we used hinge loss which gives linear Support Vector Machine. Also set the learning rate to 0.0001, and if we talk about Naive Bayes is one of the most widely used classification algorithm in text mining applications. Based on Bayes theorem, this model makes the assumption that all the features are independent of each other and uses the probabilities of each attribute belonging to each class to make a prediction. The condition of independence may not be valid in many circumstances but as a base line model, its a good starting point to test its performance on the provided data. There are Bernoulli and Multinomial, where we note that the value Accuracy Score Multinomial Naive Bayes Very close of value Accuracy Score Random Forest, It remains valuable Accuracy Score Bernoulli Naive Bayes a few comparing in terms of Accuracy Score.

Conclusion:

So it can be gathered clearly from the tables that LinearSVM is clearly the winner in terms of strong accuracy score, training time and testing time. Random Forest classifiers' accuracy score is also quite appealing but the high magnitude of their training time is somewhat degrading, Where we will apply both of algorithms (Random Forests and Linear SVM) in our app.

Chapter 3: The evaluation and deployment of the model

Evaluation of the model:

We have our classification model ready. Now, it's time to evaluate it.

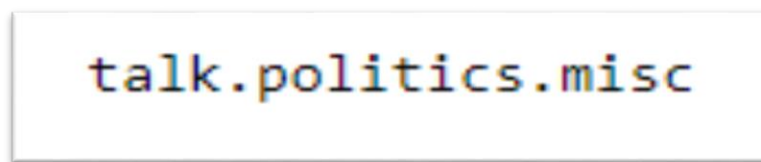
We choose some examples from our corpus and we classify them:

Example 1:

“I'm flattered by your invitation, but I'm afraid you have the wrong person.

Although I completely agree with your civil liberties agenda, I'm not in support of your economic agenda. What I DO like about the Libertarian party is that you guys are so good at shaking up the tired ideas of the past. I encourage you guys to continue your crusade, but I'm afraid I can't ride along. “

The text above is an example that we took from the (talk.politics.misc), and after the classification the result was:



talk.politics.misc

Figure 5 : Result of example's 1 classification

And it's exactly the same class!

Now let's do another essay

Example 2:

“Make it easy to plot real valued functions of 2 variables but I want to plot functions whose values are 2-vectors. I have been doing this by plotting arrays of arrows (complete with arrowheads) but before going further, I thought I would ask whether someone has already done the work. Any pointers??”.

This time the example taken from (comp.graphics) and the result after classification was also (comp.graphics)

After other essays we confirmed that our model is successful.

Deployment of the model:

In the first step we had to transform our code from (.iynb) used in Jupiter to (.py) so we can use it in Spyder.

The choice was Tkinter Library to create our graphic interface:

The result is:

