greatlearning
Learning for Life

Automatic Ticket Assignment

(Interim Report)

Submitted by
Group 5: NLP B
PGP-AIML
Jan 2020 Batch

Supervised By
Mr. Sumit Kumar

greatlearning
Learning for Life

Index

Contents

greatlearning
Learning for Life

Team Details

Ashish Roy	ashish12459@gmail.com
Ashitha KR	itsashkr@gmail.com
Bragadeesh Sundararajan	bragadeeshs@gmail.com
Pawan Gupta	gupta.pawan227@gmail.com

greatlearning
Learning for Life

Summary of Problem Statement, Data, and Findings

Understanding the Problem Statement/Business Case

In any Support System which is following Customer Centric Approach, Incident Management plays an important role in delivering quality support to customers. An incident ticket is created by various groups of people within the organization to resolve an issue as quickly as possible based on its severity. Whenever an incident is created, it reaches the Service desk team and then it gets assigned to the respective teams to work on the incident.

The Service Desk team (L1/L2) will perform basic analysis on the user's requirement, identify the issue based on given descriptions and assign it to the respective teams.

The manual assignment of these incidents might have below disadvantages:

1. More resource usage and expenses.
2. Human errors - Incidents get assigned to the wrong assignment groups
3. Delay in assigning the tickets
4. More resolution times
5. If a particular ticket takes more time in analysis, other productive tasks get affected for the Service Desk

If this ticket assignment is automated, it can be more cost-effective, less resolution time and the Service Desk team can focus on other productive tasks.

greatlearning
Learning for Life

Objective

From the given problem description, manual assignment has possibility of wrong allocation of ticket so manual intervention is required to re-allocate the tickets which is causing delay in resolution. Also this requires Man resources for this process which can be automated.

So we want to evaluate an approach which is based upon Deep Learning OR MLbased model.

1. Using a traditional Machine learning algorithm to resolve the problem
2. Using an AI Based algorithm to resolve the problem

Understanding the Data

- Four columns – Short Description, Description, Caller and Assignment group

- There are null values in Short Description and Description column.

```
In [3]: #data=pd.read_excel('/content/drive/MyDrive/Capstone/input_data.xlsx')
        data=pd.read_excel('input_data.xlsx')
        data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8500 entries, 0 to 8499
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Short description  8492 non-null   object
 1   Description        8499 non-null   object
 2   Caller             8500 non-null   object
 3   Assignment group   8500 non-null   object
```

- 74 Assignment groups found - Target classes. The dataset looks highly skewed for some target classes.

greatlearning
Learning for Life

```
In [5]: assignment_group_count=data['Assignment group'].value_counts()
        assignment_group_count.describe()

Out[5]: count      74.000000
        mean      114.864865
        std       465.747516
        min         1.000000
        25%         5.250000
        50%        26.000000
        75%        84.000000
        max      3976.000000
        Name: Assignment group, dtype: float64
```

- Caller names in a random fashion (may not be useful for training data)

Description	Caller	Assi
employee# & manager na...	spxjnwir pjlcoqds	
ijdrvpb.komuaywn@gmail...	hmjdrvpb komuaywn	
lqgodm.ybqkwiam@gmail...	eylqgodm ybqkwiam	
able to access hr_tool page	xbkucsvz gcpydteq	
skype error	owlgqjme qhcozdfx	

- European non-English language (German) also found in the dataset by visual inspection

- Email/chat format in description

\r\n\r\nreceived from: bgqpotek.cuxakvml@gmail...
from: tvcdfqgp nrbcqwgj \nsent: friday, octobe...
\n\nreceived from: abcdri@company.com\n\nwindy...
\n\nreceived from: fbvpcytz.nokypgvx@gmail.com...

- Symbols & other special characters in the description

greatlearning
Learning for Life

\r\n\r\nreceived from: bgqpotek.cuxakvml@gmail...
from: tvcdfqgp nrbcqwgj \nsent: friday, octobe...
\n\nreceived from: abcdri@company.com\n\nwindy...
\n\nreceived from: fbvpcytz.nokypgvx@gmail.com...

- Hyperlinks, URLS & few image data found in the description Blanks found either in the short description or description field

GRP_16	from: mikhghytr wafglhdrhjop \nsent: thursday,...
GRP_30	to å°□è´°ï¼Œæ—©ä¸Šç"µè„'å¼€æœºå¼€ä¸□å‡°æ□¥ ç"µ...
GRP_9	\r\n\r\nreceived from: nlearzwi.ukdzstwi@gmail...
GRP_62	i am unable to access the machine utilities to...
GRP_49	an mehreren pc`s lassen sich verschiedene prgr...

- Few descriptions same as the short description

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

- Few words were combined together

-user unable tologin to vpn.\r\n-connected to.

- Spelling mistakes and typo errors are found

cant log in to vpn

greatlearning
Learning for Life

Summary of the Approach to EDA and Pre-Processing

Observations from Target Class

Exploratory Data Analysis

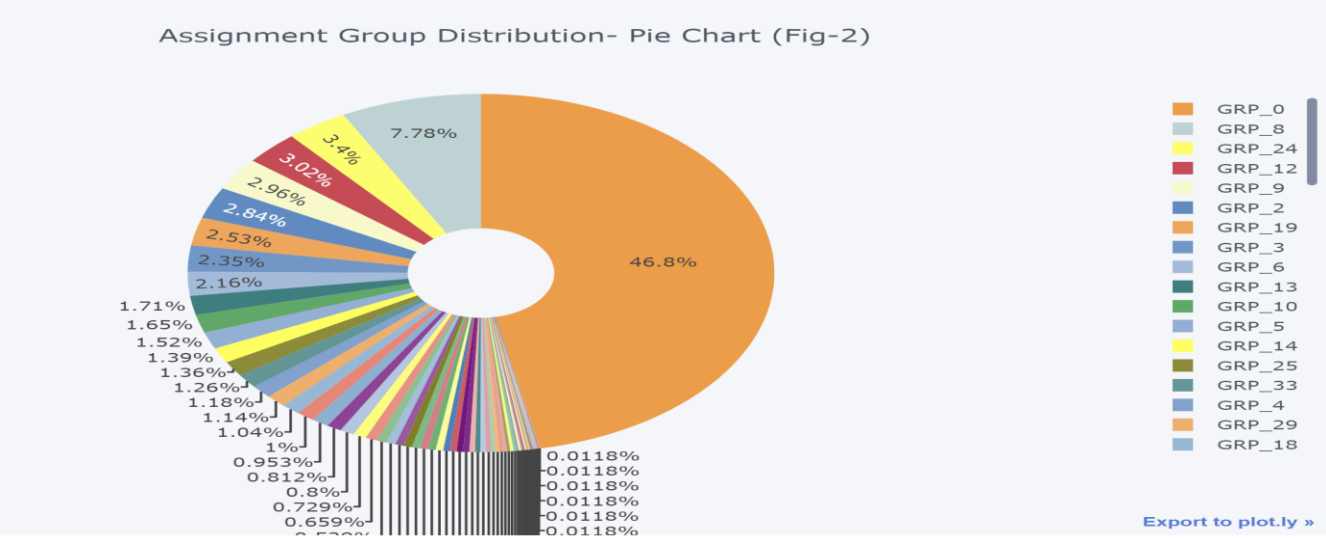
In [4]:
```
data.head()
```

Out[4]:

	Short description	Description	Caller	Assignment 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

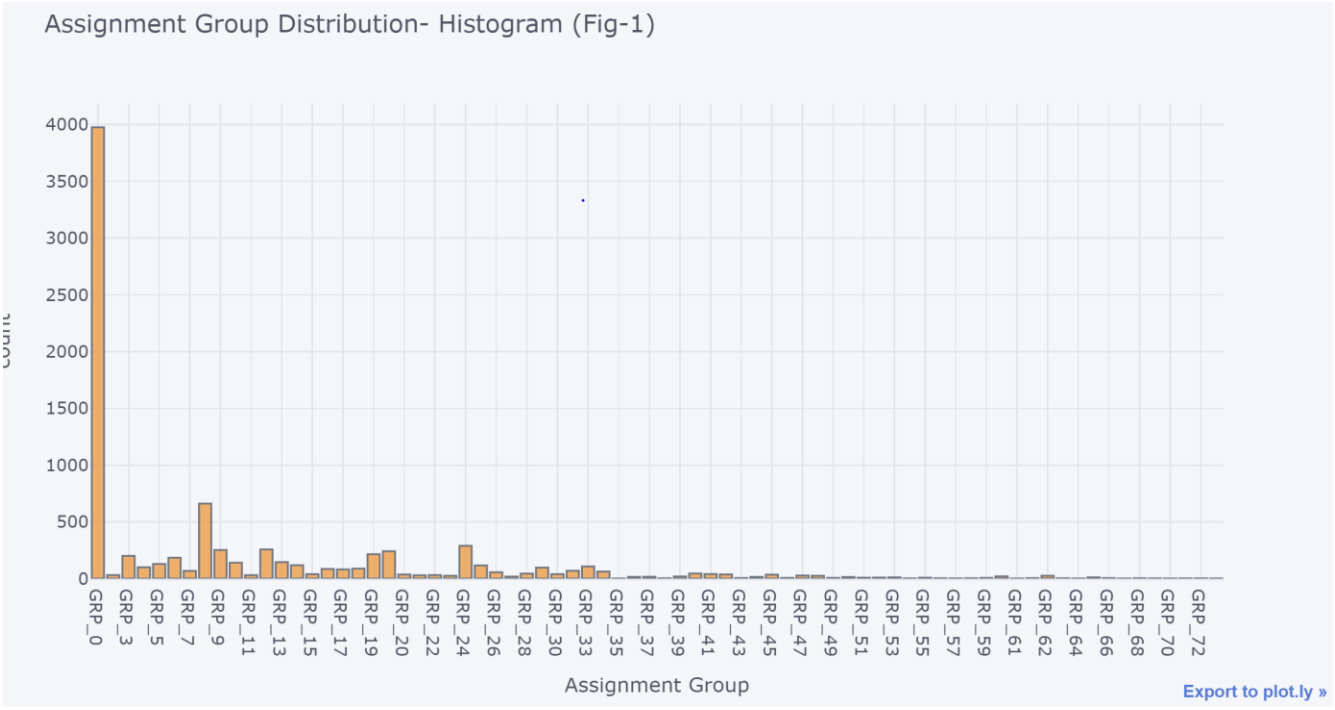
In [5]:
```
assignment_group_count=data['Assignment group'].value_counts()
assignment_group_count.describe()
```

Out[5]:
```
count      74.000000
mean      114.864865
std       465.747516
min         1.000000
25%         5.250000
50%        26.000000
75%        84.000000
max      3976.000000
Name: Assignment group, dtype: float64
```

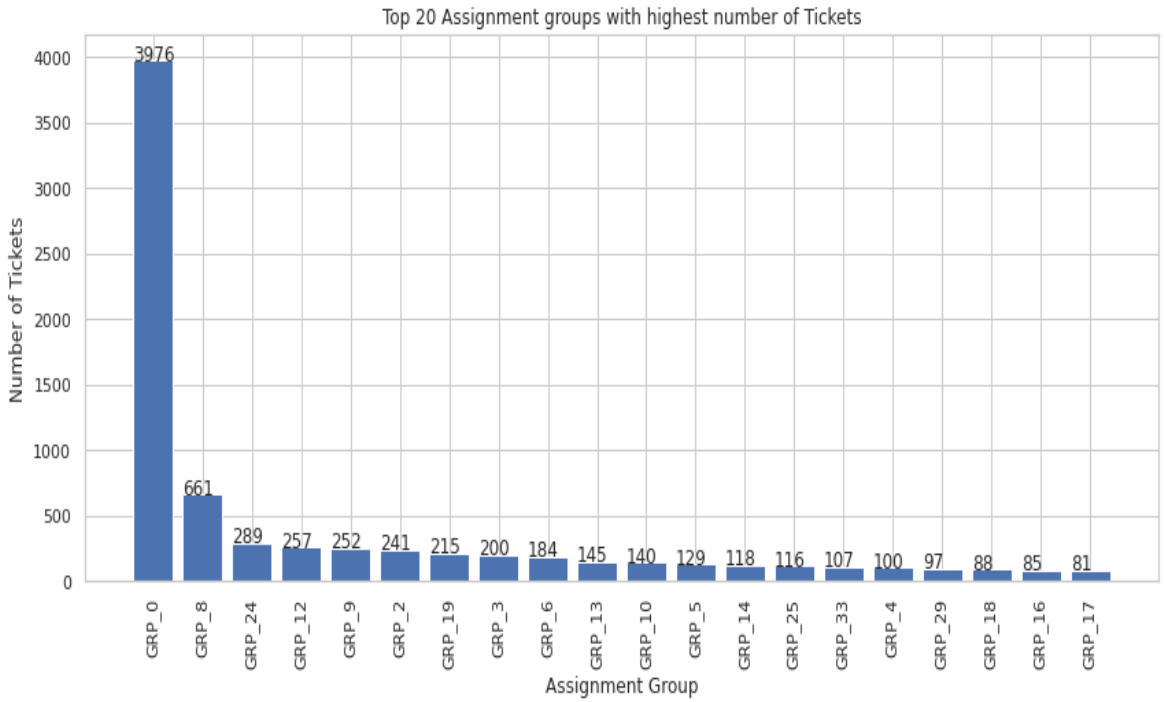


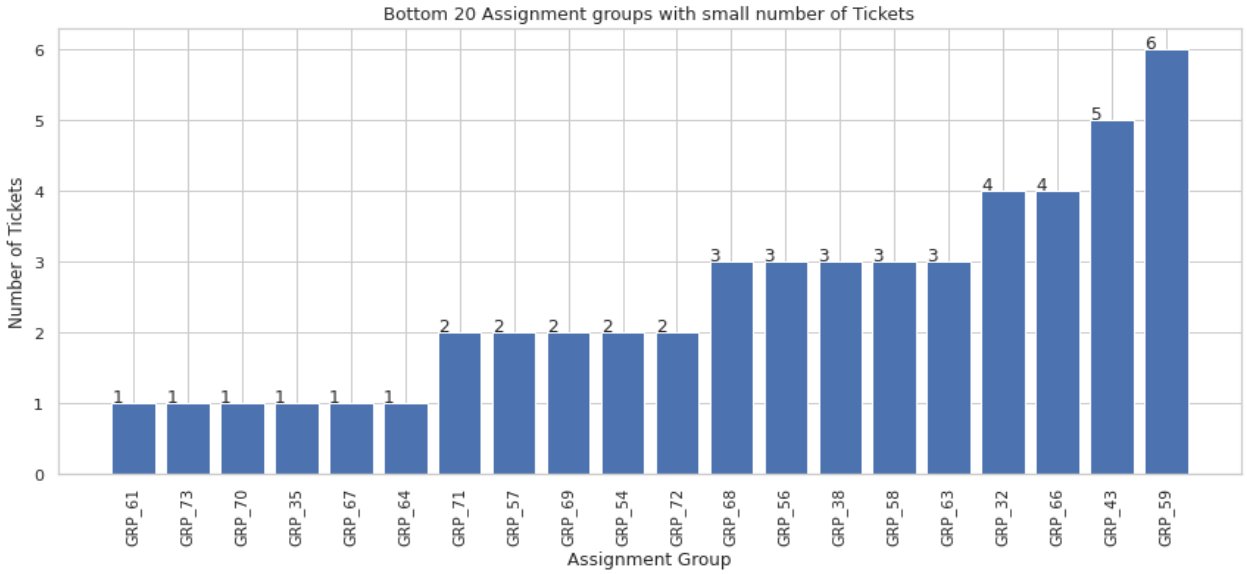
Following is the list of observation from EDA that we have performed on given dataset.

- There are 4 Columns and all are of string types.
- Total 8500 rows are available in given data set.
- Total 74 assignment group are available in given data set.
- Assignment group is our target to resolve the problem.
- We can classify this problem as **multi-class classification problem**.
- From the below screenshot we can say that 1 group (GRP_0) is having max number of ticket allocation while 6 group is having only 1 ticket allocated.

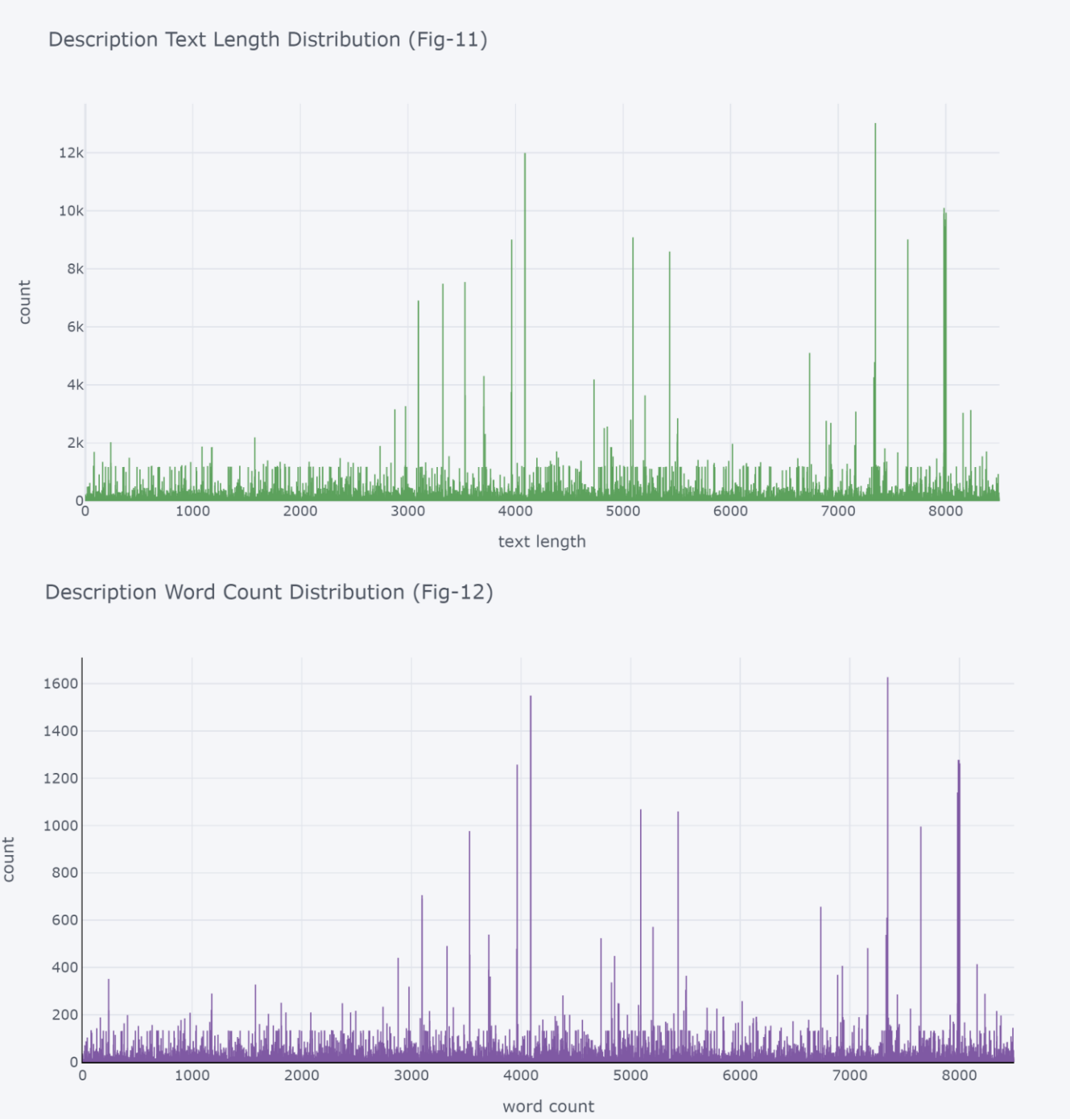


- Top 20 and bottom 20 group assignments.





- Word counts and Description text lengths for the dataset



greatlearning
Learning for Life

```
In [9]: data.isnull().sum()

Out[9]: Short description    8
        Description          1
        Caller               0
        Assignment group     0
        dtype: int64
```

- From the above screenshot we can say that there are 8 cells which are blank in Short Description and 1 cell in Description. We have handled by copying the short description to Description column for 1 missing value. For Short description, we replace null values with space " " character.

- Created a **Rule based engine** for allocation of tickets which have a deterministic rule to assign to target assignment group. This way those tickets don't need to go through AI/ML based prediction. Also for some target groups which has 15 or lesser tickets, it can be deterministically removed from the dataset and reduce imbalance.

```
In [18]: rules_applied_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8500 entries, 0 to 8499
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Short description   8500 non-null   object
 1   Description         8500 non-null   object
 2   Caller              8500 non-null   object
 3   Assignment group    8500 non-null   object
 4   pred_group          296 non-null    object
dtypes: object(5)
memory usage: 332.2+ KB

In [19]: rules_applied_df = rules_applied_df[(rules_applied_df['pred_group'].isna())]
         rules_applied_df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 8204 entries, 0 to 8499
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Short description   8204 non-null   object
 1   Description         8204 non-null   object
 2   Caller              8204 non-null   object
 3   Assignment group    8204 non-null   object
 4   pred_group          0 non-null      object
dtypes: object(5)
memory usage: 384.6+ KB
```

greatlearning
Learning for Life

Before applying the rules we are having 8500 record with allocation. After applying rules we are having 8200 records in our data set.

Create a rule based engine

In [15]:
```
#df_rules = pd.read_csv('/content/drive/MyDrive/Capstone/Rule_matrix.csv')
df_rules = pd.read_csv("Rule_matrix.csv")
```

In [16]:
```
def applyRules(datadf,rulesdf,Description,ShortDescription):
    datadf['pred_group'] = np.nan
    for i, row in rulesdf.iterrows():
        for j, row in datadf.iterrows():
            if pd.notna(datadf[ShortDescription][j]):
                if (('erp' in datadf[ShortDescription][j]) and (('EU_tool' in datadf[ShortDescription][j]))):
                        datadf['pred_group'][j] = 'GRP_25'
        for j, row in datadf.iterrows():
            if pd.notna(datadf[Description][j]):
                if (datadf[Description][j] == 'the'):
```

After applying rules we are having now 43 groups in our data set.

- From the given data set we have identified that there are some garbled text (called mojbake) which has been treated by using FTFY library. The outcome is text in Chinese language.

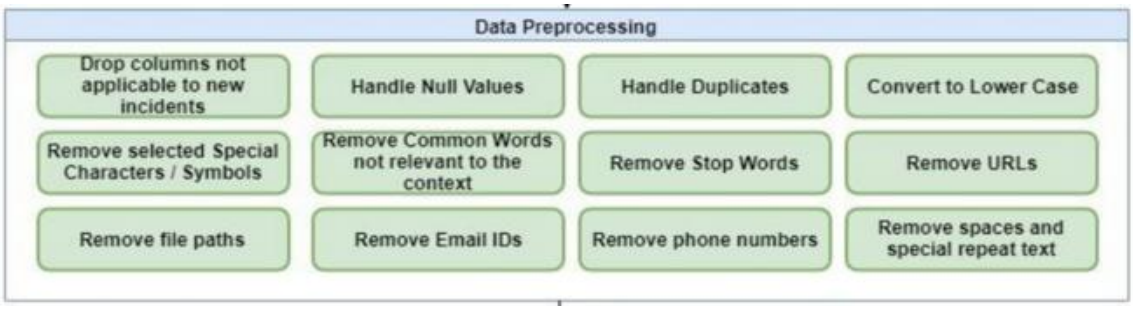
In [24]:
```
# Take an example of row# 8471 Short Desc and fix it
print('Grabled text: \033[1m%s\033[0m\nFixed text: \033[1m%s\033[0m' % (clean_data['New Description'][8471],
                                                                       fix_text(clean_data['New Description'][8471])))

# List all mojibakes defined in ftfy library
print('\nMojibake Symbol RegEx:\n', badness.MOJIBAKE_SYMBOL_RE.pattern)
```

```
Grabled text: to å°è´ºï¼Œæ–°å¹´å¿«ä¹ çˆ¶è¯´å¼€æœºå¼€ä¸å‡ºæ¥ çˆ¶è¯´å¼€æœºå¼€ä¸å‡ºæ¥
Fixed text: to 小贺,早上电脑开机开不出来 电脑开机开不出来

Mojibake Symbol RegEx:
 [ÂÃÎÏÐÑØÙÅĎĐŃŘŮ][-€ƒ„…†‡ˆ‰‹Œ•˜œŸ¡¢£¤¥¦§¨ª«¬¯°±²³µ¶·¸¹º¼½¾¿˘˙˝]|[ÂÃÎÏÐÑØÙÅĎĐŃŘŮ][›»‘’“”™]\w|×[-ƒ„…†‡ˆ‰‹Œ•˜œŸ¡¦§¨ª«¬¯°²³˘
˙›‘’“”™]|[¬√][ÄÅÇÉÑÖÜáàâäãåçéèêëíìîïñóúùûü†¢£§¶ß®©™≠ÆØ¥ªæø≤≥]|\w√[±∂]\w|◊|[ðđ][Ÿ]|â€|вЂ[љћ¦°¥™ќ“”]
```

greatlearning
Learning for Life

Cleaning Process Steps Undertaken

Below steps have been performed for initial pre-processing and cleanup of data.



- Merged Short Description & Description fields into new column (New Desciption) for further processing (instead of dropping the Short description column)

```
[42]
    def process(text_string):
        text=text_string.lower()
        text_string = ' '.join([w for w in text_string.split() if not date_validity(w)])
        text_string = re.sub(r"received from:",'',text_string)
        text_string = re.sub(r"from:",' ',text_string)
        text_string = re.sub(r"to:",' ',text_string)
        text_string = re.sub(r"subject:",' ',text_string)
        text_string = re.sub(r"sent:",' ',text_string)
        text_string = re.sub(r"ic:",' ',text_string)
        text_string = re.sub(r"cc:",' ',text_string)
        text_string = re.sub(r"bcc:",' ',text_string)
        text_string = re.sub(r'\S*@\S*\s?', '', text_string)
        text_string = re.sub(r'\d+','' ,text_string)
        text_string = re.sub(r'\n',' ',text_string)
        text_string = re.sub(r'#','', text_string)
        text_string = re.sub(r'&;?', 'and',text_string)
        text_string = re.sub(r'\&\w*;', '', text_string)
        text_string = re.sub(r'https?:\/\/.*\/\w*', '', text_string)
        #text_string= ''.join(c for c in text_string if c <= '\uFFFF')
        text_string = text_string.strip()
        #text_string = ' '.join(re.sub("[^\u0030-\u0039\u0041-\u005a\u0061-\u007a]", " ", text_string).split())
        text_string = re.sub(r"\s+[a-zA-Z]\s+", ' ', text_string)
```

greatlearning
Learning for Life

```
[44] clean_data
```

	Caller	Assignment group	New Description	Clean_Description
0	spxjnwir pjlcoqds	GRP_0	-verified user details.(employee# & manager na...	-verified user details.(employee and manager n...
1	hmjdrvpb komuaywn	GRP_0	\n\nreceived from: hmjdrvpb.komuaywn@gmail.com...	hello team, my meetings/skype meetings etc are...
2	eylqgodm ybqkwiam	GRP_0	\n\nreceived from: eylqgodm.ybqkwiam@gmail.com...	hi cannot log on to vpn best cant log in to vpn
3	xbkucsvz gcpydteq	GRP_0	unable to access hr_tool page unable to access...	unable to access hr_tool page unable to access...
4	owlgqjme qhcozdfx	GRP_0	skype error skype error	skype error skype error
...	...	...	...	...
8495	avglmrts vhqmtiua	GRP_29	\n\nreceived from: avglmrts.vhqmtiua@gmail.com...	good afternoon, am not receiving the emails th...
8496	rbozivdq gmlhrtvp	GRP_0	telephony_software issue telephony_software issue	telephony_software issue telephony_software issue

Below is the snapshot of outcome.

p	New Description	Clean_Description
	-verified user details.(employee# & manager na...	-verified user details.(employee and manager n...
	\n\nreceived from: hmjdrvpb.komuaywn@gmail.com...	hello team, my meetings/skype meetings etc are...
	\n\nreceived from: eylqgodm.ybqkwiam@gmail.com...	hi cannot log on to vpn best cant log in to vpn
	unable to access hr_tool page unable to access...	unable to access hr_tool page unable to access...
	skype error skype error	skype error skype error
	...	...
	\n\nreceived from: avglmrts.vhqmtiua@gmail.com...	good afternoon, am not receiving the emails th...
	telephony_software issue telephony_software issue	telephony_software issue telephony_software issue
	vip2: windows password reset for tifpdchb pedx...	vip: windows password reset for tifpdchb pedxr...
	i am unable to access the machine utilities to...	i am unable to access the machine utilities to...
	an mehreren pc`s lassen sich verschiedene prgr...	an mehreren pc`s lassen sich verschiedene prgr...

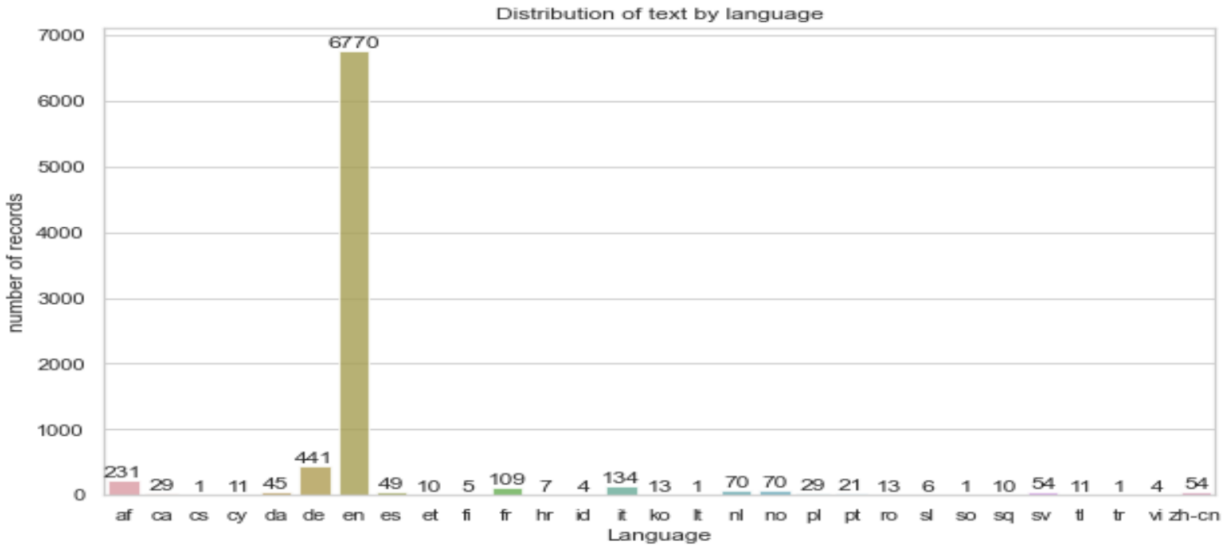
- Changed the case sensitivity of words to the lower case.
- Removed Hashtags and kept the words, Hyperlinks, URLs, HTML tags & non-ASCII symbols from merged fields.
- Process Date, Email strings in the description.
- Translating all foreign languages (German, Chinese) to English. Note* -

greatlearning
Learning for Life

Even though langdetect library says that we have multiple languages, visual inspection of those rows show that it incorrect. The description contains only German and Chinese language.

Language Translation

```
In [30]: !pip install langdetect

Looking in indexes: https://pypi.org/simple, https://pypi.nvidia.com/simple, https://urm.nvidia.com/artifactory/api/pypi/sw-col
ossus-pypi/simple
Requirement already satisfied: langdetect in c:\users\aroy\anaconda3\envs\myenv\lib\site-packages (1.0.8)
Requirement already satisfied: six in c:\users\aroy\anaconda3\envs\myenv\lib\site-packages (from langdetect) (1.15.0)

In [31]: from langdetect import detect

         def fn_lang_detect(df):
            try:
               return detect(df)
            except:
               return 'no'

         clean_data['language'] = clean_data['Clean_Description'].apply(fn_lang_detect)

In [32]: x = clean_data["language"].value_counts()
         x=x.sort_index()
         plt.figure(figsize=(10,6))
         ax= sns.barplot(x.index, x.values, alpha=0.8)
         plt.title("Distribution of text by language")
```



- We have tried to translate the data with google translate library with ‘goslate’ Python package but due to limit of processing not more than 100 records from one IP address every 24 hours we are not able to process complete dataset at one go. So to overcome from this problem we have

created batching of 100 records and process the same and create pkl file of each 100 records. Once we are able to do the translating all the data we have merged the all into one single files.

```
#!pip install goslate
```

```
[52] '''import goslate
     gs = goslate.Goslate()
     def translate(text_string):
       translation = gs.translate(text_string, 'en')
       return translation'''
```

```
In [42]: #with open('/content/drive/MyDrive/Capstone/Final_Translated_combined.pkl','rb') as f:
         with open('Final_Translated_combined.pkl','rb') as f:
             clean_data = pickle.load(f)
```

- We have also tried the method do the language translation by creating corpus of keywords and after that applied the same in data set. But google translate has a better language conversion accuracy.

greatlearning

We can see that most of the tickets are in english, followed by tickets in German language. We need to translate these into english.

```
[53] #german_data = pd.read_csv("/content/drive/MyDrive/Capstone/german.csv")
     #german_data = pd.read_csv('german.csv')

[54] #german_data

[55] #german_dictionary = german_data.to_dict(orient='records')

[56] '''
     def translate_function(text):
         translated_text = []
         text_split = text.split()
         for text in text_split:
             word_found = False
             for item in range(len(german_dictionary)):
                 if text == german_dictionary[item]["German"]:
                     translated_text.append(german_dictionary[item]["English"])
                     word_found = True
```

Below is the snapshot for language translation that we have done.

	Clean_Description	language	Translated Text
	早上开机后显示器不出图像。 显示器不亮	zh-cn	The display does not appear in the morning. Di...
机	prtSID_--文件无法打印到打印机,提示打印机错误。 文件无法打印到打印机,提示打印机错误。	zh-cn	The prtsid _- file cannot be printed to the pr...
解	客户提供的在线送货单生成系统打不开,需尽快解决 客户提供的在线系统打不开	zh-cn	The online delivery unit provided by the custo...
号,	进行采购时显示"找不到员工的数据,请通知系统管理员" erp无法进行采购(转给贺正平)	zh-cn	Show "Data from the employee, please notify th...

Data Augmentation

For data augmentation we have used nltk library to do the same.

greatlearning
Learning for Life

Data Augmentation

```
!pip3 install nltk
import nltk
nltk.download('wordnet')
nltk.download('punkt')
from nltk.corpus import wordnet
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.6/dist-packages (3.2.5)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from nltk) (1.15.0)
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

- First we wrote a function to find synonyms for a given word order.

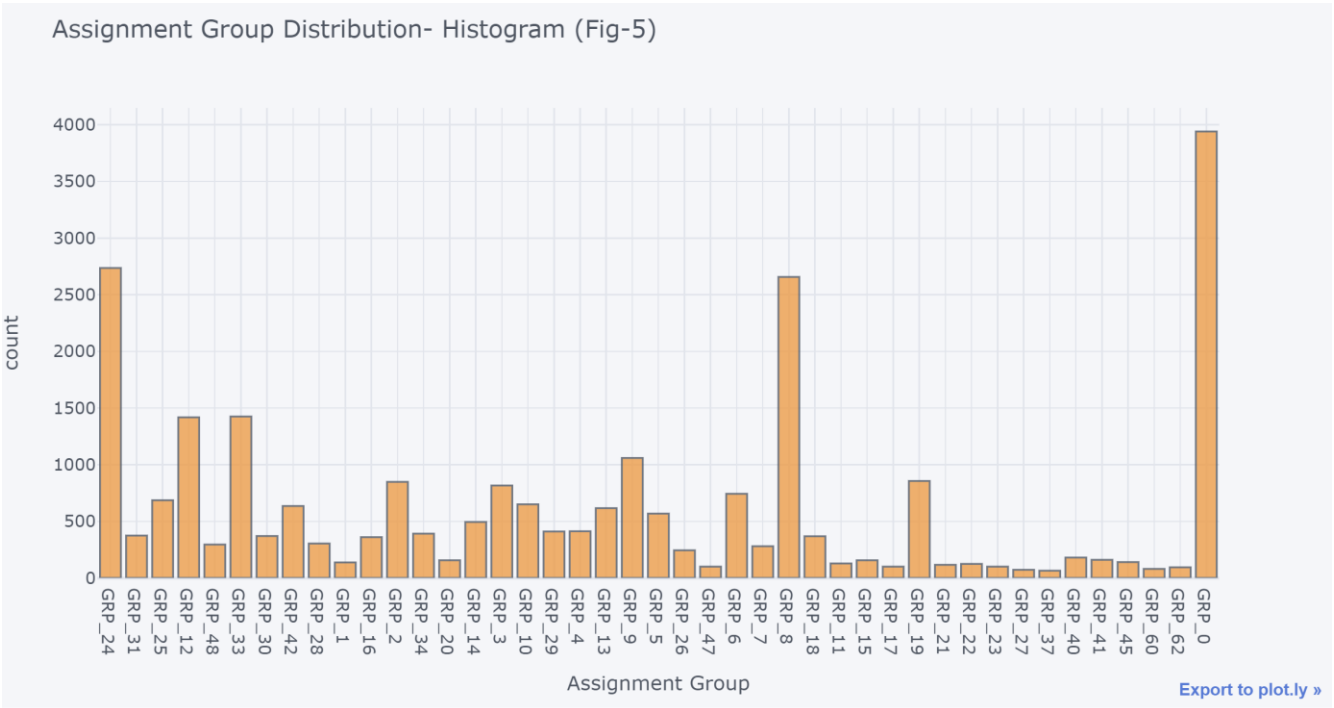
```
from collections import OrderedDict
from nltk.tokenize import word_tokenize
def find_synonyms(word):
  synonyms = []
  for synset in wordnet.synsets(word):
    for syn in synset.lemma_names():
      synonyms.append(syn)

  # using this to drop duplicates while maintaining word order (closest synonyms comes first)
  synonyms_without_duplicates = list(OrderedDict.fromkeys(synonyms))
  return synonyms_without_duplicates
```

- Created new set of sentences for data augmentation by replacing synonyms in original sentences to create new sentences.

greatlearning
Learning for Life

```
[69] def create_set_of_new_sentences(sentence, max_syn_per_word = 3):
       count = 0
       new_sentences = []
       for word in word_tokenize(sentence):
         if len(word)<=3 : continue
         for synonym in find_synonyms(word)[0:max_syn_per_word]:
           synonym = synonym.replace('_', ' ') #restore space character
           new_sentence = sentence.replace(word,synonym)
           if count <= 4:
             new_sentences.append(new_sentence)
             count += 1
       return new_sentences
```

- After Data Augmentation, we are able to increase the number of rows from 8000+ to 25000+ rows, below is the distribution. Note* - We excluded GRP_0 from the Data Augmentation process.



greatlearning
Learning for Life

- **Stop word removal & Lemmatize Text**:- For the purpose of analyzing text data and building NLP models, stop words need to removed.

```
import re
import string
nltk.download('stopwords')
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

stop_words = set(stopwords.words('english'))

processed_all_documents = list()

for desc in clean_data_result['Final_Text']:
    word_tokens = word_tokenize(desc)

    filtered_sentence = []

    # Removing Stopwords
    for w in word_tokens:
        if w not in stop_words:
            filtered_sentence.append(w)

    words = ' '.join(filtered_sentence)
    processed_all_documents.append(words)
```

- Performing lemmatization using the ‘Spacy’ Python library

```
# Need to run "python -m spacy download en" in anaconda prompt to avoid 'en' not found issue.
```

```
[100] import spacy

    nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
    allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']
    def lemmatize_text(text):
        doc = nlp(text)
        return ' '.join([token.lemma_ for token in doc])

    clean_data_result['Final_Text'] = clean_data_result['Final_Text'].apply(lemmatize_text)
```

```
clean_data_result
```

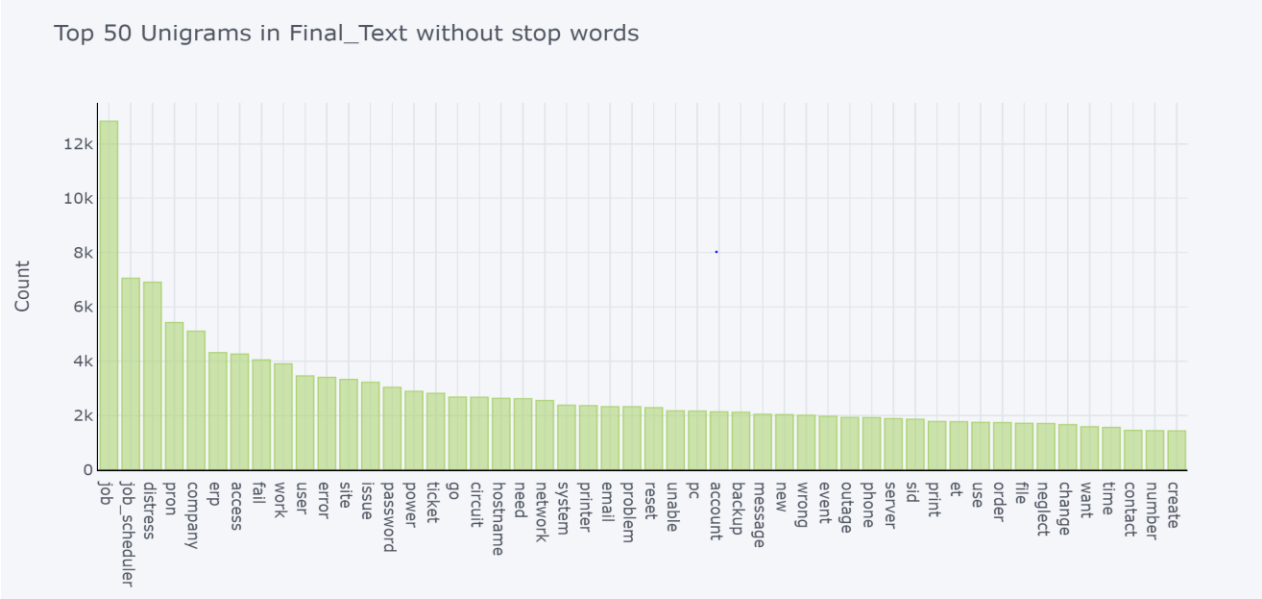
	Caller	Assignment group	language	Final_Text
1	tqrylspg ijzghqwy	GRP_12	de	message mistake re - connection Hostname no ac...

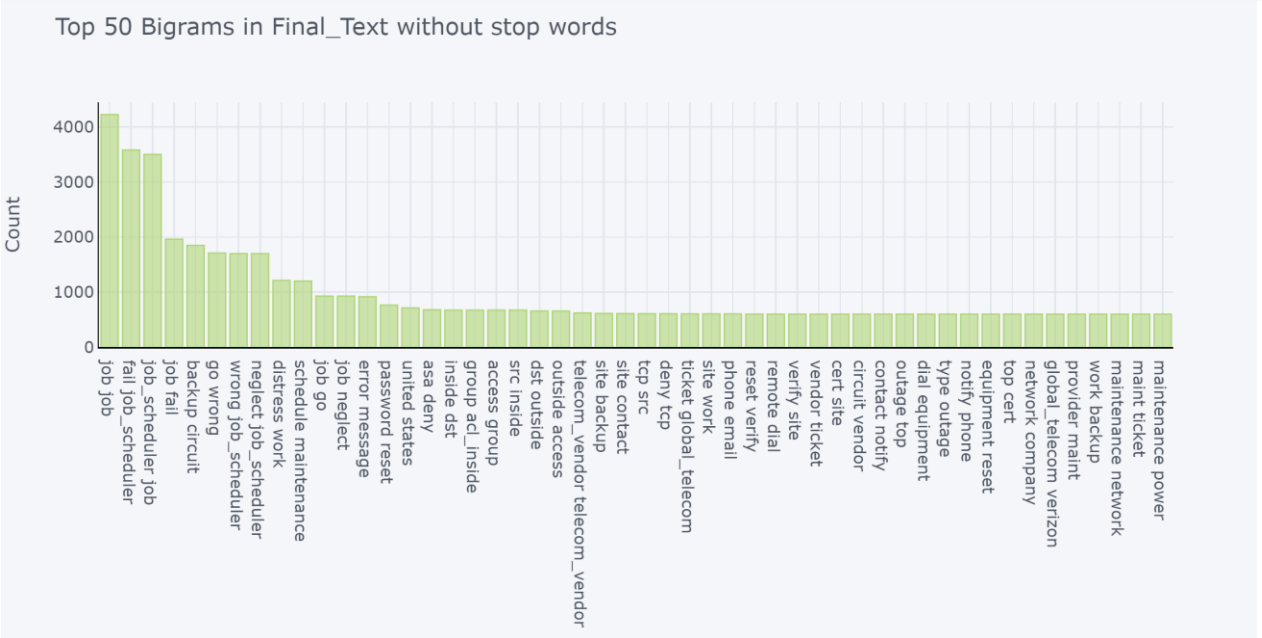
greatlearning
Learning for Life

Visual Analysis using N-Grams and Word Cloud

Analyzed words using N-Grams and Word Cloud to find out top occurring words in the dataset/per target group

- Top 50 **Unigrams** and **Bigrams** in dataset





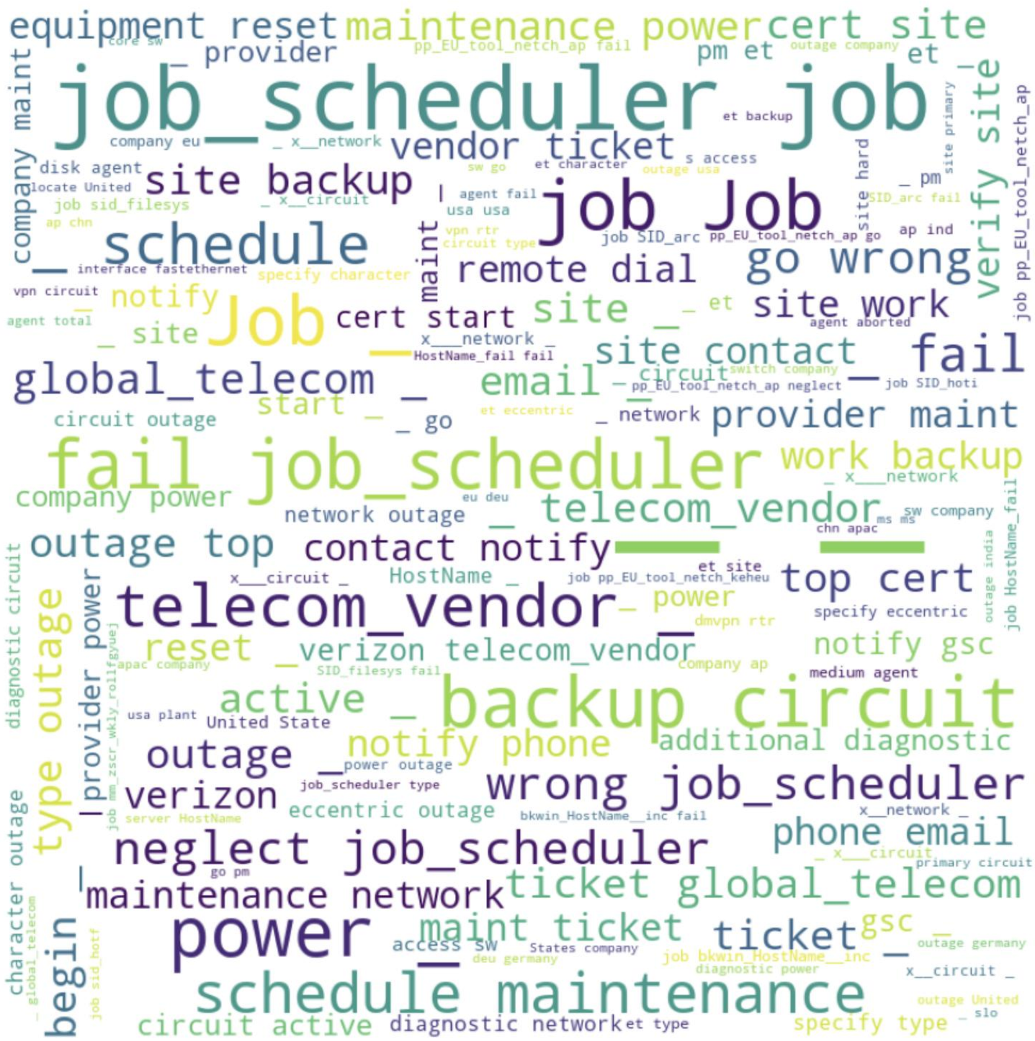
Word Cloud:-Visualizing this as a word cloud for top three groups that has got maximum records. A word cloud enables us to visualize the data as cluster of words and each words displayed in different font size based on the number of occurences of that word . Basically; the bolder and bigger the word show up in the visualization, it implies its more often it’s mentioned within a given text compared to other words in the cloud and therefore would be more important for us.

greatlearning
Learning for Life

In [88]:
```
# Word Cloud for all tickets assigned to GRP_0
generate_word_cloud(' '.join(clean_data[clean_data['Assignment group'] == 'GRP_0'].Final_Text.str.strip()))
```

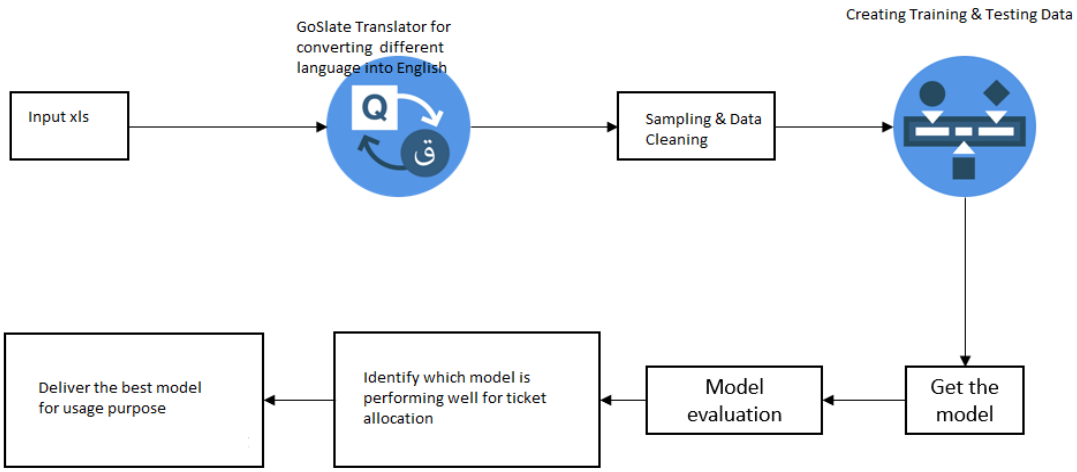
In [89]:

```
# Word Cloud for all tickets assigned to GRP_8
generate_word_cloud(' '.join(clean_data[clean_data['Assignment group'] == 'GRP_8'].Final_Text.str.strip()))
```



Deciding Models and Model Building

Overview of Solution



Preparing Dataframe for Model Building

- Created a targeted Categorical Column and assign unique label code to understand word labels.

```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
clean_data['Assignment group LabelEncoded']= label_encoder.fit_transform(clean_data['Assignment group'])

clean_data['Assignment group LabelEncoded'].unique()
```

```
array([ 4, 12, 23,  5, 41, 11, 17, 25, 18, 26, 24, 32, 21,  1,  8, 27, 13,
        6,  2, 22, 29, 42, 36, 19, 37, 40, 10,  3,  7,  9, 15, 30, 31, 33,
       35, 34, 14, 16, 20, 28, 38, 39,  0])
```

Feature Extraction

Bag of Words using CountVectorizer

Feature Extraction : Bag of Words using CountVectorizer

```
from sklearn.feature_extraction.text import CountVectorizer

CV = CountVectorizer(max_features=2000)

X_BoW = CV.fit_transform(clean_data['Final_Text']).toarray()
y = clean_data['Assignment group LabelEncoded']

print("Shape of Input Feature :",np.shape(X_BoW))
print("Shape of Target Feature :",np.shape(y))
```

```
Shape of Input Feature : (28208, 2000)
Shape of Target Feature : (28208,)
```

Algorithms Used:

The algorithms used fall under Machine Learning models and Deep Learning models and is briefed further below

Machine Learning Models

We have used following ML models to achieve the required accuracy and following is the output of each model.

We have split our data as per below approach.

- Splitting into Testing/Training data 70/30 ratio. We have used stratify flag as

```
[129] # Splitting Train Test
      from sklearn.model_selection import train_test_split

      X_train, X_test, y_train, y_test = train_test_split(X_BoW, y, test_size=0.3, random_state = 0, stratify=y)
      print('\033[1mShape of the training set:\033[0m', X_train.shape, X_test.shape)
      print('\033[1mShape of the test set:\033[0m', y_train.shape, y_test.shape)

      Shape of the training set: (19745, 2000) (8463, 2000)
      Shape of the test set: (19745,) (8463,)
```

  Target group which ensures that there is equal distribution of data among groups while creating testing and training data.

Also created a common function to run the models and check confusion matrix for F1 score, Precision, Recall.

greatlearning
Learning for Life

We have used Tf-idf transformer for our classification. The goal of using tf-idf is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus.

In [99]:

```
def run_classification(estimator, X_train, X_test, y_train, y_test, arch_name=None, pipelineRequired=True, isDeepModel=False
    # train the model
    clf = estimator

    if pipelineRequired :
        clf = Pipeline([('tfidf', TfidfTransformer()),
                        ('clf', estimator),
                        ])

    if isDeepModel :
        clf.fit(X_train, y_train, validation_data=(X_test, y_test),epochs=25, batch_size=128,verbose=1,callbacks=call_backs(
        # predict from the clasiffier
        y_pred = clf.predict(X_test)
        y_pred = np.argmax(y_pred, axis=1)
        y_train_pred = clf.predict(X_train)
        y_train_pred = np.argmax(y_train_pred, axis=1)
    else :
        clf.fit(X_train, y_train)
        # predict from the clasiffier
        y_pred = clf.predict(X_test)
        y_train_pred = clf.predict(X_train)

    print('Estimator:', clf)
    print('='*80)
    print('Training accuracy: %.2f%%' % (accuracy_score(y_train,y_train_pred) * 100))
    print('Testing accuracy: %.2f%%' % (accuracy_score(y_test, y_pred) * 100))
    print('='*80)
    print('Confusion matrix:\n %s' % (confusion_matrix(y_test, y_pred)))
    print('='*80)
    print('Classification report:\n %s' % (classification_report(y_test, y_pred)))
```

greatlearning
Learning for Life

Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model(a form of binary regression). We have achieved the accuracy of 70.92 % by using this method.

```
Estimator: Pipeline(memory=None,
         steps=[('tfidf',
                 TfidfTransformer(norm='l2', smooth_idf=True,
                                  sublinear_tf=False, use_idf=True)),
                ('clf',
                 LogisticRegression(C=1.0, class_weight=None, dual=False,
                                    fit_intercept=True, intercept_scaling=1,
                                    l1_ratio=None, max_iter=100,
                                    multi_class='auto', n_jobs=None,
                                    penalty='l2', random_state=None,
                                    solver='lbfgs', tol=0.0001, verbose=0,
                                    warm_start=False))],
         verbose=False)
================================================================================
Training accuracy: 70.92%
Testing accuracy: 66.07%
================================================================================
```

Naive Bayes Classifier

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is

independent of each other. By using this model we have achieved the accuracy of

```
In [392]: run_classification(MultinomialNB(), X_train, X_test, y_train, y_test)

Estimator: Pipeline(memory=None,
         steps=[('tfidf',
                 TfidfTransformer(norm='l2', smooth_idf=True,
                                  sublinear_tf=False, use_idf=True)),
                ('clf',
                 MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True))],
         verbose=False)
================================================================================
Training accuracy: 53.56%
Testing accuracy: 51.10%
================================================================================
```

53%.

K-Nearest Neighbor

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique. By using this method we have achieved the accuracy of 76.69%.

K-nearest Neighbor

```
[393]: run_classification(KNeighborsClassifier(), X_train, X_test, y_train, y_test)

Estimator: Pipeline(memory=None,
         steps=[('tfidf',
                 TfidfTransformer(norm='l2', smooth_idf=True,
                                  sublinear_tf=False, use_idf=True)),
                ('clf',
                 KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                      metric='minkowski', metric_params=None,
                                      n_jobs=None, n_neighbors=5, p=2,
                                      weights='uniform'))],
         verbose=False)
================================================================================
Training accuracy: 76.69%
Testing accuracy: 69.79%
================================================================================
```

greatlearning
Learning for Life

Support Vector Machine

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to

categorize new text. So we're working on a text classification problem . By using this we achieved accuracy of 78.43%.

```
In [394]: run_classification(LinearSVC(), X_train, X_test, y_train, y_test)

Estimator: Pipeline(memory=None,
         steps=[('tfidf',
                 TfidfTransformer(norm='l2', smooth_idf=True,
                                  sublinear_tf=False, use_idf=True)),
                ('clf',
                 LinearSVC(C=1.0, class_weight=None, dual=True,
                           fit_intercept=True, intercept_scaling=1,
                           loss='squared_hinge', max_iter=1000,
                           multi_class='ovr', penalty='l2', random_state=None,
                           tol=0.0001, verbose=0))],
         verbose=False)
================================================================================
Training accuracy: 78.43%
Testing accuracy: 72.61%
--------------------------------------------------------------------------------
```

Decision Tree

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). By using this we achieved accuracy of 82.08%.

```
In [395]: run_classification(DecisionTreeClassifier(), X_train, X_test, y_train, y_test)

Estimator: Pipeline(memory=None,
         steps=[('tfidf',
                 TfidfTransformer(norm='l2', smooth_idf=True,
                                  sublinear_tf=False, use_idf=True)),
                ('clf',
                 DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
                                        criterion='gini', max_depth=None,
                                        max_features=None, max_leaf_nodes=None,
                                        min_impurity_decrease=0.0,
                                        min_impurity_split=None,
                                        min_samples_leaf=1, min_samples_split=2,
                                        min_weight_fraction_leaf=0.0,
                                        presort='deprecated', random_state=None,
                                        splitter='best'))],
         verbose=False)
================================================================================
Training accuracy: 82.08%
Testing accuracy: 72.13%
================================================================================
```

Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean/average prediction of the individual trees.By using this we achieved accuracy of 82.08%.

greatlearning
Learning for Life

```
In [396]: run_classification(RandomForestClassifier(n_estimators=100), X_train, X_test, y_train, y_te

          Estimator: Pipeline(memory=None,
                   steps=[('tfidf',
                           TfidfTransformer(norm='l2', smooth_idf=True,
                                            sublinear_tf=False, use_idf=True)),
                          ('clf',
                           RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                                  class_weight=None, criterion='gini',
                                                  max_depth=None, max_features='auto',
                                                  max_leaf_nodes=None, max_samples=None,
                                                  min_impurity_decrease=0.0,
                                                  min_impurity_split=None,
                                                  min_samples_leaf=1, min_samples_split=2,
                                                  min_weight_fraction_leaf=0.0,
                                                  n_estimators=100, n_jobs=None,
                                                  oob_score=False, random_state=None,
                                                  verbose=0, warm_start=False))],
                   verbose=False)
          ================================================================================
          Training accuracy: 82.08%
          Testing accuracy: 76.70%
          --------------------------------------------------------------------------------
```

Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. By using this model we have achieved the accuracy of 76.33%.

```
                                                  verbose=0, warm_start=False))],
                   verbose=False)
================================================================================
Training accuracy: 76.33%
Testing accuracy: 66.91%
================================================================================
Confusion matrix:
 [[914   1   4 ...   5   1   9]
 [  0  30   0 ...   0   6   0]
 [  0   0 146 ...   0  13   4]
 ...
 [  2   0   0 ...  96   0   0]
 [  6   0   4 ...   0 536   1]
 [  4   0   0 ...   0  88  93]]
================================================================================
Classification report:
```

greatlearning
Learning for Life

XGBoost

Beauty of this powerful algorithm lies in its scalability, which drives fast learning through parallel and distributed computing and offers efficient memory usage. By using this model we have achieved the accuracy of 76.33%.

```
        verbose=False)
==========================================================================
Training accuracy: 69.08%
Testing accuracy: 61.90%
==========================================================================
Confusion matrix:
 [[922   0   4 ...   2   0   8]
 [  2  26   0 ...   0   3   0]
 [  6   0 114 ...   0  14   4]
 ...
 [  9   0   0 ...  83   0   0]
 [  9   1   4 ...   0 524   0]
 [  9   0   0 ...   0  88  81]]
==========================================================================
Classification report:
```

Bagging

Bootstrap aggregating, also called bagging (from bootstrap aggregating), is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid over fitting. By using this model, we have achieved the accuracy of 81.87%.

```
        verbose=False)
================================================================================
Training accuracy: 81.87%
Testing accuracy: 72.65%
================================================================================
Confusion matrix:
 [[994   0   2 ...   2   7   2]
 [  0  29   0 ...   0   3   0]
 [  2   0 147 ...   0  13   0]
 ...
 [  1   0   0 ...  92   0   0]
 [  1   0   4 ...   0 567   2]
 [  2   0   0 ...   0  97  95]]
================================================================================
Classification report:
```

Stacking

Stacked Generalization or “Stacking” for short is an ensemble machine learning algorithm. It involves combining the predictions from multiple machine learning models on the same dataset, like bagging and boosting. By using this model, we have achieved the accuracy of 79.17%

```
================================================================================
Training accuracy: 79.17%
Testing accuracy: 73.41%
================================================================================
Confusion matrix:
 [[1047    0    4 ...    0    0    0]
 [   0   33    0 ...    0    4    0]
 [   1    0  147 ...    0   13    0]
 ...
 [   1    0    0 ...   90    0    0]
 [   1    0    4 ...    0  551    0]
 [   8    0    0 ...    0   88   92]]
================================================================================
```

Deep Learning Models

For DL models, we reload the dataset **without** Stop Words removal and

Lemmatization. This is important step.
Also created a checkpoint function for Early stopping and Model checkpoint creation

Create checkpoints function

```
In [120]: #Path where you want to save the weights, model and checkpoints
          model_path = "Weights/"
          %mkdir Weights

          # Define model callbacks
          def call_backs(name):
              early_stopping = EarlyStopping(monitor='val_loss', mode='min', min_delta=0.01, patience=3)
              model_checkpoint =  ModelCheckpoint(model_path + name + '_epoch{epoch:02d}_loss{val_loss:.4f}.h5',
                                                  monitor='val_loss',
                                                  verbose=1,
                                                  save_best_only=True,
                                                  save_weights_only=False,
                                                  mode='min',
                                                  period=1)
              return [model_checkpoint, early_stopping]
```

We have the below DL models explored.

DNN

This command builds a feed forward multi layer neural network that is trained with a set of labeled data in order to perform classification on similar, unlabeled data . By using this model we have achieved the accuracy of 80.73%.

greatlearning
Learning for Life

In [121]:
```
# Function to build Neural Network
def Build_Model_DNN_Text(shape, nClasses, dropout=0.3):
    """
    buildModel_DNN_Tex(shape, nClasses,dropout)
    Build Deep neural networks Model for text classification
    Shape is input feature space
    nClasses is number of classes
    """
    model = Sequential()
    node = 512 # number of nodes
    nLayers = 4 # number of  hidden layer
    model.add(Dense(node,input_dim=shape,activation='relu'))
    model.add(Dropout(dropout))
    for i in range(0,nLayers):
        model.add(Dense(node,input_dim=node,activation='relu'))
        model.add(Dropout(dropout))
        model.add(BatchNormalization())
    model.add(Dense(nClasses, activation='softmax'))
    model.compile(loss='sparse_categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    print(model.summary())
    return model
```

In [122]:
```
Tfidf_vect = TfidfVectorizer(max_features=2000)
Tfidf_vect.fit(clean_data_DL.Final_Text.astype(str))
X_train_tfidf = Tfidf_vect.transform(X_train)
X_test_tfidf = Tfidf_vect.transform(X_test)

# Instantiate the network
model_DNN = Build_Model_DNN_Text(X_train_tfidf.shape[1], 43)
```

```
Epoch 00015: val_loss did not improve from 0.68931
Estimator: <keras.engine.sequential.Sequential object at 0x7fca7fcca898>
================================================================================
Training accuracy: 80.73%
Testing accuracy: 74.73%
================================================================================
Confusion matrix:
 [[1021    0    2 ...    0    6    4]
 [   0   28    0 ...    0    3    0]
 [   0    0  152 ...    0   13    0]
 ...
 [   1    0    0 ...   95    0    0]
 [   0    0    4 ...    0  565    0]
 [   6    0    0 ...    0   98   96]]
================================================================================
```

LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. By using this model we have achieved the

accuracy of 76.05%. For LSTM and CNN model, we are extracting **Glove embeddings** with 200 dimensions for Word2Vec conversion on the model

Extract Glove Embeddings

In [ ]:
```
#download the glove embedding zip file from http://nlp.stanford.edu/data/wordvecs/glove.6B.zip
from zipfile import ZipFile
# Check if it is already extracted else Open the zipped file as readonly
if not os.path.isfile('glove.6B/glove.6B.200d.txt'):
    glove_embeddings = 'glove.6B.zip'
    #glove_embeddings = '/content/drive/MyDrive/Capstone/glove.6B.zip'
    with ZipFile(glove_embeddings, 'r') as archive:
        archive.extractall('glove.6B')

# List the files under extracted folder
os.listdir('glove.6B')
```

RNN with LSTM networks

In [ ]:
```
EMBEDDING_DIM = 200
#gloveFileName = 'glove.6B/glove.6B.100d.txt'
gloveFileName = '/content/glove.6B/glove.6B.200d.txt'

from keras.models import Sequential
from keras.layers import Dense, LSTM, TimeDistributed, Activation
from keras.layers import Flatten, Permute, merge, Input
from keras.layers import Embedding
from keras.models import Model
from keras.layers import Input, Dense, multiply, concatenate, Dropout
from keras.layers import GRU, Bidirectional


def Build_Model_LTSM_Text(word_index, embeddings_matrix, nclasses):
    kernel_size = 2
    filters = 256
    pool_size = 2
    gru_node = 256

    model = Sequential()
    model.add(Embedding(len(word_index) + 1,
                                EMBEDDING_DIM,
                                weights=[embeddings_matrix],
                                input_length=MAX_SEQUENCE_LENGTH,
                                trainable=True))
    model.add(Dropout(0.25))
    model.add(Conv1D(filters, kernel_size, activation='relu'))
    model.add(MaxPooling1D(pool_size=pool_size))
    model.add(Conv1D(filters, kernel_size, activation='relu'))
    model.add(MaxPooling1D(pool_size=pool_size))
    model.add(Conv1D(filters, kernel_size, activation='relu'))
    model.add(MaxPooling1D(pool_size=pool_size))
    model.add(Conv1D(filters, kernel_size, activation='relu'))
    model.add(MaxPooling1D(pool_size=pool_size))
    model.add(Bidirectional(LSTM(gru_node, return_sequences=True, recurrent_dropout=0.2)))
    model.add(Bidirectional(LSTM(gru_node, return_sequences=True, recurrent_dropout=0.2)))
    model.add(Bidirectional(LSTM(gru_node, return_sequences=True, recurrent_dropout=0.2)))
    model.add(Bidirectional(LSTM(gru_node, recurrent_dropout=0.2)))
    model.add(Dense(1024,activation='relu'))
    model.add(Dense(nclasses))
    model.add(Activation('softmax'))
    model.compile(loss='sparse_categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])

    print(model.summary())
    return model
```

greatlearning

```
Epoch 00025: val_loss did not improve from 1.02446
Estimator: <keras.engine.sequential.Sequential object at 0x7fca6171ca58>
================================================================================
Training accuracy: 76.05%
Testing accuracy: 68.24%
================================================================================
Confusion matrix:
 [[876   0   2 ...  15   0   5]
 [  0  24   0 ...   0   3   4]
 [  0   0 138 ...   0   1  11]
 ...
```

Convolutional Neural Networks (CNN)

TensorFlow is an open-source software library created by Google for numerical computation using data flow graphs. By using this model we have achieved the accuracy of 65.63%

Convolutional Neural Networks (CNN)

In [ ]:

```
gloveFileName = 'glove.6B/glove.6B.200d.txt'
#gloveFileName = '/content/glove.6B/glove.6B.200d.txt'
MAX_SEQUENCE_LENGTH = 500
EMBEDDING_DIM=200
MAX_NB_WORDS=75000

# Function to generate Embedding
def loadData_Tokenizer(X_train, X_test,filename):
    np.random.seed(7)
    text = np.concatenate((X_train, X_test), axis=0)
    text = np.array(text)
    tokenizer = Tokenizer(num_words=MAX_NB_WORDS)
    tokenizer.fit_on_texts(text)
    sequences = tokenizer.texts_to_sequences(text)
    word_index = tokenizer.word_index
    text = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)
    print('Found %s unique tokens.' % len(word_index))
    indices = np.arange(text.shape[0])
    # np.random.shuffle(indices)
    text = text[indices]
    print(text.shape)
    X_train = text[0:len(X_train), ]
    X_test = text[len(X_train):, ]
    embeddings_index = {}
    f = open(filename, encoding="utf8")
    for line in f:
        values = line.split()
        word = values[0]
        try:
            coefs = np.asarray(values[1:], dtype='float32')
        except:
            pass
        embeddings_index[word] = coefs
    f.close()
    print('Total %s word vectors.' % len(embeddings_index))
    return (X_train, X_test, word_index,embeddings_index)


embedding_matrix = []

def buildEmbed_matrices(word_index,embedding_dim):
    embedding_matrix = np.random.random((len(word_index) + 1, embedding_dim))
    for word, i in word_index.items():
        embedding_vector = embeddings_index.get(word)
        if embedding_vector is not None:
            # words not found in embedding index will be all-zeros.
            if len(embedding_matrix[i]) !=len(embedding_vector):
                print("could not broadcast input array from shape",str(len(embedding_matrix[i])), "into shape",str(len(embedc
                      " Please make sure your"" EMBEDDING_DIM is equal to embedding_vector file ,GloVe,")
                exit(1)
            embedding_matrix[i] = embedding_vector
    return embedding_matrix
```

```
Epoch 00025: val_loss did not improve from 1.32720
Estimator: <keras.engine.training.Model object at 0x7fca7f422f98>
================================================================================
Training accuracy: 65.63%
Testing accuracy: 60.33%
================================================================================
Confusion matrix:
 [[1002    0    1 ...    0    0    0]
 [   0    2    0 ...    0    7    0]
 [   0    0  103 ...    0   12    0]
 ...
 [   0    0    0 ...   22    0    0]
 [  17    2    4 ...    0  545    0]
 [   9    2    0 ...    0   76   96]]
================================================================================
Classification report:
```

greatlearning
Learning for Life

Model Performance

In order to train our models, we used different parameters (F1 score & Result) to overcome with which model suits. I have added remarks columns to brief about the models.

Accuracy of Different Models

Model	Sub-Category	Result	F1 Accuracy
Traditional Machine Learning Models			
	Logistic Regression	70.92%	62.00%
	Naive Bayes Classifier	53.56%	47.00%
	K- Nearest Neighbor	76.69%	65.00%
	SVM	78.43%	70.00%
	Decision Tree	82.08%	70.00%
	Random Forest	82.08%	75.00%
	Gradient Boosting	76.33%	67.00%
	XGBoosting	69.08%	61.00%
	Bagging	81.87%	72.00%
	Stacking	79.17%	73.00%
Deep Learning Models			
	DNN	80.73%	74.00%
	LSTM	76.05%	68.00%
	CNN	65.63%	58.00%

Summary

Based on above parameters, we can say that in the current stage; Random Forest performance is best for machine learning language while for advance deep learning model DNN has given better scores as seen from the above table. **In the next phase we look forward to do hyper parameter tuning and model tuning to achieve better F1 score and accuracy.**

Code Snippet

https://github.com/guptapawan227/Capstone_AIML

Finalized Results

Results will be added in this report once we will complete the project before final submission. Current status of model results is noted under section: Summary

Link to Code & References

https://github.com/guptapawan227/Capstone_AIML