In [ ]:
```
import numpy as np
import pandas as pd
from prettytable import PrettyTable
```

Importing Necessary Libraries

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Text Processing Libraries

In [ ]:
```
import re
import nltk
import string
import nlputils
import contractions
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import PorterStemmer, LancasterStemmer, SnowballStemmer
```

Data Visualization Libraries

In [ ]:
```
import plotly.express as px
import plotly.offline as pyo
import plotly.graph_objects as go
from wordcloud import WordCloud, STOPWORDS
```

Machine Learning Libraries

In [ ]:
```
import torch
from torch.utils.data import DataLoader, TensorDataset
from transformers import BertTokenizer, BertForSequenceClassification, AdamW
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.metrics import (
    f1_score, precision_score, recall_score, precision_recall_curve,
    fbeta_score, confusion_matrix, roc_auc_score, roc_curve
)
from sklearn import preprocessing
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.svm import LinearSVC
from sklearn.svm import SVC
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
```

Miscellaneous Libraries

In [ ]:
```
import os
import zipfile
import warnings
warnings.filterwarnings('ignore')
```

Data Loading and Extraction

In [ ]:
```
# Path to the Kaggle input directory
kaggle_input_path = '/media/gamedisk/Code/College/Chat-toxicity/Data/jigsaw-

# List files in the Kaggle input directory
files_in_directory = os.listdir(kaggle_input_path)

# Extract and load the data from the Kaggle zip files
for file_name in files_in_directory:
    if file_name.endswith('.zip'):
        zip_file_path = os.path.join(kaggle_input_path, file_name)
        output_dir = '/media/gamedisk/Code/College/Chat-toxicity/Data/output
        with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
            zip_ref.extractall(output_dir)
            extracted_files = zip_ref.namelist()
            for extracted_file in extracted_files:
                complete_path = os.path.join(output_dir, extracted_file)
                print("Extracted:", complete_path)

# Load the data into a Pandas DataFrame
data = pd.read_csv("/media/gamedisk/Code/College/Chat-toxicity/Data/output/t
data.sample(5)
```

```
Extracted: /media/gamedisk/Code/College/Chat-toxicity/Data/output/sample_sub
mission.csv
Extracted: /media/gamedisk/Code/College/Chat-toxicity/Data/output/test.csv
Extracted: /media/gamedisk/Code/College/Chat-toxicity/Data/output/test_label
s.csv
Extracted: /media/gamedisk/Code/College/Chat-toxicity/Data/output/train.csv
```

Out[ ]:

	id	comment_text	toxic	severe_toxic	obscene	thre
133101	c812ebcd014cb54f	Thanks for the advice \n\nThank you for taking ...	0	0	0	
68055	b60bfca3190b7b5e	The only BULLSHIT on Olbermann's article comes ...	1	0	1	
118458	78eacfeaf7c73238	SIOS\nI have been a Marine for ten years, and ...	0	0	0	
137369	df02065cd5d2cb74	People Killed \n\nWhy does this article say 73 ...	0	0	0	
105779	35f437fca3cc80bf	"\n\n \n\n Dear Administrator,\n this mornin ...	0	0	0	

Exploratory Data Analysis (EDA)

