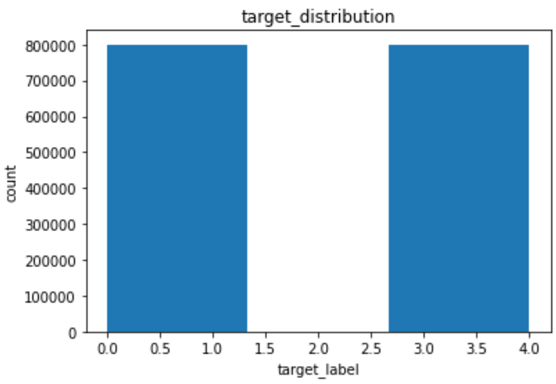
Q7 Sentiment Analysis and Opinion Mining

Author: Jiyao Wang 20797324
Date: 2021.12.13

Task 1, after check label distribution, I find out there is an imbalance in target 2. There are no target 2 and similar number on other two classes.




```
data['target'].value_counts()
```

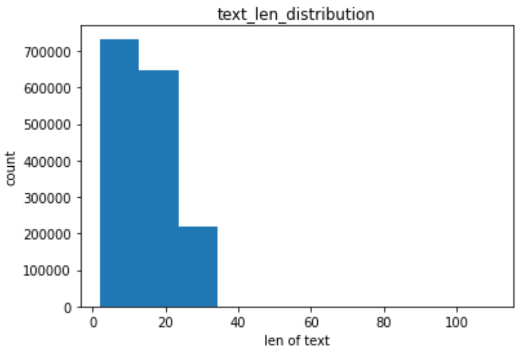
```
4    800000
0    800000
Name: target, dtype: int64
```

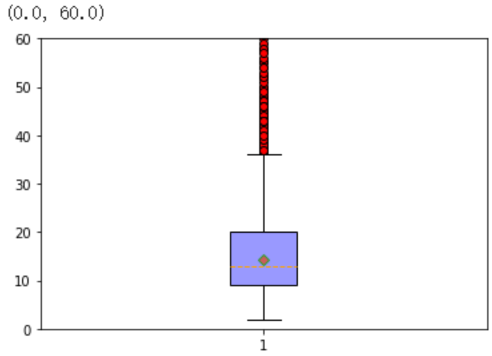
Task 2, each record has ['id','date','flag','user','text'] as its features. As we know, id can identify each record, so it unworthy to keep. For date, I find there are many different values, but there are also some records share same date, it is worthy to keep. And for flag, I find there is only one value in all of records, so I delete this feature. For user, it means user's name, we cannot get enough information from such short and meaningless text, so I delete it too.

```
Mon Jun 15 12:53:14 PDT 2009    20
Fri May 22 05:10:17 PDT 2009    17
Mon Jun 15 13:39:50 PDT 2009    17
Fri May 29 13:40:04 PDT 2009    17
Fri Jun 05 14:13:07 PDT 2009    16
                                ..
Mon Apr 20 04:09:04 PDT 2009     1
Wed Jun 17 23:13:43 PDT 2009     1
Sun May 17 10:37:52 PDT 2009     1
Thu Jun 25 08:43:59 PDT 2009     1
Fri May 22 01:40:14 PDT 2009     1
Name: date, Length: 774363, dtype: int64
```

```
NO_QUERY    1600000
Name: flag, dtype: int64
```

Task 3, for each sentence, the length distribution is following:





Task 4&5&6, result can refer to my Q7_clean.csv, the function is following

```
def clean_txt(text):
    text = BeautifulSoup(text,'html.parser').get_text() #remove html tag
    text = re.sub(r'[^a-zA-Z]',' ',text)
    text = re.sub(r"@\S+", "", text) #remove@ and following signal
    text = re.sub(r"http:\S+", "", text) #remove url
    text = text.lower()
    stopwords = nltk.corpus.stopwords.words('english')
    text = [s for s in text.replace(",", "").replace(".", "").split(' ') if s != '']
    words_list = [w for w in text if w not in stopwords]
    return words_list
```

Task 7, I utilize genism to generate word dictionary and word embedding with CBOW model. After prepocessing, I realize that max length of each text is 37, so I set the max length 30 for padding.

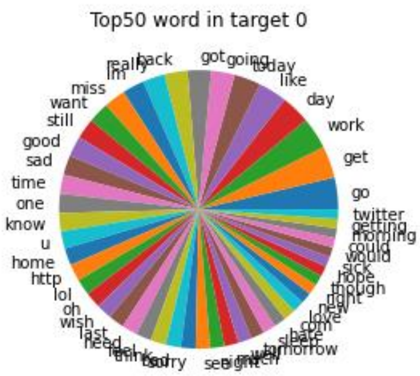
```
def get_index(sentence):
    sequence = []
    for word in sentence:
        try:
            sequence.append(word_dict[word])
        except KeyError:
            pass
    return sequence

def list_pad_2D(data, max_len, val):
    tmp = []
    for i in data:
        if (len(i) >= max_len):
            tmp.append(i[0:max_len])
        else:
            tmp.append(i + [val] * (max_len - len(i)))
    return np.array(tmp)
```

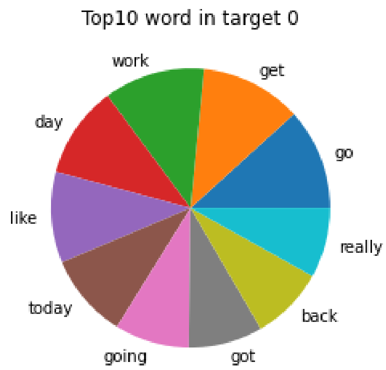
Task 8, the generated word cloud is following, we can see lol, love haha have big size. So I think it may reflect most of tweets have positive emotion.

Task 9&10, following is the word dictionary of each class and the bar chart of top50 and top10 on class0. We can see in class4, frequent words have positive meaning but not in class0.

	key	positive
494	good	62161
84	day	48366
0	love	47868
180	http	47218
65	like	37529

	key	negative
32	go	45661
126	get	45622
189	work	45484
12	day	41482
38	like	41062





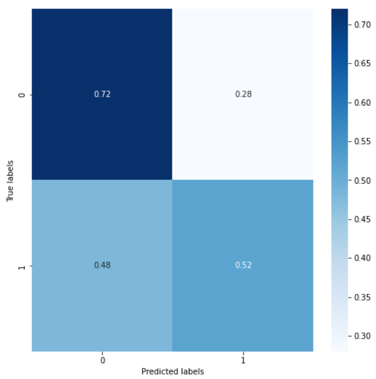
Task 11, for ML model, I set RandomForest and XGBoost to train on word vector without word embedding. And I set two NNs models: BiLSTM and XMLCNN to compare. For NNs models, I initialize word embedding layer with previous pre-trained embedding matrix.

```
BiLSTM(
  (emb): Embedding(135319, 64, padding_idx=0)
  (lstm): LSTM(64, 128, num_layers=2, batch_first=True, dropout=0.2, bidirectional=True)
  (L1): Linear(in_features=256, out_features=128, bias=True)
  (L2): Linear(in_features=128, out_features=2, bias=True)
)
```

```
XMLCNN(
  (emb): Embedding(135319, 64, padding_idx=0)
  (conv1): Conv2d(1, 128, kernel_size=(2, 64), stride=(1, 1), padding=(1, 0))
  (conv2): Conv2d(1, 128, kernel_size=(4, 64), stride=(1, 1), padding=(3, 0))
  (conv3): Conv2d(1, 128, kernel_size=(8, 64), stride=(1, 1), padding=(7, 0))
  (pool): AdaptiveMaxPool1d(output_size=300)
  (mlp): Sequential(
    (0): Linear(in_features=115200, out_features=512, bias=True)
    (1): Tanh()
    (2): Linear(in_features=512, out_features=64, bias=True)
  )
)
```

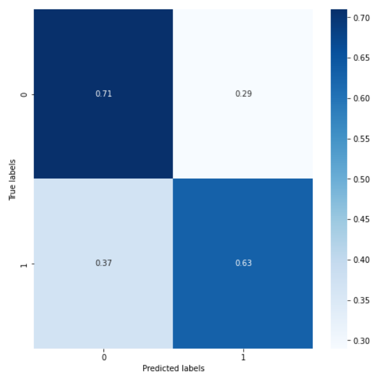
Task 12, for each model, I evaluate it on validation data by Accuracy, Recall, F1 Score and AUC Curve. Following are their performance:

	precision	recall	f1-score	support
0	0.60	0.72	0.65	79836
4	0.65	0.52	0.58	80164
accuracy			0.62	160000
macro avg	0.62	0.62	0.61	160000
weighted avg	0.62	0.62	0.61	160000



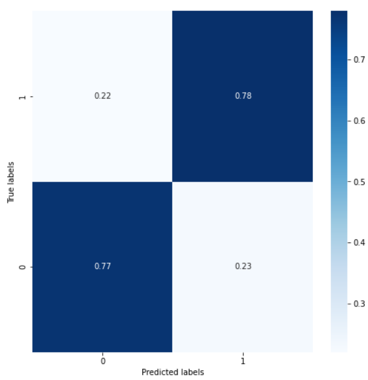
RandomForest

	precision	recall	f1-score	support
0	0.66	0.71	0.69	79836
4	0.69	0.63	0.66	80164
accuracy			0.67	160000
macro avg	0.67	0.67	0.67	160000
weighted avg	0.67	0.67	0.67	160000



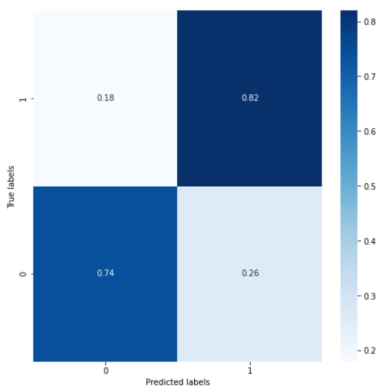
XGBoost

	precision	recall	f1-score	support
0	0.78	0.77	0.78	79836
1	0.78	0.78	0.78	80164
accuracy			0.78	160000
macro avg	0.78	0.78	0.78	160000
weighted avg	0.78	0.78	0.78	160000



BiLSTM

	precision	recall	f1-score	support
0	0.81	0.74	0.77	79836
1	0.76	0.82	0.79	80164
accuracy			0.78	160000
macro avg	0.78	0.78	0.78	160000
weighted avg	0.78	0.78	0.78	160000



XMLCNN

Model	Accuracy
RandomForest	0.62

XGBoost	0.67
BiLSTM	0.78
XMLCNN	0.78

From above charts, we can see NNs models outperform ML models. For all of models, because of limitation on time, they can achieve a better performance if I adjust parameters by AutoML or empirical methods, limited by time. All detail please refer to my notebook or output files.

Reference:

[1] http://nyc.lti.cs.cmu.edu/yiming/Publications/jliu-sigir17.pdf