NLP Final Project Report

Name: Ramya Chowdary Patchala **Email**: rpatchal@syr.edu

Dataset:

I have used **Kaggle Movie Reviews Dataset** to perform Classification.

Kaggle Movie Reviews dataset has 156,060 phrases in Training dataset, and test dataset has 66,292 reviews.

In the dataset, for each review has a sentiment where 0 represents negative sentiment, 1 represents slightly negative sentiment, 2 is neutral sentiment, 3 is slightly positive sentiment, 4 is positive sentiment.

I have achieved an over-all accuracy of **85%** using Naïve-Bayes Algorithm. You can see how I got it step-by-step below:

Text Processing:

1. **Tokenization**: I have performed tokenization on the reviews statements and obtained tokens.
2. **Normalization**: I have lowercased all the tokens just to ensure that same words in different cases are not treated as different tokens.
3. **Lemmetization**: Applied Lemmetization so that we can reduce the words into their base format and reduce the dimensionality of the feature space.
4. **Filtering**: Removed stopwords, punctuations, digits and special symbols.

We have 5 sentiments, I have converted 0,1 as negative, 2 is neutral, 3, 4 as positive.

In our data, we now have three groups of data like **negative**, **neutral** and **positive**. Further I will be training our model in two cases which is to classify the reviews into these three classes. The other one is to classify the reviews into two classes positive and negative, by excluding neutral class.

```
for phrase in phraselist:
    tokens = nltk.word_tokenize(phrase[0])
    tokens = [word for word in tokens if word not in string.punctuation]
    tokens = [word.lower() for word in tokens]
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    tokens = [word for word in tokens if not word.isdigit()]
    tokens = [re.sub(r'[^\w\s]', '', word) for word in tokens]

    #tokens = [word for word in tokens if word.lower() not in stop_words]
    sentiment = int(phrase[1])
    if (sentiment == 2):
        neutraldocs.append((tokens, 'neutral'))
        #phrasedocs.append((tokens, 'neutral'))
    if ((sentiment == 0) or (sentiment == 1)):
        phrasedocs.append((tokens, 'negative'))
    if ((sentiment == 3) or (sentiment == 4)):
        phrasedocs.append((tokens, 'positive'))
```

Feature Engineering:

1. **Bag of words feature function**:
   This function takes a document and a set of word features. It then creates a dictionary where each word feature is represented by a Boolean value indicating whether or not it appears in the document, prefixed with 'V_' for identification.

```
def document_features(document, word_features):
  document_words = set(document)
  features = {}
  for word in word_features:
    features['V_{}'.format(word)] = (word in document_words)
  return features
```

Model to classify into Positive, Negative and Neutral:

We have trained Naïve Bayes Model on 40000 reviews. We have split training dataset into two parts, 70% for training the model, and 30% for testing the model. Upon using the Bag of words feature function, we have achieved **accuracy of 64.02% and F-1 Score of 55.81%**

```
PS C:\Users\sonar\Downloads\FinalProjectData (5)\FinalProjectData\kagglemoviereviews> python classifyKaggle.py corpus/ 40000
Read 156060 phrases, using 40000 random phrases
['irresistible package', '3']
['is little else', '1']
['that spends a bit too much time on its fairly ludicrous plot .', '0']
['oddly winning portrayal', '2']
['of the power of inertia to arrest development in a dead-end existence', '2']
['silent-movie comedy', '2']
['The movie is silly beyond comprehension , and', '0']
['a wonderfully warm human drama', '4']
['a text to ` lick ,', '1']
['Me no lika da accents so good ,', '1']
(['irresistible', 'package'], 'positive')
(['is', 'little', 'else'], 'negative')
         |    e    l    e |
---------+---------------+
negative |<1142>1220  213 |
 neutral |  664<5003> 503 |
positive |  348 1370<1537>|
---------+---------------+
(row = reference; col = test)

Overall Accuracy 0.6401666666666667
Precision: 0.47219662058371736
Recall: 0.6822015090989791
F1 Score: 0.5580973129992737
```

Model to classify into Positive and Negative, excluding Neutral:

