NLP Phase 3

Purvaj Desai (A20469336)

In [32]:
```
import pandas as pd
import numpy as np
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
import langdetect
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, accuracy_score, confusion_
```

In [33]:
```
df_primary_main = pd.read_csv(r'C:\College\SEM 2\NLP\Assignment 2\Final Datas
df_secondary_main = pd.read_csv(r'C:\College\SEM 2\NLP\Assignment 2\Final Dat
df_primary_main = df_primary_main.iloc[:,1:3]
df_secondary_main = df_secondary_main.iloc[:,1:3]
```

In [34]:
```
df_primary = df_primary_main.copy()
df_secondary = df_secondary_main.copy()
```

Base Model

In [35]:
```
df_primary['text'] = df_primary['text'].apply(lambda x:' '.join([word for wor

df_secondary['text'] = df_secondary['text'].apply(lambda x:' '.join([word for
```

In [36]:
```
X_pri = df_primary.drop(["label"],axis = 1)
y_pri = df_primary["label"]

X_sec = df_secondary.drop(["label"],axis = 1)
y_sec = df_secondary["label"]
```

In [37]:
```
X_pri_train,X_pri_test,y_pri_train,y_pri_test = train_test_split(X_pri,y_pri,
```

In [38]:

```
tf = TfidfVectorizer()

final_x_pri_train = tf.fit_transform(X_pri_train.text).toarray()

final_x_pri_test = tf.transform(X_pri_test.text).toarray()


final_x_sec_test = tf.transform(X_sec.text).toarray()
```

In [39]:

```
clf = GradientBoostingClassifier(n_estimators = 100,max_depth = 5, random_sta
clf.fit(final_x_pri_train, y_pri_train)
y_pred_pri = clf.predict(final_x_pri_test)

print("The Classification Report :\n",classification_report(y_pri_test,y_pred
print("\nThe Accuracy in Primary Set is:",round((accuracy_score(y_pri_test,y_
print("\nConfusion Matrix : \n",confusion_matrix(y_pri_test,y_pred_pri))
```

```
The Classification Report :
                  precision    recall  f1-score   support

anti-mitigation       0.17      0.08      0.11        24
 pro-mitigation       0.71      0.80      0.75       147
        unclear       0.55      0.49      0.52        73
unclear,unclear       0.00      0.00      0.00         0

       accuracy                           0.64       244
      macro avg       0.36      0.34      0.34       244
   weighted avg       0.61      0.64      0.62       244


The Accuracy in Primary Set is: 63.52

Confusion Matrix :
 [[  2  14   8   0]
 [  7 117  22   1]
 [  3  34  36   0]
 [  0   0   0   0]]

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.
py:1245: UndefinedMetricWarning: Recall and F-score are ill-defined and bei
ng set to 0.0 in labels with no true samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.
py:1245: UndefinedMetricWarning: Recall and F-score are ill-defined and bei
ng set to 0.0 in labels with no true samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.
py:1245: UndefinedMetricWarning: Recall and F-score are ill-defined and bei
ng set to 0.0 in labels with no true samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [40]:
```
y_pred_sec = clf.predict(final_x_sec_test)

print("The Classification Report: \n",classification_report(y_sec,y_pred_sec)
print("\nThe Accuracy in Secondary Set is:",round((accuracy_score(y_sec,y_pre
print("\nConfusion Matrix : \n",confusion_matrix(y_sec,y_pred_sec))
```

```
The Classification Report:
                  precision    recall  f1-score   support

anti-mitigation       0.31      0.20      0.24        25
 pro-mitigation       0.64      0.71      0.67       155
        unclear       0.58      0.54      0.56       120

       accuracy                           0.60       300
      macro avg       0.51      0.48      0.49       300
   weighted avg       0.59      0.60      0.59       300


The Accuracy in Secondary Set is: 60.0

Confusion Matrix :
 [[  5  12   8]
 [  6 110  39]
 [  5  50  65]]
```

Model using Feature Engineering

In [41]:
```
df_primary1 = df_primary_main.copy()
df_secondary1 = df_secondary_main.copy()
```

In [42]:

```
import nltk
# count number of characters
def count_chars(text):
    return len(text)

# count number of words
def count_words(text):
    return len(text.split())

# count number of capital characters
def count_capital_chars(text):
    count=0
    for i in text:
        if i.isupper():
            count+=1
    return count

# count number of capital words
def count_capital_words(text):
    return sum(map(str.isupper,text.split()))

# count number of words in quotes
def count_words_in_quotes(text):
    x = re.findall("\'.\'|\".\"", text)
    count=0
    if x is None:
        return 0
    else:
        for i in x:
            t=i[1:-1]
            count+=count_words(t)
        return count

# count number of sentences
def count_sent(text):
    return len(nltk.sent_tokenize(text))


# count number of unique words
def count_unique_words(text):
    return len(set(text.split()))


from porter2stemmer import Porter2Stemmer
def text_processing(text):

    #Removing Stopwords and Punctuations using Reegular Expression
    data = re.sub('[^a-zA-Z]', ' ',text)
    data = data.lower()
    data = data.split()
    data = [word for word in data if not word in set(stopwords.words('english
    data = ' '.join(data)

    #Performing Stemming using PortarStemmer2
    stemmer = Porter2Stemmer()
    words = word_tokenize(data)
```

```
    words = [ stemmer.stem(word) for word in words ]
    data = ' '.join(words)

    return data
```

In [43]:

```
df_primary1['char_count'] = df_primary1["text"].apply(lambda x:count_chars(x)
df_primary1['word_count'] = df_primary1["text"].apply(lambda x:count_words(x)
df_primary1['sent_count'] = df_primary1["text"].apply(lambda x:count_sent(x))
df_primary1['capital_char_count'] = df_primary1["text"].apply(lambda x:count_
df_primary1['capital_word_count'] = df_primary1["text"].apply(lambda x:count_
df_primary1['quoted_word_count'] = df_primary1["text"].apply(lambda x:count_w
df_primary1['unique_word_count'] = df_primary1["text"].apply(lambda x:count_u
df_primary1['text'] = df_primary1["text"].apply(text_processing)
```

In [44]:

```
df_secondary1['char_count'] = df_secondary1["text"].apply(lambda x:count_char
df_secondary1['word_count'] = df_secondary1["text"].apply(lambda x:count_word
df_secondary1['sent_count'] = df_secondary1["text"].apply(lambda x:count_sent
df_secondary1['capital_char_count'] = df_secondary1["text"].apply(lambda x:cc
df_secondary1['capital_word_count'] = df_secondary1["text"].apply(lambda x:cc
df_secondary1['quoted_word_count'] = df_secondary1["text"].apply(lambda x:cou
df_secondary1['unique_word_count'] = df_secondary1["text"].apply(lambda x:cou
df_secondary1['text'] = df_secondary1["text"].apply(text_processing)
```

In [53]:
```
df_primary1
```

Out[53]:

	text	label	char_count	word_count	sent_count	capital_char_count	capital_wo
0	gloat elect consequ let administ vaccin consequ	unclear	108	16	1	9	
1	corpor health privaci us start outlaw drug tes...	unclear	183	29	2	4	
2	done	unclear	17	4	2	2	
3	eaten outsid hous sinc march miss favorit plac...	unclear	304	63	6	10	
4	peopl die day us virus taiwan new case tell pe...	pro-mitigation	146	29	2	5	
...	...	...	...	...	...	...	
808	went lunch yesterday friend second time fun si...	pro-mitigation	701	126	9	21	
809	even year half later understand peopl object s...	pro-mitigation	862	156	6	10	
810	let get straight vaccin peopl suppos go back w...	unclear	278	48	3	3	
811	suffer cold flu year henc alway wear mask arou...	pro-mitigation	188	37	2	4	
812	can not understand peopl still stand line toda...	pro-mitigation	244	45	4	4	

813 rows × 9 columns

In [54]:
```
df_secondary1
```

Out[54]:

	text	label	char_count	word_count	sent_count	capital_char_count	capital_wor
0	give peopl wide berth nee protect given peopl ...	pro-mitigation	151	30	2	4	
1	would love know well daycar preschool union su...	unclear	219	39	2	2	
2	moral peopl see easi way wear mask mayb encour...	pro-mitigation	209	34	2	2	
3	mask time dont trust variant dont want surpris...	pro-mitigation	257	51	4	0	
4	insid someth like groceri store think think un...	unclear	166	28	2	2	
...	...	...	...	...	...	...	
295	vaccin presum vac ever exist kinda put cart hors	unclear	106	20	3	3	
296	nope still wear mask want protect longer manda...	pro-mitigation	168	34	1	1	
297	bad publish district declin open fall probabl ...	pro-mitigation	186	33	4	4	

	text	label	char_count	word_count	sent_count	capital_char_count	capital_wor
298	simpl either youreal care kid one peopl see go...	unclear	164	31	1	2	
299	see report guy much may want remov feel like c...	pro-mitigation	497	95	8	14	

300 rows × 9 columns

In [45]:
```
X_pri1 = df_primary1.drop(["label"],axis = 1)
y_pri1 = df_primary1["label"]

X_sec1 = df_secondary1.drop(["label"],axis = 1)
y_sec1 = df_secondary1["label"]

print(y_pri1.unique())
print(y_sec1.unique())
```

```
['unclear' 'pro-mitigation' 'anti-mitigation' 'unclear,unclear']
['pro-mitigation' 'unclear' 'anti-mitigation']
```

In [46]:
```
y_pri1 = y_pri1.replace('unclear,unclear','unclear')
print(y_pri1.unique())
```

```
['unclear' 'pro-mitigation' 'anti-mitigation']
```

In [47]:
```
X_pri_train_new,X_pri_test_new,y_pri_train_new,y_pri_test_new = train_test_sp
```

In [48]:

```
X_pri_train_new.reset_index(inplace=True)
X_pri_test_new.reset_index(inplace=True)

X_pri_train_new = X_pri_train_new.drop("index",axis = 1)
X_pri_test_new = X_pri_test_new.drop("index",axis = 1)
tf = TfidfVectorizer()

final_x_pri_train1 = tf.fit_transform(X_pri_train_new.text).toarray()
df = pd.DataFrame(final_x_pri_train1)
final_x_pri_train1 = X_pri_train_new.join(df)


final_x_pri_test1 = tf.transform(X_pri_test_new.text).toarray()
df1 = pd.DataFrame(final_x_pri_test1)
final_x_pri_test1 = X_pri_test_new.join(df1)

final_x_sec_test1 = tf.transform(X_sec1.text).toarray()
df2 = pd.DataFrame(final_x_sec_test1)
final_x_sec_test1 = X_sec1.join(df2)

final_x_pri_train1 = final_x_pri_train1.drop("text",axis = 1)
final_x_pri_test1 = final_x_pri_test1.drop("text",axis = 1)
final_x_sec_test1 = final_x_sec_test1.drop("text",axis = 1)
```

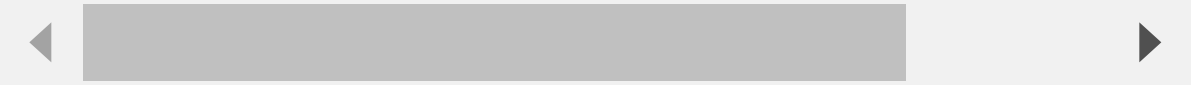
In [55]:

```
df1
```

Out[55]:

	0	1	2	3	4	5	6	7	8	9	...	3510	3511	3512	3513	3514	3515	3
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
239	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
240	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
241	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
242	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
243	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	

244 rows × 3520 columns

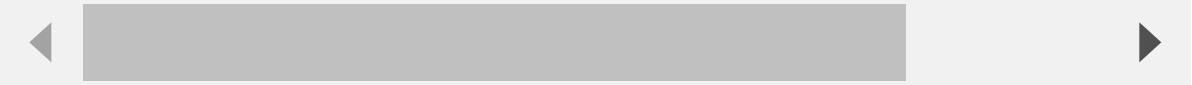


In [56]:
```
df2
```

Out[56]:

	0	1	2	3	4	5	6	7	8	9	...	3510	3511	3512	3513	3514	3515	3
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
295	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
296	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
297	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
298	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
299	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	

300 rows × 3520 columns



In [49]:
```
clf1 = GradientBoostingClassifier(n_estimators = 100,max_depth = 5, random_st
clf1.fit(final_x_pri_train1, y_pri_train_new)
y_pred_pri1 = clf1.predict(final_x_pri_test1)

print("The Classification Report :\n",classification_report(y_pri_test_new,y_
print("\nThe Accuracy in Primary Set is:",round((accuracy_score(y_pri_test_ne
print("\nConfusion Matrix : \n",confusion_matrix(y_pri_test_new,y_pred_pri1))
```

```
The Classification Report :
                  precision    recall  f1-score   support

anti-mitigation       0.38      0.12      0.19        24
 pro-mitigation       0.71      0.82      0.76       147
        unclear       0.50      0.45      0.47        73

       accuracy                           0.64       244
      macro avg       0.53      0.46      0.47       244
   weighted avg       0.61      0.64      0.62       244


The Accuracy in Primary Set is: 63.93

Confusion Matrix :
 [[  3  12   9]
 [  3 120  24]
 [  2  38  33]]
```

In [50]:
```
y_pred_sec1 = clf1.predict(final_x_sec_test1)

print("The Classification Report: \n",classification_report(y_sec1,y_pred_sec
print("\nThe Accuracy in Secondary Set is:",round((accuracy_score(y_sec1,y_pr
print("\nConfusion Matrix : \n",confusion_matrix(y_sec1,y_pred_sec1))
```

```
The Classification Report:
                   precision    recall  f1-score   support

anti-mitigation       0.11      0.04      0.06        25
 pro-mitigation       0.65      0.68      0.67       155
        unclear       0.58      0.62      0.60       120

       accuracy                           0.61       300
      macro avg       0.45      0.45      0.44       300
   weighted avg       0.58      0.61      0.59       300


The Accuracy in Secondary Set is: 60.67

Confusion Matrix :
 [[  1  14  10]
 [  5 106  44]
 [  3  42  75]]
```

In [51]:
```
labels_pri = df_primary['label'].value_counts()
labels_pri
```

Out[51]:
```
pro-mitigation     471
unclear            262
anti-mitigation     79
unclear,unclear      1
Name: label, dtype: int64
```

In [52]:
```
labels_sec = df_secondary['label'].value_counts()
labels_sec
```

Out[52]:
```
pro-mitigation     155
unclear            120
anti-mitigation     25
Name: label, dtype: int64
```