In [ ]:
```
%tensorflow_version 2.x
import tensorflow
tensorflow.__version__
```

Out[ ]:
```
'2.3.0'
```

In [ ]:

```
# Install the required libraries
!pip install numpy requests nlpaug
!pip install googletrans
!pip install fuzzywuzzy
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages
(1.18.5)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packag
es (2.23.0)
Collecting nlpaug
  Downloading https://files.pythonhosted.org/packages/ed/c1/aa7f3ece5ecf6eac7fb
fd2a419818e89433d1e7cfd78efdb05a1686b510d/nlpaug-1.0.1-py3-none-any.whl (376kB)
     |████████████████████████████████| 378kB 2.8MB/s
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/d
ist-packages (from requests) (2020.6.20)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/
local/lib/python3.6/dist-packages (from requests) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/di
st-packages (from requests) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-pa
ckages (from requests) (2.10)
Installing collected packages: nlpaug
Successfully installed nlpaug-1.0.1
Collecting googletrans
  Downloading https://files.pythonhosted.org/packages/71/3a/3b19effdd4c03958b90
f40fe01c93de6d5280e03843cc5adf6956bfc9512/googletrans-3.0.0.tar.gz
Collecting httpx==0.13.3
  Downloading https://files.pythonhosted.org/packages/54/b4/698b284c6aed4d7c2b4
fe3ba5df1fcf6093612423797e76fbb24890dd22f/httpx-0.13.3-py3-none-any.whl (55kB)
     |████████████████████████████████| 61kB 2.9MB/s
Collecting rfc3986<2,>=1.3
  Downloading https://files.pythonhosted.org/packages/78/be/7b8b99fd74ff5684225
f50dd0e865393d2265656ef3b4ba9eaaaffe622b8/rfc3986-1.4.0-py2.py3-none-any.whl
Collecting httpcore==0.9.*
  Downloading https://files.pythonhosted.org/packages/dd/d5/e4ff9318693ac6101a2
095e580908b591838c6f33df8d3ee8dd953ba96a8/httpcore-0.9.1-py3-none-any.whl (42k
B)
     |████████████████████████████████| 51kB 4.2MB/s
Collecting sniffio
  Downloading https://files.pythonhosted.org/packages/b3/82/4bd4b7d9c0d1dc0fbfb
c2a1e00138e7f3ab85bc239358fe9b78aa2ab586d/sniffio-1.1.0-py3-none-any.whl
Collecting hstspreload
  Downloading https://files.pythonhosted.org/packages/f0/16/59b51f8e640f16acad2
c0e101fa1e55a87bfd80cff23f77ab23cb8927541/hstspreload-2020.10.6-py3-none-any.wh
l (965kB)
     |████████████████████████████████| 972kB 8.6MB/s
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-package
s (from httpx==0.13.3->googletrans) (2020.6.20)
Requirement already satisfied: idna==2.* in /usr/local/lib/python3.6/dist-packa
ges (from httpx==0.13.3->googletrans) (2.10)
Requirement already satisfied: chardet==3.* in /usr/local/lib/python3.6/dist-pa
ckages (from httpx==0.13.3->googletrans) (3.0.4)
Collecting h2==3.*
  Downloading https://files.pythonhosted.org/packages/25/de/da019bcc539eeab02f6
d45836f23858ac467f584bfec7a526ef200242afe/h2-3.2.0-py2.py3-none-any.whl (65kB)
     |████████████████████████████████| 71kB 8.6MB/s
Collecting h11<0.10,>=0.8
  Downloading https://files.pythonhosted.org/packages/5a/fd/3dad730b0f95e78aeeb
742f96fa7bbecbdd56a58e405d3da440d5bfb90c6/h11-0.9.0-py2.py3-none-any.whl (53kB)
     |████████████████████████████████| 61kB 6.6MB/s
Collecting contextvars>=2.1; python_version < "3.7"
  Downloading https://files.pythonhosted.org/packages/83/96/55b82d9f13763be9d67
2622e1b8106c85acb83edd7cc2fa5bc67cd9877e9/contextvars-2.4.tar.gz
Collecting hyperframe<6,>=5.2.0
  Downloading https://files.pythonhosted.org/packages/19/0c/bf88182bcb5dce3094e
2f3e4fe20db28a9928cb7bd5b08024030e4b140db/hyperframe-5.2.0-py2.py3-none-any.whl
```

```
Collecting hpack<4,>=3.0
  Downloading https://files.pythonhosted.org/packages/8a/cc/e53517f4a1e13f74776
ca93271caef378dadec14d71c61c949d759d3db69/hpack-3.0.0-py2.py3-none-any.whl
Collecting immutables>=0.9
  Downloading https://files.pythonhosted.org/packages/99/e0/ea6fd4697120327d267
73b5a84853f897a68e33d3f9376b00a8ff96e4f63/immutables-0.14-cp36-cp36m-manylinux1
_x86_64.whl (98kB)
    |████████████████████████████████| 102kB 8.9MB/s
Building wheels for collected packages: googletrans, contextvars
  Building wheel for googletrans (setup.py) ... done
  Created wheel for googletrans: filename=googletrans-3.0.0-cp36-none-any.whl s
ize=15736 sha256=69c410d048b41f69bd527bdeb0d703449e8ba0671de42914bb6cdd2d90b43c
c2
  Stored in directory: /root/.cache/pip/wheels/28/1a/a7/eaf4d7a3417a0c65796c547
cff4deb6d79c7d14c2abd29273e
  Building wheel for contextvars (setup.py) ... done
  Created wheel for contextvars: filename=contextvars-2.4-cp36-none-any.whl siz
e=7666 sha256=2282d4903f23e9fdb6a718d8ba8da0c575ba5641f5fa5f0261345252e0123a13
  Stored in directory: /root/.cache/pip/wheels/a5/7d/68/1ebae2668bda2228686e3c1
cf16f2c2384cea6e9334ad5f6de
Successfully built googletrans contextvars
Installing collected packages: rfc3986, immutables, contextvars, sniffio, hyper
frame, hpack, h2, h11, httpcore, hstspreload, httpx, googletrans
Successfully installed contextvars-2.4 googletrans-3.0.0 h11-0.9.0 h2-3.2.0 hpa
ck-3.0.0 hstspreload-2020.10.6 httpcore-0.9.1 httpx-0.13.3 hyperframe-5.2.0 imm
utables-0.14 rfc3986-1.4.0 sniffio-1.1.0
Collecting fuzzywuzzy
  Downloading https://files.pythonhosted.org/packages/43/ff/74f23998ad2f93b945c
0309f825be92e04e0348e062026998b5eefef4c33/fuzzywuzzy-0.18.0-py2.py3-none-any.wh
l
Installing collected packages: fuzzywuzzy
Successfully installed fuzzywuzzy-0.18.0
```

In [ ]:
```
import spacy
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer
import nlpaug.augmenter.char as nac
import nlpaug.augmenter.word as naw
import nlpaug.augmenter.sentence as nas
from googletrans import Translator
import time
import re
import string
from sklearn.preprocessing import LabelEncoder
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, Flatten
from tensorflow.keras import regularizers, optimizers
from tensorflow.keras.initializers import Constant
import pickle
from tqdm import tqdm
import os
from sklearn.preprocessing import LabelEncoder
from tqdm import tqdm
import numpy as np
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from bs4 import BeautifulSoup
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score,recall_score,accuracy_score
from IPython.display import display, HTML
```

In [ ]:
```
nlp = spacy.load('en_core_web_sm')
```

In [ ]:
```
# Mounting Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```
import os
%cd "/content/drive/My Drive/Data/"
!pwd
```

```
/content/drive/My Drive/Data
/content/drive/My Drive/Data
```

In [ ]:
```
import pandas as pd
raw_data = pd.read_excel('/content/drive/My Drive/Data/input_data.xlsx')
data = raw_data.copy()
data.head()
```

Out[ ]:

| | 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 |

# Data Preprocessing

In [ ]:
```
#Caller is unique and encrypted column, sicnce it will not add any value to our model we can drop it.
data = data.drop(columns='Caller')
data['Description'] = data['Description'].replace(to_replace=[r"\\t|\\n|\\r", "\t|\n|\r"], value=[" "," "], regex=True)
data['Short description'] = data['Short description'].replace(to_replace=[r"\\t|\\n|\\r", "\t|\n|\r"], value=[" "," "], regex=True)
```

In [ ]:
```
duplicate = data[data.duplicated(keep=False)]
print('Preview of some dulplicate values in data\n')
display(duplicate.sort_values(by=['Short description']).head(10))
duplicate = data[data.duplicated(keep='first')]
print(f'\n\nTotal number of duplicate rows in data {duplicate.shape[0]}')
print(f'Duplicate rows droped from data.')
print(f'Number of rows in data before dropping duplicates rows: {data.shape[0]}
')
data = data.drop_duplicates( keep='first')
print(f'Number of rows in data after dropping duplicates rows: {data.shape[0]}'
)
data = data.reset_index(drop=True)
```

```
Preview of some dulplicate values in data
```

| | Short description | Description | Assignment group |
|---|---|---|---|
| **899** | HostName_1030 is currently experiencing high c... | HostName_1030 is currently experiencing high c... | GRP_12 |
| **474** | HostName_1030 is currently experiencing high c... | HostName_1030 is currently experiencing high c... | GRP_12 |
| **2701** | account got locked | account got locked | GRP_0 |
| **2387** | account got locked | account got locked | GRP_0 |
| **1988** | account got locked | account got locked | GRP_0 |
| **7058** | account is locked | account is locked | GRP_0 |
| **7170** | account is locked | account is locked | GRP_0 |
| **4688** | account locked | account locked | GRP_0 |
| **3800** | account locked | account locked | GRP_0 |
| **3396** | account locked | account locked | GRP_0 |

```
Total number of duplicate rows in data 591
Duplicate rows droped from data.
Number of rows in data before dropping duplicates rows: 8500
Number of rows in data after dropping duplicates rows: 7909
```

In [ ]:
```
data['Description'] = data['Description'].astype(str)
data['Short description'] = data['Short description'].astype(str)
```

In [ ]:

```
from fuzzywuzzy import fuzz
from fuzzywuzzy import process

data['short full desc similarity'] = 0
for index in data.index:
   data['short full desc similarity'][index] = fuzz.partial_ratio(data['Short description'][index].lower(),data['Description'][index].lower())
print(' \nAdded column for similatiry score between short and long description')
display(data.head())
for index in data.index:
  if data['short full desc similarity'][index] < 100 :
    data['Description'][index] = data['Short description'][index] + ' ' +  data['Description'][index]
print('\n\n\nConcatnated short description to description if similarity is less then 100')
display(data.head())
print('\n\n\nDropped Description and short description columns')
data = data.drop(columns=[ 'Short description' , 'short full desc similarity'])
display(data.head())
```

Added column for similatiry score between short and long description

| | Short description | Description | Assignment group | short full desc similarity |
|---|---|---|---|---|
| **0** | login issue | -verified user details.(employee# & manager na... | GRP_0 | 27 |
| **1** | outlook | received from: hmjdrvpb.komuaywn@gmail.com... | GRP_0 | 100 |
| **2** | cant log in to vpn | received from: eylqgodm.ybqkwiam@gmail.com... | GRP_0 | 83 |
| **3** | unable to access hr_tool page | unable to access hr_tool page | GRP_0 | 100 |
| **4** | skype error | skype error | GRP_0 | 100 |

Concatnated short description to description if similarity is less then 100

| | Short description | Description | Assignment group | short full desc similarity |
|---|---|---|---|---|
| **0** | login issue | login issue -verified user details. (employee# ... | GRP_0 | 27 |
| **1** | outlook | received from: hmjdrvpb.komuaywn@gmail.com... | GRP_0 | 100 |
| **2** | cant log in to vpn | cant log in to vpn received from: eylqgodm... | GRP_0 | 83 |
| **3** | unable to access hr_tool page | unable to access hr_tool page | GRP_0 | 100 |
| **4** | skype error | skype error | GRP_0 | 100 |

Dropped Description and short description columns

| | Description | Assignment group |
|---|---|---|
| **0** | login issue -verified user details.(employee# ... | GRP_0 |
| **1** | received from: hmjdrvpb.komuaywn@gmail.com... | GRP_0 |
| **2** | cant log in to vpn received from: eylqgodm... | GRP_0 |
| **3** | unable to access hr_tool page | GRP_0 |
| **4** | skype error | GRP_0 |

In [ ]:
```
# Drop any null values from data
data = data.dropna()
data = data.reset_index(drop=True)
```

In [ ]:
```
data.shape
```

Out[ ]: (7909, 2)

