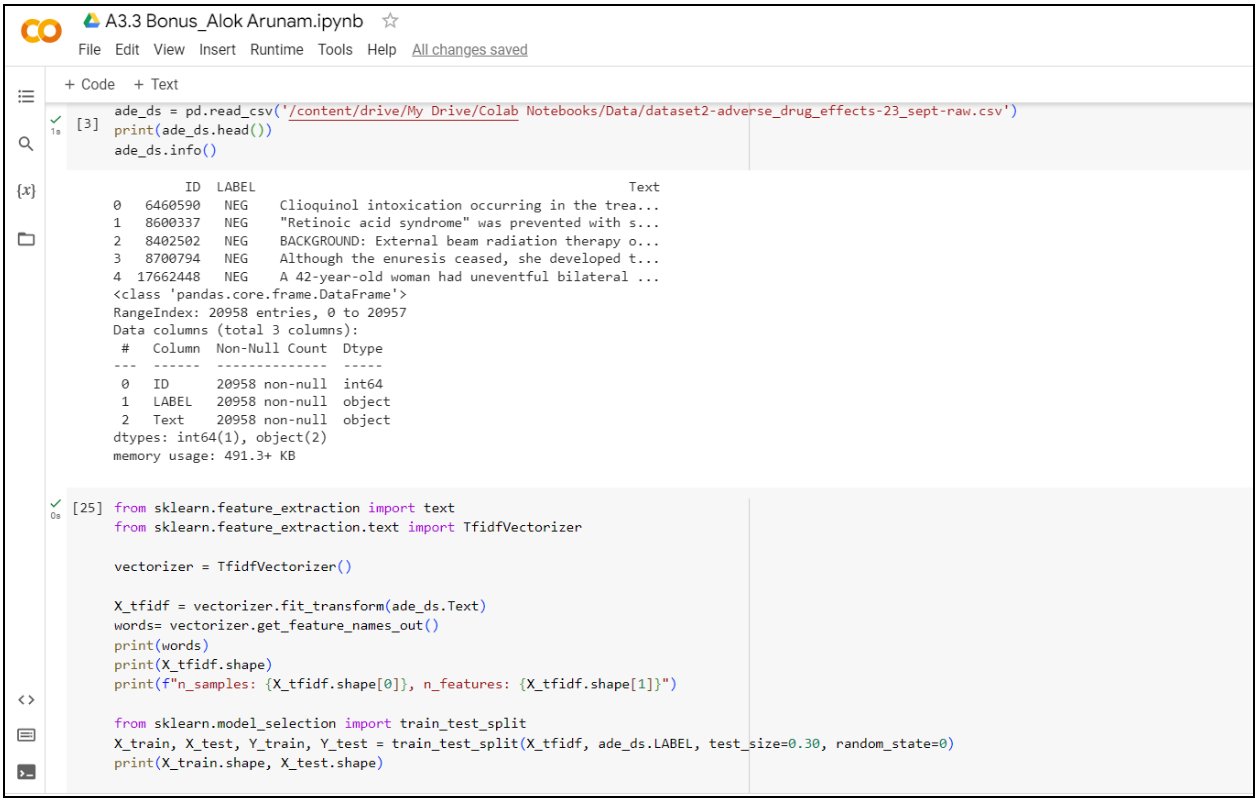
The code is here.

**Q3) Write code to run k-means clustering algorithm over the bio-medical statements in the above-mentioned file.**

**a. Starting with k=2, you can go up to k=10 or more.**

Ans: The first step is to tokenise bio-medical statements in the dataset and transform using TfidfVectorizer() to build a dictionary of feature vectors. This gives a matrix of 20958x17122.


