Data Science Assignment: Category Dataset

**Note** : This problem is in 2 jupyter-notebooks with part-1 and 2.

1. Part-1 mostly comprises of **data cleaning , pre-processing , NLP processing** [**notebook-part1**](https://drive.google.com/drive/u/1/folders/1q_YYcp_tggZlNKtURjnV1Z3XhDA5rt51)
2. Part-2 comprises of **dimensionality reduction , feature selection , model training , validation and evaluation**

[**notebook-part2**](https://colab.research.google.com/drive/16hLodjw0UyLWepr_pBtMaG0S8anfrAQb?authuser=1)

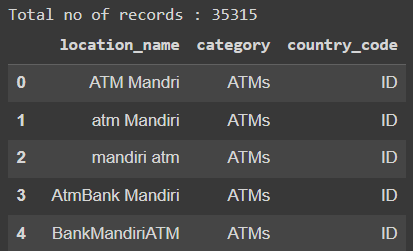
1. All of the files will be embedded into the document for the reference at each step.

**Dataset description:**

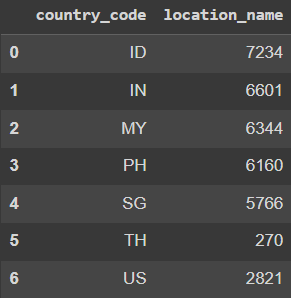
The given file is a csv file comprising of the following columns

1. Location\_name : text field consisting of the geo location names like atms names, parks , restaurants etc..
2. Category : text field describes the location category
3. Country\_id : country code such in,th,sg etc..

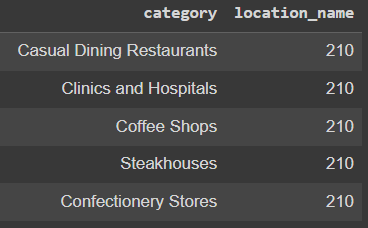
Quick snapshot of data



Country wise record count



Most common categories with their record count



# 2. Task Description

The problem statement can be perfectly framed as text classification problem. As it is evident to us that the **primary objective** is to categorize the **location\_name** into one of the **categories**. So this type of problem is usually referred to as **MULTICLASS CLASSIFICATION** problem.

# 3. Objective

1. Since the problem statement is primarily text based data and multiclass classification problem , we aim to build a multiclass classification model and evaluate by using metrics such as **accuracy** , **F1-score, precision , recall.**
2. Some common classification algorithms are **Multinomial Naive Bayes , SVM , k-NN classifier** and several other deep learning based models.

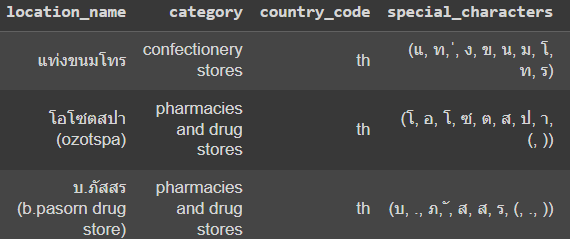
# 4. Tasks

## 4.1 Data Cleaning

The primary objective at this stage is to clean the data by applying various techniques based on the field types. Since in our case it is text data steps that I performed to clean the data are as follows

1. **Check for nulls** in the dataset . (119/35315 ~ 0.33%) location\_name were comprising nulls. So in our case **we dropped the null records** since the null records are very **minimal** and in this case imputing nulls doesn't help us much with statistical calculations
2. **Çonverting to uni-case(here I chose lower case) .** This step ensures that the same word with different cases can be uniformly cased and helps in eliminating the duplicates if needed
3. **Check for punctuation marks and symbols**  and replace with default values. Here I have removed all the symbols from the text such as $, & , ‘, etc…
4. **Check for non-english alphabet :** In our case I have identified several records with siamese characters , spanish , french , portuguese etc. So I have applied **machine translation techniques using some available libraries** and translated them to english.

