**FINAL REPORT – AUTOMATIC TICKET ASSIGNMENT**

**Team: Krupa, Rennick, Ritisha, Unnimaya, Yash (Batch 4)**

**Mentor and Supervised by Shelton**

**PM – Navya**

**Date :12 July 2020**

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **No.** | **Contents** | **Page No.** |
|  | Problem | 1 |
|  | Business Domain Value | 1 |
|  | Abstract | 1 |
|  | Solution (Step by Step) | 1 |
|  | Model evaluation | 23 |
|  | Visualization | 35 |
|  | Implications | 37 |
|  | Limitations | 37 |
|  | Closing Reflections | 37 |

1. Problem

Manual assignment of incidents is time consuming and requires human efforts. There may be mistakes due to human errors and resource consumption is carried out ineffectively because of the misaddressing. On the other hand, manual assignment increases the response and resolution times which result in user satisfaction deterioration / poor customer service.

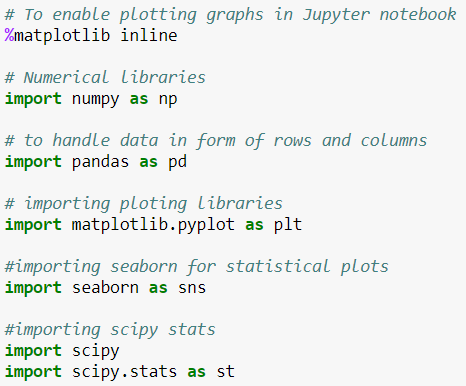
1. Business Value Proposition

In the support process, incoming incidents are analysed and assessed by organization’s support teams to fulfil the request. In many organizations, better allocation and effective usage of the valuable support resources will directly result in substantial cost savings.

1. Abstract

An attempt at Leveraging Machine Learning and Artificial intelligence to automatically classify tickets and assign them to the right owner in a timely manner to save effort, increase user satisfaction and improve throughput in the ticketing pipeline of an organization

1. **Solution Step by Step**

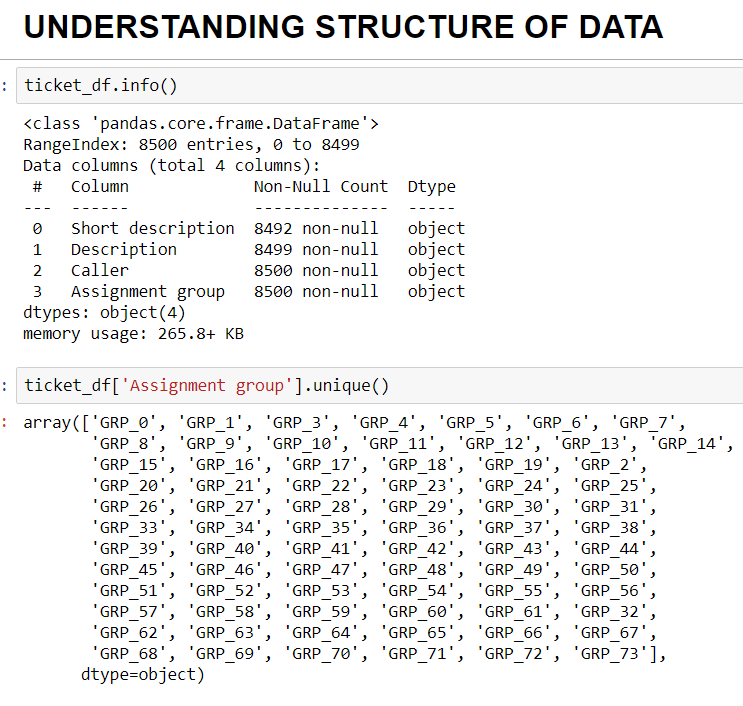
We need to import and install few of the pre-requisites as below   


Import the DATA file from your location



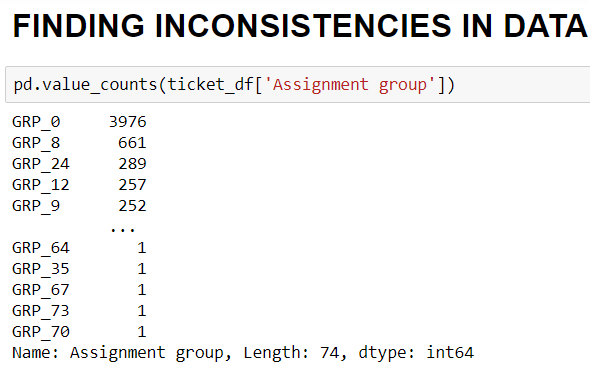
**Install FASTTEXT and WORDCLOUD**   
fast Text is a library for efficient learning of word representations and sentence classification. it performs better than Word2Vec and allows rare words to be represented appropriately

Word cloud is used to analyse the text data through visualization in the form of tags, or words, where the importance of a word is explained by its frequency  
pip install fasttext  
pip install wordcloud.



**Inference**

* Structure of data: We have 8500 rows and 4 columns
  + Short Description
  + Description
  + Caller
  + Assignment Group

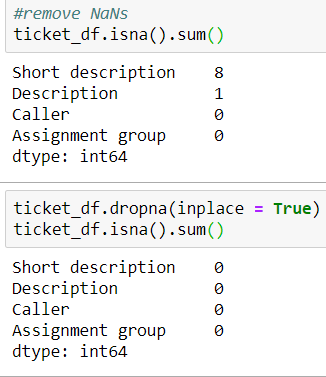


**Findings**

**Dataset has 8500 Documents(records) and 4 features(columns)**

**Inference**

* 74 unique labels
* Null values: 8 Short description records, 1 Description record
* Dataset is highly imbalanced and skewed, since there is huge variance in the assignment, as in mostly all of them are assigned to GRP\_0. Many groups have only 1 datapoint
* Dataset is highly imbalanced and skewed! Around 46% of the dataset is represented by just one class GRP\_0. There are many classes that hardly have 1 datapoint. We would need to up sample the underrepresented class and down sample the strongly represented classes.
* We have 74 categories. There are few categories with very few tickets. So, it is better to take into consideration where at least 20 tickets are assigned or else it will lead to higher noise levels.

**Checking missing values and dealing with it.   
Dropping missing values.**  


## **PIE CHART FOR FREQUENCY OF TICKETS ASSIGNED TO GROUPS**

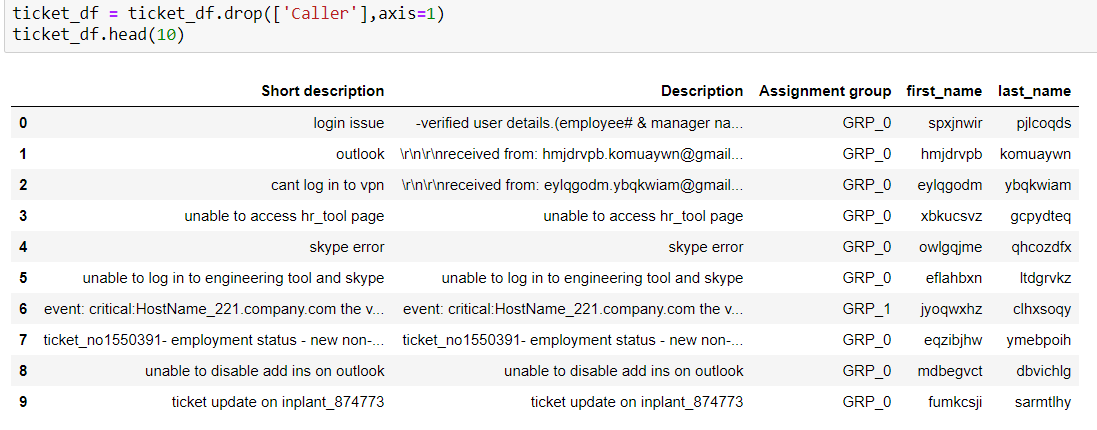
**WORD CLOUD FOR ALL THE COLUMNS TO KNOW THE DISTRIBUTION:  
a. Short Description**

**b. Description**

**c. Caller**

**TEXT PRE-PROCESSING**As caller names create noise, we need to remove them by splitting the names into first name and last name further adding into stop words later.  
Remove special characters, stop words, carriage returns, email id

**Caller Split:**

**Dropping column Caller: (As we no longer need that)**

**DATA CLEANING:**

The main aim of Data Cleaning is to identify and remove errors & duplicate data, in order to create a reliable dataset. This improves the quality of the training data for analytics and enables accurate decision-making.

