

Detecting Cyber-bullying in text

import essential libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
↳ /usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning
import pandas.util.testing as tm
```

import dataset for cyber-bullying

```
from google.colab import files
files.upload()
```

```
↳ Choose Files cyber_cleandata (1).csv
• cyber_cleandata (1).csv(application/vnd.ms-excel) - 2977565 bytes, last modified: 9/14/2020 - 100%
done
Saving cyber_cleandata (1).csv to cyber_cleandata (1).csv
{'cyber_cleandata (1).csv': b'label,untokenized_text,preprocessed_text\r\n0,As a woma
```

```
data1=pd.read_csv('cyber_data.csv')
data2=pd.read_csv('cyber_data2.csv')
data2.head()
```

```
↳
```

	id	label	full_text
0	5.723430e+17	Offensive	So Drasko just said he was impressed the girls...
1	5.723430e+17	Offensive	So Drasko just said he was impressed the girls...
2	5.723410e+17	Offensive	Drasko they didn't cook half a bird you idiot ...
3	5.723410e+17	Offensive	Drasko they didn't cook half a bird you idiot ...
4	5.723400e+17	Offensive	Hopefully someone cooks Drasko in the next ep ...

```
# data2=data2.drop_duplicates(inplace=True)
del data2['id']
data2.head()
```

```
↳
```

	label	full_text
0	Offensive	So Drasko just said he was impressed the girls...
1	Offensive	So Drasko just said he was impressed the girls...
2	Offensive	Drasko they didn't cook half a bird you idiot ...

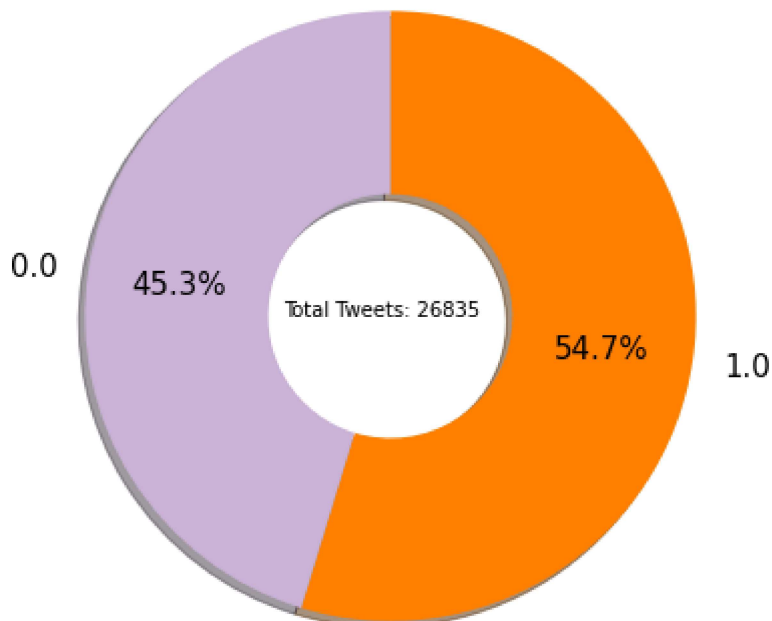
Combining two dataframes

```
data=pd.concat([data1,data2])
data.shape
```

```
↳ (35873, 2)
```

```
plt.figure(figsize = (7,7))
sorted_counts = data['label'].value_counts()
plt.pie(sorted_counts, labels = sorted_counts.index, startangle = 90, counterclock = False
        autopct='%1.1f%%', pctdistance = 0.7, textprops = {'color': 'black', 'fontsize' : 1
        colors = sns.color_palette("Paired")[7:])
plt.text(x = -0.35, y = 0, s = 'Total Tweets: {}'.format(data.shape[0]))
plt.title('Distribution of Tweets in the Dataset', fontsize = 16);
```

```
↳ Distribution of Tweets in the Dataset
```



```
data['label']=data.label.map({'Offensive':1,'Non-offensive':0}) # encoding output
data['label'].value_counts()
```

```
↳ 1    23574
   0    12299
   Name: label, dtype: int64
```

Now its time for text preprocessing. In order to deal with text we have to install NLTK library and their dependencies

```
!pip install nltk
```

```
↳ Requirement already satisfied: nltk in /usr/local/lib/python3.6/dist-packages (3.2.5)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from n
```

```
import nltk
nltk.download('stopwords')
nltk.download('words')
from nltk.corpus import stopwords
# print(stopwords.words('english'))
!pip install -q wordcloud
import wordcloud
# import nltk
from nltk.corpus import stopwords
# print(stopwords.words('english'))

nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
from bs4 import BeautifulSoup
import re
```

```
↳ [nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]   /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
```

```
def clean_sentences(df):
    reviews = []

    for sent in (df['full_text']):

        #remove html content
        review_text = BeautifulSoup(sent).get_text()

        #remove non-alphabetic characters
        review_text = re.sub("[^a-zA-Z]", " ", review_text)
```

```

# removing names in corpus
vocab = set(nltk.corpus.words.words())
vocab_text=" ".join(w for w in nltk.wordpunct_tokenize(review_text) if w.lower() i

#tokenize the sentences
# words = word_tokenize(review_text.lower())

# stops = set(stopwords.words("english"))
#
# 5. Remove stop words
# meaningful_words = [w for w in words if not w in stops]

#lemmatize each word to its lemma
# lemma_words = [lemmatizer.lemmatize(i) for i in meaningful_words]

reviews.append(vocab_text)

return(reviews)
data['untokenized_text']=clean_sentences(data)
data.head()

```

	label	full_text	untokenized_text
0	0	!!! RT @mayasolovely: As a woman you shouldn't...	As a woman you t complain about cleaning up yo...
1	1	!!!! RT @mleew17: boy dats cold...tyga dwn ba...	boy cold bad for cuffin hoe in the st place
2	1	!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...	life You ever a bitch and she start to cry You...
3	1	!!!!!!!!! RT @C G Anderson: @viva based	...

```

offensive=data[data['label']==1]
nonoffensive=data[data['label']==0]

```

```

from wordcloud import WordCloud, STOPWORDS
# import matplotlib.pyplot as plt
stopwords = set(STOPWORDS)

```

```

def show_wordcloud(data, title ):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

```

```

fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')
if title:
    fig.suptitle(title, fontsize=20)
    fig.subplots_adjust(top=2.3)

```

```
show_wordcloud(offensive['untokenized_text'],'offensive')
print('\n\n\n')
show_wordcloud(nonoffensive['untokenized_text'],'Non offensive')
```

<https://colab.research.google.com/drive/1j1vzcTNigPZlzlXDmkuXl-mnyilt2rZ-#scrollTo=hRcw7acEGNR2&printMode=true>

```

def clean_sentences(df):
    reviews = []

    for sent in (df['preprocessed_text']):

        #remove html content
        review_text = BeautifulSoup(sent).get_text()

        #remove non-alphabetic characters
        review_text = re.sub("[^a-zA-Z]", " ", review_text)
        # removing names in corpus
        # vocab = set(nltk.corpus.words.words())
        # vocab_text=" ".join(w for w in nltk.wordpunct_tokenize(review_text) if w.lower()

        #tokenize the sentences
        # words = word_tokenize(sent.lower())

        # stops = set(stopwords.words("english"))
        # stops = set(STOPWORDS)
#
        # 5. Remove stop words
        # meaningful_words = [w for w in words if not w in stops]

        #lemmatize each word to its lemma
        # lemma_words = [lemmatizer.lemmatize(i) for i in meaningful_words]

        reviews.append(review_text)

    return(reviews)
data['preprocessed_text']=clean_sentences(data)

# data=pd.read_csv('cyber_cleandata (1).csv')
# data=data.drop(columns=['Unnamed: 0', 'full_text'])
# data=data.dropna()
data.head()

```

		label	untokenized_text	preprocessed_text
	0	0.0	As a woman you t complain about cleaning up yo...	woman t complain cleaning house ...
	40	0.0	said no pussy inside my doghouse	said pussy inside doghouse
	63	0.0	Addicted t hi woof woof hot lad	addicted t hi woof woof hot ...
	66	0.0	t woof woof and hot soles	t woof woof hot sol
	67	0.0	eat a do these One	eat one

```

from sklearn.model_selection import train_test_split
# X_train,X_test,y_train,y_test=train_test_split(data['preprocessed_text'],data['label'],s
X_train.shape

```

↳ (18784,)

```
X_test.shape
```

```
(8051,)
```

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer
count_vect=CountVectorizer()
X_train_counts=count_vect.fit_transform(X_train)
# tfidf=TfidfTransformer()
# X_train_tfidf=tfidf.fit_transform(X_train_counts)
X_test_counts=count_vect.transform(X_test)

# X_test_tfidf=tfidf.transform(X_test_counts)
# test=pd.DataFrame(X_test_counts.toarray())
# train=pd.DataFrame(X_train_counts.toarray())
```

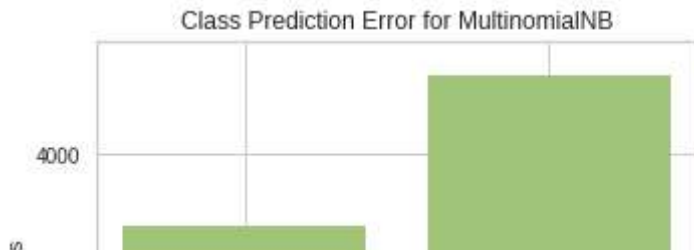
```
from sklearn.naive_bayes import MultinomialNB
naive=MultinomialNB(alpha = 0.1)
clf=naive.fit(X_train_counts,y_train)
naive_score=naive.score(X_test_counts,y_test)
```

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(bootstrap=True,max_depth=10,max_features='sqrt',min_samples_leaf
rf.fit(X_train_counts,y_train)
rf_score=rf.score(X_test_counts,y_test)
```

```
from yellowbrick.classifier import ClassPredictionError
visualizer = ClassPredictionError(
    naive, classes=['Non-offensive','offensive'])
visualizer.fit(X_train_counts,y_train)
```

```
# Evaluate the model on the test data
visualizer.score(X_test_counts,y_test)
```

```
# Draw visualization
g = visualizer.poof()
```



```
from sklearn.svm import LinearSVC
```

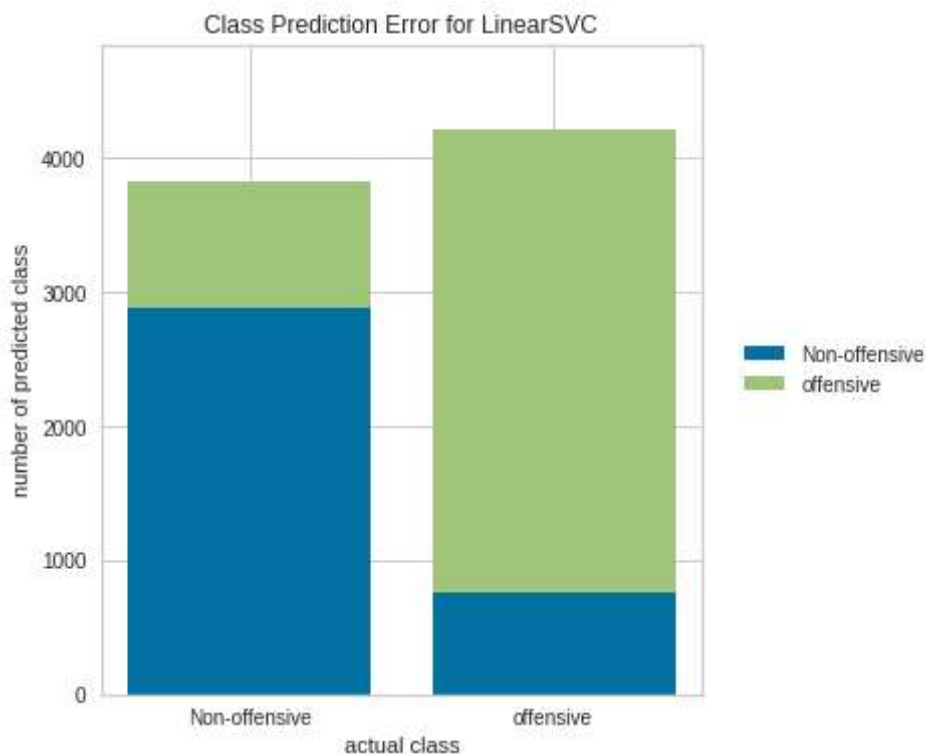
```
svm=LinearSVC()
svm.fit(X_train_counts,y_train)
svm_score=svm.score(X_test_counts,y_test)
```



```
visualizer = ClassPredictionError(
    svm, classes=['Non-offensive','offensive'])
visualizer.fit(X_train_counts,y_train)
```

```
# Evaluate the model on the test data
visualizer.score(X_test_counts,y_test)
```

```
# Draw visualization
g = visualizer.poof()
```



```
from sklearn.linear_model import LogisticRegression
```

```
lr=LogisticRegression()
lr.fit(X_train_counts,y_train)
lr_score=lr.score(X_test_counts,y_test)
```

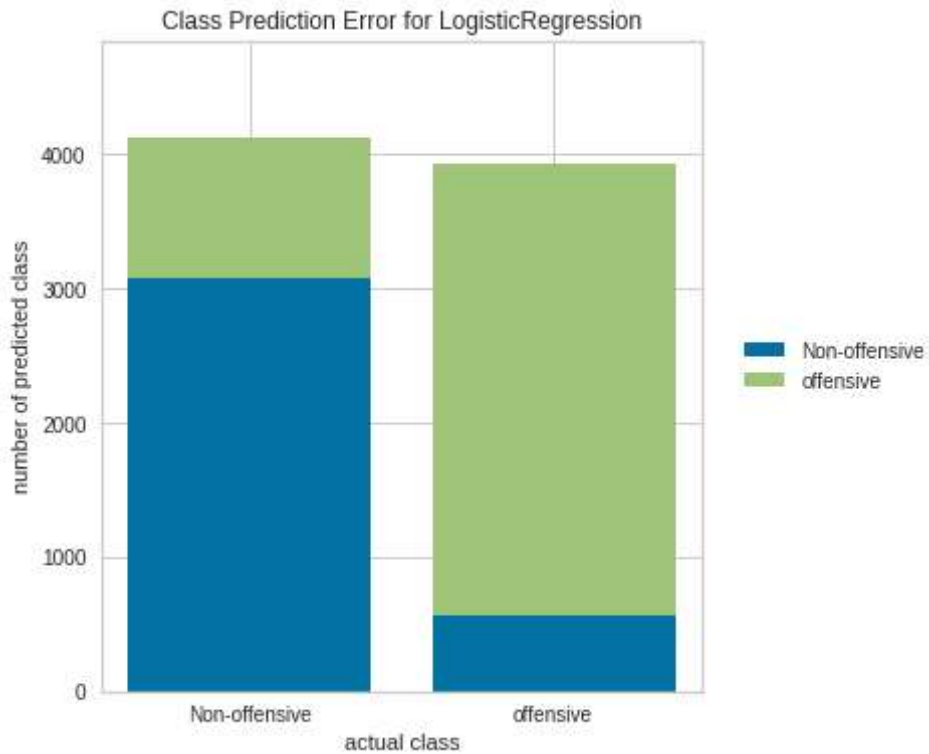
```
visualizer = ClassPredictionError(
    lr, classes=['Non-offensive','offensive'])
```



```
lr, classes=['Non-offensive','offensive'])
visualizer.fit(X_train_counts,y_train)
```

```
# Evaluate the model on the test data
visualizer.score(X_test_counts,y_test)
```

```
# Draw visualization
g = visualizer.poof()
```



```
from xgboost import XGBClassifier
```

```
xgb=XGBClassifier()
xgb.fit(X_train_counts,y_train)
xgb_score=xgb.score(X_test_counts,y_test)
```

```
DataFrame({'Algorithms':['Naive-Bayes','SVM','Logistic Regression','Random forest','Xgboost']
() )
```



	Algorithms	Accuracy
0	Naive-Bayes	0.79
1	SVM	0.79
2	Logistic Regression	0.80
3	Random forest	0.59
4	Xgboost	0.76

```
# sns.set_theme(style="whitegrid")
ax=ax = sns.barplot(x="Algorithms", y="Accuracy", data=visual_data)
plt.figure(1,figsize=(12,8))
plt.title('Accuracies of different algorithms')
plt.xlabel('Algorithms')
plt.ylabel('Accuracy')
# plt.legend()
plt.show()
```

