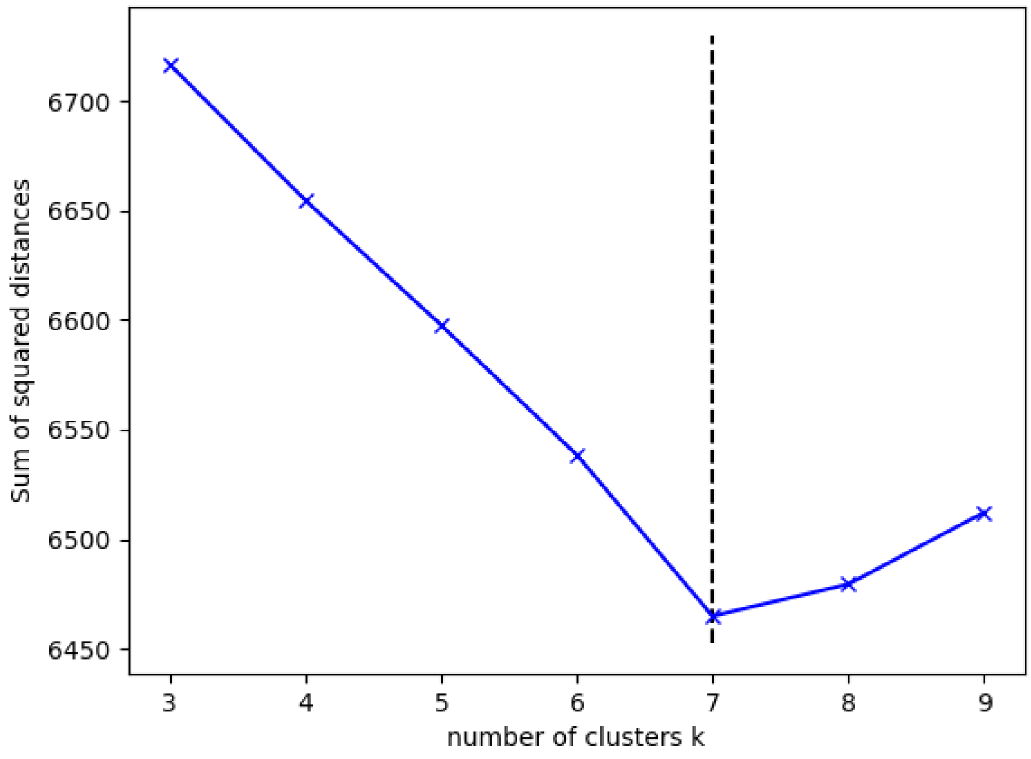
Name: Krishna Pullakandam

Data of submission: June 20th, 2020

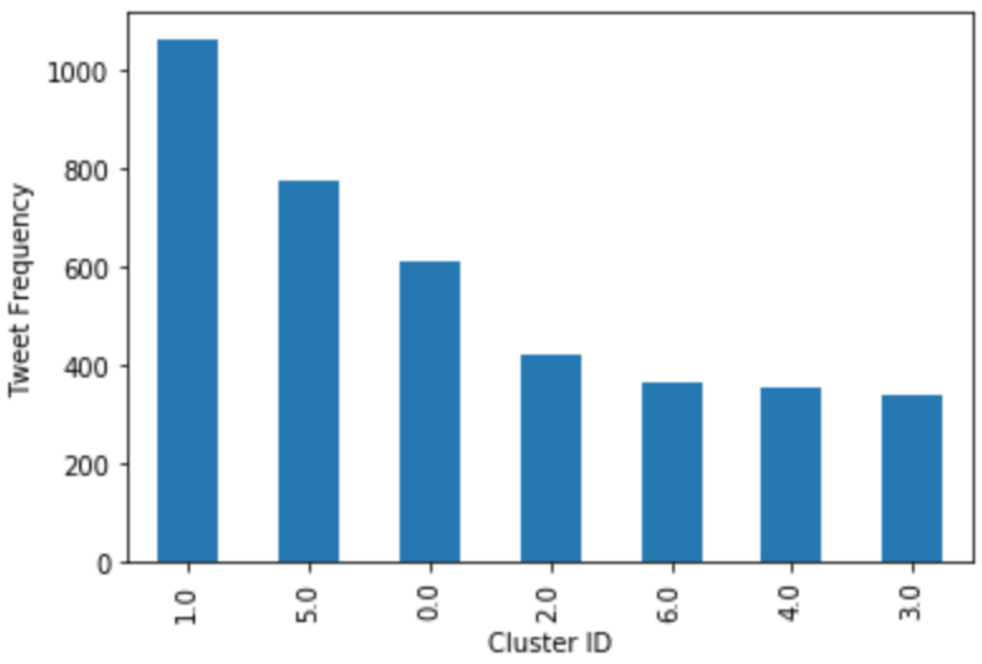
ICX Media - Take home exercise

Code is available at: https://github.com/kpullak/healthTweetsClustering

1) Combine all the tweet files (/Health-Tweets/*.txt) into a single big text file (/result.txt) using the script – combine_data.py script file
2) Pre-process the big text file to extract only the tweets and remove the unnecessary information like the time stamps etc., - preprocessing.py script file
3) Perform k-means clustering on the tweets extracted from the big text file using – main_clustering.py script file
4) Finding the best k by iterating over a few values of k, and selecting the best value based on knee plot using – main_clustering.py script file

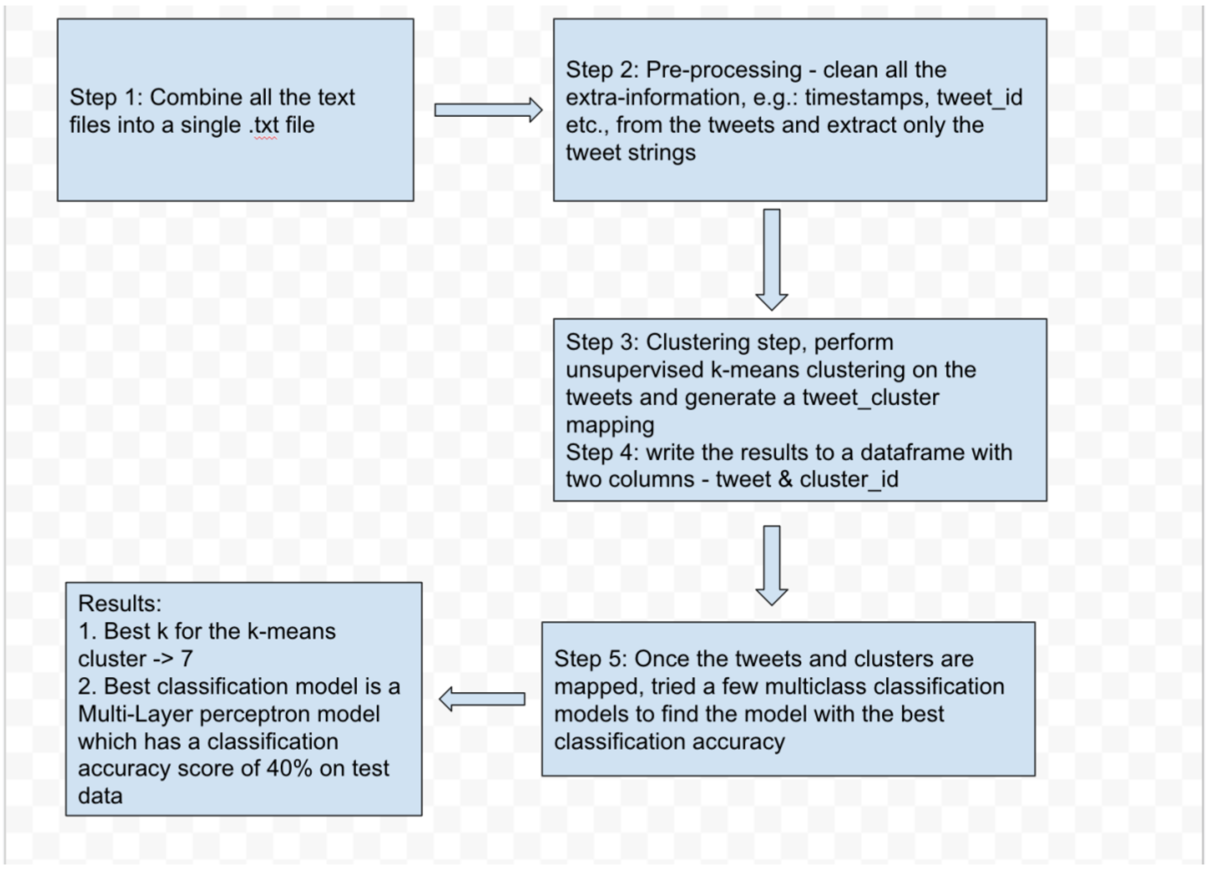


5) Using the k-value we found earlier, generating a data frame with tweets mapped to their cluster number using – main_clustering.py script file



6) Split the data frame formed in the above step into train and test and input the training data frame to different classification models (Naïve Bayes, Logistic Regression, Random Forest Classifier, Linear Support Vector Classifier, Multi-Layer Perceptron using Bag of Words) and find the one which has the best classification accuracy – using classification_bow.ipynb and multinomial_classification.ipynb python notebook files
7) Generate the predictions on the test dataset using the best model and report the results and perform misclassification analysis – using misclassification_analysis.ipynb python notebook

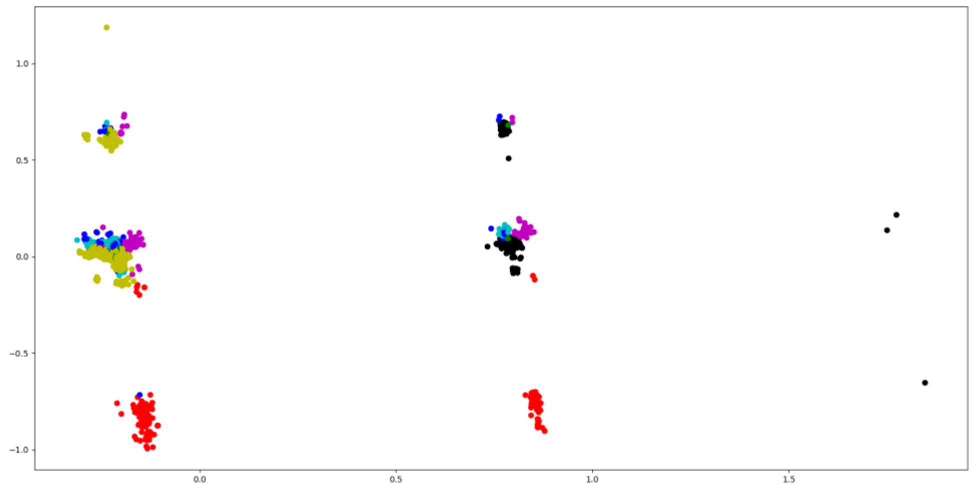
Schematics / Workflow (Data Extraction and Modelling) diagram:


Step 1: Combine all the text files into a single .txt file
Step 2: Pre-processing - clean all the extra-information, e.g.: timestamps, tweet_id etc., from the tweets and extract only the tweet strings
Step 3: Clustering step, perform unsupervised k-means clustering on the tweets and generate a tweet_cluster mapping
Step 4: write the results to a dataframe with two columns - tweet & cluster_id
Step 5: Once the tweets and clusters are mapped, tried a few multiclass classification models to find the model with the best classification accuracy
Results:
1. Best k for the k-means cluster -> 7
2. Best classification model is a Multi-Layer perceptron model which has a classification accuracy score of 40% on test data

Combine_data.py (generates the result.txt file) -> preprocessing.py (generates the combined_processed_tweets.txt file) -> main_clustering.py (generates the tweet_cluster_mapping.csv file) -> classification_bow.ipynb / multinomial_classification.ipynb (for classification task) -> misclassification_analysis.ipynb (to interpret the results of the best performing classification model)

Generating the 2D scatter plot of the cluster analysis after performing unsupervised clustering using k-means clustering:

- To run PCA, the sparse matrix that is the output of the vectorization of the words, must be first converted to a dense matrix as the PCA methods of sklearn doesn't handle sparse matrices. After PCA has been applied the first two principal components are added to the data frame. These are a series of numeric values and thus can be visualized on a scatter plot.
- The components are plotted on a scatter plot using matplotlib. In addition, the points are colored based on a particular nominal field, including the membership label as a result of the clustering.

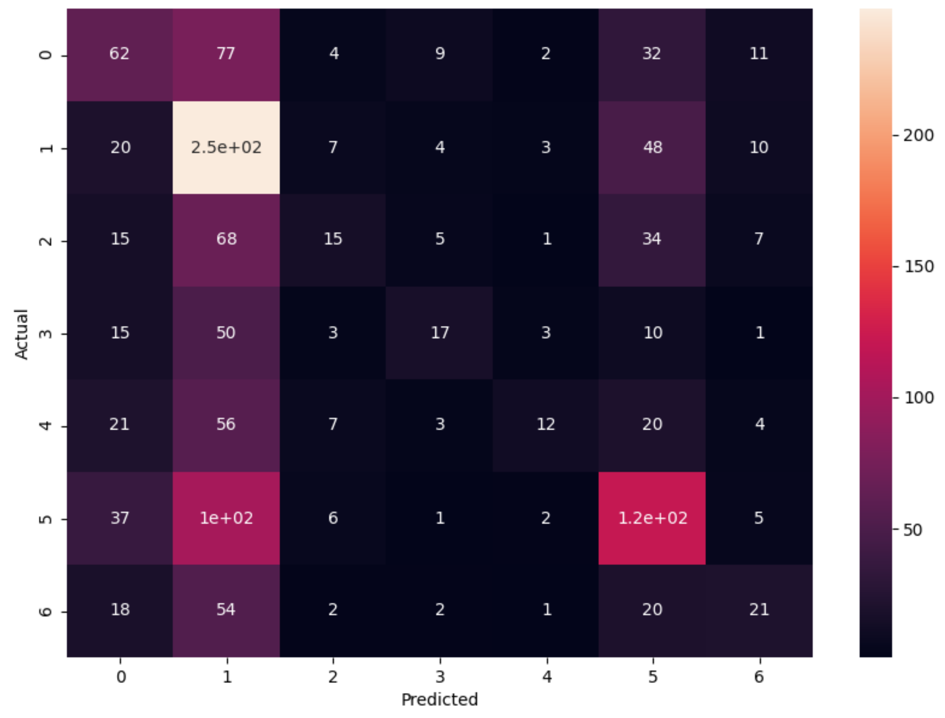


Bag of Words multiclass classification model: (Best performing model ~ 40% classification accuracy)

Text classification using Keras deep learning library. The process looks like this:

- Separate the data into training and test sets - used a 70:30 ratio to split the data into training and test sets
- Used tokenizer methods to count the unique words in our vocabulary and assign each of those words to indices
- Used fit_on_texts() to automatically create a word index lookup on our vocabulary
- Limiting vocabulary to the top words by passing a num_words param to the tokenizer
- With our tokenizer, we can now use the texts_to_matrix method to create the training data that we'll pass to the model
- We feed a one-hot vector to our model
- After we transform our features and labels in a format Keras can read, we are ready to build our text classification model
- When we build our model, all we need to do is to tell Keras the shape of our input data, output data, and the type of each layer. keras will look after the rest
- When training the model, we'll call the fit() method, pass it our training data and labels, batch size, epochs and other hyperparameters

Misclassification_Analysis:



	0	1	2	3	4	5	6
0	62	77	4	9	2	32	11
1	20	2.5e+02	7	4	3	48	10
2	15	68	15	5	1	34	7
3	15	50	3	17	3	10	1
4	21	56	7	3	12	20	4
5	37	1e+02	6	1	2	1.2e+02	5
6	18	54	2	2	1	20	21

Heatmap of the confusion matrix from the best performing model

Observation:
The vast majority of the predictions end up on the diagonal (predicted label = actual label), where we want them to be. However, there are a few misclassifications, most noticeably the ground truth labels of 5 being clustered as 1 (as shown in the light pink coloured square)

- My guess is that the misclassified tweets are tweets that touch on more than one subjects (for example, tweets involving both cancer and treatment). This sort of errors will always happen.

- We can use the chi-squared test to find the terms (unigrams/bigrams) that are the most correlated with each of the cluster ids, in order to understand the high amount of overlap that we have seen.

```
# '0':
  . Top unigrams:
       . video
       . savile
  . Top bigrams:
       . video ebola
       . video uk
# '1':
  . Top unigrams:
       . aampe
       . hospital
  . Top bigrams:
       . dementia patients
       . special measures
# '2':
  . Top unigrams:
       . cigarettes
       . linked
  . Top bigrams:
       . cancer drug
       . nhs staff
# '3':
  . Top unigrams:
       . warning
       . review
  . Top bigrams:
       . aampe waiting
       . video warning
# '4':
  . Top unigrams:
       . video
       . discovery
  . Top bigrams:
       . video ebola
       . video mental
# '5':
  . Top unigrams:
       . young
       . emergency
  . Top bigrams:
       . video mental
       . mental health
# '6':
  . Top unigrams:
       . ebola
       . antibiotics
  . Top bigrams:
       . hospital death
       . cancer patients
```

Observation:
Based on the above results, we can see that the top bigrams for cluster_5 are: (video mental, mental health) and the top bigrams for cluster_1 are: (dementia patients, special measures), we can clearly see that there is some sort of overlap between the topics in both the clusters and hence see the highest misclassification between these two classes.