Upon running the same model, but excluding the neutrals, and just classifying the reviews into Positive and Negative classes, we have achieved **accuracy of 76.76% and F-1 Score of 79.92%**

```
PS C:\Users\sonar\Downloads\FinalProjectData (5)\FinalProjectData\kagglemoviereviews> python classifyKaggle.py corpus/ 40000
Read 156060 phrases, using 40000 random phrases
['sink it', '1']
['sequences boring', '1']
['the first 2\\/3 of the film are incredibly captivating and insanely funny , thanks in part to interesting cinematic devices -LRB- cool visual backmasking -RRB- , a solid cast , and
some wickedly sick and twisted humor ...', '4']
['be captivated , as I was , by its moods , and by its subtly transformed star , and still wonder why Paul Thomas Anderson ever had the inclination to make the most sincere and artful
 movie in which Adam Sandler will probably ever appear', '4']
["'ve got the wildly popular Vin Diesel in the equation", '2']
['Crush could be the worst film a man has made about women since Valley of the Dolls .', '0']
["she 's pretty and", '4']
['An old-fashioned but emotionally stirring adventure tale of the kind they', '3']
['that deserves recommendation', '3']
['the last 10 minutes', '2']
(['sink', 'it'], 'negative')
(['sequence', 'boring'], 'negative')
(['the', 'first', '23', 'of', 'the', 'film', 'are', 'incredibly', 'captivating', 'and', 'insanely', 'funny', 'thanks', 'in', 'part', 'to', 'interesting', 'cinematic', 'device', 'lrb',
        |    i    i |
        |    v    v |
        |    e    e |
---------+----------+
negative |<3660>1745 |
positive | 1044<5551>|
---------+----------+
(row = reference; col = test)

Overall Accuracy 0.7675833333333333
Precision: 0.8416982562547385
Recall: 0.760827850877193
F1 Score: 0.799222518177237
```

2. **Positive and Negative Classes Feature function:**

This function takes a document, a set of word features, and lists of positive and negative prefixes. It creates a feature vector where each word feature is represented by a Boolean value indicating its presence in the document, and it adds additional features to check if any word in the document starts with a positive or negative prefix, providing context beyond just word occurrences.

```python
# define a feature definition function here
def document_features_list(document, word_features, poslist, neglist):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['V_{}'.format(word)] = (word in document_words)
    # Check if any word in the document starts with a positive prefix
    features['contains_positive'] = any(word.startswith(pos) for pos in poslist for word in document)
    # Check if any word in the document starts with a negative prefix
    features['contains_negative'] = any(word.startswith(neg) for neg in neglist for word in document)
    return features
```

Model to classify into Positive, Negative and Neutral:

We have trained Naïve Bayes Model on 40000 reviews. We have split training dataset into two parts, 70% for training the model, and 30% for testing the model. Upon using the feature function with Positive and Negative Classes, we have achieved **accuracy of 63.52% and F-1 Score of 55.76%**

```
PS C:\Users\sonar\Downloads\FinalProjectData (5)\FinalProjectData\kagglemoviereviews> python classifyKaggle.py corpus/ 40000
Read 156060 phrases, using 40000 random phrases
positive |  279 1390<1561>|
---------+----------------+
(row = reference; col = test)

Overall Accuracy 0.63525
Precision: 0.48328173374613004
Recall: 0.659206081081081
F1 Score: 0.5576991782779563
PS C:\Users\sonar\Downloads\FinalProjectData (5)\FinalProjectData\kagglemoviereviews>
```

Model to classify into Positive and Negative, excluding Neutral:

Upon running the same model, but excluding the neutrals, and just classifying the reviews into Positive and Negative classes, we have achieved **accuracy of 76.82% and F-1 Score of 79.93%**

```
['feel more like a non-stop cry for attention , than an attempt at any kind of satisfying entertainment .', '1']
['from darkness', '2']
["he should have shaped the story to show us why it 's compelling", '2']
['adolescent sturm und drang', '2']
['care about cleverness , wit or any other kind of intelligent humor', '3']
['Academy Award', '3']
['feels more like an extended , open-ended poem than a traditionally structured story', '3']
['a fairly weak retooling', '1']
(['predictably', 'soulless', 'technotripe'], 'negative')
         |    g    s |
         |    a    i |
         |    t    t |
         |    i    i |
         |    v    v |
         |    e    e |
---------+-----------+
negative |<3682>1687 |
positive | 1094<5537>|
---------+-----------+
(row = reference; col = test)

Overall Accuracy 0.76825
Precision: 0.8350173427838938
Recall: 0.7664728682170543
F1 Score: 0.7992782389029232
PS C:\Users\sonar\Downloads\FinalProjectData (5)\FinalProjectData\kagglemoviereviews>
```

