# Clustering on HSBS_not_General

## 1.Loading Data

When we look at human labels we see that %51 of the data is General. So it is unbalanced data and I have dropped 1500 general data randomly.

```
General                         2269
DELAY                            716
Customer Service Response        653
BAD REPUTATION                   229
CUSTOMER_SERVICE_ISSUES          227
Customer Query                   168
GOOD REPUTATION                  101
COVID19                           50
CHURN                             37
ESG                                6
Junk                               4
Language                           2
Name: labels, dtype: int64
```

```
General                          769
DELAY                            716
Customer Service Response        653
BAD REPUTATION                   229
CUSTOMER_SERVICE_ISSUES          227
Customer Query                   168
GOOD REPUTATION                  101
COVID19                           50
CHURN                             37
ESG                                6
Junk                               4
Language                           2
Name: labels, dtype: int64
```

## 2.Preprocessing

- First removed punctiations
- After that I replaced some words which will be removed on the next steps with **stop_words** and **isalpha** methods.

```
replace_word = {"covid":"covid", "corona":"covid","pandemi":"covid",
                "bouncebackloan":"bounce back loan", "noresponse":"no response",
                "bounceback":"bounce back", "backloan":"back loan","on hold":"on_hold",
                "bbl":"bounce back loan", "any news":"any_news","give up":"give_up",
                "gave up":"give_up", "well done":"well_done"}
for key, value in replace_word.items():
    df.tweet.replace(f"\S*{key}\S*" , f"{value}", regex=True, inplace = True)
```

- Dropped emojis
- Dropped Startswith http
- Dropped not isalpha
- Lemmatize it so turned words to their roots
- Applied bigram and trigram and I have decided these ngrams needs to have more frequency so i have multiplied that with three.

  For example : I applied week_ago week_ago week_ago and nothing happened

```
{"bounce back loan":"bounce_back_loan", "name post code":"name_post_code","full name":"full_name",
 "credit card":"credit_card","click link below":"click_link_below",
 "worst customer experience":"worst_customer_experience",
 "thank respond back":"thank_respond_back","phone service team":"phone_service_team",
 "thank write back":"thank_write_back","feeder account":"feeder_account", "still wait":"still_wait",
 "business account":"business_account","business customer":"business_customer","good morning":"good_morning",
 "week ago":"week_ago","post code":"post_code","click link":"click_link","let know":"let_know",
 "call back":"call_back","personal account":"personal_account","refer link":"refer_link","link below":"link_below",
 "name post":"name_post","hear nothing":"hear_nothing","sorry hear":"sorry_hear", "hello thank":"hello_thank",
 "loan application":"loan_application","loan apply":"loan_apply"}
```

- I have realize some tweet contains "tatacrucible" which are same tweets and there are some unenglish words in it. And they act like outliers so I have dropped the lines.

# 3.Model

There is three steps here;

**1.Vectorization:**

I have used two different BERT model and two for normalized versions totally get 4 vectorized versions:

a. 'distilbert-base-nli-mean-tokens'
b. 'distilbert-base-nli-mean-tokens' -- normalized
c. 'paraphrase-distilroberta-base-v1'
d. 'paraphrase-distilroberta-base-v1' -- normalized

**2. Dimention Reduction:**

The models returns 768 dim arrays so before clustering I need reduced dimensions. I have tried 3 of them.

a. UMAP
b. PCA
c. tSNE

**3. Clustering:**

I have tried three different clustering:

a. KMeans
b. hdbscan
c. Agglomerative Clustering

So with these 10 different techniques I have tried all combinations and finally get insight of the best results are with 'paraphrase-distilroberta-base-v1'(normalized)---UMAP---KMeans

# 4.Results:

In order to get results I have used TF-IDF vectorization.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | thank | covid | gold | sony | support | nigeria | market | help | new | money |
| 1 | bounce_back_loan | feeder_account | apply | still_wait | day | email | business_account | week | application | wait |
| 2 | credit_card | account | thank | call | hello | customer | worst_customer_experience | feeder_account | try | please |
| 3 | full_name | thank | hello | hello_thank | please | refer_link | link_below | let_know | send | name_post_code |

| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|
| amid | hongkong | china | account | branch | fintech | company | million | business | say |
| hear_nothing | account | nothing | week_ago | still | loan_apply | say | business | customer | sign |
| help | time | call_back | service | day | need | number | send | hour | team |
| sorry_hear | click_link | team | help | assist | kindly | call | detail | dm | message |

# 5.What is Next:

I believe I found good results bu searching for other models to improve it. Now I am searching SpiCy library which i believe it might be useful.