

```
In [1]: ▶ import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import sqlite3
%matplotlib inline
```

```
In [2]: ▶ #import previous df from sqlite
con = sqlite3.connect('twitter_hate.db')
sql = """
SELECT * FROM tweets_nlp
"""
with sqlite3.connect('twitter_hate.db') as con:
    df = pd.read_sql_query(sql, con)
```

```
In [3]: ▶ tweets = df['tweet_clean']

mentions = []
urls = []
hashtags = []
i = 0
for tweet in tweets:
    tweet = tweet.split()
    mentions.append(tweet.count('mentionhere')+tweet.count('mentionhere:')+tw
    urls.append(tweet.count('urlhere'))
    hashtags.append(tweet.count('hashtaghere'))
    tweet = [token for token in tweet if token not in [';&','']]
    tweet = [token for token in tweet if token not in ['#;mentionhere:', 'men
    tweet = " ".join(tweet)
    tweets[i] = tweet
    i += 1

df['tweet_no_others'] = tweets
df['mention_count'] = mentions
df['url_count'] = urls
df['hashtag_count'] = hashtags
```

C:\Users\seanx\anaconda3\lib\site-packages\ipykernel_launcher.py:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
from ipykernel import kernelapp as app
```

In [4]: `df.head()`

Out[4]:

	index	count	hate_speech	offensive_language	neither	class	tweet	tweet_
0	17	3	1	2	0	1	" bitch who do you love "	bitc
1	23	3	0	3	0	1	" fuck no that bitch dont even suck dick " ...	fuc don suc vide
2	38	3	0	2	1	1	" lames crying over hoes thats tears of a clown "	lames hoes tears
3	59	3	0	3	0	1	"..All I wanna do is get money and fuck model ...	all i v get r fuck b rt
4	62	3	0	3	0	1	"@ARIZZLEINDACUT: Females think dating a pussy...	fe think puss no rr

In [5]: `corpus = df['tweet_no_others']`

Bag of Words Features

In [6]: `from sklearn.feature_extraction.text import CountVectorizer`

```
cv = CountVectorizer(min_df=0., max_df=1.)
cv_matrix = cv.fit_transform(corpus)
cv_matrix = cv_matrix.toarray()
cv_matrix
```

Out[6]: `array([[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0],
...,
[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0]], dtype=int64)`

```
In [7]: # get all unique words in the corpus
vocab = cv.get_feature_names()
# show document feature vectors
df_BOW = pd.DataFrame(cv_matrix, columns=vocab)
df_BOW['class'] = df['class']
df_BOW
```

Out[7]:

	aa	aaaaaaaaand	aap	aaron	aaronmacgruder	ab	ability	abortion	about	abraham	..
0	0	0	0	0	0	0	0	0	0	0	..
1	0	0	0	0	0	0	0	0	0	0	..
2	0	0	0	0	0	0	0	0	0	0	..
3	0	0	0	0	0	0	0	0	0	0	..
4	0	0	0	0	0	0	0	0	0	0	..
...
2855	0	0	0	0	0	0	0	0	0	0	..
2856	0	0	0	0	0	0	0	0	0	0	..
2857	0	0	0	0	0	0	0	0	0	0	..
2858	0	0	0	0	0	0	0	0	0	0	..
2859	0	0	0	0	0	0	0	0	0	0	..

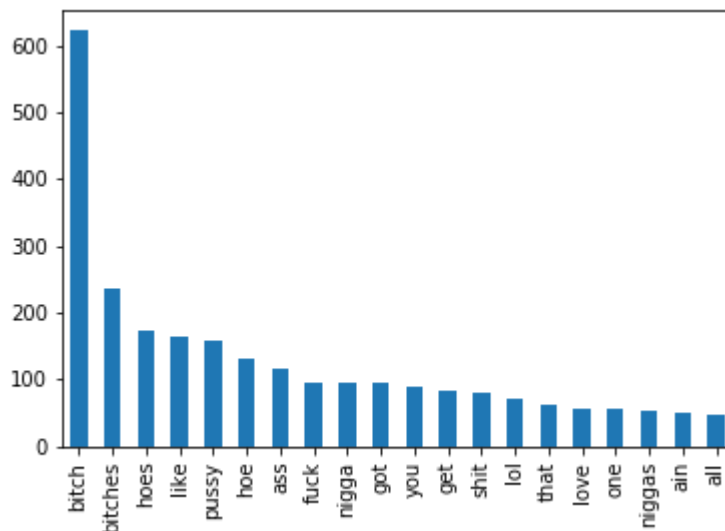
2860 rows × 5139 columns



```
In [8]: BOW_offensive = df_BOW[df_BOW['class'] == 1]
BOW_hate = df_BOW[df_BOW['class'] == 0]
BOW_offensive = BOW_offensive.drop(columns=['class'])
BOW_hate = BOW_hate.drop(columns=['class'])
```

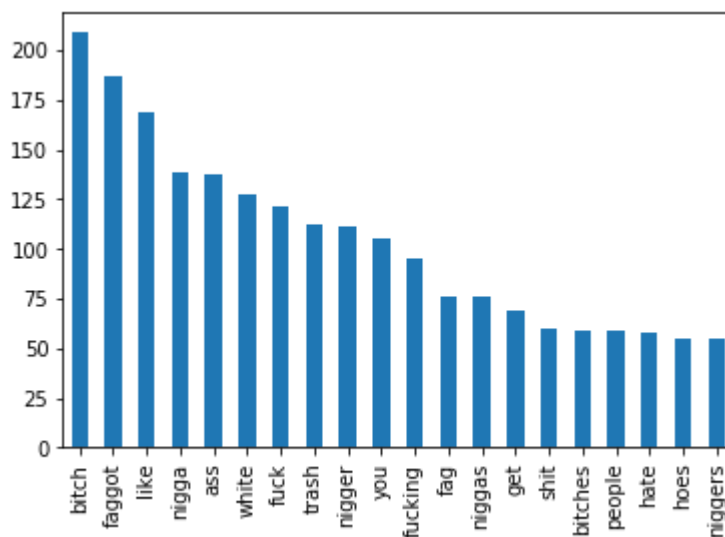
```
In [9]: ▶ BOW_count_off=BOW_offensive.sum()
BOW_off_largest = BOW_count_off.nlargest(20)
BOW_off_largest.plot(kind='bar')
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x18401166448>



```
In [10]: ▶ BOW_count_hate=BOW_hate.sum()
BOW_hate_largest = BOW_count_hate.nlargest(20)
BOW_hate_largest.plot(kind='bar')
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x18403cd4388>



Training BOW with Logistic Regression and Decision Tree

```
In [11]: ▶ X = pd.concat([df_BOW.drop(columns = ['class']), df[['mention_count', 'url_co
y = df['class'].astype(int)
```

```
In [12]: ▶ from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

```
In [13]: from sklearn.pipeline import Pipeline
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import classification_report
```

```
In [14]: param_grid = [{}]
lg = GridSearchCV(LogisticRegression(),
                  param_grid,
                  cv=KFold(n_splits=5,
                           random_state=42).split(X_train,
                                                    y_train),
                  verbose=2)
y_preds_lg = lg.fit(X_train, y_train).predict(X_test)
```

C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection_split.py:296: FutureWarning: Setting a random_state has no effect since shuffle is False. This will raise an error in 0.24. You should leave random_state to its default (None), or set shuffle=True.

FutureWarning

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits

```
[CV] .....
[CV] ..... , total= 0.7s
[CV] .....
```

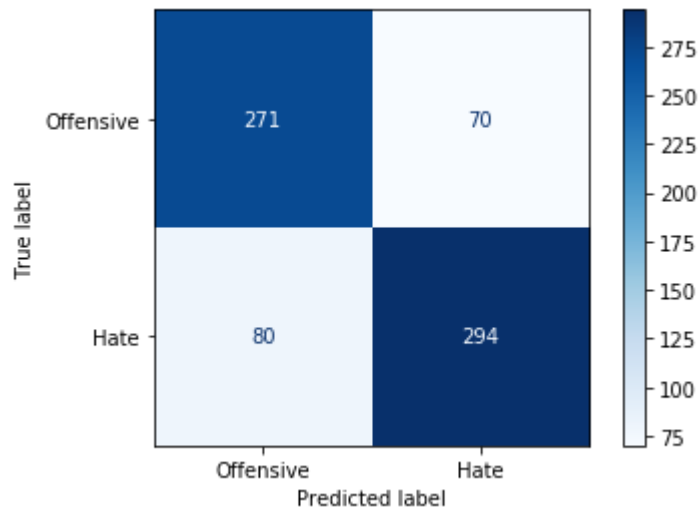
[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 0.7s remaining: 0.0s

```
[CV] ..... , total= 0.9s
[CV] .....
[CV] ..... , total= 0.8s
[CV] .....
[CV] ..... , total= 0.9s
[CV] .....
[CV] ..... , total= 0.8s
```

