In [4]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
```

In [5]:
```
def convert_sentiment (score):
    score = int(score)
    if score > 2:
        label = 1
    elif score == 2:
        label = 0
    else:
        label = -1

    return label
```

In [6]:
```
dataset = pd.read_csv('../data/train.tsv', sep='\t', header=0)

dataset = dataset.sample(frac =.50)
# Convert text to lowercase
dataset['Phrase'] = dataset['Phrase'].str.strip().str.lower()
dataset['Sentiment'] = dataset['Sentiment'].map(lambda a: convert_sentiment(a))
dataset.info()

dataset.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 78030 entries, 29747 to 131857
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   PhraseId     78030 non-null  int64 
 1   SentenceId   78030 non-null  int64 
 2   Phrase       78030 non-null  object
 3   Sentiment    78030 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 3.0+ MB
```

Out[6]:

	PhraseId	SentenceId	Phrase	Sentiment
29747	29748	1377	their 70s ,	0
140739	140740	7635	more overtly silly dialogue	-1
114728	114729	6105	see all summer	1
13525	13526	581	the subject 's	0
109528	109529	5802	young guns	0

In [7]:
```
vectorizer = TfidfVectorizer(min_df = 5,
                             max_df = 0.8,
                             sublinear_tf = True,
                             use_idf = True)
x = dataset['Phrase']
y = dataset['Sentiment']


x, x_test, y, y_test = train_test_split(x,y,test_size=0.25, random_state=101)
print(x_test)
```

```
11511                                              dignity
101940                                            cut open
115960                                 an unrewarding collar
1220                                     have that option .
20803     collateral damage is trash , but it earns extr...
                                ...                        
103984                                  knows the mistakes
57302          overall an overwhelmingly positive portrayal
31422                                  a dependable concept
95287     it 's too interested in jerking off in all its...
54294                                           the band
Name: Phrase, Length: 19508, dtype: object
```

In [23]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

#Vectorizing the text data
X_train_vec = vectorizer.fit_transform(x)
X_test_vec = vectorizer.transform(x_test)
```

In [24]:
```
from sklearn.linear_model import LogisticRegression
#Training the model
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train_vec, y)
```

Out[24]:
```
LogisticRegression(max_iter=1000)
```

In [27]:
```
from sklearn.metrics import classification_report

lr_score = lr.score(X_test_vec, y_test)
print("Results for Logistic Regression with TfidfVectorizer")
print(lr_score)
print("\n")

prediction_lr = lr.predict(X_test_vec)
lr_report = classification_report(y_test, prediction_lr, output_dict=True)
print(lr_report)
```

```
Results for Logistic Regression with TfidfVectorizer
0.7020196842321099


{'-1': {'precision': 0.6912235746316464, 'recall': 0.511011129528771, 'f1-score': 0.5876106194690266, 'support': 4223}, '0': {'precision': 0.6903300713219439, 'recall': 0.8354074668807707, 'f1-score': 0.7559713014258468, 'support': 9964}, '1': {'precision': 0.7423752310536045, 'recall': 0.6038338658146964, 'f1-score': 0.6659757487822573, 'support': 5321}, 'accuracy': 0.7020196842321099, 'macro avg': {'precision': 0.7079762923357317, 'recall': 0.6500841540747461, 'f1-score': 0.6698525565590435, 'support': 19508}, 'weighted avg': {'precision': 0.7047193249311832, 'recall': 0.7020196842321099, 'f1-score': 0.6949781962628269, 'support': 19508}}
```

In [12]:
```
import joblib
import pickle

# Save model
joblib.dump(lr, '../model/linear_regression.pkl')
```

Out[12]:
```
['../model/linear_regression.pkl']
```

In [13]:
```
review = """Do not purchase this product. My cell phone blast when I switched the charger"""
review_vector = vectorizer.transform([review])
print(vectorizer.transform([review]))
print(lr.predict(review_vector))
```

```
  (0, 8972)	0.27395844780858786
  (0, 8230)	0.19735076325449857
  (0, 8189)	0.10800617494822799
  (0, 6282)	0.3771024540970851
  (0, 5971)	0.4331715490291515
  (0, 5538)	0.21863459150407488
  (0, 5385)	0.3050127014620418
  (0, 2383)	0.26687417594949736
  (0, 1287)	0.39788769009814856
  (0, 882)	0.41727687108223477
[0]
```

In [64]:
```
from sklearn import svm
from sklearn.metrics import classification_report

print("This SVm fitting process could take a while..")

svm_model = svm.SVC(kernel='linear')
svm_model.fit(X_train_vec, y)

prediction_linear = svm_model.predict(X_test_vec)
report_svm = classification_report(y_test, prediction_linear, output_dict=True)
```

```
This SVm fitting process could take a while..
```

In [72]:
```
import joblib
import pickle

# Save model
joblib.dump(svm_model, '../model/svm_linear_model_2.pkl')

file = open('../temp/vec_svm', 'wb')
pickle.dump(vectorizer, file)
file.close()
```

In [73]:
```
svm_model = joblib.load('../model/svm_linear_model.pkl')
from sklearn import metrics

print("Results for SVM with TfidfVectorizer")

accuracy_score = metrics.accuracy_score(prediction_linear, y_test)
print(str('Accuracy for SVM Model is {:04.2f}'.format(accuracy_score*100))+'%')

print("\n")
print(report_svm)
```

```
Results for SVM with TfidfVectorizer
Accuracy for SVM Model is 70.56%


{'-1': {'precision': 0.6993159203980099, 'recall': 0.5191597414589104, 'f1-score': 0.5959194488606253, 'support': 4332}, '0': {'precision': 0.6886714214629291, 'recall': 0.8381106176826807, 'f1-score': 0.7560775744433215, 'support': 9908}, '1': {'precision': 0.7583845063769485, 'recall': 0.6095292331055429, 'f1-score': 0.6758577114165438, 'support': 5268}, 'accuracy': 0.7055566946893582, 'macro avg': {'precision': 0.7154572827459624, 'recall': 0.6555599864082378, 'f1-score': 0.6759515795133093, 'support': 19508}, 'weighted avg': {'precision': 0.7098607028200044, 'recall': 0.7055566946893582, 'f1-score': 0.6988496051964435, 'support': 19508}}
```

In [68]:
```
def get_sentiment_category(score, threshold):
    if score > threshold:
        label = 'This input is positive :)'
    elif score == threshold:
        label = 'This input is neutral -_-'
    else:
        label = 'This input is negative :('

    return label
```

In [69]:
```
# loaded_model = joblib.load('../model/svm_linear_model.pkl')
review = """I love this movie"""
review_vector = vectorizer.transform([review]) # vectorizing
review_score = svm_model.predict(review_vector)

print(get_sentiment_category(review_score[0], 0))
```

```
This input is positive :)
```

In [71]:
```
review = """I hate this movie"""
review_vector = vectorizer.transform([review]) # vectorizing
review_score = svm_model.predict(review_vector)

print(get_sentiment_category(review_score[0], 0))
```

```
This input is negative :(
```

In [ ]:
```

```