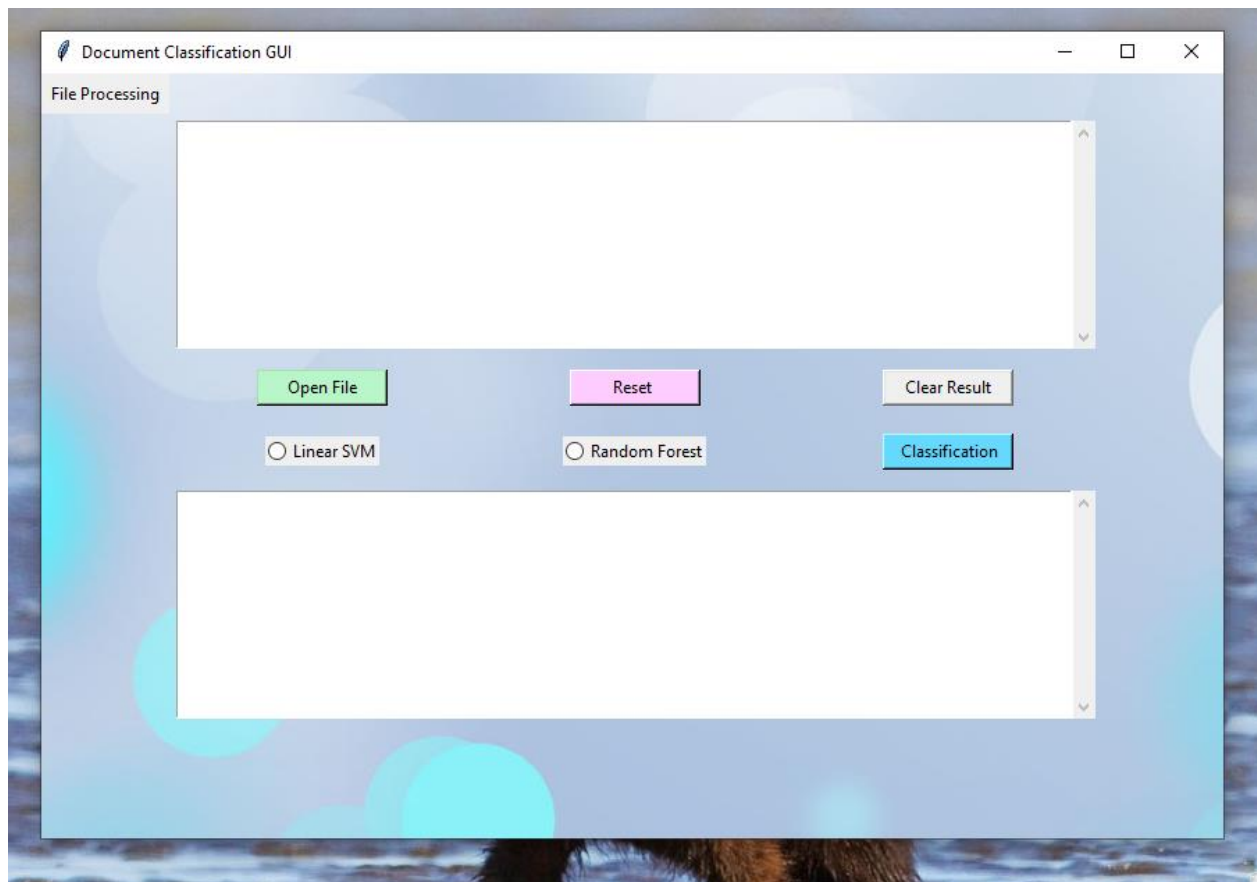


Figure 6 : Graphic interface

You can access to the document that u want to classify by clicking on the « **open file** » button and a window we'll appear .

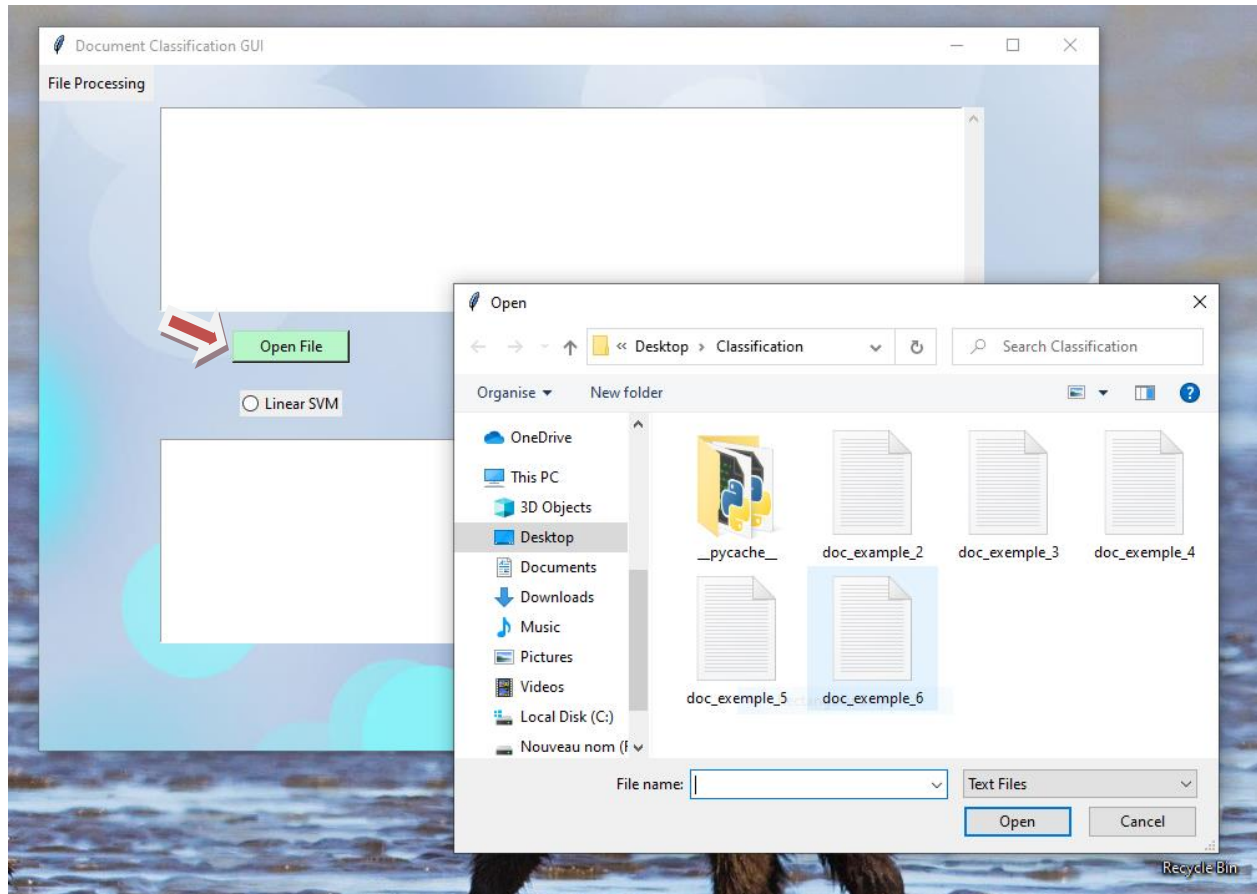


Figure 7 : Open a document on the interface

Then you select the document, the text will appear in the first Text area:

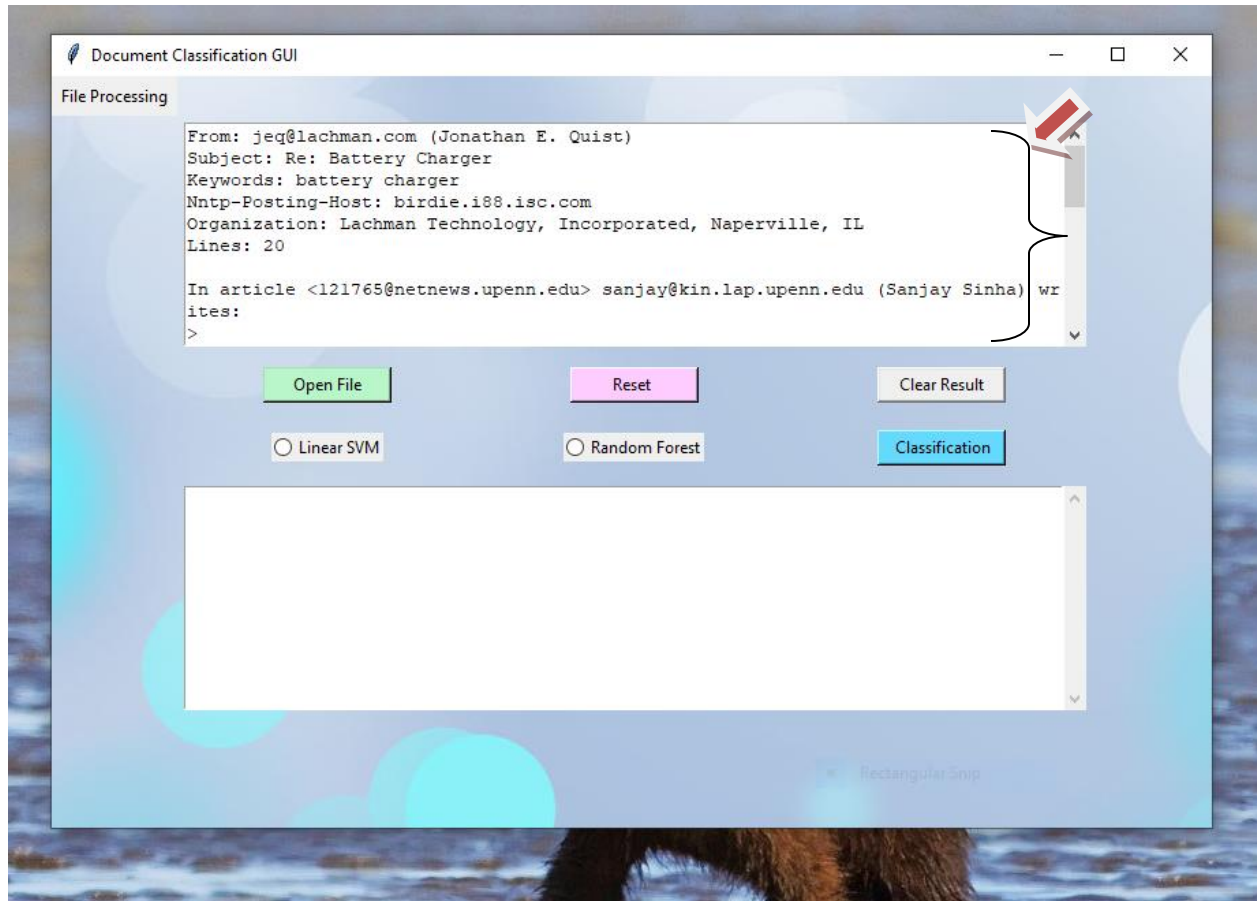


Figure 8 : Text shown in graphic interface

There are two radio Buttons using them you can choose algorithm to classifier your text document:

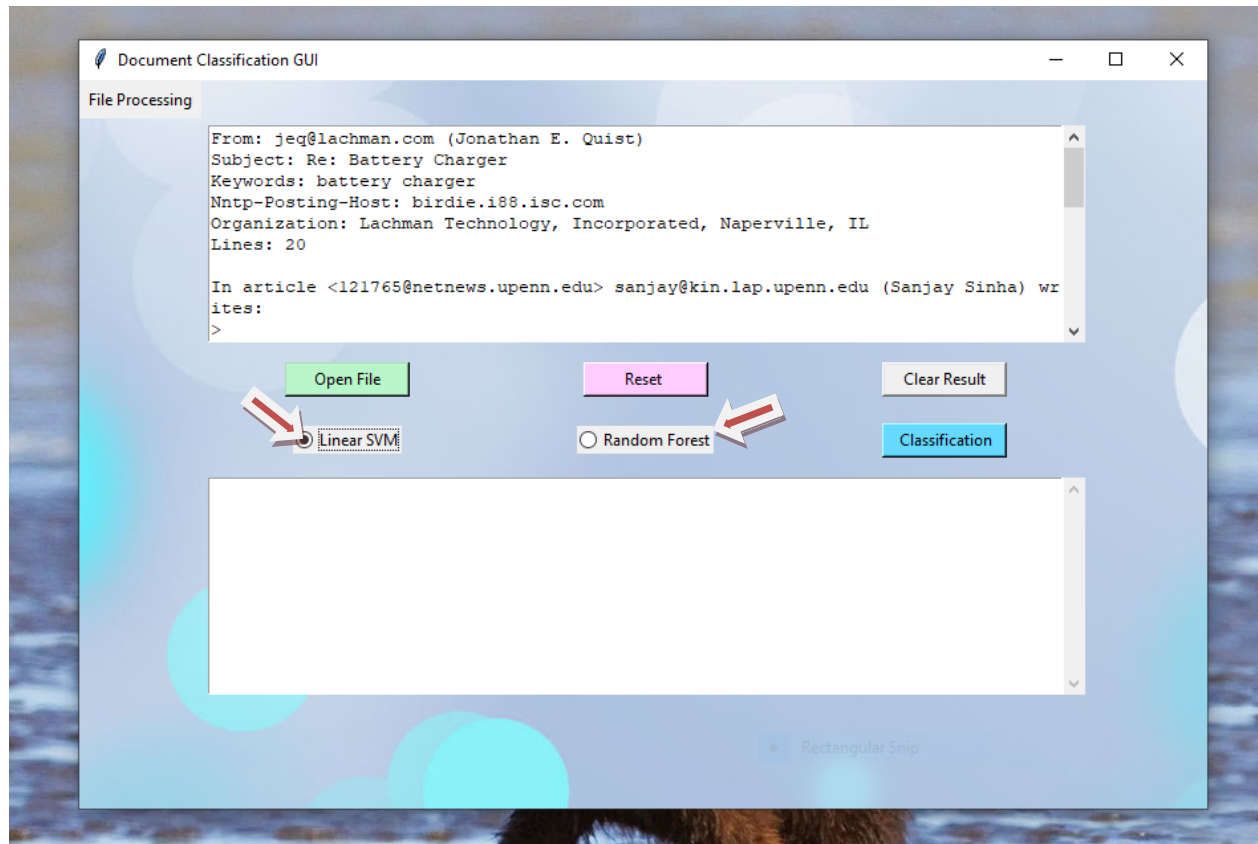


Figure 9 : Radio Buttons

For example, if you choose Linear SVM and click button classification to do our classification, then the result we'll appear in the second text area.