**Cleaning data**

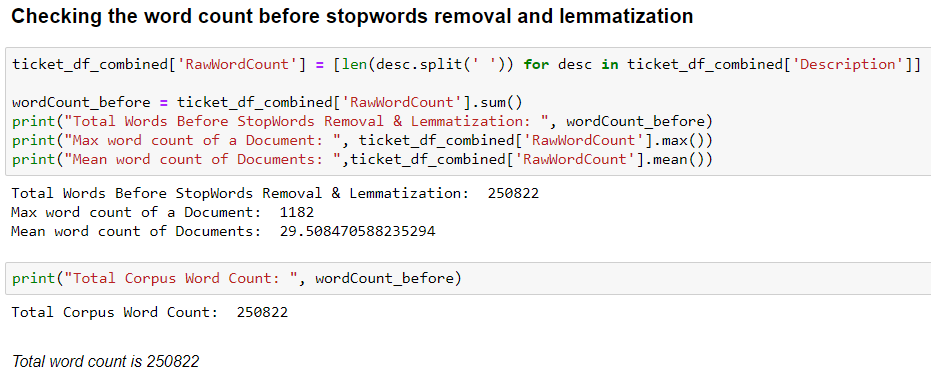
Remove trailing spaces and unwanted characters like carriage return, digits and extra spaces.

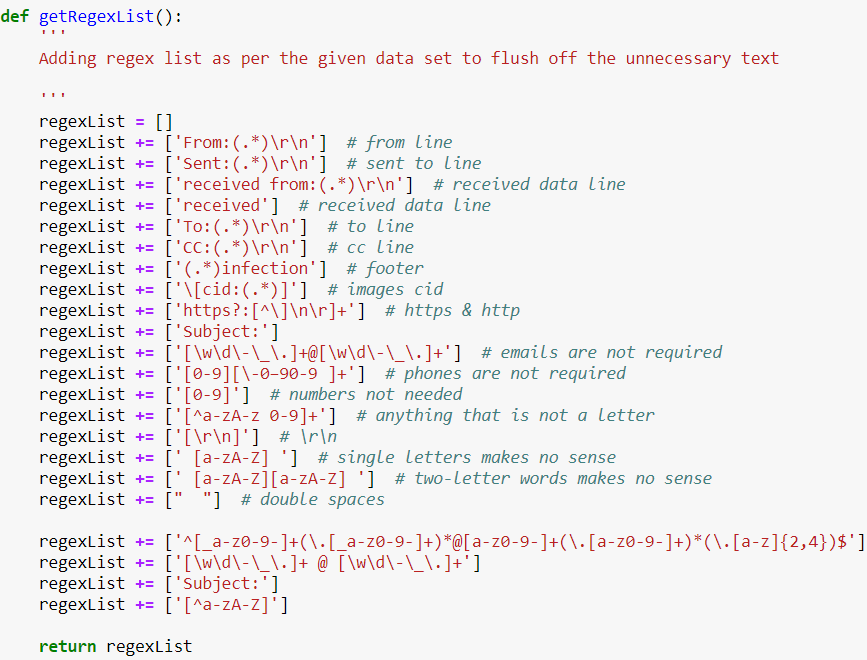
Converting the whole data frame to lower case.



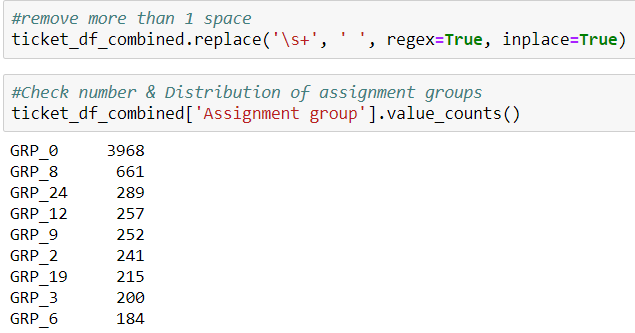
Observation: We have a few short descriptions common with description. Merging them as a set makes sense.  

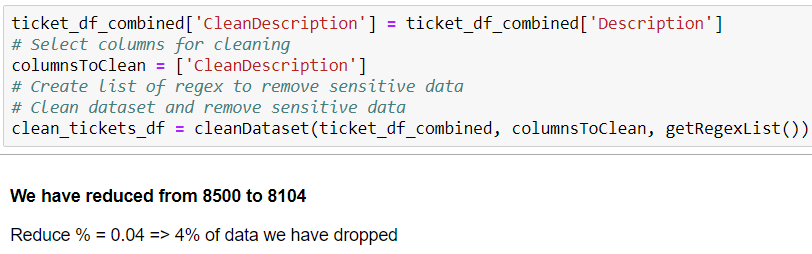

Total word count in the corpus:



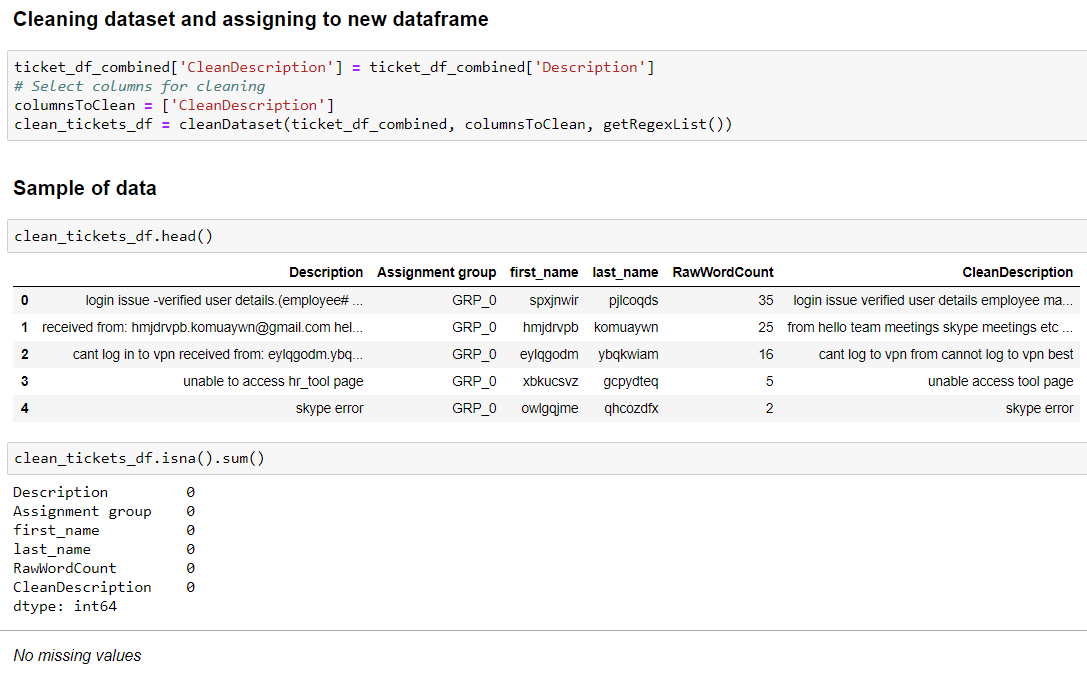
Now we use REGEX list to remove irrelevant data or unnecessary text based on the given data set.

For example, here we remove the extra spaces, any letter which doesn’t exist in dictionary, any contact details etc.  
Assigning Groups:

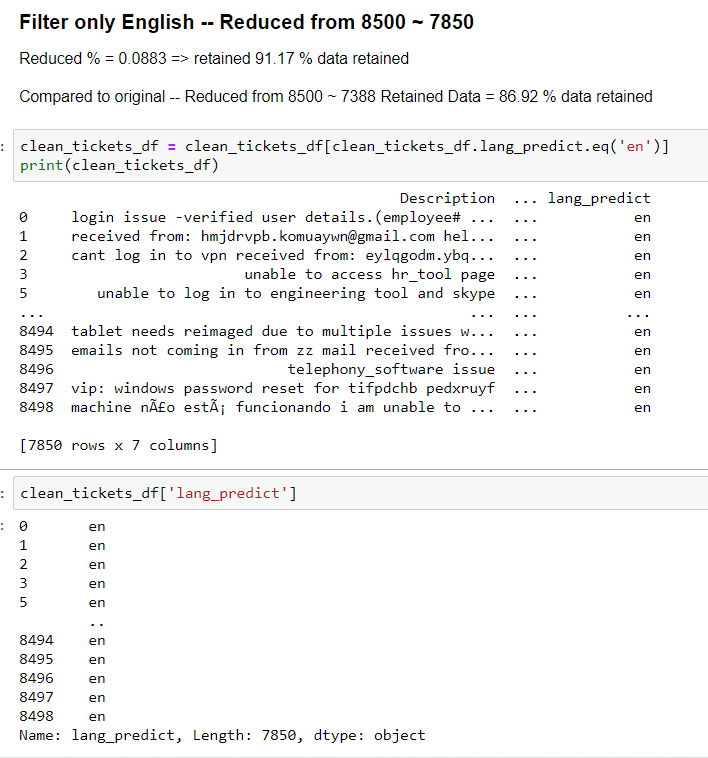




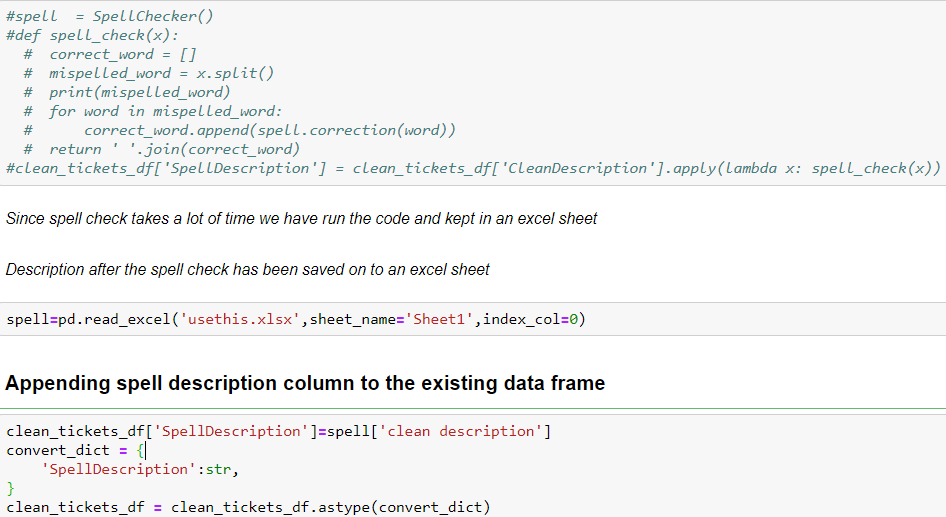
**NEW DATA FRAME**

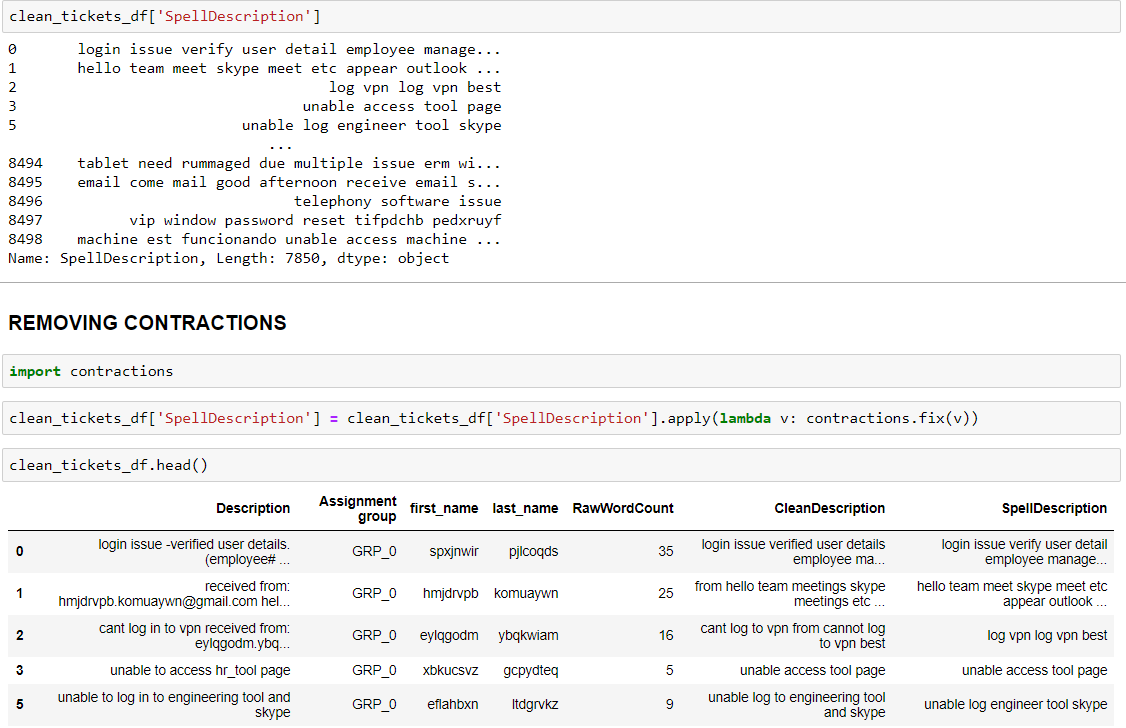


**LANGUAGE DETECTION.**After importing Fast text, we have reduced the data set  
We can see many languages apart from English, so for ease let’s keep the English alone

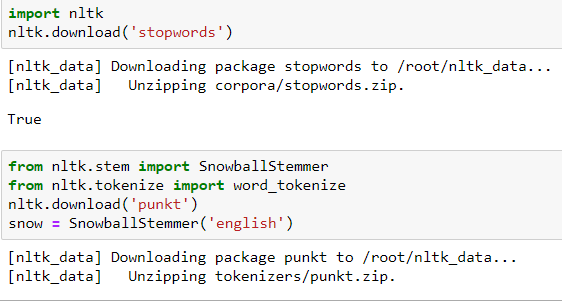


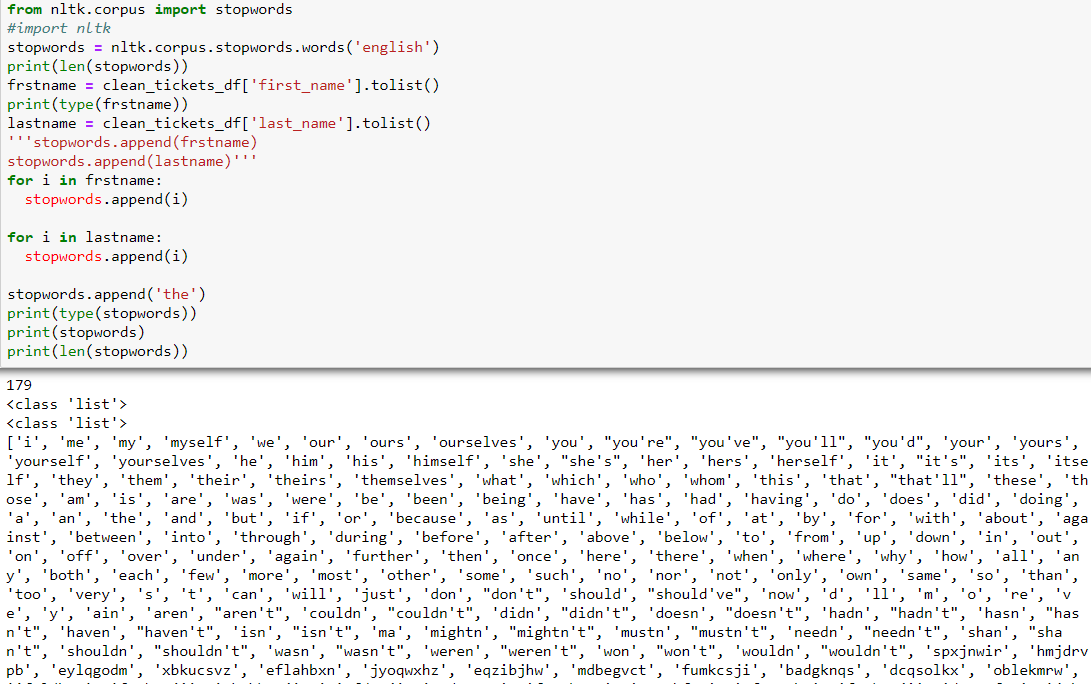
**SPELL CHECK:** There are a lot of misspelled words so we need to treat that



Spell Description after Spell check and removing contractions

**Stopwords Removal**Stop words occur in abundance in learning models, hence providing little to no unique information that can be used, so we filtered it out before processing of the natural language data.



Adding first\_name and last\_name to stopwords  


**STEMMIG**

**Stemming** is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. **Stemming** is important in natural language understanding (NLU) and **natural language processing** (**NLP**)



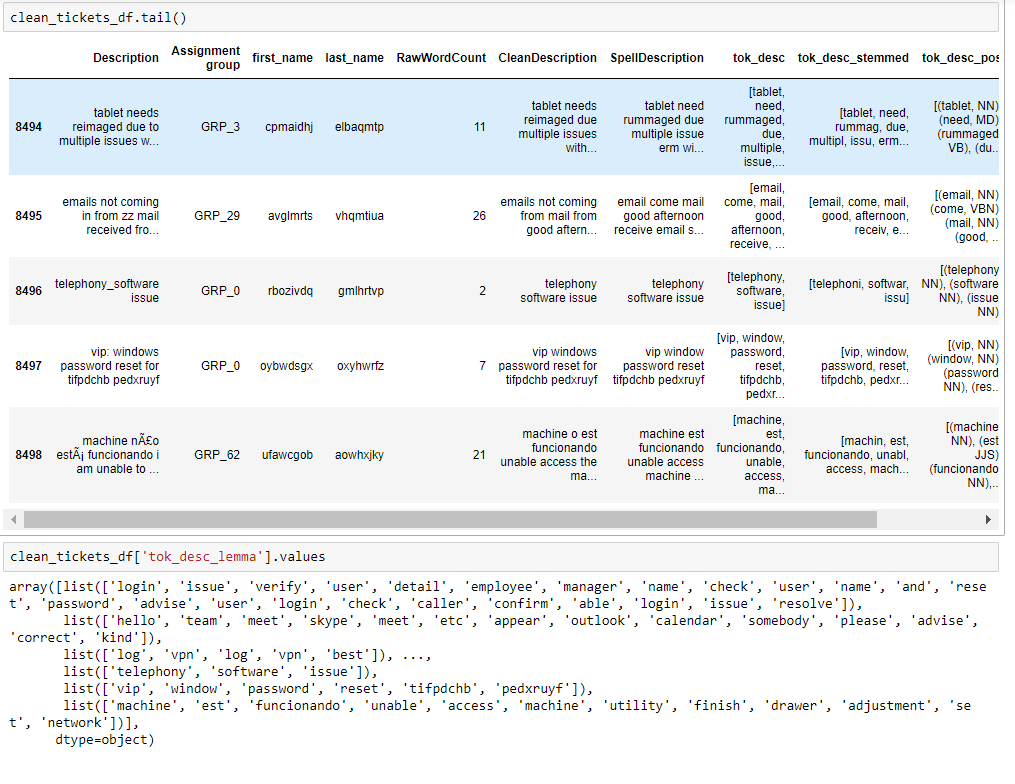
**POS TAGGING:**

Identifying Part of Speech **Tags** using Conditional Random Fields. ... **POS Tagging** is also essential for building lemmatizers which are used to reduce a word to its root form. **POS tagging** is the process of marking up a word in a corpus to a corresponding part of a speech **tag**, based on its context and definition.  


**LEMMATIZATION**

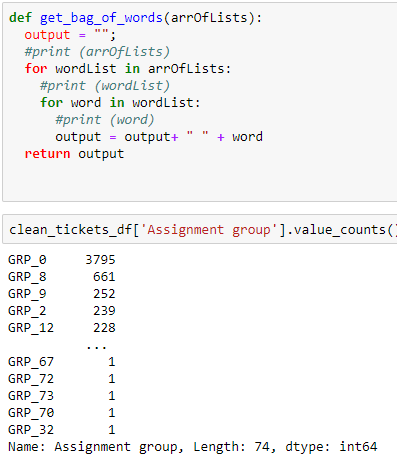
In Lemmatization root word is called Lemma. ... Because lemmatization returns an actual word of the language, it is used where it is necessary to get valid words. Python NLTK provides WordNet Lemmatizer that uses the WordNet Database to lookup lemmas of words.

Output after lemmatization is below

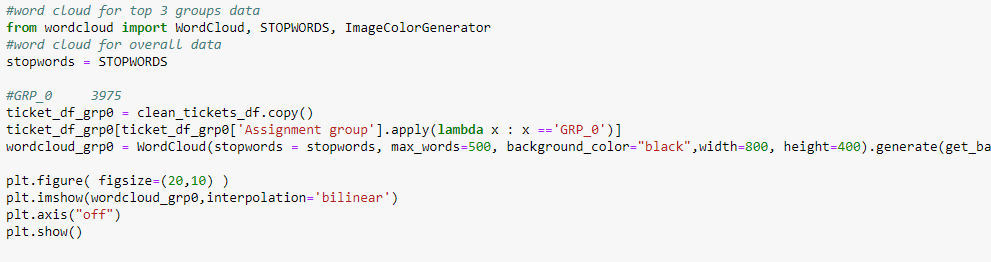


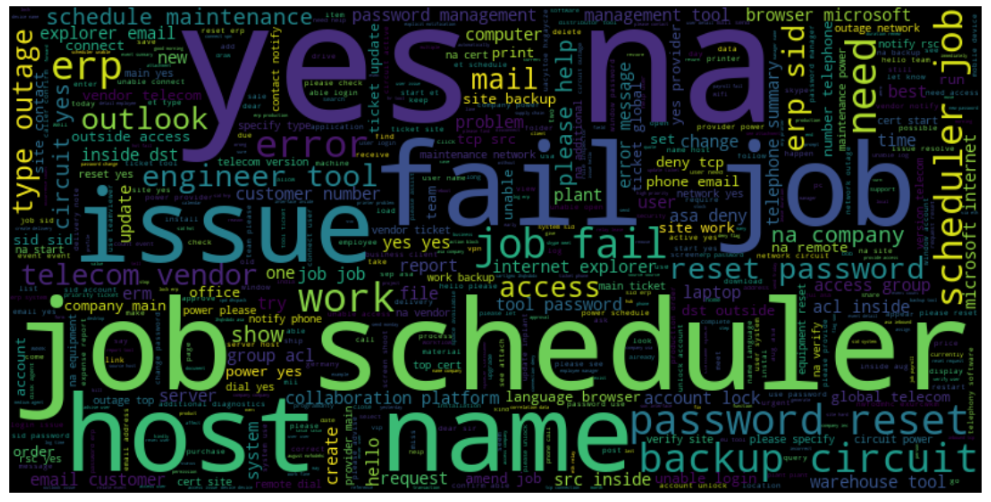
**Create BAG of WORDS.**

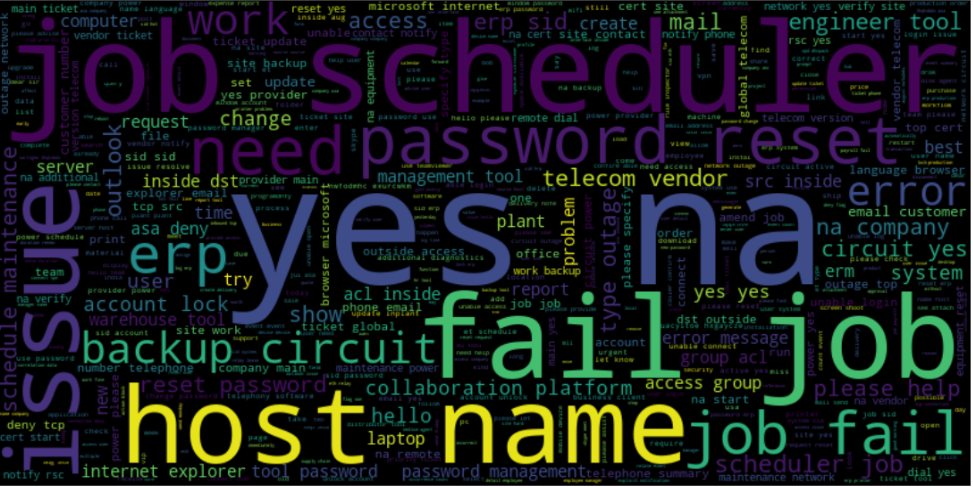
Bag of Words (BOW) is a method to extract features from text documents. These features can be used for training machine learning algorithms. It creates a vocabulary of all the unique words occurring in all the documents in the training set.



**WORD CLOUD FOR MOST COMMON THREE GROUPS:**





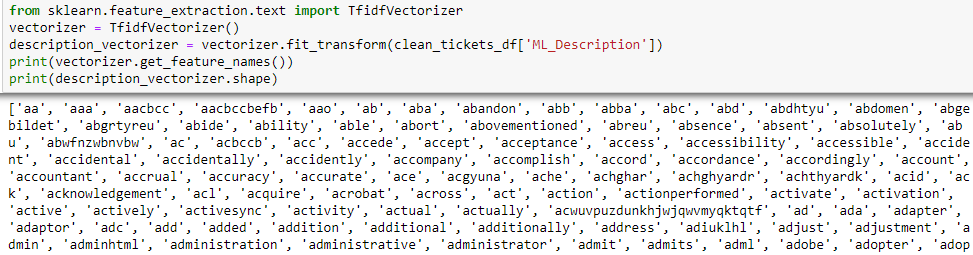


A close up of a newspaper

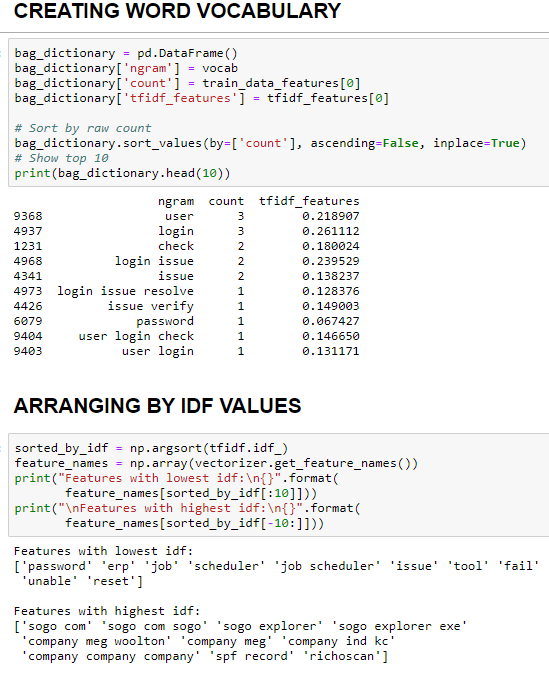
Description automatically generated

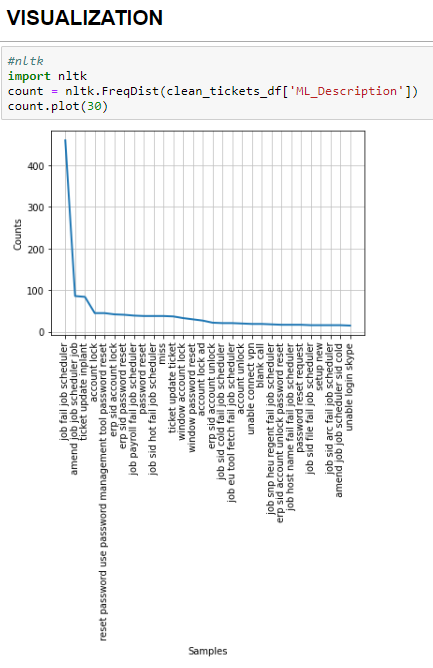
**TF\_IDF:**

TF-IDF or ( Term Frequency(TF) — Inverse Dense Frequency(IDF) )is a technique which is used to find meaning of sentences consisting of words and cancels out the in capabilities of Bag of Words technique which is good for text classification or for helping a machine read words in numbers.

Here we import the Count Vectorizer which provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

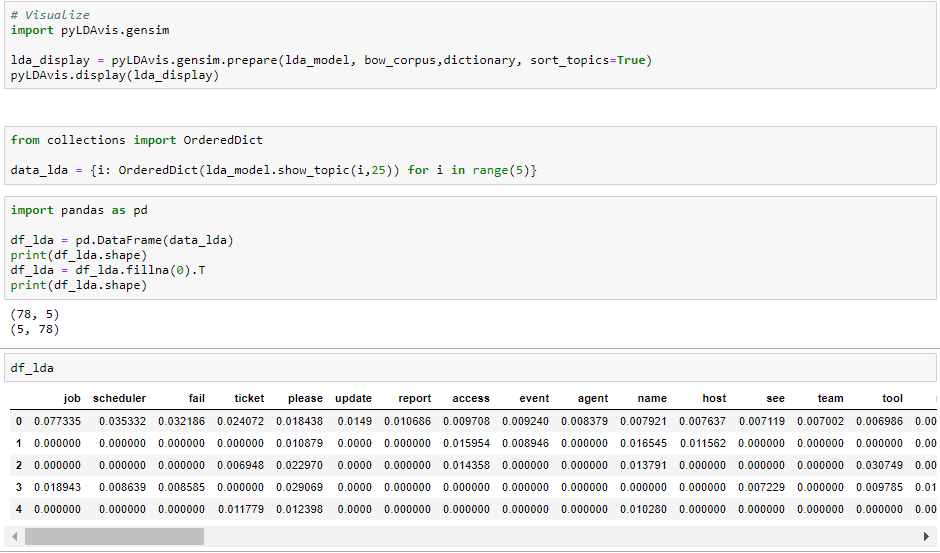






**LDA**

LDA’s approach to topic modelling is it considers each document as a collection of topics in a certain proportion. And each topic as a collection of keywords, again, in a certain proportion.

