```
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.        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.        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.        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
```

Emotion Recognition as Text Classification

alt text Learning to Classifying text (source: https://www.nltk.org/book/ch06.html

```
from sklearn import svm

Y_training = data_training.label
clf = svm.SVC(kernel='linear')
print("Training Classifier with BoW representation")
clf.fit(X_BoW, Y_training)
```

```
Training Classifier with BoW representation
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
cols = ['id','text','label','intensity']

anger_test = pd.read_csv('anger_test.txt', header=None, sep="\t", names= cols, index_col=0)
fear_test = pd.read_csv('fear_test.txt', header=None, sep="\t", names= cols, index_col=0)
sad_test = pd.read_csv('sadness_test.txt', header=None, sep="\t", names= cols, index_col=0)
joy_test = pd.read_csv('joy_test.txt', header=None, sep="\t", names= cols, index_col=0)

frames_test = [anger_test, fear_test, sad_test, joy_test]
```

```
data_test = pd.concat(frames_test)
data_test.reset_index(inplace=True)
print (data_test.head(20))
data_test.label.value_counts()
```

```
          id                                                text  label  intensity
    0   10941  At the point today where if someone says somet...  anger      0.319
    1   10942  @CorningFootball  IT'S GAME DAY!!!!      T MIN...  anger      0.144
    2   10943  This game has pissed me off more than any othe...  anger      0.898
    3   10944  @spamvicious I've just found out it's Candice ...  anger      0.271
    4   10945  @moocowward @mrsajhargreaves @Melly77 @GaryBar...  anger      0.646
    5   10946  @moocowward @mrsajhargreaves @Melly77 @GaryBar...  anger      0.583
    6   10947  wanna go home and focus up on this game . Don'...  anger      0.375
    7   10948  @virginmedia I've been disconnected whilst on ...  anger      0.625
    8   10949  @virginmedia I've been disconnected whilst on ...  anger      0.396
    9   10950  I wanna see you smile I don't wanna see you ma...  anger      0.250
    10  10951       @shae_caitlin ur road rage gives me anxiety.  anger      0.438
    11  10952                    @eMilsOnWheels I'm furious 😫😫😫  anger      0.708
    12  10953  @EtherealMystic_ She was winning this war that...  anger      0.333
    13  10954  They gonna give this KKk police bitch the mini...  anger      0.877
    14  10955  They gonna give this KKk police bitch the mini...  anger      0.708
    15  10956                   I just got murdered in madden. 😤  anger      0.417
    16  10957  My nephew sees that i have a frown on my face ...  anger      0.229
    17  10958  @CUTEFUNNYANIMAL @luvcaps19 My sister's dog do...  anger      0.375
    18  10959  new madden 16 video was gonna be up but xbox i...  anger      0.667
    19  10960  The animosity toward Hillary comes from conspi...  anger      0.354
    fear       995
    anger      760
    joy        714
    sadness    673
    Name: label, dtype: int64
```

```
clean_data_test_list= tweet_cleaner(data_test)
data_test.text = pd.DataFrame(clean_data_test_list)
data_test.head(10)
```

```
Cleaning and parsing the tweets...

Done!
```

	id	text	label	intensity
1	10942	game day !!!! minus : # relentless	anger	0.144

```
X_BoW_test = vectorizer.transform(data_test.text)
Y_test = data_test.label
```

Evaluation

source: https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226 alt text

Accuracy

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN)}$$

Accuracy is a valid choice of evaluation for classification problems which are well balanced and not skewed or No class imbalance.

Precision

$$Precision = \frac{TP}{TP+FP}$$

Precision answers the following question: what proportion of predicted Positives is truly Positive?

Recall

$$Recall = \frac{TP}{TP+FN}$$

Recall answers the following question: what proportion of actual Positives is correctly classified?

F1-Score

$$F1 = 2 * \frac{precision*recall}{precision+recall}$$

F1 score sort of maintains a balance between the precision and recall for your classifier.

If you are a police inspector and you want to catch criminals, you want to be sure that the person you catch is a criminal (Precision) and you also want to capture as many criminals (Recall) as possible. The F1 score manages this tradeoff

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

prediction = clf.predict(X_BoW_test)
accuracy = accuracy_score(Y_test, prediction)

# 'micro':Calculate metrics globally by counting the total true positives, false negatives an
precision = precision_score(Y_test, prediction, average="micro")
recall = recall_score(Y_test, prediction, average="micro")
```

```
F1 = f1_score(Y_test, prediction, average="micro")
print ('Accuracy:', accuracy)
print ('Precision:', precision)
print ('Recall:', recall)
print ('F1:', F1)
```

```
Accuracy: 0.7348822406110758
Precision: 0.7348822406110758
Recall: 0.7348822406110758
F1: 0.7348822406110759
```

```
#'macro':Calculate metrics for each label, and find their unweighted mean. This does not take
precision = precision_score(Y_test, prediction, average="macro")
recall = recall_score(Y_test, prediction, average="macro")
F1 = f1_score(Y_test, prediction, average="macro")
print ('Precision:', precision)
print ('Recall:', recall)
print ('F1:', F1)
```

```
Precision: 0.7515294254824031
Recall: 0.7271537615074718
F1: 0.7357228658048169
```

```
X_tfidf_test = vectorizer_tfidf.transform(data_test.text)
print("Training Classifier with Tfidf representation")
clf.fit(X_tfidf, Y_training)
```

```
Training Classifier with Tfidf representation
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
prediction_tfidf = clf.predict(X_tfidf_test)
accuracy_tfidf = accuracy_score(Y_test, prediction_tfidf)

# 'micro':Calculate metrics globally by counting the total true positives, false negatives an
precision_tfidf = precision_score(Y_test, prediction_tfidf, average="micro")
recall_tfidf = recall_score(Y_test, prediction_tfidf, average="micro")
F1_tfidf = f1_score(Y_test, prediction_tfidf, average="micro")
print ('Accuracy:', accuracy_tfidf)
print ('Precision:', precision_tfidf)
print ('Recall:', recall_tfidf)
print ('F1:', F1_tfidf)
```

```
Accuracy: 0.75493316359007
Precision: 0.75493316359007
Recall: 0.75493316359007
F1: 0.7549331635900699
```

```
precision_tfidf = precision_score(Y_test, prediction_tfidf, average="macro")
```

```
recall_tfidf = recall_score(Y_test, prediction_tfidf, average="macro")
F1_tfidf = f1_score(Y_test, prediction_tfidf, average="macro")
print ('Precision:', precision_tfidf)
print ('Recall:', recall_tfidf)
print ('F1:', F1_tfidf)
```

```
Precision: 0.7635306830072374
Recall: 0.7315009983217239
F1: 0.7418755829707557
```