Logo, company name

Description automatically generated

**CSE523 Machine Learning**

**Prof. Mehul Raval**

**Weekly report**

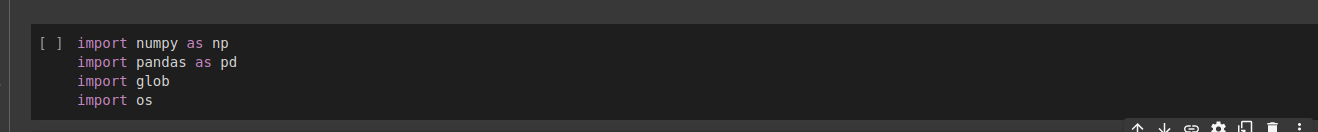
**Group number: 17**

**Group name: The Mandelbrot set**

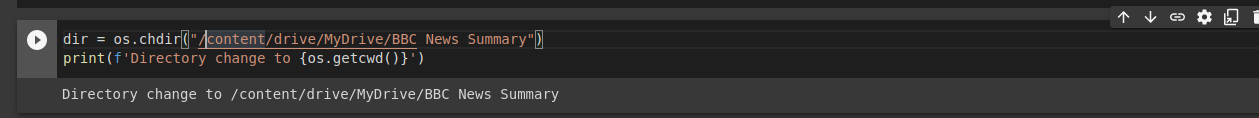
|  |  |
| --- | --- |
| **Name** | **Enrolment Number** |
| **Aastha Gaudani** | **AU2040032** |
| **Khushi Patel** | **AU2040068** |
| **Devyash Shah** | **AU2040152** |
| **Simran Khoja** | **AU1910606** |

**Weekly Report**

Firstly we imported the necessary libraries:

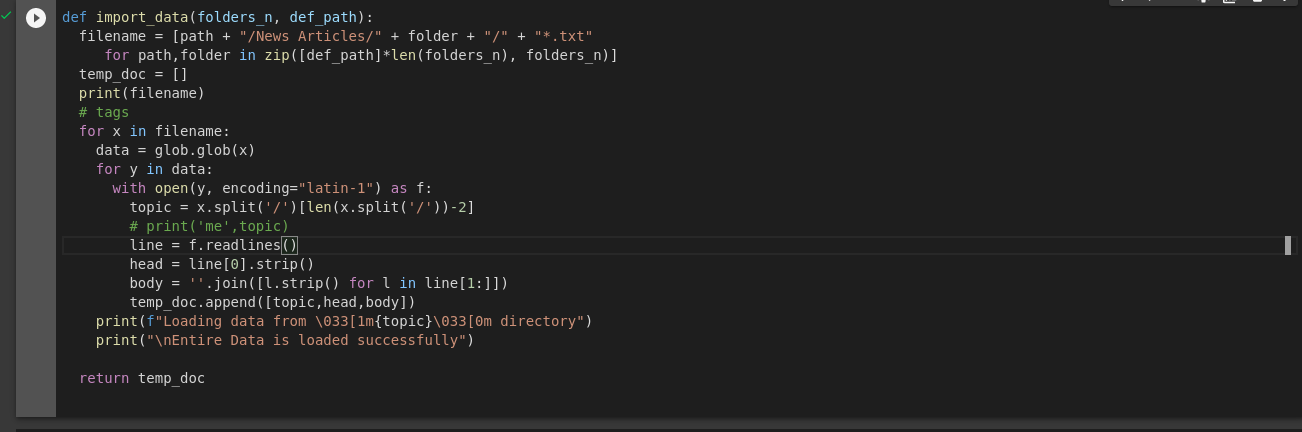


Then the code below changes the current working directory to "/content/drive/MyDrive/BBC News Summary" using the ‘os.chdir()’ function. The ‘os.getcwd()’ function is then used to print the current working directory, which should now be "/content/drive/MyDrive/BBC News Summary".



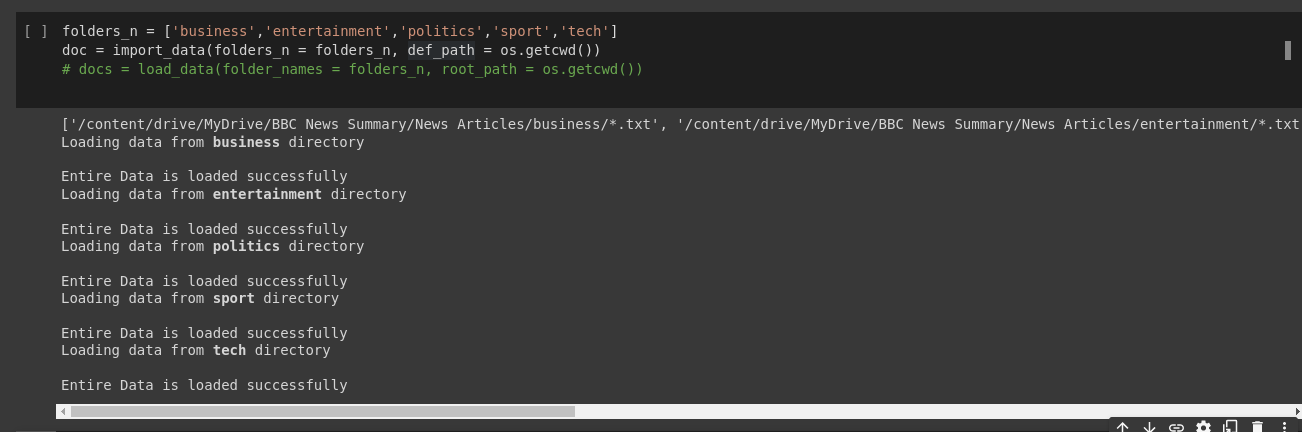
The purpose of changing the directory is to make it easier to work with files and directories within that directory or folder. Once the directory is changed, any file or directory operations that are performed using relative paths will be relative to the new working directory.

The import\_data() function takes two arguments: ‘folders\_n’, a list of folder names, and ‘def\_path’, a string representing the default path to the directory containing the folders. Using the ‘glob’ module, the function searches for all files with a ".txt" extension in each folder specified by the ‘folders\_n’ list. For each file, the function extracts the first line as the article headline and the remaining lines as the article body. It also extracts the folder name as the article topic. The function appends the article topic, headline, and body to a temporary list called ‘temp\_doc’, which contains all of the articles from all of the folders. The function then prints a message indicating which folder's data is being loaded and a message indicating that the entire data has been loaded successfully. Finally, the function returns the ‘temp\_doc’ list containing all of the articles. Overall, the ‘import\_data()’ function is designed to import text data from multiple files in multiple directories and store the data in a structured format for further analysis.

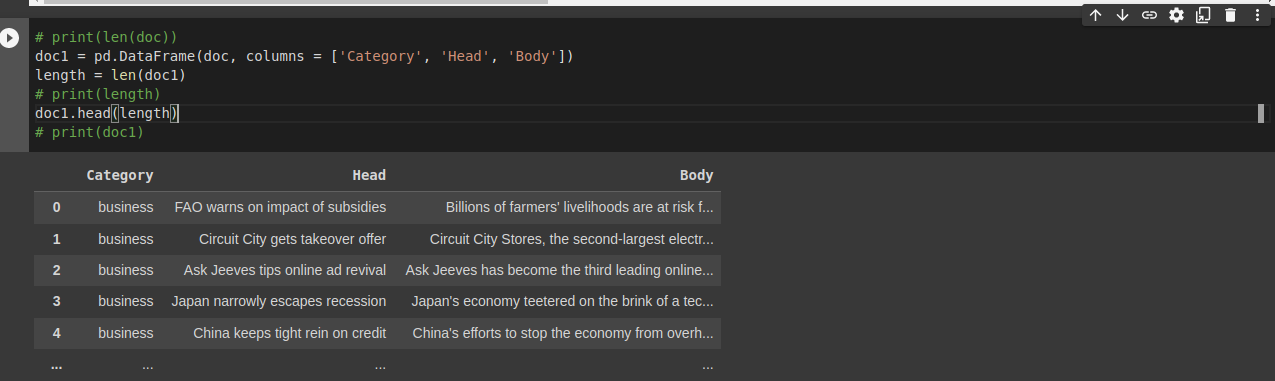


The encoding="latin-1" parameter in the ‘open()’ function is used to specify that the text files should be read using the Latin-1 encoding. This ensures that all characters in the files can be properly read and processed because some text files may contain characters that cannot be decoded by the default UTF-8 encoding.

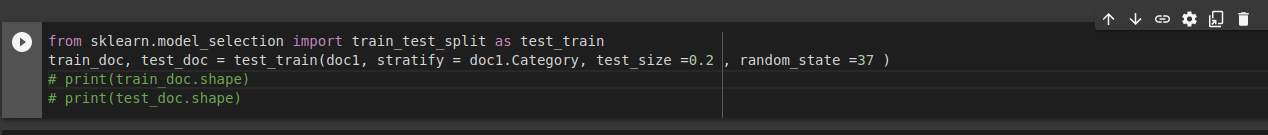
Then onwards, we import text data from multiple directions and store the data in a structured format for further analysis. A list of folder names ‘folder\_n’ is created containing the names of the folder from which h=the data needs to be imported. As the ‘import\_data()’ function is called with the ‘folder\_n’ and ‘def\_path’ arguments, and its returned value is assigned to a variable ‘doc’ which is a list of all the articles from the data folder in a structured format.



Then we create a data frame using pandas which we name as ‘doc1’, taking the ‘doc’ list as input and specifying the column names as ‘Category’, ‘Head’, and ‘Body’. Then we display all the rows using ‘length’.



Now, we would have to split the data into training and testing sets. We accomplish the task by using ‘train\_test\_split’ function from the ‘sklearn.model\_selection’ module and split the ‘doc1’ Data frame. We have separated 20% data for testing. The ‘stratify’ parameter ensures that the split is done in such a way that preserves the relative proportion of every category in the target variable in both training as well as testing data.

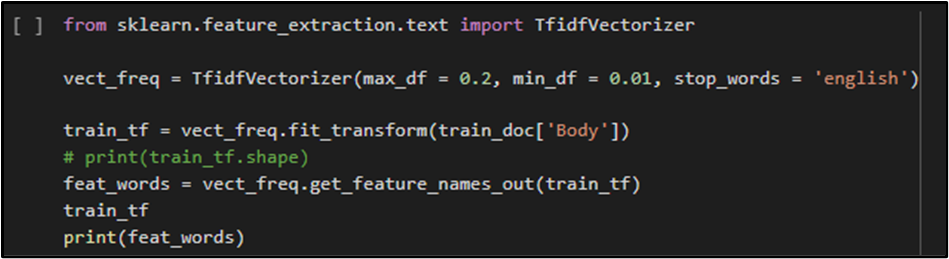


**TFIDF**

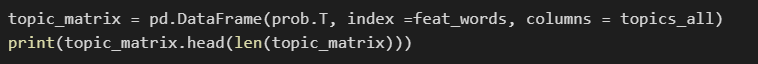
Here we are keeping stop words as English. This means it will ignore insignificant words like the, a, an, have etc. This reduces the length of our sentences.

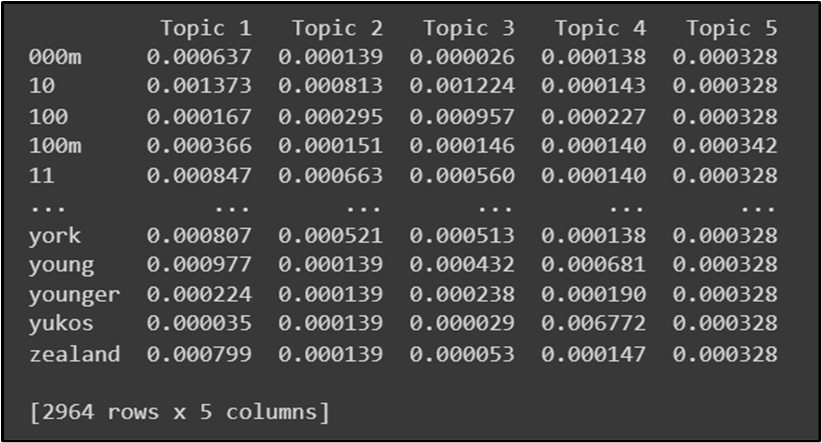
Here we are using max\_df = 0.2, this means that the words that appear in more than 20 percentage of the documents will be ignored. Min\_df = 0.01, this means that any words that appear in less than 1 percent of the documents will be ignored.

Setting appropriate values for max\_df and min\_df can help to reduce noise in the data and improve the quality of the TF-IDF matrix. However, it is important to choose these values carefully based on the characteristics of the data being analysed.



We have also printed the probability matrix of words with their probability.





**LDA**

After this we are training our model using LDA.

For Linear Dimensionality Reduction, LatentDirichletAllocation is imported from sklearn.decomposition.

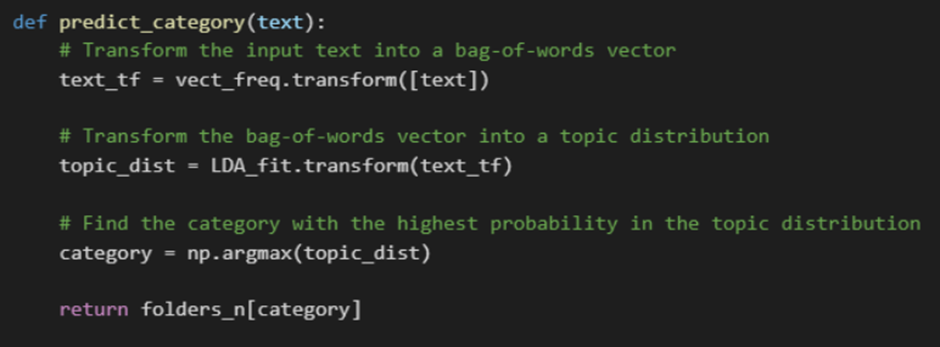
LDA is then fit with 5 components and batch learning method with maximum iterations set to 10. This is used on the training dataset.

A matrix is formed with the words and topics to see the frequency of each word under a particular topic.

A list of frequent words is formed.

The input text is transformed into a bag of words vector which is then transformed into a topic distribution. The category with the highest probability in the topic distribution if found.

This is tested of different texts to find out the category that they may belong to.



But the problem with this approach is its accuracy. We are getting very less accuracy. So we are planning to again train the model with SVM. So our next weeks target is to train the model with SVM.