Snapshot of non-english characters in the dataset



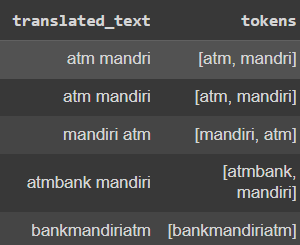
1. **Replaced numbers and eliminate words whose length < 3** from the text if any as that doesn't help us in classifying the data with respect to our problem statement.
2. **Misspellings :** Tried to use some libraries for **SpellChecker** to identify mis-spellings but due to the computation limitation and time constraints , stopped this step.

## 4.2 NLP preprocessing techniques

Since the data we have is text classification , applying some nlp preprocessing steps(such as word tokenizing , removing stop words , lemmatization , tf-idf ) would yield better results while model training

### 4.2.1 Word Tokenizing : converting the words into tokens

Doing this help us in later computing the **word frequency and** later convertingthe **word to vectors**



Quick snapshot of Word Cloud of location\_name



### 4.2.2 Stopwords removal using spacy/nltk stopwords list

Removing the stop words(such as of,in ,as , etc..) in english help us in reducing the number of features which help us train faster later.



### 4.2.3 Lemmatization

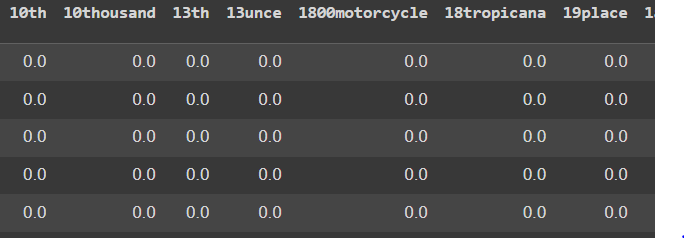
Applying lemmatization helps us reduce the number of words by reducing it to the base word for example : **running , ran , run , runs all can be reduced to single word -> ‘run’**

In our case most of the location\_names are primarily nouns like names , places etc.. applying lemmatization does not help much. And hence applying **stemming** also doesn’t help much from the intuition.

### 4.2.4 Vectorization (TF-IDF (term frequency- inverse document frequency)

This helps us in identifying the importance of the words by computing their frequency and giving the score for each word. This will be mostly sparse.

Sample snapshot after tf-idf vectorization



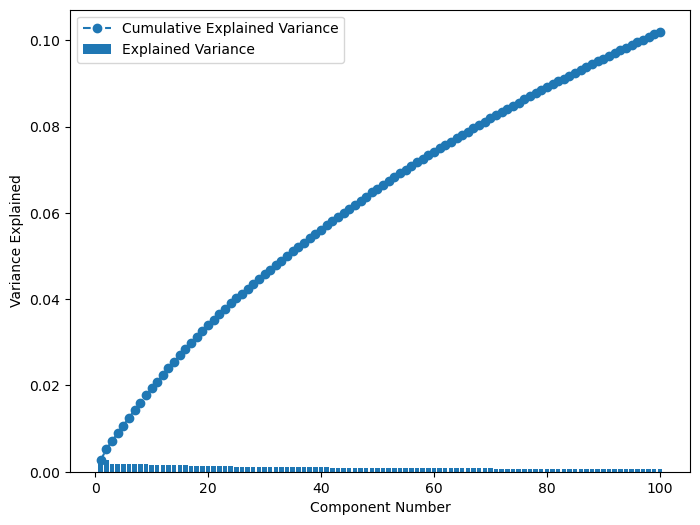
In our case the total number of features it generated is **30781.**

All 30781 features may not be useful in classifying . So we can perform feature selection and engineering techniques to look at what features.

### 4.2.4 Dimensionality Reduction

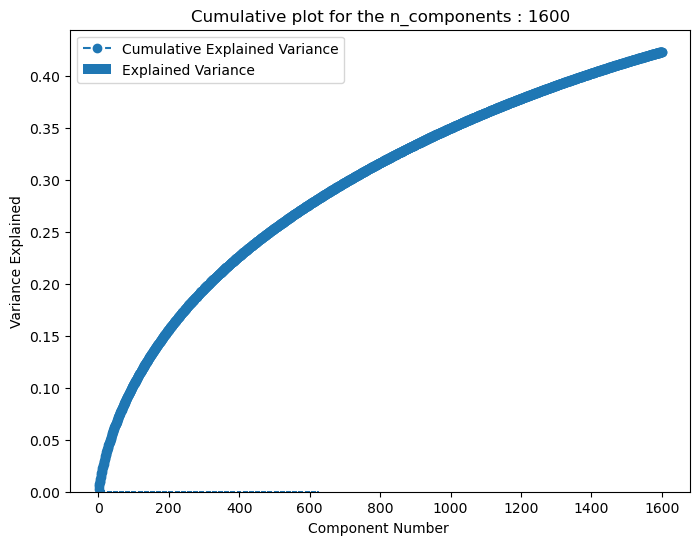
Since we have **30781 features** , applying dimensionality reduction techniques(such as PCA , Truncated SVD) helps in reducing the **number of features but preserving most of the variance**.

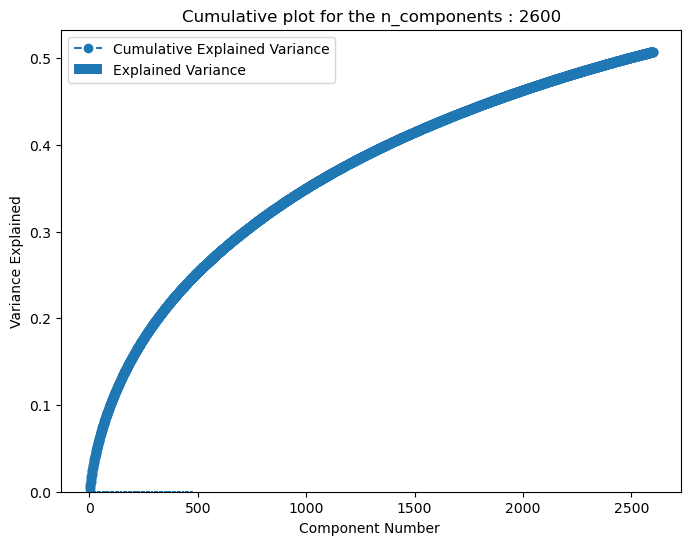
Sample plot for n\_components = 100

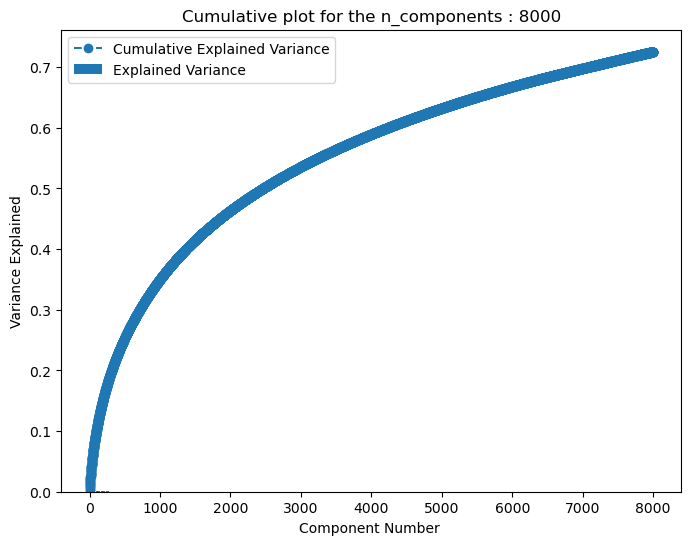


From the above plot what we can infer is if we reduce it to 100 features , it can only retrieve the 10% of variance from the data.

For 1600 dimensions ,







So choosing the **right balance between the variance and the no.of features impacts the model performance**. In our case due to the computation resource limitation and as well as time crunch , I didn't go deep further with dimensionality reduction

### 4.2.4 Feature selection

Some common techniques for feature selection incase of classification problem is **mutual\_info\_classif /chi-square technique.** It looks at the **target\_variable** and the features and applies some probability scores which adds more weightage in classifying the **target\_variable.**

Based on the scores we can derive the top features and use them in our training. Thereby reducing the number of features as well as feature importance.

**To evaluate the feature selection we can do the trial and error** with the set of features with feature selection technique and **train the model** , and by checking the performance metrics we can emphasize on the features that impacts the model performance during training.

As it is an iterative process , I didn't delve much deeper into it , since it involves a good amount of time to be invested.

### 4.2.5 Model Training , Validation and Testing

At this stage , identifying the choice of the algorithm is very important. (**Multinomial Naive Bayes , OnevsRestClassifer, RandomForestClassifier**)

For multi-class classification algorithm ,

**Multinomial Naive Bayes approach**

1. I initially considered the **Multinomial Naive Bayes** algorithm . Naive Bayes is known to work well with text datasets and can be easily extended to Multi-class classification.
2. While training the model with Naive Bayes , I faced a couple of challenges such as memory issues . I initially tried to train on the **TF-IDF** version of data.

Then I faced memory issues since it has a lot of features.

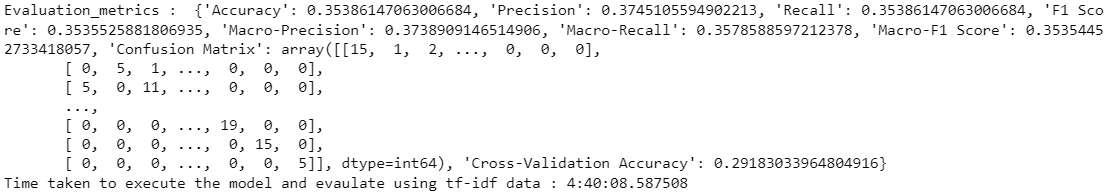
**MemoryError**: Unable to allocate 6.45 GiB for an array with shape (30781, 28123) and data type float64

So I tried to convert to **32 bit** and then retrain ,still the issue persisted . Then changed the **dataset with reduced dimension .** Then this error raised :

**ValueError**: Negative values in data passed to MultinomialNB (input X)

**RandomForestClassifier approach**

1. I considered the **Dimensionality Reduction feature dataset** for RFC with **n\_estimators**=20 . It took almost **4.5 hours to train .**
2. The results are as follows :



1. The model is classifying almost **35% accurately** , we can fine tune this by further **improving the dimensionality reduction** to increase the data variance from 70% to almost 85-90% but that can increase the features and computation time.
2. And also we can perform **feature selection techniques**
3. Also play around with **no of estimators** parameter in RFC algorithm to improve the model prediction performance.
4. Also we can increase the precision from **32-bit** to **64-bit** but that can increase the memory and thereby affects the computation time and power.

**OneVsRestClassifer**

Executed the code but the execution time is taking long and is going beyond the deadline and hence terminated the execution.

## Challenges Faced

1. Dealing with **misspelled** words with some of the libraries such as **SpellChecker**  is an extremely time consuming process. Hence halted the stop due to time constraints.

**Note:** Handling the data cleaning steps with **multi-processing** techniques can process it much faster.

1. **Dimensionality reduction and feature selection** steps took a very long time..
2. Since the data is text data , once converting to word vectors the data becomes huge with almost 30K+ features. Storing these features in normal csv files almost consumes the GBs of data .So chosen **parquet** format and also tried to reduce the precision to **32-bit** instead of 64-bit. And alternative technique I applied is to generate the files at every necessary step and thereby reducing the memory usage for previous data frame variables
3. Due to this **RAM** is **out of space** and making it difficult to train the model

Sample errors associated with it ,

**MemoryError**: Unable to allocate 6.45 GiB for an array with shape (30781, 28123) and data type float64

1. When choosing the dimensional data for training the data with Multinomial NB , it has given the following error

**ValueError**: Negative values in data passed to MultinomialNB (input X)

# 5. Summary

The conclusion is we can experiment with multiple techniques to improve the model performance.

1. By fine tuning the **dimensionality reduction** to improve the variance of the data
2. By applying the **feature selection strategy effectively** for better selection of features by using techniques like mutual information or chi-square
3. By increasing the precision from **32-bit to 64-bit** values but at a cost of additional memory and computational power
4. **Fine-tuning the hyperparameters** such as n\_estimators in **RandomForestClassifier** and alpha parameters in **OneVsRestClassifier** and several other parameters.
5. Also by fixing the **misspelled words** in the dataset
6. We can identify the classes which are predicted poorly and give more **additional weightage** **to misclassified classes** while training.

These are some of the techniques to improve that are commonly practiced and we can incorporate several other techniques.