[2]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/jigsaw-toxic-comment-classification-challenge/train.csv.zip
/kaggle/input/jigsaw-toxic-comment-classification-challenge/sample_submission.csv.zip
/kaggle/input/jigsaw-toxic-comment-classification-challenge/test_labels.csv.zip
/kaggle/input/jigsaw-toxic-comment-classification-challenge/test.csv.zip
```

+ Code + Markdown

Introduction

Let's filter out the hate from the comments and tag the comments in their category.

Lets import the necessary Libraries

[3]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter
import re
from sklearn.feature_extraction.text import CountVectorizer
from wordcloud import WordCloud
import spacy
import nltk
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
```

+ Code + Markdown

Lets import some more important Libraries

[ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import csv
import re
import string
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

Lets import the data and examine it

[11]:
```
train = pd.read_csv("/kaggle/input/jigsaw-toxic-comment-classification-challenge/train.csv.zip")
test = pd.read_csv("/kaggle/input/jigsaw-toxic-comment-classification-challenge/test.csv.zip")
train.head()
train.tail(20)
```

[11]:

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
159551	ffbc2db4225258dd	While about half the references are from BYU-I...	0	0	0	0	0	0
159552	ffbcd64a71775e04	Prague Spring \n\nI think that Prague Spring d...	0	0	0	0	0	0
159553	ffbd331a3aa269b9	I see this as having been merged; undoing one ...	0	0	0	0	0	0
159554	ffbdbb0483ed0841	and i'm going to keep posting the stuff u dele...	1	0	1	0	1	0
159555	ffc2f40965857 1f1	"\n\nHow come when you download that MP3 it's ...	0	0	0	0	0	0
159556	ffc671f2acdd80e1	I'll be on IRC, too, if you have a more specif...	0	0	0	0	0	0
159557	ffc7bbb177c3c966	It is my opinion that that happens to be off-t...	0	0	0	0	0	0
159558	ffca1e81aefc48ac	Please stop removing content from Wikipedia; i...	0	0	0	0	0	0
159559	ffca8d71d71a3fae	Image:Barack-obama-mother.jpg listed for delet...	0	0	0	0	0	0
159560	ffcdcb71854f6d8a	"Editing of article without Consensus & Remova...	0	0	0	0	0	0
159561	ffd2e85b07b3c7e4	"\nNo he did not, read it again (I would have ...	0	0	0	0	0	0
159562	ffd72e9766c09c97	"\n Auto guides and the motoring press are not...	0	0	0	0	0	0
159563	ffe029a7c79dc7fe	"\nplease identify what part of BLP applies be...	0	0	0	0	0	0
159564	ffe897e7f7182c90	Catalan independentism is the social movement ...	0	0	0	0	0	0
159565	ffe8b9316245be30	The numbers in parentheses are the additional ...	0	0	0	0	0	0
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

[12]:
```
test.head(10)
```

[12]:

	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.
5	0001ea8717f6de06	Thank you for understanding. I think very high...
6	00024115d4cbde0f	Please do not add nonsense to Wikipedia. Such ...
7	000247e83dcc1211	:Dear god this site is horrible.
8	00025358d4737918	" \n Only a fool can believe in such numbers. ...
9	00026d1092fe71cc	== Double Redirects == \n\n When fixing double...

```
train.shape
```

```
(159571, 8)
```

```
test.shape
```

```
(153164, 2)
```

In the training data, the comments are labelled as one or more of the six categories; toxic, severe toxic, obscene, threat, insult and identity hate. This is essentially a multi-label classification problem.

```
train.describe()
```

	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

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
 #   Column         Non-Null Count   Dtype 
---  ------         --------------   ----- 
 0   id             159571 non-null  object
 1   comment_text   159571 non-null  object
 2   toxic          159571 non-null  int64 
 3   severe_toxic   159571 non-null  int64 
 4   obscene        159571 non-null  int64 
 5   threat         159571 non-null  int64 
 6   insult         159571 non-null  int64 
 7   identity_hate  159571 non-null  int64 
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```

```
train.nunique()
```

```
id               159571
comment_text     159571
toxic                 2
severe_toxic          2
obscene               2
threat                2
insult                2
identity_hate         2
dtype: int64
```

Lets Discover the NULL values Below

```
total_null = train.isnull().sum().sort_values(ascending = False)
percent = ((train.isnull().sum()/train.isnull().count())*100).sort_values(ascending = False)
print("Total records = ", train.shape[0])

missing_data = pd.concat([total_null,percent.round(2)],axis=1,keys=['Total Missing','In Percent'])
missing_data
```

```
Total records =  159571
```

	Total Missing	In Percent
id	0	0.0
comment_text	0	0.0
toxic	0	0.0
severe_toxic	0	0.0
obscene	0	0.0
threat	0	0.0
insult	0	0.0
identity_hate	0	0.0

```
total_null = test.isnull().sum().sort_values(ascending = False)
percent = ((test.isnull().sum()/test.isnull().count())*100).sort_values(ascending = False)
print("Total records = ", test.shape[0])

missing_data = pd.concat([total_null,percent.round(2)],axis=1,keys=['Total Missing','In Percent'])
missing_data
```

```
Total records =  153164
```

	Total Missing	In Percent
id	0	0.0
comment_text	0	0.0

So no NULL values present in both test and train data

Lets check how many features have the label 1

```
cols_target = ['obscene','insult','toxic','severe_toxic','identity_hate','threat']
print(train[cols_target].sum())
```

```
obscene          8449
insult           7877
toxic           15294
severe_toxic     1595
identity_hate    1405
threat            478
dtype: int64
```

```
train.shape
```

```
(159571, 8)
```

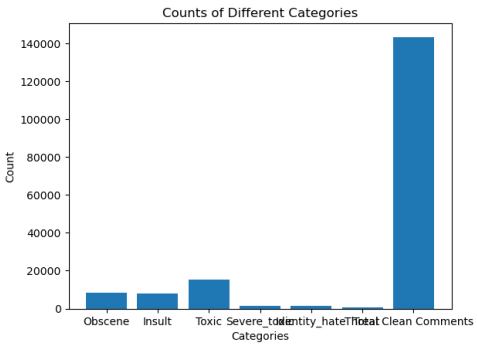
We can see that very few comments are labelled in each category

[24]:
```
x=train.iloc[:,2:].sum()
#marking comments without any tags as "clean"
rowsums=train.iloc[:,2:].sum(axis=1)
train['clean']=(rowsums==0)
#count number of clean entries
train['clean'].sum()
print("Total comments = ",len(train))
print("Total clean comments = ",train['clean'].sum())
print("Total tags =",x.sum())
```

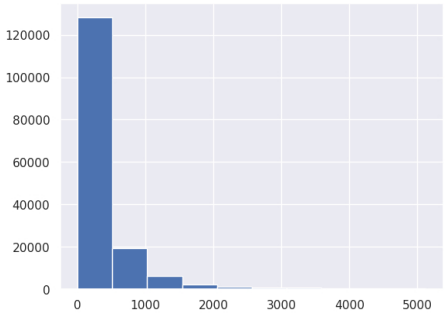
```
Total comments =  159571
Total clean comments =  143346
Total tags = 35098
```

Lets plot the Barplot

[29]:
```
# Data
categories = ['Obscene', 'Insult', 'Toxic', 'Severe_toxic', 'Identity_hate', 'Threat', 'Total Clean Comments']
counts = [8449, 7877, 15294, 1595, 1405, 478, 143346]

# Create the bar plot
plt.bar(categories, counts)

# Labeling the axes and giving a title
plt.xlabel('Categories')
plt.ylabel('Count')
plt.title('Counts of Different Categories')

# Show the plot
plt.show()
```



[41]:
```
# Let's look at the character length for the rows in the training data and record these
train['char_length'] = train['comment_text'].apply(lambda x: len(str(x)))
# look at the histogram plot for text length
sns.set()
train['char_length'].hist()
plt.show()
```

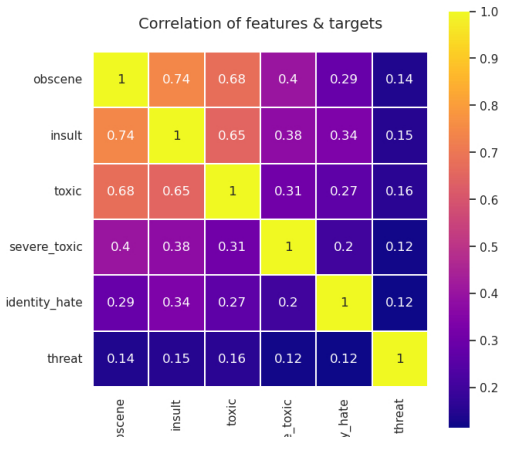


Most of the text length are within 500 characters, with some up to 5,000 characters long.

Lets see the correlation matrix

[34]:
```
data = train[cols_target]
colormap = plt.cm.plasma
plt.figure(figsize=(7,7))
plt.title('Correlation of features & targets',y=1.05,size=14)
sns.heatmap(data.astype(float).corr(),linewidths=0.1,vmax=1.0,square=True,cmap=colormap,
            linecolor='white',annot=True)
```

[34]:
```
<Axes: title={'center': 'Correlation of features & targets'}>
```



ot severe identit

Indeed, it looks like some of the labels are higher correlated. e.g. insult-obscene has the highest at 0.74, followed by toxic-obscene and toxic-insult

Clean up the comment text

[36]:
```
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
```

[37]:
```
train['comment_text'] = train['comment_text'].map(lambda com : clean_text(com))
test['comment_text'] = test['comment_text'].map(lambda com : clean_text(com))
```

[42]:
```
train.head()
```

[42]:

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate	clean	char_length
0	0000997932d777bf	explanation why the edits made under my userna...	0	0	0	0	0	0	True	264
1	000103f0d9cfb60f	d aww he matches this background colour i am s...	0	0	0	0	0	0	True	103
2	000113f07ec002fd	hey man i am really not trying to edit war it ...	0	0	0	0	0	0	True	228
3	0001b41b1c6bb37e	more i cannot make any real suggestions on imp...	0	0	0	0	0	0	True	600
4	0001d958c54c6e35	you sir are my hero any chance you remember wh...	0	0	0	0	0	0	True	61

[43]:
```
train = train.drop('char_length',axis=1) #we used char_length for plotting distribution now lets remove it
X = train.comment_text
test_X = test.comment_text
```

Vectorize the data

Instantiate the TfidfVectorizer object with the desired parameters. In this case, it sets max_features to 5000 (to limit the vocabulary size) and stop_words to 'english' (to remove common English stop words).

[44]:
```
# import and instantiate TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
vect = TfidfVectorizer(max_features=5000,stop_words='english')
vect
```

[44]:
```
TfidfVectorizer(max_features=5000, stop_words='english')
```

Fit and transform the training data X into a document-term matrix using the fit_transform method. The fit_transform method learns the vocabulary from the training data and transforms it into a matrix.

Examine the resulting document-term matrix X_dtm, which represents the TF-IDF representation of the training data.

[45]:
```
# learn the vocabulary in the training data, then use it to create a document-term matrix
X_dtm = vect.fit_transform(X)
# examine the document-term matrix created from X_train
X_dtm
```

[45]:
```
<159571x5000 sparse matrix of type '<class 'numpy.float64'>'
	with 3178792 stored elements in Compressed Sparse Row format>
```

Transform the test data test_X using the vocabulary learned from the training data into a document-term matrix using the transform method.

Examine the resulting document-term matrix test_X_dtm, which represents the TF-IDF representation of the test data using the same vocabulary learned from the training data.

[46]:
```
# transform the test data using the earlier fitted vocabulary, into a document-term matrix
test_X_dtm = vect.transform(test_X)
# examine the document-term matrix from X_test
test_X_dtm
```

[46]:
```
<153164x5000 sparse matrix of type '<class 'numpy.float64'>'
	with 2618972 stored elements in Compressed Sparse Row format>
```

The resulting X_dtm and test_X_dtm matrices can be used as input to machine learning models for various NLP tasks, such as text classification or sentiment analysis. The TF-IDF representation assigns weights to each term based on its frequency in a specific document and its inverse frequency across the entire corpus, which helps capture the importance of words in differentiating documents.

Solving a multi-label classification problem

One way to approach a multi-label classification problem is to transform the problem into separate single-class classifier problems. This is known as 'problem transformation'. There are three methods:

Binary Relevance This is probably the simplest which treats each label as a separate single classification problems. The key assumption here though, is that there are no correlation among the various labels.

Classifier Chains In this method, the first classifier is trained on the input X. Then the subsequent classifiers are trained on the input X and all previous classifiers' predictions in the chain. This method attempts to draw the signals from the correlation among preceding target variables.

Label Powerset This method transforms the problem into a multi-class problem where the multi-class labels are essentially all the unique label combinations. In our case here, where there are six labels, Label Powerset would in effect turn this into a 2^6 or 64-class problem. (Thanks Joshua for pointing out.)

Binary Relevance - build a multi-label classifier using Logistic Regression

[48]:
```
# import and instantiate the Logistic Regression model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
logreg = LogisticRegression(C=12.0)

# create submission file
submission_binary = pd.read_csv('/kaggle/input/jigsaw-toxic-comment-classification-challenge/sample_submission.csv.zip')

for label in cols_target:
    print('... Processing {}'.format(label))
    y = train[label]
    # train the model using X_dtm & y
    logreg.fit(X_dtm, y)
    # compute the training accuracy
    y_pred_X = logreg.predict(X_dtm)
    print('Training accuracy is {}'.format(accuracy_score(y, y_pred_X)))
    # compute the predicted probabilities for X_test_dtm
    test_y_prob = logreg.predict_proba(test_X_dtm)[:,1]
    submission_binary[label] = test_y_prob
```

```
... Processing obscene
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Training accuracy is 0.9832425691385026
... Processing insult
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Training accuracy is 0.9755344016143285
... Processing toxic
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Training accuracy is 0.9639470831166064
... Processing severe_toxic
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Training accuracy is 0.9920724943755445
... Processing identity_hate
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Training accuracy is 0.9939901360522903
... Processing threat
Training accuracy is 0.9981199591404453
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

[49]:
```
submission_binary
```

[49]:

	id	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	00001cee341fdb12	0.999957	0.460334	0.999965	0.050124	0.972259	0.444720
1	0000247867823ef7	0.002463	0.000443	0.000401	0.000324	0.003252	0.000408
2	00013b17ad220c46	0.010776	0.000117	0.003201	0.000035	0.007652	0.001218
3	00017563c3f7919a	0.001342	0.002159	0.000973	0.000117	0.000835	0.000033
4	00017695ad8997eb	0.019448	0.000813	0.001246	0.000602	0.003271	0.000565
...	...	...	...	...	...	...	...
153159	fffcd0960ee309b5	0.625931	0.000978	0.061163	0.000792	0.023834	0.001451
153160	fffd7a9a6eb32c16	0.031912	0.001002	0.036477	0.001816	0.029426	0.017003
153161	fffda9e8d6fafa9e	0.001122	0.000177	0.005382	0.000108	0.004031	0.000149
153162	fffe8f1340a79fc2	0.007861	0.000326	0.016519	0.000996	0.016097	0.015795
153163	ffffce3fb183ee80	0.999370	0.000337	0.981467	0.005147	0.696680	0.006571

153164 rows × 7 columns

[56]:
```
# generate submission file
submission_binary.to_csv('submission_binary.csv',index=False)
```

Classifier Chains - build a multi-label classifier using Logistic Regression

[51]:
```
# create submission file
submission_chains = pd.read_csv('/kaggle/input/jigsaw-toxic-comment-classification-challenge/sample_submission.csv.zip')

# create a function to add features
def add_feature(X, feature_to_add):
    '''
    Returns sparse feature matrix with added feature.
    feature_to_add can also be a list of features.
    '''
    from scipy.sparse import csr_matrix, hstack
    return hstack([X, csr_matrix(feature_to_add).T], 'csr')
```

[52]:
```
for label in cols_target:
    print('... Processing {}'.format(label))
    y = train[label]
    # train the model using X_dtm & y
    logreg.fit(X_dtm,y)
    # compute the training accuracy
    y_pred_X = logreg.predict(X_dtm)
    print('Training Accuracy is {}'.format(accuracy_score(y,y_pred_X)))
    # make predictions from test_X
    test_y = logreg.predict(test_X_dtm)
    test_y_prob = logreg.predict_proba(test_X_dtm)[:,1]
    submission_chains[label] = test_y_prob
    # chain current label to X_dtm
    X_dtm = add_feature(X_dtm, y)
    print('Shape of X_dtm is now {}'.format(X_dtm.shape))
    # chain current label predictions to test_X_dtm
    test_X_dtm = add_feature(test_X_dtm, test_y)
```

```
    print('Shape of test_X_dtm is now {}'.format(test_X_dtm.shape))
```

```
... Processing obscene
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Training Accuracy is 0.9832425691385026
Shape of X_dtm is now (159571, 5001)
Shape of test_X_dtm is now (153164, 5001)
... Processing insult
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Training Accuracy is 0.9818137380852411
Shape of X_dtm is now (159571, 5002)
Shape of test_X_dtm is now (153164, 5002)
... Processing toxic
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Training Accuracy is 0.9675943623841425
Shape of X_dtm is now (159571, 5003)
Shape of test_X_dtm is now (153164, 5003)
... Processing severe_toxic
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Training Accuracy is 0.9930689160311084
Shape of X_dtm is now (159571, 5004)
Shape of test_X_dtm is now (153164, 5004)
... Processing identity_hate
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Training Accuracy is 0.9955568367685858
Shape of X_dtm is now (159571, 5005)
Shape of test_X_dtm is now (153164, 5005)
... Processing threat
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Training Accuracy is 0.9985649021438733
Shape of X_dtm is now (159571, 5006)
Shape of test_X_dtm is now (153164, 5006)
```

[53]:
```
submission_chains.head()
```

[53]:

	id	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	00001cee341fdb12	0.999969	4.157091e-01	0.999965	0.193270	0.893696	0.596652
1	0000247867823ef7	0.002614	8.432571e-10	0.000401	0.000072	0.003678	0.000135
2	00013b17ad220c46	0.008356	2.339971e-10	0.003201	0.000009	0.004084	0.000525
3	00017563c3f7919a	0.001136	1.283481e-08	0.000973	0.000063	0.000569	0.000004
4	00017695ad8997eb	0.021045	1.231601e-09	0.001246	0.000158	0.001547	0.000102

[55]:
```
# generate submission file
submission_chains.to_csv('submission_chains.csv', index=False)
```

[58]:
```
# create submission file
submission_combined = pd.read_csv('/kaggle/input/jigsaw-toxic-comment-classification-challenge/sample_submission.csv.zip')
```

[59]:
```
for label in cols_target:
    submission_combined[label] = 0.5*(submission_chains[label]+submission_binary[label])
```

[60]:
```
submission_combined.head()
```

[60]:

	id	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	00001cee341fdb12	0.999963	0.438022	0.999965	0.121697	0.932977	0.520686
1	0000247867823ef7	0.002539	0.000222	0.000401	0.000198	0.003465	0.000271
2	00013b17ad220c46	0.009566	0.000059	0.003201	0.000022	0.005868	0.000872
3	00017563c3f7919a	0.001239	0.001080	0.000973	0.000090	0.000702	0.000018
4	00017695ad8997eb	0.020246	0.000406	0.001246	0.000380	0.002409	0.000334

```
# generate submission file
submission_combined.to_csv('submission_combined.csv', index=False)
```

+ Code + Markdown