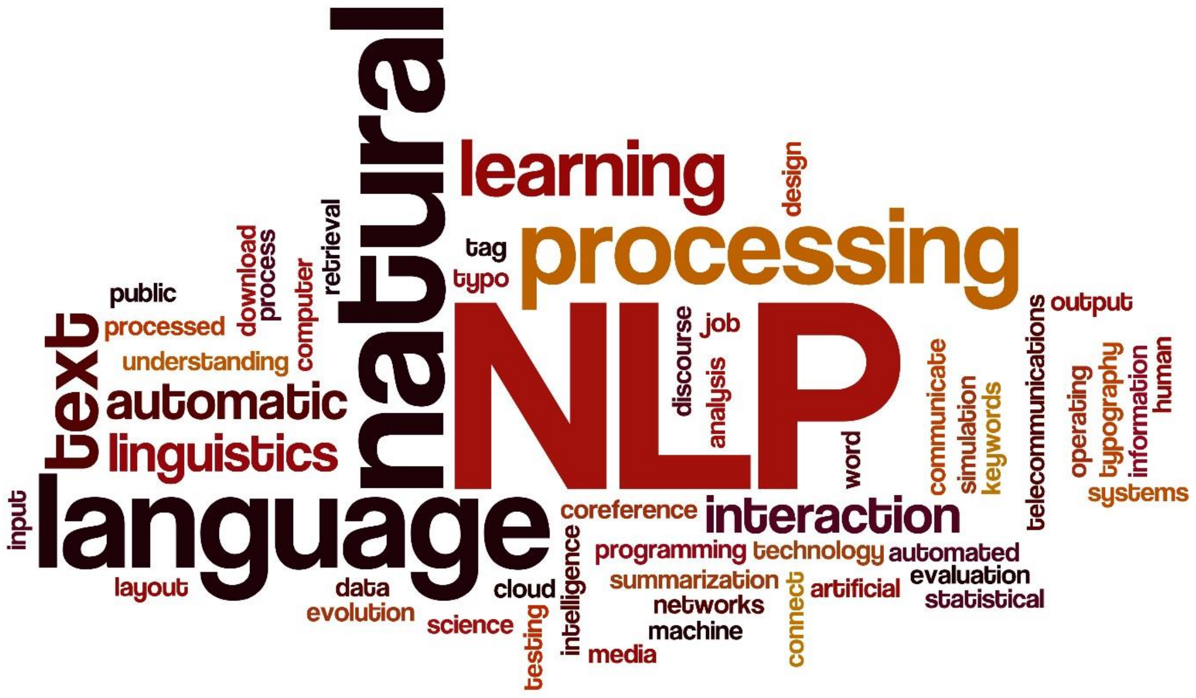
greatlearning
Learning for Life

Capstone Project
Natural Language Processing (NLP) Application
Automated Ticket Assignment



greatlearning
Learning for Life

- Program: Post Graduate Program (PGP) in Artificial Intelligence and Machine Learning(AIML)
- Batch: April 2019 – April 2020
- Group: 3
- Deliverable - Milestone - 1
- Team members:
    - Gaurav Walia
    - Karishma Dcosta
    - Lavanya Harry Pandian
    - Pallavi Kumari
    - Swati Tyagi
- Project Guide: Sanjay Tiwari
- Program Manager: Anurag Shah
- Date of Submission: 29-March-2020

Project Goal

One of the key activities of any IT function is to ensure there is no impact to the Business operations through Incident Management process. An incident is an unplanned interruption to an IT service or reduction in the quality of an IT service that affects the Users and the Business. The main goal of the Incident Management process is to provide a quick fix / workarounds or solutions that resolves the interruption and restores the service to its full capacity to ensure no business impact.
These incidents are recorded as tickets that 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).

The *goal* of this project is to build a classifier that can classify the tickets by analyzing the text using Natural Language Processing(NLP) techniques in AIML.

Milestone 1 Detailed Report:

1. Summary of problem statement, data and findings

In this section, we describe the problem statement: explaining the current situation, opportunities for improvement and data findings: data requirement, size and source of data with its challenges and techniques to overcome the same.

Problem Statement

Current Situation –
Given the data that is collected from the IT Service Management Tool, the issues are recorded as tickets and are assigned to respective groups based on the type of issues that need to be addressed. Assigning the incidents to the appropriate group has critical importance to provide improved user satisfaction while ensuring better allocation of support resources, thus maintaining the organization's efficiency in the service.
However, the assignment of incidents to appropriate IT groups is still a manual process in many of the IT organizations.

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.

Opportunity for improvement –
This manual process can be improvised using machine learning based systems such as automatic ticket classification mechanisms that would:

- Reduce/remove the count for human error
- Use the allocated resources efficiently
- Ensure efficient ticket classification
- Provide quick solutions and turn-around times for the organization as a whole.

By leveraging the AI technology, we shall *build a classifier that can classify the tickets into respective Groups by analyzing the text using NLP techniques in AIML.*

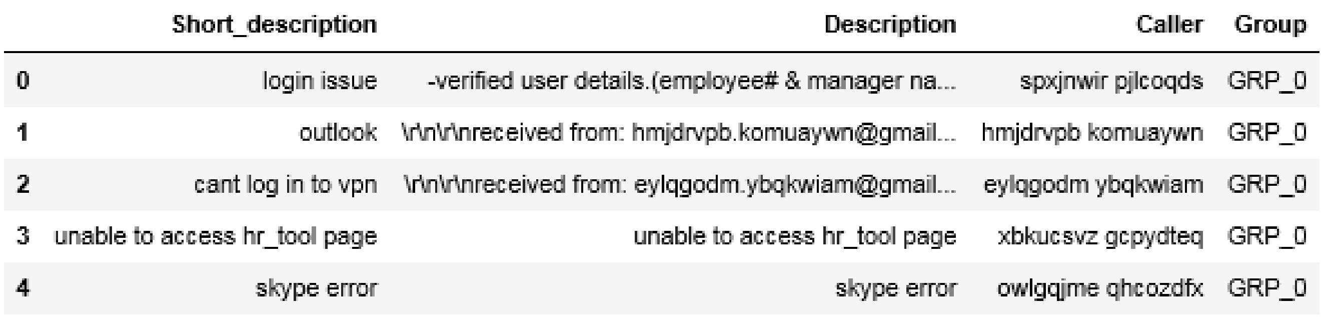
Data Findings

In Natural Language Processing (NLP), most of the data in the form of documents and text contain many words that are redundant for text classification, such as stop-words, misspellings, slangs; and also contain various languages since the users could potentially be located globally.

Data Requirement –
In order to understand the tickets, we require the past ticket information comprising of the ticket summary/ title which captures the essence of the issue, the detailed description for additional details, the user information and the group assigned to the respective tickets. Additional information such as separate fields for timestamp, geographic location of user, etc.,would be useful in understanding the traffic and geo of the tickets logged to assign resources as per the demand of the tickets.

The Dataset used for the project can be referred from the following location in an excel format(*.xlsx):
https://drive.google.com/drive/u/0/folders/1xOCdNI2R5hiodskIJbj-QySMQs6ccehL

Source of data and challenges –
The data that has been captured by the IT Management System Tools is unclean. The data requires to be devoid from noise, punctuations, misspellings, htmls, etc. as a start and further pre-process the data to be able to remove words which do not contribute to meaning (stop-words) and extract meaningful words (tokens) to feed the data to modelling algorithms.

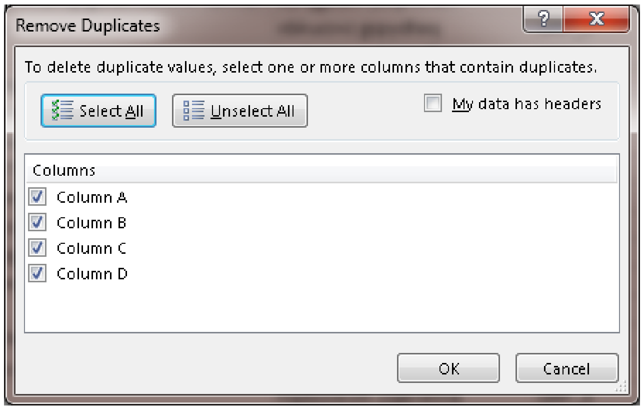


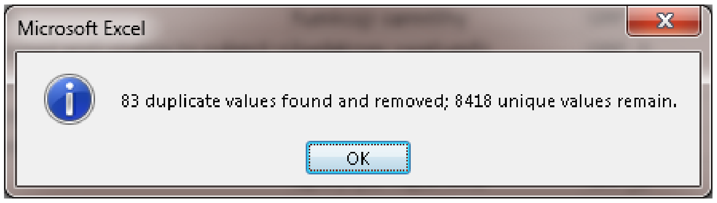
	Short_description	Description	Caller	Group
0	login issue	-verified user details.(employee# & manager na...	spxjnwir pjlcoqds	GRP_0
1	outlook	\r\n\r\nreceived from: hmjdrvpb.komuaywn@gmail...	hmjdrvpb komuaywn	GRP_0
2	cant log in to vpn	\r\n\r\nreceived from: eylqgodm.ybqkwiam@gmail...	eylqgodm ybqkwiam	GRP_0
3	unable to access hr_tool page	unable to access hr_tool page	xbkucsvz gcpydteq	GRP_0
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0

Figure 1 Raw data read from the Excel file with 4 columns

The methods employed to clean our data are used from the NLTK and Regular Expression (re) library.

