DATA MINING LAB 2 HOMEWORK REPORT

PRANAV NAGPURE
109065427

Task

Given: A dataset is which was crawled from Twitter, and already labelled with the emotion for these tweets by some specific hashtags in the original text. There are 8 classes (or say emotions) in the dataset: anger, anticipation, disgust, fear, sadness, surprise, trust, and joy.

Objectives: Clean the data by doing some pre-processing first. Then, apply feature engineering or any other data mining technique you have or haven't learned in the Data Mining course. The final goal is to learn a model that can predict the emotion behind each tweet.

1 Pre-processing

1.1 Importing the Data

Tweets in the dataset exist as dictionary with three (key, value) pairs. So, first I create three more columns in the dataframe one for each pair. Then filter the tweets for which emotion is given, i.e., tweets in the training set. However, **for the purpose of evaluating the models I create testing data out of this set later.**

```
data_identification = pd.read_csv('data_identification.csv')
emotion = pd.read_csv('emotion.csv')
tweets = pd.read_json('tweets_DM.json', lines = True)
```

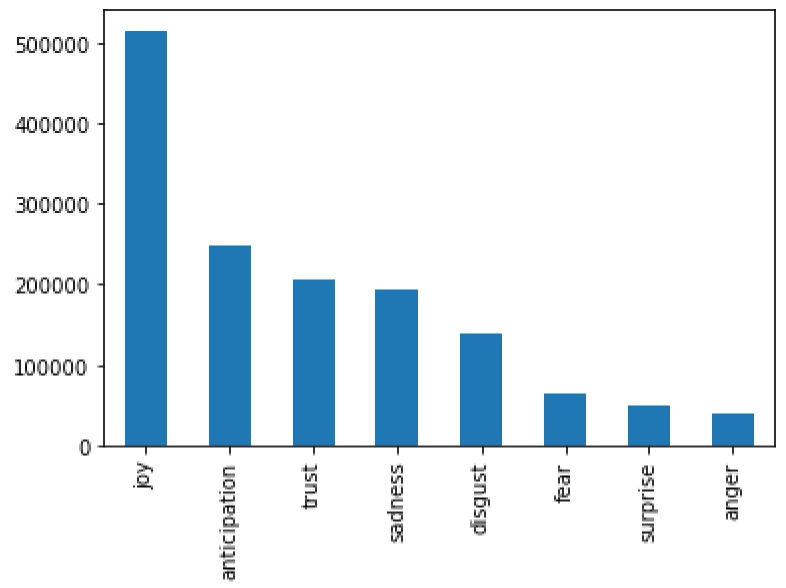
```
tweets['text'] = tweets['_source'].apply(lambda x: x['tweet']['text'])
tweets['hashtags'] = tweets['_source'].apply(lambda x: x['tweet']['hashtags'])
tweets['id'] = tweets['_source'].apply(lambda x: x['tweet']['tweet_id'])
```

1.2 Distribution of classes

I visualized the distribution of classes using a bar graph.

```
tweets_train.emotion.value_counts().plot(kind = 'bar')
```

```
<AxesSubplot:>
```



2 Feature Engineering

2.1 Cleaning

We have only one input feature i.e., tweet in text format. And only one output feature i.e., emotion which belong to a finite set of emotions

I perform following operations on the text (in order):

1. Remove all the stop words using neattext library. Stop words are the very common words like 'if', ‘but’, ‘we', 'he', 'she', and 'they'.
2. Remove all the userhandles using neattext library.
3. Remove all punctuations using neattext library.
4. There is ‘<LH>’ in almost every tweet. I remove this using a lambda function.
5. I decapitalize the first character of every word which is not fully capitalized. Fully capitalized words usually suggest that the author of tweet wanted to emphasize on it. So, fully capitalized words remain unchanged. This is done by decap function.
6. Remove all tweets which are blank or became blank after above operations.

Note: The emojis are not affected by these operations and remain unchanged. However, they do not appear in Visualizations. Rather I checked it separately.

```
def decap(text):
    decap_text = ''
    for word in text.split():
        if word.isupper():
            decap_text += word
        else:
            decap_text += word.lower()
        decap_text += ' '
    return decap_text[:-1]
```

```
tweets_train['text'] = tweets_train['text'].apply(nfx.remove_stopwords)
tweets_train['text'] = tweets_train['text'].apply(nfx.remove_userhandles)
tweets_train['text'] = tweets_train['text'].apply(nfx.remove_punctuations)
tweets_train['text'] = tweets_train['text'].apply(lambda x: x.replace('<LH>', ''))
tweets_train['text'] = tweets_train['text'].apply(decap)
```

```
tweets_train = tweets_train[~(tweets_train['text'] == '')]
```

2.2 Converting to Bag Of Words

Converted text to bag of words for all the models (After Visualizations)

```
count_vectorizer = CountVectorizer()
data_X = count_vectorizer.fit_transform(tweets_train['text'])
data_Y = tweets_train['emotion']
X_train, X_test, Y_train, Y_test = train_test_split(data_X, data_Y, test_size = 0.22)
```

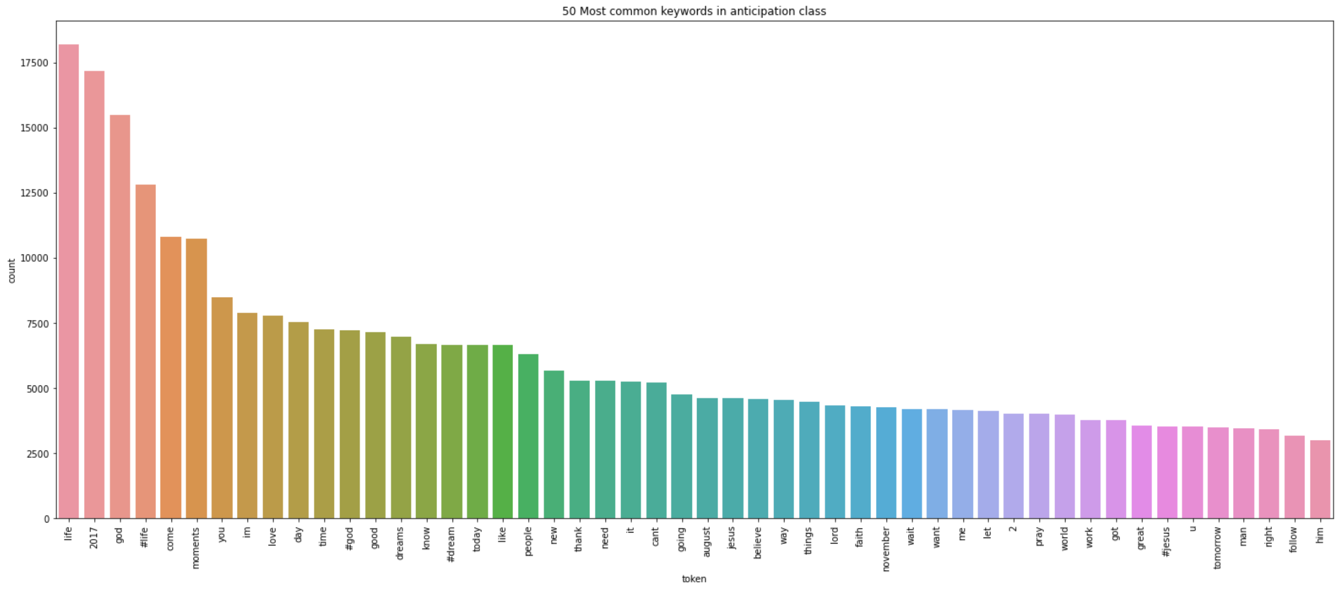
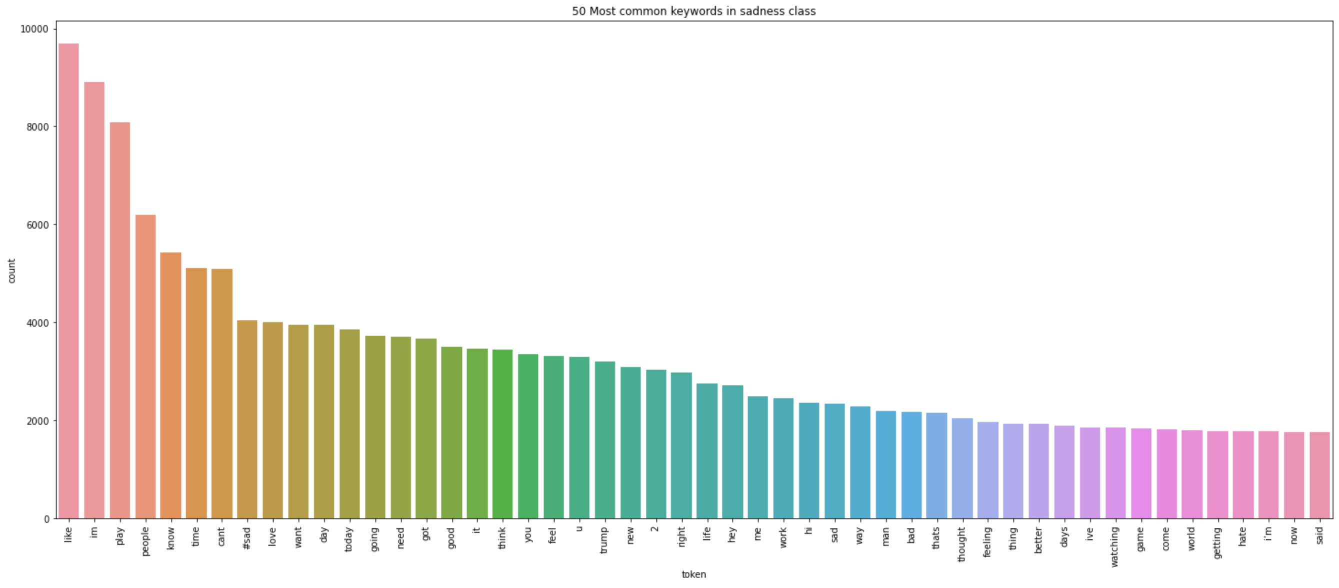
3 Visualizing the Data

3.1 50 Most Common Words by class

Made 8 bar plots one for each emotion (Two examples (Anticipation and Sadness) in next page)

```python
def plot_mc_keywords(texts, title):
    keywords = dict(Counter(' '.join(texts.tolist()).split()).most_common(50))
    df = pd.DataFrame(keywords.items(), columns = ['token', 'count'])
    plt.figure(figsize = (25, 10))
    sns.barplot(data = df, x = 'token', y= 'count')
    plt.xticks(rotation = 90)
    plt.title('50 Most common keywords in '+ title + ' class')
    plt.show()
```

```python
for e in tweets_train.emotion.unique():
    plot_mc_keywords(tweets_train[tweets_train['emotion'] == e]['text'], e)
```


50 Most common keywords in anticipation class
count
token
life
2017
god
#life
come
moments
you
im
love
day
time
#god
good
dreams
know
#dream
today
like
people
new
thank
need
it
cant
going
august
jesus
believe
way
things
lord
faith
november
wait
want
me
let
2
pray
world
work
got
great
#jesus
u
tomorrow
man
right
follow
him

50 Most common keywords in sadness class
count
token
like
im
play
people
know
time
cant
#sad
love
want
day
today
going
need
got
good
it
think
you
feel
u
trump
new
2
right
life
hey
me
work
hi
sad
way
man
bad
thats
thought
feeling
thing
better
days
ive
watching
game
come
world
getting
hate
i'm
now
said

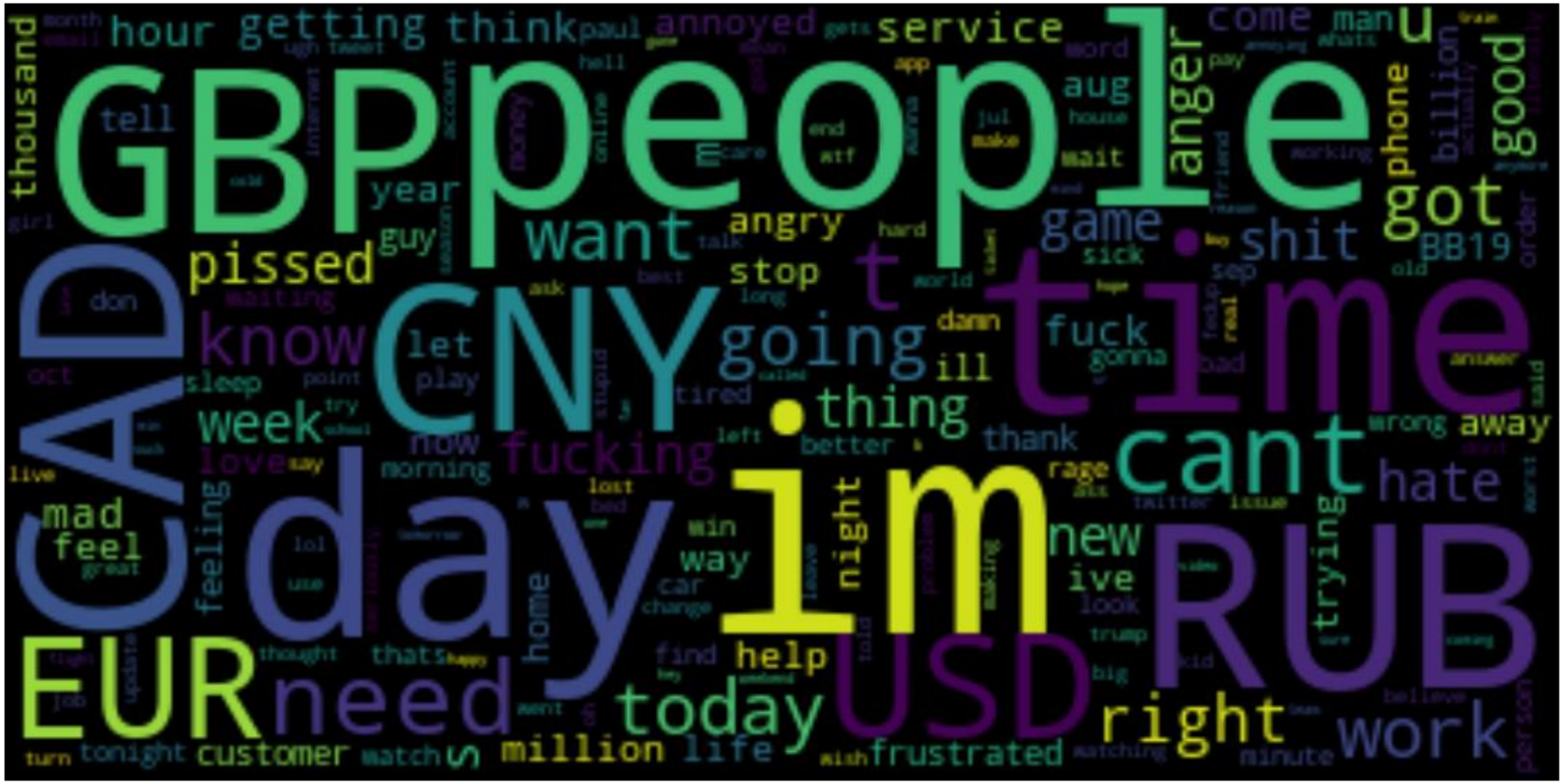
3.2 Word clouds

Made 8 word clouds one for each emotion. These helped to figure out what kind of data cleaning is required. (One example (Anger) in next page)

```python
def plot_wordcloud(texts, title):
    plt.figure(figsize = (25, 10))
    plt.imshow(WordCloud(collocations = False).generate(' '.join(texts.tolist())), interpolation = 'bilinear')
    plt.axis('off')
    plt.title(title)
    plt.show()
```

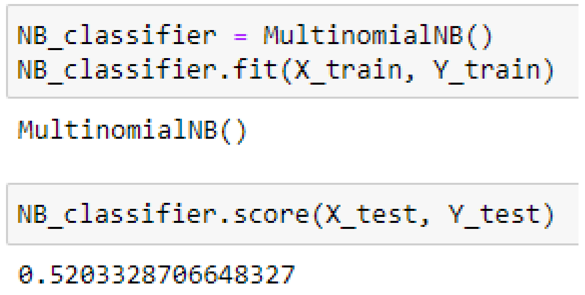
```python
for e in tweets_train.emotion.unique():
    plot_wordcloud(tweets_train[tweets_train['emotion'] == e]['text'], e)
```

anger


GBP
people
time
CAD
CNY
day
im
RUB
USD
EUR
need
today
cant
know
going
want
right
work
got
anger
fucking
thing
week
pissed
hate
game
shit
new
hour
getting
think
service
annoyed
come
man
u
good
phone
billion
thousand
angry
stop
BB19
mad
feel
million
customer
frustrated
help
life
tonight
trying
home
night
feeling
way

4 Naïve Bayes Classifier

Implemented a multinomial naïve bayes classifier, which gives accuracy of 52%. Also, tried complement naïve bayes but it had poor performance compared to multinomial



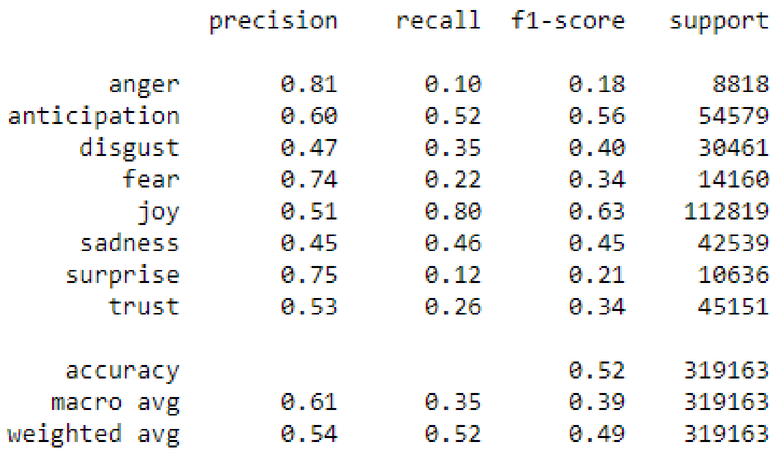
```
NB_classifier = MultinomialNB()
NB_classifier.fit(X_train, Y_train)
```

```
MultinomialNB()
```

```
NB_classifier.score(X_test, Y_test)
```

```
0.5203328706648327
```

Evaluation

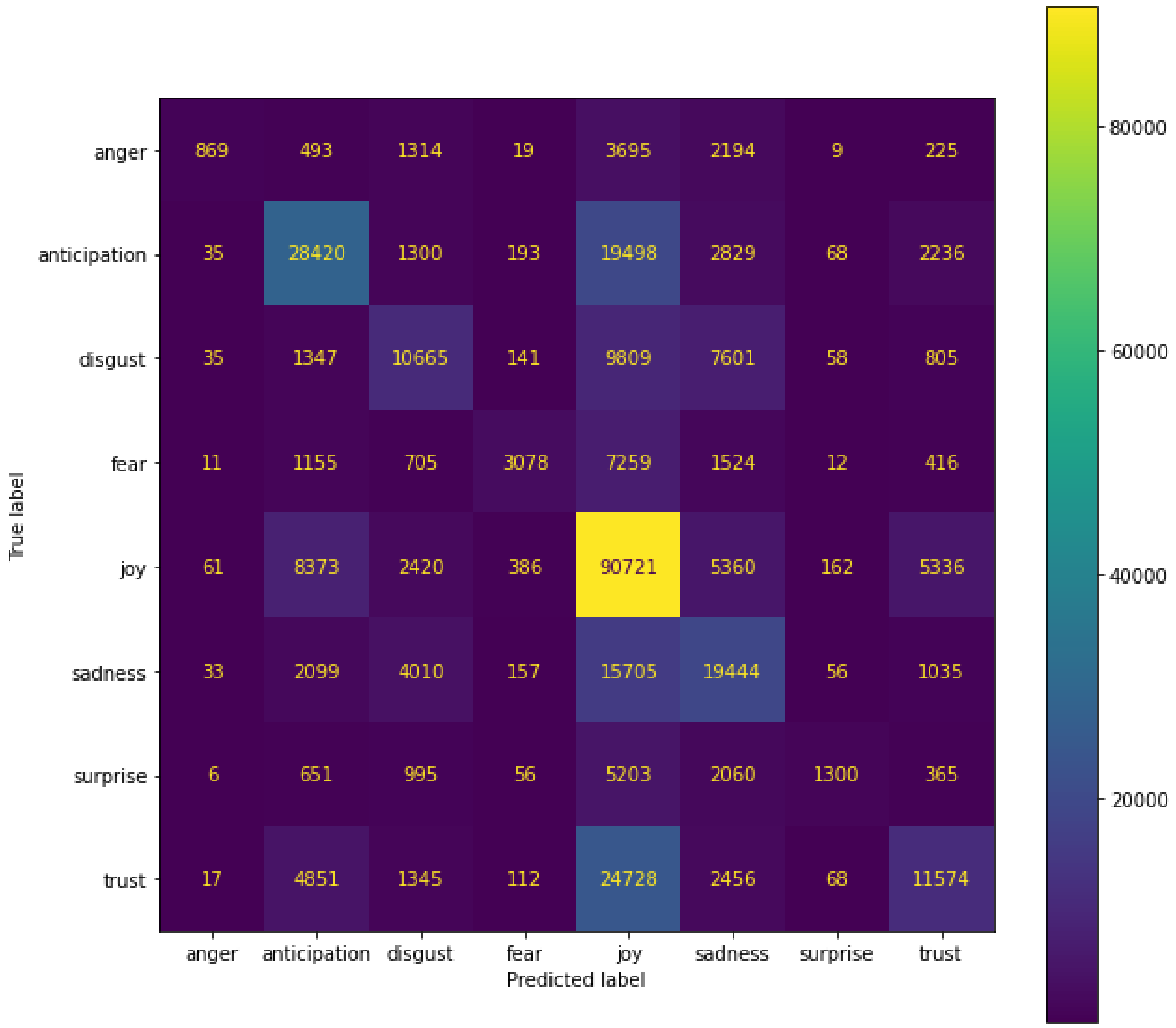


```
               precision    recall  f1-score   support

       anger       0.81      0.10      0.18      8818
anticipation       0.60      0.52      0.56     54579
     disgust       0.47      0.35      0.40     30461
        fear       0.74      0.22      0.34     14160
         joy       0.51      0.80      0.63    112819
     sadness       0.45      0.46      0.45     42539
    surprise       0.75      0.12      0.21     10636
       trust       0.53      0.26      0.34     45151

    accuracy                           0.52    319163
   macro avg       0.61      0.35      0.39    319163
weighted avg       0.54      0.52      0.49    319163
```

Very high recall for joy compared to other emotions implies that model is struggling because of skew in distribution

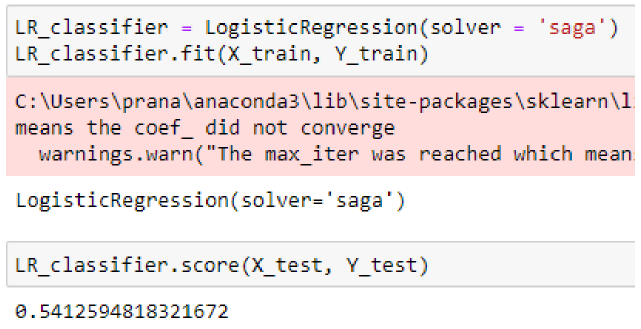


	anger	anticipation	disgust	fear	joy	sadness	surprise	trust
anger	869	493	1314	19	3695	2194	9	225
anticipation	35	28420	1300	193	19498	2829	68	2236
disgust	35	1347	10665	141	9809	7601	58	805
fear	11	1155	705	3078	7259	1524	12	416
joy	61	8373	2420	386	90721	5360	162	5336
sadness	33	2099	4010	157	15705	19444	56	1035
surprise	6	651	995	56	5203	2060	1300	365
trust	17	4851	1345	112	24728	2456	68	11574

Confucian Matrix for Multinomial Naïve Bayes Classifier

5 Logistic Regression Classifier

Implemented a Logistic Regression classifier, which gives slightly improved accuracy of 54%. Also, tried it with different solvers available. Saga solver gave the best results. It is the recommended solver for high dimensional sparse matrix as per sklearn's documentation.



```
LR_classifier = LogisticRegression(solver = 'saga')
LR_classifier.fit(X_train, Y_train)
```

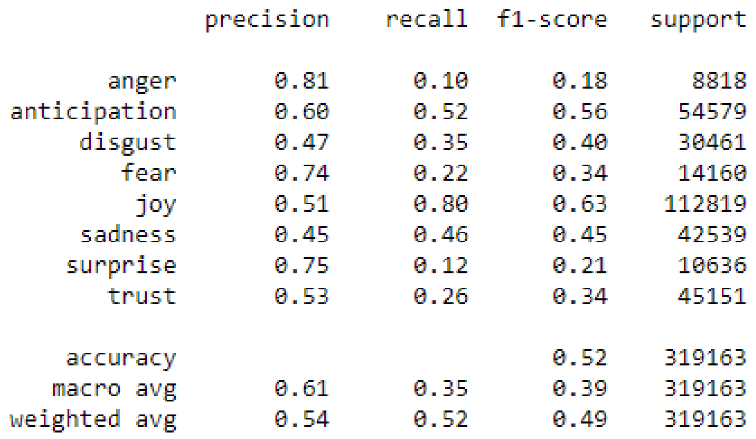
```
C:\Users\prana\anaconda3\lib\site-packages\sklearn\l
means the coef_ did not converge
  warnings.warn("The max_iter was reached which mean
```

```
LogisticRegression(solver='saga')
```

```
LR_classifier.score(X_test, Y_test)
```

```
0.5412594818321672
```

Evaluation

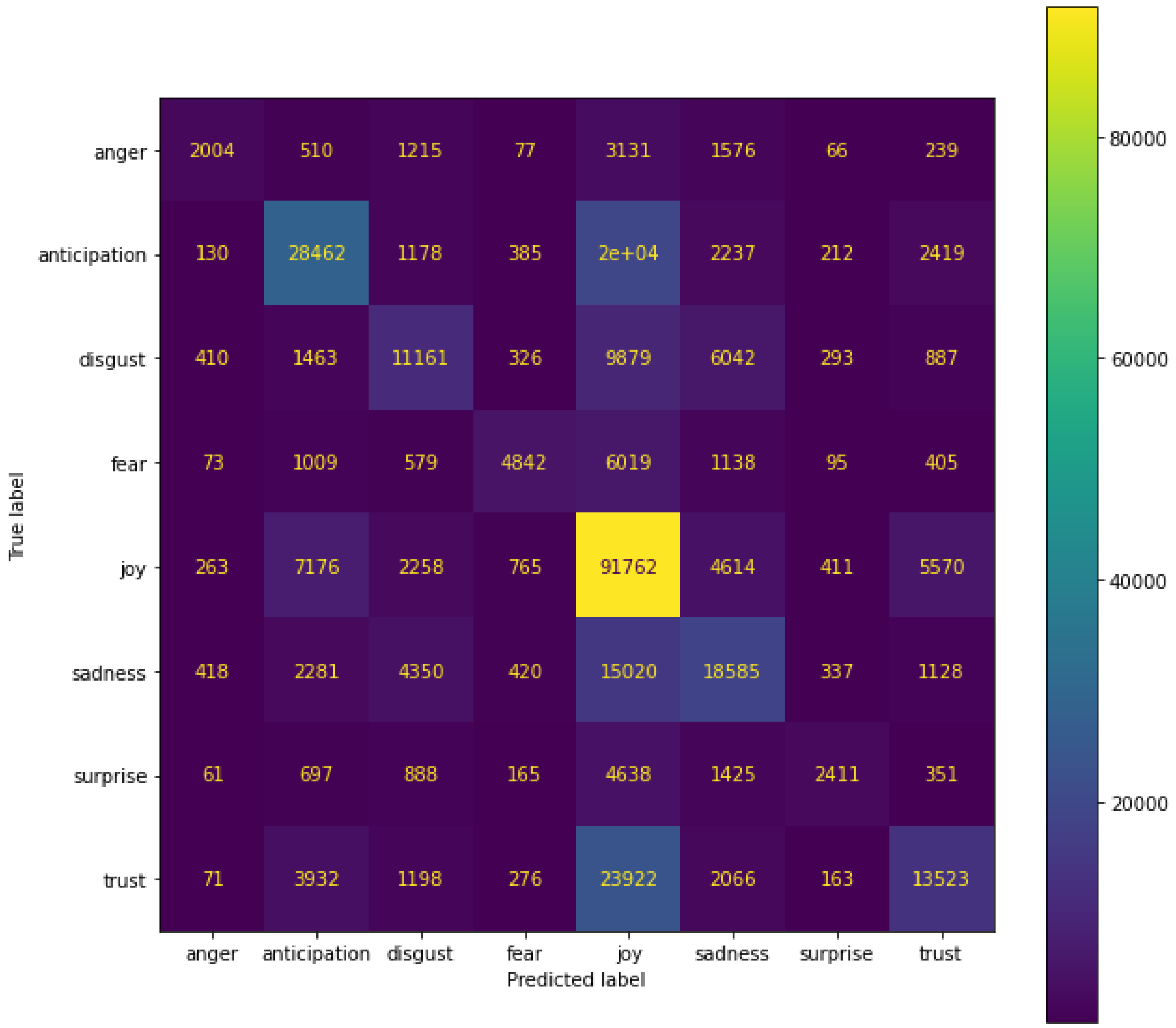


```
              precision    recall  f1-score   support

       anger       0.81      0.10      0.18      8818
anticipation       0.60      0.52      0.56     54579
     disgust       0.47      0.35      0.40     30461
        fear       0.74      0.22      0.34     14160
         joy       0.51      0.80      0.63    112819
     sadness       0.45      0.46      0.45     42539
    surprise       0.75      0.12      0.21     10636
       trust       0.53      0.26      0.34     45151

    accuracy                           0.52    319163
   macro avg       0.61      0.35      0.39    319163
weighted avg       0.54      0.52      0.49    319163
```

Once again, very high recall for joy compared to other emotions implies that model is struggling because of skew in distribution



True label \ Predicted label	anger	anticipation	disgust	fear	joy	sadness	surprise	trust
anger	2004	510	1215	77	3131	1576	66	239
anticipation	130	28462	1178	385	2e+04	2237	212	2419
disgust	410	1463	11161	326	9879	6042	293	887
fear	73	1009	579	4842	6019	1138	95	405
joy	263	7176	2258	765	91762	4614	411	5570
sadness	418	2281	4350	420	15020	18585	337	1128
surprise	61	697	888	165	4638	1425	2411	351
trust	71	3932	1198	276	23922	2066	163	13523

Confucian Matrix for Multinomial Naïve Bayes Classifier