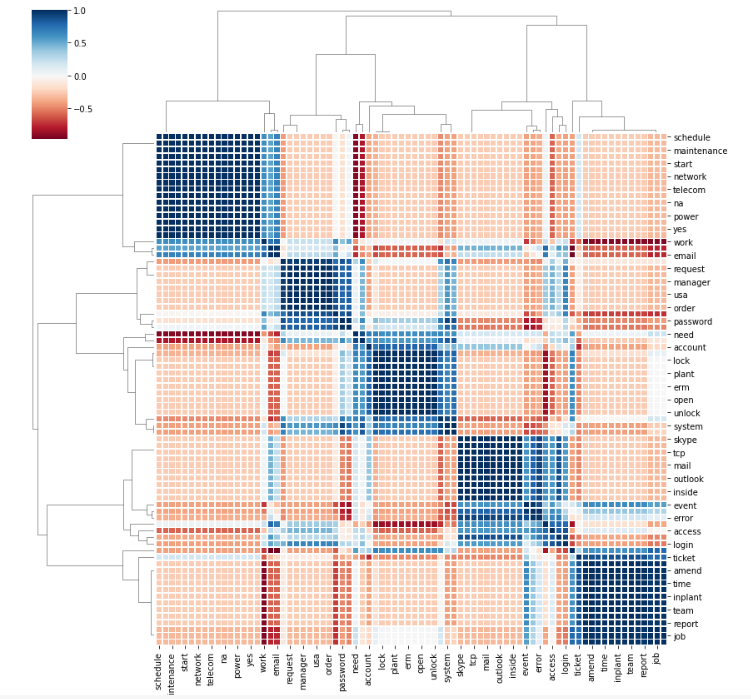


**Cluster MAP:**

Cluster Map helps in finding the pairwise correlation of all columns in the data frame. Any na values are automatically excluded.

Cluster Map clusters both columns and rows and adds dendrograms to show the clustering of similar keywords in this case.

Hierarchical clusters help to order data by similarity and displays similar content next to one another for even more depth of understanding the data.



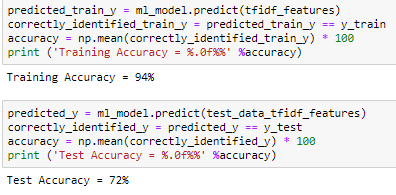
**REMOVING UNNECESSARY GROUPS**

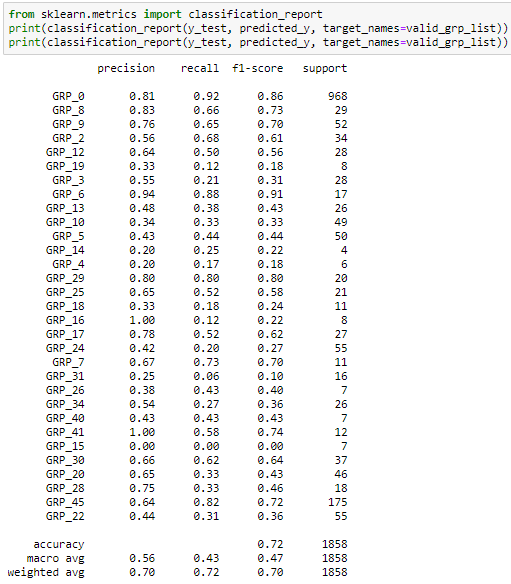
Keeping minimum group frequency as 31 (95th percentile)



1. **Modelling for benchmark**

**Model Using Logistic Regression**



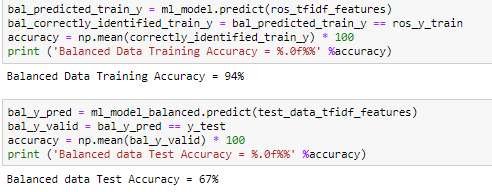


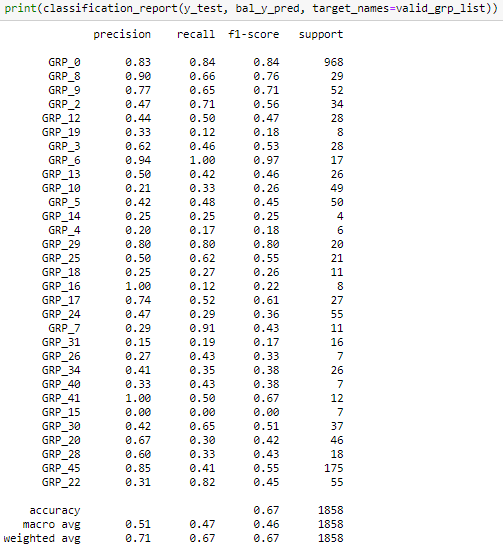
Logistic Regression Summary

Train Accuracy: 94%

Test Accuracy: 72%

**Model Using Logistic Regression with Balance**



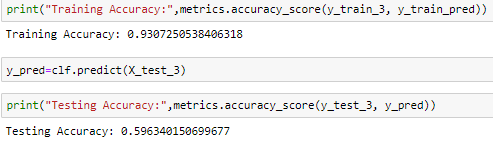


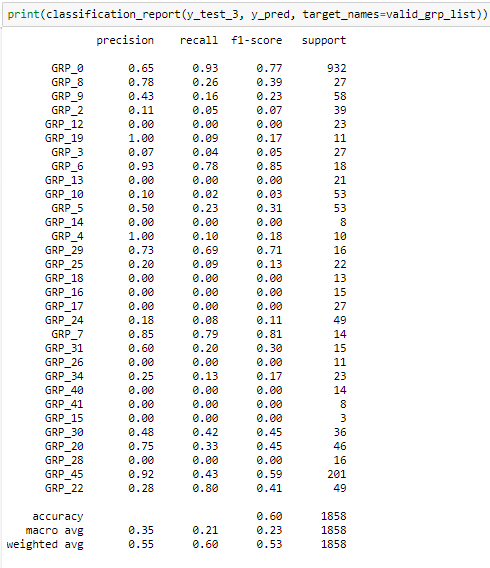
Logistic Regression with Balance Summary

Train Accuracy: 94%

Test Accuracy: 67%

**Model Using Random Forest**



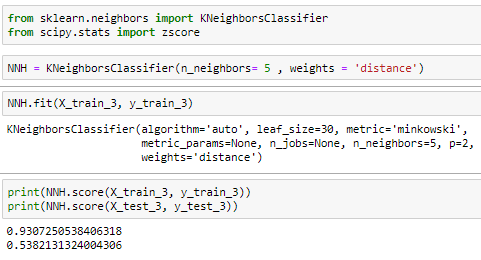
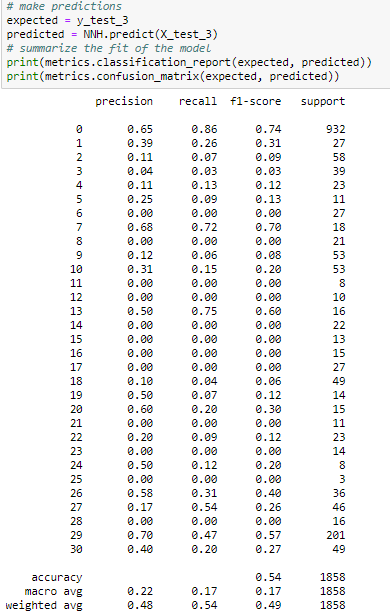