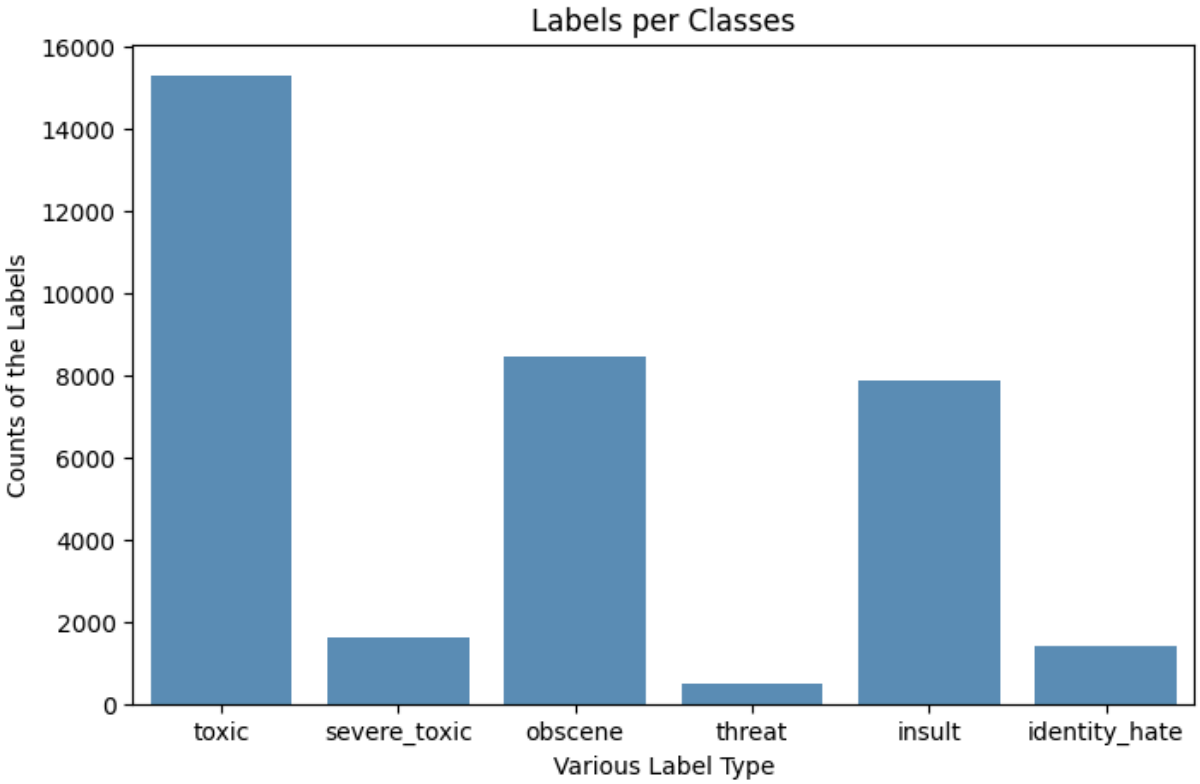
In [ ]:
```
# Display basic information about the dataset
data.info()
```

```
<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
```

In [ ]:
```
# Check for missing values
data.isnull().sum()
```

Out[ ]:
```
id               0
comment_text     0
toxic            0
severe_toxic     0
obscene          0
threat           0
insult           0
identity_hate    0
dtype: int64
```

In [ ]:
```
# Visualize label distribution
label_counts = data.iloc[:, 2:].sum()
plt.figure(figsize=(8, 5))
sns.barplot(x=label_counts.index, y=label_counts.values, alpha=0.8)
plt.title("Labels per Classes")
plt.xlabel("Various Label Type")
plt.ylabel("Counts of the Labels")
plt.show()
df = pd.DataFrame(data)
```



