Title:

Twitter Data Analysis

Problem Statement:

Use Twitter data for sentimental analysis. The dataset is 3 MB in size & has 31,962 tweets.

Objectives:

To classify tweets as hate tweets or not.

Outcome:

Identifying and removing hate tweets from twitter.

Software and Hardware Requirement:

- Python 3
- Jupiter notebook
- 64-bit OS
- 4GB RAM

Theory:

NLP:

It is subfield of linguistics, CS and AI concerned with interaction between computer ad human language in particular how to program computer to process and analyze large amounts of natural language data.

Stop words are words that are filtering out before/after the natural language data is processed.

Stemming for grammatical reasons text can use different forms of words There are also families of derivatively related words with similar meaning.

Stemming reduces inflectional forms and sometimes derivationally linked forms of a word to its common used form.

When applied to document the result like

ORIGINAL: the boy’s cars are different color

STEMMED: - The by car be different color

Feature selection: -

It is process of selection a subset of terms occurring in training set & using only the subset of feature in test classification.

Code:

```
import re
import pandas as pd
pd.set_option("display.max_colwidth", 200)
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
%matplotlib inline


train  = pd.read_csv('dataset/tweets_train.csv')
test = pd.read_csv('dataset/tweets_test.csv')


train.shape, test.shape


train["label"].value_counts()
```

Data Cleaning

```
def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for i in r:
        input_txt = re.sub(i, '', input_txt)

    return input_txt
```

1. Removing Twitter Handles (@user)

```
train['tidy_tweet'] = np.vectorize(remove_pattern)(train['tweet'], "@[\w]*
")
train.head()
```

2. Removing Punctuations, Numbers, and Special Characters

```
train['tidy_tweet'] = train['tidy_tweet'].str.replace("[^a-zA-Z#]", " ")
train.head(10)
```

3. Removing Short Words

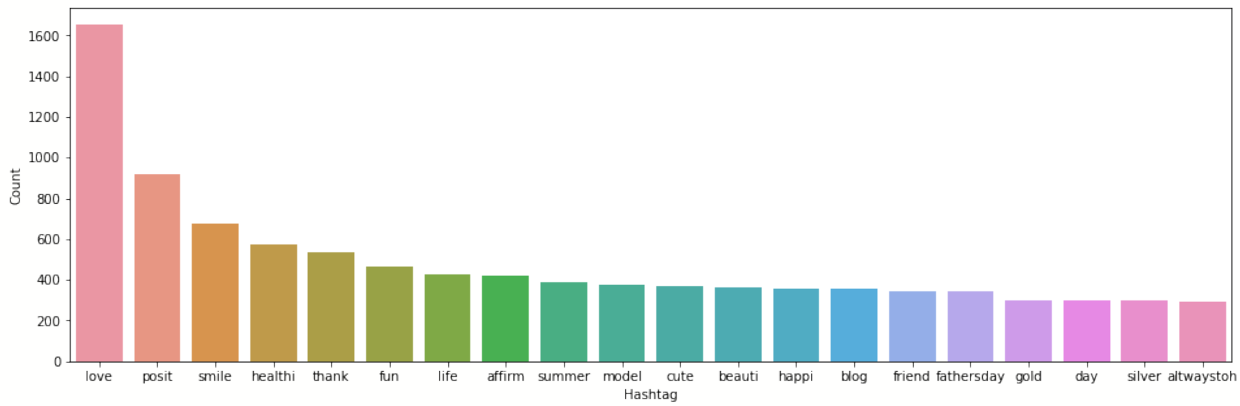
```
train['tidy_tweet'] = train['tidy_tweet'].apply(lambda x: ' '.join([w for
w in x.split() if len(w)>3]))
train.head()

tokenized_tweet = train['tidy_tweet'].apply(lambda x: x.split()) #tokeniza
tion

tokenized_tweet.head()

from nltk.stem.porter import *
stemmer = PorterStemmer()

tokenized_tweet = tokenized_tweet.apply(lambda x: [stemmer.stem(i) for i i
n x]) # stemming

for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = ' '.join(tokenized_tweet[i])

train['tidy_tweet'] = tokenized_tweet

def hashtag_extract(x):
    hashtags = []
    for i in x:
        ht = re.findall(r"#(\w+)", i)
        hashtags.append(ht)

    return hashtags

# extracting hashtags from normal tweets
HT_regular = hashtag_extract(train['tidy_tweet'][train['label'] == 0])

# extracting hashtags from hate tweets tweets
HT_negative = hashtag_extract(train['tidy_tweet'][train['label'] == 1])
```

```
# unnesting list
HT_regular = sum(HT_regular,[])
HT_negative = sum(HT_negative,[])

# Non Hate Tweets
a = nltk.FreqDist(HT_regular)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})

# selecting top 20 most frequent hashtags
d = d.nlargest(columns="Count", n = 20)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()

# Hate tweets

b = nltk.FreqDist(HT_negative)
e = pd.DataFrame({'Hashtag': list(b.keys()), 'Count': list(b.values())})

# selecting top 20 most frequent hashtags
e = e.nlargest(columns="Count", n = 20)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=e, x= "Hashtag", y = "Count")


from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

X = train["tidy_tweet"]
y = train["label"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=1)

vectorizer = TfidfVectorizer()
train_vectors = vectorizer.fit_transform(X_train)
test_vectors = vectorizer.transform(X_test)
print(train_vectors.shape, test_vectors.shape)

lreg = LogisticRegression()
lreg.fit(train_vectors,y_train)
```

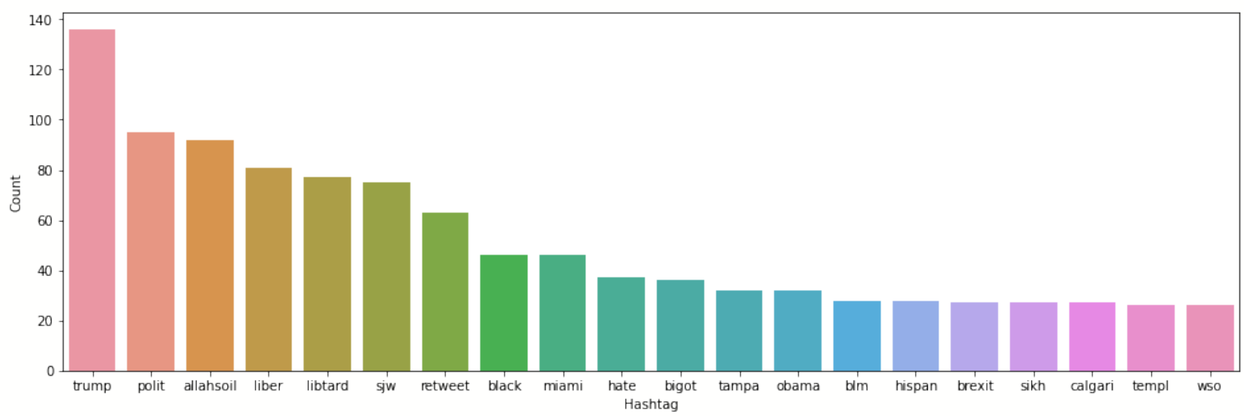
```
from  sklearn.metrics  import accuracy_score
from sklearn import metrics

predicted = lreg.predict(test_vectors)

print("Accuracy:",accuracy_score(y_test,predicted))
print("Precision:",metrics.precision_score(y_test, predicted))
print("Recall:",metrics.recall_score(y_test, predicted))
```

Output:

Non Hate:



Hate:




```
Accuracy: 0.9489002654531665
Precision: 0.8836206896551724
Recall: 0.2859135285913529
```

Conclusion:

Thus Successfully implemented and classified tweets as hate tweets or not.