Size of the data-
Duplicate entries - The dataset consists of 8500 entries of tickets. On analyzing for duplicate entries across all the 4 columns, 83 duplicates were observed and removed thus leading to unique 8417 values in the dataset.





83 duplicate values found and removed; 8418 unique values remain.

Class Imbalance – Datasets require proper representation of Class information, i.e equal representation of all Groups. This would enable the modelling algorithms to be trained on equal amounts of data of any given class (Group). However, this is not the case, and we observe data imbalance in the dataset.

Given the Group information, the Unique Group Count = 74. However, there is a class imbalance w.r.t the representation of the groups in the data. Out of the total 8500 tickets, 47% represents Group_0 tickets.

One way to control the group data and maintain imbalance is by setting a 'Threshold' value which filters out the minority Group data. The threshold value is set at default 50.

```
# Reset Assignment Group for group types with less data
Frequency_Threshold = 50
count = df_v1['Group'].value_counts(ascending=True)
idx = count[count.lt(Frequency_Threshold)].index
df_v1.loc[df_v1['Group'].isin(idx), 'Group'] = 'GRP_Manual'
print("Updated unique group types",df_v1['Group'].nunique())
df_v1['Group'].value_counts(ascending=True)
```

Figure 2 Setting a threshold at 50 to control the class imbalance during modelling

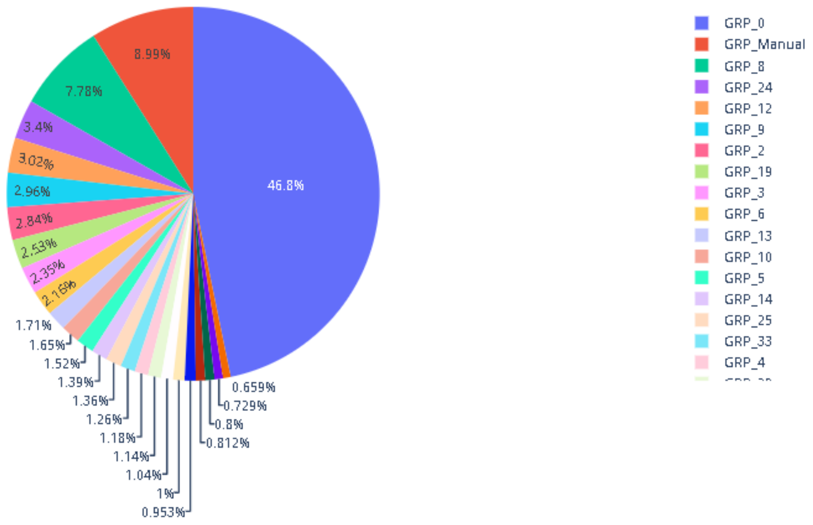
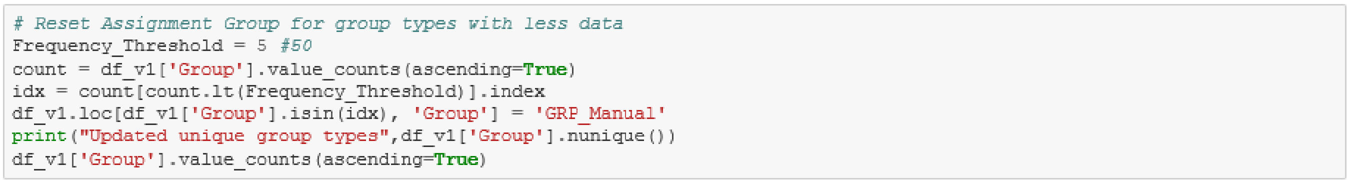


Figure 3 Out of the total 8500 tickets, 47% represents Group_0 tickets

We can tune this Threshold value based on the business requirements to filter the appropriate information into our dataset.



```
# Reset Assignment Group for group types with less data
Frequency_Threshold = 5 #50
count = df_v1['Group'].value_counts(ascending=True)
idx = count[count.lt(Frequency_Threshold)].index
df_v1.loc[df_v1['Group'].isin(idx), 'Group'] = 'GRP_Manual'
print("Updated unique group types",df_v1['Group'].nunique())
df_v1['Group'].value_counts(ascending=True)
```

Figure 4 Setting a threshold = 5 to control the class imbalance during modelling

2. Summary of the Approach to EDA and Pre-processing

Analyze and understand the structure of data –

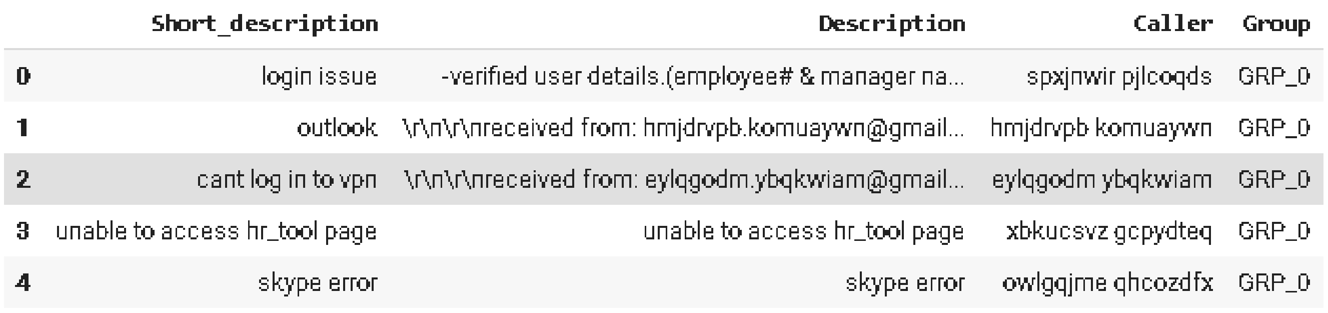
Reading the dataset – The dataset is located in the google drive as describe above and accessed using Pandas library. This file is then stored in a dataframe. While reading the file, 'Assignment group' is renamed to 'Group' and 'Short description' to 'Short_description'.

```
# Read Dataset
file_name = "Ticket_Data.xlsx"
df = pd.read_excel(file_name,encoding='cp1252')
df = df.rename(columns = {"Short description": "Short_description",
                          "Assignment group": "Group"})
```

Figure 5 Reading the dataset using Pandas library

Inorder to get a glimpse of the dataframe, we use dataframe.head()

```
df.head()
```

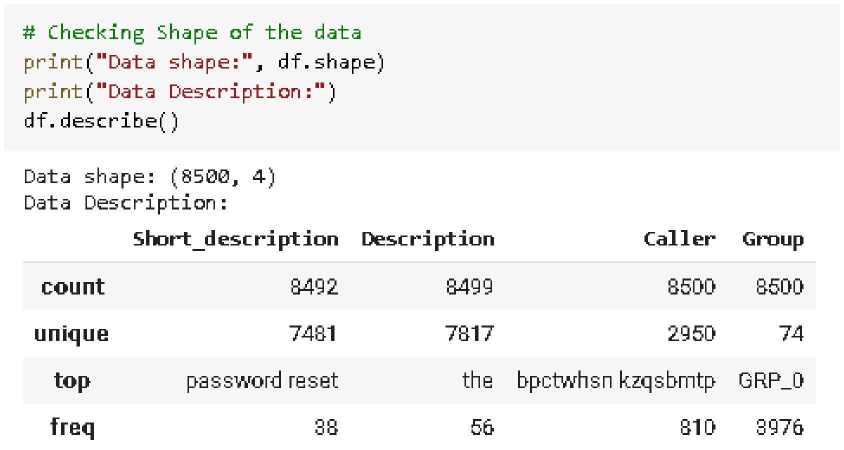


	Short_description	Description	Caller	Group
0	login issue	-verified user details.(employee# & manager na...	spxjnwir pjlcoqds	GRP_0
1	outlook	\r\n\r\nreceived from: hmjdrvpb.komuaywn@gmail...	hmjdrvpb komuaywn	GRP_0
2	cant log in to vpn	\r\n\r\nreceived from: eylqgodm.ybqkwiam@gmail...	eylqgodm ybqkwiam	GRP_0
3	unable to access hr_tool page	unable to access hr_tool page	xbkucsvz gcpydteq	GRP_0
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0

Figure 6 Getting a preview of the dataframe

We need to analyze the shape of the data and get a basic notion of the data columns. We can use dataframe.shape and dataframe.describe() respectively to achieve the tasks.

- There are 4 columns namely Short_description, Description, Caller and Group. The total number of entries are 8500 in the dataframe.
- The count is different for each column, indicating missing values at first glimpse.
- Unique captures the unique description/ content available in each column. Eg. There are 74 unique groups in the dataframe.
- Top identifies the top word/content in the columns
- Frequency captures the frequency at which the top word/content appears in the columns.



```python
# Checking Shape of the data
print("Data shape:", df.shape)
print("Data Description:")
df.describe()
```

```
Data shape: (8500, 4)
Data Description:
```

	Short_description	Description	Caller	Group
count	8492	8499	8500	8500
unique	7481	7817	2950	74
top	password reset	the	bpctwhsn kzqsbmtp	GRP_0
freq	38	56	810	3976

Figure 7 Checking shape and basic description of the data

Visualize data -

Visuals are a great way to analyze data and get an idea about the data that we are handling. We resorted to using Word Clouds to analyze the following:

- Most frequent words in Short_description
- Most frequent words in Description
- And later, most frequent words in Summary

```
def wordCloudText(df_column):
  title = ("Most Frequent words in ") + df_column.name
  stopwords = set(STOPWORDS)
  wordcloud = WordCloud(background_color='black', stopwords=stopwords, max_words=200,
                        max_font_size=40, random_state=42).generate(str(df_column.values.tolist()))
  print(wordcloud)
  fig = plt.figure(1,figsize = (20, 8))
  plt.imshow(wordcloud)
  plt.title(title ,fontsize=30)
  plt.axis('off')
  plt.show()
```

```
wordCloudText(df.Short_description)
```

```
<wordcloud.wordcloud.WordCloud object at 0x7f9b3897c6a0>
```

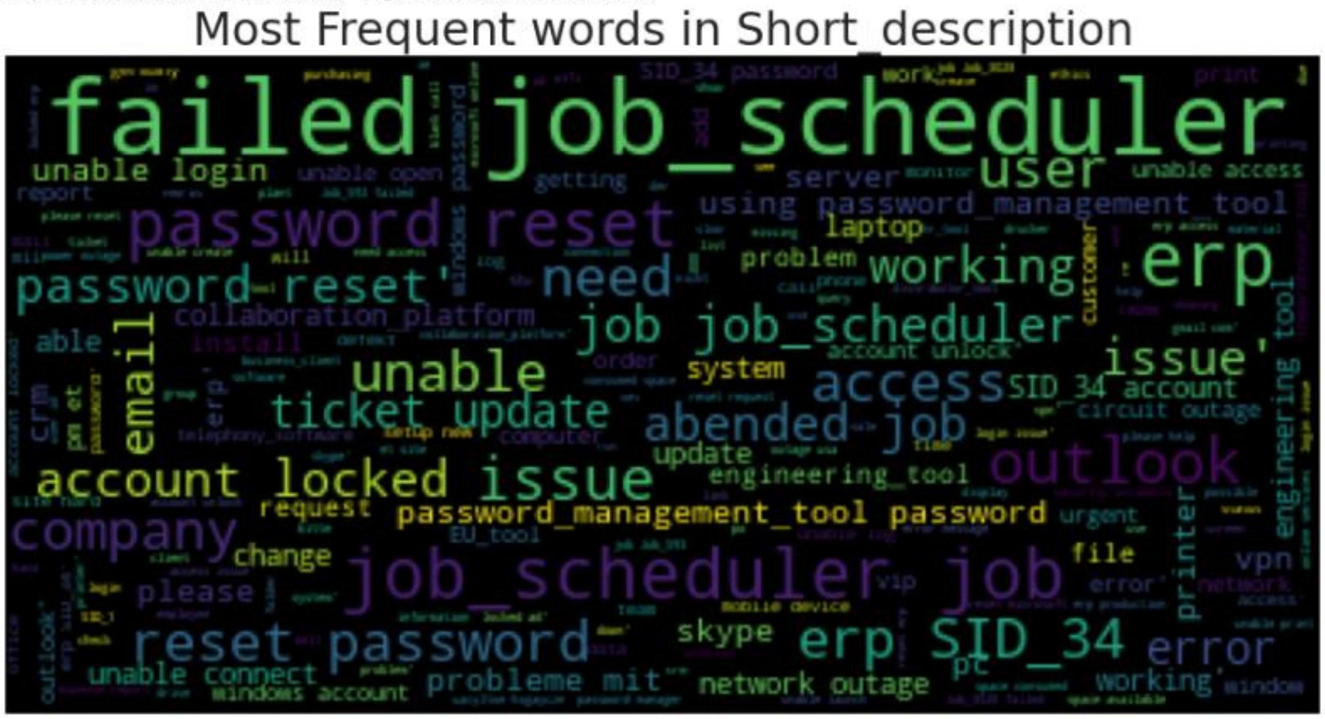


Figure 8 Most frequent words in Short_description

```
wordCloudText(df.Description)
```

```
<wordcloud.wordcloud.WordCloud object at 0x7f9b387b42e8>
```

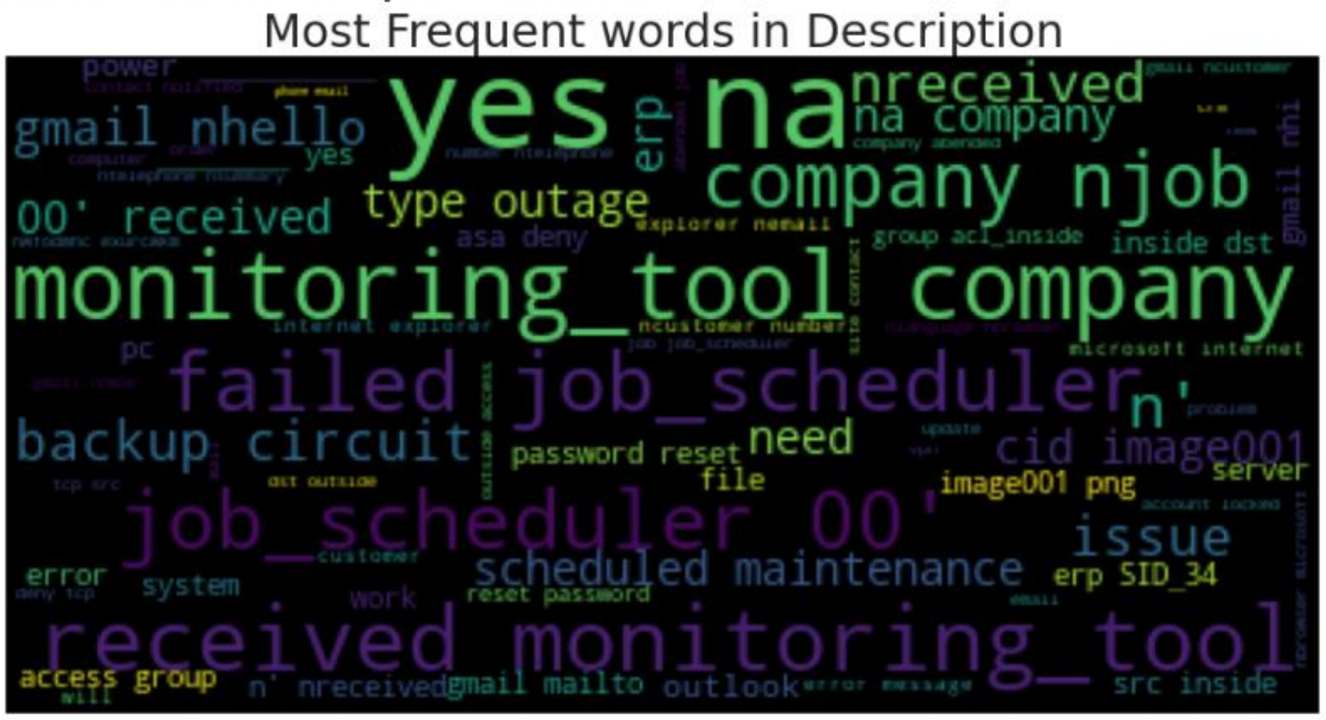


Figure 9 Most frequent words in Description

On finding the counts of each group using dataframe['col'].value_counts(), we observe that the GRP_0 has the highest presence with a frequency of 3976. The top 5 Groups are listed below.

```
#Creating dataframe of Groups on the basis of their value counts
n_grp = list(df['Group'].value_counts())
grp_name = list(df['Group'].value_counts().index)

grp = pd.DataFrame(data=grp_name,columns=['grp_name'])
grp['n_grp'] = n_grp
print(len(grp['n_grp']))
print(grp.head())
```

```
74
  grp_name  n_grp
0    GRP_0   3976
1    GRP_8    661
2   GRP_24    289
3   GRP_12    257
4    GRP_9    252
```

Figure 10 Top 5 Groups with their frequencies

This barplot represents the frequency distribution of the Groups present in the dataframe.

```
plt.figure(figsize=(15,8))
sns.barplot(x="grp_name", y="n_grp", data=grp,palette='plasma')
plt.xticks(rotation=90, ha='right')
plt.tight_layout()
```

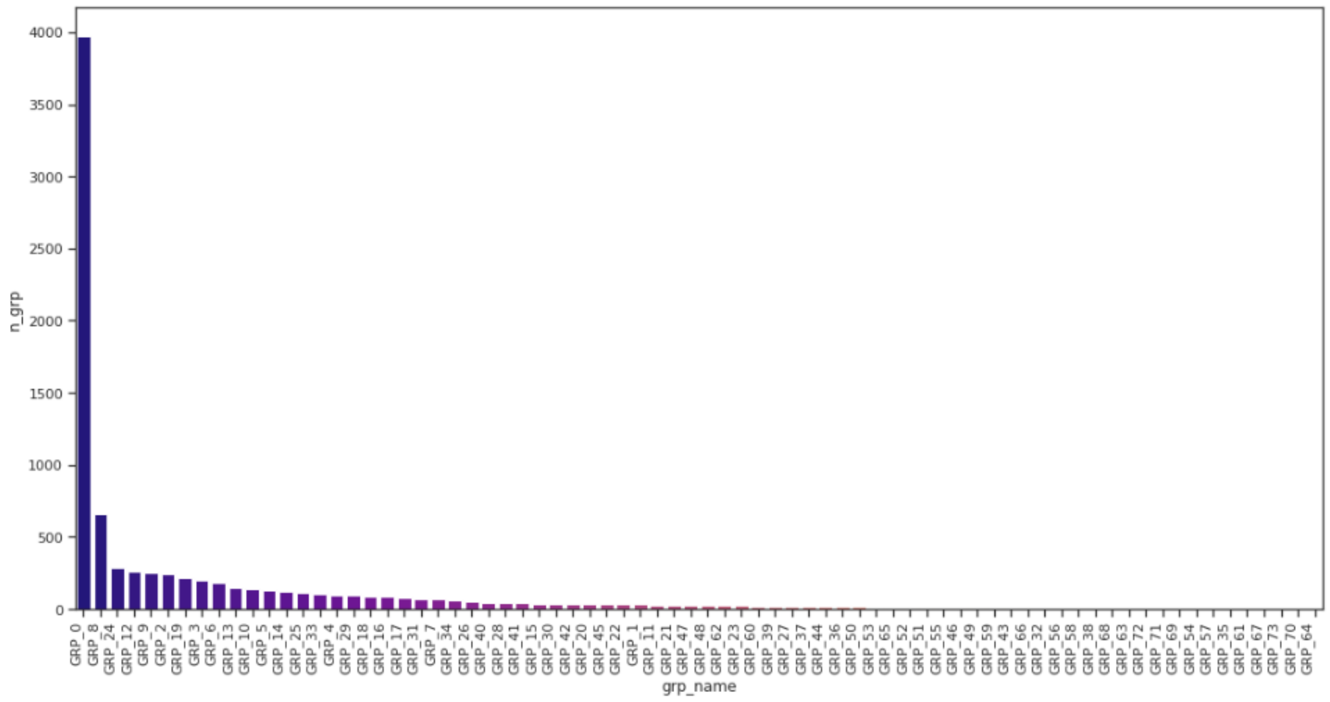


Figure 11 Barplot representing the frequency distribution of the groups

```
#Visualization of ditribution of data after merging of groups have less data than the threshold frequency
fig = px.pie(grp,values='n_grp', names='grp_name')
fig.show()
```

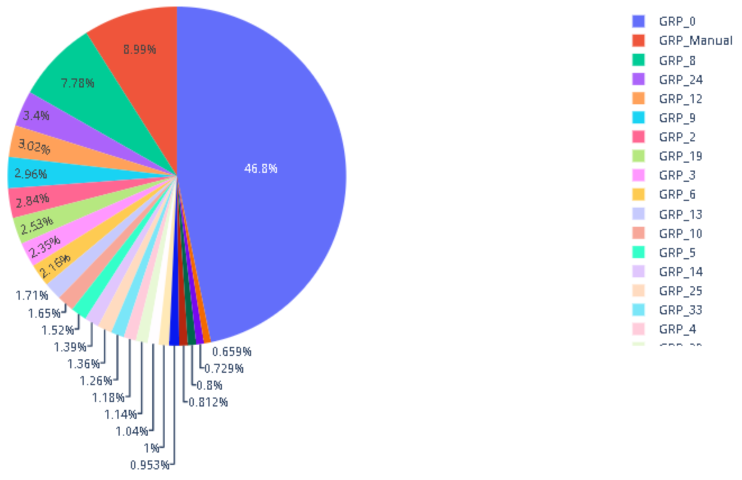


Figure 12 The pie chart showing the representation of Groups present in the data with Threshold VALUE = 50

The “Column – Caller” is anonymously provided in the dataset and hence it is difficult to comprehend. As seen in Figure 13 Caller Data anonymously provided in the dataset.

```
#Visualizing some of the Caller data
plt.figure(figsize=(20, 10))
df['Caller'].value_counts().head(30).plot(kind='barh', figsize
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9b332aeb70>
```

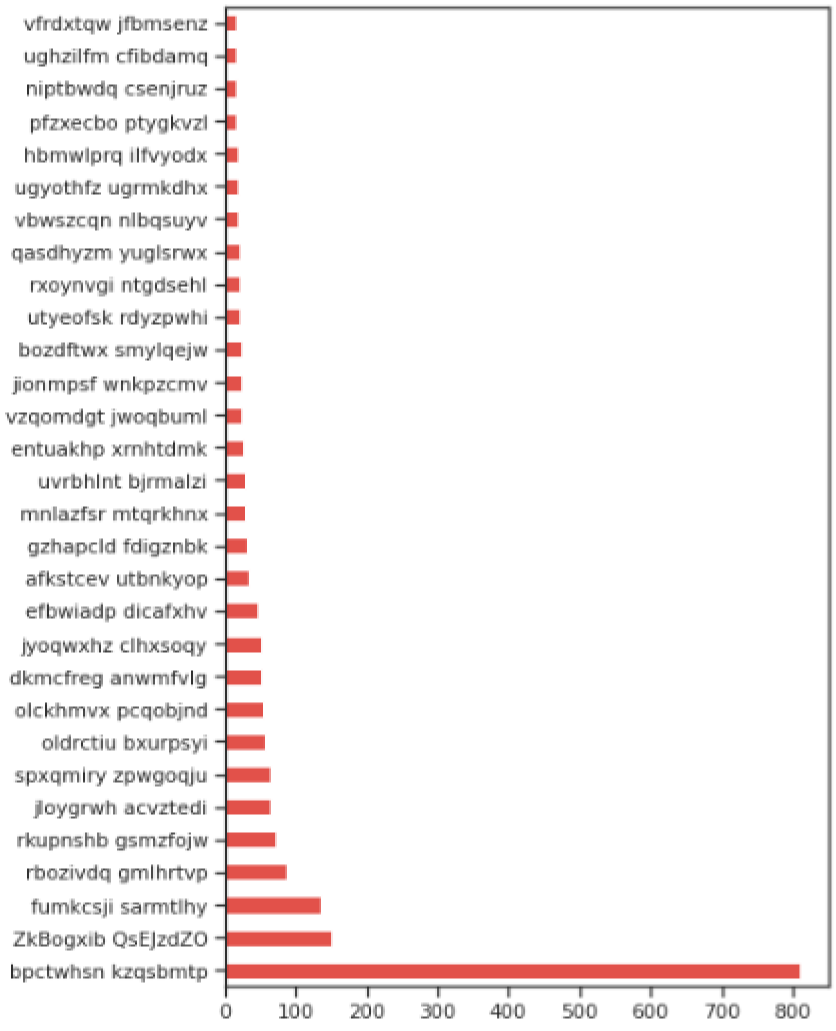


Figure 13 Caller Data anonymously provided in the dataset

The Caller data is then processed by encoding it with a uniform Prefix followed by a number. The data shows that Caller 1 has the highest frequency of 810 tickets logged. The top 10 Caller information is visible below.

```
#Since caller column contains anonymous data, assigning name Caller1, Caller2,..... for better visualization
count = 0
new_caller = []
while count != len(data):
    new_caller.append('Caller'+''+ str(count+1))
    count = count +1
data['caller'] = new_caller
data = data.head(20)
data.head(10)
```

	caller	n_caller
0	Caller1	810
1	Caller2	151
2	Caller3	134
3	Caller4	87
4	Caller5	71
5	Caller6	64
6	Caller7	63
7	Caller8	57
8	Caller9	54
9	Caller10	51

Figure 14 Caller frequency

This graph depicts the frequency distribution of the caller information.

```
#top 20 callers
fig = px.bar(data, x='caller', y='n_caller',hover_data=['n_caller'])
fig.show()
```

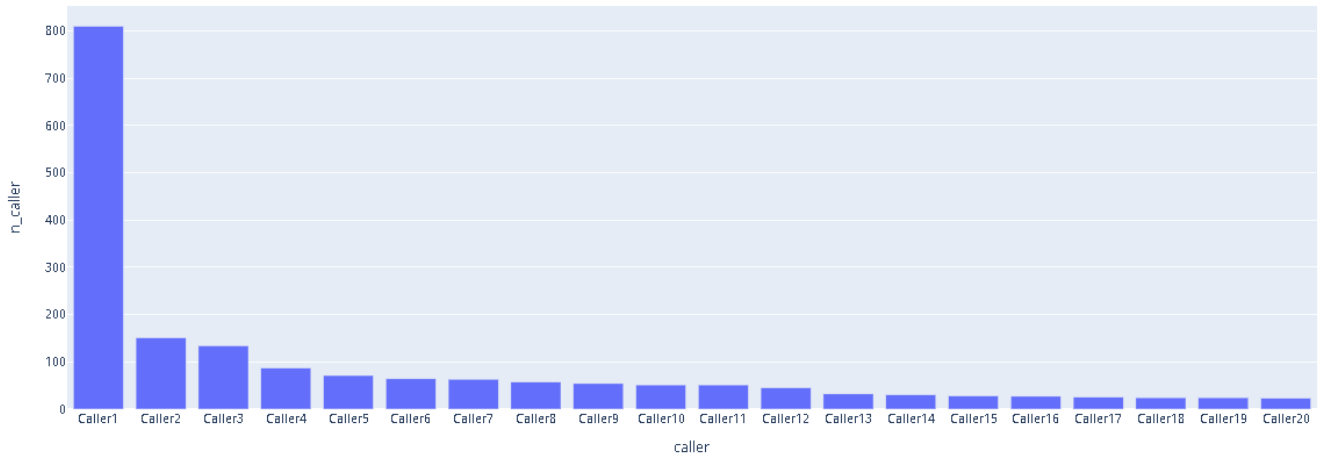
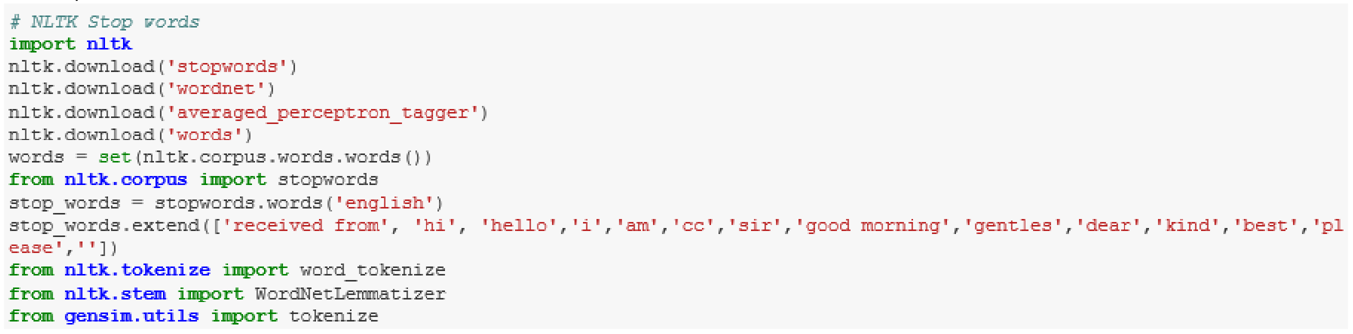


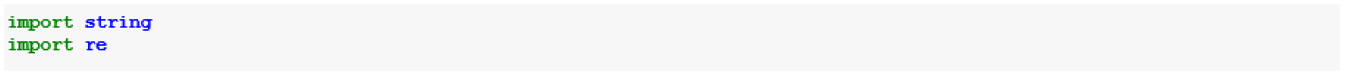
Figure 15 Barplot representing the Caller Frequency Data for Top 20 Callers

Text preprocessing -

The methods employed to clean our data are used from the NLTK and Regular Expression (re) library.



```python
# NLTK Stop words
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('words')
words = set(nltk.corpus.words.words())
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['received from', 'hi', 'hello','i','am','cc','sir','good morning','gentles','dear','kind','best','pl
ease',''])
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from gensim.utils import tokenize
```



```python
import string
import re
```

Figure 16 - Utilizing the NLTK and re Library to clean the data

<u>Stop-words</u> - Text and document classification includes many words which do not contain important significance to be used in classification algorithms, such as {‘a’, ‘about’, ‘about’, ‘after’, ‘again’,..}.
The most common technique to deal with these words is to remove them from the texts and documents.

<u>Noise removal</u> **-** Another issue of text cleaning as a pre-processing step is noise removal. Text documents generally contains characters like punctuations or special characters and they are not necessary for text mining or classification purposes. Although punctuation is critical to understand the meaning of the sentence, but it can affect the classification algorithms negatively.

Noise that is addressed as a part of this section includes – removal of brackets, newline, multi-line spaces, numeric and alpha numeric, non-ascii text, underscores, email addresses, and disclaimers.

```python
import string
import re

# Function for Text Cleaning with regex. Pass the column
def text_preprocessing(df_column):
  data = df_column.values.tolist() # Convert to list
  temp = []
  for sentence in data:
      sentence = sentence.replace("select the following link to view the disclaimer in an alternate language", '')  # r
emove disclaimer text
      sentence = re.sub(r"\[(.*?)\]"," ", sentence)  # remove text in []
      sentence = re.sub(r"\((.*?)\)"," ", sentence)  # remove text in ()
      sentence = re.sub(r"[[h][t][t][p][\S]+|[w][w][w][\S]+|[\S]+[@][\S]+"," ", sentence)  # remove email addresses, we
b address and urls
      sentence = re.sub(r"[\S]+[\d]+[\S]+"," ", sentence) # remove alphanumerics and numerics (dates, time, request id
etc.)
      sentence = re.sub(r"\W(?<!['. ])"," ", sentence)  # remove all non words with negative look back except ('. space
s)
      sentence = re.sub(r"[^a-zA-z.| ]+"," ", sentence) # remove non-alphabetic text
      sentence = re.sub(r"[\_]+"," ", sentence) # remove underscores
      sentence = re.sub(r"[\s]+"," ", sentence) # replace multiple spaces with single space
      sentence = sentence.strip('\n')
      sentence = sentence.lower()
      temp.append(sentence)
  return(temp)
```

Figure 17 Noise Removal from text

Capitalization - Sentences can contain a mixture of uppercase and lower case letters. Multiple sentences make up a text document. To reduce the problem space, the most common approach is to reduce everything to lower case. This brings all words in a document in same space, but it often changes the meaning of some words, such as "US" to "us" where first one represents the United States of America and second one is a pronoun. To solve this, slang and abbreviation converters can be applied.

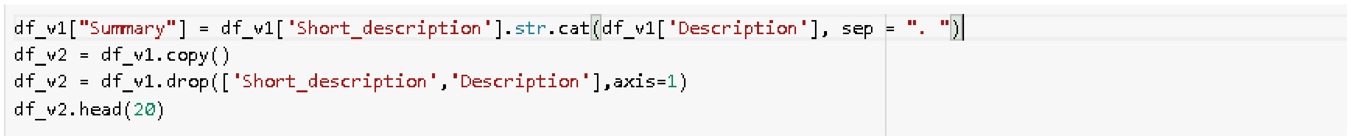
sentence: Cant log into vpn
capitalized text: cant log into vpn

Lemmatization - Text lemmatization is the process of eliminating redundant prefix or suffix of a word and extract the base word (lemma).
word: see or saw
lemma text: see or saw depending on whether the use of the token was as a verb or a noun

```python
# lemmetise words
wordnet_lemmatizer = WordNetLemmatizer()
temp = []
for eachrow in data:
    lemma_words = []
    for eachword in eachrow:
        eachword = wordnet_lemmatizer.lemmatize(eachword, pos = "n")
        eachword = wordnet_lemmatizer.lemmatize(eachword, pos = "v")
        eachword = wordnet_lemmatizer.lemmatize(eachword, pos = ("a"))
        lemma_words.append(eachword)
    temp.append(lemma_words)
```

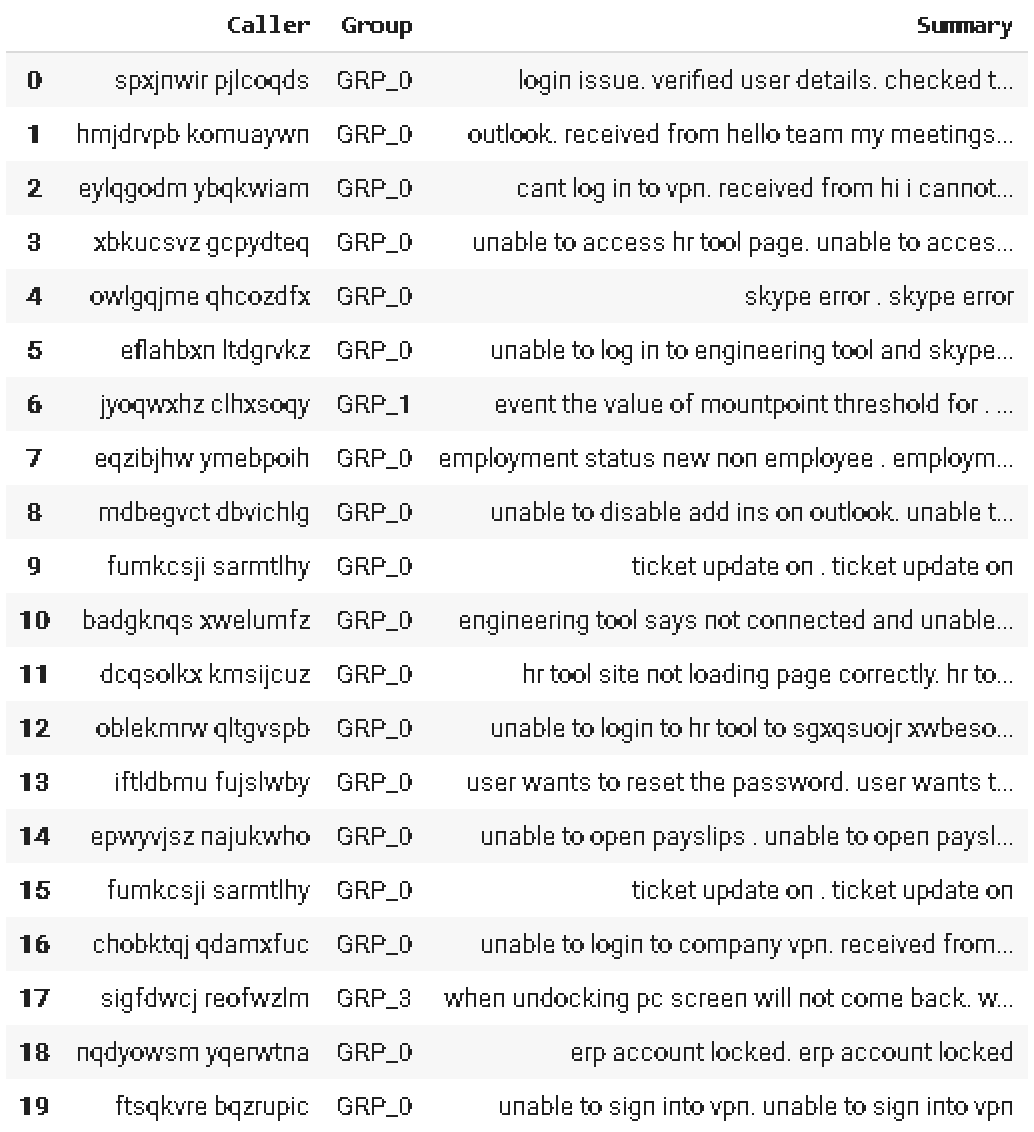
Figure 18 Lemmatizing words

Concatenation of Short_description and Description into a new field – 'Summary'



```
df_v1["Summary"] = df_v1['Short_description'].str.cat(df_v1['Description'], sep = ". ")
df_v2 = df_v1.copy()
df_v2 = df_v1.drop(['Short_description','Description'],axis=1)
df_v2.head(20)
```

Figure 19 Concatenation of Short_description and Description into a new field – 'Summary'

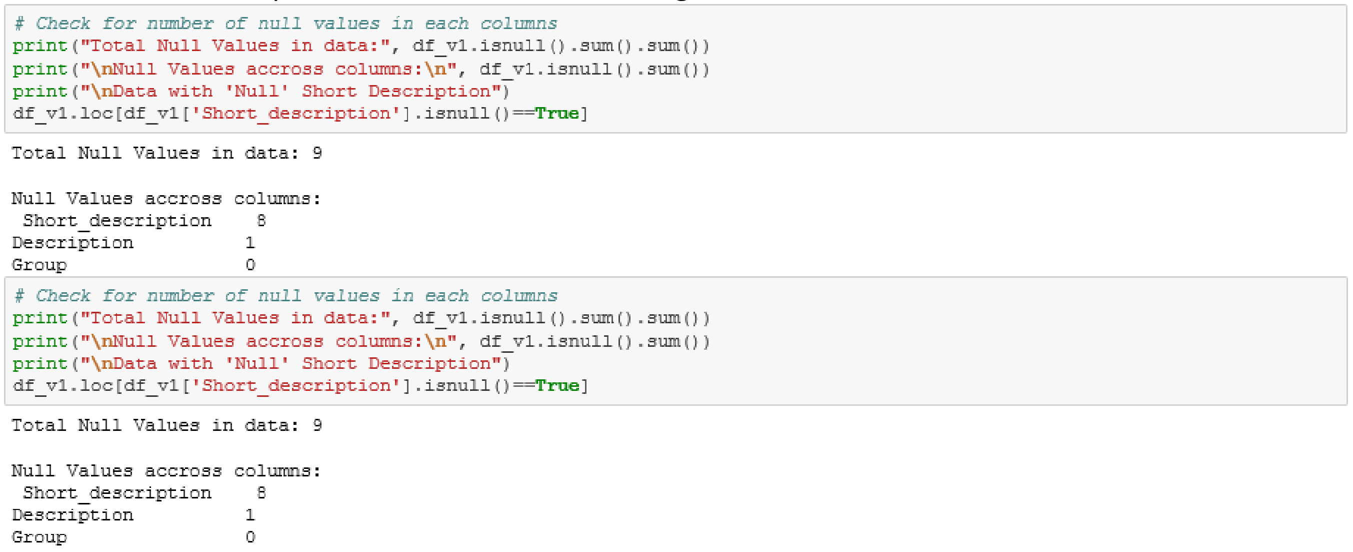


	Caller	Group	Summary
0	spxjnwir pjlcoqds	GRP_0	login issue. verified user details. checked t...
1	hmjdrvpb komuaywn	GRP_0	outlook. received from hello team my meetings...
2	eylqgodm ybqkwiam	GRP_0	cant log in to vpn. received from hi i cannot...
3	xbkucsvz gcpydteq	GRP_0	unable to access hr tool page. unable to acces...
4	owlgqjme qhcozdfx	GRP_0	skype error . skype error
5	eflahbxn ltdgrvkz	GRP_0	unable to log in to engineering tool and skype...
6	jyoqwxhz clhxsoqy	GRP_1	event the value of mountpoint threshold for . ...
7	eqzibjhw ymebpoih	GRP_0	employment status new non employee . employm...
8	mdbegvct dbvichlg	GRP_0	unable to disable add ins on outlook. unable t...
9	fumkcsji sarmtlhy	GRP_0	ticket update on . ticket update on
10	badgknqs xwelumfz	GRP_0	engineering tool says not connected and unable...
11	dcqsolkx kmsijcuz	GRP_0	hr tool site not loading page correctly. hr to...
12	oblekmrw qltgvspb	GRP_0	unable to login to hr tool to sgxqsuojr xwbeso...
13	iftldbmu fujslwby	GRP_0	user wants to reset the password. user wants t...
14	epwyvjsz najukwho	GRP_0	unable to open payslips . unable to open paysl...
15	fumkcsji sarmtlhy	GRP_0	ticket update on . ticket update on
16	chobktqj qdamxfuc	GRP_0	unable to login to company vpn. received from...
17	sigfdwcj reofwzlm	GRP_3	when undocking pc screen will not come back. w...
18	nqdyowsm yqerwtna	GRP_0	erp account locked. erp account locked
19	ftsqkvre bqzrupic	GRP_0	unable to sign into vpn. unable to sign into vpn

Figure 20 Viewing the dataframe with the new field – 'Summary'

<u>Checking for Missing values</u> - On analyzing the raw data, the fields (Short Description, Description, Caller and Group) were verified for missing values. It was observed that there are 8 missing values in Column - Short Description and 1 missing value in Column – Description and none in Column – Group.

The missing values were addressed by imputing values like space, filler words that would be eradicated in the stop-word removal or text cleaning.



```
# Check for number of null values in each columns
print("Total Null Values in data:", df_v1.isnull().sum().sum())
print("\nNull Values accross columns:\n", df_v1.isnull().sum())
print("\nData with 'Null' Short Description")
df_v1.loc[df_v1['Short_description'].isnull()==True]
```

```
Total Null Values in data: 9

Null Values accross columns:
 Short_description    8
Description          1
Group                0
```

```
# Check for number of null values in each columns
print("Total Null Values in data:", df_v1.isnull().sum().sum())
print("\nNull Values accross columns:\n", df_v1.isnull().sum())
print("\nData with 'Null' Short Description")
df_v1.loc[df_v1['Short_description'].isnull()==True]
```

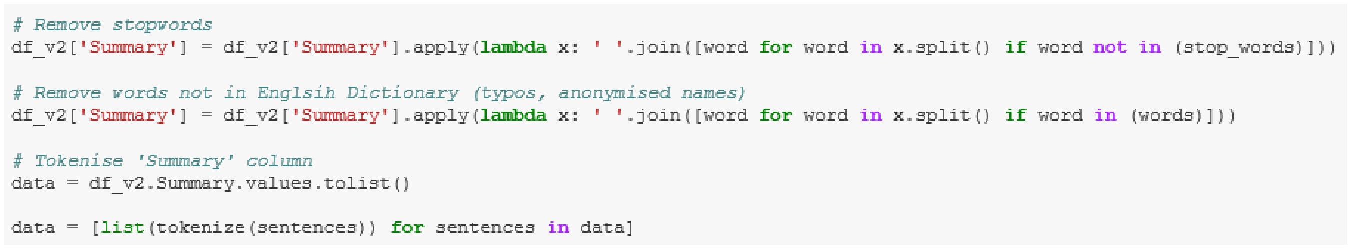
```
Total Null Values in data: 9

Null Values accross columns:
 Short_description    8
Description          1
Group                0
```

Figure 21 Checking for missing values and imputing data in the dataset columns

Language translations - Translation? – minority and omitted from the modelling for Milestone 1

Create word vocabulary and Tokens -

Tokenization - Tokenization is the process of breaking down a stream of text into words, phrases, symbols, or any other meaningful elements called tokens. The main goal of this step is to extract individual words in a sentence. Along with text classification, in text mining, it is necessary to incorporate a parser in the pipeline which performs the tokenization of the documents; for example:
sentence: cant log into vpn
tokens: {'cant', 'log', 'into', 'vpn'}

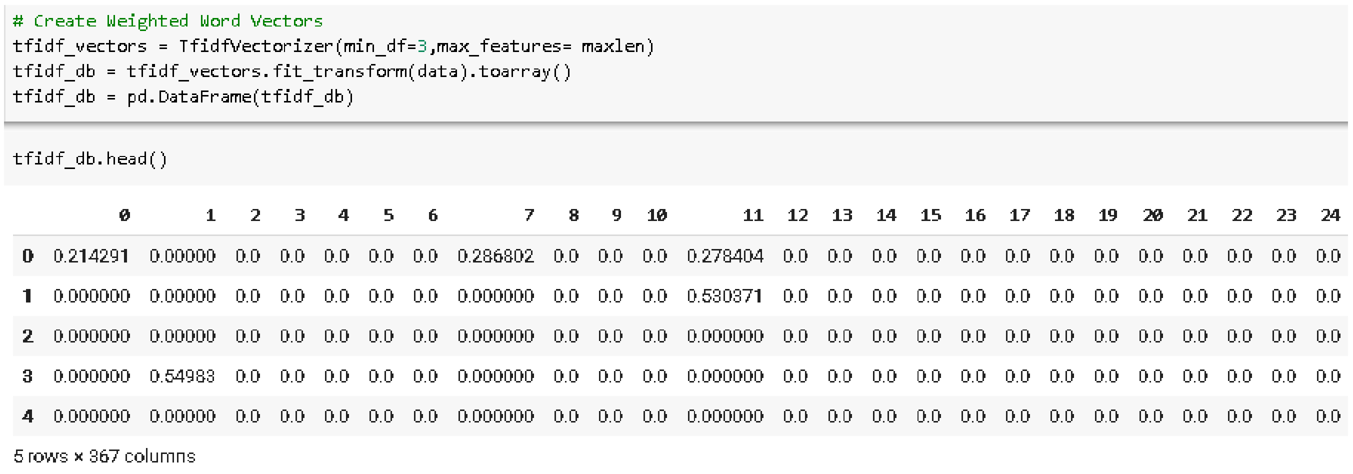


```
# Remove stopwords
df_v2['Summary'] = df_v2['Summary'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_words)]))

# Remove words not in Englsih Dictionary (typos, anonymised names)
df_v2['Summary'] = df_v2['Summary'].apply(lambda x: ' '.join([word for word in x.split() if word in (words)]))

# Tokenise 'Summary' column
data = df_v2.Summary.values.tolist()

data = [list(tokenize(sentences)) for sentences in data]
```

Figure 22 Utilizing Tokenize to create word tokens

Weighted Words **-** The most basic form of weighted word feature extraction is TF, where each word is mapped to a number corresponding to the number of occurrences of that word in the whole corpora. Methods that extend the results of TF generally use word frequency as a boolean or logarithmically scaled weighting.
In all weight words methods, each document is translated to a vector (with length equal to that of the document) containing the frequency of the words in that document. Although this approach is intuitive, it is limited by the fact that particular words that are commonly used in the language may dominate such representations.

Term Frequency-Inverse Document Frequency (tf-idf) - Different word embedding procedures have been proposed to translate these unigrams into consummable input for machine learning algorithms. A very simple way to perform such embedding is weighted words term-frequency~(TF) and TF-IDF where each word will be mapped to a number corresponding to the number of occurrence of that word in the whole corpora.
Although tf-idf tries to overcome the problem of common terms in document, it still suffers from some other descriptive limitations. Namely, tf-idf cannot account for the similarity between words in the document since each word is presented as an index.



```
# Create Weighted Word Vectors
tfidf_vectors = TfidfVectorizer(min_df=3,max_features= maxlen)
tfidf_db = tfidf_vectors.fit_transform(data).toarray()
tfidf_db = pd.DataFrame(tfidf_db)
```

```
tfidf_db.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	0.214291	0.00000	0.0	0.0	0.0	0.0	0.0	0.286802	0.0	0.0	0.0	0.278404	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.000000	0.00000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.530371	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.000000	0.00000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.000000	0.54983	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.000000	0.00000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 367 columns

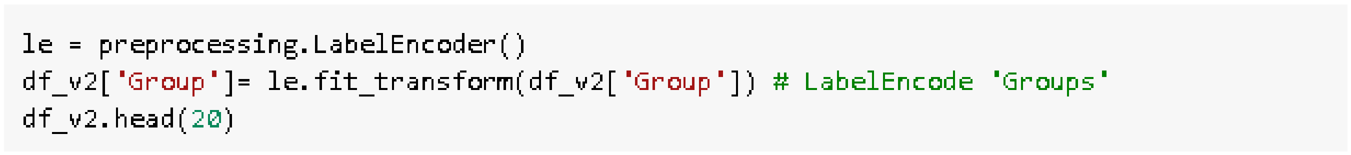
Figure 23 Creating weighted vectors of the vocabulary

Word Embedding- Word embedding is a feature learning technique in which each word or phrase from the vocabulary is mapped to a N dimension vector of real numbers.

In the recent years, with development of more complex models, such as neural nets, new methods have been presented that can incorporate concepts, such as similarity of words and part of speech tagging.
Word2vec and Glove, two of the most common methods that have been successfully used for deep learning techniques that will be employed in Milestone 2 of this project.

Build a Classification model – Train and Test the models

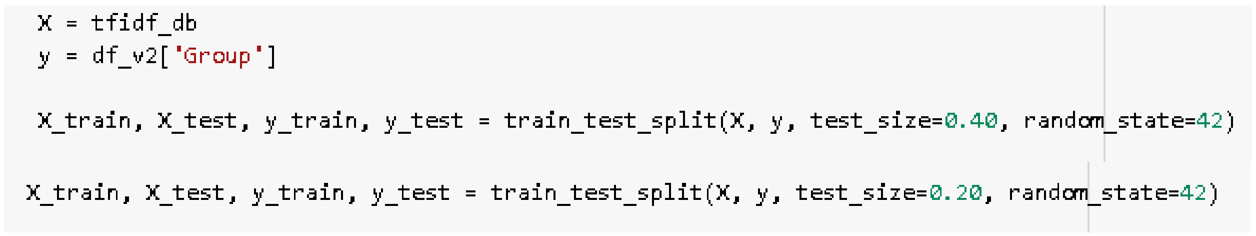
Once we create a vectorized form representing the vocabulary, we now encode the Group information labelled as GRP_X as 0,1...using label encoder(). This field represents the Target column.



```
le = preprocessing.LabelEncoder()
df_v2['Group']= le.fit_transform(df_v2['Group']) # LabelEncode 'Groups'
df_v2.head(20)
```

Figure 24 Encoding the Group Data using LabelEncoder

Mapping the X(input data) and y(target) to the vectorized data and encoded groups inorder to build our models. The training and testing datasets are formulated from the X and y data, in a ratio of 60%/40% training and test data respectively. The ideal range is 60-40 to 80-20. On changing the *test_size* parameter, we can modify the testing datasize.



```
X = tfidf_db
y = df_v2['Group']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40, random_state=42)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

Figure 25 Setting X and y(Target), splitting between training and testing sets

3. Deciding Models and Model Building

Intro…(TBD)

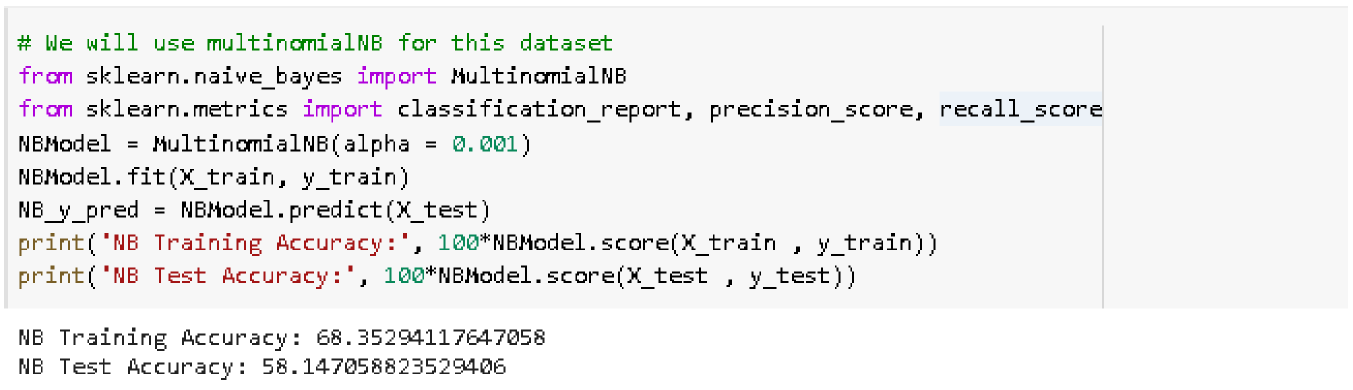
Naive Bayes

Naive Bayes is a Probablistic learning algorithm derived from Bayes Theorem. Naive Bayes Model is considered to be extremely fast, reliable, and has stable classification ability relative to other classification algorithms. The algorithm is based on the assumption that each feature in independent of each other while predicting the classification.

Pros: Simple, fast and well in multi class prediction Performs better with less training data as it assumes feature independence

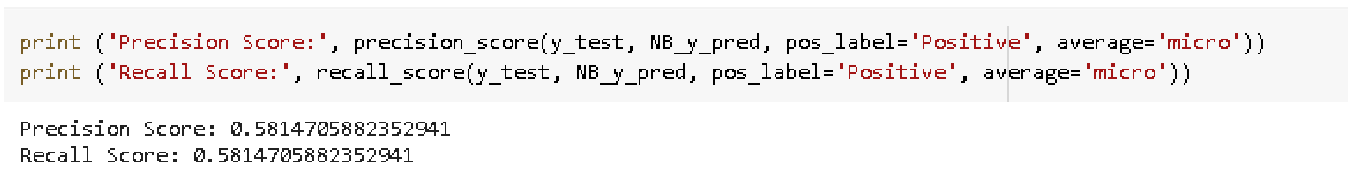
Cons: bad estimator hence the probability outputs are not taken too seriously Assumptions of independent feature cannot represent real time data Zero frequency - If training data set gets a category not trained on earlier, then model will assign a 0 (zero) probability and will be unable to make a prediction.

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification), we will build Multinomial Naive Bayes model for our dataset.



```
# We will use multinomialNB for this dataset
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, precision_score, recall_score
NBModel = MultinomialNB(alpha = 0.001)
NBModel.fit(X_train, y_train)
NB_y_pred = NBModel.predict(X_test)
print('NB Training Accuracy:', 100*NBModel.score(X_train , y_train))
print('NB Test Accuracy:', 100*NBModel.score(X_test , y_test))
```

```
NB Training Accuracy: 68.35294117647058
NB Test Accuracy: 58.147058823529406
```



```
print ('Precision Score:', precision_score(y_test, NB_y_pred, pos_label='Positive', average='micro'))
print ('Recall Score:', recall_score(y_test, NB_y_pred, pos_label='Positive', average='micro'))
```

```
Precision Score: 0.5814705882352941
Recall Score: 0.5814705882352941
```

Result –

We can see that the Training accuracy is 67% and testing accuracy is 58% with Naive Bayes Mode. The model is able to predict True Positives and False Negatives equally.

Support Vector Classifier (SVC)

Pros:

Cons:

```
# Creating SVC Model
X = tfidf_db
y = df_v2['Group']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40, random_state=42)

svm_model = SVC(kernel='linear',C=10)
svm_model.fit(X_train, y_train)
y_pred = svm_model.predict(X_test)
print('Training Accuracy:', 100*svm_model.score(X_train , y_train))
print('Test Accuracy:',100*svm_model.score(X_test , y_test))
```

```
Training Accuracy: 80.56862745098039
Test Accuracy: 59.470588235294116
```

Result-

LDA

Describe what you have learned so far, what models you have used and the progress you have made towards your intended solution.
For e.g.
Since this is a text manipulation based project, we would like to build a NLP based Bayesian model. The input would be a labelled data of those who made through the first filter and those who did not. Both the classes with have same attributes. The accuracy will be a function of both True Positive Rate and False Positive Rates. We will evaluate the Naive Bayes, Logistic Regression and Decision Tree Classifier using ROC to select top 3 models and build an ensemble based on these to ensure the accuracy is high and the model generalizes.

(Based on the nature of the problem, decide what algorithms will be suitable and why? Experiment with different algorithms and get the performance of each algorithm.)

4. How to improve your model performance?

- Hyper-paramter tuning for the models.
- Using Grid Search for the optimum hyper-parameter values.
- Address class imbalance by creating data entries, reducing data entries to balance the classes out.
- Use enhanced Word embeddings – Word2Ved and Glove
- Employ Deeplearning techniques with neural networks

(What are the approaches you can take to improve your model? Can you do some feature selection, data manipulation and model improvements.
Provide your code and as much as visualizations you can share to describe what you have done so far.)

References and sources:

1. Xxx
2. Xxx
3. Xxx

Challenges, Approach and Mitigation:

1. HW
2. SW
    a. Google Translate and other language translation modules
3. Code
4. Collaboration
    a. Github and Google Drive
    https://github.com/GLNLPGroup3/Capstone

Code and Deliverables:

1. Filenames listed, attachments in the GL portal
2. Snapshot of Github

Future Scope of this Project – Milestone 2:

1. Word embeddings using Word2vec and Glove
2. Neural nets and transfer learning modelling
    RNN
    LSTM
    BI-LSTM
3. Modularize the code into packages to maintain code re-usability and independence.
4. Translation? – minority and omitted from the modelling for Milestone 1
5. EDA – interactive graphs
6. Expectations in Module 2

<END OF REPORT>