3. Bigrams Feature function:

This function takes a document and two sets of features (individual words and bigrams), then checks if each feature is present in the document, returning a dictionary indicating their presence.

```python
def bigram_document_features(document, word_features, bigram_features):
    document_words = set(document)
    document_bigrams = nltk.bigrams(document)
    features = {}
    for word in word_features:
        features['V_{}'.format(word)] = (word in document_words)
    for bigram in bigram_features:
        features['B_{}_{}'.format(bigram[0], bigram[1])] = (bigram in document_bigrams)
    return features
```

We have used PMI measure for bigrams. We have also tried to use Chi.sq, and raw-freq measures for bigrams, but found PMI Measure has better performance.

```python
finder = BigramCollocationFinder.from_words(all_words_list)
bigram_features = finder.nbest(bigram_measures.pmi, 250)
```

Model to classify into Positive, Negative and Neutral:

We have trained Naïve Bayes Model on 40000 reviews. We have split training dataset into two parts, 70% for training the model, and 30% for testing the model. Upon using the feature function with bigrams, we have achieved **accuracy of 63.64% and F-1 Score of 54.72%**

```
PS C:\Users\sonar\Downloads\FinalProjectData (5)\FinalProjectData\kagglemoviereviews> python classifyKaggle.py corpus/ 40000
Read 156060 phrases, using 40000 random phrases
['those who pride themselves on sophisticated , discerning taste', '2']
['portraits', '2']
[', there would be a toss-up between presiding over the end of cinema as we know it and another night of delightful hand shadows .', '1']
['to the endlessly repetitive scenes of embarrassment', '1']
["fairly solid -- not to mention well edited so that it certainly does n't feel like a film that strays past the two and a half mark", '3']
['our tears , our sympathies', '2']
["There 's back-stabbing , inter-racial desire and , most importantly , singing and dancing .", '3']
['a theatrical air', '2']
['Duvall -LRB- also a producer -RRB- peels layers from this character that may well not have existed on paper .', '3']
['perfectly respectable', '3']
(['those', 'who', 'pride', 'themselves', 'on', 'sophisticated', 'discerning', 'taste'], 'neutral')
(['portrait'], 'neutral')
(['there', 'would', 'be', 'a', 'tossup', 'between', 'presiding', 'over', 'the', 'end', 'of', 'cinema', 'a', 'we', 'know', 'it', 'and', 'another', 'night', 'of', 'delightful', 'hand',
'shadow'], 'negative')
---------+----------------+
(row = reference; col = test)

Overall Accuracy 0.6364166666666666
Precision: 0.4600182426269383
Recall: 0.6754464285714286
F1 Score: 0.5472960752396455
```

Model to classify into Positive and Negative, excluding Neutral:

Upon running the same model, but excluding the neutrals, and just classifying the reviews into Positive and Negative classes, we have achieved **accuracy of 77.34% and F-1 Score of 80.41%**

```
PS C:\Users\sonar\Downloads\FinalProjectData (5)\FinalProjectData\kagglemoviereviews> python classifyKaggle.py corpus/ 40000
Read 156060 phrases, using 40000 random phrases
['gobble', '2']
['the eventual DVD release', '2']
['movie that portrays the frank humanity of ... emotional recovery', '3']
["Shunji Iwai 's", '2']
['an object of intense scrutiny for 104 minutes', '2']
['and just plain dumb', '0']
['turn in perfectly executed and wonderfully sympathetic characters , who are alternately touching and funny', '3']
['the tailor for some major alterations', '2']
["her son 's discovery", '2']
['maybe the original inspiration', '2']
(['movie', 'that', 'portrays', 'the', 'frank', 'humanity', 'of', '', 'emotional', 'recovery'], 'positive')
(['and', 'just', 'plain', 'dumb'], 'negative')
(['turn', 'in', 'perfectly', 'executed', 'and', 'wonderfully', 'sympathetic', 'character', 'who', 'are', 'alternately', 'touching', 'and', 'funny'], 'positive')
(['is', 'nt', 'one', 'moment', 'in', 'the', 'film', 'that', 'surprise', 'or', 'delight'], 'negative')
(['is', 'visually', 'smart', 'cleverly', 'written'], 'positive')
(['goldbacher', 'draw', 'on', 'an', 'elegant', 'visual', 'sense', 'and', 'a', 'talent', 'for', 'easy', 'seductive', 'pacing'], 'positive')
(['this', 'comic', 'gem'], 'positive')
(['trap', 'audience', 'in', 'a', 'series', 'of', 'relentlessly', 'nasty', 'situation', 'that', 'we', 'would', 'pay', 'a', 'considerable', 'ransom', 'not', 'to', 'be', 'looking', 'at']
, 'negative')
(['too', 'mainstream'], 'negative')
(['until', 'thing', 'fall', 'apart'], 'negative')
12264
Confusion Matrix:
         |    n    p |
         |    e    o |
         |    g    s |
         |    a    i |
         |    t    t |
         |    i    i |
         |    v    v |
         |    e    e |
---------+-----------+
negative |<3701>1672 |
positive | 1047<5580>|
---------+-----------+
(row = reference; col = test)

Overall Accuracy 0.7734166666666666
Precision: 0.8420099592575826
Recall: 0.7694429123000551
F1 Score: 0.8040925138698753
```

4. Subjectivity Lexicon (SL) Feature function:

This function extracts features from a document, including word presence and counts of positive and negative subjectivity words, categorized by their strength and polarity according to a subjectivity lexicon (SL).

```python
def SL_features(document, word_features, SL):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['V_{}'.format(word)] = (word in document_words)
    # count variables for the 4 classes of subjectivity
    weakPos = 0
    strongPos = 0
    weakNeg = 0
    strongNeg = 0
    for word in document_words:
        if word in SL:
            strength, posTag, isStemmed, polarity = SL[word]
            if strength == 'weaksubj' and polarity == 'positive':
                weakPos += 1
            if strength == 'strongsubj' and polarity == 'positive':
                strongPos += 1
            if strength == 'weaksubj' and polarity == 'negative':
                weakNeg += 1
            if strength == 'strongsubj' and polarity == 'negative':
                strongNeg += 1
        features['positivecount'] = weakPos + (5 * strongPos)
        features['negativecount'] = weakNeg + (5 * strongNeg)
    return features
```

Model to classify into Positive, Negative and Neutral:

We have trained Naïve Bayes Model on 40000 reviews. We have split training dataset into two parts, 70% for training the model, and 30% for testing the model. Upon using the feature function with Subjectivity Lexicons, we have achieved **accuracy of 64.81% and F-1 Score of 58.85%**

```
PS C:\Users\sonar\Downloads\FinalProjectData (5)\FinalProjectData\kagglemoviereviews> python classifyKaggle.py corpus/ 40000
Read 156060 phrases, using 40000 random phrases
['hyped up', '2']
['right questions', '3']
['are ones in formulaic mainstream movies', '2']
['become almost as operatic to us', '1']
["from Elfriede Jelinek 's novel", '2']
['workshops', '2']
["both people 's", '2']
["Almost as offensive as `` Freddy Got Fingered . ''", '1']
['when the melodramatic aspects start to overtake the comedy', '2']
['every cliche in the war movie compendium across its indulgent two-hour-and-fifteen-minute length', '1']
(['hyped', 'up'], 'neutral')
(['right', 'question'], 'positive')
         |    v    a    v |
         |    e    l    e |
---------+----------------+
negative |<1247>1153  268 |
 neutral |  612<4772> 620 |
positive |  277 1293<1758>|
---------+----------------+
(row = reference; col = test)

Overall Accuracy 0.6480833333333333
Precision: 0.5282451923076923
Recall: 0.6643990929705216
F1 Score: 0.5885503850016738
```

Model to classify into Positive and Negative, excluding Neutral:

Upon running the same model, but excluding the neutrals, and just classifying the reviews into Positive and Negative classes, we have achieved **accuracy of 80.18% and F-1 Score of 82.44%**

```
PS C:\Users\sonar\Downloads\FinalProjectData (5)\FinalProjectData\kagglemoviereviews> python classifyKaggle.py corpus/ 40000
Read 156060 phrases, using 40000 random phrases
['the thin veneer of nationalism that covers our deepest , media-soaked fears', '2']
['a story , an old and scary one , about the monsters we make , and the vengeance they', '2']
['The Waterboy', '2']
['that is tragically rare in the depiction of young women in film', '2']
['Sand , who brings to the role her pale , dark beauty and characteristic warmth', '3']
['the performances by Harris , Phifer and Cam', '2']
['Where Art Thou', '2']
['Those of you who are not an eighth grade girl will most likely doze off during this one .', '0']
['impressively discreet filmmakers', '3']
['An emotionally strong and politically potent piece', '3']
(['sand', 'who', 'brings', 'to', 'the', 'role', 'her', 'pale', 'dark', 'beauty', 'and', 'characteristic', 'warmth'], 'positive')

Overall Accuracy 0.8018333333333333
Precision: 0.8415259348612787
Recall: 0.8079038795599305
F1 Score: 0.8243722304283604
PS C:\Users\sonar\Downloads\FinalProjectData (5)\FinalProjectData\kagglemoviereviews>
```

5. Not features function:

This function marks word presence in a document and identifies negated instances by checking for negation words or word endings like "n't", assigning corresponding features accordingly.

```
negationwords = ['no', 'not', 'never', 'none', 'nowhere', 'nothing', 'noone', 'rather',
                 'hardly', 'scarcely', 'rarely', 'seldom', 'neither', 'nor']

def Not_features(document, word_features, negationwords):
    features = {}
    for word in word_features:
        features['V_{}'.format(word)] = False
        features['V_NOT{}'.format(word)] = False
        #go through document words in order
    for i in range(0, len(document)):
        word = document[i]
        if((i + 1) < len(document)) and ((word in negationwords) or (word.endswith("n't"))):
            i += 1
            features['V_NOT{}'.format(document[i])] = (document[i] in word_features)
        else:
            features['V_{}'.format(word)] = (word in word_features)
    return features
```

Model to classify into Positive, Negative and Neutral:

We have trained Naïve Bayes Model on 40000 reviews. We have split training dataset into two parts, 70% for training the model, and 30% for testing the model. Upon using the feature function with Subjectivity Lexicons, we have achieved **accuracy of 66% and F-1 Score of 59.82%**

```
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ python classifyKaggle.py 40000
[nltk_data] Downloading package punkt to /home/azureuser/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
PWD: /mnt/batch/tasks/shared/LS_root/mounts/clusters/rpatchal1/code/Users/rpatchal/project
Read 156060 phrases, using 40000 random phrases
["hits every cliche we 've come to expect , including the assumption that `` crazy '' people are innocent , childlike and inherently funny .",
 '0']
['to two', '2']
['a powerful , naturally dramatic piece of low-budget filmmaking', '4']
['brash young men set out to conquer the online world with laptops , cell phones and sketchy business plans', '3']
['with the sensibility of a particularly nightmarish fairytale', '2']
['with a kiss', '2']
["worth seeing for Ambrose 's performance .", '3']
['Sly , sophisticated and surprising .', '3']
['any of the character dramas , which never reach satisfying conclusions', '1']
['give a pretty good overall picture of the situation in Laramie following the murder of Matthew Shepard', '3']
(['hit', 'every', 'cliche', 'we', 've', 'come', 'to', 'expect', 'including', 'the', 'assumption', 'that', '', 'crazy', '', 'people', 'are', 'i
nnocent', 'childlike', 'and', 'inherently', 'funny'], 'negative')
(['to', 'two'], 'neutral')
(['a', 'powerful', 'naturally', 'dramatic', 'piece', 'of', 'lowbudget', 'filmmaking'], 'positive')
(['brash', 'young', 'men', 'set', 'out', 'to', 'conquer', 'the', 'online', 'world', 'with', 'laptop', 'cell', 'phone', 'and', 'sketchy', 'busi
ness', 'plan'], 'positive')
(['with', 'the', 'sensibility', 'of', 'a', 'particularly', 'nightmarish', 'fairytale'], 'neutral')
(['with', 'a', 'kiss'], 'neutral')
(['worth', 'seeing', 'for', 'ambrose', 's', 'performance'], 'positive')
(['sly', 'sophisticated', 'and', 'surprising'], 'positive')
(['any', 'of', 'the', 'character', 'drama', 'which', 'never', 'reach', 'satisfying', 'conclusion'], 'negative')
(['give', 'a', 'pretty', 'good', 'overall', 'picture', 'of', 'the', 'situation', 'in', 'laramie', 'following', 'the', 'murder', 'of', 'matthew
', 'shepard'], 'positive')
14126
Confusion Matrix:
        |    n         p |
        |    e    n    o |
```

```
14126
Confusion Matrix:
          |    n         p |
          |    e    n    o |
          |    g    e    s |
          |    a    u    i |
          |    t    t    t |
          |    i    r    i |
          |    v    a    v |
          |    e    l    e |
----------+----------------+
negative |<1475> 956  163 |
 neutral |  791<4708> 670 |
positive |  294 1206<1737>|
---------+----------------+
(row = reference; col = test)

Overall Accuracy 0.66
Precision: 0.5366079703429101
Recall: 0.675875486381323
F1 Score: 0.5982434992250731
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ ~
```

