ML PROJECT 1

Project Report

TEAM MEMBERS

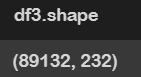
* VANSH MAHESHWARI
* PARTH PARASHAR

SUBMISSION

We have created 2 different codes which we have submitted. This is due to the fact that we did not properly read the string data and applied clustering on the majority of the metadata. As told by the professor during the presentation we considered his opinions and started working on the code from scratch. Hence, our first code represents our efforts before the presentation and the second code is our understanding of what the professor meant should have been done in the code instead.

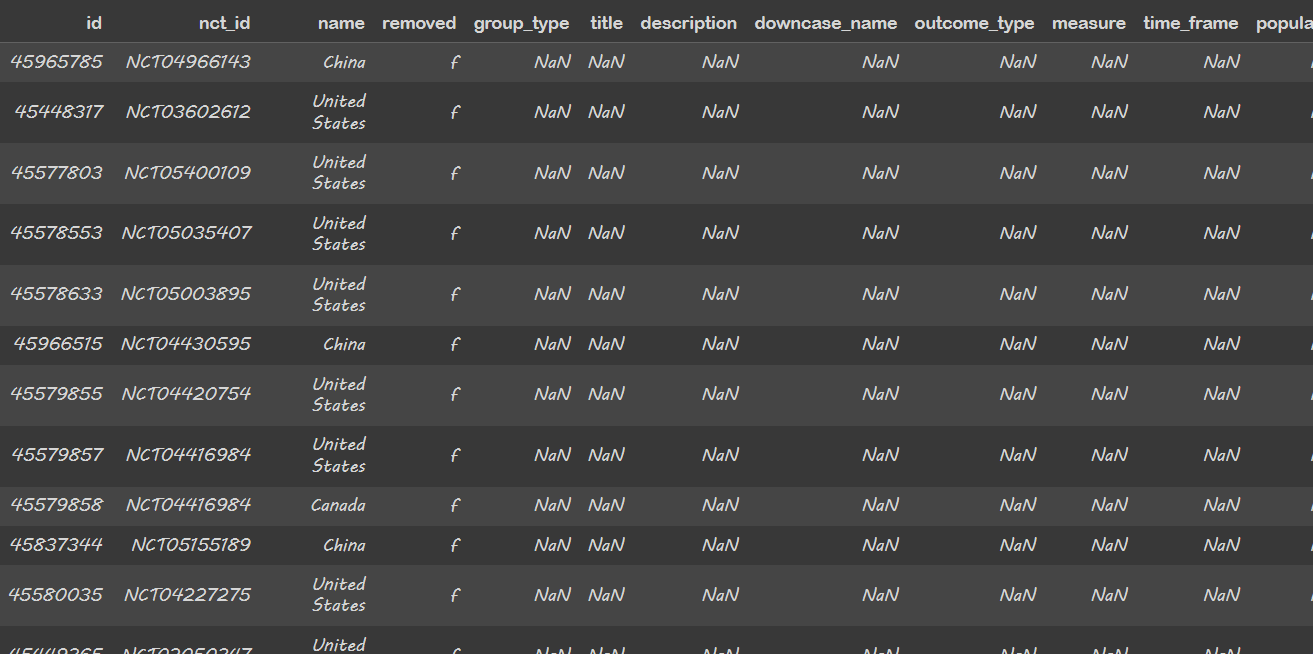
We have attached the first code, and this report explains both of our codes. This is necessary as we have repeated quite a few steps from code 1 to code 2. Due to lack of time post-presentation, we weren't able to show all the steps in code 2 and used them without much explanation. Hence this report consists of the combined effort put into the project in order to help the invigilator understand the approach better.

For CODE 1

* Once we import the data, it is important to check its shape and size.
* Then check out various columns and see how can we preprocess the data to clean it and make it more usable.
* The following figure shows the shape of the data.
* Clearly, it has way too high dimensionality and must be reduced for better clustering.