In [ ]:
```
for index in data.index:
  if len(data['Description'][index]) > 5000:
    data = data.drop(index=[index])
    print(f'Row droped from index position {index}')

data = data.reset_index(drop=True)
```

```
Row droped from index position 2930
Row droped from index position 2931
Row droped from index position 3148
Row droped from index position 3342
Row droped from index position 3734
Row droped from index position 3848
Row droped from index position 3850
Row droped from index position 4769
Row droped from index position 5083
Row droped from index position 6269
Row droped from index position 6839
Row droped from index position 7120
Row droped from index position 7429
Row droped from index position 7431
Row droped from index position 7435
Row droped from index position 7437
Row droped from index position 7441
Row droped from index position 7442
Row droped from index position 7443
Row droped from index position 7448
```

In [ ]:
```
data.shape
```

Out[ ]:
```
(7889, 2)
```

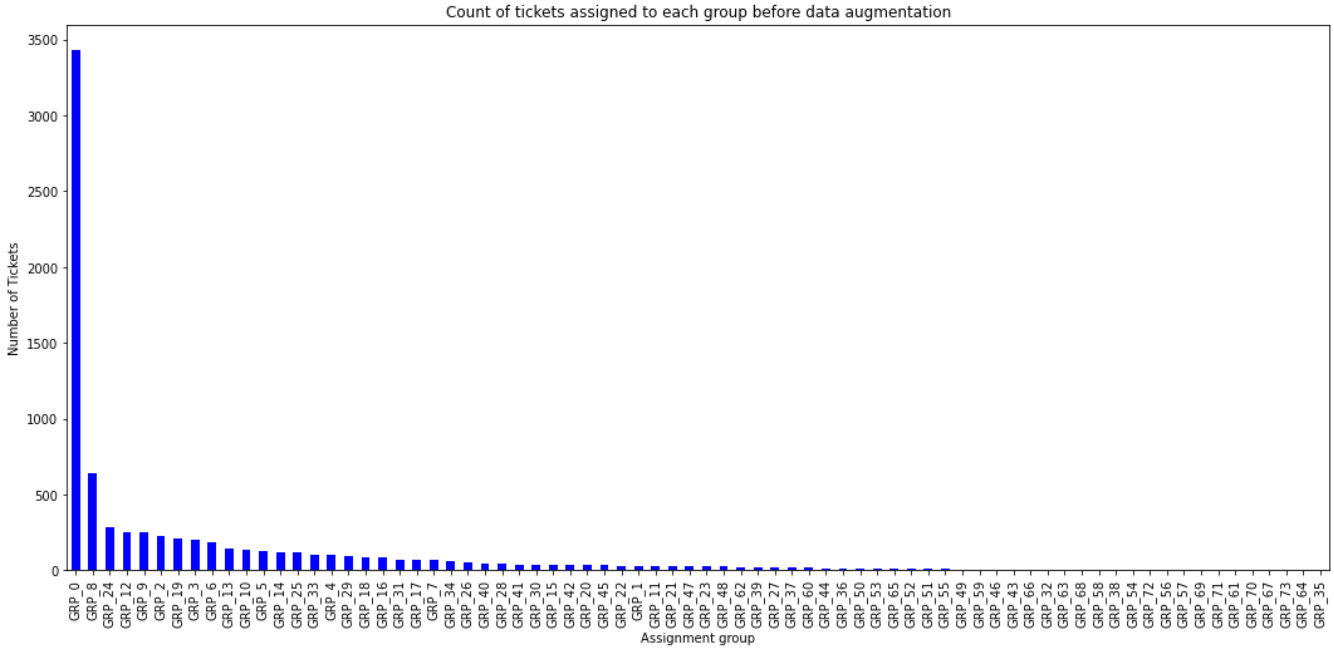
In [ ]:
```
translator = Translator()

def synonymAug(desc):
  lang_det = translator.detect(desc)
  if lang_det.lang != 'en':
    eng_translate = translator.translate(desc,dest='en')
    return eng_translate.text
  return desc


start = time.time()
data['Description'] = data['Description'].apply(synonymAug)
end = time.time()

print(f"Runtime for description translation is {end - start}")
```

```
Runtime for description translation is 1589.14901304245
```

In [ ]:

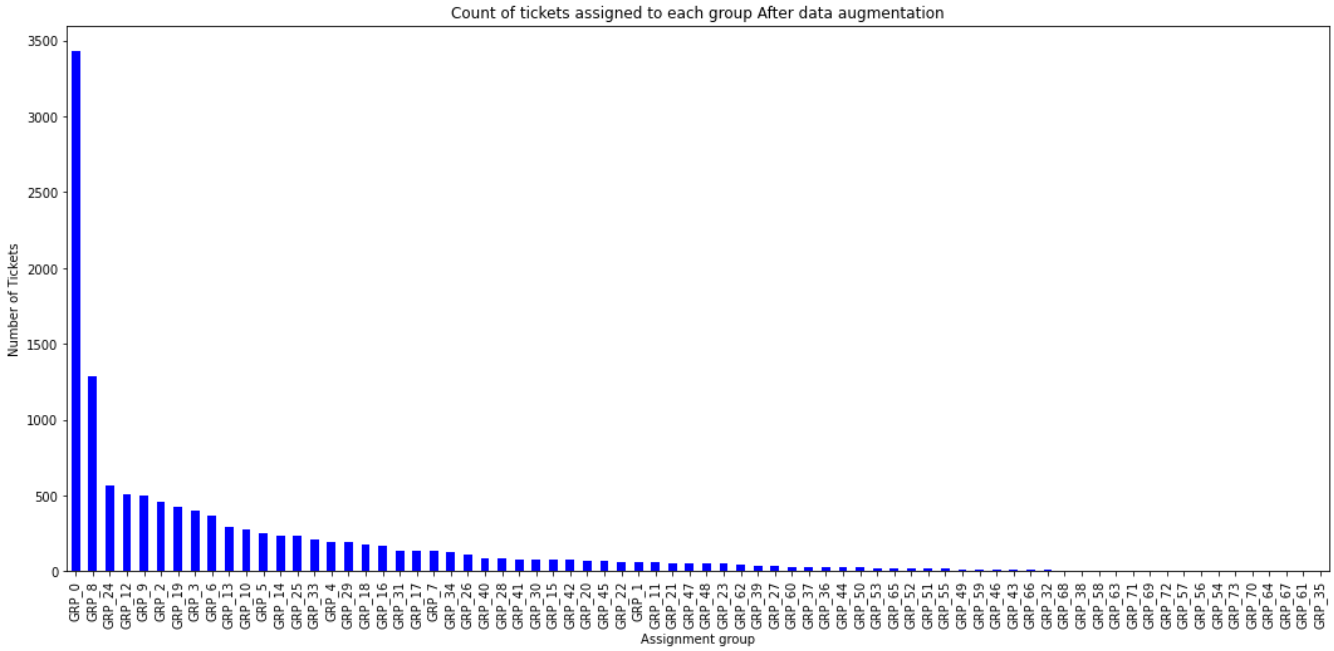
```
print(f'Shape of data before augmentation : {data.shape}')
plt.figure(figsize=(18,8))
plt.xticks(rotation=90)
plt.title('Count of tickets assigned to each group before data augmentation')
plt.xlabel("Assignment group")
plt.ylabel("Number of Tickets")
data['Assignment group'].value_counts().plot.bar(color ='blue')
plt.show()


aug = naw.SynonymAug()
au_data = aug.augment(data[data['Assignment group'] != 'GRP_0']['Description'].
tolist())
y_au_data =  data[data['Assignment group'] != 'GRP_0']['Assignment group'].toli
st()
xnew = data['Description'].tolist()
xnew.extend(au_data)
ynew = data['Assignment group'].tolist()
ynew.extend(y_au_data)
val = {'Description':xnew, 'Assignment group':ynew}
data = pd.DataFrame(val)


print(f'\n\n\nShape of data after augmentation : {data.shape}')
plt.figure(figsize=(18,8))
plt.xticks(rotation=90)
plt.title('Count of tickets assigned to each group After data augmentation')
plt.xlabel("Assignment group")
plt.ylabel("Number of Tickets")
data['Assignment group'].value_counts().plot.bar(color ='blue')
plt.show()
```

```
Shape of data before augmentation : (7889, 2)
```




```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
```

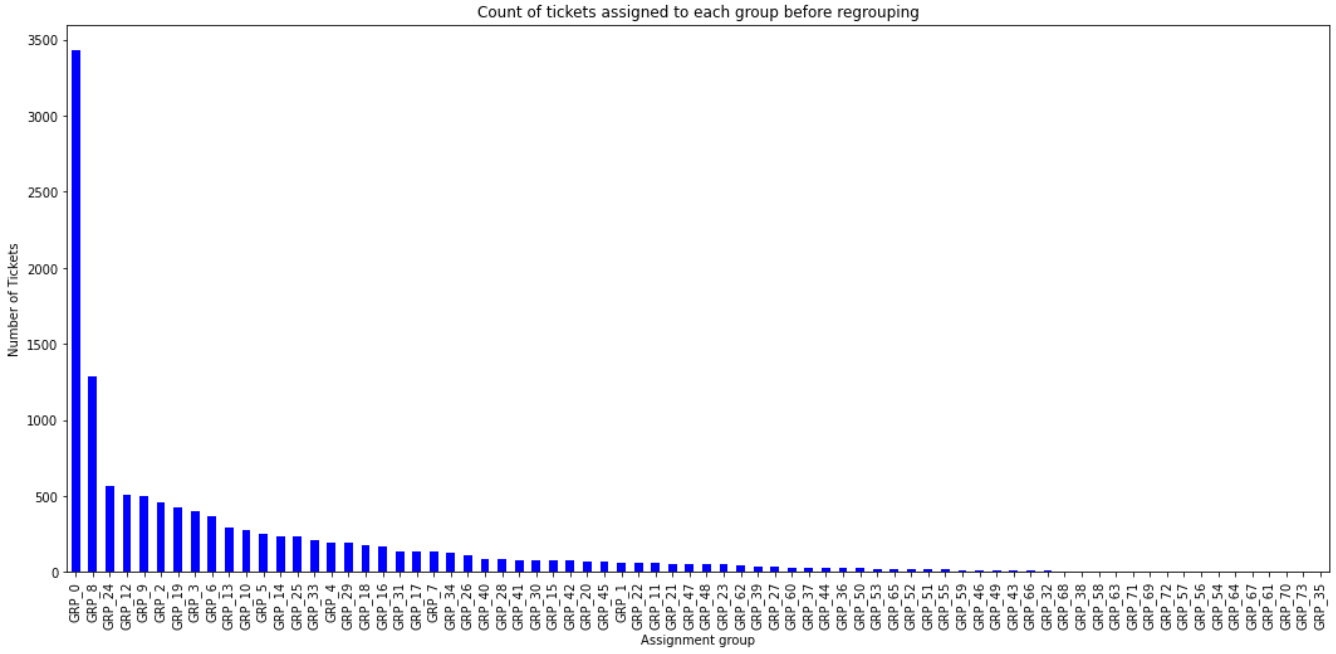
```
Shape of data after augmentation : (12349, 2)
```



