In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [1]:
```
cd /content/drive/MyDrive
```

```
/content/drive/MyDrive
```

In [2]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
asm_with_size=pd.read_csv("asm_with_size.csv").drop(["Unnamed: 0"],axis=1)
asm_with_size.head()
```

Out[3]:

| | ID | size_asm | Class |
|---|---|---|---|
| **0** | 01azqd4InC7m9JpocGv5 | 56.229886 | 9 |
| **1** | 01IsoiSMh5gxyDYTl4CB | 13.999378 | 2 |
| **2** | 01jsnpXSAlgw6aPeDxrU | 8.507785 | 9 |
| **3** | 01kcPWA9K2BOxQeS5Rju | 0.078190 | 1 |
| **4** | 01SuzwMJEIXsK7A8dQbl | 0.996723 | 8 |

# Byte Unigrams + size of Byte file

In [4]:
```
byte_unigrams=pd.read_csv("result_with_size.csv")
byte_unigrams.drop(["Unnamed: 0"],axis=1,inplace=True)
byte_unigrams.head()
```

Out[4]:

| | ID | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0a |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 01azqd4InC7m9JpocGv5 | 601905 | 3905 | 2816 | 3832 | 3345 | 3242 | 3650 | 3201 | 2965 | 3205 | 3211 |
| **1** | 01IsoiSMh5gxyDYTl4CB | 39755 | 8337 | 7249 | 7186 | 8663 | 6844 | 8420 | 7589 | 9291 | 358 | 340 |
| **2** | 01jsnpXSAlgw6aPeDxrU | 93506 | 9542 | 2568 | 2438 | 8925 | 9330 | 9007 | 2342 | 9107 | 2457 | 2655 |
| **3** | 01kcPWA9K2BOxQeS5Rju | 21091 | 1213 | 726 | 817 | 1257 | 625 | 550 | 523 | 1078 | 473 | 516 |
| **4** | 01SuzwMJEIXsK7A8dQbl | 19764 | 710 | 302 | 433 | 559 | 410 | 262 | 249 | 422 | 223 | 237 |

5 rows × 260 columns

# Top 800 Image Features of asm file

In [5]:
```
top_800_image_asm_df3=pd.read_csv("top_800_image_asm_df3.csv")
top_800_image_asm_df3.drop_duplicates(inplace=True)
top_800_image_asm_df3.head()
```

Out[5]:

| | 01azqd4InC7m9JpocGv5 | 72 | 69 | 65 | 68 | 69.1 | 82 | 58 | 48 | 48.1 | 52 | 48.2 | 48.3 | 48.4 | 48.5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 01IsoiSMh5gxyDYTl4CB | 46 | 116 | 101 | 120 | 116 | 58 | 48 | 48 | 52 | 48 | 49 | 48 | 48 | 48 |
| **1** | 01jsnpXSAlgw6aPeDxrU | 72 | 69 | 65 | 68 | 69 | 82 | 58 | 48 | 48 | 52 | 48 | 48 | 48 | 48 |
| **2** | 01kcPWA9K2BOxQeS5Rju | 72 | 69 | 65 | 68 | 69 | 82 | 58 | 49 | 48 | 48 | 48 | 48 | 48 | 48 |
| **3** | 01SuzwMJEIXsK7A8dQbl | 72 | 69 | 65 | 68 | 69 | 82 | 58 | 48 | 48 | 52 | 48 | 48 | 48 | 48 |
| **4** | 02IOCvYEy8mjiuAQHax3 | 72 | 69 | 65 | 68 | 69 | 82 | 58 | 48 | 48 | 52 | 48 | 48 | 48 | 48 |

5 rows × 801 columns

In [6]:
```
top_800_image_asm_df3.shape
```

Out[6]:
```
(10867, 801)
```

In [7]:
```
row0=[top_800_image_asm_df3.columns[0]]+list(map(float,top_800_image_asm_df3
print(row0)
a=pd.DataFrame(np.array(row0).reshape(1,-1))
a.columns=["ID"]+[str(i)+"image_features" for i in range(1,801)]
top_800_image_asm_df3.columns=["ID"]+[str(i)+"image_features" for i in range
image_features=pd.concat([a,top_800_image_asm_df3])
```

```
['01azqd4InC7m9JpocGv5', 72.0, 69.0, 65.0, 68.0, 69.1, 82.0, 58.0, 48.0, 48.1,
52.0, 48.2, 48.3, 48.4, 48.5, 48.6, 9.0, 9.1, 9.2, 9.3, 9.4, 9.5, 9.6, 59.0, 1
3.0, 10.0, 72.1, 69.2, 65.1, 68.1, 69.3, 82.1, 58.1, 48.7, 48.8, 52.1, 48.9, 4
8.1, 48.11, 48.12, 48.13, 9.7, 9.8, 9.9, 9.1, 9.11, 9.12, 9.13, 59.1, 32.0, 43.
0, 45.0, 45.1, 45.2, 45.3, 45.4, 45.5, 45.6, 45.7, 45.8, 45.9, 45.1, 45.11, 45.
12, 45.13, 45.14, 45.15, 45.16, 45.17, 45.18, 45.19, 45.2, 45.21, 45.22, 45.23,
45.24, 45.25, 45.26, 45.27, 45.28, 45.29, 45.3, 45.31, 45.32, 45.33, 45.34, 45.
35, 45.36, 45.37, 45.38, 45.39, 45.4, 45.41, 45.42, 45.43, 45.44, 45.45, 45.46,
45.47, 45.48, 45.49, 45.5, 45.51, 45.52, 45.53, 45.54, 45.55, 45.56, 45.57, 45.
58, 45.59, 45.6, 45.61, 45.62, 45.63, 45.64, 45.65, 45.66, 45.67, 45.68, 45.69,
45.7, 45.71, 45.72, 43.1, 13.1, 10.1, 72.2, 69.4, 65.2, 68.2, 69.5, 82.2, 58.2,
48.14, 48.15, 52.2, 48.16, 48.17, 48.18, 48.19, 48.2, 9.14, 9.15, 9.16, 9.17,
9.18, 9.19, 9.2, 59.2, 32.1, 124.0, 32.2, 32.3, 32.4, 84.0, 104.0, 105.0, 115.
0, 32.5, 102.0, 105.1, 108.0, 101.0, 9.21, 104.1, 97.0, 115.1, 32.6, 98.0, 101.
1, 101.2, 110.0, 32.7, 103.0, 101.3, 110.1, 101.4, 114.0, 97.1, 116.0, 101.5, 1
00.0, 32.8, 98.1, 121.0, 32.9, 84.1, 104.2, 101.6, 32.1, 73.0, 110.2, 116.1, 10
1.7, 114.1, 97.2, 99.0, 116.2, 105.2, 118.0, 101.8, 32.11, 68.3, 105.3, 115.2,
97.3, 115.3, 115.4, 101.9, 109.0, 98.2, 108.1, 101.1, 114.2, 32.12, 40.0, 73.1,
68.4, 65.3, 41.0, 32.13, 32.14, 32.15, 32.16, 124.1, 13.2, 10.2, 72.3, 69.6, 6
5.4, 68.5, 69.7, 82.3, 58.3, 48.21, 48.22, 52.3, 48.23, 48.24, 48.25, 48.26, 4
8.27, 9.22, 9.23, 9.24, 9.25, 9.26, 9.27, 9.28, 59.3, 32.17, 124.2, 9.29, 32.1
8, 32.19, 32.2, 32.21, 32.22, 32.23, 67.0, 111.0, 112.0, 121.1, 114.3, 105.4, 1
03.1, 104.3, 116.3, 9.3, 40.1, 99.1, 41.1, 32.24, 50.0, 48.28, 49.0, 51.0, 32.2
5, 72.4, 101.11, 120.0, 45.73, 82.4, 97.4, 121.2, 115.5, 44.0, 32.26, 60.0, 11
5.6, 117.0, 112.1, 112.2, 111.1, 114.4, 116.4, 64.0, 104.4, 101.12, 120.1, 45.7
4, 114.5, 97.5, 121.3, 115.7, 46.0, 99.2, 111.2, 109.1, 62.0, 9.31, 32.27, 32.2
8, 32.29, 32.3, 124.3, 13.3, 10.3, 72.5, 69.8, 65.5, 68.6, 69.9, 82.5, 58.4, 4
8.29, 48.3, 52.4, 48.31, 48.32, 48.33, 48.34, 48.35, 9.32, 9.33, 9.34, 9.35, 9.
36, 9.37, 9.38, 59.4, 32.31, 124.4, 9.39, 9.4, 9.41, 32.32, 76.0, 105.5, 99.3,
101.13, 110.3, 115.8, 101.14, 32.33, 105.6, 110.4, 102.1, 111.3, 58.5, 32.34, 3
2.35, 32.36, 32.37, 32.38, 32.39, 32.4, 32.41, 32.42, 32.43, 32.44, 32.45, 32.4
6, 32.47, 32.48, 32.49, 9.42, 9.43, 9.44, 32.5, 32.51, 32.52, 32.53, 124.5, 13.
4, 10.4, 72.6, 69.1, 65.6, 68.7, 69.11, 82.6, 58.6, 48.36, 48.37, 52.5, 48.38,
48.39, 48.4, 48.41, 48.42, 9.45, 9.46, 9.47, 9.48, 9.49, 9.5, 9.51, 59.5, 32.5
4, 124.6, 9.52, 9.53, 9.54, 9.55, 32.55, 32.56, 32.57, 77.0, 105.7, 99.4, 114.
6, 111.4, 115.9, 111.5, 102.2, 116.5, 9.56, 9.57, 9.58, 9.59, 32.58, 32.59, 32.
6, 32.61, 124.7, 13.5, 10.5, 72.7, 69.12, 65.7, 68.8, 69.13, 82.7, 58.7, 48.43,
48.44, 52.6, 48.45, 48.46, 48.47, 48.48, 48.49, 9.6, 9.61, 9.62, 9.63, 9.64, 9.
65, 9.66, 59.6, 32.62, 43.2, 45.75, 45.76, 45.77, 45.78, 45.79, 45.8, 45.81, 4
5.82, 45.83, 45.84, 45.85, 45.86, 45.87, 45.88, 45.89, 45.9, 45.91, 45.92, 45.9
3, 45.94, 45.95, 45.96, 45.97, 45.98, 45.99, 45.1, 45.101, 45.102, 45.103, 45.1
04, 45.105, 45.106, 45.107, 45.108, 45.109, 45.11, 45.111, 45.112, 45.113, 45.1
14, 45.115, 45.116, 45.117, 45.118, 45.119, 45.12, 45.121, 45.122, 45.123, 45.1
24, 45.125, 45.126, 45.127, 45.128, 45.129, 45.13, 45.131, 45.132, 45.133, 45.1
34, 45.135, 45.136, 45.137, 45.138, 45.139, 45.14, 45.141, 45.142, 45.143, 45.1
44, 45.145, 45.146, 45.147, 43.3, 13.6, 10.6, 72.8, 69.14, 65.8, 68.9, 69.15, 8
2.8, 58.8, 48.5, 48.51, 52.7, 48.52, 48.53, 48.54, 48.55, 48.56, 9.67, 9.68, 9.
69, 9.7, 9.71, 9.72, 9.73, 59.7, 13.7, 10.7, 72.9, 69.16, 65.9, 68.1, 69.17, 8
2.9, 58.9, 48.57, 48.58, 52.8, 48.59, 48.6, 48.61, 48.62, 48.63, 9.74, 9.75, 9.
```

```
76, 9.77, 9.78, 9.79, 9.8, 59.8, 32.63, 13.8, 10.8, 72.1, 69.18, 65.1, 68.11, 6
9.19, 82.1, 58.1, 48.64, 48.65, 52.9, 48.66, 48.67, 48.68, 48.69, 48.7, 13.9, 1
0.9, 72.11, 69.2, 65.11, 68.12, 69.21, 82.11, 58.11, 48.71, 48.72, 52.1, 48.73,
48.74, 48.75, 48.76, 48.77, 13.1, 10.1, 72.12, 69.22, 65.12, 68.13, 69.23, 82.1
2, 58.12, 48.78, 48.79, 52.11, 48.8, 48.81, 48.82, 48.83, 48.84, 9.81, 9.82, 9.
83, 9.84, 9.85, 9.86, 9.87, 9.88, 9.89, 46.1, 54.0, 56.0, 54.1, 112.3, 13.11, 1
0.11, 72.13, 69.24, 65.13, 68.14, 69.25, 82.13, 58.13, 48.85, 48.86, 52.12, 48.
87, 48.88, 48.89, 48.9, 48.91, 9.9, 9.91, 9.92, 9.93, 9.94, 9.95, 9.96, 9.97,
9.98, 46.2, 109.2, 109.3, 120.2, 13.12, 10.12, 72.14, 69.26, 65.14, 68.15, 69.2
7, 82.14, 58.14, 48.92, 48.93, 52.13, 48.94, 48.95, 48.96, 48.97, 48.98, 9.99,
9.1, 9.101, 9.102, 9.103, 9.104, 9.105, 9.106, 9.107, 46.3, 109.4, 111.6, 100.
1, 101.15, 108.2, 32.64, 102.3, 108.3, 97.6, 116.6, 13.13, 10.13, 72.15, 69.28,
65.15, 68.16, 69.29, 82.15, 58.15, 48.99, 48.1, 52.14, 48.101, 48.102, 48.103,
48.104, 48.105, 13.14, 10.14, 72.16, 69.3, 65.16, 68.17, 69.31, 82.16, 58.16, 4
8.106, 48.107, 52.15, 48.108, 48.109, 48.11, 48.111, 48.112, 9.108, 9.109, 9.1
1, 9.111, 9.112, 9.113, 9.114, 59.9, 32.65, 61.0, 61.1, 61.2, 61.3, 61.4, 61.5,
61.6, 61.7, 61.8, 61.9, 61.1, 61.11, 61.12, 61.13, 61.14, 61.15, 61.16, 61.17,
61.18, 61.19, 61.2, 61.21, 61.22, 61.23, 61.24, 61.25, 61.26, 61.27, 61.28, 61.
29, 61.3, 61.31, 61.32, 61.33, 61.34, 61.35, 61.36]
```

In [8]:

```
image_features.reset_index(drop=True,inplace=True)
image_features.head()
```

Out[8]:

| | ID | 1image_features | 2image_features | 3image_features | 4image_features | 5i |
|---|---|---|---|---|---|---|
| **0** | 01azqd4InC7m9JpocGv5 | 72.0 | 69.0 | 65.0 | 68.0 | |
| **1** | 01IsoiSMh5gxyDYTl4CB | 46 | 116 | 101 | 120 | |
| **2** | 01jsnpXSAlgw6aPeDxrU | 72 | 69 | 65 | 68 | |
| **3** | 01kcPWA9K2BOxQeS5Rju | 72 | 69 | 65 | 68 | |
| **4** | 01SuzwMJEIXsK7A8dQbl | 72 | 69 | 65 | 68 | |

5 rows × 801 columns

# Byte file bigrams

In [ ]:

```
cd /content
```

```
/content
```

In [ ]:
```
1 !wget --header="Host: doc-00-b0-docs.googleusercontent.com" --header="User-A
```

```
--2022-01-25 15:44:53--  https://doc-00-b0-docs.googleusercontent.com/docs/securesc/c2vv85f62mghnuqi9819ajflj77aiu7l/f79h5kit5q0q35sabl6uevabap90q1lc/1643125425000/18018082833065558536/18018082833065558536/1XGz3v8bq-C8KSN_RShkTZbqPX-F-iZu_?e=download&authuser=0&nonce=jonlj1dqub4lk&user=18018082833065558536&hash=0k4a6vadhjkh0chdqce6dlojqiepui4d (https://doc-00-b0-docs.googleusercontent.com/docs/securesc/c2vv85f62mghnuqi9819ajflj77aiu7l/f79h5kit5q0q35sabl6uevabap90q1lc/1643125425000/18018082833065558536/18018082833065558536/1XGz3v8bq-C8KSN_RShkTZbqPX-F-iZu_?e=download&authuser=0&nonce=jonlj1dqub4lk&user=18018082833065558536&hash=0k4a6vadhjkh0chdqce6dlojqiepui4d)
Resolving doc-00-b0-docs.googleusercontent.com (doc-00-b0-docs.googleusercontent.com)... 173.194.218.132, 2607:f8b0:400c:c14::84
Connecting to doc-00-b0-docs.googleusercontent.com (doc-00-b0-docs.googleusercontent.com)|173.194.218.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1622933383 (1.5G) [text/csv]
Saving to: ‘byte_files_bigram_df_2.csv’

byte_files_bigram_d 100%[===================>]   1.51G   112MB/s    in 14s

2022-01-25 15:45:07 (112 MB/s) - ‘byte_files_bigram_df_2.csv’ saved [1622933383/1622933383]
```

In [ ]:
```
1 byte_files_bigram_df_2=pd.read_csv("byte_files_bigram_df_2.csv")
2 byte_files_bigram_df_2.head()
```

Out[4]:

| | 01azqd4lnC7m9JpocGv5 | 273053 | 1002 | 801 | 1170 | 943 | 840 | 1125 | 1003 | 860 | 987 | 973 | 1127 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 01IsoiSMh5gxyDYTl4CB | 19852 | 719 | 64 | 43 | 159 | 10 | 6 | 10 | 35 | 8 | 12 | 9 |
| **1** | 01jsnpXSAlgw6aPeDxrU | 16032 | 592 | 157 | 144 | 509 | 590 | 551 | 146 | 523 | 154 | 155 | 150 |
| **2** | 01kcPWA9K2BOxQeS5Rju | 9903 | 204 | 59 | 69 | 103 | 34 | 19 | 21 | 55 | 14 | 21 | 17 |
| **3** | 01SuzwMJEIXsK7A8dQbl | 15288 | 58 | 20 | 110 | 8 | 11 | 3 | 5 | 8 | 2 | 0 | 3 |
| **4** | 02IOCvYEy8mjiuAQHax3 | 78958 | 19 | 3 | 4 | 5 | 2 | 0 | 2 | 5 | 2 | 2 | 2 |

5 rows × 66050 columns

In [ ]:
```
row0=list(byte_files_bigram_df_2.columns[0:1])+list(map(float,byte_files_big
print(row0)
a=pd.DataFrame(np.array(row0).reshape(1,-1))
a.columns=["ID"]+[i for i in range(1,66050)]
byte_files_bigram_df_2.columns=["ID"]+[i for i in range(1,66050)]
```

```
['01azqd4InC7m9JpocGv5', 273053.0, 1002.0, 801.0, 1170.0, 943.0, 840.0, 1125.
0, 1003.0, 860.0, 987.0, 973.0, 1127.0, 1278.0, 997.0, 1041.0, 889.0, 963.0,
820.0, 1079.0, 800.0, 1008.0, 1091.0, 966.0, 1102.0, 806.0, 864.0, 1056.0, 96
0.0, 1109.0, 1126.0, 799.0, 997.1, 813.0, 960.1, 835.0, 1331.0, 1305.0, 1112.
0, 983.0, 836.0, 806.1, 1160.0, 943.1, 1233.0, 1029.0, 857.0, 1086.0, 909.0,
1016.0, 831.0, 1111.0, 936.0, 1001.0, 1223.0, 1051.0, 918.0, 1015.0, 839.0, 9
77.0, 1049.0, 786.0, 1164.0, 836.1, 961.0, 1282.0, 1120.0, 1005.0, 1027.0, 82
8.0, 974.0, 1079.1, 1158.0, 1372.0, 971.0, 872.0, 862.0, 962.0, 851.0, 812.0,
980.0, 1192.0, 1540.0, 1311.0, 1018.0, 1004.0, 873.0, 1149.0, 998.0, 852.0, 1
043.0, 1025.0, 1040.0, 786.1, 825.0, 845.0, 1072.0, 1112.1, 962.1, 1146.0, 86
0.1, 961.1, 858.0, 847.0, 1418.0, 1052.0, 871.0, 1534.0, 874.0, 823.0, 999.0,
986.0, 987.1, 946.0, 1148.0, 1011.0, 1095.0, 1085.0, 835.1, 1125.1, 1015.1, 9
50.0, 1080.0, 821.0, 872.1, 954.0, 1144.0, 1284.0, 834.0, 824.0, 1195.0, 100
2.1, 1620.0, 954.1, 1632.0, 877.0, 805.0, 1126.1, 1049.1, 997.2, 1810.0, 814.
0, 1072.1, 1202.0, 817.0, 992.0, 836.2, 989.0, 877.1, 1223.1, 1016.1, 925.0,
815.0, 814.1, 873.1, 1127.1, 818.0, 951.0, 964.0, 1019.0, 1361.0, 983.1, 126
9.0, 1062.0, 1215.0, 792.0, 1128.0, 831.1, 841.0, 1014.0, 840.1, 1136.0, 929.
0, 1092.0, 1109.1, 800.1, 903.0, 853.0, 831.2, 964.1, 1065.0, 1112.2, 860.2,
950.1, 817.1, 1028.0, 1167.0, 892.0, 1056.1, 1124.0, 1030.0, 978.0, 858.1, 96
```

In [ ]:
```
byte_bigrams=pd.concat([a,byte_files_bigram_df_2])
byte_bigrams.head()
```

Out[6]:

| | ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 01azqd4InC7m9JpocGv5 | 273053.0 | 1002.0 | 801.0 | 1170.0 | 943.0 | 840.0 | 1125.0 | 1003.0 | 860.0 | 9 |
| **0** | 01IsoiSMh5gxyDYTl4CB | 19852 | 719 | 64 | 43 | 159 | 10 | 6 | 10 | 35 | |
| **1** | 01jsnpXSAlgw6aPeDxrU | 16032 | 592 | 157 | 144 | 509 | 590 | 551 | 146 | 523 | |
| **2** | 01kcPWA9K2BOxQeS5Rju | 9903 | 204 | 59 | 69 | 103 | 34 | 19 | 21 | 55 | |
| **3** | 01SuzwMJEIXsK7A8dQbl | 15288 | 58 | 20 | 110 | 8 | 11 | 3 | 5 | 8 | |

5 rows × 66050 columns

In [ ]:
```
cd /content
```

```
/content
```

In [ ]:
```
byte_bigrams.to_csv("byte_bigrams")
```

In [ ]:
```
byte_bigrams.shape
```

Out[8]:
```
(10868, 66050)
```

# Important features in byte bigrams

## Important features in byte bigrams

In [9]:
```
byte_bigrams=pd.read_csv("byte_bigrams")
```

In [10]:
```
byte_bigrams
```

Out[10]:

| | Unnamed: 0 | ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 01azqd4InC7m9JpocGv5 | 273053.0 | 1002.0 | 801.0 | 1170.0 | 943.0 | 840.0 | 1125.0 |
| **1** | 0 | 01IsoiSMh5gxyDYTl4CB | 19852.0 | 719.0 | 64.0 | 43.0 | 159.0 | 10.0 | 6.0 |
| **2** | 1 | 01jsnpXSAlgw6aPeDxrU | 16032.0 | 592.0 | 157.0 | 144.0 | 509.0 | 590.0 | 551.0 |
| **3** | 2 | 01kcPWA9K2BOxQeS5Rju | 9903.0 | 204.0 | 59.0 | 69.0 | 103.0 | 34.0 | 19.0 |
| **4** | 3 | 01SuzwMJEIXsK7A8dQbl | 15288.0 | 58.0 | 20.0 | 110.0 | 8.0 | 11.0 | 3.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **10863** | 10862 | loIP1tiwELF9YNZQjSUO | 3189.0 | 8.0 | 10.0 | 24.0 | 7.0 | 7.0 | 3.0 |
| **10864** | 10863 | LOP6HaJKXpkic5dyuVnT | 1805.0 | 27.0 | 7.0 | 2.0 | 3.0 | 2.0 | 1.0 |
| **10865** | 10864 | LOqA6FX02GWguYrI1Zbe | 2640.0 | 13.0 | 2.0 | 85.0 | 3.0 | 1.0 | 3.0 |
| **10866** | 10865 | LoWgaidpb2IUM5ACcSGO | 2476.0 | 2.0 | 1.0 | 0.0 | 4.0 | 0.0 | 3.0 |
| **10867** | 10866 | lS0IVqXeJrN6Dzi9Pap1 | 2481.0 | 2.0 | 4.0 | 3.0 | 3.0 | 2.0 | 4.0 |

10868 rows × 66051 columns

In [11]:
```
byte_bigrams.drop(["Unnamed: 0"],axis=1,inplace=True)
```

In [12]:
```
byte_bigrams.head(2)
```

Out[12]:

| | ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 01azqd4InC7m9JpocGv5 | 273053.0 | 1002.0 | 801.0 | 1170.0 | 943.0 | 840.0 | 1125.0 | 1003.0 | 860.0 | 987 |
| **1** | 01IsoiSMh5gxyDYTl4CB | 19852.0 | 719.0 | 64.0 | 43.0 | 159.0 | 10.0 | 6.0 | 10.0 | 35.0 | 8 |

2 rows × 66050 columns

In [13]:
```
from sklearn.ensemble import RandomForestClassifier

RandClf= RandomForestClassifier(n_estimators=1000,n_jobs=-1)

RandClf.fit(byte_bigrams.drop(['ID'],axis=1),asm_with_size['Class'])
```

Out[13]:
```
RandomForestClassifier(n_estimators=1000, n_jobs=-1)
```

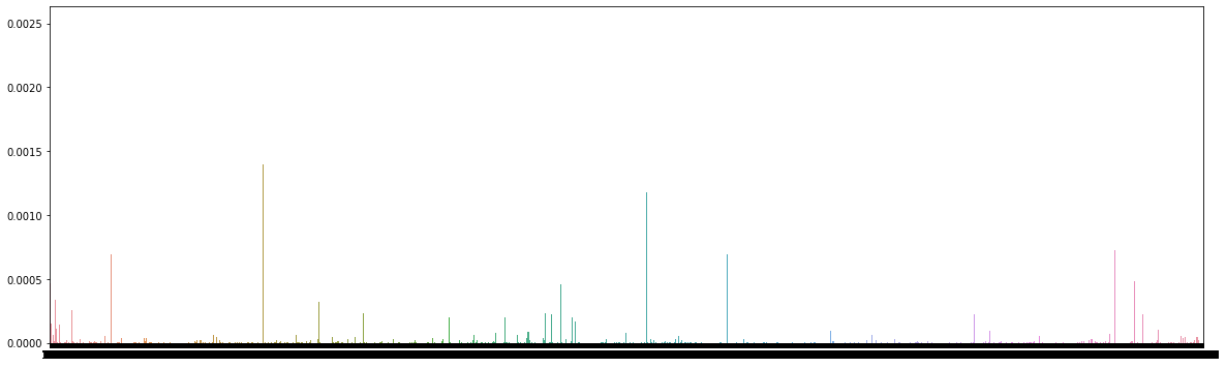
In [15]:
```
importance=RandClf.feature_importances_
```

In [18]:
```
selected_features=[]
columns=byte_bigrams.columns
for i,v in enumerate(importance):
  if v >10**-4:
    print('Feature: {}, Score: {}'.format(columns[i+1],v))
    selected_features.append(i+1)
print(len(selected_features))
```

```
Feature: 1, Score: 0.0007536980579742581
Feature: 2, Score: 0.00107016808786619
Feature: 3, Score: 0.000776354953533376
Feature: 4, Score: 0.00015946251674409237
Feature: 5, Score: 0.0008051904790494697
Feature: 6, Score: 0.0005004804410631287
Feature: 7, Score: 0.0004809434585655047
Feature: 8, Score: 0.0001340394277421781
Feature: 9, Score: 0.000565073138584958
Feature: 10, Score: 0.00015890756020492947
Feature: 11, Score: 0.00010964565518978904
Feature: 12, Score: 0.00027771770175073947
Feature: 13, Score: 0.0002985524534632795
Feature: 14, Score: 0.0002044041501576407
Feature: 15, Score: 0.00034195143109695496
Feature: 16, Score: 0.0003969604383023127
Feature: 17, Score: 0.000538866771814381
Feature: 18, Score: 0.00016230667601659275
Feature: 19, Score: 0.0006146987073346273
```

In [19]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(20,6))
sns.barplot([x for x in range(len(importance))], importance)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```




In [20]:
```
important_byte_bigrams=byte_bigrams.iloc[:,selected_features]
```

In [21]:
```
1 important_byte_bigrams.head(3)
```

Out[21]:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 1: |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 273053.0 | 1002.0 | 801.0 | 1170.0 | 943.0 | 840.0 | 1125.0 | 1003.0 | 860.0 | 987.0 | 973.0 | 1127.0 | 1278.( |
| **1** | 19852.0 | 719.0 | 64.0 | 43.0 | 159.0 | 10.0 | 6.0 | 10.0 | 35.0 | 8.0 | 12.0 | 9.0 | 23.( |
| **2** | 16032.0 | 592.0 | 157.0 | 144.0 | 509.0 | 590.0 | 551.0 | 146.0 | 523.0 | 154.0 | 155.0 | 150.0 | 525.( |

3 rows × 2199 columns

# asm unigrams

In [22]:
```
1 asm_unigrams=pd.read_csv("asmoutputfile.csv")
2 asm_unigrams.head()
```

Out[22]:

| | ID | HEADER: | .text: | .Pav: | .idata: | .data: | .bss: | .rdata: | .edata: | .rsrc: | .tl |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 01kcPWA9K2BOxQeS5Rju | 19 | 744 | 0 | 127 | 57 | 0 | 323 | 0 | 3 | |
| **1** | 1E93CpP60RHFNiT5Qfvn | 17 | 838 | 0 | 103 | 49 | 0 | 0 | 0 | 3 | |
| **2** | 3ekVow2ajZHbTnBcsDfX | 17 | 427 | 0 | 50 | 43 | 0 | 145 | 0 | 3 | |
| **3** | 3X2nY7iQaPBIWDrAZqJe | 17 | 227 | 0 | 43 | 19 | 0 | 0 | 0 | 3 | |
| **4** | 46OZzdsSKDCFV8h7XWxf | 17 | 402 | 0 | 59 | 170 | 0 | 0 | 0 | 3 | |

In [ ]:
```
1
```

In [23]:
```
1  #normalizing byte unigrams
2  from tqdm import tqdm
3  def normalize(df):
4      result1 = df.copy()
5      for feature_name in tqdm(df.columns):
6          if (str(feature_name) != str('ID') and str(feature_name)!=str('Class
7              max_value = df[feature_name].max()
8              min_value = df[feature_name].min()
9              result1[feature_name] = (df[feature_name] - min_value) / (max_va
10     return result1
```

In [24]:
```
1 normalized_byte_unigrams=normalize(byte_unigrams)
2 normalized_byte_unigrams.head()
```

```
100%|██████████| 260/260 [00:02<00:00, 124.84it/s]
```

Out[24]:

| | ID | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|---|
| **0** | 01azqd4InC7m9JpocGv5 | 0.262806 | 0.005498 | 0.001567 | 0.002067 | 0.002048 | 0.001835 | 0.002058 |
| **1** | 01IsoiSMh5gxyDYTl4CB | 0.017358 | 0.011737 | 0.004033 | 0.003876 | 0.005303 | 0.003873 | 0.004747 |
| **2** | 01jsnpXSAlgw6aPeDxrU | 0.040827 | 0.013434 | 0.001429 | 0.001315 | 0.005464 | 0.005280 | 0.005078 |
| **3** | 01kcPWA9K2BOxQeS5Rju | 0.009209 | 0.001708 | 0.000404 | 0.000441 | 0.000770 | 0.000354 | 0.000310 |
| **4** | 01SuzwMJEIXsK7A8dQbl | 0.008629 | 0.001000 | 0.000168 | 0.000234 | 0.000342 | 0.000232 | 0.000148 |

5 rows × 260 columns

In [25]:
```
1 normalized_image_features=normalize(image_features.drop(["ID"],axis=1).astyp
2 normalized_image_features["ID"]=image_features["ID"]
3 normalized_image_features.head()
```

```
100%|██████████| 800/800 [00:00<00:00, 1148.27it/s]
```

Out[25]:

| | 1image_features | 2image_features | 3image_features | 4image_features | 5image_features | 6image_fe |
|---|---|---|---|---|---|---|
| **0** | 0.481928 | 0.302632 | 0.000000 | 0.277778 | 0.293056 | 0.5 |
| **1** | 0.168675 | 0.921053 | 0.705882 | 1.000000 | 0.944444 | 0.1 |
| **2** | 0.481928 | 0.302632 | 0.000000 | 0.277778 | 0.291667 | 0.5 |
| **3** | 0.481928 | 0.302632 | 0.000000 | 0.277778 | 0.291667 | 0.5 |
| **4** | 0.481928 | 0.302632 | 0.000000 | 0.277778 | 0.291667 | 0.5 |

5 rows × 801 columns

In [26]:
```
normalized_byte_bigrams=normalize(important_byte_bigrams)
normalized_byte_bigrams["ID"]=byte_bigrams["ID"]
normalized_byte_bigrams.head()
```

```
100%|██████████| 2199/2199 [00:01<00:00, 1142.91it/s]
```

Out[26]:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.127389 | 0.079943 | 0.054323 | 0.088980 | 0.064972 | 0.090303 | 0.109255 | 0.121901 | 0.057772 | 0.044 |
| **1** | 0.009262 | 0.057364 | 0.004340 | 0.003270 | 0.010955 | 0.001075 | 0.000583 | 0.001215 | 0.002351 | 0.000 |
| **2** | 0.007479 | 0.047232 | 0.010648 | 0.010951 | 0.035070 | 0.063427 | 0.053511 | 0.017744 | 0.035134 | 0.006 |
| **3** | 0.004620 | 0.016276 | 0.004001 | 0.005248 | 0.007097 | 0.003655 | 0.001845 | 0.002552 | 0.003695 | 0.000 |
| **4** | 0.007132 | 0.004627 | 0.001356 | 0.008366 | 0.000551 | 0.001183 | 0.000291 | 0.000608 | 0.000537 | 0.000 |

5 rows × 2200 columns

In [27]:
```
normalized_asm_unigrams=normalize(asm_unigrams)
normalized_asm_unigrams.head()
```

```
100%|██████████| 52/52 [00:00<00:00, 642.24it/s]
```

Out[27]:

| | ID | HEADER: | .text: | .Pav: | .idata: | .data: | .bss: | .rdata: | .edata |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 01kcPWA9K2BOxQeS5Rju | 0.107345 | 0.001092 | 0.0 | 0.000761 | 0.000023 | 0.0 | 0.000084 | 0.0 |
| **1** | 1E93CpP60RHFNiT5Qfvn | 0.096045 | 0.001230 | 0.0 | 0.000617 | 0.000019 | 0.0 | 0.000000 | 0.0 |
| **2** | 3ekVow2ajZHbTnBcsDfX | 0.096045 | 0.000627 | 0.0 | 0.000300 | 0.000017 | 0.0 | 0.000038 | 0.0 |
| **3** | 3X2nY7iQaPBIWDrAZqJe | 0.096045 | 0.000333 | 0.0 | 0.000258 | 0.000008 | 0.0 | 0.000000 | 0.0 |
| **4** | 46OZzdsSKDCFV8h7XWxf | 0.096045 | 0.000590 | 0.0 | 0.000353 | 0.000068 | 0.0 | 0.000000 | 0.0 |

In [28]:
```
normalized_asm_size=normalize(asm_with_size)
normalized_asm_size.head()
```

```
100%|██████████| 3/3 [00:00<00:00, 917.52it/s]
```

Out[28]:

| | ID | size_asm | Class |
|---|---|---|---|
| **0** | 01azqd4InC7m9JpocGv5 | 0.400910 | 9 |
| **1** | 01IsoiSMh5gxyDYTl4CB | 0.099719 | 2 |
| **2** | 01jsnpXSAlgw6aPeDxrU | 0.060553 | 9 |
| **3** | 01kcPWA9K2BOxQeS5Rju | 0.000432 | 1 |
| **4** | 01SuzwMJEIXsK7A8dQbl | 0.006983 | 8 |

# Checking NULL values

In [29]:
```
def Null_values(df):
  count=0
  for index, i in enumerate(df.isnull().sum()):
    if i >0:
      count=count+1
      print(df.isnull().sum().index[index],"number of null values",i)
  if count==0:
    print("Zero Null values")
```

In [30]:
```
Null_values(normalized_asm_unigrams)
```

```
.BSS: number of null values 10868
.CODE number of null values 10868
rtn number of null values 10868
```

In [31]:
```
# removing Null values
normalized_asm_unigrams.columns
```

Out[31]:
```
Index(['ID', 'HEADER:', '.text:', '.Pav:', '.idata:', '.data:', '.bss:',
       '.rdata:', '.edata:', '.rsrc:', '.tls:', '.reloc:', '.BSS:', '.CODE',
       'jmp', 'mov', 'retf', 'push', 'pop', 'xor', 'retn', 'nop', 'sub', 'inc',
       'dec', 'add', 'imul', 'xchg', 'or', 'shr', 'cmp', 'call', 'shl', 'ror',
       'rol', 'jnb', 'jz', 'rtn', 'lea', 'movzx', '.dll', 'std::', ':dword',
       'edx', 'esi', 'eax', 'ebx', 'ecx', 'edi', 'ebp', 'esp', 'eip'],
      dtype='object')
```

In [32]:
```
normalized_asm_unigrams.drop([".BSS:",".CODE","rtn"],axis=1,inplace=True)
```

In [33]:
```
Null_values(normalized_byte_unigrams)
```

```
Zero Null values
```

In [34]:
```
Null_values(normalized_byte_bigrams)
```

```
Zero Null values
```

In [35]:
```
Null_values(normalized_image_features)
```

```
Zero Null values
```

In [36]:
```
Null_values(normalized_asm_size)
```

```
Zero Null values
```

## asm file size + asm file unigrams + byte file unigrams + byte file size + byte file bigrams

In [37]:
```
y_labels=normalized_asm_size["Class"]

merged_asm=normalized_asm_size.drop(["Class"],axis=1).merge(normalized_asm_u
```

In [38]:
```
merged_asm
```

Out[38]:

| | ID | size_asm | HEADER: | .text: | .Pav: | .idata: | .data: | .bss |
|---|---|---|---|---|---|---|---|---|
| **0** | 01azqd4InC7m9JpocGv5 | 0.400910 | 0.101695 | 0.032927 | 0.0 | 0.006937 | 0.542847 | 0.000000 |
| **1** | 01IsoiSMh5gxyDYTl4CB | 0.099719 | 0.000000 | 0.161391 | 0.0 | 0.003690 | 0.009758 | 0.000000 |
| **2** | 01jsnpXSAlgw6aPeDxrU | 0.060553 | 0.101695 | 0.101121 | 0.0 | 0.001821 | 0.000263 | 0.000000 |
| **3** | 01kcPWA9K2BOxQeS5Rju | 0.000432 | 0.107345 | 0.001092 | 0.0 | 0.000761 | 0.000023 | 0.000000 |
| **4** | 01SuzwMJEIXsK7A8dQbl | 0.006983 | 0.101695 | 0.015220 | 0.0 | 0.001234 | 0.001825 | 0.012842 |
| **...** | ... | ... | ... | ... | ... | ... | ... | .. |
| **10863** | loIP1tiwELF9YNZQjSUO | 0.080249 | 0.096045 | 0.000926 | 0.0 | 0.000653 | 0.104938 | 0.000000 |
| **10864** | LOP6HaJKXpkic5dyuVnT | 0.014462 | 0.096045 | 0.005131 | 0.0 | 0.000018 | 0.011050 | 0.000000 |
| **10865** | LOqA6FX02GWguYrI1Zbe | 0.013834 | 0.096045 | 0.004483 | 0.0 | 0.000018 | 0.010981 | 0.000000 |
| **10866** | LoWgaidpb2IUM5ACcSGO | 0.027948 | 0.096045 | 0.019997 | 0.0 | 0.000018 | 0.029290 | 0.000000 |
| **10867** | lS0IVqXeJrN6Dzi9Pap1 | 0.025171 | 0.096045 | 0.018791 | 0.0 | 0.000018 | 0.026036 | 0.000000 |

10868 rows × 50 columns

In [39]:
```
merged_byte=normalized_byte_unigrams.drop(['Class'],axis=1).merge(normalized
merged_byte.head()
```

Out[39]:

| | ID | 0 | 1_x | 2_x | 3_x | 4_x | 5_x | 6_x |
|---|---|---|---|---|---|---|---|---|
| **0** | 01azqd4InC7m9JpocGv5 | 0.262806 | 0.005498 | 0.001567 | 0.002067 | 0.002048 | 0.001835 | 0.002058 |
| **1** | 01IsoiSMh5gxyDYTl4CB | 0.017358 | 0.011737 | 0.004033 | 0.003876 | 0.005303 | 0.003873 | 0.004747 |
| **2** | 01jsnpXSAlgw6aPeDxrU | 0.040827 | 0.013434 | 0.001429 | 0.001315 | 0.005464 | 0.005280 | 0.005078 |
| **3** | 01kcPWA9K2BOxQeS5Rju | 0.009209 | 0.001708 | 0.000404 | 0.000441 | 0.000770 | 0.000354 | 0.000310 |
| **4** | 01SuzwMJEIXsK7A8dQbl | 0.008629 | 0.001000 | 0.000168 | 0.000234 | 0.000342 | 0.000232 | 0.000148 |

5 rows × 2458 columns

In [40]:
```
dataset1=merged_asm.merge(merged_byte,on="ID")
dataset1.head()
```

Out[40]:

| | ID | size_asm | HEADER: | .text: | .Pav: | .idata: | .data: | .bss: | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 01azqd4InC7m9JpocGv5 | 0.400910 | 0.101695 | 0.032927 | 0.0 | 0.006937 | 0.542847 | 0.000000 | 0. |
| **1** | 01IsoiSMh5gxyDYTl4CB | 0.099719 | 0.000000 | 0.161391 | 0.0 | 0.003690 | 0.009758 | 0.000000 | 0. |
| **2** | 01jsnpXSAlgw6aPeDxrU | 0.060553 | 0.101695 | 0.101121 | 0.0 | 0.001821 | 0.000263 | 0.000000 | 0. |
| **3** | 01kcPWA9K2BOxQeS5Rju | 0.000432 | 0.107345 | 0.001092 | 0.0 | 0.000761 | 0.000023 | 0.000000 | 0. |
| **4** | 01SuzwMJEIXsK7A8dQbl | 0.006983 | 0.101695 | 0.015220 | 0.0 | 0.001234 | 0.001825 | 0.012842 | 0. |

5 rows × 2507 columns

In [54]:
```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(dataset1.drop(["ID"],axis=1),
```

In [55]:
```
print("shape of X_train ",X_train.shape)
print("shape of y_train ",y_train.shape)
print("shape of X_test ",X_test.shape)
print("shape of y_test ",y_test.shape)
```

```
shape of X_train  (8694, 2506)
shape of y_train  (8694,)
shape of X_test  (2174, 2506)
shape of y_test  (2174,)
```

In [ ]:
```
pip install xgboost --upgrade

```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.7/dist-package
s (0.90)
Collecting xgboost
  Downloading xgboost-1.5.2-py3-none-manylinux2014_x86_64.whl (173.6 MB)
     |████████████████████████████████| 173.6 MB 9.5 kB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages
(from xgboost) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from xgboost) (1.19.5)
Installing collected packages: xgboost
  Attempting uninstall: xgboost
    Found existing installation: xgboost 0.90
    Uninstalling xgboost-0.90:
      Successfully uninstalled xgboost-0.90
Successfully installed xgboost-1.5.2
```

In [46]:

```
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import log_loss
from sklearn.calibration import CalibratedClassifierCV

```

In [ ]:
```
x_cfl=XGBClassifier(eval_metric="mlogloss")

prams={
    'learning_rate':[0.001,0.01,0.03,0.05,0.1,0.15,0.2],
     'n_estimators':[100,200,500,900,1500],
     'max_depth':[3,5,10],
    'colsample_bytree':[0.1,0.3,0.5,1],
    'subsample':[0.1,0.3,0.5,1]
}
random_cfl=RandomizedSearchCV(x_cfl,param_distributions=prams,verbose=10,n_j
random_cfl.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits

/usr/local/lib/python3.7/dist-packages/xgboost/sklearn.py:1224: UserWarning: Th
e use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option use_la
bel_encoder=False when constructing XGBClassifier object; and 2) Encode your la
bels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

Out[45]:
```
RandomizedSearchCV(estimator=XGBClassifier(base_score=None, booster=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None,
                                           enable_categorical=False,
                                           eval_metric='mlogloss', gamma=None,
                                           gpu_id=None, importance_type=None,
                                           interaction_constraints=None,
                                           learning_rate=None,
                                           max_delta_step=None, max_depth=None,
                                           min_child_weight=None, missing=nan,
                                           mo...
                                           predictor=None, random_state=None,
                                           reg_alpha=None, reg_lambda=None,
                                           scale_pos_weight=None,
                                           subsample=None, tree_method=None,
                                           validate_parameters=None,
                                           verbosity=None),
                   n_jobs=-1,
                   param_distributions={'colsample_bytree': [0.1, 0.3, 0.5, 1],
                                        'learning_rate': [0.001, 0.01, 0.03,
                                                          0.05, 0.1, 0.15,
                                                          0.2],
                                        'max_depth': [3, 5, 10],
                                        'n_estimators': [100, 200, 500, 900,
                                                         1500],
                                        'subsample': [0.1, 0.3, 0.5, 1]},
                   verbose=10)
```

In [ ]:
```
print (random_cfl.best_params_)
```

```
{'subsample': 0.5, 'n_estimators': 500, 'max_depth': 5, 'learning_rate': 0.2,
'colsample_bytree': 0.3}
```

In [56]:
```
x_cfl=XGBClassifier(eval_metric="mlogloss",n_estimators=500,max_depth=5,lear
x_cfl.fit(X_train,y_train,verbose=True)
```

```
/usr/local/lib/python3.7/dist-packages/xgboost/sklearn.py:1224: UserWarning: Th
e use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option use_la
bel_encoder=False when constructing XGBClassifier object; and 2) Encode your la
bels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

Out[56]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.3,
              enable_categorical=False, eval_metric='mlogloss', gamma=0,
              gpu_id=-1, importance_type=None, interaction_constraints='',
              learning_rate=0.2, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=500, n_jobs=4, nthread=-1, num_parallel_tree=1,
              objective='multi:softprob', predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=None, subsample=0.5,
              tree_method='exact', validate_parameters=1, ...)
```

In [57]:
```
predict_y = x_cfl.predict_proba(X_train)
print ("The train log loss is:",log_loss(y_train, predict_y))
predict_y = x_cfl.predict_proba(X_test)
print("The test log loss is:",log_loss(y_test, predict_y))
```

```
The train log loss is: 0.0007918391707702082
The test log loss is: 0.007935065298023179
```

In [41]:
```
dataset2=dataset1.merge(normalized_image_features,on="ID")
dataset2
```

Out[41]:

| | ID | size_asm | HEADER: | .text: | .Pav: | .idata: | .data: | .bss |
|---|---|---|---|---|---|---|---|---|
| **0** | 01azqd4InC7m9JpocGv5 | 0.400910 | 0.101695 | 0.032927 | 0.0 | 0.006937 | 0.542847 | 0.000000 |
| **1** | 01IsoiSMh5gxyDYTl4CB | 0.099719 | 0.000000 | 0.161391 | 0.0 | 0.003690 | 0.009758 | 0.000000 |
| **2** | 01jsnpXSAlgw6aPeDxrU | 0.060553 | 0.101695 | 0.101121 | 0.0 | 0.001821 | 0.000263 | 0.000000 |
| **3** | 01kcPWA9K2BOxQeS5Rju | 0.000432 | 0.107345 | 0.001092 | 0.0 | 0.000761 | 0.000023 | 0.000000 |
| **4** | 01SuzwMJEIXsK7A8dQbl | 0.006983 | 0.101695 | 0.015220 | 0.0 | 0.001234 | 0.001825 | 0.012842 |
| **...** | ... | ... | ... | ... | ... | ... | ... | .. |
| **10863** | loIP1tiwELF9YNZQjSUO | 0.080249 | 0.096045 | 0.000926 | 0.0 | 0.000653 | 0.104938 | 0.000000 |
| **10864** | LOP6HaJKXpkic5dyuVnT | 0.014462 | 0.096045 | 0.005131 | 0.0 | 0.000018 | 0.011050 | 0.000000 |
| **10865** | LOqA6FX02GWguYrI1Zbe | 0.013834 | 0.096045 | 0.004483 | 0.0 | 0.000018 | 0.010981 | 0.000000 |
| **10866** | LoWgaidpb2IUM5ACcSGO | 0.027948 | 0.096045 | 0.019997 | 0.0 | 0.000018 | 0.029290 | 0.000000 |
| **10867** | lS0IVqXeJrN6Dzi9Pap1 | 0.025171 | 0.096045 | 0.018791 | 0.0 | 0.000018 | 0.026036 | 0.000000 |

10868 rows × 3307 columns

In [43]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(dataset2.drop(["ID"],axis=1),
```

In [44]:
```
print("shape of X_train ",X_train.shape)
print("shape of y_train ",y_train.shape)
print("shape of X_test ",X_test.shape)
print("shape of y_test ",y_test.shape)
```

```
shape of X_train  (8151, 3306)
shape of y_train  (8151,)
shape of X_test  (2717, 3306)
shape of y_test  (2717,)
```

In [49]:
```
x_cfl=XGBClassifier(eval_metric='mlogloss')

prams={
    'learning_rate':[0.001,0.01,0.03,0.05,0.1,0.15,0.2],
     'n_estimators':[100,200,500,900,1500],
     'max_depth':[3,5,10],
    'colsample_bytree':[0.1,0.3,0.5,1],
    'subsample':[0.1,0.3,0.5,1]
}
random_cfl=RandomizedSearchCV(x_cfl,param_distributions=prams,verbose=10,n_j
random_cfl.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits

/usr/local/lib/python3.7/dist-packages/joblib/externals/loky/process_executor.p
y:705: UserWarning: A worker stopped while some jobs were given to the executo
r. This can be caused by a too short worker timeout or by a memory leak.
  "timeout or by a memory leak.", UserWarning
/usr/local/lib/python3.7/dist-packages/xgboost/sklearn.py:1224: UserWarning: Th
e use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option use_la
bel_encoder=False when constructing XGBClassifier object; and 2) Encode your la
bels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

Out[49]:
```
RandomizedSearchCV(estimator=XGBClassifier(base_score=None, booster=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None,
                                           enable_categorical=False,
                                           eval_metric='mlogloss', gamma=None,
                                           gpu_id=None, importance_type=None,
                                           interaction_constraints=None,
                                           learning_rate=None,
                                           max_delta_step=None, max_depth=None,
                                           min_child_weight=None, missing=nan,
                                           mo...
                                           predictor=None, random_state=None,
                                           reg_alpha=None, reg_lambda=None,
                                           scale_pos_weight=None,
                                           subsample=None, tree_method=None,
                                           validate_parameters=None,
                                           verbosity=None),
                   n_jobs=-1,
                   param_distributions={'colsample_bytree': [0.1, 0.3, 0.5, 1],
                                        'learning_rate': [0.001, 0.01, 0.03,
                                                          0.05, 0.1, 0.15,
                                                          0.2],
                                        'max_depth': [3, 5, 10],
                                        'n_estimators': [100, 200, 500, 900,
                                                         1500],
                                        'subsample': [0.1, 0.3, 0.5, 1]},
                   verbose=10)
```

In [50]:
```
1 random_cfl.best_params_
```

Out[50]:
```
{'colsample_bytree': 0.3,
 'learning_rate': 0.05,
 'max_depth': 10,
 'n_estimators': 200,
 'subsample': 1}
```

In [52]:
```
1 x_cfl=XGBClassifier(eval_metric='mlogloss',n_estimators=200,max_depth=10,lea
2 x_cfl.fit(X_train,y_train,verbose=True)
3
```

```
/usr/local/lib/python3.7/dist-packages/xgboost/sklearn.py:1224: UserWarning: Th
e use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option use_la
bel_encoder=False when constructing XGBClassifier object; and 2) Encode your la
bels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

Out[52]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.3,
              enable_categorical=False, eval_metric='mlogloss', gamma=0,
              gpu_id=-1, importance_type=None, interaction_constraints='',
              learning_rate=0.05, max_delta_step=0, max_depth=10,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=200, n_jobs=4, nthread=-1, num_parallel_tree=1,
              objective='multi:softprob', predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=None, subsample=1,
              tree_method='exact', validate_parameters=1, ...)
```

In [53]:
```
1 predict_y = x_cfl.predict_proba(X_train)
2 print ("The train log loss is:",log_loss(y_train, predict_y))
3 predict_y = x_cfl.predict_proba(X_test)
4 print("The test log loss is:",log_loss(y_test, predict_y))
```

```
The train log loss is: 0.000920365343813046
The test log loss is: 0.011034436608762126
```

In [ ]:
```
1
```