A3.3 Bonus_Alok Arunam.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

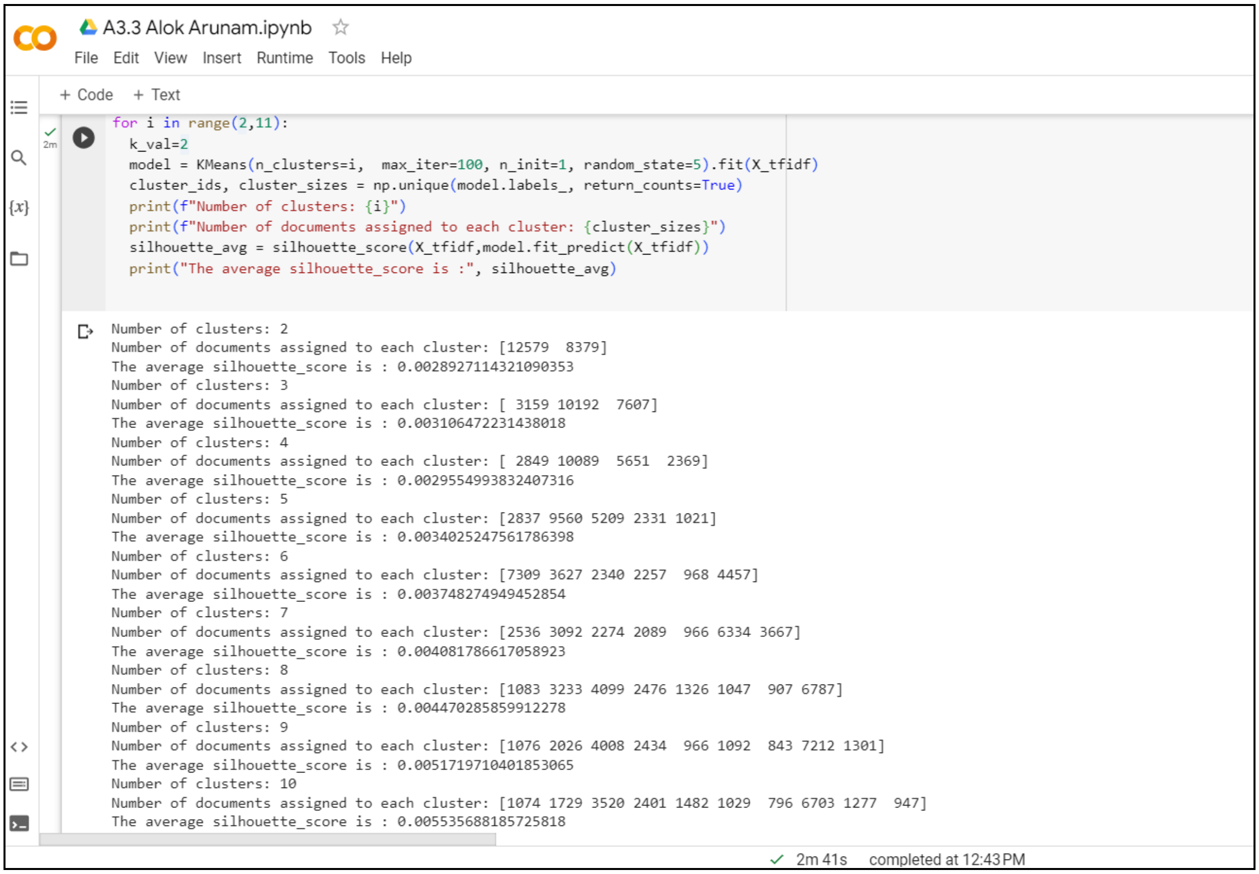
```
[3]
ade_ds = pd.read_csv('/content/drive/My Drive/Colab Notebooks/Data/dataset2-adverse_drug_effects-23_sept-raw.csv')
print(ade_ds.head())
ade_ds.info()
```

```
         ID  LABEL                                              Text
0   6460590    NEG   Clioquinol intoxication occurring in the trea...
1   8600337    NEG   "Retinoic acid syndrome" was prevented with s...
2   8402502    NEG   BACKGROUND: External beam radiation therapy o...
3   8700794    NEG   Although the enuresis ceased, she developed t...
4  17662448    NEG   A 42-year-old woman had uneventful bilateral ...
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20958 entries, 0 to 20957
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   ID      20958 non-null  int64 
 1   LABEL   20958 non-null  object
 2   Text    20958 non-null  object
dtypes: int64(1), object(2)
memory usage: 491.3+ KB
```

```
[25]
from sklearn.feature_extraction import text
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()

X_tfidf = vectorizer.fit_transform(ade_ds.Text)
words= vectorizer.get_feature_names_out()
print(words)
print(X_tfidf.shape)
print(f"n_samples: {X_tfidf.shape[0]}, n_features: {X_tfidf.shape[1]}")

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X_tfidf, ade_ds.LABEL, test_size=0.30, random_state=0)
print(X_train.shape, X_test.shape)
```

The next step is to import KMeans and build model. K-means clustering is an unsupervised learning method to find the latent groupings in the data. The number of documents assigned to each cluster (from k=2 to k=10) is shown in the results below.

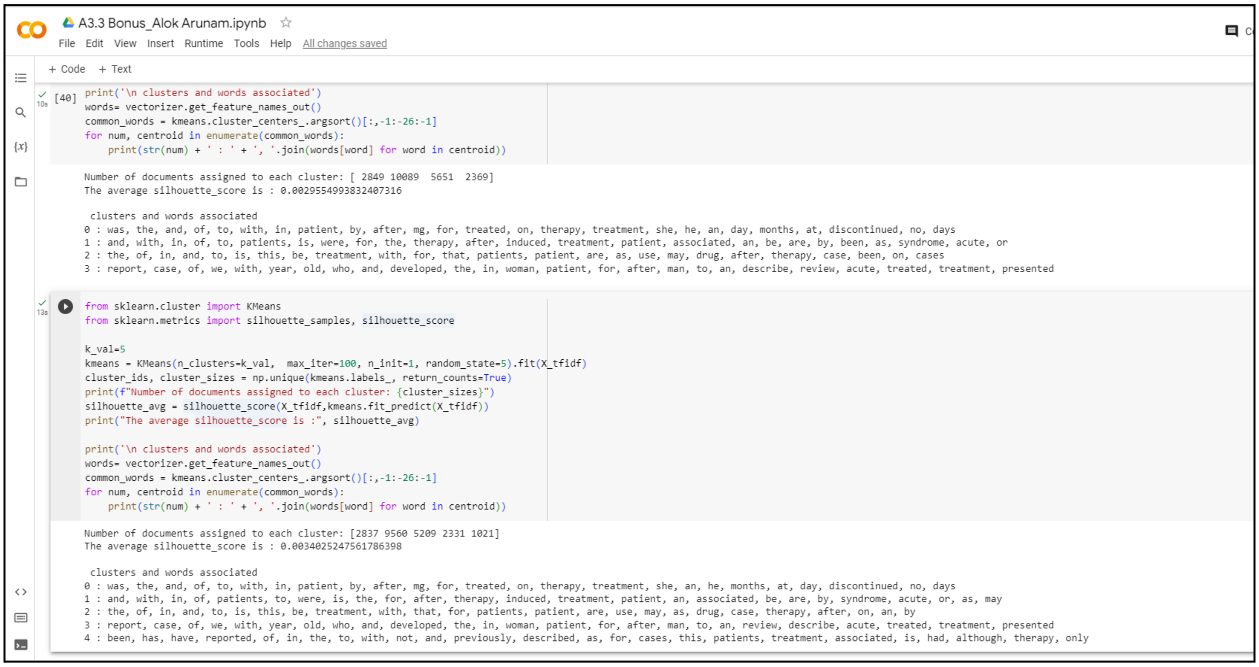
I’ve also tried to generate the common words associated with each clusters and these have been plotted further using the WordCloud.



```python
for i in range(2,11):
  k_val=2
  model = KMeans(n_clusters=i,  max_iter=100, n_init=1, random_state=5).fit(X_tfidf)
  cluster_ids, cluster_sizes = np.unique(model.labels_, return_counts=True)
  print(f"Number of clusters: {i}")
  print(f"Number of documents assigned to each cluster: {cluster_sizes}")
  silhouette_avg = silhouette_score(X_tfidf,model.fit_predict(X_tfidf))
  print("The average silhouette_score is :", silhouette_avg)
```

```
Number of clusters: 2
Number of documents assigned to each cluster: [12579  8379]
The average silhouette_score is : 0.0028927114321090353
Number of clusters: 3
Number of documents assigned to each cluster: [ 3159 10192  7607]
The average silhouette_score is : 0.003106472231438018
Number of clusters: 4
Number of documents assigned to each cluster: [ 2849 10089  5651  2369]
The average silhouette_score is : 0.0029554993832407316
Number of clusters: 5
Number of documents assigned to each cluster: [2837 9560 5209 2331 1021]
The average silhouette_score is : 0.0034025247561786398
Number of clusters: 6
Number of documents assigned to each cluster: [7309 3627 2340 2257  968 4457]
The average silhouette_score is : 0.003748274949452854
Number of clusters: 7
Number of documents assigned to each cluster: [2536 3092 2274 2089  966 6334 3667]
The average silhouette_score is : 0.004081786617058923
Number of clusters: 8
Number of documents assigned to each cluster: [1083 3233 4099 2476 1326 1047  907 6787]
The average silhouette_score is : 0.004470285859912278
Number of clusters: 9
Number of documents assigned to each cluster: [1076 2026 4008 2434  966 1092  843 7212 1301]
The average silhouette_score is : 0.005171971040185306S
Number of clusters: 10
Number of documents assigned to each cluster: [1074 1729 3520 2401 1482 1029  796 6703 1277  947]
The average silhouette_score is : 0.005535688185725818
```

✓ 2m 41s completed at 12:43 PM

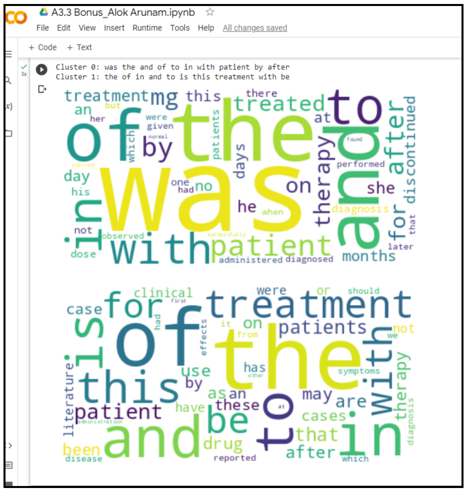

A3.3 Bonus_Alok Arunam.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[40]
10s
13s

```python
print('\n clusters and words associated')
words= vectorizer.get_feature_names_out()
common_words = kmeans.cluster_centers_.argsort()[:,-1:-26:-1]
for num, centroid in enumerate(common_words):
    print(str(num) + ' : ' + ', '.join(words[word] for word in centroid))
```

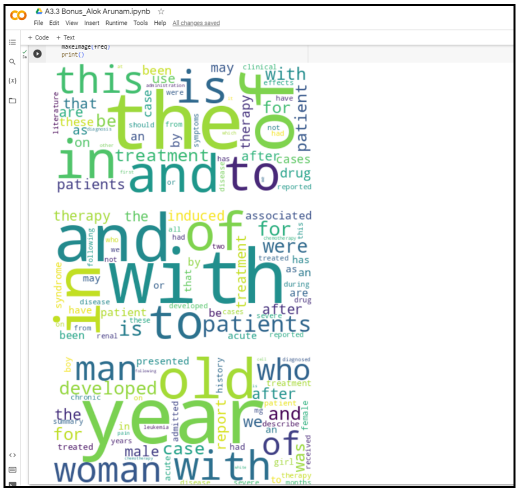
```
Number of documents assigned to each cluster: [ 2849 10089  5651  2369]
The average silhouette_score is : 0.0029554993832407316

 clusters and words associated
0 : was, the, and, of, to, with, in, patient, by, after, mg, for, treated, on, therapy, treatment, she, he, an, day, months, at, discontinued, no, days
1 : and, with, in, of, to, patients, is, were, for, the, therapy, after, induced, treatment, patient, associated, an, be, are, by, been, as, syndrome, acute, or
2 : the, of, in, and, to, is, this, be, treatment, with, for, that, patients, patient, are, as, use, may, drug, after, therapy, case, been, on, cases
3 : report, case, of, we, with, year, old, who, and, developed, the, in, woman, patient, for, after, man, to, an, describe, review, acute, treated, treatment, presented
```

```python
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score

k_val=5
kmeans = KMeans(n_clusters=k_val,  max_iter=100, n_init=1, random_state=5).fit(X_tfidf)
cluster_ids, cluster_sizes = np.unique(kmeans.labels_, return_counts=True)
print(f"Number of documents assigned to each cluster: {cluster_sizes}")
silhouette_avg = silhouette_score(X_tfidf,kmeans.fit_predict(X_tfidf))
print("The average silhouette_score is :", silhouette_avg)

print('\n clusters and words associated')
words= vectorizer.get_feature_names_out()
common_words = kmeans.cluster_centers_.argsort()[:,-1:-26:-1]
for num, centroid in enumerate(common_words):
    print(str(num) + ' : ' + ', '.join(words[word] for word in centroid))
```

```
Number of documents assigned to each cluster: [2837 9560 5209 2331 1021]
The average silhouette_score is : 0.0034025247561786398

 clusters and words associated
0 : was, the, and, of, to, with, in, patient, by, after, mg, for, treated, on, therapy, treatment, she, an, he, months, at, day, discontinued, no, days
1 : and, with, in, of, patients, to, were, is, the, for, after, therapy, induced, treatment, patient, an, associated, be, are, by, syndrome, acute, or, as, may
2 : the, of, in, and, to, is, this, be, treatment, with, that, for, patients, patient, are, use, may, as, drug, case, therapy, after, on, an, by
3 : report, case, of, we, with, year, old, who, and, developed, the, in, woman, patient, for, after, man, to, an, review, describe, acute, treated, treatment, presented
4 : been, has, have, reported, of, in, the, to, with, not, and, previously, described, as, for, cases, this, patients, treatment, associated, is, had, although, therapy, only
```

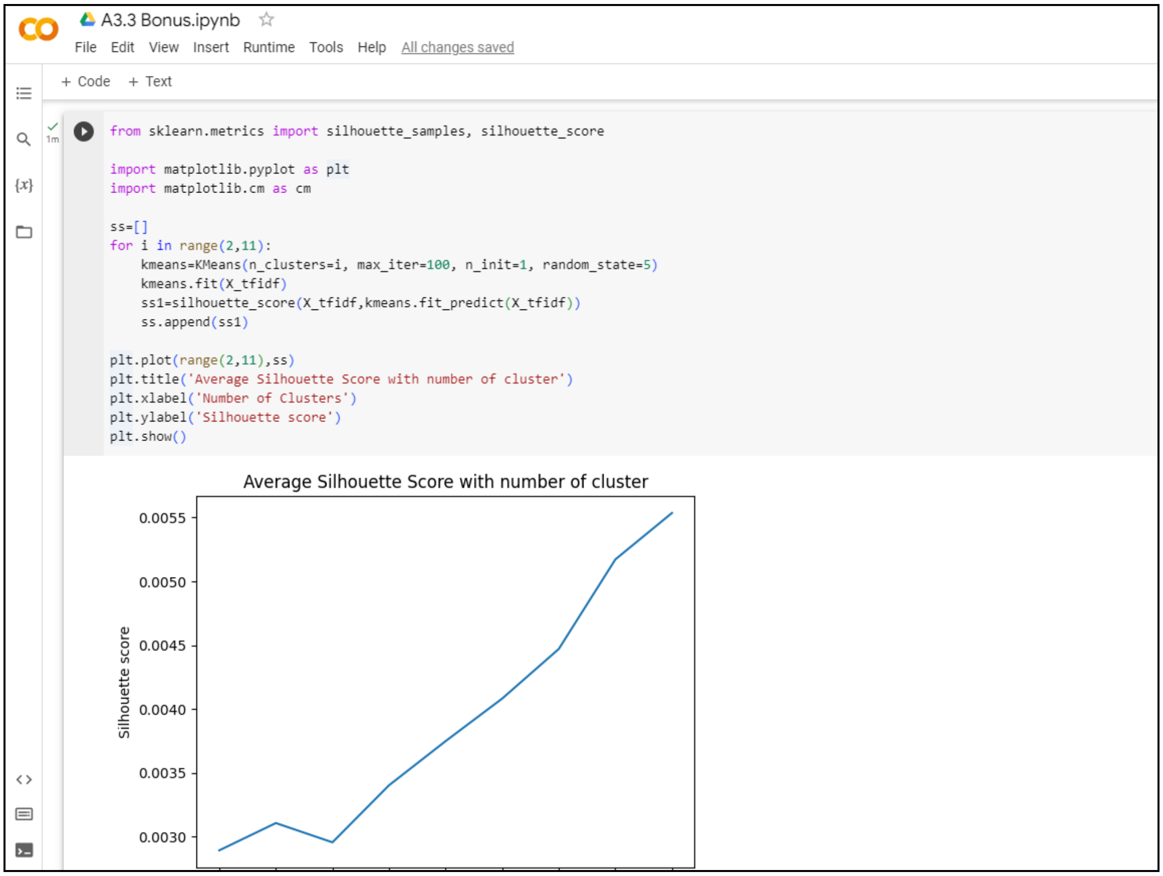

A3.3 Bonus_Alok Arunam.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
Cluster 0: was the and of to in with patient by after
Cluster 1: the of in and to is this treatment with be
```

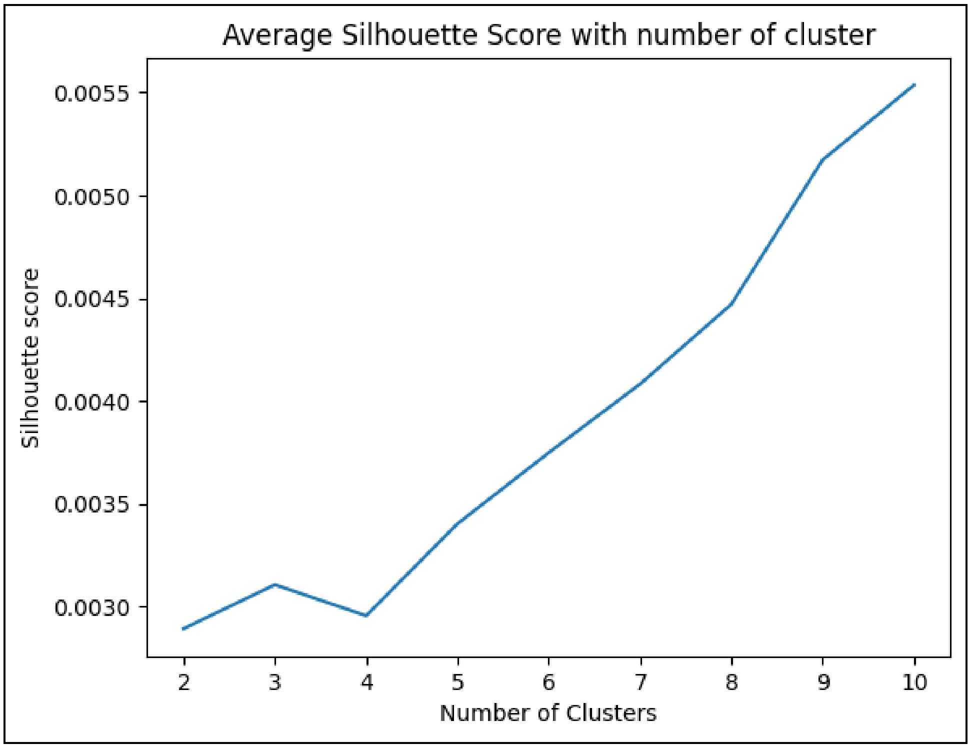

A3.3 Bonus_Alok Arunam.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
    makeimage(freq)
    print()
```

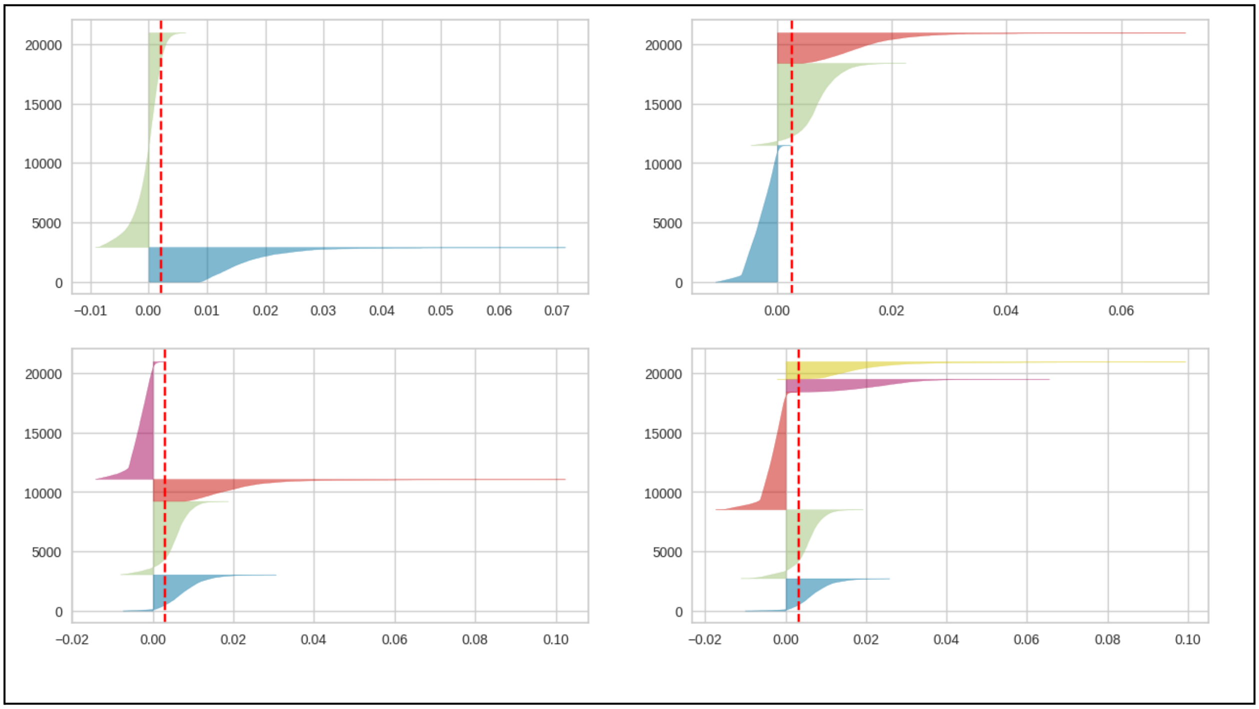
b. **Compute the silhouette score for each run and plot a graph to show how they change with k.**
**Ans:** Silhouette score is a metric used to calculate the goodness of clustering technique. It’s value ranges from -1 to 1. The silhouette score for the different k-value has been calculated.



```
from sklearn.metrics import silhouette_samples, silhouette_score

import matplotlib.pyplot as plt
import matplotlib.cm as cm

ss=[]
for i in range(2,11):
    kmeans=KMeans(n_clusters=i, max_iter=100, n_init=1, random_state=5)
    kmeans.fit(X_tfidf)
    ss1=silhouette_score(X_tfidf,kmeans.fit_predict(X_tfidf))
    ss.append(ss1)

plt.plot(range(2,11),ss)
plt.title('Average Silhouette Score with number of cluster')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette score')
plt.show()
```



The silhouette scores of clusters for different k_value have been visualised to indicate good clusters. Each color in the su-graph indicates distinct cluster. The negative value within the cluster grouping indicates there are data points within the cluster which are closer to some other cluster. As indicated in the graph below, the higher number of clusters represents the good grouping.



c. **Write a program that can determine automatically the value for “k” that finds the best segregation of POS and NEG statements. (Hint: k may be greater than 2. Use Purity measures.)**
Ans: The clusters are analysed further to analyse the segregation of POS and NEG statements. For k=3 and k=4, the proportion of POS and NEG statements across cluster groups are shown in the following table:

| **Cluster/Label** | **POS** | **NEG** |
|---|---|---|
| **0** | 320 | 2839 |
| **1** | 2696 | 7496 |
| **2** | 1256 | 6351 |

| **Cluster/Label** | **POS** | **NEG** |
|---|---|---|
| **0** | 269 | 2580 |
| **1** | 2135 | 7954 |
| **2** | 931 | 4720 |
| 3 | 937 | 1432 |

As evident, the higher K value= 4 or above will be suitable for the best segregation.