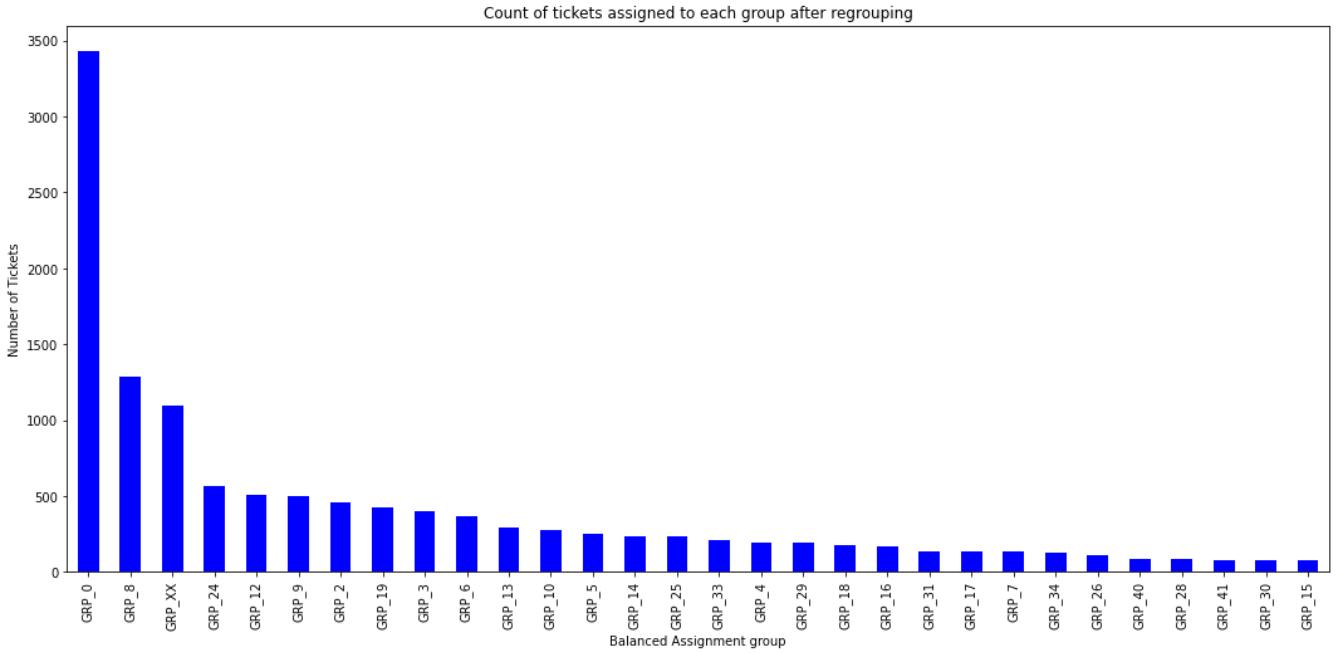
In [ ]:
```
data_groups = (data['Assignment group'].unique())
print(data_groups)
```

```
['GRP_0' 'GRP_1' 'GRP_3' 'GRP_4' 'GRP_5' 'GRP_6' 'GRP_7' 'GRP_8' 'GRP_9'
 'GRP_10' 'GRP_11' 'GRP_12' 'GRP_13' 'GRP_14' 'GRP_15' 'GRP_16' 'GRP_17'
 'GRP_18' 'GRP_19' 'GRP_2' 'GRP_20' 'GRP_21' 'GRP_22' 'GRP_23' 'GRP_24'
 'GRP_25' 'GRP_26' 'GRP_27' 'GRP_28' 'GRP_29' 'GRP_30' 'GRP_31' 'GRP_33'
 'GRP_34' 'GRP_35' 'GRP_36' 'GRP_37' 'GRP_38' 'GRP_39' 'GRP_40' 'GRP_41'
 'GRP_42' 'GRP_43' 'GRP_44' 'GRP_45' 'GRP_46' 'GRP_47' 'GRP_48' 'GRP_49'
 'GRP_50' 'GRP_51' 'GRP_52' 'GRP_53' 'GRP_54' 'GRP_55' 'GRP_56' 'GRP_57'
 'GRP_58' 'GRP_59' 'GRP_60' 'GRP_61' 'GRP_32' 'GRP_62' 'GRP_63' 'GRP_64'
 'GRP_65' 'GRP_66' 'GRP_67' 'GRP_68' 'GRP_69' 'GRP_70' 'GRP_71' 'GRP_72'
 'GRP_73']
```

In [ ]:
```
counts = (data['Assignment group'].value_counts())
plt.figure(figsize=(18,8))
counts.sort_values(ascending=False).plot.bar(color ='blue')
plt.xticks(rotation=90)
plt.title('Count of tickets assigned to each group before regrouping')
plt.xlabel("Assignment group")
plt.ylabel("Number of Tickets")
plt.show()
```



In [ ]:
```
number_of_classes = 30
def regroup_labels(data1): #first 1-22 groups and all other together
  counts = (data['Assignment group'].value_counts())
  grouplist=list(counts.index[(number_of_classes-1):])
  if  data1 in grouplist:
    return 'GRP_XX'
  else: return data1

data['Balanced Assignment Group'] = [regroup_labels(x) for x in data['Assignmen
t group']]

counts = (data['Balanced Assignment Group'].value_counts())
plt.figure(figsize=(18,8))
counts.sort_values(ascending=False).plot.bar(color ='blue')
plt.xticks(rotation=90)
plt.title('Count of tickets assigned to each group after regrouping')
plt.xlabel("Balanced Assignment group")
plt.ylabel("Number of Tickets")
plt.show()

data = data.drop(columns='Assignment group')
```




In [ ]:
```
data['Raw Desc Word Count'] = 0
for index in data.index:
  data['Raw Desc Word Count'][index] = len(data['Description'][index].split())
data.head()
```

Out[ ]:

| | Description | Balanced Assignment Group | Raw Desc Word Count |
|---|---|---|---|
| **0** | login issue -verified user details.(employee# ... | GRP_0 | 35 |
| **1** | received from: hmjdrvpb.komuaywn@gmail.com... | GRP_0 | 25 |
| **2** | cant log in to vpn received from: eylqgodm... | GRP_0 | 16 |
| **3** | unable to access hr_tool page | GRP_0 | 5 |
| **4** | skype error | GRP_0 | 2 |

In [ ]:
```
def custom_replacement(phrase):
    # Actual URl's are encrypted in data, so it is good to remove them from inp
ut data.
    phrase = re.sub(r'https?:\/\/.\/\w', ' ', phrase)
    phrase = re.sub(r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+', ' ', phr
ase)
    return phrase
```

In [ ]:
```
data['Processed Description'] = data.apply(lambda _: '', axis=1)
for index, row in data.iterrows():
  row['Description'] = BeautifulSoup(row['Description'], 'lxml').get_text()
  row['Description'] = custom_replacement(row['Description'])
  row['Description'] = re.sub('[!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n]+', ' ', r
ow['Description'])
  data['Processed Description'][index] = row['Description']

data.head()
```

Out[ ]:

| | Description | Balanced Assignment Group | Raw Desc Word Count | Processed Description |
|---|---|---|---|---|
| **0** | login issue -verified user details. (employee# ... | GRP_0 | 35 | login issue verified user details employee ... |
| **1** | received from: hmjdrvpb.komuaywn@gmail.com... | GRP_0 | 25 | received from hello team my meetings... |
| **2** | cant log in to vpn received from: eylqgodm... | GRP_0 | 16 | cant log in to vpn received from hi ... |
| **3** | unable to access hr_tool page | GRP_0 | 5 | unable to access hr tool page |
| **4** | skype error | GRP_0 | 2 | skype error |

