

```
import pandas as pd
data = pd.read_csv("https://raw.githubusercontent.com/Dataminingproject/dataminingassignme
names=["key", "tweet", "status"])
```

data

	key	tweet	status
0	611857364396965889	@aandraous @britishmuseum @AndrewsAntonio Merc...	nocode
1	614484565059596288	Dorian Gray with Rainbow Scarf #LoveWins (from...	happy
2	614746522043973632	@SelectShowcase @Tate_Stlves ... Replace with ...	happy
3	614877582664835073	@Sofabsports thank you for following me back. ...	happy
4	611932373039644672	@britishmuseum @TudorHistory What a beautiful ...	happy
...
3080	613678555935973376	MT @AliHaggett: Looking forward to our public ...	happy
3081	613294681225621504	@britishmuseum Upper arm guard?	nocode
3082	615246897670922240	@MrStuchbery @britishmuseum Mesmerising.	happy
3083	613016084371914753	@NationalGallery The 2nd GENOCIDE against #Bia...	not-relevant
3084	611566876762640384	@britishmuseum Experience #battlewaterloo from...	nocode

```
data=data.iloc[:,1:]
```

data

	tweet	status
0	@aandraous @britishmuseum @AndrewsAntonio Merc...	nocode
1	Dorian Gray with Rainbow Scarf #LoveWins (from...	happy
2	@SelectShowcase @Tate_Stlves ... Replace with ...	happy
3	@Sofabsports thank you for following me back. ...	happy
4	@britishmuseum @TudorHistory What a beautiful ...	happy
...
3080	MT @AliHaggett: Looking forward to our public ...	happy
3081	@britishmuseum Upper arm guard?	nocode
3082	@MrStuchbery @britishmuseum Mesmerising.	happy
3083	@NationalGallery The 2nd GENOCIDE against #Bia...	not-relevant
3084	@britishmuseum Experience #battlewaterloo from...	nocode

3085 rows × 2 columns

▼ New Section

DATA ANALYSIS

Counting the no. of labels

```
data.status.value_counts()

nocode          1572
happy           1137
not-relevant    214
angry           57
surprise        35
sad             32
happy|surprise  11
happy|sad       9
disgust|angry   7
disgust         6
sad|angry       2
sad|disgust     2
sad|disgust|angry 1
Name: status, dtype: int64
```

Removing null values such as "nocode" and "not-relevant"

```
data.drop(data[data['status']=='nocode'].index,inplace=True)
data.drop(data[data['status']=='not-relevant'].index,inplace=True)
data
```

tweet status

```
data.status.value_counts()
```

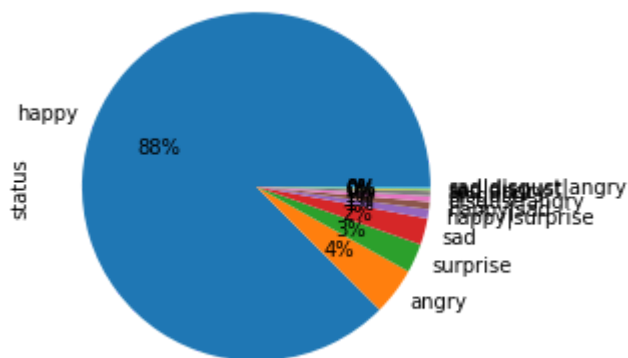
```
happy          1137
angry           57
surprise        35
sad             32
happy|surprise  11
happy|sad        9
disgust|angry    7
disgust          6
sad|angry         2
sad|disgust       2
sad|disgust|angry 1
Name: status, dtype: int64
```

```
3079 @britishmuseum. Thanks for ranking us #1 in @... happy
```

```
import matplotlib.pyplot as plt
```

```
data.status.value_counts().plot(kind='pie', autopct='%1.0f%%')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8e8ab19f50>
```




```
import matplotlib.pyplot as plt
```

```
data.status.value_counts().plot(kind='bar')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f8e86223190>



importing libraries for preprocessing **bold text**

```
|  |
```

```
import numpy as np
import pandas as pd
import re
import nltk
import matplotlib.pyplot as plt
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('stopwords')
stop_words=set(stopwords.words('english'))

[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 stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

processing fxn **bold text**

```
def twitter_process(tweet):
    tweet = tweet.lower()
    #for removing numbers
    tweet = re.sub(r'\d+', '', tweet)
    #for removing urls
    tweet = re.sub(r"http\S+|https\S+", "", tweet, flags = re.MULTILINE)
    #maintain the punctuation
    tweet = tweet.translate(str.maketrans(" ", " ", string.punctuation))
    #for removing special symbols
    tweet = re.sub(r'\@\w|\#', "", tweet)
    tweet_tokens = word_tokenize(tweet)
    filtered_words = [word for word in tweet_tokens if word not in stop_words and len(word)>
    lemmatizer = WordNetLemmatizer()
    Output = [lemmatizer.lemmatize(w, pos = 'a') for w in filtered_words]
    return " ".join(Output)

data["tweet"] = data.tweet.apply(twitter_process)
data
```

	tweet	status
1	dorian gray rainbow scarf lovewins britishmuseum	happy
2	selectshowcase tatestives replace wish artist ...	happy
3	sofabsports thank following back great hear di...	happy
4	britishmuseum tudorhistory beautiful jewel por...	happy
5	nationalgallery thepoldarkian always loved pai...	happy
...
3076	good see liveatlicas art collection leedsartga...	happy
3077	rammuseum thanks well look next week friday done	happy
3079	britishmuseum thanks ranking tripadvisor thing...	happy
3080	alibeggett looking forward public engagement a	happy

**USING BOW extraction **

```
tweets=data['tweet'].tolist()
from sklearn.feature_extraction.text import CountVectorizer
BOW_Vector=CountVectorizer(tweets)
print(BOW_Vector)

CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8',
                input=['dorian gray rainbow scarf lovewins britishmuseum',
                    'selectshowcase tatestives replace wish artist uses '
                    'next installation entralling',
                    'sofabsports thank following back great hear diverse '
                    'amp interesting panel defeatingdepression rammuseum',
                    'britishmuseu...',
                    'castlehill kettlesyard explorearchives',
                    'week writing reports reports suddenly brightened '
                    'thought definingbeauty britishmuseum tomorrow '
                    'littlemissmoo',
                    'samba time britishmuseum camdentalking discoveryday', ...],
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)
```

```
BOW_Vector.fit(tweets)
BOW_Vector.get_feature_names()
#we will get bag of word
```

```
['aaroo',
 'abducted',
 'able',
 'aboard',
 'aboriginal',
 'aboutlondon',
 'abroadbrush',
 'absolutely',
 'absorbing',
```

```

'abt',
'abu',
'accessibility',
'accessible',
'acclimatize',
'accounts',
'ace',
'acenational',
'acenterprises',
'achieve',
'achieved',
'achievement',
'acquire',
'acquired',
'acquires',
'acropolismuseum',
'act',
'action',
'activities',
'activity',
'actually',
'adamkosczyk',
'add',
'adding',
'addressed',
'adelegeras',
'adjournment',
'admire',
'admiring',
'admission',
'adorons',
'adrianmurdoch',
'adventureswithtuck',
'advocating',
'aesthetic',
'aestheticmag',
'affogato',
'afneil',
'africa',
'african',
'africanrockart',
'aft',
'afternoon',
'afterwards',
'age',
'agefriendlymuseums',
'ageofcreativity',
'ages',
'ageuk',
'ago',

```

```

array=BOW_Vector.transform(tweets).toarray()
array

```

```

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]]
```

TF-IDF vectorization

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
vectorizer = TfidfVectorizer (max_features=3000, min_df=6, max_df=0.75, stop_words=stopwor
processed_features=vectorizer.fit_transform(tweets).toarray()
processed_features
```

```
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.]])
```

*ENCODING OF Target *

```
##make label likehappy|surprise as happy
data.status.value_counts()
```

```
happy          1137
angry           57
surprise        35
sad             32
happy|surprise  11
happy|sad        9
disgust|angry    7
disgust          6
sad|angry        2
sad|disgust      2
sad|disgust|angry 1
Name: status, dtype: int64
```

```
data=data.replace('happy|surprise','happy')
data=data.replace('happy|sad','happy')
data=data.replace('sad|disgust','sad')
data=data.replace('sad|disgust|angry','sad')
data=data.replace('sad|angry','sad')
data=data.replace('disgust|angry','sad')
data.status.value_counts()
status=data.iloc[:,1].values
status
```

```
array(['happy', 'happy', 'happy', ..., 'happy', 'happy', 'happy'],
      dtype=object)
```

```
data.status.value_counts()
```

```

happy      1157
angry      57
sad        44
surprise   35
disgust     6
Name: status, dtype: int64

```

TEST AND TRAIN DATA

```

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(processed_features, status, test_size=
X_train

```

```

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.          , 0.44888323, ..., 0.          , 0.          ,
        0.          ],
       [0.          , 0.          , 0.          , ..., 0.          , 0.          ,
        0.          ]])

```

APPLY ALGORITHM

Decision Tree info gain

```

from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
dt=DecisionTreeClassifier(criterion='entropy')
dt.fit(X_train,y_train)
predictions=dt.predict(X_test)
predictions

```

```

array(['happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'sad',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'sad', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'angry', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'angry', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'angry', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',

```



```
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'surprise',
'angry', 'happy', 'happy', 'angry', 'happy', 'happy', 'happy',
'angry', 'happy', 'happy', 'happy', 'happy', 'happy', 'disgust',
'happy', 'sad', 'happy', 'happy', 'happy', 'happy', 'happy',
'angry', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'happy', 'happy', 'angry', 'angry', 'happy', 'happy', 'happy',
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'happy', 'happy', 'happy', 'happy', 'happy', 'sad', 'happy',
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'happy', 'angry', 'angry', 'happy', 'happy', 'happy', 'happy',
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'happy', 'happy', 'angry', 'happy', 'happy', 'happy', 'sad',
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'happy', 'happy', 'angry', 'happy', 'happy', 'happy', 'happy',
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'angry', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'sad',
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
'happy'], dtype=object)
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

```
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

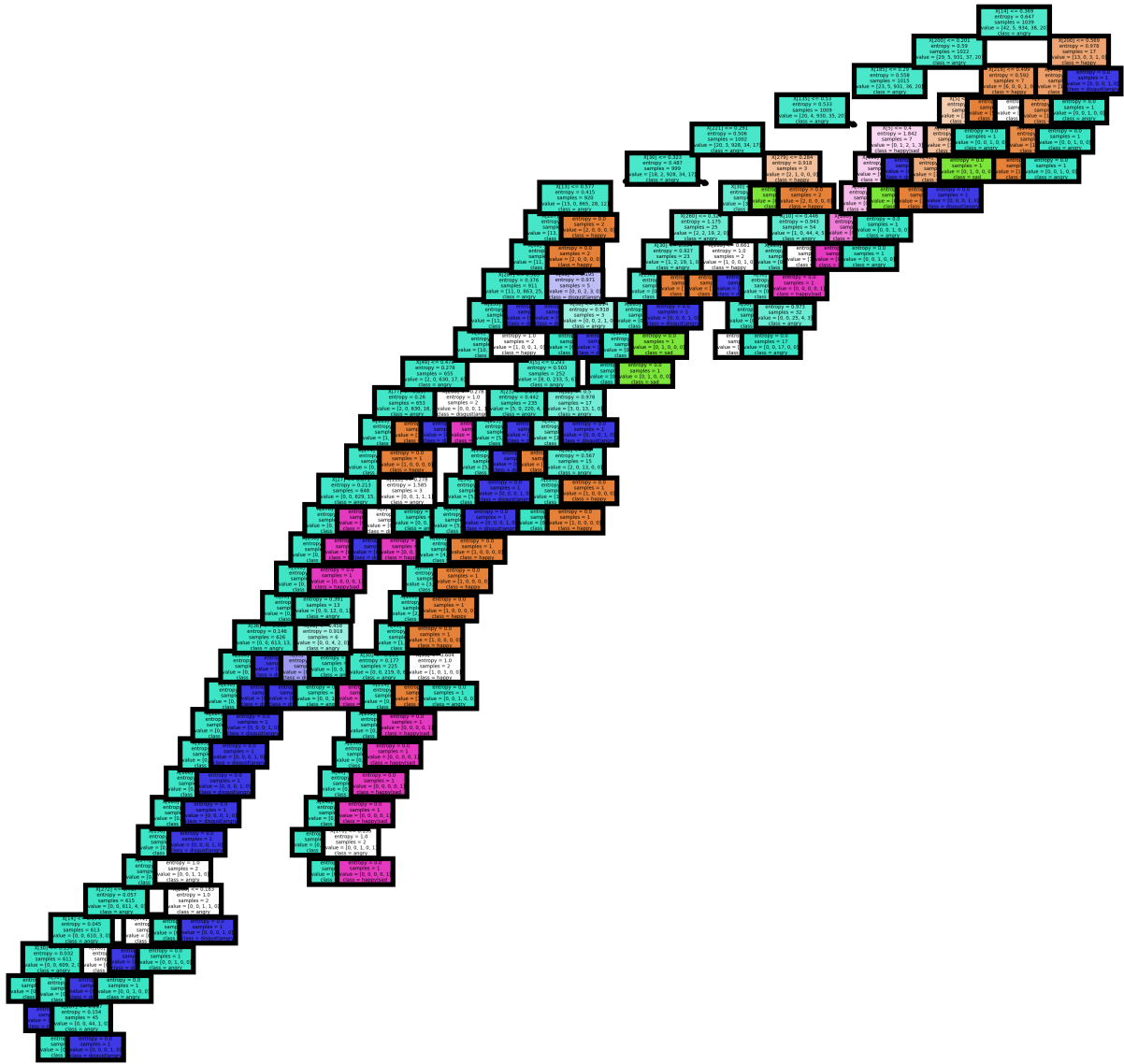
```
[[ 7  0  7  1  0]
 [ 0  0  1  0  0]
 [ 7  0 210  5  1]
 [ 0  0  6  0  0]
 [ 0  1 14  0  0]]
      precision    recall  f1-score   support

   angry           0.50      0.47      0.48         15
  disgust           0.00      0.00      0.00          1
    happy           0.88      0.94      0.91        223
     sad           0.00      0.00      0.00          6
  surprise           0.00      0.00      0.00         15

 accuracy                   0.83         260
 macro avg           0.28      0.28      0.28         260
 weighted avg           0.79      0.83      0.81         260
```

```
0.8346153846153846
```

```
fig,axes=plt.subplots(nrows=1,ncols=1,figsize=(4,4),dpi=1000)
dtree=tree.plot_tree(dt,class_names=['happy','sad','angry','disgust|angry','happy|sad','su
```



```

from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
dt1=DecisionTreeClassifier(criterion='gini',max_depth=5)
dt1.fit(X_train,y_train)
predictions=dt1.predict(X_test)
predictions

```

[illegible]

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

```
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

```
[[ 6  0  9  0  0]
 [ 0  0  1  0  0]
 [ 0  0 222  1  0]
 [ 0  0  6  0  0]
 [ 0  0 15  0  0]]
```

	precision	recall	f1-score	support
angry	1.00	0.40	0.57	15
disgust	0.00	0.00	0.00	1
happy	0.88	1.00	0.93	223
sad	0.00	0.00	0.00	6
surprise	0.00	0.00	0.00	15

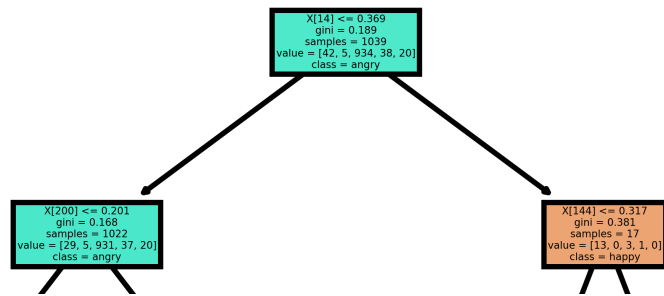
accuracy			0.88	260
macro avg	0.38	0.28	0.30	260
weighted avg	0.81	0.88	0.83	260

0.8769230769230769

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedWarning:  $\text{warn\_prf}(\text{average}, \text{modifier}, \text{msg\_start}, \text{len}(\text{result}))$ 
```

```
fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(4, 4), dpi=1000)
```

```
dtree = tree.plot_tree(dt1, class_names=['happy', 'sad', 'angry', 'disgust|angry', 'happy|sad', 's
```



NAIVE BAYES

```

from sklearn.naive_bayes import GaussianNB
nv = GaussianNB() # create a classifier
nv.fit(X_train,y_train) # fitting the data
#Import metrics class from sklearn
from sklearn.metrics import accuracy_score
y_pred = nv.predict(X_test) # store the prediction data
y_pred

array(['happy', 'sad', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'surprise', 'happy', 'disgust', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'disgust', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'sad',
       'happy', 'happy', 'happy', 'happy', 'disgust', 'disgust', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'angry', 'happy', 'happy', 'disgust',
       'disgust', 'sad', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'angry', 'disgust', 'happy',
       'angry', 'happy', 'happy', 'happy', 'sad', 'disgust', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'disgust', 'happy', 'angry', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'angry', 'happy', 'happy', 'sad',
       'happy', 'happy', 'happy', 'happy', 'angry', 'happy', 'disgust',
       'sad', 'happy', 'sad', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'sad', 'happy',
       'happy', 'happy', 'happy', 'happy', 'surprise', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'sad', 'happy', 'disgust', 'happy', 'happy', 'happy', 'happy',
       'happy', 'sad', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'angry', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'disgust', 'happy', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'surprise', 'happy', 'happy', 'happy', 'angry', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'angry', 'happy', 'disgust',
       'happy', 'happy', 'happy', 'sad', 'happy', 'happy', 'happy',
       'happy', 'happy', 'happy', 'happy', 'happy', 'happy', 'happy',
       'angry', 'happy', 'happy', 'happy', 'happy', 'angry', 'happy',

```

```
'happy', 'happy', 'angry', 'happy', 'happy', 'disgust', 'happy'],
dtype='<U8')

```

```
accuracy_score(y_test,y_pred) # calculate the accuracy

```

```
0.8153846153846154

```

```
predictions=nv.predict(X_test)

```

```
predictions

```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

```

```
print(confusion_matrix(y_test,predictions))

```

```
print(classification_report(y_test,predictions))

```

```
print(accuracy_score(y_test, predictions))

```

```
[[ 7  1  6  1  0]
 [ 0  1  0  0  0]
 [ 5  7 20 1  9  1]
 [ 0  1  4  1  0]
 [ 0  4  9  0  2]]

```

	precision	recall	f1-score	support
angry	0.58	0.47	0.52	15
disgust	0.07	1.00	0.13	1
happy	0.91	0.90	0.91	223
sad	0.09	0.17	0.12	6
surprise	0.67	0.13	0.22	15
accuracy			0.82	260
macro avg	0.47	0.53	0.38	260
weighted avg	0.86	0.82	0.82	260

```
0.8153846153846154

```

KNN

```
from sklearn.neighbors import KNeighborsClassifier

```

```
knn = KNeighborsClassifier(n_neighbors=20)

```

```
knn.fit(X_train, y_train)

```

```
# Predict on dataset which model has not seen before

```

```
print(knn.predict(X_test))

```

```
['happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy'
 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy'
 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy'
 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy'
 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy'
 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy'
 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy'
 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy'
 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy'
 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy'
 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy'
 'angry' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy' 'happy']

```



```
[ [ 8   0   7   0   0 ]
[  0   0   1   0   0 ]
[  0   0 223   0   0 ]
[  0   0   6   0   0 ]
[  0   0  15   0   0 ]]
```

	precision	recall	f1-score	support
angry	1.00	0.53	0.70	15
disgust	0.00	0.00	0.00	1
happy	0.88	1.00	0.94	223
sad	0.00	0.00	0.00	6
surprise	0.00	0.00	0.00	15
accuracy			0.89	260
macro avg	0.38	0.31	0.33	260
weighted avg	0.82	0.89	0.85	260

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedWarning: warn prf(average, modifier, msg start, len(result))
```

[illegible]

[illegible]

```
print(classification_report(y_test,grd_boost.predict(X_test)))
```

	precision	recall	f1-score	support
angry	0.67	0.27	0.38	15
disgust	0.00	0.00	0.00	1
happy	0.87	0.96	0.92	223
sad	0.00	0.00	0.00	6
surprise	0.00	0.00	0.00	15
accuracy			0.84	260
macro avg	0.31	0.25	0.26	260
weighted avg	0.79	0.84	0.81	260

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undet
    warn_prf(average, modifier, msg_start, len(result))
```

```
grd boost.score(X test, y test)
```

0.8423076923076923

Accuracy Comparasion

```

algos = [
    #Ensemble Methods
    GradientBoostingClassifier(),
    RandomForestClassifier(),

    #Navies Bayes
    GaussianNB(),

    #Nearest Neighbor
    KNeighborsClassifier(),

    #Trees
    DecisionTreeClassifier(criterion='gini',max_depth=5),
    DecisionTreeClassifier(criterion='entropy')

    #tree.ExtraTreeClassifier(),

]

GradientBoostingClassifier()

    GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                               learning_rate=0.1, loss='deviance', max_depth=3,
                               max_features=None, max_leaf_nodes=None,
                               min_impurity_decrease=0.0, min_impurity_split=None,
                               min_samples_leaf=1, min_samples_split=2,
                               min_weight_fraction_leaf=0.0, n_estimators=100,
                               n_iter_no_change=None, presort='deprecated',
                               random_state=None, subsample=1.0, tol=0.0001,
                               validation_fraction=0.1, verbose=0,
                               warm_start=False)

import pandas as pd
algos_col=[]
algos_compare = pd.DataFrame(columns = algos_col)

index = 0
for alg in algos:

    predicted = alg.fit(X_train, y_train).predict(X_test)
    algo_name = alg.__class__.__name__
    algos_compare.loc[index, ' Name'] = algo_name
    algos_compare.loc[index, 'Accuracy'] = round(alg.score(X_test, y_test), 3)
    index+=1

```

```
algos_compare.sort_values(by = ['Accuracy'], ascending = False, inplace = True)
algos_compare
```

	Name	Accuracy
3	KNeighborsClassifier	0.881
1	RandomForestClassifier	0.877
4	DecisionTreeClassifier	0.869
0	GradientBoostingClassifier	0.858
5	DecisionTreeClassifier	0.846
2	GaussianNB	0.815

```
import seaborn as sns
```

```
algos_compare.plot(kind='bar')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f8e6821f5d0>