[Parallel(n_jobs=1)]: Done 5 out of 5 | elapsed: 4.2s finished

```
In [15]: from sklearn.metrics import plot_confusion_matrix
class_names = ['Offensive', 'Hate']
plot_confusion_matrix(lg, X_test, y_test, cmap=plt.cm.Blues, display_labels =
```

```
Out[15]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1840ee14088>
```



```
In [43]: from sklearn.metrics import classification_report
report_lg_BOW = classification_report( y_test, y_preds_lg)
print(report_lg_BOW)
```

	precision	recall	f1-score	support
0	0.77	0.79	0.78	341
1	0.81	0.79	0.80	374
accuracy			0.79	715
macro avg	0.79	0.79	0.79	715
weighted avg	0.79	0.79	0.79	715

```
In [16]: ► importance_logreg = lg.best_estimator_.coef_.tolist()[0]

features = list(df_BOW.columns)
feature_importance_logreg = pd.DataFrame(list(zip(features, importance_logreg)))
feature_importance_logreg = feature_importance_logreg.sort_values(by='importance')
feature_importance_logreg.head(20)
```

Out[16]:

	features	importance
1455	faggit	-2.108559
3030	niggaz	-2.079960
3027	niggahs	-1.998250
1456	faggot	-1.949334
3033	niggerous	-1.639718
1732	gave	-1.594995
3023	nigerian	-1.555284
2411	kike	-1.518631
4949	whistle	-1.385182
3546	queen	-1.284641
444	black	-1.274186
1305	dwn	-1.250660
1458	fagjo	-1.218483
1453	facts	-1.196915
911	cool	-1.137583
3283	pennsylvanians	-1.103863
4160	sperm	-1.076972
851	comfortable	-1.046262
348	beaner	-1.045217
4061	smfh	-1.037146

```
In [17]: tree = GridSearchCV(DecisionTreeClassifier(),
                             param_grid,
                             cv=KFold(n_splits=5,
                                       random_state=42).split(X_train,
                                       y_train),
                             verbose=2)

y_preds_tree = tree.fit(X_train, y_train).predict(X_test)
```

C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection_split.py:296: FutureWarning: Setting a random_state has no effect since shuffle is False. This will raise an error in 0.24. You should leave random_state to its default (None), or set shuffle=True.

FutureWarning

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits

```
[CV] .....
[CV] ..... , total= 1.1s
[CV] .....
```

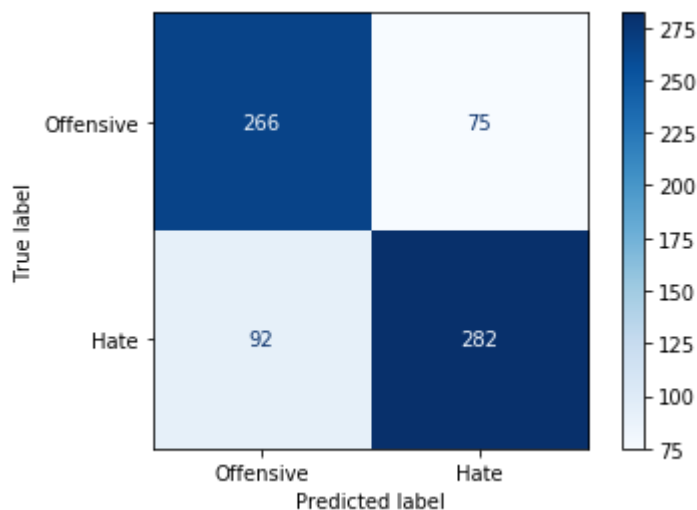
[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 1.0s remaining: 0.0s

```
[CV] ..... , total= 1.1s
[CV] .....
[CV] ..... , total= 1.2s
[CV] .....
[CV] ..... , total= 1.8s
[CV] .....
[CV] ..... , total= 1.9s
```

[Parallel(n_jobs=1)]: Done 5 out of 5 | elapsed: 7.2s finished

```
In [18]: plot_confusion_matrix(tree, X_test, y_test, cmap=plt.cm.Blues, display_labels=
```

Out[18]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x18403d0a0c8>




```
In [19]: > report_tree = classification_report( y_test, y_preds_tree)
print(report_tree)
```

	precision	recall	f1-score	support
0	0.74	0.78	0.76	341
1	0.79	0.75	0.77	374
accuracy			0.77	715
macro avg	0.77	0.77	0.77	715
weighted avg	0.77	0.77	0.77	715

```
In [20]: > importance_tree = tree.best_estimator_.feature_importances_.tolist()

features = list(df_BOW.columns)
feature_importance_logreg = pd.DataFrame(list(zip(features,importance_tree)),
feature_importance_logreg = feature_importance_logreg.sort_values(by='importance')
feature_importance_logreg.head(20)
```

Out[20]:

	features	importance
432	bitch	0.090740
434	bitches	0.072084
3535	pussies	0.069759
2046	hoeing	0.064999
2044	hockey	0.054338
3027	niggahs	0.025053
3023	nigerian	0.023572
5138	zzzzzz	0.012307
3025	niggaa	0.008869
4949	whistle	0.007725
1305	dwn	0.007683
1667	fucking	0.007497
224	ass	0.007476
1676	fuckin	0.007473
3937	shirts	0.006702
423	bird	0.006569
4683	tv	0.006502
4759	upset	0.006126
4716	ugliest	0.005897
1942	hat	0.005024

Word2vec embedding

```
In [21]:  from gensim.models import word2vec
import nltk
```

```
In [22]:  feature_size = 100    # Word vector dimensionality
window_context = 30           # Context window size
min_word_count = 1           # Minimum word count
sample = 1e-3                 # Downsample setting for frequent words

wpt = nltk.WordPunctTokenizer()
tokenized_corpus = [wpt.tokenize(document) for document in corpus]

# Set values for various parameters
feature_size = 100    # Word vector dimensionality
window_context = 30           # Context window size
min_word_count = 1           # Minimum word count
sample = 1e-3                 # Downsample setting for frequent words

w2v_model = word2vec.Word2Vec(tokenized_corpus, size=feature_size,
                              window=window_context, min_count=min_word_count,
                              sample=sample, iter=50)

# view similar words based on gensim's model
similar_words = {search_term: [item[0] for item in w2v_model.wv.most_similar(
    search_term) for search_term in ['hate', 'love', 'nigger', 'faggot', 'bitch']]
similar_words
```

```
Out[22]: {'hate': ['goddamit', 'cripples', 'escape', 'dairy', 'ing'],
'love': ['mitchell', 'bread', 'victoria', 'baltimore', 'emojis'],
'nigger': ['hoodrats', 'ebloa', 'kidnapped', 'traditions', 'tyler'],
'faggot': ['tear', 'little', 'bitching', 'ultimate', 'fag'],
'bitch': ['next', 'knowin', 'tf', 'mobbin', 'meal'],
'pussy': ['stank', 'swimm', 'strap', 'poo', 'another'],
'cracker': ['hypocrisy', 'friday', 'fathom', 'blocked', 'statement'],
'nigga': ['lame', 'scary', 'yah', 'dont', 'fuk'],
'homo': ['bullet', 'snapchat', 'cortez', 'twisted', 'fosters'],
'cunt': ['profile', 'managers', 'piss', 'dress', 'pm'],
'fuck': ['fish', 'hmm', 'zima', 'kermit', 'whiney'],
'trash': ['white',
'westbrook',
'trailer',
'hashtagherehashtaghere',
'doesnt'],
'queer': ['yost', 'pathetic', 'traitor', 'project', 'lbum']}
```

```
In [23]: ▶ def average_word_vectors(words, model, vocabulary, num_features):

    feature_vector = np.zeros((num_features,), dtype="float64")
    nwords = 0.

    for word in words:
        if word in vocabulary:
            nwords = nwords + 1.
            feature_vector = np.add(feature_vector, model[word])

    if nwords:
        feature_vector = np.divide(feature_vector, nwords)

    return feature_vector

def averaged_word_vectorizer(corpus, model, num_features):
    vocabulary = set(model.wv.index2word)
    features = [average_word_vectors(tokenized_sentence, model, vocabulary, num_features)
                for tokenized_sentence in corpus]
    return np.array(features)

w2v_feature_array = averaged_word_vectorizer(corpus=tokenized_corpus, model=model,
                                              num_features=feature_size)

pd.DataFrame(w2v_feature_array)
```

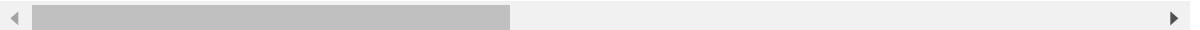
C:\Users\seanx\anaconda3\lib\site-packages\ipykernel_launcher.py:9: DeprecationWarning: Call to deprecated `__getitem__` (Method will be removed in 4.0.0, use self.wv.__getitem__() instead).

```
if __name__ == '__main__':
```

Out[23]:

	0	1	2	3	4	5	6	7	
0	0.727941	0.244606	0.683794	-0.061855	-0.406099	-0.468852	0.655809	-0.275200	-0.
1	0.842390	0.307193	0.882716	0.294135	-0.236299	-0.177043	0.579353	-0.165394	-0.
2	0.546560	0.224021	0.425261	-0.028157	-0.306209	-0.039596	0.662744	-0.180299	-0.
3	0.692076	0.149320	-0.006210	-0.070063	-0.443279	-0.198863	0.479766	-0.269004	-0.
4	0.518252	0.081868	0.732512	-0.030877	-0.265558	0.084669	0.658960	-0.165031	-0.
...
2855	-0.051713	-0.128862	0.805924	-0.030887	-0.387116	-0.644496	0.266638	0.223754	-0.
2856	-1.109598	-0.135360	-0.197205	-0.557631	-0.728592	-0.173033	0.391669	0.502999	-1.
2857	0.140394	0.190423	0.698273	0.058907	-0.212432	-0.033794	0.315742	0.242145	-0.
2858	0.718705	0.362663	0.732503	-0.118662	-0.687060	0.056234	0.649663	-0.468604	-0.
2859	0.547129	0.186977	0.305199	-0.090508	-0.358534	-0.156237	0.567174	-0.303735	-0.

2860 rows × 100 columns

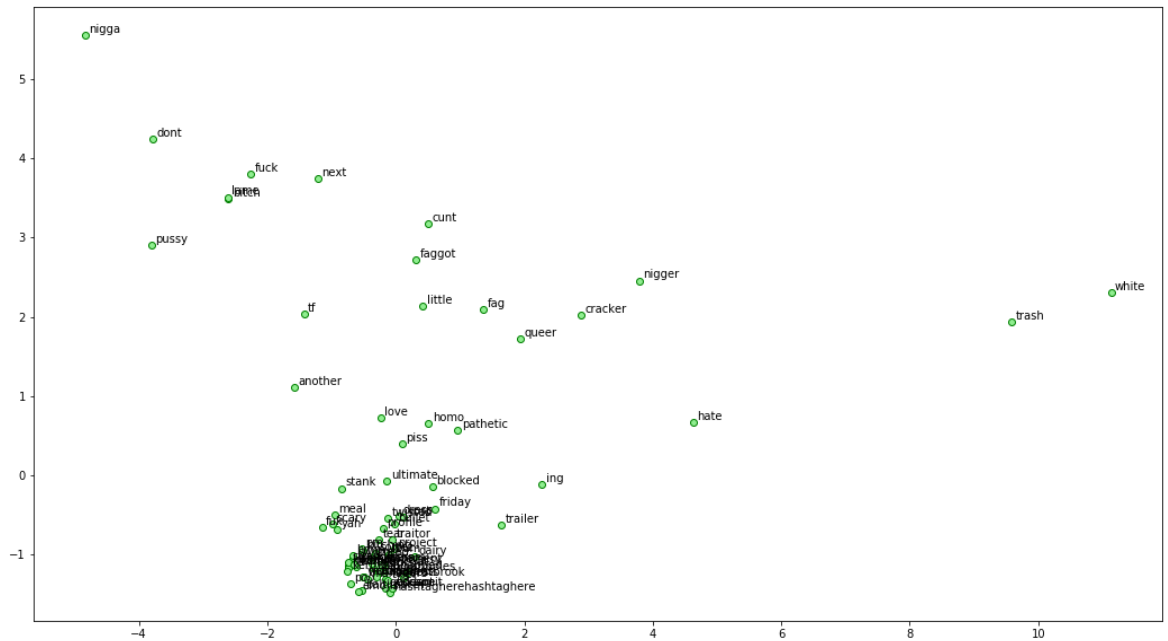


```
In [24]: ▶ from sklearn.decomposition import PCA

words = sum([[k] + v for k, v in similar_words.items()], [])
wvs = w2v_model.wv[words]

pca = PCA(n_components=2)
np.set_printoptions(suppress=True)
P = pca.fit_transform(wvs)
labels = words

plt.figure(figsize=(18, 10))
plt.scatter(P[:, 0], P[:, 1], c='lightgreen', edgecolors='g')
for label, x, y in zip(labels, P[:, 0], P[:, 1]):
    plt.annotate(label, xy=(x+0.06, y+0.03), xytext=(0, 0), textcoords='offset')
```



```
In [25]: X_w2v = pd.DataFrame(w2v_feature_array)
         y = df['class'].astype(int)
         X_train_w2v, X_test_w2v, y_train_w2v, y_test_w2v = train_test_split(X_w2v, y,
```

```
In [26]: lg_w2v = GridSearchCV(LogisticRegression(max_iter = 1000),
                                param_grid,
                                cv=KFold(n_splits=5,
                                           random_state=42).split(X_train_
                                verbose=2)
                                y_preds_w2v_lg = lg_w2v.fit(X_train_w2v, y_train_w2v).predict(X_test_w2v)
```

C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection_split.py:296: FutureWarning: Setting a random_state has no effect since shuffle is False. This will raise an error in 0.24. You should leave random_state to its default (None), or set shuffle=True.

FutureWarning

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 0.0s remaining: 0.0s

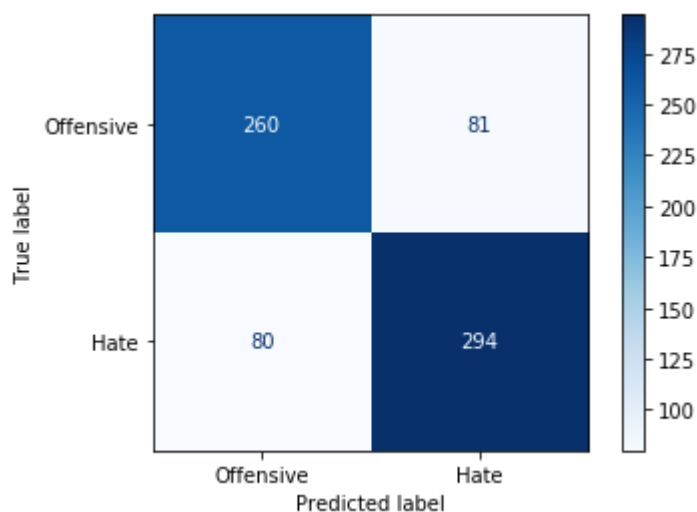
Fitting 5 folds for each of 1 candidates, totalling 5 fits

```
[CV] .....
[CV] ..... , total= 0.1s
[CV] .....
[CV] ..... , total= 0.1s
[CV] .....
[CV] ..... , total= 0.1s
[CV] .....
[CV] ..... , total= 0.1s
[CV] .....
[CV] ..... , total= 0.1s
```

[Parallel(n_jobs=1)]: Done 5 out of 5 | elapsed: 0.3s finished

```
In [27]: plot_confusion_matrix(lg_w2v, X_test_w2v, y_test_w2v, cmap=plt.cm.Blues, disp
```

Out[27]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1840ede0708>



```
In [28]: > report_w2v = classification_report( y_test_w2v, y_preds_w2v_lg)
print(report_w2v)
```

	precision	recall	f1-score	support
0	0.76	0.76	0.76	341
1	0.78	0.79	0.79	374
accuracy			0.77	715
macro avg	0.77	0.77	0.77	715
weighted avg	0.77	0.77	0.77	715

```
In [29]: > tree_w2v = GridSearchCV(DecisionTreeClassifier(),
                                param_grid,
                                cv=KFold(n_splits=5,
                                random_state=42).split(X_train_
                                verbose=2)
y_preds_w2v_tree = tree_w2v.fit(X_train_w2v, y_train_w2v).predict(X_test_w2v)
```

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[CV]

C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection_split.py:296: FutureWarning: Setting a random_state has no effect since shuffle is False. This will raise an error in 0.24. You should leave random_state to its default (None), or set shuffle=True.

FutureWarning

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 0.1s remaining: 0.0s

[CV] , total= 0.2s

[CV]

[CV] , total= 0.2s

[CV]

[CV] , total= 0.2s

[CV]

[CV] , total= 0.2s

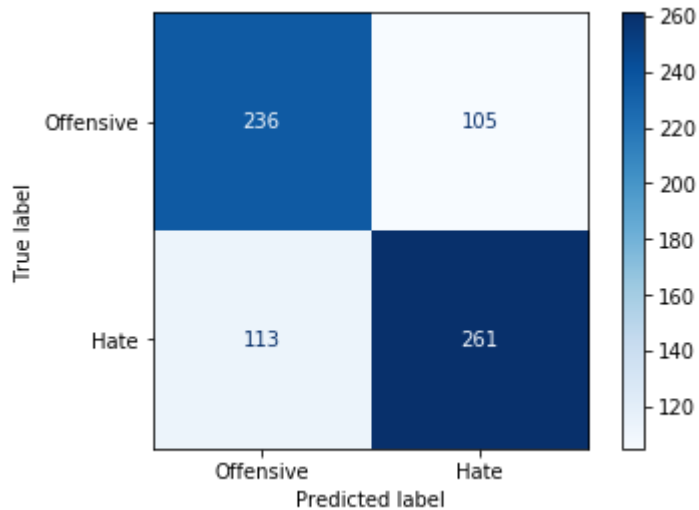
[CV]

[CV] , total= 0.2s

[Parallel(n_jobs=1)]: Done 5 out of 5 | elapsed: 0.8s finished

In [30]: `plot_confusion_matrix(tree_w2v, X_test_w2v, y_test_w2v, cmap=plt.cm.Blues, di`

Out[30]: `<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1841d3721c8>`



In [31]: `report_w2v_tree = classification_report(y_test_w2v, y_preds_w2v_tree)`
`print(report_w2v_tree)`

	precision	recall	f1-score	support
0	0.68	0.69	0.68	341
1	0.71	0.70	0.71	374
accuracy			0.70	715
macro avg	0.69	0.69	0.69	715
weighted avg	0.70	0.70	0.70	715

Combining BOW and W2V

In [32]: `X_mixed = pd.concat([pd.DataFrame(w2v_feature_array), df_BOW.drop(columns=['c`
`y = df['class'].astype(int)`
`X_train_mixed, X_test_mixed, y_train_mixed, y_test_mixed = train_test_split(X`

```
In [33]: lg_mixed = GridSearchCV(LogisticRegression(max_iter = 1000),
                                param_grid,
                                cv=KFold(n_splits=5,
                                           random_state=42).split(X_train_
                                verbose=2)
y_preds_mixed = lg_mixed.fit(X_train_mixed, y_train_mixed).predict(X_test_mixed)
```

C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection_split.py:296: FutureWarning: Setting a random_state has no effect since shuffle is False. This will raise an error in 0.24. You should leave random_state to its default (None), or set shuffle=True.

FutureWarning
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits

```
[CV] .....
[CV] ..... , total= 1.5s
[CV] .....
```

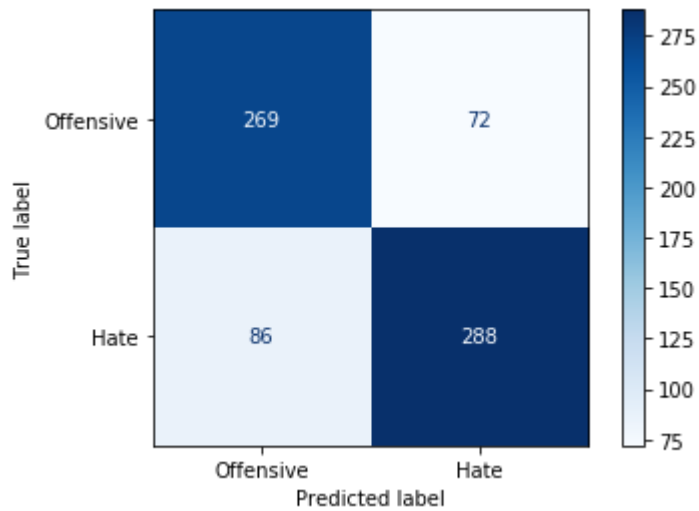
[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 1.4s remaining: 0.0s

```
[CV] ..... , total= 1.5s
[CV] .....
[CV] ..... , total= 1.3s
[CV] .....
[CV] ..... , total= 1.3s
[CV] .....
[CV] ..... , total= 1.4s
```

[Parallel(n_jobs=1)]: Done 5 out of 5 | elapsed: 6.9s finished

In [34]: `plot_confusion_matrix(lg_mixed, X_test_mixed, y_test_mixed, cmap=plt.cm.Blues)`

Out[34]: `<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1841d429808>`



In [35]: `report_lg_mixed = classification_report(y_test_mixed, y_preds_mixed)`
`print(report_lg_mixed)`

	precision	recall	f1-score	support
0	0.76	0.79	0.77	341
1	0.80	0.77	0.78	374
accuracy			0.78	715
macro avg	0.78	0.78	0.78	715
weighted avg	0.78	0.78	0.78	715

```
In [36]: features = list(X_mixed.columns)
feature_importance_logreg = pd.DataFrame(list(zip(features, importance_logreg)))
feature_importance_logreg = feature_importance_logreg.sort_values(by='importance_logreg', ascending=False)
feature_importance_logreg.head(20)
```

Out[36]:

	features	importance
1455	emoji	-2.108559
3030	muslims	-2.079960
3027	murdered	-1.998250
1456	emojis	-1.949334
3033	muthafucka	-1.639718
1732	found	-1.594995
3023	muhhfuckin	-1.555284
2411	jezzy	-1.518631
4949	wannabe	-1.385182
3546	pray	-1.284641
444	bc	-1.274186
1305	dm	-1.250660
1458	encrusted	-1.218483
1453	emm	-1.196915
911	closer	-1.137583
3283	otter	-1.103863
4160	smfh	-1.076972
851	chill	-1.046262
348	aunt	-1.045217
4061	shout	-1.037146

```
In [37]: tree_mixed = GridSearchCV(DecisionTreeClassifier(),
                                   param_grid,
                                   cv=KFold(n_splits=5,
                                             random_state=42).split(X_train_
                                             verbose=2)
                                   y_preds_mixed_tree = tree_mixed.fit(X_train_mixed, y_train_mixed).predict(X_t
```

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[CV]

C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection_split.p
y:296: FutureWarning: Setting a random_state has no effect since shuffle is
False. This will raise an error in 0.24. You should leave random_state to i
ts default (None), or set shuffle=True.

FutureWarning

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.

[CV] , total= 1.3s

[CV]

[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 1.2s remaining:
0.0s

[CV] , total= 1.4s

[CV]

[CV] , total= 1.5s

[CV]

[CV] , total= 1.3s

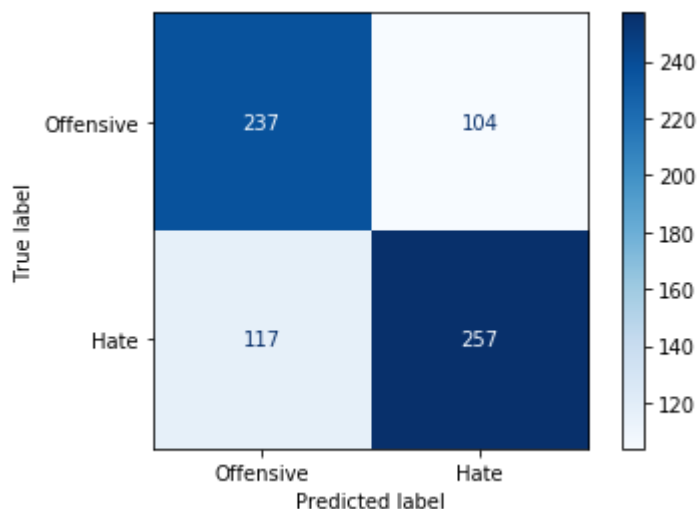
[CV]

[CV] , total= 1.3s

[Parallel(n_jobs=1)]: Done 5 out of 5 | elapsed: 6.7s finished

```
In [38]: plot_confusion_matrix(tree_mixed, X_test_mixed, y_test_mixed, cmap=plt.cm.Blu
```

Out[38]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x18410ae
8648>



```
In [39]: ► report_tree_mixed = classification_report( y_test_mixed, y_preds_mixed_tree)
print(report_tree_mixed)
```

	precision	recall	f1-score	support
0	0.67	0.70	0.68	341
1	0.71	0.69	0.70	374
accuracy			0.69	715
macro avg	0.69	0.69	0.69	715
weighted avg	0.69	0.69	0.69	715

Using LIME to interpret predictions

```

In [53]: import lime
import lime.lime_tabular

i = np.random.randint(0, X_test.shape[0])

explainer = lime.lime_tabular.LimeTabularExplainer(training_data = X_train.to
                                                    mode = 'classification',
                                                    feature_names = features,
                                                    class_names = ['Hate', 'Off

exp = explainer.explain_instance(data_row = X_test.iloc[i].to_numpy(),
                                predict_fn = lg.predict_proba)

actual = df_BOW['class'][i]

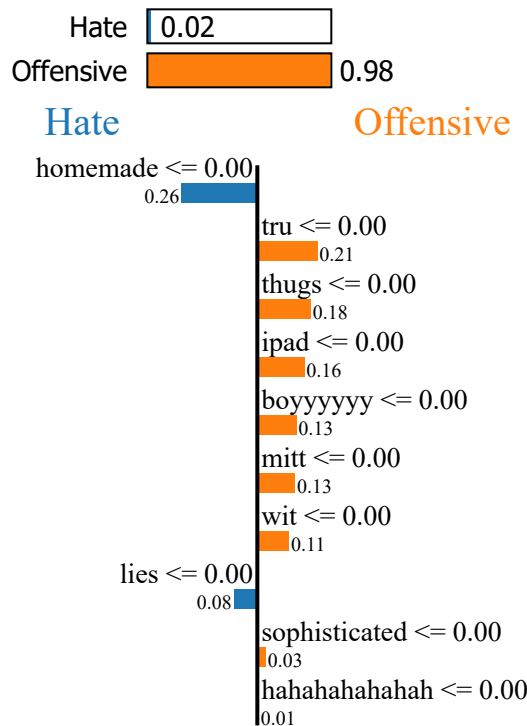
if actual == 0:
    actual = 'Hate'
else:
    actual = 'Offensive'

print(f'Actual classification: {actual}')
exp.show_in_notebook()

```

Actual classification: Offensive

Prediction probabilities



Feature Value

homemade	0.00
tru	0.00
thugs	0.00
ipad	0.00
boyyyyyy	0.00

.....	
mitt	0.00
wit	0.00
lies	0.00
sophisticated	0.00

In []: