Toxic Comment Classification

Nishant Aman

27 May 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

The aim of the project is to detect potentially toxic comments on Wikipedia. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments.

The multi headed model should be able of classifying comments into 6 categories based on their toxicity(threat,obscene,toxic,severe toxic,insult,hate speech) and to find patterns in the sentences using NLP techniques.

1.2 Data

We have been provided with 2 datasets i.e. Training and Test.
The training dataset contains 159571 observations of 6 variables.
id – unique id for comment text column.
Comment_text- Sentence to be classified.
threat,obscene,toxic,severe toxic,insult,hate speech- Categories to which comment text to be classified.

Our task is to build classification models which will predict probability of each type of comment.
Given below is a sample of the data set that we are using to predict probability of different toxicity :

Figure 1.1: Sample train Data

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

The test dataset contains 153164 observations of 2 variables.
id – unique id for comment text column.
Comment_text- Sentence to be classified.

Figure 1.2: Sample test Data

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.

Chapter 2

Methodology

2.1 Exploratory Data Analysis

The summary of train dataset is shown below for various categories is shown as follows:

Fig 2.1 Summary

	toxic	severe_toxic	obscene	threat	insult	identity_hate
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

It can be seen that majority(around 75%) of dataset are labeled as '0' in each category.

The count of various categories present in the 'train' file:

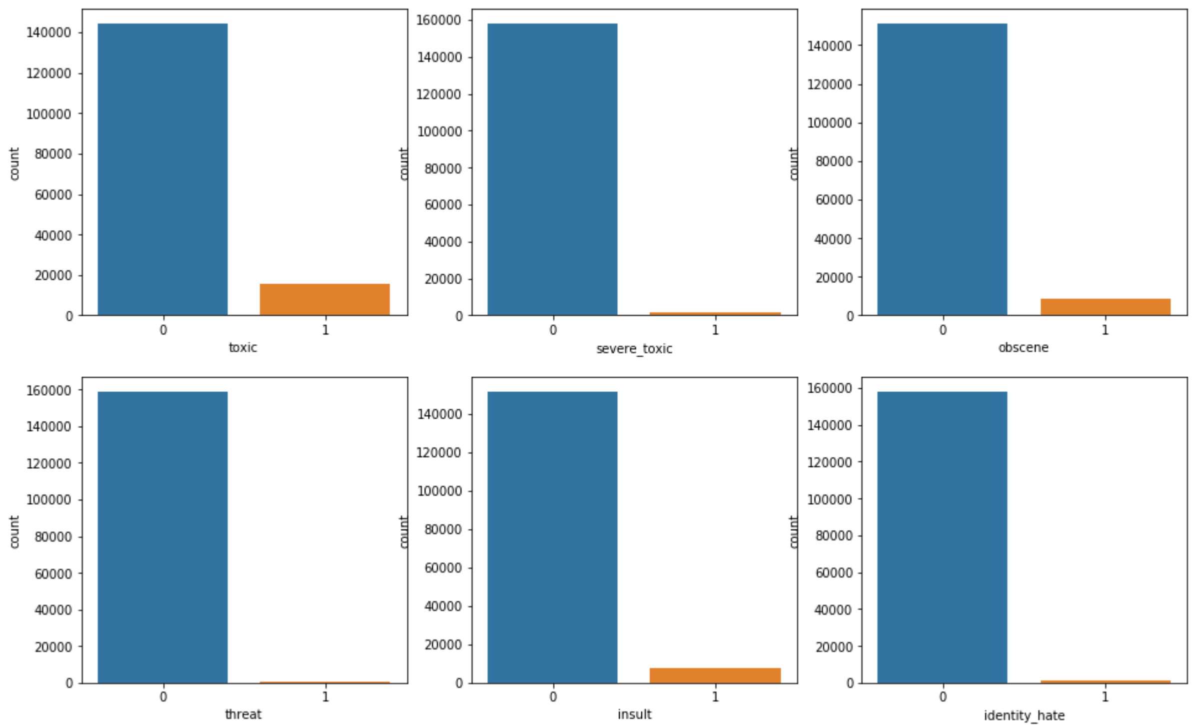
Fig 2.2 Counts of various categories

	count
none	143346
toxic	15294
obscene	8449
insult	7877
severe_toxic	1595
identity_hate	1405
threat	478

Note- None refers to the unlabeled columns

The distribution of various labeled categories as follows:

Fig 2.3 Data visualization in various categories



From above figure, we can see that data is highly imbalanced in all the categories.

2.2 Pre Processing

As we can see in above datasets, there are punctuation marks, Numbers, unwanted spaces which doesn't impart any information.
These are removed using regex,String and nltk packages.

2.2.1 Case Folding

The first preprocessing step is Case folding. Here, we are converting all the letters in the Corpus to lowercase using tolower() function.

2.2.2 Removing Stopwords

Stopwords are the commonly used words like "the", "is" etc, these words often does not help in the model building.
We have removed all the English stopwords from our corpus after evaluating that removing stopwords is helping our model to predict the output better.

2.2.3 Removing Numbers

Numbers are removed as it does not help in prediction.

2.2.4 Punctuation Marks
Punctuation marks like comma, fullstop are removed as it does not help in prediction.

2.2.5 Stemming
Stemming is the process of converting each word of the sentence to its root form by deleting or replacing suffixes.
Here we are using SnowballStemmer from nltk package for this purpose.

2.2.6 TF-IDF Vectorizer

Term frequency and inverse term frequency are defined as follows:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)

IDF(t) = log10(Total number of documents / Number of documents with term t in it)

IDF actually tells us how important the word is to the document. This is because when we calculate TF, we give equal importance to every single word. Now, if the word appears in the dataset more frequently, then its term frequency (TF) value is high while not being that important to the document.
So, if the word the appears in the document 100 times, then it's not carrying that much information compared to words that are less frequent in the dataset.

2.3 Vizualization

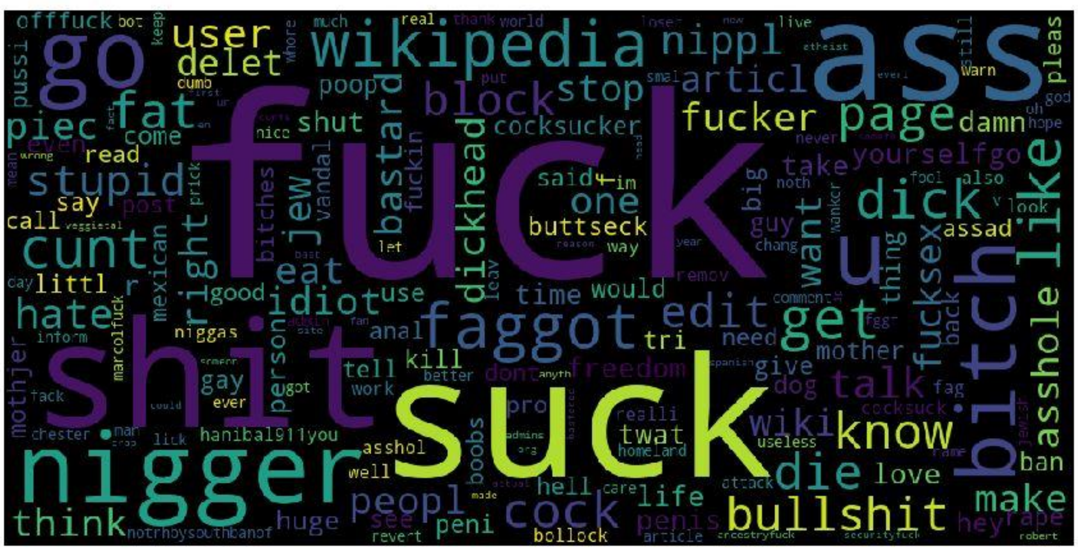
Word cloud for different categories is shown as follows:

Fig 2.4

Toxic

Severe_toxic

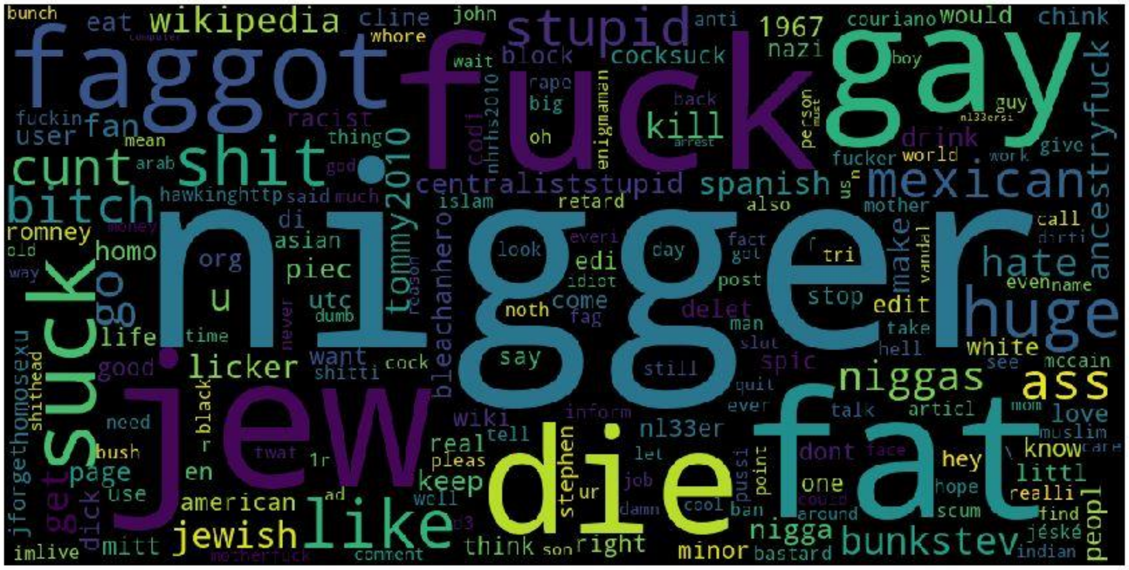
Obscene



Threat

Insult

Identity hate


nigger
fuck
gay
faggot
jew
fat
die
shit
suck
cunt
bitch
huge
mexican
stupid
like
niggas
ass
jewish
wikipedia
kill
hate
spanish
centraliststupid
ancestryfuck
bunkstev
nigga
tommy2010
bleachanhero
licker
american
cocksuck
homo
fan
go
get
u
org
piec
utc
asian
romney
mitt
page
keep
real
wiki
stephen
nl33er
edit
stop
make
white
love
know
hey
littl
peopl
one
dont
minor
right
think
say
want
edi
come
noth
block
nazi
1967
couriano
would
chink
cline
eat
john
bunch
anti
whore
wait
racist
drink
life
good
user
fuckin
mean
thing
oh
big
rape
boy
guy
give
world
fucker
person
retard
islam
hawkinghttp
said
mother
call
also
look
day
fact
post
tri
even
name
take
hell
man
spic
see
mccain
muslim
talk
articl
ever
still
cock
shitti
dumb
time
need
bush
use
dick
en
american
ur
job
cool
ban
pussi
point
around
hope
scum
realli
find
jéské
indian
bastard
imlive
son
jforgethomosexu
shithead
black
twat
1r
ad
well
pleas
tell
inform
let
damn
quit
slut
delet
fag
vandal
enigmaman

Chapter 3

Modelling

3.1 Modelling and evaluation

We have the data in proper format that is fed into machine learning algorithms. Our data is in form of matrix where all the words are a feature and the values are tf-idf that we have calculated earlier. Now we will built multiple machine learning models on top of the data and compare the accuracy of each algorithm to determine the best fit for multi-label classification.

First we will build a model using Random forest algorithm and then used Naive Bayes algorithm, based on the popular Bayes' probability theorem..

Then we will build a model using logistic-regression algorithm since it performs very well when we have large amount of text data.

Various terms used below are defined as follows:

Accuracy = TP+TN/Total

Precision = TP/(TP+FP)

Recall = TP/(TP+FN)

where

TP = True positive means no of positive cases which are predicted positive

TN = True negative means no of negative cases which are predicted negative

FP = False positive means no of negative cases which are predicted positive

FN= False Negative means no of positive cases which are predicted negative

Support:

The support is the number of samples of the true response that lie in that class.

F1 score:

The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0.

3.1.1 Random Forest

The random forest algorithm generates a random subset of the data from the training dataset and uses this to generate a decision tree for each of the subsets of the data.

Upon using random forest with default parameter we get an accuracy for various categories as follows:

Fig 3.1 Accuracy

```
Processing toxic
Training accuracy is        95.28285763117415
Processing severe_toxic
Training accuracy is        98.98213031010843
Processing obscene
Training accuracy is        97.66801496420365
Processing threat
Training accuracy is        99.71514840767959
Processing insult
Training accuracy is        96.65204428492756
Processing identity_hate
Training accuracy is        99.17772840350177
```

Fig 3.2 Classification Report

```
Processing toxic
             precision    recall  f1-score   support

          0       0.96      0.99      0.97     47576
          1       0.84      0.62      0.72      5083

avg / total       0.95      0.95      0.95     52659

Processing severe_toxic
             precision    recall  f1-score   support

          0       0.99      1.00      0.99     52133
          1       0.47      0.08      0.14       526

avg / total       0.99      0.99      0.99     52659

Processing obscene
             precision    recall  f1-score   support

          0       0.98      0.99      0.99     49828
          1       0.86      0.67      0.75      2831

avg / total       0.97      0.98      0.98     52659

Processing threat
             precision    recall  f1-score   support

          0       1.00      1.00      1.00     52507
          1       0.64      0.06      0.11       152

avg / total       1.00      1.00      1.00     52659
```

```
Processing insult
             precision    recall  f1-score   support

          0       0.97      0.99      0.98     50016
          1       0.74      0.51      0.60      2643

avg / total       0.96      0.97      0.96     52659

Processing identity_hate
             precision    recall  f1-score   support

          0       0.99      1.00      1.00     52188
          1       0.62      0.10      0.17       471

avg / total       0.99      0.99      0.99     52659
```

3.1.2 Multinomial Naïve Bayes

Naïve bayes with default parameters gives an accuracy for various categories as follows:

Fig 3.3 Accuracy

```
Processing toxic
Training accuracy is        94.71885147837976
Processing severe_toxic
Training accuracy is        99.04099963918799
Processing obscene
Training accuracy is        96.91220873924685
Processing threat
Training accuracy is        99.70565335460225
Processing insult
Training accuracy is        96.47163827645797
Processing identity_hate
Training accuracy is        99.09986896826753
```

Fig 3.4 Classification report

```
Processing toxic
             precision    recall  f1-score   support

          0       0.95      1.00      0.97     47576
          1       0.94      0.48      0.64      5083

avg / total       0.95      0.95      0.94     52659

Processing severe_toxic
             precision    recall  f1-score   support

          0       0.99      1.00      1.00     52133
          1       0.71      0.07      0.12       526

avg / total       0.99      0.99      0.99     52659

Processing obscene
             precision    recall  f1-score   support

          0       0.97      1.00      0.98     49828
          1       0.92      0.47      0.62      2831

avg / total       0.97      0.97      0.96     52659

Processing threat
             precision    recall  f1-score   support

          0       1.00      1.00      1.00     52507
          1       0.00      0.00      0.00       152

avg / total       0.99      1.00      1.00     52659

Processing insult
             precision    recall  f1-score   support

          0       0.97      1.00      0.98     50016
          1       0.83      0.37      0.51      2643

avg / total       0.96      0.96      0.96     52659

Processing identity_hate
             precision    recall  f1-score   support

          0       0.99      1.00      1.00     52188
          1       0.20      0.00      0.00       471

avg / total       0.98      0.99      0.99     52659
```

3.1.3 Logistic Regression

Logistic regression with default parameters gives an accuracy for various categories as follows

Fig 3.5

```
Processing toxic
Training accuracy is        95.71203403027023
Processing severe_toxic
Training accuracy is        99.07897985149738
Processing obscene
Training accuracy is        97.74397538882242
Processing threat
Training accuracy is        99.73223950321882
Processing insult
Training accuracy is        96.93689587724795
Processing identity_hate
Training accuracy is        99.20051653088741
```

```
Processing toxic
             precision    recall  f1-score   support

          0       0.96      0.99      0.98     47576
          1       0.92      0.61      0.73      5083

avg / total       0.96      0.96      0.95     52659

Processing severe_toxic
             precision    recall  f1-score   support

          0       0.99      1.00      1.00     52133
          1       0.59      0.25      0.35       526

avg / total       0.99      0.99      0.99     52659

Processing obscene
             precision    recall  f1-score   support

          0       0.98      1.00      0.99     49828
          1       0.92      0.64      0.75      2831

avg / total       0.98      0.98      0.98     52659

Processing threat
             precision    recall  f1-score   support

          0       1.00      1.00      1.00     52507
          1       0.72      0.12      0.20       152

avg / total       1.00      1.00      1.00     52659
```

```
Processing insult
             precision    recall  f1-score   support

          0       0.97      0.99      0.98     50016
          1       0.81      0.50      0.62      2643

avg / total       0.97      0.97      0.97     52659

Processing identity_hate
             precision    recall  f1-score   support

          0       0.99      1.00      1.00     52188
          1       0.72      0.17      0.28       471

avg / total       0.99      0.99      0.99     52659
```

4. Conclusion

The accuracy of all the models are above 90% for various categories but the time consumption by Random forest is more than the Naïve Bayes and Logistic Regression.

Hence, Logistic regression and Naïve bayes are preferable over Random Forest.

After analyzing the conclusion reports and accuracy score of various model above , I have choosen Logistic regression over naïve bayes as it performs a little better compared to than Naïve Bayes.

5. References

https://www.analyticsvidhya.com/

https://www.kaggle.com/

https://www.datasciencecentral.com/

https://www.wikipedia.org/

APPENDIX

Python Code

```
#Set working directory
os.chdir("D:\Edwisor\Pro1")
```

```
# Labels to be classified
target = ['toxic','severe_toxic','obscene','threat','insult','identity_hate']
```

```
# Load training dataset
train = pd.read_csv("train.csv")
```

```
#Dataset vizualization
fig,ax = plt.subplots(2,3,figsize=(16,10))
ax1,ax2,ax3,ax4,ax5,ax6 = ax.flatten()
sns.countplot(train['toxic'],ax=ax1)
sns.countplot(train['severe_toxic'],ax=ax2)
sns.countplot(train['obscene'],ax=ax3)
sns.countplot(train['threat'],ax = ax4)
sns.countplot(train['insult'],ax=ax5)
sns.countplot(train['identity_hate'], ax = ax6)
```

```
# Load testing dataset
test = pd.read_csv("test.csv")
```

```
#Removal of stopwords
sw = stopwords.words('english')
def stp(text):
    text = [text.lower() for text in text.split() if text.lower() not in sw]
    return " ".join(text)
```

```
train['comment_text'] = train['comment_text'].apply(stp)
test['comment_text'] = test['comment_text'].apply(stp)
```

```
# Removal of numbers,punctuation marks
def clean_text(text):
    text = text.lower()
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"can't", "cannot ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"\'re", " are ", text)
    text = re.sub(r"\'d", " would ", text)
    text = re.sub(r"\'ll", " will ", text)
    text = re.sub(r"\'scuse", " excuse ", text)
    text = re.sub('\W', ' ', text)
    text = re.sub('\s+', ' ', text)
    text = text.strip(' ')
    return text
```

```
cleaned_train_comment = []
for i in range(0,len(train)):
    cleaned_comment = clean_text(train['comment_text'][i])
    cleaned_train_comment.append(cleaned_comment)
train['comment_text'] = pd.Series(cleaned_train_comment).astype(str)
```

```
cleaned_test_comment = []
for i in range(0,len(test)):
    cleaned_comment = clean_text(test['comment_text'][i])
    cleaned_test_comment.append(cleaned_comment)
test['comment_text'] = pd.Series(cleaned_test_comment).astype(str)
```

```
#Stemming
stemmer = SnowballStemmer("english")
def stemming(text):
    text = [stemmer.stem(word) for word in text.split()]
    return " ".join(text)
```

```
train['comment_text'] = train['comment_text'].apply(stemming)
test['comment_text'] = test['comment_text'].apply(stemming)
```

```
#WordCloud
word_counter = {}

def split_text(text):
    return ' '.join([word for word in text.split() if word not in (sw)])

for label in target:
  d = Counter()
  train[train[label] == 1]['comment_text'].apply(lambda t: d.update(split_text(t).split()))
  word_counter[label] = pd.DataFrame.from_dict(d, orient='index')\
                                        .rename(columns={0: 'count'})\
                                        .sort_values('count', ascending=False)
for label in target:
  print(label)
  wc = word_counter[label]
  wordcloud = WordCloud(width = 1000, height = 500,stopwords=STOPWORDS, background_color = 'black').generate_from_frequencies(
                         wc.to_dict()['count'])

  plt.figure(figsize = (15,8))
  plt.imshow(wordcloud)
  plt.axis('off')
  plt.show()
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
vec_tf = TfidfVectorizer(stop_words='english',lowercase=True,
                max_features = 10000)
X_train_vec = vec_tf.fit_transform(X_train)
X_train_vec
```

```
<106912x10000 sparse matrix of type '<class 'numpy.float64'>'
	with 2373507 stored elements in Compressed Sparse Row format>
```

```
X_test_vec = vec_tf.transform(X_test)
X_full_test = vec_tf.transform(test.comment_text)
X_test_vec
```

```
<52659x10000 sparse matrix of type '<class 'numpy.float64'>'
	with 1166300 stored elements in Compressed Sparse Row format>
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import classification_report
lr = LogisticRegression()

# create submission file
submission = pd.read_csv("sample_submission.csv")

for label in target:
    print('Processing {}'.format(label))
    lr.fit(X_train_vec, y_train[label])
    y_pred1 = lr.predict(X_test_vec)
    print('Training accuracy is         {}'.format(accuracy_score(y_test[label], y_pred1)*100))
    test_y_prob = lr.predict_proba(X_full_test)[:,1]
    print(classification_report(y_test[label], y_pred1))
    submission[label] = test_y_prob
    submission.to_csv('submission.csv', index=False)
```

```
nb=MultinomialNB()

# create submission file
submission = pd.read_csv("sample_submission.csv")

for label in target:
    print('Processing {}'.format(label))
    nb.fit(X_train_vec, y_train[label])
    y_pred1 = nb.predict(X_test_vec)
    print('Training accuracy is         {}'.format(accuracy_score(y_test[label], y_pred1)*100))
    test_y_prob = nb.predict_proba(X_full_test)[:,1]
    print(classification_report(y_test[label], y_pred1))
    submission[label] = test_y_prob
    submission.to_csv('submission.csv', index=False)
```

```python
rf=RandomForestClassifier()

# create submission file
submission = pd.read_csv("sample_submission.csv")

for label in target:
    print('Processing {}'.format(label))
    rf.fit(X_train_vec, y_train[label])
    y_pred1 = rf.predict(X_test_vec)
    print('Training accuracy is         {}'.format(accuracy_score(y_test[label], y_pred1)*100))
    test_y_prob = nb.predict_proba(X_full_test)[:,1]
    print(classification_report(y_test[label], y_pred1))
    submission[label] = test_y_prob
    submission.to_csv('submission.csv', index=False)
```