Data Cleaning

In [ ]:
```
# Sample comment before cleaning
df['comment_text'][10]
```

Out[ ]:
```
'"\nFair use rationale for Image:Wonju.jpg\n\nThanks for uploading Image:Wo
nju.jpg. I notice the image page specifies that the image is being used und
er fair use but there is no explanation or rationale as to why its use in W
ikipedia articles constitutes fair use. In addition to the boilerplate fair
use template, you must also write out on the image description page a speci
fic explanation or rationale for why using this image in each article is co
nsistent with fair use.\n\nPlease go to the image description page and edit
it to include a fair use rationale.\n\nIf you have uploaded other fair use
media, consider checking that you have specified the fair use rationale on
those pages too. You can find a list of \'image\' pages you have edited by
clicking on the ""my contributions"" link (it is located at the very top of
any Wikipedia page when you are logged in), and then selecting ""Image"" fr
om the dropdown box. Note that any fair use images uploaded after 4 May, 20
06, and lacking such an explanation will be deleted one week after they hav
e been uploaded, as described on criteria for speedy deletion. If you have
any questions please ask them at the Media copyright questions page. Thank
you. (talk • contribs • ) \nUnspecified source for Image:Wonju.jpg\n\nThank
s for uploading Image:Wonju.jpg. I noticed that the file\'s description pag
e currently doesn\'t specify who created the content, so the copyright stat
us is unclear. If you did not create this file yourself, then you will need
to specify the owner of the copyright. If you obtained it from a website, t
hen a link to the website from which it was taken, together with a restatem
ent of that website\'s terms of use of its content, is usually sufficient i
nformation. However, if the copyright holder is different from the website
\'s publisher, then their copyright should also be acknowledged.\n\nAs well
as adding the source, please add a proper copyright licensing tag if the fi
le doesn\'t have one already. If you created/took the picture, audio, or vi
deo then the  tag can be used to release it under the GFDL. If you believe
the media meets the criteria at Wikipedia:Fair use, use a tag such as  or o
ne of the other tags listed at Wikipedia:Image copyright tags#Fair use. See
Wikipedia:Image copyright tags for the full list of copyright tags that you
can use.\n\nIf you have uploaded other files, consider checking that you ha
ve specified their source and tagged them, too. You can find a list of file
s you have uploaded by following [ this link]. Unsourced and untagged image
s may be deleted one week after they have been tagged, as described on crit
eria for speedy deletion. If the image is copyrighted under a non-free lice
nse (per Wikipedia:Fair use) then the image will be deleted 48 hours after
. If you have any questions please ask them at the Media copyright question
s page. Thank you. (talk • contribs • ) "'
```

In [ ]:
```
# Data cleaning functions
alphanumeric = lambda x: re.sub('\w*\d\w*', ' ', x)
punc_lower = lambda x: re.sub('[%s]' % re.escape(string.punctuation), ' ', x
remove_n = lambda x: re.sub("\n", " ", x)
remove_non_ascii = lambda x: re.sub(r'[^\x00-\x7f]', r' ', x)

# Apply data cleaning functions to the 'comment_text' column
df['comment_text'] = df['comment_text'].map(alphanumeric).map(punc_lower).ma
```

Class Balancing

In [ ]:
```
print(label_counts)
```

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

In [ ]:
```
Insulting_comment_df=df.loc[:,['id','comment_text','insult']]
Threatening_comment_df=df.loc[:,['id','comment_text','threat']]
IdentityHate_comment_df=df.loc[:,['id','comment_text','identity_hate']]
Obscene_comment_df=df.loc[:,['id','comment_text','obscene']]
Severetoxic_comment_df=df.loc[:,['id','comment_text','severe_toxic']]
Toxic_comment_df=df.loc[:,['id','comment_text','toxic']]
```

In [ ]:
```
# Balancing the 'toxic' class
Toxic_comment_balanced_1 = Toxic_comment_df[Toxic_comment_df['toxic'] == 1].
Toxic_comment_balanced_0 = Toxic_comment_df[Toxic_comment_df['toxic'] == 0].
Toxic_comment_balanced = pd.concat([Toxic_comment_balanced_1,Toxic_comment_b

# Balancing the 'severe_toxic' class
Severetoxic_comment_df_1 = Severetoxic_comment_df[Severetoxic_comment_df['se
Severetoxic_comment_df_0 = Severetoxic_comment_df[Severetoxic_comment_df['se
Severe_toxic_comment_balanced = pd.concat([Severetoxic_comment_df_1,Severetc

# Balancing the 'obscene' class
Obscene_comment_df_1 = Obscene_comment_df[Obscene_comment_df['obscene'] == 1
Obscene_comment_df_0 = Obscene_comment_df[Obscene_comment_df['obscene'] == 0
Obscene_comment_balanced = pd.concat([Obscene_comment_df_1,Obscene_comment_d

# Balancing the 'threat' class
Threatening_comment_df_1 = Threatening_comment_df[Threatening_comment_df['th
Threatening_comment_df_0 = Threatening_comment_df[Threatening_comment_df['th
Threatening_comment_balanced = pd.concat([Threatening_comment_df_1,Threateni

# Balancing the 'insult' class
Insulting_comment_df_1 = Insulting_comment_df[Insulting_comment_df['insult']
Insulting_comment_df_0 = Insulting_comment_df[Insulting_comment_df['insult']
Insulting_comment_balanced = pd.concat([Insulting_comment_df_1,Insulting_com

# Balancing the 'identity_hate' class
IdentityHate_comment_df_1 = IdentityHate_comment_df[IdentityHate_comment_df[
IdentityHate_comment_df_0 = IdentityHate_comment_df[IdentityHate_comment_df[
IdentityHate_comment_balanced = pd.concat([IdentityHate_comment_df_1,Identit
```

