ML_TeamTwo_script

December 13, 2022

[1]:
```
# from google.colab import drive
# drive.mount('/content/drive')
```

[2]:
```
! pip install contractions
! pip install wordcloud
! pip install scikit-learn-intelex
! pip install symspellpy
! pip install lazypredict

from sklearnex import patch_sklearn
patch_sklearn()

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import pandas as pd
import random
import re
import contractions
import time
from collections import defaultdict
import seaborn as sns

import nltk
from nltk import pos_tag
from nltk.corpus import stopwords, wordnet
from nltk.stem import WordNetLemmatizer
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, BernoulliNB
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.multioutput import ClassifierChain
from sklearn.calibration import CalibratedClassifierCV
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score
from wordcloud import WordCloud,STOPWORDS
```

```python
from symspellpy import SymSpell, Verbosity
from lazypredict.Supervised import LazyClassifier

import pickle

nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')

start_time = time.time()


train_df = pd.read_csv('/kaggle/input/whysoharsh/train.csv')
test_df = pd.read_csv('/kaggle/input/whysoharsh/test.csv')
sample_df = pd.read_csv('/kaggle/input/whysoharsh/sample.csv')
```

```
Collecting contractions
  Downloading contractions-0.1.73-py2.py3-none-any.whl (8.7 kB)
Collecting textsearch>=0.0.21
  Downloading textsearch-0.0.24-py2.py3-none-any.whl (7.6 kB)
Collecting anyascii
  Downloading anyascii-0.3.1-py3-none-any.whl (287 kB)
                           287.5/287.5

kB 2.5 MB/s eta 0:00:0000:0100:01
Collecting pyahocorasick
  Downloading
pyahocorasick-1.4.4-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(106 kB)
                           106.5/106.5

kB 7.6 MB/s eta 0:00:00
Installing collected packages: pyahocorasick, anyascii, textsearch,
contractions
Successfully installed anyascii-0.3.1 contractions-0.1.73 pyahocorasick-1.4.4
textsearch-0.0.24
WARNING: Running pip as the 'root' user can result in broken permissions

and conflicting behaviour with the system package manager. It is recommended to

use a virtual environment instead: https://pip.pypa.io/warnings/venv

Requirement already satisfied: wordcloud in /opt/conda/lib/python3.7/site-
packages (1.8.2.2)
Requirement already satisfied: numpy>=1.6.1 in /opt/conda/lib/python3.7/site-
packages (from wordcloud) (1.21.6)
Requirement already satisfied: pillow in /opt/conda/lib/python3.7/site-packages
(from wordcloud) (9.1.1)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.7/site-
packages (from wordcloud) (3.5.3)
```

```
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.7/site-packages (from matplotlib->wordcloud) (1.4.3)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.7/site-
packages (from matplotlib->wordcloud) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in
/opt/conda/lib/python3.7/site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.7/site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.7/site-packages (from matplotlib->wordcloud) (4.33.3)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-
packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: typing-extensions in
/opt/conda/lib/python3.7/site-packages (from
kiwisolver>=1.0.1->matplotlib->wordcloud) (4.4.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-
packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.15.0)
WARNING: Running pip as the 'root' user can result in broken permissions

and conflicting behaviour with the system package manager. It is recommended to

use a virtual environment instead: https://pip.pypa.io/warnings/venv

Requirement already satisfied: scikit-learn-intelex in
/opt/conda/lib/python3.7/site-packages (2021.6.3)
Requirement already satisfied: daal4py==2021.6.3 in
/opt/conda/lib/python3.7/site-packages (from scikit-learn-intelex) (2021.6.3)
Requirement already satisfied: scikit-learn>=0.22 in
/opt/conda/lib/python3.7/site-packages (from scikit-learn-intelex) (1.0.2)
Requirement already satisfied: numpy>=1.15 in /opt/conda/lib/python3.7/site-
packages (from daal4py==2021.6.3->scikit-learn-intelex) (1.21.6)
Requirement already satisfied: daal==2021.6.0 in /opt/conda/lib/python3.7/site-
packages (from daal4py==2021.6.3->scikit-learn-intelex) (2021.6.0)
Requirement already satisfied: tbb==2021.* in /opt/conda/lib/python3.7/site-
packages (from daal==2021.6.0->daal4py==2021.6.3->scikit-learn-intelex)
(2021.7.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.7/site-packages (from scikit-learn>=0.22->scikit-learn-
intelex) (3.1.0)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.7/site-
packages (from scikit-learn>=0.22->scikit-learn-intelex) (1.0.1)
Requirement already satisfied: scipy>=1.1.0 in /opt/conda/lib/python3.7/site-
packages (from scikit-learn>=0.22->scikit-learn-intelex) (1.7.3)
WARNING: Running pip as the 'root' user can result in broken permissions

and conflicting behaviour with the system package manager. It is recommended to

use a virtual environment instead: https://pip.pypa.io/warnings/venv

Collecting symspellpy
```

```
  Downloading symspellpy-6.7.7-py3-none-any.whl (2.6 MB)
                           2.6/2.6 MB
15.2 MB/s eta 0:00:0000:0100:01
Collecting editdistpy>=0.1.3
  Downloading editdistpy-0.1.3-cp37-cp37m-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (125 kB)
                           125.5/125.5

kB 3.7 MB/s eta 0:00:00
Installing collected packages: editdistpy, symspellpy
Successfully installed editdistpy-0.1.3 symspellpy-6.7.7
WARNING: Running pip as the 'root' user can result in broken permissions

and conflicting behaviour with the system package manager. It is recommended to

use a virtual environment instead: https://pip.pypa.io/warnings/venv

Collecting lazypredict
  Downloading lazypredict-0.2.12-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.7/site-
packages (from lazypredict) (1.0.2)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-packages
(from lazypredict) (4.64.0)
Requirement already satisfied: joblib in /opt/conda/lib/python3.7/site-packages
(from lazypredict) (1.0.1)
Requirement already satisfied: xgboost in /opt/conda/lib/python3.7/site-packages
(from lazypredict) (1.6.2)
Requirement already satisfied: lightgbm in /opt/conda/lib/python3.7/site-
packages (from lazypredict) (3.3.2)
Requirement already satisfied: click in /opt/conda/lib/python3.7/site-packages
(from lazypredict) (8.0.4)
Requirement already satisfied: pandas in /opt/conda/lib/python3.7/site-packages
(from lazypredict) (1.3.5)
Requirement already satisfied: importlib-metadata in
/opt/conda/lib/python3.7/site-packages (from click->lazypredict) (4.13.0)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages
(from lightgbm->lazypredict) (1.21.6)
Requirement already satisfied: wheel in /opt/conda/lib/python3.7/site-packages
(from lightgbm->lazypredict) (0.37.1)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages
(from lightgbm->lazypredict) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.7/site-packages (from scikit-learn->lazypredict) (3.1.0)
Requirement already satisfied: python-dateutil>=2.7.3 in
/opt/conda/lib/python3.7/site-packages (from pandas->lazypredict) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-
packages (from pandas->lazypredict) (2022.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-
```

```
packages (from python-dateutil>=2.7.3->pandas->lazypredict) (1.15.0)
Requirement already satisfied: zipp>=0.5 in /opt/conda/lib/python3.7/site-
packages (from importlib-metadata->click->lazypredict) (3.8.0)
Requirement already satisfied: typing-extensions>=3.6.4 in
/opt/conda/lib/python3.7/site-packages (from importlib-
metadata->click->lazypredict) (4.4.0)
Installing collected packages: lazypredict
Successfully installed lazypredict-0.2.12
WARNING: Running pip as the 'root' user can result in broken permissions
and conflicting behaviour with the system package manager. It is recommended to
use a virtual environment instead: https://pip.pypa.io/warnings/venv


Intel(R) Extension for Scikit-learn* enabled (https://github.com/intel/scikit-
learn-intelex)

<IPython.core.display.HTML object>

[nltk_data] Downloading package stopwords to /usr/share/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /usr/share/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /usr/share/nltk_data…
```

