Tarea 2

NOMBRE ESTUDIANTE:

CODIGO ESTUDIANTE

Use este mismo Jupyter Notebook pero cambiele de nombre a que sea:

Tarea2_ML_Nombre_Apellido.ipynb

Bueno el proposito de esta tarea de enfrentarlos a un problema "real" en donde aplicaran varios de los modelos que discutimos en los notebooks.

lo primero que haremos es encontrar/buscar/cargar los datos

Ahora para esta tarea haremos uso de una base de datos que guarda texto. Cualquier persona que hay sido abusada o que haya sido amenazada "online" sabe perfectamente que este tipo de cosas no se va cuando tu apagas el telefono o cuando apagas la computadora. En esta tare vamos a trabajar en un modelo multi clase que detecta varios tipos de toxicidad desde toxico severo, amenazas, obsenidad, insultos, etc. Antes de comenzar, si usted es sensible a estas palabras, por favor no continue y hable conmigo, mi plan no es ofenderlo, ni que se sienta mal pero darle un problema que tiene un interes muy grande en la comunidad.

En esta tarea usaremos clasificadores supervisados y representacion de textos. Un comentario toxico puede ser considerado, toxico, severamente toxico, obceno, amenazante, insultante o que tenga identidad de odio, todo al mismo tiempo o ninguno de ellos.

Los datos los vamos a sacar de este link

https://www.kaggle.com/code/jhoward/nb-svm-strong-linear-baseline/data

Este archivo se puede bajar en formato csv. Hagalo y si esta usando google collab, entonces pasa el archivo a su google drive. Lo mas seguro es que tengas que abrir una cuenta en este sitio, lo cual no cuesta nada pero para poder bajar los datos. Por supuesto, parte de esta tarea es buscar como bajar los datos. Al final vas a poder hacerlo :-). El mas importante, que usaran para la tarea se llama train.csv.

De ahora en adelante asumo que los datos estan ya disponible para ser leido por este Jupyter notebook.

In [1]:
```
# vamos a importar todas las librerias necesarias


%matplotlib inline

# de esta libreria no hemos hablado, pero es una libreria en Python para manipular texto
# hay varios links que les puede servir para entenderlo, yo les recomiendo
#
# https://www.mygreatlearning.com/blog/regular-expression-in-python/
# https://towardsdatascience.com/regex-with-python-b4c5ca7c1eba
#
import re
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.model_selection import train_test_split

# la siguiente hace lo siguiente
# convierte documentos (o textos) a una matrix  TF-IDF
# este concepto de TF-IDF significa en ingles Term Frequenct Inverse Document Frequency
# que es una manera de medir que tan importante es una palabra en el contexto del documento
# EL TF = (Nuvemo de veces que el termino t aparece en el documento) / (Numero total de terminos en el documento)
# IDF = log10(Numero total de documentos/ Numero total de documentos con el termino t en el).
# TF-IDF = TF*IDF
#
from sklearn.feature_extraction.text import TfidfVectorizer


from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

#este es un clasificador my paredico al que hablamos en clase

from sklearn.multiclass import OneVsRestClassifier

# El proceso de convertir datos a algo que el computador pueda entender se llama preprocesamiento.
# Uno de las etapas mas importante es la de filtrar datos que no tienen ninguna utilidad
# En procesamiento de lenguaje (Natural Language Processing) estas se llaman "stop words"
# es decir eliminar del analisis palabras como "the" "a", etc.

from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
import seaborn as sns
```

In [2]:
```
#Ok ahora ya tienes todos los paquetes y ademas tienes el archivo de datos en disco
# Importe el archivo train.csv, cuando lo lean y para evitar problemas de caracteres extraños
# use encoding = "ISO-8859-1"

# ......

# Lo que deberia de darle es:
```

In [3]:
```
df = pd.read_csv("train.csv", encoding = "ISO-8859-1")
df.head()
```

Out[3]:

	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

In [4]:
```
# cuente el numero de comentarios en cada categoria
# cada categoria esta guardada en cada columna, exceptiando el "id" y el "comment text"

#......


#Esto deberia de darles:
```

In [5]:
```
df_toxic = df.drop(['id', 'comment_text'], axis=1)
counts = []
categories = list(df_toxic.columns.values)
for i in categories:
    counts.append((i, df_toxic[i].sum()))
df_stats = pd.DataFrame(counts, columns=['category', 'number_of_comments'])
df_stats
```

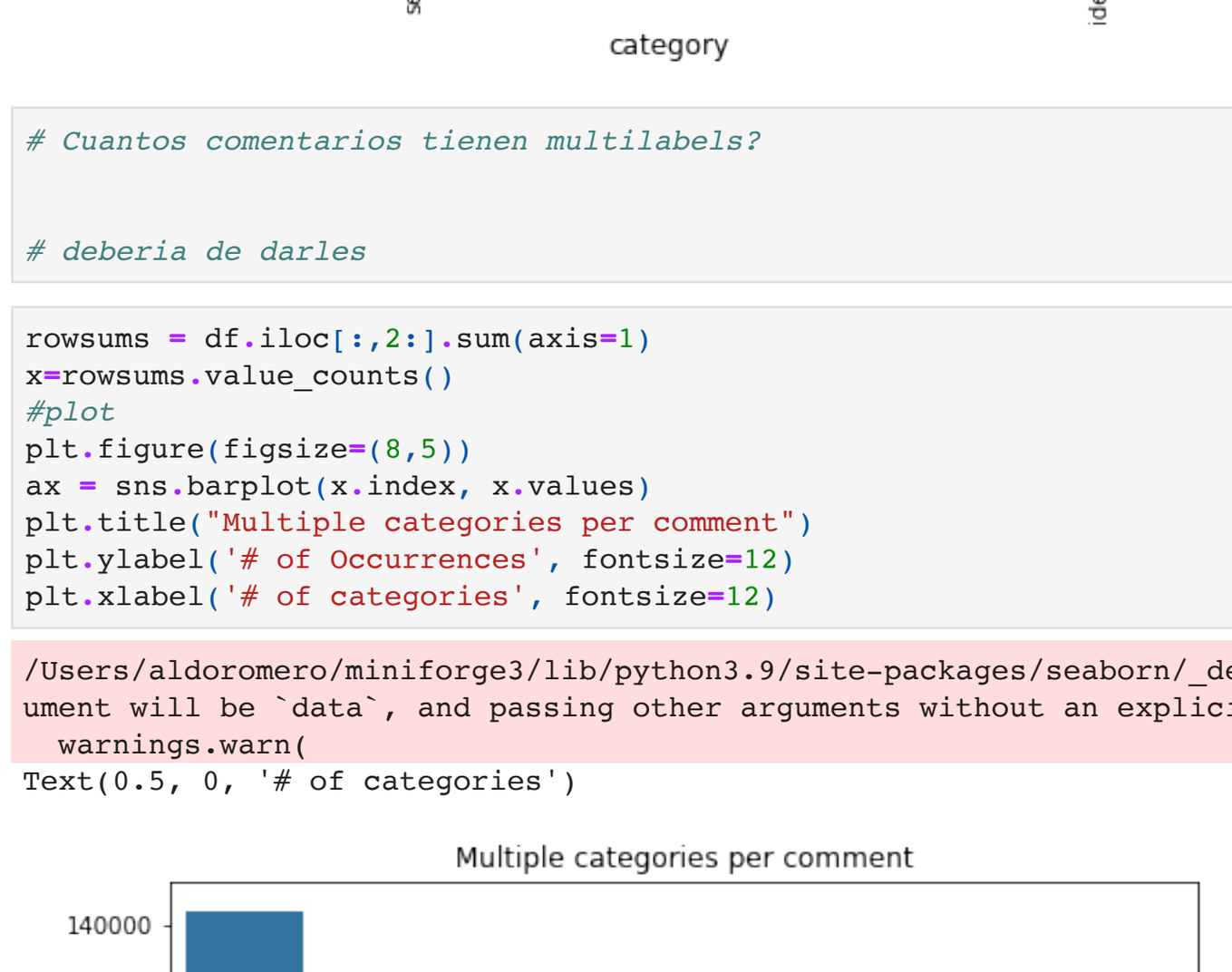
Out[5]:

	category	number_of_comments
0	toxic	15294
1	severe_toxic	1595
2	obscene	8449
3	threat	478
4	insult	7877
5	identity_hate	1405

In [6]:
```
# Ahora vamos a graficar estos datos

#.....

# lo que deberia de darles:
```

In [7]:
```
df_stats.plot(x='category', y='number_of_comments', kind='bar', legend=False, grid=True, figsize=(8, 5))
plt.title("Number of comments per category")
plt.ylabel('# of Occurrences', fontsize=12)
plt.xlabel('category', fontsize=12)
```

Out[7]:
```
Text(0.5, 0, 'category')
```

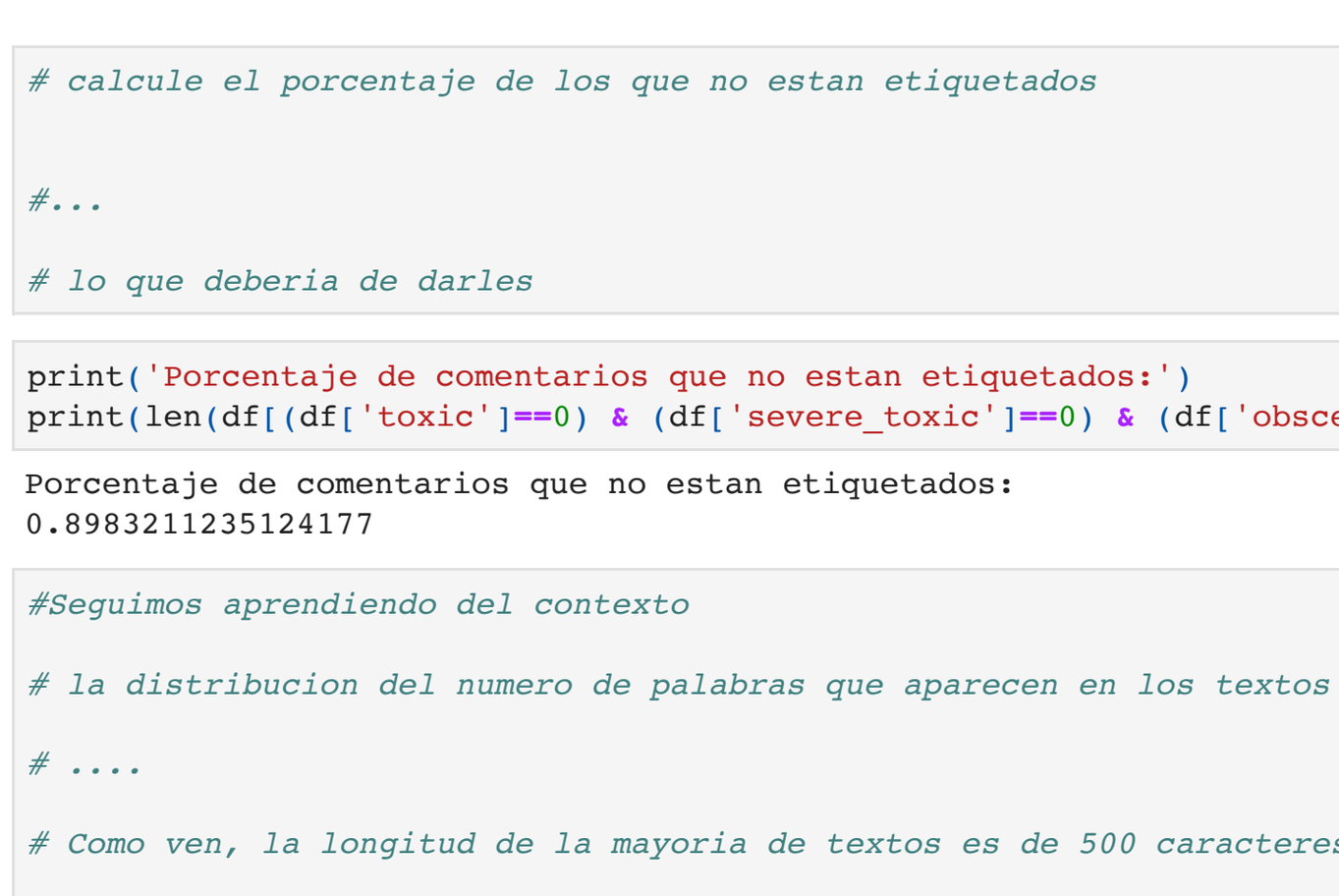


In [8]:
```
# Cuantos comentarios tienen multilabels?


# deberia de darles
```

In [9]:
```
rowsums = df.iloc[:,2:].sum(axis=1)
x=rowsums.value_counts()
#plot
plt.figure(figsize=(8,5))
ax = sns.barplot(x.index, x.values)
plt.title("Multiple categories per comment")
plt.ylabel('# of Occurrences', fontsize=12)
plt.xlabel('# of categories', fontsize=12)
```

```
/Users/aldoromero/miniforge3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional arg
ument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[9]:
```
Text(0.5, 0, '# of categories')
```



De la grafica anterior se puede ver que una gran cantidad de comentarios no tienen ninguna etiqueta.

In [10]:
```
# calcule el porcentaje de los que no estan etiquetados


#...

# lo que deberia de darles
```

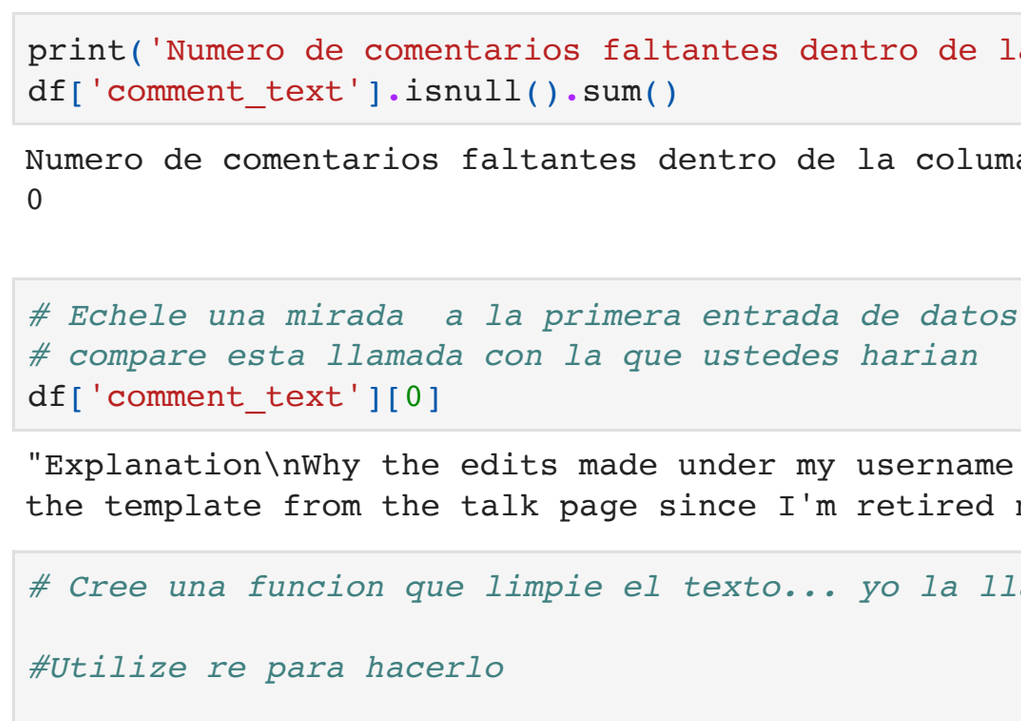
In [11]:
```
print('Porcentaje de comentarios que no estan etiquetados:')
print(len(df[(df['toxic']==0) & (df['severe_toxic']==0) & (df['obscene']==0) & (df['threat']== 0) & (df['insult']==0) & (df['identity_hate']==0)]) / len(df))
```

```
Porcentaje de comentarios que no estan etiquetados:
0.8983211235124177
```

In [14]:
```
#Seguimos aprendiendo del contexto

# la distribucion del numero de palabras que aparecen en los textos de los comentarios

# ....

# Como ven, la longitud de la mayoria de textos es de 500 caracteres, exceptuando unos que son hasta de 5000

# y luce como:
```

In [13]:
```
lens = df.comment_text.str.len()
lens.hist(bins = np.arange(0,5000,50))
```

Out[13]:
```
<AxesSubplot:>
```



In [ ]:
```
# Miremos si hay comentarios faltantes, mirando dentro de la columna "comment text"

# ...

# Lo que deberia de darles
```

In [15]:
```
print('Numero de comentarios faltantes dentro de la columan "comment text"')
df['comment_text'].isnull().sum()
```

```
Numero de comentarios faltantes dentro de la columan "comment text"
```

Out[15]:
```
0
```

In [16]:
```
# Echele una mirada  a la primera entrada de datos
# compare esta llamada con la que ustedes harian
df['comment_text'][0]
```

Out[16]:
```
"Explanation\nWhy the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure on some GAs after I voted at New York Dolls FAC. And please don't remove
the template from the talk page since I'm retired now.89.205.38.27"
```

In [17]:
```
# Cree una funcion que limpie el texto... yo la llame limpie_mi_texto

#Utilize re para hacerlo

#La funcion recibe un texto y hace los siguiente
# Lo convierte a minusculas  (ayuda:  text.lower())
# Reemplazar cosas raras como \'s  \W \s+ por " "
# cambie ciertas cadenas por otras como
#  "\'ve"   por " have "
# "can't" por "can not "
# "n't" por " not "
# "i'm" por " i am "
# "\'re" por " are "
# "\'d" por " would "
# "\'ll" por " will "
# "\'scuse" por " excuse "
# al final use strip() para borrar de la cadena de caracteres los posible caracteres vacios al principio y al final

# la funcion debe regresa el nuevo texto o cadena de caracteres
```

In [18]:
```
def limpie_mi_texto(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", "can not ", 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
```

In [19]:
```
# Por ejempo, si hago uso de esta les deberia de dar:
df['comment_text'] = df['comment_text'].map(lambda myt : limpie_mi_texto(myt))
df['comment_text'][0]

# compare el output que sale aqui con el que tenias antes
```

Out[19]:
```
'explanation why the edits made under my username hardcore metallica fan were reverted they were not vandalisms just closure on some gas after i voted at new york dolls fac and please do not remove t
he template from the talk page since i am retired now 89 205 38 27'
```

In [20]:
```
# Divida los datos entre entrenamiento y prueba con un 67% para test y 33% para prueba
# cuando use la funcion de scikit, echele una mirada al parametro shuffle=True de esa funcion
# este mezcla los datos de manera aleatoria antes de dividir los datos

# ....

# voy a imprimir el "shape" del X_train y del X_test
```

In [21]:
```
categories = ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']
train, test = train_test_split(df, random_state=42, test_size=0.33, shuffle=True)
X_train = train.comment_text
X_test = test.comment_text
print(X_train.shape)
print(X_test.shape)
```

```
(106912,)
(52659,)
```

In [23]:
```
# Ahora comenzamos el entrenamiento de varios clasificadores

# primero defina un pipeline (recuerde que esto es para hacer varias cosas en scikit)
# primero haga uso del TfidfVectorizer(stop_words=stop_words)
# Luego defina el clasificador, comenzemoz por uno de los OneVsRestClassifier en particular MultinomialNB
# parte de la linea seria:  OneVsRestClassifier(MultinomialNB(fit_prior=True, class_prior=None)

# luego que define el pipeline, vamos por las diferentes categorias y entremos el algoritmo y precedimos
# aqui imprimo lo que da para categoria (sus numeros puede variar un poco)

# serian algo asi como
# NB_Pipeline = Pipeline([....])
```

In [24]:
```
# Define a pipeline combining a text feature extractor with multi lable classifier
NB_pipeline = Pipeline([
                ('tfidf', TfidfVectorizer(stop_words=stop_words)),
                ('clf', OneVsRestClassifier(MultinomialNB(
                    fit_prior=True, class_prior=None))),
            ])
for category in categories:
    print('... Processing {}'.format(category))
    # train the model using X_dtm & y
    NB_pipeline.fit(X_train, train[category])
    # compute the testing accuracy
    prediction = NB_pipeline.predict(X_test)
    print('Test accuracy is {}'.format(accuracy_score(test[category], prediction)))
```

```
... Processing toxic
Test accuracy is 0.9192350785240889
... Processing severe_toxic
Test accuracy is 0.9900112041626312
... Processing obscene
Test accuracy is 0.9515752293055318
... Processing threat
Test accuracy is 0.9971135038644866
... Processing insult
Test accuracy is 0.9517271501547694
... Processing identity_hate
Test accuracy is 0.9910556600011394
```

SVC Lineal

In [25]:
```
# Haga los mismo pero ahora con el SVC lineal
```

In [26]:
```
SVC_pipeline = Pipeline([
                ('tfidf', TfidfVectorizer(stop_words=stop_words)),
                ('clf', OneVsRestClassifier(LinearSVC(), n_jobs=1)),
            ])
for category in categories:
    print('... Processing {}'.format(category))
    # train the model using X_dtm & y
    SVC_pipeline.fit(X_train, train[category])
    # compute the testing accuracy
    prediction = SVC_pipeline.predict(X_test)
    print('Test accuracy is {}'.format(accuracy_score(test[category], prediction)))
```

```
... Processing toxic
Test accuracy is 0.9600068364382157
... Processing severe_toxic
Test accuracy is 0.9906948479842003
... Processing obscene
Test accuracy is 0.9788830019559809
... Processing threat
Test accuracy is 0.9974363356691164
... Processing insult
Test accuracy is 0.971135038644866
... Processing identity_hate
Test accuracy is 0.9919861752027194
```

Regresion Logistica

In [27]:
```
LogReg_pipeline = Pipeline([
                ('tfidf', TfidfVectorizer(stop_words=stop_words)),
                ('clf', OneVsRestClassifier(LogisticRegression(solver='sag'), n_jobs=1)),
            ])
for category in categories:
    print('... Processing {}'.format(category))
    # train the model using X_dtm & y
    LogReg_pipeline.fit(X_train, train[category])
    # compute the testing accuracy
    prediction = LogReg_pipeline.predict(X_test)
    print('Test accuracy is {}'.format(accuracy_score(test[category], prediction)))
```

```
... Processing toxic
Test accuracy is 0.9548415275641391
... Processing severe_toxic
Test accuracy is 0.9910746501072941
... Processing obscene
Test accuracy is 0.9760724662450863
... Processing threat
Test accuracy is 0.9973603752444976
... Processing insult
Test accuracy is 0.9687422852693747
... Processing identity_hate
Test accuracy is 0.991758293928863
```

In [ ]: