The code has 2 classifier: implemented for 3 authors.

1. Decision tree
2. Perceptron

List of files:

1. **Decision tree (DT)**
    a. Start.py→ the tree is generated here.
    b. Main.py→ it has a data_obj with helps in handling data for decision tree . it has many helper function like get_row_with_max_entropy, calculate entropy, class_entropy etc
    c. Create_a_tree_from_file→start.py will create a file and store it and then create_a_tre… will build a tree from that file . the file of stored tree acts as a stored model for DT.
    d. Data_extrated.txt→contains training data.
    e. Alpha.txt->contains the decision tree..
        i. Srno → serial no of the node
        ii. left_node → leftnode of the current node
        iii. right_node→right node of the current node
        iv. column_no-> column number of the attribute to be checked for
        v. split_val->split value of the attribute of the column no
        vi. majority_val->the leaf attribute used to decide majority of class
        vii. are_you_root->identifier for root .(useless…. kind of)
2. **Perceptron**
    a. Perceptron_model_gen .py→ this will create the model for perceptron.
    b. P_model.py-> the actual perceptron object
    c. P_test_percept.py→this has the code for testing perceptron model.
    d. P_weights.txt-> this acts as a storage for the model generated.
3. **Data creation:**
    a. Get_data-> this is the main file which is used to feed the folder names and author ids
    b. Getfiledata→this file reads the data from the text file and will and will create a object for each 250 words.
    c. Process250data.py-> the actual feature extraction happens here.

d. Data_extracted.txt-> this file stores the training data for the perceptron and the decision tree

e. Author_identification.csv-> it is a file whith same data as file (d).

FEATURES:

I have parsed the data string (i.e the content of the book) divided into of length 250 and extracted following features.

1. No of words per line-> average number of lines per word.
2. No of stop -> i.e number of full stops == no of lines
3. No of stop word count -> I have used the nltk library . from it the corpus i.e set of stop words for English language. Examples of stop words→"a," "and," "but," "how," "or," and "what.
4. No of "the"→ number of occurance of word "the"
5. No of "that"→ no of occurance of that
6. No of comas ','-> no of comas in 250 word para
7. No of words above 6→ count of words whose length is above 7
8. Average_word_length→ the average word length in 250 words paragraph
9. No of semi colon→ no of semicolon in 250 word para
10. No of colon→ the count of no of colon in 250 para.

These features are extracted on text whose word count is 250 .

A vectored representation of the extracted data is made and saved on a file i.e. data_extracted.txt

Test data is calculated in similar way.

I have tired many other combinations of data features but these seem to best classify the data specifically for DT . the best feature I found was a semicolon. Which drastically increasd the accuracy to 75% for DT

DECISION TREE:

We have used decision tree which has only 2 leaf i.e. a BST(Binary search Tree) type model.

Splitting the column.---- This is 2 step process

1. step 1--> select a column and find the best split for the column
    - We first find the no of unique values from a particular attribute column.

- Then we try different combination of averaged values of 2 numbers for splitting and then calculate the entropy .
- We do this for all possible size 2 combinations and select the value for which we get highest gain.

2. We do the step 1 for all columns .and find the column who has the highest gain.
    a. Then we use that column as a spilt column and then the corresponding split value to divide the data.
    b. No of attributes and no of classes can n . it is not fixed.
    c. The model can be scaled to n attributes and n output classes.

After selecting a column we split the dataset in 2 parts based on the split value of the column wit highest gain.

The model will send a part of dataset to left and other to right.

The column wont be used again after it using it for 1 time.

We have a cutoff for depth anyways after exhausting the column numbers the DT won't grow.

We can reuse column , by just commenting a single line.

Perceptron:

- I have trained 3 perceptron ,each one is fed with data for one author only by making all other data points to be zero.
- Then we calculate the weights and use them for finding the value of author.
- The weights are saved in p_weights.txt file.
- Later the file is used to update the weight to directly find the author.
- Sigmoid is used as transfer function.

TRAINING DATASET:

- The folder for training data are included with the uploaded document.
- 3 folder for 3 authors
- They have approximately 6-7 books each
- They create 7303 data rows, approx… equally distributed.

- The data is extracted by reading the book ->dividing the text in 250 word batch->then performing extraction operation on the 250 word input.

HOW TO TRAIN MODELS:

For decisiontree→run the file start.py. it will prompt you for a name of file, enter the name of file which has data.(data file should have 1st line as length of data vector.)

At last after creating the tree the output is saved in a alpha.txt file.

This file can later be used to create a tree.

For perceptron->run the perceptron _model_gen.py file which will again prompt you for a file name enter the same file as above and then let it run. The weights will be stored in p_weights file which is latter used to create a perceptron.

HOW TO RECREATE MODELS:

For DT -> use the file create_a_tree_from_file.py it asks for 2 files one testdata file and tree file.enter those 2 files and u can see the output.(test_data_file_should be same as data_extracted_file.)

For perceptron->use the file p_test_percept.py it asks for 2 files i.e. testdata and p_weights file.

Accuracy→

DT has shown an accuracy of 75% for three authors on the given data set for the available testdata from the file test_data.txt

Perceptron shown accuracy of 49% on 3 authors.the perceptron doesn't work as expected on 3 authors as it cant classify the plane in 3 parts ways using a line.

Even though I have used 3 perceptron as the data is not linearly separable it fails to classify accurately.

I have also added the pair plots for the data which clearly shows that the data is not linearly separable.

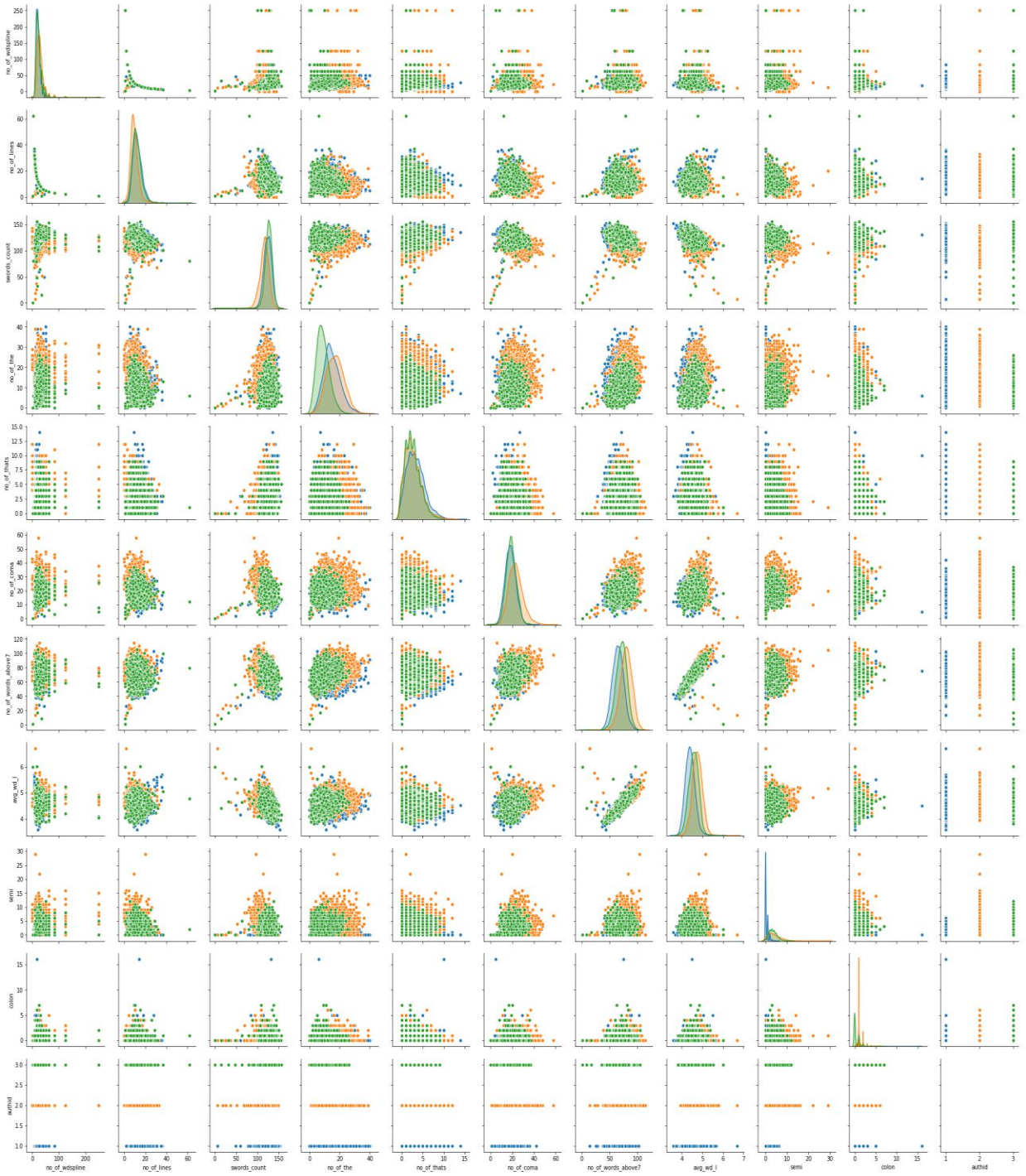

Figure 1 pairplots for datapoints