In [ ]:
```
# add custom words to spacy stopwords
custom_stopwords = {'hello','hi','ic','ic:','cc','cc:','bcc','bcc:','to:','subj
ect','subject:','sent:','received','from:',
                    'received from:','etc','com'}
nlp.Defaults.stop_words |= custom_stopwords
```

In [ ]:
```
from spacy.tokens import Doc

desc = list(data['Processed Description'])
docs = [nlp.make_doc(text) for text in desc]

def remove_tokens_on_match(doc):
    indexes = []
    for index, token in enumerate(doc):
        if ((token.is_stop) or (token.is_punct) or (token.like_email) or (token
.is_space) or (token.like_url)):
            indexes.append(index)
    doc2 = Doc(doc.vocab, words=[t.lemma_ for i, t in enumerate(doc) if i not i
n indexes])
    return doc2
```

In [ ]:

```
data['Number of Tokens'] = 0

for doc, ind in zip(docs, range(len(docs))):
  doc2 = remove_tokens_on_match(doc)
  data['Number of Tokens'][ind] = len(doc2)
  if len(doc2):
    data['Processed Description'][ind] =  ' '.join([t.text for t in doc2 ])
  else:
    data['Processed Description'][ind] = ' '

data = data[data['Number of Tokens']!=0]
data.head()
```

Out[ ]:

| | Description | Balanced Assignment Group | Raw Desc Word Count | Processed Description | Number of Tokens |
|---|---|---|---|---|---|
| **0** | login issue -verified user details. (employee# ... | GRP_0 | 35 | login issue verify user detail employee manage... | 22 |
| **1** | received from: hmjdrvpb.komuaywn@gmail.com... | GRP_0 | 25 | team meeting skype meeting appear outlook cale... | 11 |
| **2** | cant log in to vpn received from: eylqgodm... | GRP_0 | 16 | not log vpn log vpn well | 6 |
| **3** | unable to access hr_tool page | GRP_0 | 5 | unable access hr tool page | 5 |
| **4** | skype error | GRP_0 | 2 | skype error | 2 |

In [ ]:
```
for i in range(30,45):
  print("Ticket Description:")
  print(data['Description'][i])
  print("\n\nProcessed Description:")
  print(data['Processed Description'][i])
  print("\n----------------------\n")
```

Ticket Description:
password reset for collaboration_platform

Processed Description:
password reset collaboration platform

---

Ticket Description:
reset  users hi  please reset users password  client id: 794 username : xyz

Processed Description:
reset user reset user password client would 794 username xyz

---

Ticket Description:
duplication of network address.   received from: kxsceyzo.naokumlb@gmail.com  g
entles,  i have two devices that are trying to share an ip address. they are tr
ying to share 96.26.27.9619. one is a printer with the hostname of prtjc0074, a
nd the other is a new display for erp. the display is using dhcp to get its add
ress assigned and the printer is hard coded.  my guess is that the address 96.2
6.27.9619 did not get set to a static address in dhcp. i need this corrected so
the display will pick up another address.

Processed Description:
duplication network address gentles device try share ip address try share 96 26
27 9619 printer hostname prtjc0074 new display erp display dhcp address assign
printer hard code guess address 96 26 27 9619 set static address dhcp need corr
ect display pick address

---

Ticket Description:
ess password reset

Processed Description:
ess password reset

---

Ticket Description:
unable to install flash player

Processed Description:
unable install flash player

---

Ticket Description:
ticket_no1564677-employment status - new non-employee

Processed Description:
ticket no1564677 employment status new non employee

---

Ticket Description:
erp SID_34 account unlock and password reset


Processed Description:
erp SID 34 account unlock password reset

---

Ticket Description:
unable to resolve ticket_no assigned to self the status button is dierppearing
after a few seconds.


Processed Description:
unable resolve ticket assign self status button dierppearing second

---

Ticket Description:
installing engineering tool need to install engineering tool on the pc


Processed Description:
install engineer tool need install engineer tool pc

---

Ticket Description:
call for ecwtrjnq jpecxuty


Processed Description:
ecwtrjnq jpecxuty

---

Ticket Description:
ticket update - inplant_874615


Processed Description:
ticket update inplant 874615

---

Ticket Description:
tablet 7350-sound not working


Processed Description:
tablet 7350 sound work

---

Ticket Description:
unable to login to system


Processed Description:

unable login system

---

Ticket Description:
please reroute jobs on printer01 to printer02 - issue needs to be resolved toda
y   received from: yisohglr.uvteflgb@gmail.com  hi -  the printer01 printer is
not working and needs a part replaced.  can you reroute the jobs in queue to pr
inter printer02?   wihuyjdo qpogfwkb has indicated that prqos001 needs a new pa
rt and it may not deliver for a few days so the inwarehouse_tools will need to
print on printer02 for now.  this needs to be taken care of today since the inw
arehouse_tools are printed and are picked up by an outside vendor at 2:30 pm in
usa on a daily basis.   please contact dkmcfreg anwmfvlgenkataramdntyana if you
have questions about the jobs in queue for today.


Processed Description:
reroute job printer01 printer02 issue need resolve today printer01 printer work
need replace reroute job queue printer printer02 wihuyjdo qpogfwkb indicate prq
os001 need new deliver day inwarehouse tool need print printer02 need take care
today inwarehouse tool print pick outside vendor 2 30 pm usa daily basis contac
t dkmcfreg anwmfvlgenkataramdntyana question job queue today

---

Ticket Description:
unable to login to hr_tool etime


Processed Description:
unable login hr tool etime

---

# Base Line Traditional Models

In [ ]:
```
import time
def model_fn(algo, train, test, algo_text, y_train, y_test, features, i):
  if algo == 'a':
    start = time.time()
    model = SVC()
    model.fit(train, y_train)
    end = time.time()
    print(f"model training time of {algo_text} is {end - start} seconds")
  elif algo == 'b':
    start = time.time()
    model = RandomForestClassifier()
    model.fit(train, y_train)
    end = time.time()
    print(f"model training time of {algo_text} is {end - start} seconds")
  elif algo == 'c':
    start = time.time()
    model = GaussianNB()
    model.fit(train, y_train)
    end = time.time()
    print(f"model training time of {algo_text} is {end - start} seconds")

  y_pred_train = model.predict(train)
  start_pred = time.time()
  y_pred = model.predict(test)
  end_pred = time.time()
  print(f"model predicting time of {algo_text} is {end - start} seconds")
  tr_ac = accuracy_score(y_train,y_pred_train)
  te_ac = accuracy_score(y_test,y_pred)
  print('The train accuracy of ' + features + ' with ' + algo_text + ' is: ', a
ccuracy_score(y_train,y_pred_train))
  print('The test accuracy of ' + features + ' with ' + algo_text + ' is: ', ac
curacy_score(y_test,y_pred))
  results = pd.DataFrame({'Method':[algo_text], 'Features':[features], 'train a
ccuracy': [tr_ac], 'test accuracy':[te_ac],'F1':[f1_score(y_test, y_pred, avera
ge='macro')],'Precesion':[precision_score(y_test, y_pred, average='macro')], 'R
ecall':[recall_score(y_test, y_pred, average='macro')],'Training time':[end - s
tart],'Predicting time':[end_pred - start_pred]},index={i})
  results = results[['Method', 'Features', 'train accuracy','test accuracy','F
1','Precesion','Recall','Training time','Predicting time']]
  return results
```

# Glove

In [ ]:

```
embeddings_index_glove = {}
f = open('/content/drive/My Drive/Glove /glove.6B.300d.txt')
for line in tqdm(f):
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index_glove[word] = coefs
f.close()

# print('Found %s word vectors.' % len(embeddings_index_glove))
import re, string
re_tok = re.compile(u'([{string.punctuation}“”¨«»®´·º½¾¿¡§£€‘’])')

def tokenize(s):
    return re_tok.sub(r' \1 ', s).split()

from nltk.corpus import stopwords
nltk.download("stopwords")
stop_words = set(stopwords.words('english'))

def sent2vec(s, embeddings_index):
    words = str(s)
    words = tokenize(words)
    words = [w for w in words if not w in stop_words]
    words = [w for w in words if w.isalpha()]
    M = []
    for w in words:
        try:
            M.append(embeddings_index[w])
        except:
            continue
    M = np.array(M)
    v = M.sum(axis=0)
    if type(v) != np.ndarray:
        return np.zeros(300)
    return v / np.sqrt((v ** 2).sum())


X_train, X_test, y_train, y_test = train_test_split(data['Processed Descriptio
n'], data['Balanced Assignment Group'], test_size = 0.1, random_state = 42, shu
ffle = True)
le = LabelEncoder()
le.fit(y_train)
y_train=le.transform(y_train)
le = LabelEncoder()
le.fit(y_test)
y_test=le.transform(y_test)


X_train_glove = [sent2vec(x, embeddings_index_glove) for x in (X_train)]
X_test_glove = [sent2vec(x, embeddings_index_glove) for x in (X_test)]

X_train_glove = np.array(X_train_glove)
X_test_glove = np.array(X_test_glove)
model_svm_glove = model_fn('a', X_train_glove, X_test_glove, 'SVM', y_train, y_
test, 'Glove', 1)
model_rf_glove = model_fn('b', X_train_glove, X_test_glove, 'Random Forest', y_
train,y_test,'Glove', 2)
model_nb_glove = model_fn('c', X_train_glove, X_test_glove, 'Naive Bayes', y_tr
ain,y_test,'Glove', 3)
```

```
all = pd.concat([model_svm_glove,model_rf_glove])
all = pd.concat([all,model_nb_glove])
all
```

```
400000it [00:38, 10363.37it/s]

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
model training time of SVM is 73.69811987876892 seconds
model predicting time of SVM is 73.69811987876892 seconds
The train accuracy of Glove with SVM is:  0.396508279337653
The test accuracy of Glove with SVM is:  0.36356275303643726
model training time of Random Forest is 32.37995362281799 seconds
model predicting time of Random Forest is 32.37995362281799 seconds
The train accuracy of Glove with Random Forest is:  0.9641828653707704
The test accuracy of Glove with Random Forest is:  0.5336032388663967
model training time of Naive Bayes is 0.04490542411804199 seconds
model predicting time of Naive Bayes is 0.04490542411804199 seconds
The train accuracy of Glove with Naive Bayes is:  0.20689344852411806
The test accuracy of Glove with Naive Bayes is:  0.2048582995951417
```

Out[ ]:

| | Method | Features | train accuracy | test accuracy | F1 | Precesion | Recall | Training time | Predicting time |
|---|---|---|---|---|---|---|---|---|---|
| **1** | SVM | Glove | 0.396508 | 0.363563 | 0.086464 | 0.102468 | 0.100894 | 73.698120 | 6.859559 |
| **2** | Random Forest | Glove | 0.964183 | 0.533603 | 0.437751 | 0.842476 | 0.348357 | 32.379954 | 0.069916 |
| **3** | Naive Bayes | Glove | 0.206893 | 0.204858 | 0.180636 | 0.212012 | 0.262447 | 0.044905 | 0.052021 |

In [ ]:
```
all.to_excel('results_with_preprocessing.xlsx')
```

# Neural Networks Model

In [ ]:
```
max_features = 10000
maxlen = 25
embedding_size = 300
```

In [ ]:
```
from tensorflow.keras.preprocessing.text import Tokenizer

X = list(data['Processed Description'])
tokenizer = Tokenizer(num_words=max_features , split=' ')
tokenizer.fit_on_texts(X)
X = tokenizer.texts_to_sequences(X)
```

In [ ]:
```
from tensorflow.keras.preprocessing.sequence import pad_sequences
X = pad_sequences(maxlen=maxlen, sequences=X, padding="post")
```

In [ ]:
```
EMBEDDING_FILE = '/content/drive/My Drive/Glove /glove.6B.300d.txt'

num_words = len(tokenizer.word_index) + 1
print(num_words)

embeddings = {}
for o in open(EMBEDDING_FILE):
    word = o.split(" ")[0]
    # print(word)
    embd = o.split(" ")[1:]
    embd = np.asarray(embd, dtype='float32')
    # print(embd)
    embeddings[word] = embd

# create a weight matrix for words in training docs
embedding_matrix = np.zeros((num_words, embedding_size))

for word, i in tokenizer.word_index.items():
        embedding_vector = embeddings.get(word)
        if embedding_vector is not None:
                embedding_matrix[i] = embedding_vector
```

```
21875
```

In [ ]:
```
embedding_matrix.shape
```

Out[ ]:
```
(21875, 300)
```

In [ ]:
```
num_words = len(tokenizer.word_index) + 1
print(num_words)
```

```
21875
```

In [ ]:
```
y = data['Balanced Assignment Group']
le = preprocessing.LabelEncoder()
y = le.fit_transform(y)
y = tensorflow.keras.utils.to_categorical(y, num_classes=number_of_classes)
```

# Vanilla Neural Network Model

## ( only fully connected dense layers )

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, rand
om_state = 42, shuffle = True)
```

In [ ]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, Flatten
from tensorflow.keras import regularizers, optimizers
from tensorflow.keras.initializers import Constant

model = Sequential()
model.add(Embedding(num_words, embedding_size, embeddings_initializer = Constan
t(embedding_matrix), input_length = maxlen, trainable = False))
model.add(Flatten())
model.add(Dense(500, input_shape=((embedding_size*maxlen),), activation='relu'
))
model.add(Dense(100, activation='relu'))
model.add(Dense(number_of_classes, activation='softmax'))

adam = optimizers.Adam(lr=0.0001)
# Compile model
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accura
cy'])
```

In [ ]:
```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 25, 300)           6562500
_________________________________________________________________
flatten_3 (Flatten)          (None, 7500)              0
_________________________________________________________________
dense_9 (Dense)              (None, 500)               3750500
_________________________________________________________________
dense_10 (Dense)             (None, 100)               50100
_________________________________________________________________
dense_11 (Dense)             (None, 30)                3030
=================================================================
Total params: 10,366,130
Trainable params: 3,803,630
Non-trainable params: 6,562,500
_________________________________________________________________
```

In [ ]:
```
# Fit the model and evaluate score
start = time.time()
history = model.fit(X_train, y_train, epochs=50, batch_size=128, verbose= 0)
end = time.time()
print(f"model training time is {end - start} seconds")

score_train = model.evaluate(X_train, y_train, verbose=0)
score_test = model.evaluate(X_test, y_test, verbose=0)

print(f'Train Accuracy {score_train} , Test Accuracy {score_test} ')
```

```
model training time is 204.8787019252777 seconds
Train Accuracy [0.04940839856863022, 0.9866307973861694] , Test Accuracy [1.621
4525699615479, 0.6666666865348816]
```

In [ ]:
```
y_pred_train = np.argmax(model.predict(X_train), axis=-1)

start = time.time()
y_pred = np.argmax(model.predict(X_test), axis=-1)
end = time.time()
print(f"model prediction time is {end - start} seconds")

y_train = np.argmax(y_train, axis=-1)
y_test = np.argmax(y_test, axis=-1)
print(f'train accuracy : {accuracy_score(y_train,y_pred_train)}')
print(f'test accuracy :{accuracy_score(y_test,y_pred)}')
fsc = f1_score(y_test, y_pred, average='macro')
pres = precision_score(y_test, y_pred, average='macro')
rec = recall_score(y_test, y_pred, average='macro')
print(f'F1 score : {fsc}')
print(f'Recall Score : {rec}')
print(f'Precision Score : {pres}')
```

```
model prediction time is 0.31496262550354004 seconds
train accuracy : 0.9866307761327949
test accuracy :0.6666666666666666
F1 score : 0.6242240384882887
Recall Score : 0.6070748069462366
Precision Score : 0.6592048234705572
```

# Neural network - Bidirectional Lstm

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, rand
om_state = 42, shuffle = True)
```

In [ ]:
```
from keras.models import Sequential
from keras.layers import Embedding, Flatten, Dense, LSTM, Bidirectional, Dropou
t, Conv1D, MaxPool1D
from keras.layers import GlobalMaxPool1D, GRU
from keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Embedding, Dense, TimeDistributed, Fl
atten, Dropout
```

In [ ]:
```
model = Sequential()
model.add(Embedding(num_words, embedding_size, embeddings_initializer = Constan
t(embedding_matrix), input_length = maxlen, trainable = False))
model.add(Bidirectional(LSTM(100, return_sequences=True, recurrent_dropout=0.1,
dropout=0.1)))
model.add(TimeDistributed(Dense(100, activation="softmax")))
model.add(Flatten())
model.add(Dense(300, activation='relu'))
model.add(Dense(number_of_classes, activation='softmax'))

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [ ]:
```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 25, 300)           6562500
_________________________________________________________________
bidirectional_1 (Bidirection (None, 25, 200)           320800
_________________________________________________________________
time_distributed_2 (TimeDist (None, 25, 100)           20100
_________________________________________________________________
flatten_4 (Flatten)          (None, 2500)              0
_________________________________________________________________
dense_13 (Dense)             (None, 300)               750300
_________________________________________________________________
dense_14 (Dense)             (None, 30)                9030
=================================================================
Total params: 7,662,730
Trainable params: 1,100,230
Non-trainable params: 6,562,500
_________________________________________________________________
```

In [ ]:
```
# Fit the model and evaluate score
start = time.time()
history = model.fit(X_train, y_train, epochs=30, batch_size=128, verbose= 0)
end = time.time()
print(f"model training time is {end - start} seconds")

score_train = model.evaluate(X_train, y_train, verbose=0)
score_test = model.evaluate(X_test, y_test, verbose=0)

print(f'Train Accuracy {score_train} , Test Accuracy {score_test} ')
```

```
model training time is 916.0321681499481 seconds
Train Accuracy [0.10952193289995193, 0.9668012857437134] , Test Accuracy [1.077
3478746414185, 0.7449555993080139]
```

In [ ]:
```
y_pred_train = np.argmax(model.predict(X_train), axis=-1)
start = time.time()
y_pred = np.argmax(model.predict(X_test), axis=-1)
end = time.time()
print(f"model prediction time is {end - start} seconds")
y_train = np.argmax(y_train, axis=-1)
y_test = np.argmax(y_test, axis=-1)
print(f'train accuracy : {accuracy_score(y_train,y_pred_train)}')
print(f'test accuracy :{accuracy_score(y_test,y_pred)}')
fsc = f1_score(y_test, y_pred, average='macro')
pres = precision_score(y_test, y_pred, average='macro')
rec = recall_score(y_test, y_pred, average='macro')
print(f'F1 score : {fsc}')
print(f'Recall Score : {rec}')
print(f'Precision Score : {pres}')
```

```
model prediction time is 0.9845321178436279 seconds
train accuracy : 0.9668012561686855
test accuracy :0.744955609362389
F1 score : 0.7185097273818384
Recall Score : 0.723597230300654
Precision Score : 0.7288765247407618
```