Cairo University

Faculty of engineering

Computer dept.

Pattern project document

Submitted by: Mohamed Jamal

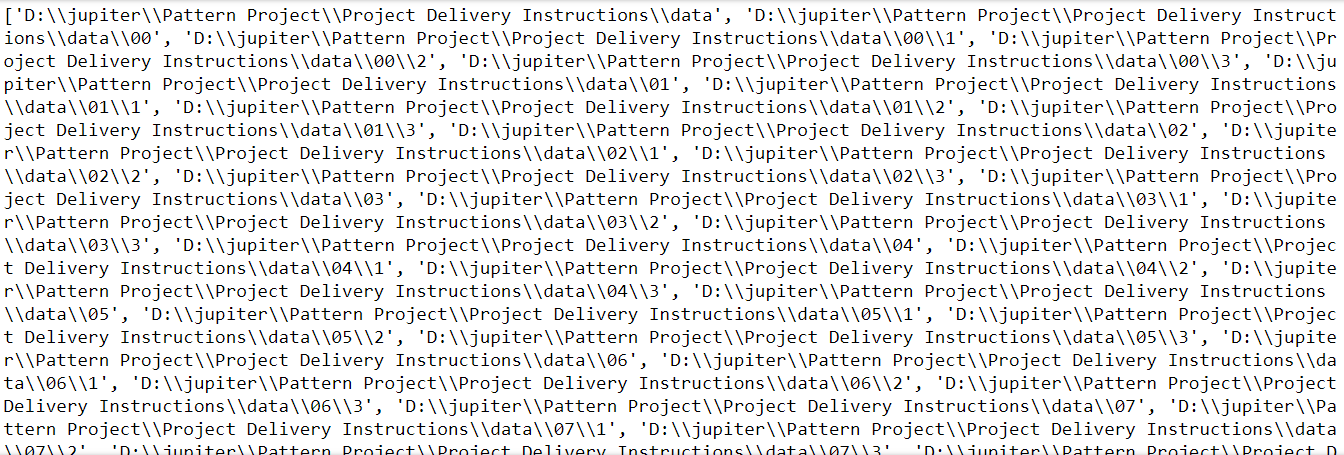
Team members:

* Mohamed Jamal
* Eman Elsyed
* Nourhan Kamal
* Nada Karam

Contents:

* Project pipelining
  + Description
  + Libraries used
* Preprocessing module
  + Techniques used
  + Libraries used
* Feature extraction/ selection module
  + Techniques used
  + Libraries used
* Model selection and training model
  + KNN algorithm
* Performance analysis module
  + Measuring accuracy
* Problems I faced (Unsuccessful trials).
* Future work and improvements

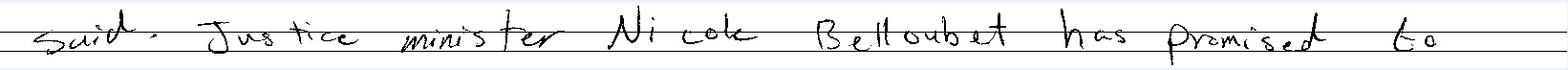
1. Project pipelining:
2. Description: we first calculate all the directories and sub directories of all the data folder, then we iterate through all directories in ascending order.
3. Example: we create a list **all\_dir** then we store all directories in it as strings and all sub strings, such as :



1. Libraries used in the project:
   1. Os for storing all directories in a list of strings.
   2. glob for fetching our images in each path

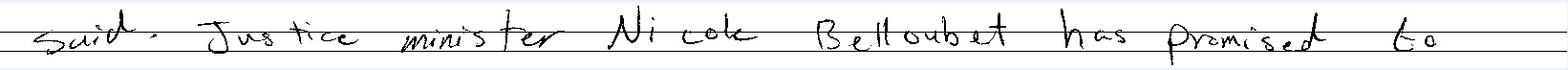


1. Preprocessing module
2. Techniques used:
   * We first convert image to grayscale image then Binary Image then Inverted Binary image.
   * We convert RGB to grayscale as a bridge for our target (Binary Image).
   * We obtain binary image to calculate the histogram of black pixels later.
   * We obtain Inverted Binary image to calculate contours of the image to separate the image into written lines to be used into feature extraction later.
3. Functions used :
   * Reading image (imageName): to convert our image.
   * Draw up down Lines (img , binaryInvertedImage) : to split our image to sub written lines for processing.
     1. It returns height and width and uppers and lowers, we focus on uppers and lowers lists as they represent top and bottom lines of each written line.

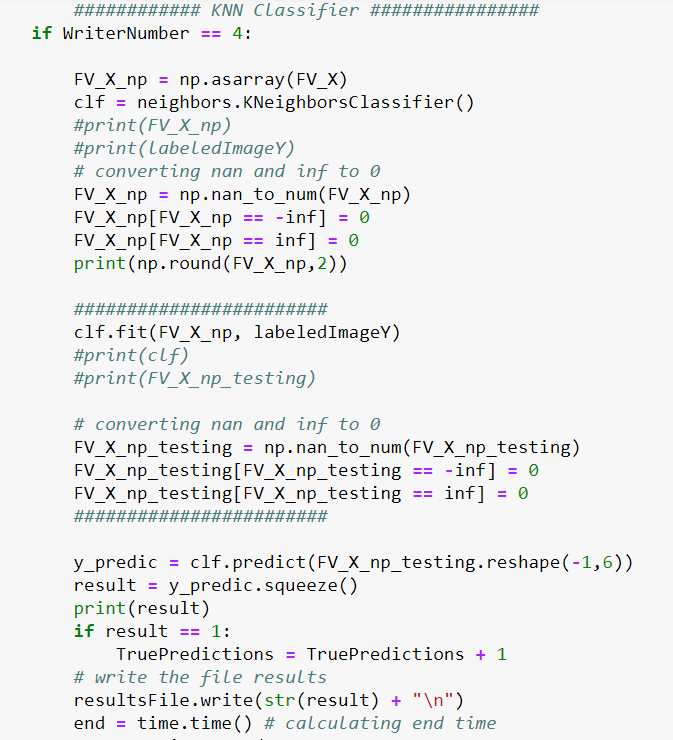


Top and bottom lines are blue lines in figure

1. Feature extraction/ selection module
2. Techniques used: we are calculating 6 features of the sub images that contains the written lines, we are calculating the upper base and lower base line of the written line then we calculate our features.

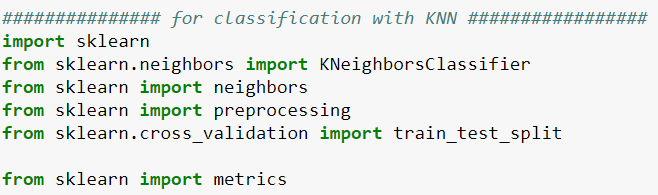


Upper base and lower base line (black lines)

1. Features calculated :
   1. F1 : Y top line – Y upper base line
   2. F2 : Y upper base line – Y lower base line
   3. F3 : Y lower base line – Y bottom line
   4. F4 : F1 / F2
   5. F5 : F1 / F3
   6. F6 : F2 / F3
2. Functions used :
   1. Divide to sub images : to get the features list
   2. Calc features: to calc average of f1, f2, f3 and to calculate f4, f5, f6.
3. Module selection and model training
4. We used KNN algorithm which computes the nearest distance of the testing feature vector to the classes of writes.
5. Illustration :

* We first filtering our list of training from Nan, inf and – inf and setting them to zero.
* Then we use fit function to train our model.
* Then we also filter our testing set if an error occurred while extracting the features from Nan, inf and –inf to zero.
* Then we use predict function to predict our writer.

1. Libraries used :

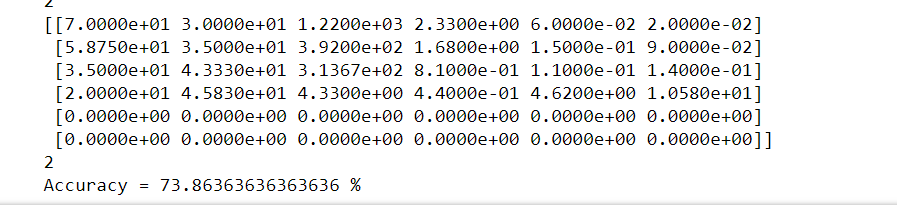


1. Performance analysis
   1. We iterated through 88 images and we calculated the accuracy as follows :

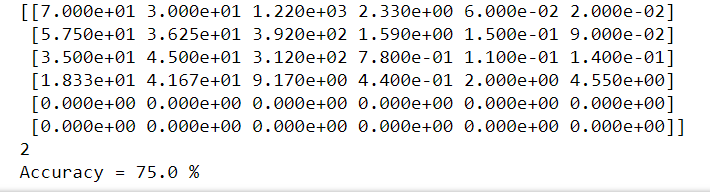
Accuracy = [(number Of True Predictions) / 88] \* 100 %

* If no intensity factor , it results in 67.04% accuracy
* If intensity factor is 1.5, it results in 68.14% accuracy
* If intensity factor is 2.5 , it results in 73% accuracy
* If intensity factor is 2.7, it results in 75 % accuracy
* if intensity factor is 2 , it results in 69% accuracy
* If intensity factor is 4 , it results in 70% accuracy
* If intensity factor is 4.5, it results in 76.13% accuracy (Highest one)
* If intensity factor is 4.7, it results in 75% accuracy



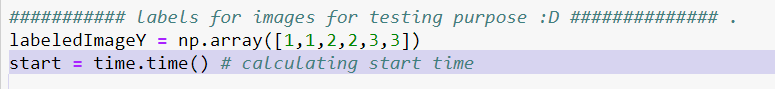


Accuracy = 73.8636 % of factor 2.5

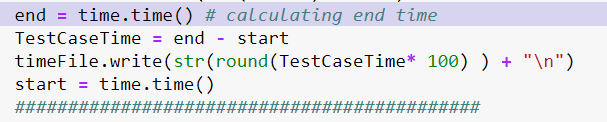


Accuracy = 75.0 % of factor 2.7

* 1. Time : we measured the time of each test case as follows:
     1. We place start before entering the loop



* + 1. We place end after classification prediction



* + 1. Then we assign start at the end of classification

1. Problems I faced
2. How to separate text to lines was hard to be understand.
3. How to find the upper and lower base lines was a little tricky as I place the lines from 1000 intensity of pixels but the images intensity was so small so I multiply by factor of 2.5 to increase the intensity.
4. How to correct the extracted features was a tricky part as nan , inf, -inf stops the working process so I converted them to zero as they are a result of low intensity writing and small text writing ( so it is logically correct ).
5. Also the pipelining was a very logical part, it needs a great focus on how we scan the directories and extract the images for them in a correct order.
6. Future work and improvements
7. First I will apply angle calculation of the hand written text.
8. Then I will find the distance between letters as a feature.