[3]:
```
train_df.head()
```

[3]:
```
                      id                                               text  \
0  a8be7c5d4527adbbf15f  ", 6 December 2007 (UTC)\nI am interested, not…
1  0b7ca73f388222aad64d  I added about three missing parameters to temp…
2  db934381501872ba6f38  SANDBOX?? \n\nI DID YOUR MADRE DID IN THE SANDBOX
3  228015c4a87c4b1f09a7  why good sir? Why? \n\nYou, sir, obviously do …
4  b18f26cfa1408b52e949  "\n\n Source \n\nIncase I forget, or someone e…

   harsh  extremely_harsh  vulgar  threatening  disrespect  targeted_hate
0      0                0       0            0           0              0
1      0                0       0            0           0              0
2      1                0       0            0           0              0
3      1                0       1            1           1              0
4      0                0       0            0           0              0
```

[4]:
```
train_df.isna().sum()
```

[4]:
```
id                 0
text               0
harsh              0
extremely_harsh    0
vulgar             0
threatening        0
```

```
disrespect         0
targeted_hate      0
dtype: int64
```

[5]:
```
train_df.nunique()
```

[5]:
```
id                 89359
text               89359
harsh                  2
extremely_harsh        2
vulgar                 2
threatening            2
disrespect             2
targeted_hate          2
dtype: int64
```

[6]:
```
len(train_df)
```

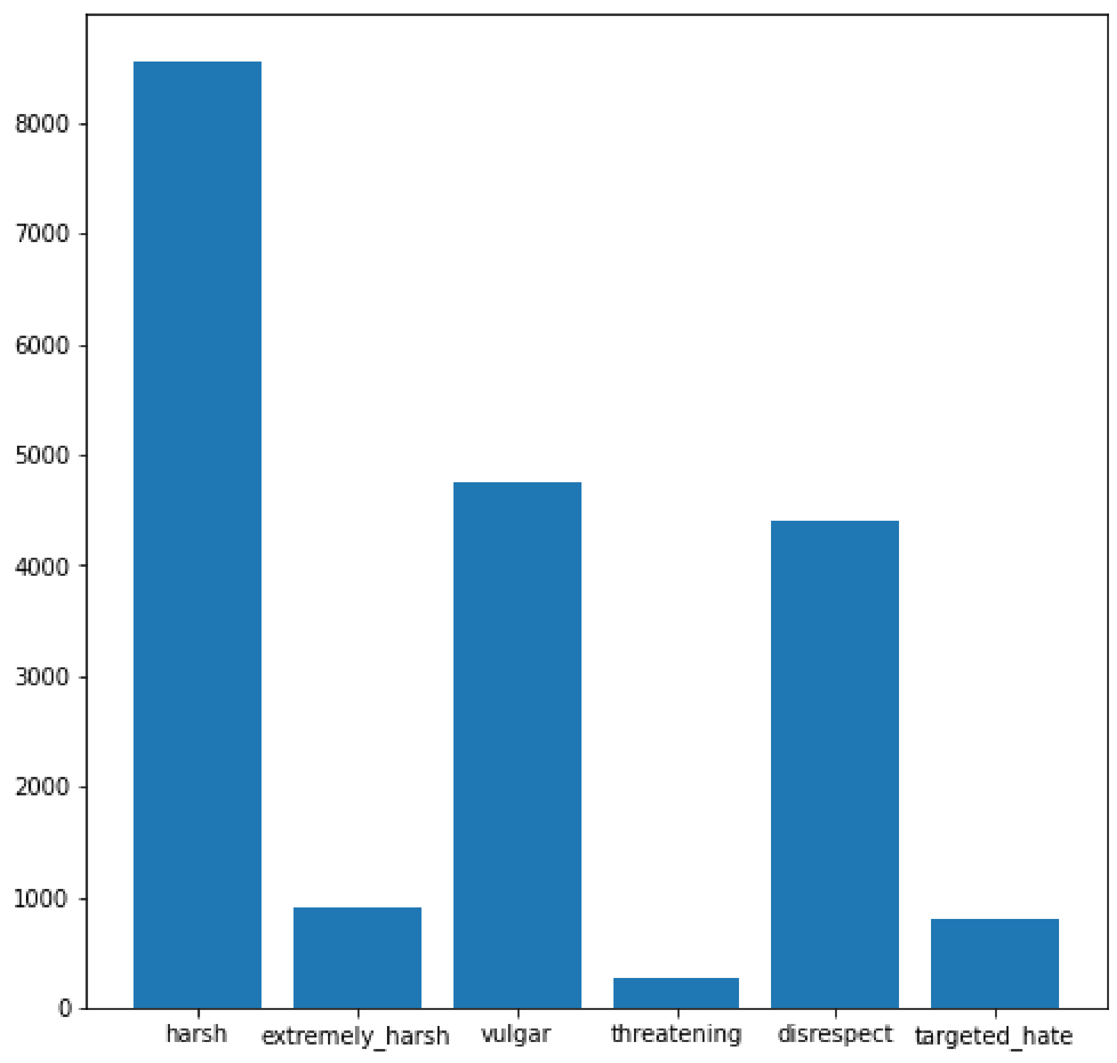
[6]:
```
89359
```

[7]:
```
print(train_df['text'][62088])
```

```
"Hello, and welcome to Wikipedia.

Helpful links  Editing || Writing a great article || Naming and Merging || Style
Manual || Policies  Reassigning old edits || What Wikipedia is not

Maintenance  Deleting articles ||  all maintenance tasks (see also open tasks,
below)

Uploading images: please note the origins and copyright status of every image
you upload.

 To sign your comments, type four tildes like this:  ~~~~.
 This automatically adds your name and the current time.

I hope you enjoy being a Wikipedian on en:.  Drop us a note at Wikipedia:New
user log so we can meet you and help you get started.
You can also leave questions on my talk page. :)

Regards, +

"
```

0.1 EDA

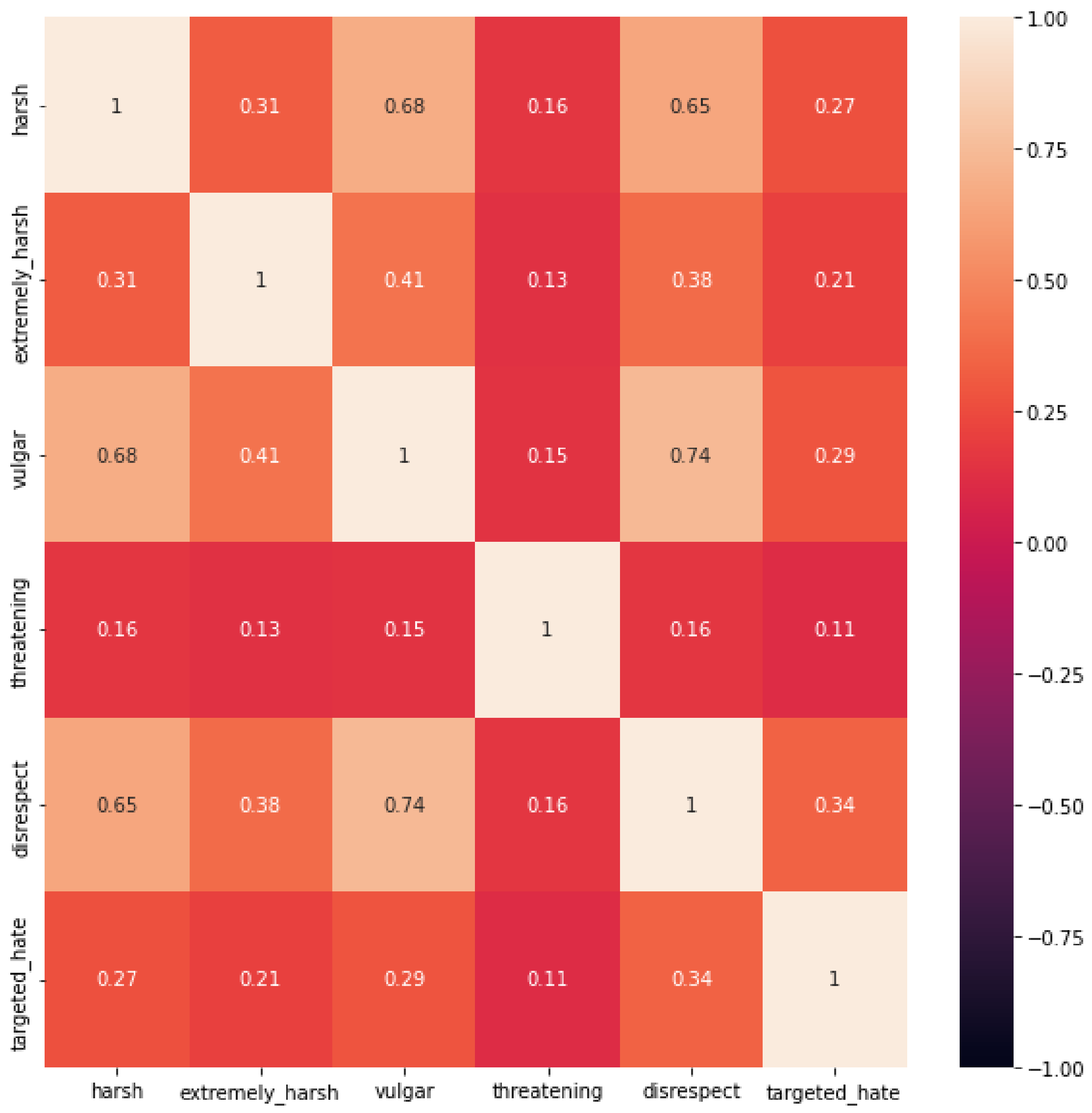
[8]:
```
map = {
    'harsh' : train_df['harsh'].sum(),
    'extremely_harsh': train_df['extremely_harsh'].sum(),
    'vulgar': train_df['vulgar'].sum(),
    'threatening': train_df['threatening'].sum(),
    'disrespect': train_df['disrespect'].sum(),
    'targeted_hate': train_df['targeted_hate'].sum()
}

names = list(map.keys())
values = list(map.values())

plt.rcParams["figure.figsize"] = (8,8)

plt.bar(range(len(map)),values, tick_label=names)
plt.show()
print(map)
```



```
{'harsh': 8559, 'extremely_harsh': 917, 'vulgar': 4742, 'threatening': 268,
'disrespect': 4392, 'targeted_hate': 802}
```

[9]:
```
train_df.corr()
plt.figure(figsize=(10, 10))
sns.heatmap(train_df.corr(), vmin=-1,annot=True)
```

[9]:
```
<AxesSubplot:>
```



	harsh	extremely_harsh	vulgar	threatening	disrespect	targeted_hate
harsh	1	0.31	0.68	0.16	0.65	0.27
extremely_harsh	0.31	1	0.41	0.13	0.38	0.21
vulgar	0.68	0.41	1	0.15	0.74	0.29
threatening	0.16	0.13	0.15	1	0.16	0.11
disrespect	0.65	0.38	0.74	0.16	1	0.34
targeted_hate	0.27	0.21	0.29	0.11	0.34	1

0.2 Fixing Contractions

[10]:
```
train_df['text'] = train_df['text'].apply(lambda s: contractions.fix(s))
test_df['text'] = test_df['text'].apply(lambda s: contractions.fix(s))
```

0.3 Converting to lowercase letters

[11]:
```
train_df['text'] = train_df['text'].apply(lambda s: s.lower())
test_df['text'] = test_df['text'].apply(lambda s: s.lower())
```

0.4 Replacing emoticons

[12]:
```python
emoticons = {
    ':-)': 'happy',
    ':)': 'happy',
    ':-(': 'frown',
    ':(': 'frown',
    'xD': 'laugh',
    ':/': 'sad',
    ':|': 'indecision',
    ':o': 'surprise',
    '<3': 'heart'
}
```

[13]:
```python
emojis = list(emoticons.keys())
meanings = list(emoticons.values())

def replace_emoji(text):
  sentence = ''
  for word in str(text).split():
    if word in emoticons:
      word = emoticons[word]
    sentence += word + ' '
  return sentence

train_df['text'] = train_df['text'].apply(lambda s: replace_emoji(s))
test_df['text'] = test_df['text'].apply(lambda s: replace_emoji(s))
```

[14]:
```python
print(train_df['text'][62088])
```

```
"hello, and welcome to wikipedia. helpful links editing || writing a great
article || naming and merging || style manual || policies reassigning old edits
|| what wikipedia is not maintenance deleting articles || all maintenance tasks
(see also open tasks, below) uploading images: please note the origins and
copyright status of every image you upload. to sign your comments, type four
tildes like this: ~~~~. this automatically adds your name and the current time.
i hope you enjoy being a wikipedian on en:. drop us a note at wikipedia:new user
log so we can meet you and help you get started. you can also leave questions on
my talk page. happy regards, + "
```

0.5 Removing punctuation marks

[15]:
```python
train_df['text'] = train_df['text'].apply(lambda s: re.sub(r'[^\w\s]', '', s))
test_df['text'] = test_df['text'].apply(lambda s: re.sub(r'[^\w\s]', '', s))
```

0.6 Removing all numbers

[16]:
```
train_df['text'] = train_df['text'].apply(lambda s: re.sub(r'\d+', '', s))
test_df['text'] = test_df['text'].apply(lambda s: re.sub(r'\d+', '', s))
```

0.7 Correcting spellings of words

[17]:
```
sym_spell = SymSpell()
corpus_path = '/kaggle/input/whysoharsh/frequency_dictionary_en_82_765.txt'
sym_spell.create_dictionary(corpus_path)
```

[17]:
```
True
```

[18]:
```
# def correct_spelling(text):
#     s = ''
#     for w in str(text).split():
#         suggestions = sym_spell.lookup(w, Verbosity.CLOSEST,
#                                        max_edit_distance=2, include_unknown=True)
#         s += suggestions[0].term
#     return s

def correct_spelling(text):
    return ' '.join([sym_spell.lookup(w, Verbosity.CLOSEST,␣
 ↪max_edit_distance=2, include_unknown=True)[0].term for w in str(text).
 ↪split()])

train_df['text'] = train_df['text'].apply(lambda s: correct_spelling(s))
test_df['text'] = test_df['text'].apply(lambda s: correct_spelling(s))
```

0.8 Removing words containing digits (Eg: hello123)

[19]:
```
train_df['text'] = train_df['text'].apply(lambda s: re.sub(r'\w*\d\w*', '', s))
test_df['text'] = test_df['text'].apply(lambda s: re.sub(r'\w*\d\w*', '', s))
```

0.9 Removing stop words

[20]:
```
# stop_words = set(stopwords.words('english'))
# stop_words.update(['http', 'https', 'www', 'html', 'jpg', 'htm']) # Add some␣
 ↪more words

# def stop(text):
#     return ' '.join([w for w in str(text).split() if w not in stop_words])

# train_df['text'] = train_df['text'].apply(lambda s: stop(s))
# test_df['text'] = test_df['text'].apply(lambda s: stop(s))
```

0.10 Lemmatization

[21]:
```
# lemmatizer = WordNetLemmatizer()

# def lemma(text):
#      return ' '.join([lemmatizer.lemmatize(w) for w in str(text).split()])

# train_df['text'] = train_df['text'].apply(lambda s: lemma(s))
# test_df['text'] = test_df['text'].apply(lambda s: lemma(s))
```

0.11 Stemming

[22]:
```
stemmer = SnowballStemmer('english')

def stemm(text):
     return ' '.join([stemmer.stem(w) for w in str(text).split()])

train_df['text'] = train_df['text'].apply(lambda s: stemm(s))
test_df['text'] = test_df['text'].apply(lambda s: stemm(s))
```

0.12 Vectorization

[23]:
```
tfidfVectorizer = TfidfVectorizer(
    strip_accents = 'unicode', # Remove accents and perform other character␣
 ↪normalization during the preprocessing step
    analyzer = 'word',
    stop_words = 'english',
    ngram_range = (1, 2) ,
    max_df = 0.5, # ignore terms that appear in more than 50% of the documents
    min_df = 2, # ignore terms that appear in less than 2 documents
    sublinear_tf = True # Apply sublinear tf scaling, i.e. replace tf with 1 +␣
 ↪log(tf)
)

char_vectorizer = TfidfVectorizer(
    strip_accents='unicode',
    analyzer = 'char',
    min_df = 2,
    max_df = 0.5,
    ngram_range = (2, 6),
    sublinear_tf = True
)

train_text_list = train_df['text']
test_text_list = test_df['text']

whole_text = pd.concat([train_text_list, test_text_list])
```

```
tfidfVectorizer.fit(whole_text)
X_train_vectorized = tfidfVectorizer.transform(train_text_list)
X_test_vectorized = tfidfVectorizer.transform(test_text_list)

char_vectorizer.fit(whole_text)
train_char_features = char_vectorizer.transform(train_text_list)
test_char_features = char_vectorizer.transform(test_text_list)

from scipy.sparse import hstack

X_train_vectorized = hstack([X_train_vectorized, train_char_features]).tocsr()
X_test_vectorized = hstack([X_test_vectorized, test_char_features]).tocsr()

pickle.dump(X_train_vectorized, open('XTrainVecMaxFeatures.pkl', 'wb'))
pickle.dump(X_test_vectorized, open('XTestVecMaxFeatures.pkl', 'wb'))
```

0.13 Extracting vectorized data from pickle files

0.13.1 Lemmatized data

[24]:
```
# train_file_path = '/kaggle/input/whysoharsh/XTrainVecMaxFeaturesupdated.pkl'
# test_file_path = '/kaggle/input/whysoharsh/XTestVecMaxFeaturesupdated.pkl'

# with open(train_file_path , 'rb') as trainf:
#     X_train_vectorized = pickle.load(trainf)

# with open(test_file_path , 'rb') as testf:
#     X_test_vectorized = pickle.load(testf)
```

0.13.2 Stemmed data

[25]:
```
train_file_path = '/kaggle/input/whysoharsh/XTrainVecMaxFeatures_Stemmer.pkl'
test_file_path = '/kaggle/input/whysoharsh/XTestVecMaxFeatures_Stemmer.pkl'

with open(train_file_path , 'rb') as trainf:
    X_train_vectorized = pickle.load(trainf)

with open(test_file_path , 'rb') as testf:
    X_test_vectorized = pickle.load(testf)
```

0.14 Logistic Regression

[26]:
```
categories = ['harsh', 'extremely_harsh', 'vulgar', 'threatening',␣
↪'disrespect', 'targeted_hate']

predictions = {
    'id': test_df['id']
}

roc_score =[]

for c in categories:
    Y = train_df[c].to_numpy()
    x_train, x_test, y_train, y_test = train_test_split(X_train_vectorized, Y,␣
  ↪test_size=0.3, random_state=42, stratify=Y)
    lr = LogisticRegression(class_weight='balanced', C=1, solver='lbfgs')
    lr.fit(x_train, y_train)

    predictions[c] = lr.predict_proba(X_test_vectorized)[:, 1]

    roc_score.append(roc_auc_score(y_test, lr.predict_proba(x_test)[:, 1]))

df_submit = pd.DataFrame(predictions)

df_submit.to_csv('submission_lr.csv', index=None)

print(np.mean(roc_score))
pickle.dump(lr, open('model_lr.pkl', 'wb'))
```

```
0.9843876380885671
```

0.15 Classifier chain

[27]:

```
categories = ['harsh', 'extremely_harsh', 'vulgar', 'threatening',␣
  ↪'disrespect', 'targeted_hate']

predictions = {
    'id': test_df['id']
}

roc_score =[]

for c in categories:
    Y = train_df[c].to_numpy()
    x_train, x_test, y_train, y_test = train_test_split(X_train_vectorized, Y,␣
  ↪test_size=0.3, random_state=42, stratify=Y)
    base_lr = LogisticRegression(class_weight='balanced', C=1, random_state=0)
    chain = ClassifierChain(base_lr, order='random', random_state=0)
    chain.fit(x_train, y_train.reshape(-1,1))
    predictions[c] = chain.predict_proba(X_test_vectorized)[:,0]

    roc_score.append(roc_auc_score(y_test, chain.predict_proba(x_test)))

df_submit = pd.DataFrame(predictions)
```

```
df_submit.to_csv('submission_cc.csv', index=None)

print(np.mean(roc_score))
pickle.dump(chain, open('model_cc.pkl', 'wb'))
```

```
0.9843876380885671
```

0.16 SGDClassifier

[28]:
```
categories = ['harsh', 'extremely_harsh', 'vulgar', 'threatening',␣
↪'disrespect', 'targeted_hate']

predictions = {
    'id': test_df['id']
}

roc_score =[]

for c in categories:
    Y = train_df[c].to_numpy()
    x_train, x_test, y_train, y_test = train_test_split(X_train_vectorized, Y,␣
↪test_size=0.3, random_state=42, stratify=Y)
    sgdc = SGDClassifier(loss="log", penalty="l2", max_iter=100000,␣
↪class_weight='balanced')
    sgdc.fit(x_train, y_train)

    predictions[c] = sgdc.predict_proba(X_test_vectorized)[:, 1]

    roc_score.append(roc_auc_score(y_test, sgdc.predict_proba(x_test)[:, 1]))

df_submit = pd.DataFrame(predictions)

df_submit.to_csv('submission_sgdc.csv', index=None)

print(np.mean(roc_score))
pickle.dump(sgdc, open('model_sgdc.pkl', 'wb'))
```

```
0.9824327470814311
```

0.17 Ridge

[29]:
```
from sklearn.linear_model import Ridge

categories = ['harsh', 'extremely_harsh', 'vulgar', 'threatening',␣
↪'disrespect', 'targeted_hate']

predictions = {
    'id': test_df['id']
}

roc_score =[]

for c in categories:
    Y = train_df[c].to_numpy()
    x_train, x_test, y_train, y_test = train_test_split(X_train_vectorized, Y,␣
↪test_size=0.3, random_state=42, stratify=Y)
    rid = Ridge(copy_X=True, solver='sag', random_state=33, alpha=45)
    rid.fit(x_train, y_train)

    pred_y = rid.predict(x_test)
    roc_score.append(roc_auc_score(y_test, pred_y))

    rid.fit(X_train_vectorized, Y)
    predictions[c] = rid.predict(X_test_vectorized)

df_submit = pd.DataFrame(predictions)

df_submit.to_csv('submission_rid.csv', index=None)

print(roc_score)
print(np.mean(roc_score))
pickle.dump(rid, open('model_rid.pkl', 'wb'))
```

```
[0.9736407906269599, 0.9876346093886516, 0.9912239771745617, 0.9887910150404071,
0.9808938879896368, 0.9783844088234132]
0.983428114840605
```

[30]:
```
# pickled_model = pickle.load(open('model.pkl', 'rb'))
# pickled_model.predict(X_test)
```

0.18 Random Forest Classifier

[31]:
```
# from sklearn.ensemble import RandomForestClassifier
# categories = ['harsh', 'extremely_harsh', 'vulgar', 'threatening',␣
↪'disrespect', 'targeted_hate']

# predictions = {
#     'id': test_df['id']
# }

# roc_score =[]

# for c in categories:
#     Y = train_df[c].to_numpy()
#     x_train, x_test, y_train, y_test = train_test_split(X_train_vectorized,␣
↪Y, test_size=0.3, random_state=42, stratify=Y)
#     p = RandomForestClassifier(criterion='gini',
#             max_depth=100, max_features=500, max_leaf_nodes=None,␣
↪min_samples_split=10,
#             min_weight_fraction_leaf=0.0, n_estimators=120)
#     clf = p.fit(x_train, y_train)
#     calibrator = CalibratedClassifierCV(clf, cv='prefit')
#     model = calibrator.fit(x_train, y_train)
#     sgdc.fit(x_train, y_train)

#     predictions[c] = model.predict_proba(X_test_vectorized)[:, 1]

#     roc_score.append(roc_auc_score(y_test, model.predict_proba(x_test)[:, 1]))

# df_submit = pd.DataFrame(predictions)

# df_submit.to_csv('submission_rfc.csv', index=None)

# print(np.mean(roc_score))
# pickle.dump(sgdc, open('model_rfc.pkl', 'wb'))
```

0.19 Hyperparameter Tuning

[32]:
```
# param = {
#         'C' : [0.001, 0.01, 0.1, 1, 10, 100, 1000],
#         'penalty':['l2', None],
# #         'solver': ['newton-cg', 'lbfgs', 'sag', 'saga'],
#         'solver': ['lbfgs', 'sag'],
#         'class_weight': ['balanced', None]
# }

# Y = train_df['targeted_hate'].to_numpy()
# x_train, x_test, y_train, y_test = train_test_split(X_train_vectorized, Y,␣
↪test_size=0.3, random_state=42)
# lr = LogisticRegression()

# rscv = RandomizedSearchCV(lr, param, cv = 5, scoring='roc_auc')

# rscv.fit(x_train, y_train)

# print("Tuned Logistic Regression: {}".format(rscv.best_params_))
# print("Best score is {}".format(rscv.best_score_))
```

[33]:
```
print("--- %s seconds ---" % (time.time() - start_time))
```

```
--- 2723.5653455257416 seconds ---
```

0.20 LazyClassifier

[34]:
```
# from sklearn.dummy import DummyClassifier
# from sklearn.svm import LinearSVC, SVC
# from sklearn.linear_model import SGDClassifier, Perceptron,␣
↪RidgeClassifierCV, RidgeClassifier
# from sklearn.neural_network import MLPClassifier
# from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier,␣
↪RandomForestClassifier
# from sklearn.tree import DecisionTreeClassifier
# from sklearn.calibration import CalibratedClassifierCV

# lst = [
#     SGDClassifier,
#     MLPClassifier,
#     Perceptron,
#     RidgeClassifierCV,
#     RidgeClassifier,
#     AdaBoostClassifier,
#     DecisionTreeClassifier,
#     RandomForestClassifier,
#     GradientBoostingClassifier,
#     CalibratedClassifierCV
# ]



# Y = train_df['harsh'].to_numpy()
# x_train, x_test, y_train, y_test = train_test_split(
#     X_train_vectorized.toarray(), Y, test_size=0.3, random_state=42,␣
↪stratify=Y)
# clf = LazyClassifier(verbose=0,ignore_warnings=True, custom_metric=None)
# models,predictions = clf.fit(x_train, x_test, y_train, y_test)
# models
```

[35]:
```
# harsh
# Tuned Logistic Regression: {'solver': 'lbfgs', 'penalty': 'l2',␣
↪'class_weight': None, 'C': 1.0}
# Best score is 0.8822488135549081

# extremely_harsh
# Tuned Logistic Regression: {'solver': 'lbfgs', 'penalty': 'l2',␣
↪'class_weight': 'balanced', 'C': 0.1}
# Best score is 0.9518445766028873

# vulgar
# Tuned Logistic Regression: {'solver': 'sag', 'penalty': 'l2', 'class_weight':␣
↪'balanced', 'C': 0.1}
# Best score is 0.919135750971037

# threatening
# Tuned Logistic Regression: {'solver': 'sag', 'penalty': 'l2', 'class_weight':␣
↪None, 'C': 0.01}
# Best score is 0.9130528254342402

# disrespect
# Tuned Logistic Regression: {'solver': 'liblinear', 'penalty': 'l1',␣
↪'class_weight': 'balanced', 'C': 10.0}
# Best score is 0.9013409662854655

# targeted_hate
# Tuned Logistic Regression: {'solver': 'saga', 'penalty': 'l2', 'class_weight':
↪ None, 'C': 0.1}
# Best score is 0.9344772282735206
```

0.21 Some other models: Support Vector Classification

[36]:
```
# from sklearn.svm import SVC
# categories = ['harsh', 'extremely_harsh', 'vulgar', 'threatening',␣
↪'disrespect', 'targeted_hate']



# predictions = {
#     'id': test_df['id']
# }

# roc_score =[]

# for c in categories:
#     Y = train_df[c].to_numpy()
#     x_train, x_test, y_train, y_test = train_test_split(X_train_vectorized,␣
↪Y, test_size=0.3, random_state=42, stratify=Y)
#     svc = SVC()
#     svc.fit(x_train, y_train)
#     predictions[c] = svc.predict(X_test_vectorized)[0]

#     roc_score.append(roc_auc_score(y_test, lr.predict_proba(x_test)[:, 1]))

# df_submit = pd.DataFrame(predictions)

# df_submit.to_csv('submission_svm_svc.csv', index=None)
# print(np.mean(roc_score))
# pickle.dump(chain, open('model_svm_svc.pkl', 'wb'))
```

0.22 References

1. https://stackoverflow.com/a/30315056/15069364
2. https://www.geeksforgeeks.org/python-remove-punctuation-from-string/
3. https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-in-nlp/
4. https://stackoverflow.com/a/18082240/15069364
5. https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d
6. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
7. https://stackoverflow.com/a/35615151/15069364
8. https://en.wikipedia.org/wiki/List_of_emoticons
9. https://stackoverflow.com/a/1557584/15069364
10. https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/
11. https://datascience.stackexchange.com/questions/40584/meaning-of-stratify-parameter