Random Forest Summary

Train Accuracy: 93.80%

Test Accuracy: 59.63%

**Model using KNN**

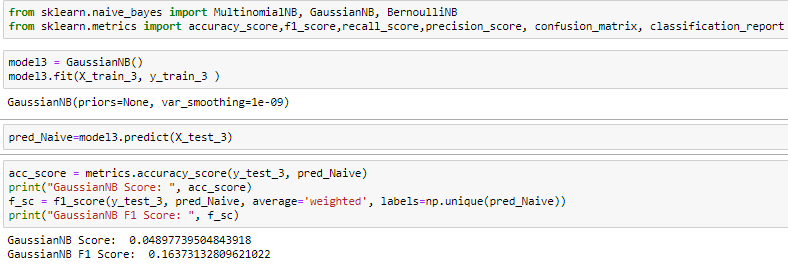
  


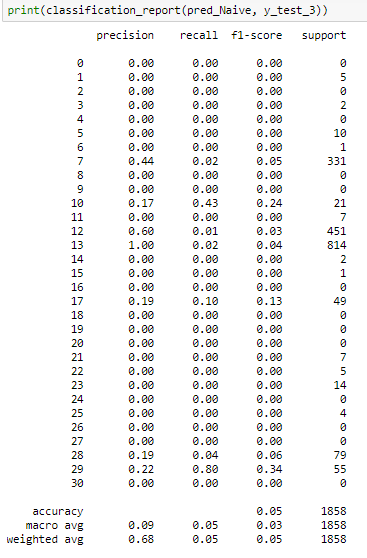
KNN Summary

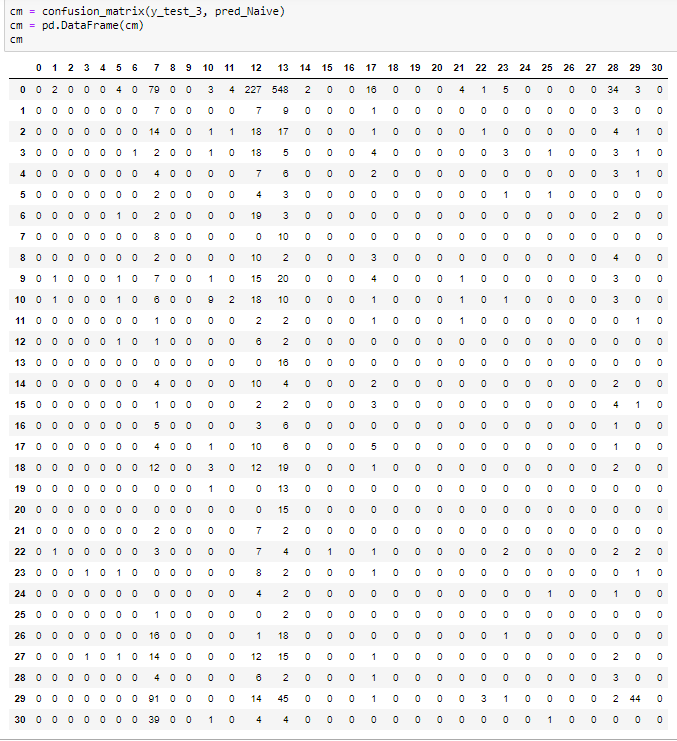
Train Accuracy: 93.80%

Test Accuracy: 53.63%

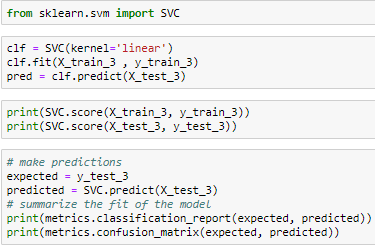
### **Gaussian Naive Bayes Model**







**Support Vector Machine (Support Vector Classifier).**

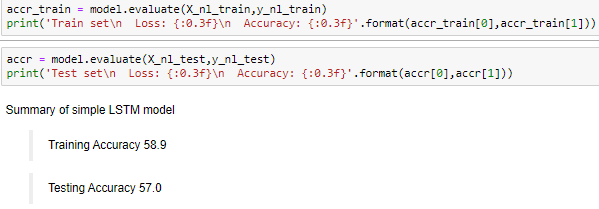


SVC algorithm

Training Accuracy: 95.30%

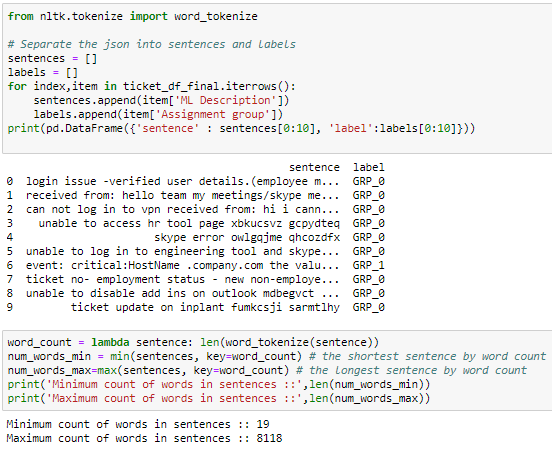
Testing Accuracy: 60.36%

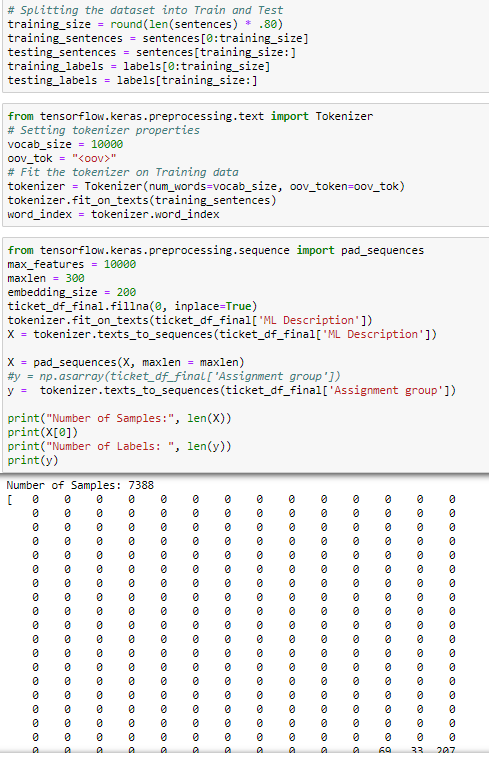
**Model using LSTM**

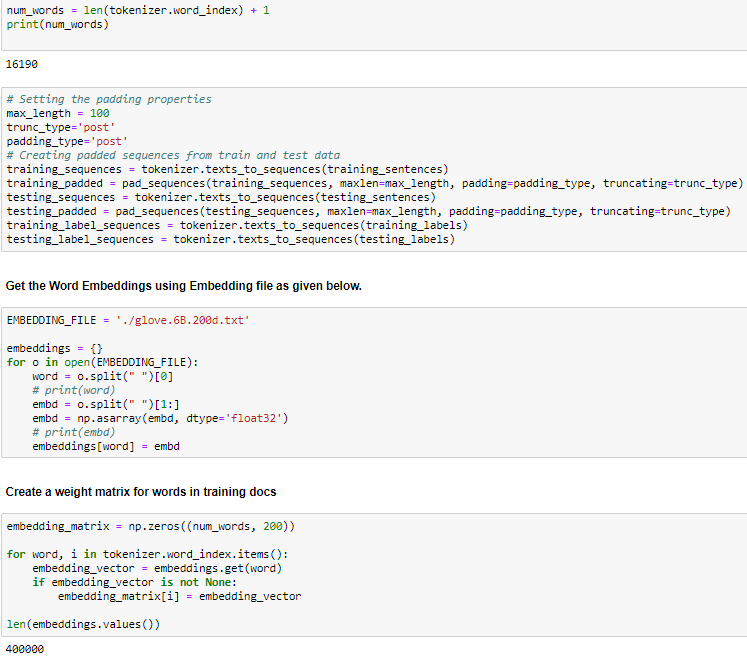


**BI-LSTM :   
Embeddings**

Get the Length of each line and find the maximum length. As different lines are of different length. We need to pad our sequences using the max length.

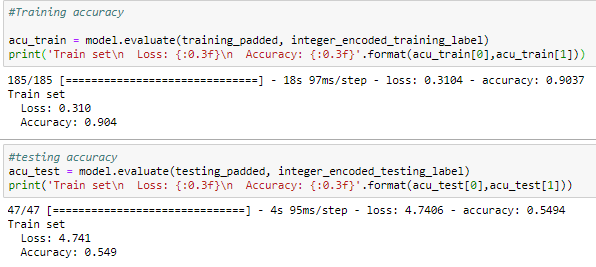


**Apply Keras Tokenizer of Description column of your data.**  


**Get vocabulary size**

**Create and compile your model**





**CONCLUSION**

We tried various ML models like Logistic Regression, LR with Balance, Random forest, KNN and SVM

LR

Training Accuracy: 94%

Testing Accuracy: 74%

LR with Balance

Training Accuracy: 94%

Testing Accuracy: 70%

Random Forest

Training Accuracy: 93.80%

Testing Accuracy: 59.88%

KNN

Training Accuracy: 95.30%

Testing Accuracy: 60.36%

SVM

Training Accuracy: 68%

Testing Accuracy: 64%

Deep Learning Models

On trying Deep Learning Models, following were the results

LSTM

Training Accuracy: 58.9%

Testing Accuracy: 57%

Bi-LSTM

Training Accuracy: 94%

Testing Accuracy: 67%

**INFERENCE**

We were able to see that in the above used models, most of the models were overfitting. SVM was better out of the lot, since there was less variance with the training and testing accuracy. Though deep learning models were used there was much variance in the training and testing accuracy, as they were overfitting

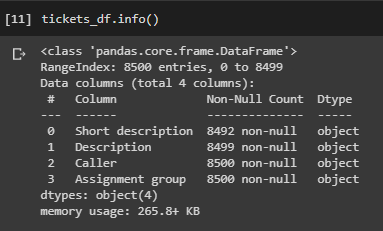
1. **Visualizations:  
   Exploratory Data Analysis:**

The data consists of 8500 rows and 4 columns:

1. Short description column - This column defines the issue in short terms using only the important words.
2. Description column - This column defines the entire description of the issue.
3. Caller column - This column is unnecessary. Hence, it is dropped.
4. Assignment group column - This is our target column which tells us the group of the corresponding issue.



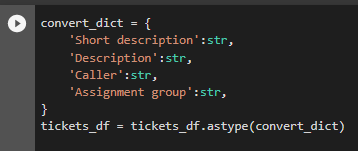
The concise summary of the data provided looks as below



**Pre-Processing, Data Visualisation**

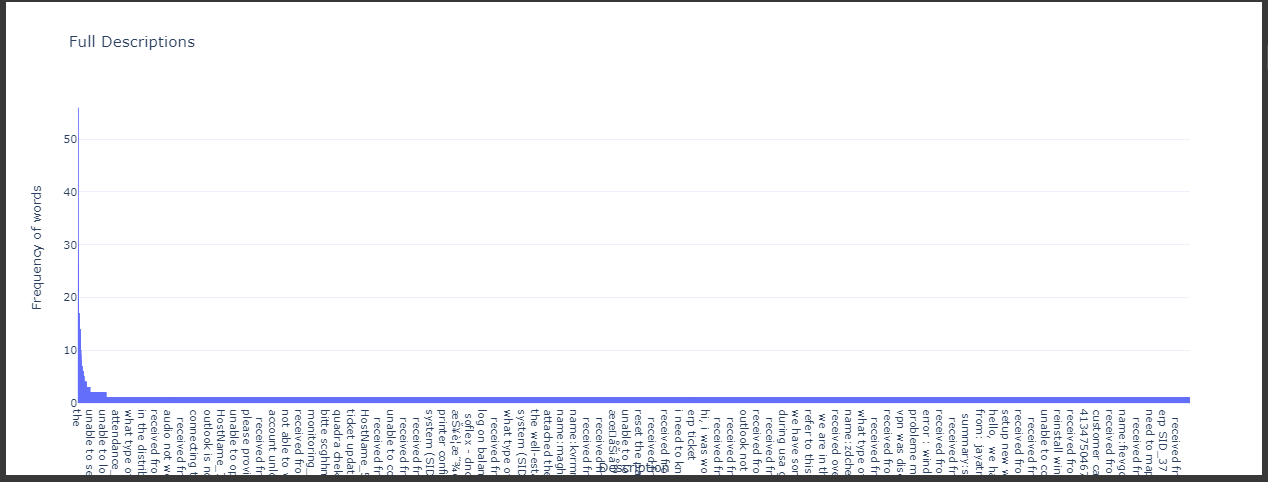
Pre-processing is an essential, wherein the text is to be cleaned up to bring it to a required format for the information extraction models. This includes normalizing different tenses of words, normalizing synonyms, spell correction etc.

For easy processing of data, we are converting all the columns to String datatype.



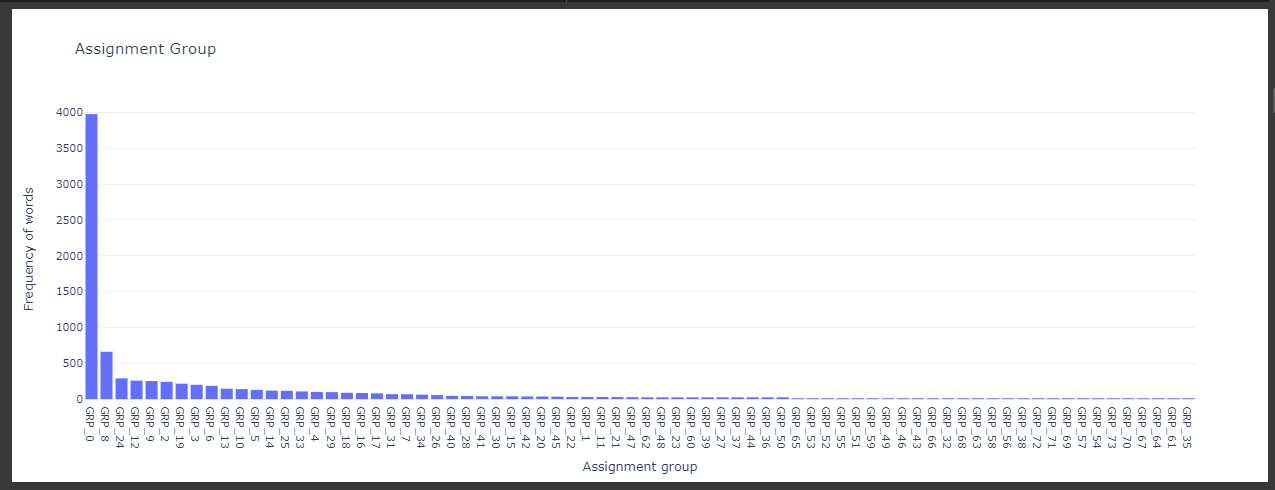
**Visualisation:**

Description:



Short Description:



Assignment Group:

1. **Implications**

**Business Domain Value**

In the support process, incoming incidents are analysed and assessed by organization’s support teams to fulfil the request. In many organizations, better allocation and effective usage of the valuable support resources will directly result in substantial cost savings. Currently the incidents are created by various stakeholders (Business Users, IT Users and Monitoring Tools) within IT Service Management Tool and are assigned to Service Desk teams (L1 / L2 teams).

This team will review the incidents for right ticket categorization, priorities and then carry out initial diagnosis to see if they can resolve.

1. **Limitations:**

Our LSTM model lacks behind in terms of testing accuracy which is basically over fitting.

Our basic ML models tend to misclassify many groups as the GRP\_0.

**How can we improve?**

For over fitting:

We can try to include more dropout layers and tune the number of epochs and the batch size accordingly.

For misclassification:

There are issues which are very similar to each other but fall into different categories.

We can group them into the same category and check for the results.

1. **Closing Reflections**

We did learn important things while working on this Capstone project.

**Teamwork:**

Throughout the process we learnt to work as a team and to contribute equally into this Capstone project.

Coming together as group we learnt how a same task can be done in different ways.

Throughout the project we understood that it is important to split up tasks from the very beginning to avoid conflicts later.

**What would you do differently?**

We would try not to procrastinate things and be up to date with our work.

We would assign ourselves with tasks and do it accordingly.

We would try to brainstorm all our ideas together on a platform and have more group discussions.