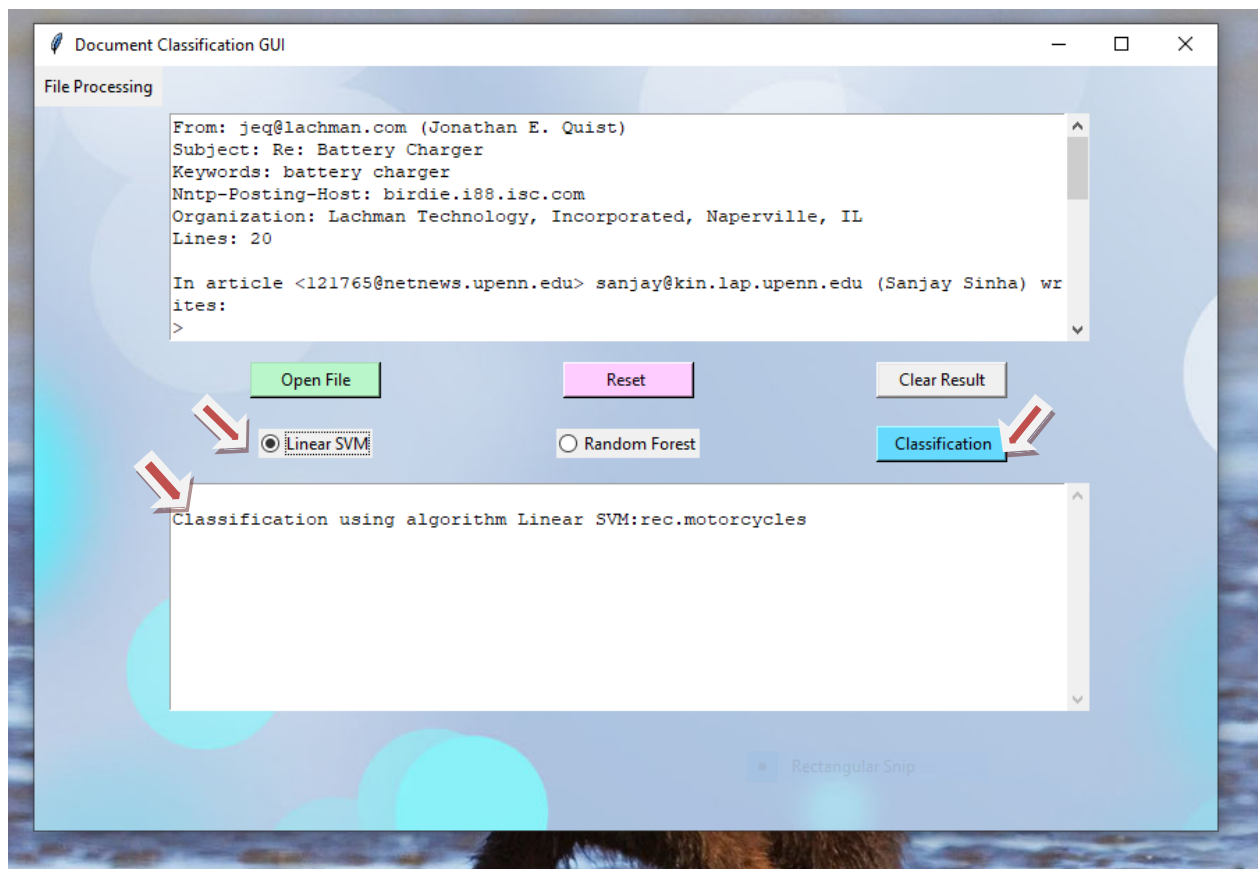


Figure 10 : Classification with Linear SVM

And the same for Random Forest :