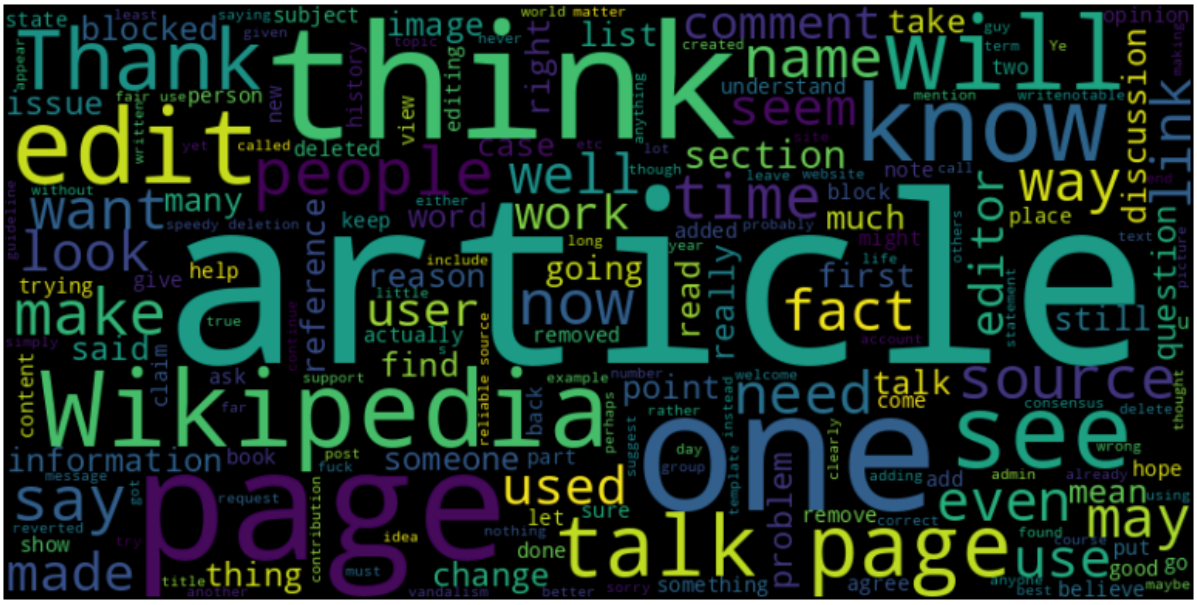
In [ ]:
```
combined_df = pd.concat([Severe_toxic_comment_balanced, Threatening_comment_
combined_df['combined_label'] = combined_df.iloc[:, 2:].sum(axis=1)
```

In [ ]:
```
def cv_tf_train_test(dataframe, label, vectorizer, ngram):
    # Split the data into X and y data sets
    X = dataframe.comment_text
    y = dataframe[label]

    # Split our data into training and test data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
```

```
    # Using vectorizer and removing stopwords
    cv1 = vectorizer(ngram_range=(ngram), stop_words='english')

    # Transforming x-train and x-test
    X_train_cv1 = cv1.fit_transform(X_train)
    X_test_cv1  = cv1.transform(X_test)

    ## Machine learning models

    ## Logistic regression
    lr = LogisticRegression()
    lr.fit(X_train_cv1, y_train)

    ## k-nearest neighbours
    knn = KNeighborsClassifier(n_neighbors=5)
    knn.fit(X_train_cv1, y_train)

    ## Naive Bayes
    bnb = BernoulliNB()
    bnb.fit(X_train_cv1, y_train)

    ## Multinomial naive bayes
    mnb = MultinomialNB()
    mnb.fit(X_train_cv1, y_train)

    ## Support vector machine
    svm_model = LinearSVC()
    svm_model.fit(X_train_cv1, y_train)

    ## Random Forest
    randomforest = RandomForestClassifier(n_estimators=100, random_state=50)
    randomforest.fit(X_train_cv1, y_train)

    f1_score_data = {'F1 Score':[f1_score(lr.predict(X_test_cv1), y_test), f
                                 f1_score(bnb.predict(X_test_cv1), y_test), f
                                 f1_score(svm_model.predict(X_test_cv1), y_te
    ## Saving f1 score results into a dataframe
    df_f1 = pd.DataFrame(f1_score_data, index=['Log Regression','KNN', 'Berr

    return df_f1
```

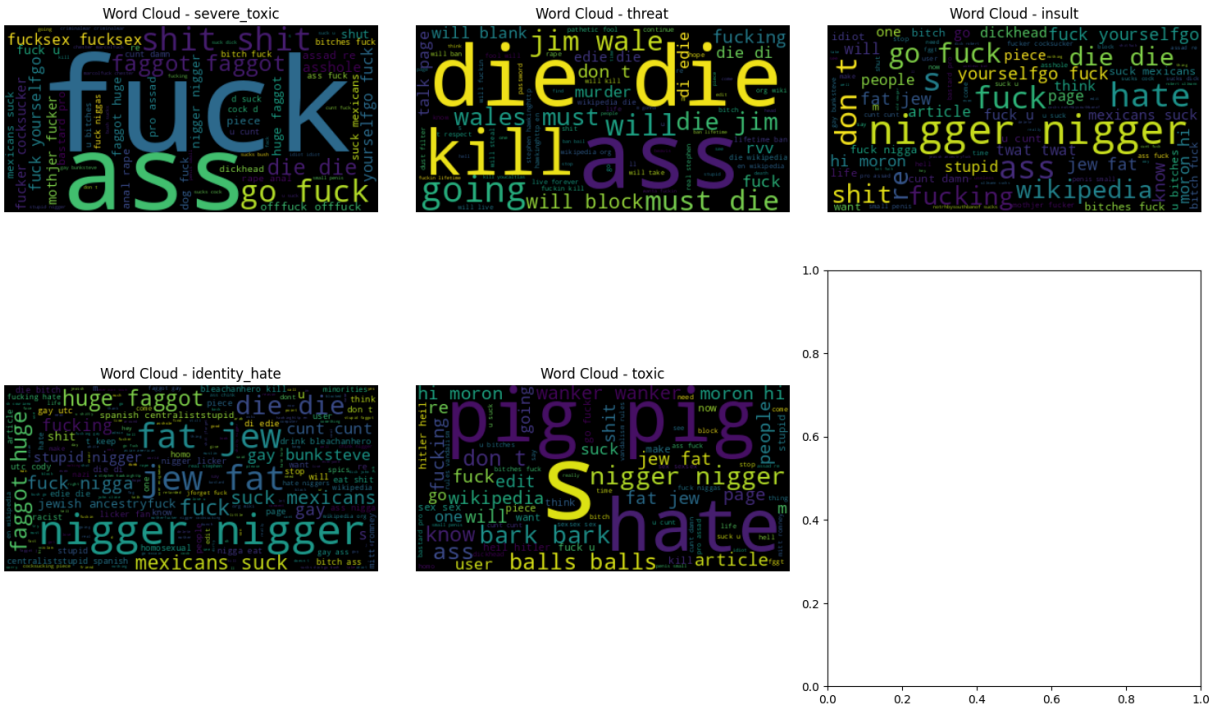
Word Cloud Visualization

In [ ]:
```
# Word frequency analysis
wordcloud = WordCloud(width=800, height=400, background_color='black').gener
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [ ]:

```
comment_types = ['severe_toxic', 'threat', 'insult', 'identity_hate', 'toxic
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
axes = axes.flatten()

for i, comment_type in enumerate(comment_types):

    comment_df = combined_df[combined_df[comment_type] == 1]
    wordcloud = WordCloud(width=400, height=200, background_color='black').g
    axes[i].imshow(wordcloud, interpolation='bilinear')
    axes[i].set_title(f'Word Cloud - {comment_type}')
    axes[i].axis('off')

plt.tight_layout()
plt.show()
```



Text Vectorization and Model Training

In [ ]:
```
# Model training and evaluation for 'severe_toxic' class
severe_toxic_comment_cv = cv_tf_train_test(Severe_toxic_comment_balanced, 's
severe_toxic_comment_cv.rename(columns={'F1 Score': 'F1 Score(severe_toxic)'
severe_toxic_comment_cv
```

Out[ ]:

	F1 Score(severe_toxic)
Log Regression	0.940282
KNN	0.860192
BernoulliNB	0.790738
MultinomialNB	0.932377
SVM	0.937901
Random Forest	0.941176

In [ ]:
```
# Model training and evaluation for 'threat' class
threat_comment_cv = cv_tf_train_test(Threatening_comment_balanced, 'threat',
threat_comment_cv.rename(columns={'F1 Score': 'F1 Score(threat)'}, inplace=T
threat_comment_cv
```

Out[ ]:

	F1 Score(threat)
Log Regression	0.751092
KNN	0.718615
BernoulliNB	0.881481
MultinomialNB	0.798354
SVM	0.851562
Random Forest	0.853755

In [ ]:
```
# Model training and evaluation for 'insult' class
insult_comment_cv = cv_tf_train_test(Insulting_comment_balanced, 'insult', T
insult_comment_cv.rename(columns={'F1 Score': 'F1 Score(insult)'}, inplace=T
insult_comment_cv
```

Out[ ]:

	F1 Score(insult)
Log Regression	0.902550
KNN	0.320661
BernoulliNB	0.776986
MultinomialNB	0.896299
SVM	0.906218
Random Forest	0.890821

In [ ]:
```
# Model training and evaluation for 'identity_hate' class
identity_hatecomment_cv = cv_tf_train_test(IdentityHate_comment_balanced, 'i
identity_hatecomment_cv.rename(columns={'F1 Score': 'F1 Score(identity_hate)
identity_hatecomment_cv
```

Out[ ]:

	F1 Score(identity_hate)
Log Regression	0.714706
KNN	0.271399
BernoulliNB	0.715976
MultinomialNB	0.564103
SVM	0.809399
Random Forest	0.846939

In [ ]:
```
X = Toxic_comment_balanced.comment_text
y = Toxic_comment_balanced['toxic']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran

# Initiate a Tfidf vectorizer
tfv = TfidfVectorizer(ngram_range=(1,1), stop_words='english')

X_train_fit = tfv.fit_transform(X_train)
X_test_fit = tfv.transform(X_test)
randomforest = RandomForestClassifier(n_estimators=100, random_state=50)

randomforest.fit(X_train_fit, y_train)
randomforest.predict(X_test_fit)
```

Out[ ]:
```
array([0, 1, 1, ..., 1, 1, 1])
```

In [ ]:
```
X_combined = combined_df.comment_text
y_combined = combined_df['combined_label']

X_train_combined, X_test_combined, y_train_combined, y_test_combined = train

# Initiate a Tfidf vectorizer
```

```
tfv_combined = TfidfVectorizer(ngram_range=(1, 1), stop_words='english')

X_train_combined_fit = tfv_combined.fit_transform(X_train_combined)
X_test_combined_fit = tfv_combined.transform(X_test_combined)

combined_label_model = SVC(kernel='linear', probability=True, random_state=4
combined_label_model.fit(X_train_combined_fit, y_train_combined)
```

Out[ ]:

SVC

SVC(kernel='linear', probability=True, random_state=42)

In [ ]:
```
new_comments = ['go kill yourself','Have a great day','nig','COCKSUCKER BEFC
new_comments_vect = tfv_combined.transform(new_comments)
combined_label_model.predict_proba(new_comments_vect)[:, 1]
```

Out[ ]:
```
array([0.99716297, 0.34719231, 0.95810562, 0.95546281])
```

In [ ]:
```
import joblib

joblib.dump(tfv_combined, 'tfidf_vectorizer.joblib')
joblib.dump(combined_label_model, 'toxicity_classification_model.joblib')
```

Out[ ]:
```
['toxicity_classification_model.joblib']
```