REMOVING NAN VALUES

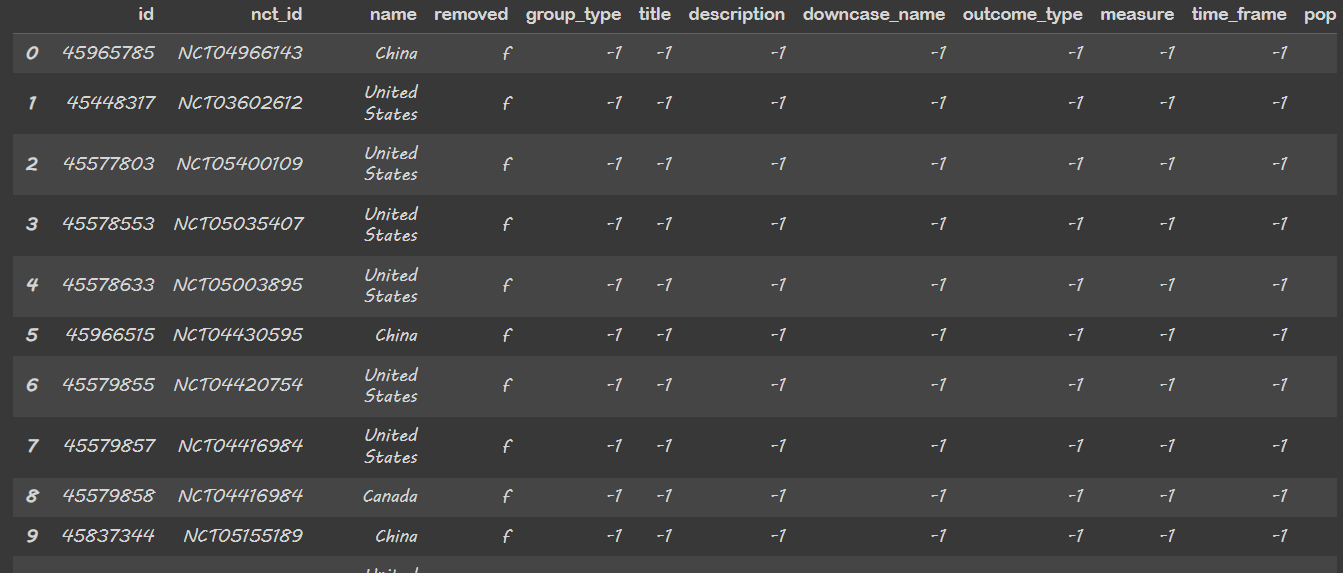
Now since the data has a lot of NaN values we must deal with it separately. The columns having all the values as NaN values must be removed.



Now after removing the columns we reduce the columns as seen here.

A black and white background with white text

Description automatically generated

Now let’s change all the NaN values to -1 using the .fillna function. This allows us to give an integer value to all the NaN values which can be later used for regression/clustering.

Now, converting all datatypes to string data type. This is done in order to make label encoding easier as during label encoding we need all the values to be of a particular datatype which can be converted to string values all at once.

A black screen with white text

Description automatically generatedA computer screen shot of a black background

Description automatically generated

**Performing Label encoding**

* + **A screenshot of a computer

    Description automatically generated**We import the LabelEncoder from sklearn to perform label encoding on our data without which clustering is not possible.

A black background with white numbers

Description automatically generatedNow it’s time to perform our first clustering. So, in this clustering we haven’t performed any special functionalities to make the result more accurate.

Now clearly this is not a very accurate clustering result. 0.55 is considered to be a very avg silhouette score. Hence, we need to improve it a lot to get anywhere close to what can be considered as good clustering.

**2D and 3D visualization of clusters**

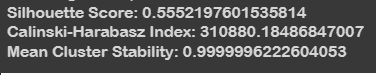
A graph of a diagram

Description automatically generated with medium confidenceA graph of a diagram

Description automatically generated with medium confidence

**Now let’s remove more useless columns**

* So, we decided to remove all the columns with more than 99% NaN values in order to reduce the useless data.
* Turns out over 70 columns were like this and hence couldn’t be removed.
* This tells a lot about how the data we are going to use is very unbalanced as very few attributes are filled for each case.

Still let’s perform the analysis after dropping all these columns

* As can be observed, there is little to no change in the silhouette score from the last change that we did. This move clearly didn’t do any good to us.
* But removing such a massive amount of data without causing any good seemed like a poor trade-off as it might result in the loss of some important data. Hence, we decided to skip this step and perform something else.

**Now we explore another method to reduce the dimensions**

* We perform cardinality analysis in our next step to reduce dimensionality.
* This means we will check the no. of unique values per column and remove the ones having extremely high cardinality.
* This is because during clustering we like to use the columns whose values can be grouped together. Having extremely high unique values gives poor clustering results as it becomes hard to group so many values.

**Performing cardinality analysis**

* + We took a random no = 50 above which the column would be discarded. Now let’s perform clustering to it to get an idea if we are moving in the right direction.
  + A massive 71 columns were dropped.

A graph with colored lines

Description automatically generated

As one can see, the graph has changed massively , this means some changes have definitely happened in the result.

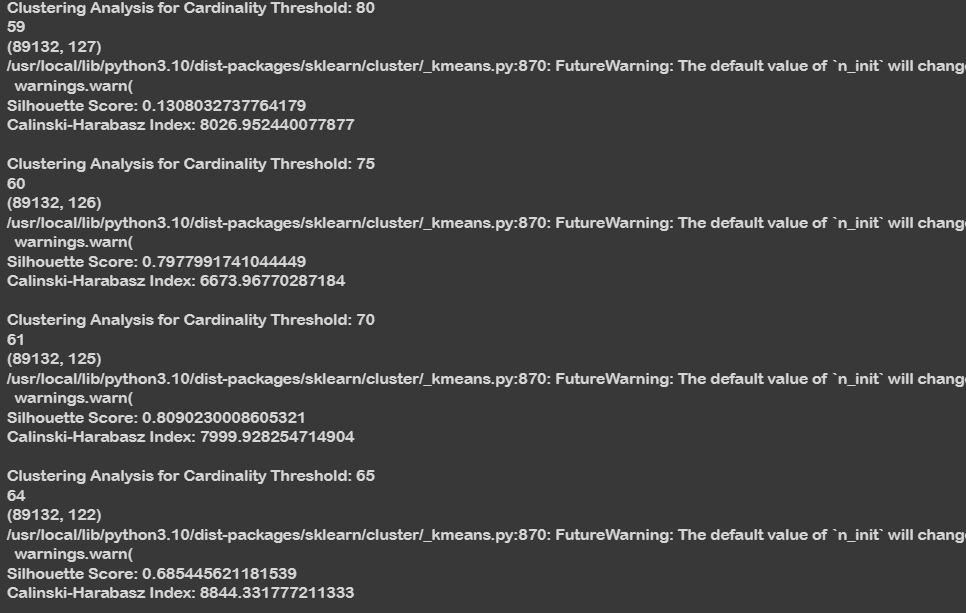
**Performance Evaluation**

The silhouette score increases massively telling us that we were indeed looking in the right direction and hence we decide to keep exploring this further in order to get the best code.

**So now we will check how clustering changes with change in cardinality.**

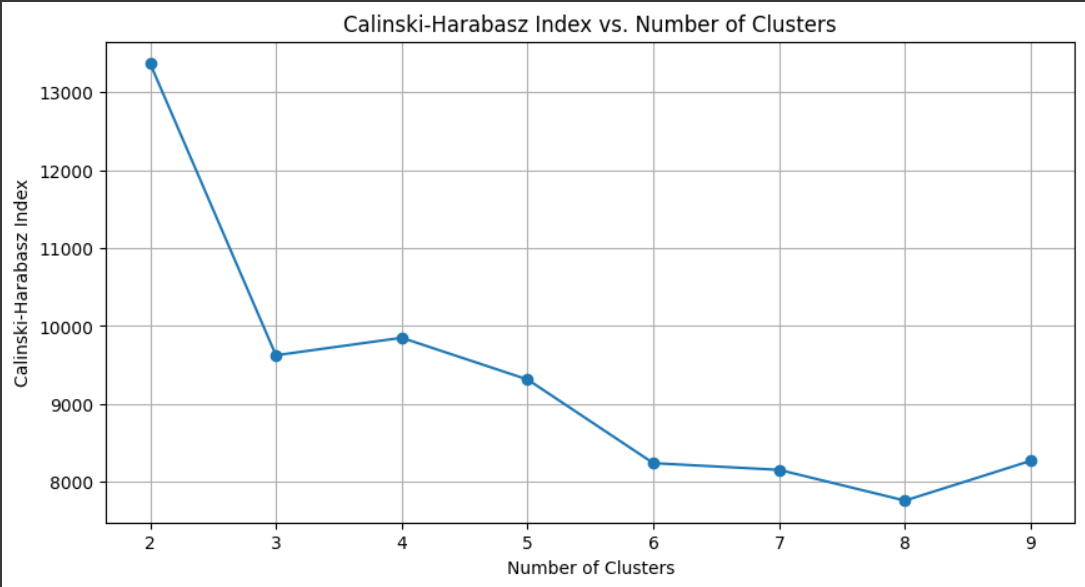
* We run the loop for different cardinality values and check how they perform one by one. We loop through a threshold of 80 and all the way down to 30 with a gap of 5.
* Here threshold means the cardinality value above which the columns are not considered in the dataset.
* For eg: a threshold of 70 means no columns having more than 70 unique values are to be considered.

A black screen with white text

Description automatically generatedA screenshot of a computer

Description automatically generated  
**Now it’s time to test our model with different number of clusters.**

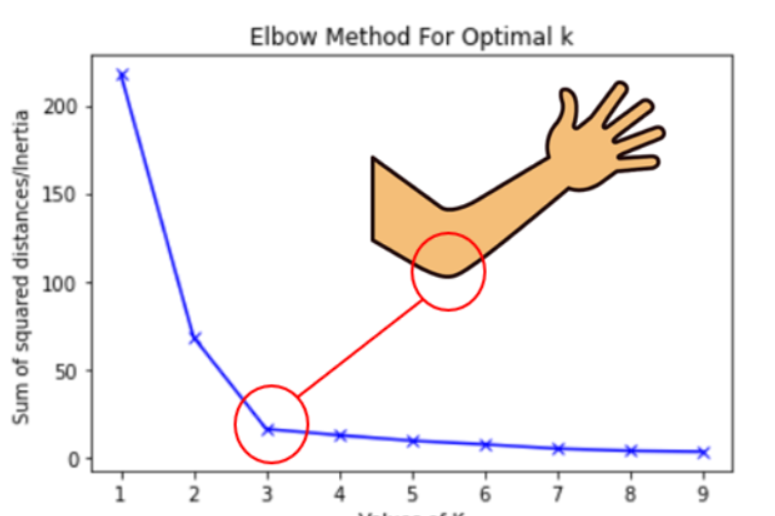
* The model is tested on clusters 2 to 9 and performance is evaluated for the same.



A graph with a line

Description automatically generated

Now let’s find out the best cluster using elbow method.

* The elbow method says that the point on the graph which shows like an elbow or the part where the rate of decrease shifts.

A graph with a line

Description automatically generated**Applying the elbow curve method to our data**

Clearly, for our data, the most optimal no of clusters turns out to be 4.

**Now adding resampling to our data**

* + A black background with white text

    Description automatically generatedA black background with white text

    Description automatically generatedWe do resampling by using the k-fold cross-validation. Let’s test the code now and see if it changes something.

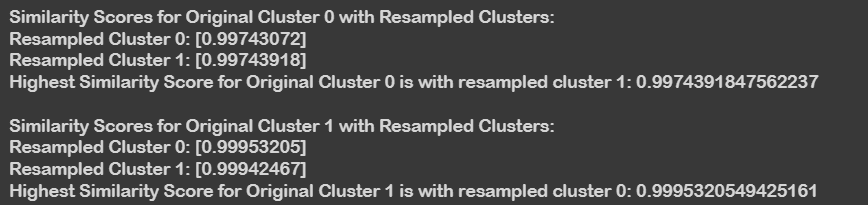
After resampling

A black background with white text

Description automatically generated

After resampling

**The second part of the question says :**  
For each cluster in the original clustering find the most similar cluster in the clusters produced from the resampled data. So, lets do that!



For 2 clusters

A screenshot of a computer program

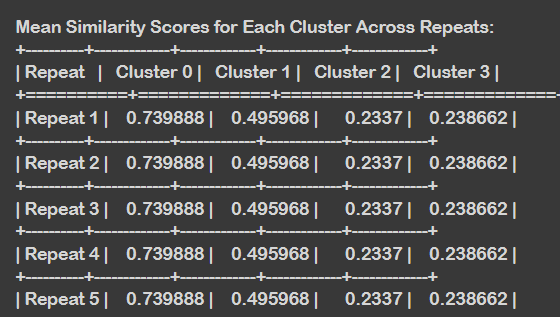
Description automatically generated

For 4 clusters

**Now for the final part of the project**

* Repeating the process K times and computing the mean similarity across all resampled sets for each cluster.

A number of white text

Description automatically generated with medium confidence

**CODE 2 REPORT STARTS FROM NEXT PAGE**

**CODE 2 REPORT**

Now in code 2 we have used TF-IDF vectorizer and the bag of words approach to input the string attributes of txt files. This is the data on which we will then apply clustering after pre-processing.

**Pre-process the data**

We used many different methods to preprocess the data.

**Tokenisation**: This process is used for tokenizing a text string into a list of words. It breaks the text into individual words or tokens and is a must in order to properly use this data.

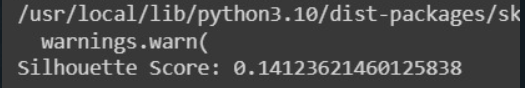
Then we will convert the characters into lowercase, first checking if it is alphabet.

**Stop Words:** Another list comprehension is used to filter out words that are in a set of stop words. Stop words are common words like "and," "the," "is," etc., which are often removed during text preprocessing as they may not carry significant meaning for certain tasks.

**Lemmatization:** Finally, we will perform lemmatization. What lemmatization does is that it involves reducing words to their base or root form. This removes all the unnecessary redundancy in the preprocessed data.

**Now we will repeat the steps as in code 1**

Check for different clusters and find out how many no. of clusters is best.



A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

As can be observed clearly that the largest value comes out when the no of clusters is 17.



**A screenshot of a computer screen

Description automatically generatedNow apply cardinality check that is removing the no. of columns that have cardinality more than the threshold value.**

As can be seen clearly now that the max value can comes out to be when cardinality threshold is 10.

Going below this value further increases the silhouette score but decreases cardinality to an extent where loss of data can become an issue.

The best case that we were able to find at the end was:

A screenshot of a computer

Description automatically generated

**Some graphs to make the representation clear:**

**A graph of colored dots

Description automatically generated with medium confidence**A graph showing the difference between cardinality and cardinality

Description automatically generated

**A graph with colored dots

Description automatically generated**

**A diagram of a diagram

Description automatically generated with medium confidence**

A screen shot of a graph

Description automatically generated

This is the end of the report. Thanks for providing us the opportunity to work on this project.

**Authors:**

Vansh Maheshwari

Parth Parashar