Muhammad Hammad Latif FA18-BCS-134

Rana Muhammad Sobaan FA18-BCS-038

Importing Libraries

In [1]:
```python
import pandas as pd
import re
import numpy as np
import nltk

from nltk.tokenize import TweetTokenizer
from nltk.stem import PorterStemmer
from nltk import ngrams

from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import pickle
```

Reading Training Dataset

We are using _ as a seperator/ delimeter

In [2]:
```python
trainData = pd.read_csv("TrainData.csv","_")

trainData
```

Out[2]:

	Index	Comment	Polarity
0	0	time to eat with my best buddy! #lunch	Happy
1	1	@user @user if they want reflection money. #ksleg	Happy
2	2	---Good job but I' will expect a lot more in f...	Happy
3	3	totally dissatisfied with the service###%%@@ n...	Sad
4	4	loved my work!!!!!	Happy
5	5	Worst customer care service......@@$$$angry	Sad
6	6	Brilliant effort guys!!!	Happy
7	7	@user @user you point one finger @user million...	Sad
8	8	words r free, it's how u use that can cost you...	Happy
9	9	you might be a libtard if... #libtard #sjw #li...	Sad

Reading Test Dataset

In [3]:
```python
testData = pd.read_csv("TestData.csv","_")

testData
```

Out[3]:

	Index	Comment	Polarity
0	0	@use the pic says otherwise for young girls co...	Sad
1	1	#good night! ?? #faith ever #vaitacacommafiasdv	Happy
2	2	@user when you're blocked by a troll because y...	Sad
3	3	dinner with sister!!	Happy
4	4	who else is planning on watching @user tomorrow?	happy

PreProcessing Begins

In [4]:
```python
#Function to remove @user in the data

def remove_pattern(text,pattern):

    # re.findall() finds the pattern i.e @user and creating list
    r = re.findall(pattern,text)

    # re.sub() removes @user from the sentences in the dataset
    for i in r:
        text = re.sub(i,"",text)

    return text
```

In [5]:
```python
#Applying function to both datasets for removing @user

trainData['Processed_Comment'] = np.vectorize(remove_pattern)(trainData['Comment'], "@[

testData['Processed_Comment'] = np.vectorize(remove_pattern)(testData['Comment'], "@[
```

In [6]:
```python
# Removing everything except text i.e letters/words
trainData['Processed_Comment'] = trainData['Processed_Comment'].str.replace("[^a-zA-Z

trainData
```

```
<ipython-input-6-ec126bf61b6b>:3: FutureWarning: The default value of regex will chang
e from True to False in a future version.
  trainData['Processed_Comment'] = trainData['Processed_Comment'].str.replace("[^a-zA-
Z]", " ")
```

Out[6]:

	Index	Comment	Polarity	Processed_Comment
0	0	time to eat with my best buddy! #lunch	Happy	time to eat with my best buddy lunch
1	1	@user @user if they want reflection money. #ksleg	Happy	if they want reflection money ksleg
2	2	---Good job but I' will expect a lot more in f...	Happy	Good job but I will expect a lot more in f...
3	3	totally dissatisfied with the service###%%@@ n...	Sad	totally dissatisfied with the service nev...
4	4	loved my work!!!!!	Happy	loved my work
5	5	Worst customer care service......@@$$$angry	Sad	Worst customer care service angry
6	6	Brilliant effort guys!!!	Happy	Brilliant effort guys
7	7	@user @user you point one finger @user million...	Sad	you point one finger millions are pointed r...
8	8	words r free, it's how u use that can cost you...	Happy	words r free it s how u use that can cost you...
9	9	you might be a libtard if... #libtard #sjw #li...	Sad	you might be a libtard if libtard sjw li...

In [7]:
```python
# Removing everything except text i.e letters/words
testData['Processed_Comment'] = testData['Processed_Comment'].str.replace("[^a-zA-Z]",

testData
```

```
<ipython-input-7-01d6dab056a2>:3: FutureWarning: The default value of regex will chang
e from True to False in a future version.
  testData['Processed_Comment'] = testData['Processed_Comment'].str.replace("[^a-zA-
Z]", " ")
```

Out[7]:

	Index	Comment	Polarity	Processed_Comment
0	0	@use the pic says otherwise for young girls co...	Sad	the pic says otherwise for young girls confin...
1	1	#good night! ?? #faith ever #vaitacacommafiasdv	Happy	good night faith ever vaitacacommafiasdv
2	2	@user when you're blocked by a troll because y...	Sad	when you re blocked by a troll because you pr...
3	3	dinner with sister!!	Happy	dinner with sister
4	4	who else is planning on watching @user tomorrow?	happy	who else is planning on watching tomorrow

In [8]:
```python
#Removing Short Words
trainData['Processed_Comment'] = trainData['Processed_Comment'].apply(lambda x: ' '.jo

trainData
```

Out[8]:

	Index	Comment	Polarity	Processed_Comment
0	0	time to eat with my best buddy! #lunch	Happy	time eat with best buddy lunch
1	1	@user @user if they want reflection money. #ksleg	Happy	they want reflection money ksleg
2	2	---Good job but I' will expect a lot more in f...	Happy	Good job but will expect lot more future
3	3	totally dissatisfied with the service###%%@@ n...	Sad	totally dissatisfied with the service never us...
4	4	loved my work!!!!!	Happy	loved work
5	5	Worst customer care service......@@$$$angry	Sad	Worst customer care service angry
6	6	Brilliant effort guys!!!	Happy	Brilliant effort guys
7	7	@user @user you point one finger @user million...	Sad	you point one finger millions are pointed righ...
8	8	words r free, it's how u use that can cost you...	Happy	words free how use that can cost you verbal ab...
9	9	you might be a libtard if... #libtard #sjw #li...	Sad	you might libtard libtard sjw liberal politics

In [9]:
```python
#Removing Short Words
testData['Processed_Comment'] = testData['Processed_Comment'].apply(lambda x: ' '.join

testData
```

Out[9]:

	Index	Comment	Polarity	Processed_Comment
0	0	@use the pic says otherwise for young girls co...	Sad	the pic says otherwise for young girls confine...
1	1	#good night! ?? #faith ever #vaitacacommafiasdv	Happy	good night faith ever vaitacacommafiasdv
2	2	@user when you're blocked by a troll because y...	Sad	when you blocked troll because you promise bla...
3	3	dinner with sister!!	Happy	dinner with sister
4	4	who else is planning on watching @user tomorrow?	happy	who else planning watching tomorrow

Making sure that all Happy and Sad are same

In [10]:
```python
trainData['Polarity'] = trainData['Polarity'].apply(lambda x: x.capitalize())

trainData
```

Out[10]:

	Index	Comment	Polarity	Processed_Comment
0	0	time to eat with my best buddy! #lunch	Happy	time eat with best buddy lunch
1	1	@user @user if they want reflection money. #ksleg	Happy	they want reflection money ksleg
2	2	---Good job but I' will expect a lot more in f...	Happy	Good job but will expect lot more future
3	3	totally dissatisfied with the service###%%@@ n...	Sad	totally dissatisfied with the service never us...
4	4	loved my work!!!!!	Happy	loved work
5	5	Worst customer care service......@@$$$angry	Sad	Worst customer care service angry
6	6	Brilliant effort guys!!!	Happy	Brilliant effort guys
7	7	@user @user you point one finger @user million...	Sad	you point one finger millions are pointed righ...
8	8	words r free, it's how u use that can cost you...	Happy	words free how use that can cost you verbal ab...
9	9	you might be a libtard if... #libtard #sjw #li...	Sad	you might libtard libtard sjw liberal politics

In [11]:
```python
testData['Polarity'] = testData['Polarity'].apply(lambda x: x.capitalize())

testData
```

Out[11]:

	Index	Comment	Polarity	Processed_Comment
0	0	@use the pic says otherwise for young girls co...	Sad	the pic says otherwise for young girls confine...
1	1	#good night! ?? #faith ever #vaitacacommafiasdv	Happy	good night faith ever vaitacacommafiasdv
2	2	@user when you're blocked by a troll because y...	Sad	when you blocked troll because you promise bla...
3	3	dinner with sister!!	Happy	dinner with sister
4	4	who else is planning on watching @user tomorrow?	Happy	who else planning watching tomorrow

Label Encoding for Train/Test Data

In [12]:
```python
def labelEncoder(polarity):
    if(polarity == 'Happy'):
        return 1
    return 0
```

In [13]:
```python
trainData['Polarity'] = trainData['Polarity'].apply(lambda x: labelEncoder(x))

trainData
```

Out[13]:

	Index	Comment	Polarity	Processed_Comment
0	0	time to eat with my best buddy! #lunch	1	time eat with best buddy lunch
1	1	@user @user if they want reflection money. #ksleg	1	they want reflection money ksleg
2	2	---Good job but I' will expect a lot more in f...	1	Good job but will expect lot more future
3	3	totally dissatisfied with the service###%%@@ n...	0	totally dissatisfied with the service never us...
4	4	loved my work!!!!!	1	loved work
5	5	Worst customer care service......@@$$$angry	0	Worst customer care service angry
6	6	Brilliant effort guys!!!	1	Brilliant effort guys
7	7	@user @user you point one finger @user million...	0	you point one finger millions are pointed righ...
8	8	words r free, it's how u use that can cost you...	1	words free how use that can cost you verbal ab...
9	9	you might be a libtard if... #libtard #sjw #li...	0	you might libtard libtard sjw liberal politics

In [14]:
```python
testData['Polarity'] = testData['Polarity'].apply(lambda x: labelEncoder(x))

testData
```

Out[14]:

	Index	Comment	Polarity	Processed_Comment
0	0	@use the pic says otherwise for young girls co...	0	the pic says otherwise for young girls confine...
1	1	#good night! ?? #faith ever #vaitacacommafiasdv	1	good night faith ever vaitacacommafiasdv
2	2	@user when you're blocked by a troll because y...	0	when you blocked troll because you promise bla...
3	3	dinner with sister!!	1	dinner with sister
4	4	who else is planning on watching @user tomorrow?	1	who else planning watching tomorrow

Tokanizing Comments of train data

In [15]:
```python
tokenized_trainComment = trainData['Processed_Comment'].apply(lambda x: x.split())

tokenized_trainComment
```

Out[15]:
```
0               [time, eat, with, best, buddy, lunch]
1              [they, want, reflection, money, ksleg]
2    [Good, job, but, will, expect, lot, more, future]
3    [totally, dissatisfied, with, the, service, ne...
4                                       [loved, work]
5             [Worst, customer, care, service, angry]
6                           [Brilliant, effort, guys]
7    [you, point, one, finger, millions, are, point...
8    [words, free, how, use, that, can, cost, you, ...
9    [you, might, libtard, libtard, sjw, liberal, p...
Name: Processed_Comment, dtype: object
```

POS Tagging

In [16]:
```python
#nltk.download('punkt')
#nltk.download('averaged_perceptron_tagger')
trainDataList = trainData['Processed_Comment'].tolist()
taggedList = list()
posList = list()

for sentence in trainDataList:
        tokenized = nltk.word_tokenize(sentence)
        taggedList.append(nltk.pos_tag(tokenized))

#removing repititions
for tList in taggedList:
    for word_tuple in tList:
        if word_tuple not in posList:
            posList.append(word_tuple)

posList
```

Out[16]:
```
[('time', 'NN'),
 ('eat', 'NN'),
 ('with', 'IN'),
 ('best', 'JJS'),
 ('buddy', 'NN'),
 ('lunch', 'NN'),
 ('they', 'PRP'),
 ('want', 'VBP'),
 ('reflection', 'NN'),
 ('money', 'NN'),
 ('ksleg', 'NN'),
 ('Good', 'JJ'),
 ('job', 'NN'),
 ('but', 'CC'),
 ('will', 'MD'),
 ('expect', 'VB'),
 ('lot', 'NN'),
 ('more', 'JJR'),
 ('future', 'JJ'),
 ('totally', 'RB'),
 ('dissatisfied', 'VBN'),
 ('the', 'DT'),
 ('service', 'NN'),
 ('never', 'RB'),
 ('use', 'VBD'),
 ('thi', 'JJ'),
 ('again', 'RB'),
 ('loved', 'VBN'),
 ('work', 'NN'),
 ('Worst', 'NNP'),
 ('customer', 'NN'),
 ('care', 'NN'),
 ('angry', 'JJ'),
 ('Brilliant', 'JJ'),
 ('effort', 'NN'),
 ('guys', 'NNS'),
 ('you', 'PRP'),
 ('point', 'VBP'),
 ('one', 'CD'),
 ('finger', 'NN'),
 ('millions', 'NNS'),
 ('are', 'VBP'),
 ('pointed', 'VBN'),
 ('right', 'RB'),
 ('back', 'RB'),
 ('jewishsupremacist', 'VBP'),
 ('words', 'NNS'),
 ('free', 'JJ'),
 ('how', 'WRB'),
 ('use', 'NN'),
 ('that', 'WDT'),
 ('can', 'MD'),
 ('cost', 'VB'),
 ('verbal', 'JJ'),
 ('abuse', 'IN'),
 ('love', 'VBP'),
 ('adult', 'NN'),
 ('teen', 'NN'),
 ('might', 'MD'),
 ('libtard', 'VB'),
 ('libtard', 'JJ'),
 ('sjw', 'NN'),
 ('liberal', 'JJ'),
 ('politics', 'NNS')]
```

Removing additional letters such as ed, 's etc.

In [17]:
```python
ps = PorterStemmer()

tokenized_trainComment = tokenized_trainComment.apply(lambda x: [ps.stem(i) for i in x])

tokenized_trainComment
```

Out[17]:
```
0              [time, eat, with, best, buddi, lunch]
1                [they, want, reflect, money, ksleg]
2     [good, job, but, will, expect, lot, more, futur]
3    [total, dissatisfi, with, the, servic, never, ...
4                                       [love, work]
5                 [worst, custom, care, servic, angri]
6                           [brilliant, effort, guy]
7    [you, point, one, finger, million, are, point,...
8    [word, free, how, use, that, can, cost, you, ...
9      [you, might, libtard, libtard, sjw, liber, polit]
Name: Processed_Comment, dtype: object
```

Replacing old Processed comments

In [18]:
```python
for i in range(len(tokenized_trainComment)):
    tokenized_trainComment[i] = ' '.join(tokenized_trainComment[i])

trainData['Processed_Comment'] = tokenized_trainComment

trainData
```

Out[18]:

	Index	Comment	Polarity	Processed_Comment
0	0	time to eat with my best buddy! #lunch	1	time eat with best buddi lunch
1	1	@user @user if they want reflection money. #ksleg	1	they want reflect money ksleg
2	2	---Good job but I' will expect a lot more in f...	1	good job but will expect lot more futur
3	3	totally dissatisfied with the service###%%@@ n...	0	total dissatisfi with the servic never use thi...
4	4	loved my work!!!!!	1	love work
5	5	Worst customer care service......@@$$$angry	0	worst custom care servic angri
6	6	Brilliant effort guys!!!	1	brilliant effort guy
7	7	@user @user you point one finger @user million...	0	you point one finger million are point right b...
8	8	words r free, it's how u use that can cost you...	1	word free how use that can cost you verbal abu...
9	9	you might be a libtard if... #libtard #sjw #li...	0	you might libtard libtard sjw liber polit

Tokanizing Comments of Test Data

In [19]:
```python
tokenized_testComment = testData['Processed_Comment'].apply(lambda x: x.split())

tokenized_testComment
```

Out[19]:
```
0    [the, pic, says, otherwise, for, young, girls,...
1       [good, night, faith, ever, vaitacacommafiasdv]
2    [when, you, blocked, troll, because, you, prom...
3                               [dinner, with, sister]
4            [who, else, planning, watching, tomorrow]
Name: Processed_Comment, dtype: object
```

POS Tagging

In [20]:
```python
#nltk.download('punkt')
#nltk.download('averaged_perceptron_tagger')
testDataList = testData['Processed_Comment'].tolist()
taggedList = list()
posList = list()

for sentence in testDataList:
        tokenized = nltk.word_tokenize(sentence)
        taggedList.append(nltk.pos_tag(tokenized))

#removing repititions
for tList in taggedList:
    for word_tuple in tList:
        if word_tuple not in posList:
            posList.append(word_tuple)

posList
```

Out[20]:
```
[('the', 'DT'),
 ('pic', 'NN'),
 ('says', 'VBZ'),
 ('otherwise', 'RB'),
 ('for', 'IN'),
 ('young', 'JJ'),
 ('girls', 'NNS'),
 ('confined', 'VBD'),
 ('their', 'PRP$'),
 ('kitchen', 'NN'),
 ('you', 'PRP'),
 ('are', 'VBP'),
 ('void', 'JJ'),
 ('meaning', 'VBG'),
 ('beyond', 'IN'),
 ('cheap', 'JJ'),
 ('publicity', 'NN'),
 ('topoli', 'NN'),
 ('good', 'JJ'),
 ('night', 'NN'),
 ('faith', 'NN'),
 ('ever', 'RB'),
 ('vaitacacommafiasdv', 'VBD'),
 ('when', 'WRB'),
 ('blocked', 'VBD'),
 ('troll', 'NN'),
 ('because', 'IN'),
 ('promise', 'VBP'),
 ('blacklivesmatter', 'JJ'),
 ('amp', 'NN'),
 ('let', 'VB'),
 ('his', 'PRP$'),
 ('nonsensical', 'JJ'),
 ('rants', 'NNS'),
 ('dinner', 'NN'),
 ('with', 'IN'),
 ('sister', 'NN'),
 ('who', 'WP'),
 ('else', 'RB'),
 ('planning', 'VBG'),
 ('watching', 'VBG'),
 ('tomorrow', 'NN')]
```

Removing additional letters such as ed, 's etc.

In [21]:
```python
ps = PorterStemmer()

tokenized_testComment = tokenized_testComment.apply(lambda x : [ps.stem(i) for i in x])

tokenized_testComment
```

Out[21]:
```
0    [the, pic, say, otherwis, for, young, girl, co...
1       [good, night, faith, ever, vaitacacommafiasdv]
2    [when, you, block, troll, becaus, you, promis,...
3                               [dinner, with, sister]
4                    [who, els, plan, watch, tomorrow]
Name: Processed_Comment, dtype: object
```

Replacing old Processed comments

In [22]:
```python
for i in range(len(tokenized_testComment)):
    tokenized_testComment[i] = ' '.join(tokenized_testComment[i])

testData['Processed_Comment'] = tokenized_testComment

testData
```

Out[22]:

	Index	Comment	Polarity	Processed_Comment
0	0	@use the pic says otherwise for young girls co...	0	the pic say otherwis for young girl confin tha...
1	1	#good night! ?? #faith ever #vaitacacommafiasdv	1	good night faith ever vaitacacommafiasdv
2	2	@user when you're blocked by a troll because y...	0	when you block troll becaus you promis blackli...
3	3	dinner with sister!!	1	dinner with sister
4	4	who else is planning on watching @user tomorrow?	1	who els plan watch tomorrow

Feature Extraction from Train Data

Bag of Words

In [23]:
```python
cv= CountVectorizer()

bag_of_words_train = cv.fit_transform(trainData['Processed_Comment']).toarray()

print(cv.vocabulary_)

print(cv.get_feature_names())
print('\n')
print(bag_of_words_train)
```

```
{'time': 49, 'eat': 15, 'with': 55, 'best': 6, 'buddi': 8, 'lunch': 31, 'they': 47, 'want': 53, 'reflect': 40, 'money': 34, 'ksleg': 26, 'good': 21, 'job': 25, 'but': 9, 'will': 54, 'expect': 17, 'lot': 29, 'more': 35, 'futur': 20, 'total': 50, 'dissatisfi': 14, 'the': 46, 'servic': 42, 'never': 36, 'use': 51, 'thi': 48, 'again': 2, 'love': 30, 'work': 57, 'worst': 58, 'custom': 13, 'care': 11, 'angri': 3, 'brilliant': 7, 'effort': 16, 'guy': 22, 'you': 59, 'point': 38, 'one': 37, 'finger': 18, 'million': 33, 'are': 4, 'right': 41, 'back': 5, 'jewishsupremacist': 24, 'word': 56, 'free': 19, 'how': 23, 'that': 45, 'can': 10, 'cost': 12, 'verbal': 52, 'abus': 0, 'adult': 1, 'teen': 44, 'might': 32, 'libtard': 28, 'sjw': 43, 'liber': 27, 'polit': 39}
['abus', 'adult', 'again', 'angri', 'are', 'back', 'best', 'brilliant', 'buddi', 'but', 'can', 'care', 'cost', 'custom', 'dissatisfi', 'eat', 'effort', 'expect', 'finger', 'free', 'futur', 'good', 'guy', 'how', 'jewishsupremacist', 'job', 'ksleg', 'liber', 'libtard', 'lot', 'love', 'lunch', 'might', 'million', 'money', 'more', 'never', 'one', 'point', 'polit', 'reflect', 'right', 'servic', 'sjw', 'teen', 'that', 'the', 'they', 'thi', 'time', 'total', 'use', 'verbal', 'want', 'will', 'with', 'word', 'work', 'worst', 'you']


[[0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 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 1 0 0 0 0 0 0 0 1 0
  0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  1 0 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 0 0 1 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 1 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 1 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 1 0 1 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 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 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 0]
 [0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0
  0 1 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2]
 [1 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0
  0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 1]
 [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 1 2 0 0 0 1 0 0 0
  0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]]
```

TF-IDF

In [24]:
```python
tfidf = TfidfVectorizer()

tfidf_matrix = tfidf.fit_transform(trainData['Processed_Comment'])

trainData_tfidf = pd.DataFrame(tfidf_matrix.todense())

display(trainData_tfidf)
```

	0	1	2	3	4	5	6	7	8	9	...
0	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.418024	0.00000	0.418024	0.000000	...
1	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	...
2	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000	0.00000	0.000000	0.353553	...
3	0.000000	0.000000	0.311046	0.000000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	...
4	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	...
5	0.000000	0.000000	0.000000	0.460158	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	...
6	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000	0.57735	0.000000	0.000000	...
7	0.000000	0.000000	0.000000	0.000000	0.27511	0.27511	0.000000	0.00000	0.000000	0.000000	...
8	0.288694	0.288694	0.000000	0.000000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	...
9	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	...

10 rows × 60 columns

N-Gram

In [25]:
```python
trainDataList = trainData['Processed_Comment'].tolist()
grams = list()
n = 3

for sentence in trainDataList:
    threeGrams = ngrams(sentence.split(), n)
    for gram in threeGrams:
      grams.append(gram)

grams
```

Out[25]:
```
[('time', 'eat', 'with'),
 ('eat', 'with', 'best'),
 ('with', 'best', 'buddi'),
 ('best', 'buddi', 'lunch'),
 ('they', 'want', 'reflect'),
 ('want', 'reflect', 'money'),
 ('reflect', 'money', 'ksleg'),
 ('good', 'job', 'but'),
 ('job', 'but', 'will'),
 ('but', 'will', 'expect'),
 ('will', 'expect', 'lot'),
 ('expect', 'lot', 'more'),
 ('lot', 'more', 'futur'),
 ('total', 'dissatisfi', 'with'),
 ('dissatisfi', 'with', 'the'),
 ('with', 'the', 'servic'),
 ('the', 'servic', 'never'),
 ('servic', 'never', 'use'),
 ('never', 'use', 'thi'),
 ('use', 'thi', 'again'),
 ('worst', 'custom', 'care'),
 ('custom', 'care', 'servic'),
 ('care', 'servic', 'angri'),
 ('brilliant', 'effort', 'guy'),
 ('you', 'point', 'one'),
 ('point', 'one', 'finger'),
 ('one', 'finger', 'million'),
 ('finger', 'million', 'are'),
 ('million', 'are', 'point'),
 ('are', 'point', 'right'),
 ('point', 'right', 'back'),
 ('right', 'back', 'you'),
 ('back', 'you', 'jewishsupremacist'),
 ('word', 'free', 'how'),
 ('free', 'how', 'use'),
 ('how', 'use', 'that'),
 ('use', 'that', 'can'),
 ('that', 'can', 'cost'),
 ('can', 'cost', 'you'),
 ('cost', 'you', 'verbal'),
 ('you', 'verbal', 'abus'),
 ('verbal', 'abus', 'love'),
 ('abus', 'love', 'adult'),
 ('love', 'adult', 'teen'),
 ('you', 'might', 'libtard'),
 ('might', 'libtard', 'libtard'),
 ('libtard', 'libtard', 'sjw'),
 ('libtard', 'sjw', 'liber'),
 ('sjw', 'liber', 'polit')]
```

Feature Extraction for Test Data

Bag of Words

In [26]:
```python
cv= CountVectorizer()

bag_of_words_test = cv.fit_transform(testData['Processed_Comment']).toarray()

print(cv.vocabulary_)

print(cv.get_feature_names())
print('\n')
print(bag_of_words_test)
```

```
{'the': 30, 'pic': 22, 'say': 27, 'otherwis': 21, 'for': 12, 'young': 41, 'girl': 13,
'confin': 7, 'that': 29, 'kitchen': 16, 'you': 40, 'are': 1, 'void': 35, 'mean': 18,
'beyond': 3, 'cheap': 6, 'public': 25, 'topoli': 32, 'good': 14, 'night': 19, 'faith':
11, 'ever': 10, 'vaitacacommafiasdv': 34, 'when': 37, 'block': 5, 'troll': 33, 'becau
s': 2, 'promis': 24, 'blacklivesmatt': 4, 'amp': 0, 'let': 17, 'hi': 15, 'nonsens': 2
0, 'rant': 26, 'dinner': 8, 'with': 39, 'sister': 28, 'who': 38, 'els': 9, 'plan': 23,
'watch': 36, 'tomorrow': 31}
['amp', 'are', 'becaus', 'beyond', 'blacklivesmatt', 'block', 'cheap', 'confin', 'dinn
er', 'els', 'ever', 'faith', 'for', 'girl', 'good', 'hi', 'kitchen', 'let', 'mean', 'n
ight', 'nonsens', 'otherwis', 'pic', 'plan', 'promis', 'public', 'rant', 'say', 'siste
r', 'that', 'the', 'tomorrow', 'topoli', 'troll', 'vaitacacommafiasdv', 'void', 'watc
h', 'when', 'who', 'with', 'you', 'young']


[[0 1 0 1 0 0 1 1 0 0 0 0 1 1 0 0 1 0 1 0 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1
  0 0 0 0 1 1]
 [0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
  0 0 0 0 0 0]
 [1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0
  0 1 0 0 2 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
  0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
  1 0 1 0 0 0]]
```

TF-IDF

In [27]:
```python
tfidf = TfidfVectorizer()

tfidf_matrix = tfidf.fit_transform(testData['Processed_Comment'])

testData_tfidf = pd.DataFrame(tfidf_matrix.todense())

display(testData_tfidf)
```

	0	1	2	3	4	5	6	7	8	9	...
0	0.000000	0.238022	0.000000	0.238022	0.000000	0.000000	0.238022	0.238022	0.00000	0.000000	...
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	...
2	0.271127	0.000000	0.271127	0.000000	0.271127	0.271127	0.000000	0.000000	0.00000	0.000000	...
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.57735	0.000000	...
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.447214	...

5 rows × 42 columns

N-Gram

In [28]:
```python
testDataList = testData['Processed_Comment'].tolist()
grams = list()
n = 3

for sentence in testDataList:
    threeGrams = ngrams(sentence.split(), n)
    for gram in threeGrams:
      grams.append(gram)

grams
```

Out[28]:
```
[('the', 'pic', 'say'),
 ('pic', 'say', 'otherwis'),
 ('say', 'otherwis', 'for'),
 ('otherwis', 'for', 'young'),
 ('for', 'young', 'girl'),
 ('young', 'girl', 'confin'),
 ('girl', 'confin', 'that'),
 ('confin', 'that', 'kitchen'),
 ('that', 'kitchen', 'you'),
 ('kitchen', 'you', 'are'),
 ('you', 'are', 'void'),
 ('are', 'void', 'mean'),
 ('void', 'mean', 'beyond'),
 ('mean', 'beyond', 'cheap'),
 ('beyond', 'cheap', 'public'),
 ('cheap', 'public', 'topoli'),
 ('good', 'night', 'faith'),
 ('night', 'faith', 'ever'),
 ('faith', 'ever', 'vaitacacommafiasdv'),
 ('when', 'you', 'block'),
 ('you', 'block', 'troll'),
 ('block', 'troll', 'becaus'),
 ('troll', 'becaus', 'you'),
 ('becaus', 'you', 'promis'),
 ('you', 'promis', 'blacklivesmatt'),
 ('promis', 'blacklivesmatt', 'amp'),
 ('blacklivesmatt', 'amp', 'let'),
 ('amp', 'let', 'hi'),
 ('let', 'hi', 'nonsens'),
 ('hi', 'nonsens', 'rant'),
 ('dinner', 'with', 'sister'),
 ('who', 'els', 'plan'),
 ('els', 'plan', 'watch'),
 ('plan', 'watch', 'tomorrow')]
```

Decision Tree Machine Learning

In [29]:
```python
# First Combine Both train and Test examples so we can have same number of lables
```

In [30]:
```python
a = trainData

b = testData

allData = pd.concat([a , b])

allData = allData.drop(['Index'], axis=1)

allData
```

Out[30]:

	Comment	Polarity	Processed_Comment
0	time to eat with my best buddy! #lunch	1	time eat with best buddi lunch
1	@user @user if they want reflection money. #ksleg	1	they want reflect money ksleg
2	---Good job but I' will expect a lot more in f...	1	good job but will expect lot more futur
3	totally dissatisfied with the service###%%@@ n...	0	total dissatisfi with the servic never use thi...
4	loved my work!!!!!	1	love work
5	Worst customer care service......@@$$$angry	0	worst custom care servic angri
6	Brilliant effort guys!!!	1	brilliant effort guy
7	@user @user you point one finger @user million...	0	you point one finger million are point right b...
8	words r free, it's how u use that can cost you...	1	word free how use that can cost you verbal abu...
9	you might be a libtard if... #libtard #sjw #li...	0	you might libtard libtard sjw liber polit
0	@use the pic says otherwise for young girls co...	0	the pic say otherwis for young girl confin tha...
1	#good night! ?? #faith ever #vaitacacommafiasdv	1	good night faith ever vaitacacommafiasdv
2	@user when you're blocked by a troll because y...	0	when you block troll becaus you promis blackli...
3	dinner with sister!!	1	dinner with sister
4	who else is planning on watching @user tomorrow?	1	who els plan watch tomorrow

Decision Tree with Bag of Words

In [31]:
```python
cv= CountVectorizer()

bag_of_words_all_data = cv.fit_transform(allData['Processed_Comment']).toarray()

print(cv.vocabulary_)

print(cv.get_feature_names())
print('\n')
print(bag_of_words_all_data)
```

```
{'time': 76, 'eat': 23, 'with': 90, 'best': 8, 'buddi': 13, 'lunch': 47, 'they': 74,
'want': 85, 'reflect': 65, 'money': 51, 'ksleg': 41, 'good': 34, 'job': 39, 'but': 14,
'will': 89, 'expect': 27, 'lot': 45, 'more': 52, 'futur': 32, 'total': 79, 'dissatisf
```

```
i': 22, 'the': 73, 'servic': 68, 'never': 53, 'use': 81, 'thi': 75, 'again': 2, 'lov
e': 46, 'work': 92, 'worst': 93, 'custom': 20, 'care': 16, 'angri': 4, 'brilliant': 1
2, 'effort': 24, 'guy': 35, 'you': 94, 'point': 60, 'one': 56, 'finger': 29, 'millio
n': 50, 'are': 5, 'right': 66, 'back': 6, 'jewishsupremacist': 38, 'word': 91, 'free':
31, 'how': 37, 'that': 72, 'can': 15, 'cost': 19, 'verbal': 83, 'abus': 0, 'adult': 1,
'teen': 71, 'might': 49, 'libtard': 44, 'sjw': 70, 'liber': 43, 'polit': 61, 'pic': 5
8, 'say': 67, 'otherwis': 57, 'for': 30, 'young': 95, 'girl': 33, 'confin': 18, 'kitch
en': 40, 'void': 84, 'mean': 48, 'beyond': 9, 'cheap': 17, 'public': 63, 'topoli': 78,
'night': 54, 'faith': 28, 'ever': 26, 'vaitacacommafiasdv': 82, 'when': 87, 'block': 1
1, 'troll': 80, 'becaus': 7, 'promis': 62, 'blacklivesmatt': 10, 'amp': 3, 'let': 42,
'hi': 36, 'nonsens': 55, 'rant': 64, 'dinner': 21, 'sister': 69, 'who': 88, 'els': 25,
'plan': 59, 'watch': 86, 'tomorrow': 77}
['abus', 'adult', 'again', 'amp', 'angri', 'are', 'back', 'becaus', 'best', 'beyond',
'blacklivesmatt', 'block', 'brilliant', 'buddi', 'but', 'can', 'care', 'cheap', 'confi
n', 'cost', 'custom', 'dinner', 'dissatisfi', 'eat', 'effort', 'els', 'ever', 'expec
t', 'faith', 'finger', 'for', 'free', 'futur', 'girl', 'good', 'guy', 'hi', 'how', 'je
wishsupremacist', 'job', 'kitchen', 'ksleg', 'let', 'liber', 'libtard', 'lot', 'love',
'lunch', 'mean', 'might', 'million', 'money', 'more', 'never', 'night', 'nonsens', 'on
e', 'otherwis', 'pic', 'plan', 'point', 'polit', 'promis', 'public', 'rant', 'reflec
t', 'right', 'say', 'servic', 'sister', 'sjw', 'teen', 'that', 'the', 'they', 'thi',
'time', 'tomorrow', 'topoli', 'total', 'troll', 'use', 'vaitacacommafiasdv', 'verbal',
'void', 'want', 'watch', 'when', 'who', 'will', 'with', 'word', 'work', 'worst', 'yo
u', 'young']


[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 2 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

In [32]:

```python
# Already have frature extraction for both test and train data in bag of words
```

In [33]:

```python
X_train = bag_of_words_all_data[:10]
Y_train = allData['Polarity'][0:10]

X_test = bag_of_words_all_data[10:]
Y_test = allData['Polarity'][10:]
```

In [34]:

```python
decisionTree_bag_of_words = DecisionTreeClassifier(criterion = "entropy" , random_stat
decisionTree_bag_of_words.fit(X_train , Y_train)
Y_pred = decisionTree_bag_of_words.predict(X_test)
print('Predicted values: ', end=' ')
print(Y_pred)

print('Accuracy:' , end=' ')
print(accuracy_score(Y_test ,Y_pred))
```

```
Predicted values:  [0 1 0 1 1]
Accuracy: 1.0
```

Decision Tree through TF-IDF

In [35]:

```python
tfidf = TfidfVectorizer()

tfidf_matrix = tfidf.fit_transform(allData['Processed_Comment'])

allData_tfidf = pd.DataFrame(tfidf_matrix.todense())

display(allData_tfidf)
```

	0	1	2	3	4	5	6	7	8	9	...
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.422559	0.000000	...
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
3	0.000000	0.000000	0.314278	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
5	0.000000	0.000000	0.000000	0.000000	0.458638	0.000000	0.000000	0.000000	0.000000	0.000000	...
6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
7	0.000000	0.000000	0.000000	0.000000	0.000000	0.246500	0.283877	0.000000	0.000000	0.000000	...
8	0.292656	0.292656	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
9	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
10	0.000000	0.000000	0.000000	0.000000	0.000000	0.212639	0.000000	0.000000	0.000000	0.244882	...
11	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
12	0.000000	0.000000	0.000000	0.281105	0.000000	0.000000	0.000000	0.281105	0.000000	0.000000	...
13	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
14	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...

15 rows × 96 columns

In [36]:

```python
X_train = allData_tfidf[:10]
Y_train = allData['Polarity'][0:10]

X_test = allData_tfidf[10:]
Y_test = allData['Polarity'][10:]
```

In [37]:

```python
decisionTree_tfidf = DecisionTreeClassifier(criterion = "entropy" , random_state = 10
decisionTree_tfidf.fit(X_train , Y_train)
Y_pred = decisionTree_tfidf.predict(X_test)
print('Predicted values: ', end=' ')
print(Y_pred)

print('Accuracy:' , end=' ')
print(accuracy_score(Y_test ,Y_pred))
```

```
Predicted values:  [1 1 0 1 1]
Accuracy: 0.8
```

Application Phase

Full Decision Tree Modal

In [38]:

```python
full_decisionTree = DecisionTreeClassifier(criterion = "entropy" , random_state = 100
full_decisionTree.fit(bag_of_words_all_data[:] , allData['Polarity'][:])
```

Out[38]:

```
DecisionTreeClassifier(criterion='entropy', random_state=100)
```

Saving in Pickle File

In [39]:

```python
filename = 'finalized_model.sav'
pickle.dump(full_decisionTree, open(filename, 'wb'))
```

Loading Pickle Model

In [40]:

```python
loaded_model = pickle.load(open(filename, 'rb'))
```

Input from User

In [41]:

```python
userTweet = input('Enter A Tweet: ')
```

```
Enter A Tweet: they want to kill this sad is it not @someone #eating
```

In [42]:

```python
userTweet = remove_pattern(userTweet , "@[\w]*")

r = re.findall("[^a-zA-Z]" , userTweet)
text = ''
for i in r:
    text = re.sub(i," ",userTweet)
userTweet = text

userTweet = userTweet.lower()

userTweet = [x for x in userTweet.split() if len(x) >= 3]
```

In [43]:

```python
ps = PorterStemmer()

for x in range(len(userTweet)):
    userTweet[x] = ps.stem(userTweet[x])

userTweet
```

Out[43]:

```
['they', 'want', 'kill', 'thi', 'sad', 'not', 'eat']
```

In [44]:

```python
# All of Our Feature names are in cv.get_feature_names()

lables = cv.get_feature_names()

userTweetArr = []

for x in lables:
    total = 0
    for y in userTweet:
        if x == y:
            total = total + 1
    userTweetArr.append(total)

predictArr = [userTweetArr]
```

In [45]:

```python
predict = loaded_model.predict(predictArr)
```

In [46]:

```python
if predict == 1:
    print('The Tweet Polarity is Happy')
else:
    print('The Tweet Polarity is Sad')
```

```
The Tweet Polarity is Happy
```

In [ ]: