In [29]:
```python
## we will now load our data
```

In [169…
```python
import random
## create set values
class Sentiment:
    NEGATIVE = "NEGATIVE"
    NEUTRAL = "NEUTRAL"
    POSITIVE = "POSITIVE"

class Review:
    def __init__(self, text, score):
        self.text = text
        self.score = score
        self.sentiment = self.get_sentiment()

    def get_sentiment(self):
        if self.score <= 2:
            return Sentiment.NEGATIVE
        elif self.score == 3:
            return Sentiment.NEUTRAL
        else: #Score of 4 or 5
            return Sentiment.POSITIVE

class ReviewContainer:
    def __init__(self, reviews):
        self.reviews = reviews

    def get_text(self):
        return [x.text for x in self.reviews]

    def get_sentiment(self):
        return [x.sentiment for x in self.reviews]

    def evenly_distribute(self):
        negative = list(filter(lambda x: x.sentiment == Sentiment.NEGATIVE, self.reviews))
        positive = list(filter(lambda x: x.sentiment == Sentiment.POSITIVE, self.reviews))
        positive_shrunk = positive[:len(negative)]
        self.reviews = negative + positive_shrunk
        random.shuffle(self.reviews)
```

In [170…
```python
## We are building this data model to extract the reviews and the overall rating
import json
file_name='./Books_small_10000.json'

reviews =[]
with open(file_name) as f:
    for line in f:
        review=json.loads(line)
        reviews.append(Review(review['reviewText'],review['overall']))

reviews[5].sentiment
```

Out[170]:
```
'POSITIVE'
```

In [191…
```python
## Now we will divide our data into training data to train the Algo and Test data to see if it accurately
##depicts the Sentiment Data Prep

from sklearn.model_selection import train_test_split

Training, Test =train_test_split(reviews, test_size=0.33,random_state=42)

train_container=ReviewContainer(Training)
test_container=ReviewContainer(Test)
cont.evenly_distribute()
len(cont.reviews)
```

```
It was okay. I have read all the Duggars books and they sorta overlap each other and it's like the same book! N
ot thrilled with ALOT of there views and actually got the book to find out more...let's just say I seen all I n
eed to. Not a big fan.
436
436
```

Out[191]:
```
872
```

In [172…
```python
print(Training[0].sentiment)
```

```
POSITIVE
```

In [192…
```python
train_container.evenly_distribute()
train_x = train_container.get_text()
train_y = train_container.get_sentiment()

test_container.evenly_distribute()
test_x = test_container.get_text()
test_y = test_container.get_sentiment()
```

```
print(train_y.count(Sentiment.POSITIVE))
print(train_y.count(Sentiment.NEGATIVE))
```

```
436
436
```

In [224…

```
#Now we will use a method called bag of words method to extract features from text documents.
#These features can be used for training machine learning algorithms.
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
vectorizer = TfidfVectorizer ()
train_x_vector= vectorizer.fit_transform(train_x)
print(train_x[0])
print(train_x_vector[0])
test_x_vectors = vectorizer.transform(test_x)
```

```
Colton - Annoying and too ControllingMelanie - submissive to his kisses and his lovemaking.Hated the book. Don'
t buy
  (0, 1177)	0.21907036353461787
  (0, 2383)	0.16289068338656323
  (0, 991)	0.07581628283131062
  (0, 7929)	0.061685676591184666
  (0, 3660)	0.2876533689028302
  (0, 4787)	0.36526329916498107
  (0, 4486)	0.36526329916498107
  (0, 3793)	0.2913499890150326
  (0, 8052)	0.0670584054770816
  (0, 7610)	0.3443414245118173
  (0, 1732)	0.36526329916498107
  (0, 8079)	0.15107091897051994
  (0, 423)	0.1305119101535069
  (0, 447)	0.25796475629751126
  (0, 1550)	0.3443414245118173
```

In [225…

```
##Classfication this is Linear SVM to predict the sentiment of some data
from sklearn import svm
clf_svm =svm.SVC(kernel='linear')
clf_svm.fit(train_x_vector,train_y)
test_x[0]
clf_svm.predict(test_x_vectors[0])
```

Out[225]:

```
array(['NEGATIVE'], dtype='<U8')
```

Decison Tree

In [226…

```
from sklearn.tree import DecisionTreeClassifier
clf_dec =DecisionTreeClassifier()
clf_dec.fit(train_x_vector,train_y)
clf_dec.predict(test_x_vectors[0])
```

Out[226]:

```
array(['NEGATIVE'], dtype='<U8')
```

In [227…

```
## Logistic Regression
from sklearn.linear_model import LogisticRegression
clf_log =LogisticRegression()
clf_log.fit(train_x_vector,train_y)
clf_log.predict(test_x_vectors[0])
```

Out[227]:

```
array(['NEGATIVE'], dtype='<U8')
```

Evaluation

In [228…

```
## Mean Accuracy
print(clf_svm.score(test_x_vectors,test_y))
print(clf_dec.score(test_x_vectors,test_y))
print(clf_log.score(test_x_vectors,test_y))
```

```
0.8076923076923077
0.6634615384615384
0.8052884615384616
```

In [229…

```
## F1 scores to see if our model predicts the values based of sentiment
from sklearn.metrics import f1_score
print(f1_score(test_y,clf_svm.predict(test_x_vectors),average=None, labels=[Sentiment.POSITIVE,Sentiment.NEUTRA
##print(f1_score(test_y,clf_dec.predict(test_x_vectors),average=None, labels=[Sentiment.POSITIVE,Sentiment.NEUT
##print(f1_score(test_y,clf_log.predict(test_x_vectors),average=None, labels=[Sentiment.POSITIVE,Sentiment.NEUT
```

```
[0.80582524 0.         0.80952381]
C:\Users\coold\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1580: UndefinedMetricWarning: F-s
core is ill-defined and being set to 0.0 in labels with no true nor predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, "true nor predicted", "F-score is", len(true_sum))
```

In [230…

```
print(test_y.count(Sentiment.POSITIVE))
print(test_y.count(Sentiment.NEGATIVE))
```

```
208
208
```

In [232…

```
### Now we will create and run some test data to see how well our machine learning model correctly predicts the
## of these reviews

test_set=['It was okay. I have read all the Duggars books and they sorta overlap each other and its like the sa
test_set = ['very fun', "bad book do not buy", 'horrible waste of time']
test_set3 =['Very Bad','I Loved it','it was okay', 'it needs better story writing','Ada Sucks','I was so good']
new_test = vectorizer.transform(test_set)
new_test2 = vectorizer.transform(test_set2)
new_test3 = vectorizer.transform(test_set3)
print(clf_svm.predict(new_test))
print(clf_svm.predict(new_test2))
print(clf_svm.predict(new_test3))
```

```
['POSITIVE' 'NEGATIVE' 'NEGATIVE']
['NEGATIVE']
['NEGATIVE' 'POSITIVE' 'NEGATIVE' 'NEGATIVE' 'POSITIVE' 'POSITIVE']
```

In [237…

```
## now we will try to improve the machine learning model one more time by making the model determine which para
## is best for the model at that moment
from sklearn.model_selection import GridSearchCV
parameters = {'kernel':('linear','rbf'), 'C': (1,4,8,16,32)}

svc = svm.SVC()
clf = GridSearchCV(svc,parameters,cv=5)
clf.fit(train_x_vector,train_y)
```

Out[237]:

```
GridSearchCV(cv=5, estimator=SVC(),
             param_grid={'C': (1, 4, 8, 16, 32), 'kernel': ('linear', 'rbf')})
```

In [252…

```
print(clf.score(test_x_vectors, test_y))
```

```
0.8197115384615384
```

In [274…

```
print(clf_svm.predict(test_x_vectors[4]))
```

```
['POSITIVE']
```