Pre-Processing:

The task requires to select data and index it into elastic search for getting responses later. This data was supposed to have parameters like views, likes, dislikes, etc. which would help us to retrieve videos later in trending order.

I have acquired the data by scraping all the videos of a particular channel(in our case “Hotstar”). The preprocessing has been done in “preprocessing_pickle.py” where I have used selenium for browser automation and pafy(a youtube scraping library) for extracting relevant content for every videos. After extracting every record for our video, we organise it into a dataframe and later on dump it into a file(‘dataframe.pkl’) so that we can access it later. Also, we convert this dataframe into json(‘data.json’) for indexing it into elastic search.

If you want to have a little peek into the data, you can open ‘data.csv’ and go through few rows.

What is Trending?

I have added an extra derived column (‘trending_score’) for our data which we will be using later for sorting our search results while querying elastic search.
My definition of trending defines how much people are acting on the video, negative or positive. So, I have considered likes, dislikes and views of a video for calculating the trending score. Formula:

Trending_score = ((likes+dislikes)*20 + no.of views)/(total time in seconds since video)

This formula is not a result of much thought process. It scales likes and dislikes to a factor of 20 since these are always comparatively less for any video. We then divide it by total amount of time in seconds since video was uploaded.

Node and Elastic-Search:

1)After getting our data in JSON format, we create a new elasticsearch client. (‘connection.py’).
2) We create a new index and mappings for our index(‘create.js’).
3) We insert the data into our index using client.bulk (‘insert.js’)
4) We query for all our data in a sorted manner(according to trending score) in descending order, thus displaying video with highest trending score at the top.
5) I have recorded the response of search query in ‘output.json’. You can look and confirm that results are sorted in descending order according to trending score.

Issues and Further work:

Next, my approach would be to parse the output.json file to generate html pages which can be viewed. But, for that I will have to figure out how to access the response from search request made to our client. To convert html pages and send data, most likely I will have to use express js and the templating engines(like pug, jade for it)

Doubts:

Is there a good way to access the response received from es-client while making search request in ‘search.js’?
While converting it to JSON, it’s not generating an ideal output.