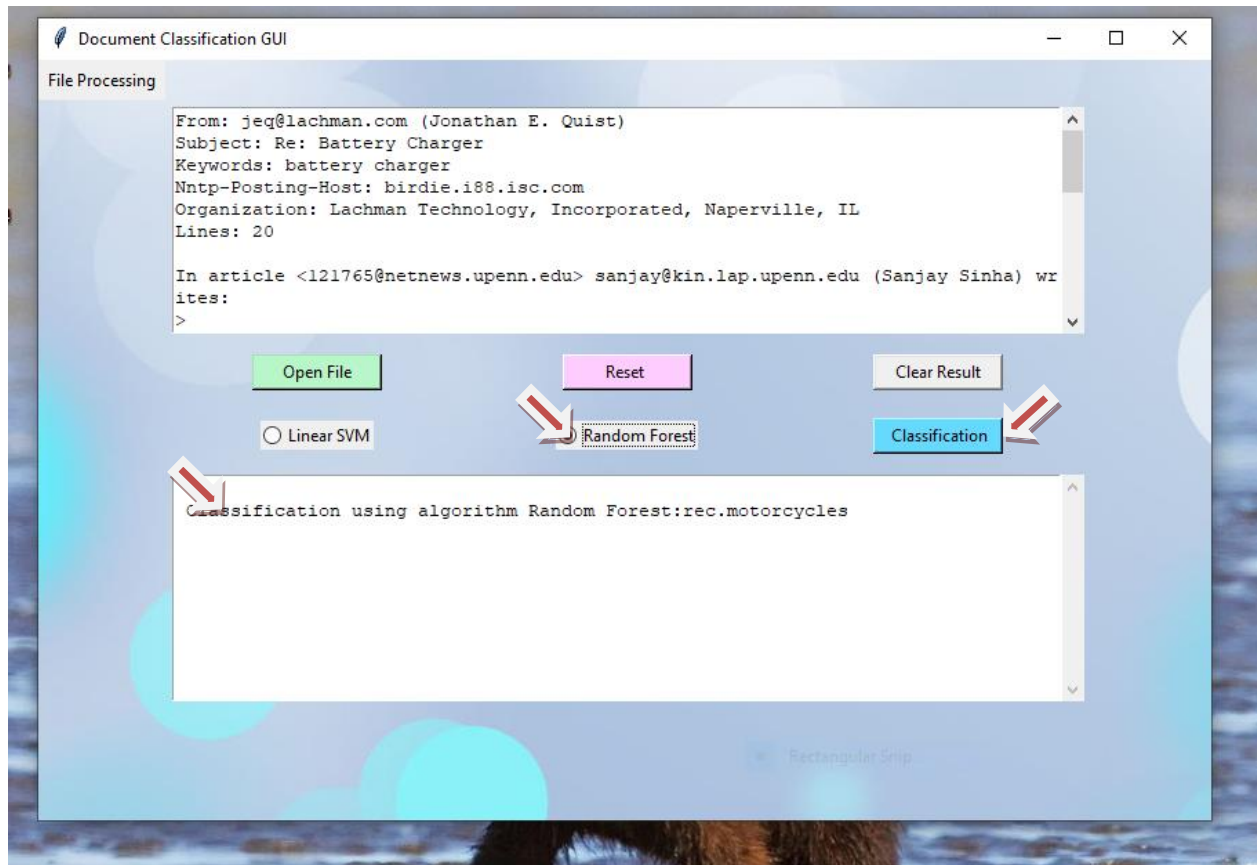


Figure 11 : Classification with Random Forest

You can also remove the text to add another one by clicking “**Reset** “,and You can also remove the result clicking « **Clear Result** »

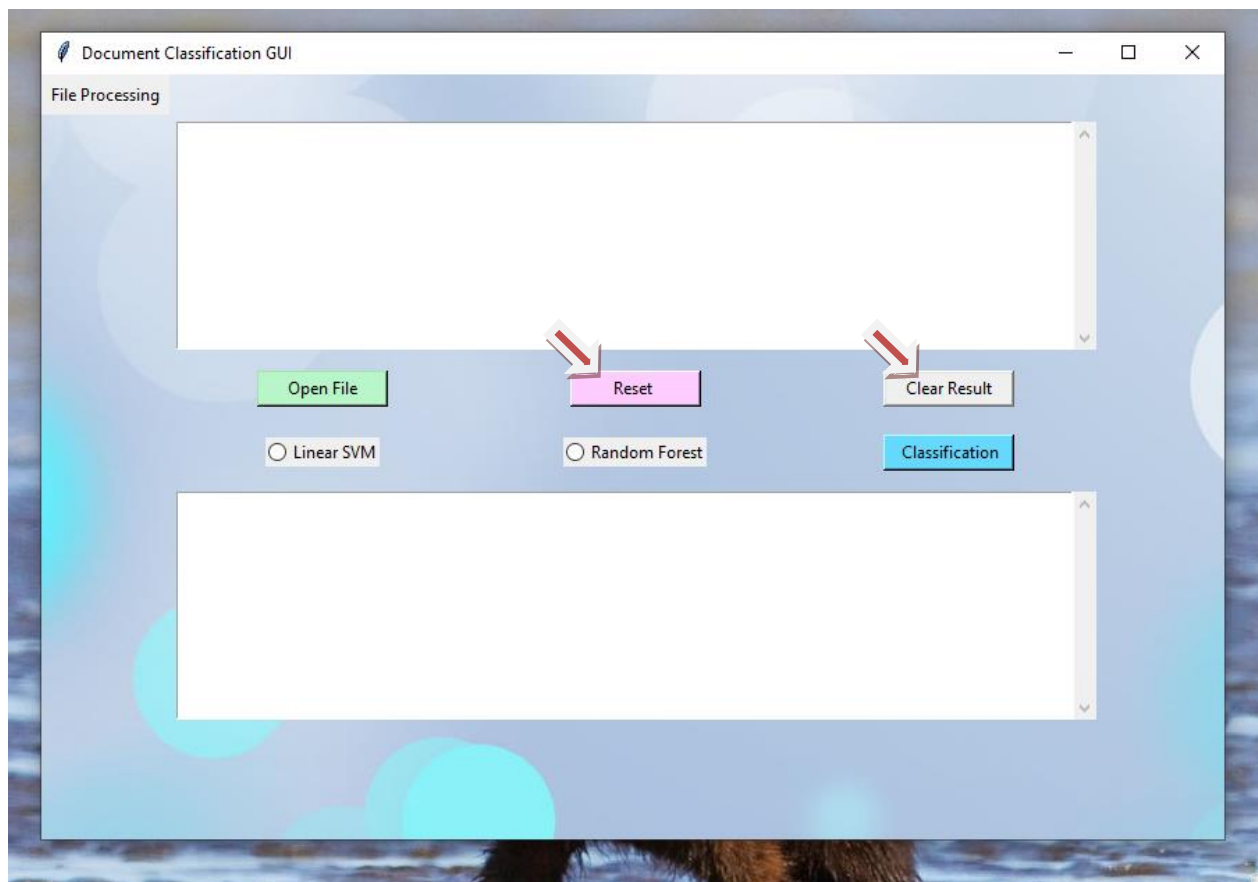


Figure 12 : Clearing text from text areas

Conclusion

In this report, we explained the different steps of our work done during the realization of this project in the aim of obtaining the Informatics License diploma.

In fact, we have started with a representative chapter where we tried to give an idea about the frame of our project. Then we have mentioned the steps used in the conception of our learning model. After that, the deployment into a graphic interface.

When realizing our project, we noticed that our project is very fruitful because it allowed us to be more familiar with document classification and enjoy the group work.

References

- <https://www.kdnuggets.com/2015/01/text-analysis-101-document-classification.html>
- <https://monkeylearn.com/blog/what-is-tf-idf/>
- https://www.researchgate.net/publication/334765873_Comparison_between_Multinomial_and_Bernoulli_Naive_Bayes_for_Text_Classification
- <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
- Chapitre 2. La Classification de Documents : Comment? *theses.univ-lyon2.fr* › pdfAmont › raheel_s_chapitre2
- https://fr.wikipedia.org/wiki/Classification_et_catégorisation_de_documents
- https://fr.talend.com/resources/what-is-data_profiling/#:~:text=Le%20processus%20de%20profilage%20permet,les%20données%20en%20connaissances%20exploitables
- <https://www.ionos.fr/digitalguide/web-marketing/analyse-web/analyse-tf-idf/>
- <https://le-datascientist.fr/les-algorithmes-de-naives-bayes>
- https://www.researchgate.net/publication/334765873_Comparison_between_Multinomial_and_Bernoulli_Naive_Bayes_for_Text_Classification
- <https://dataanalyticspost.com/Lexique/random-forest/#:~:text=L%27algorithme%20des%20«%20forêts%20aléatoires%20»%20a%20été%20proposé%20par,sous-ensembles%20de%20données%20différents.>
- [https://dataanalyticspost.com/Lexique/svm/#:~:text=SVM%20\(Support%20Vector%20Machine%20ou,ou%20de%20détection%20d%27anomalie](https://dataanalyticspost.com/Lexique/svm/#:~:text=SVM%20(Support%20Vector%20Machine%20ou,ou%20de%20détection%20d%27anomalie)
- <https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-svm.pdf>