Model to classify into Positive and Negative, excluding Neutral:

Upon running the same model, but excluding the neutrals, and just classifying the reviews into Positive and Negative classes, we have achieved **accuracy of 80.3% and F-1 Score of 82.15%**

```
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ python classifyKaggle.py 40000
[nltk_data] Downloading package punkt to /home/azureuser/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
PWD: /mnt/batch/tasks/shared/LS_root/mounts/clusters/rpatchal1/code/Users/rpatchal/project
Read 156060 phrases, using 40000 random phrases
['unimaginative', '1']
['Tries -RRB-', '2']
['counter-cultural', '3']
['reruns and supermarket tabloids', '2']
['slimed in the name of High Art', '1']
['that few movies are able to accomplish', '4']
['think the central story of Brendan Behan is that he was a bisexual sweetheart before he took to drink', '2']
['perfect cure', '2']
['one of the most high-concept sci fi adventures attempted for the screen', '4']
['misconceived', '1']
(['unimaginative'], 'negative')
(['countercultural'], 'positive')
(['slimed', 'in', 'the', 'name', 'of', 'high', 'art'], 'negative')
(['that', 'few', 'movie', 'are', 'able', 'to', 'accomplish'], 'positive')
(['one', 'of', 'the', 'most', 'highconcept', 'sci', 'fi', 'adventure', 'attempted', 'for', 'the', 'screen'], 'positive')
(['misconceived'], 'negative')
(['though', 'well', 'dressed', 'and', 'well', 'made'], 'positive')
(['on', 'the', 'young', 'woman', 's', 'infirmity', 'and', 'her', 'naive', 'dream'], 'negative')
(['the', 'product', 'of', 'loving', 'well', 'integrated', 'homage'], 'positive')
(['performance', 'great', 'to', 'look', 'at', 'and', 'funny'], 'positive')
(['performance', 'great', 'to', 'look', 'at', 'and', 'funny'], 'positive')
12319
Confusion Matrix:
          |    n    p |
          |    e    o |
          |    g    s |
          |    a    i |
          |    t    t |
          |    i    i |
          |    v    v |
          |    e    e |
----------+-----------+
negative |<4196>1234 |
positive | 1130<5440>|
---------+-----------+
(row = reference; col = test)

Overall Accuracy 0.803
Precision: 0.8280060882800608
Recall: 0.8151033862750974
F1 Score: 0.8215040773180308
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ ~
```

6. **Comprehensive Feature function with bigrams, negations and subjectivity including neutrals:** This function extracts features from a document, including word presence, negated instances, bigram occurrences, and counts of words categorized by their subjectivity and polarity, based on two subjectivity lexicons (SL and SL2). It combines these features into a single dictionary representation of the document.

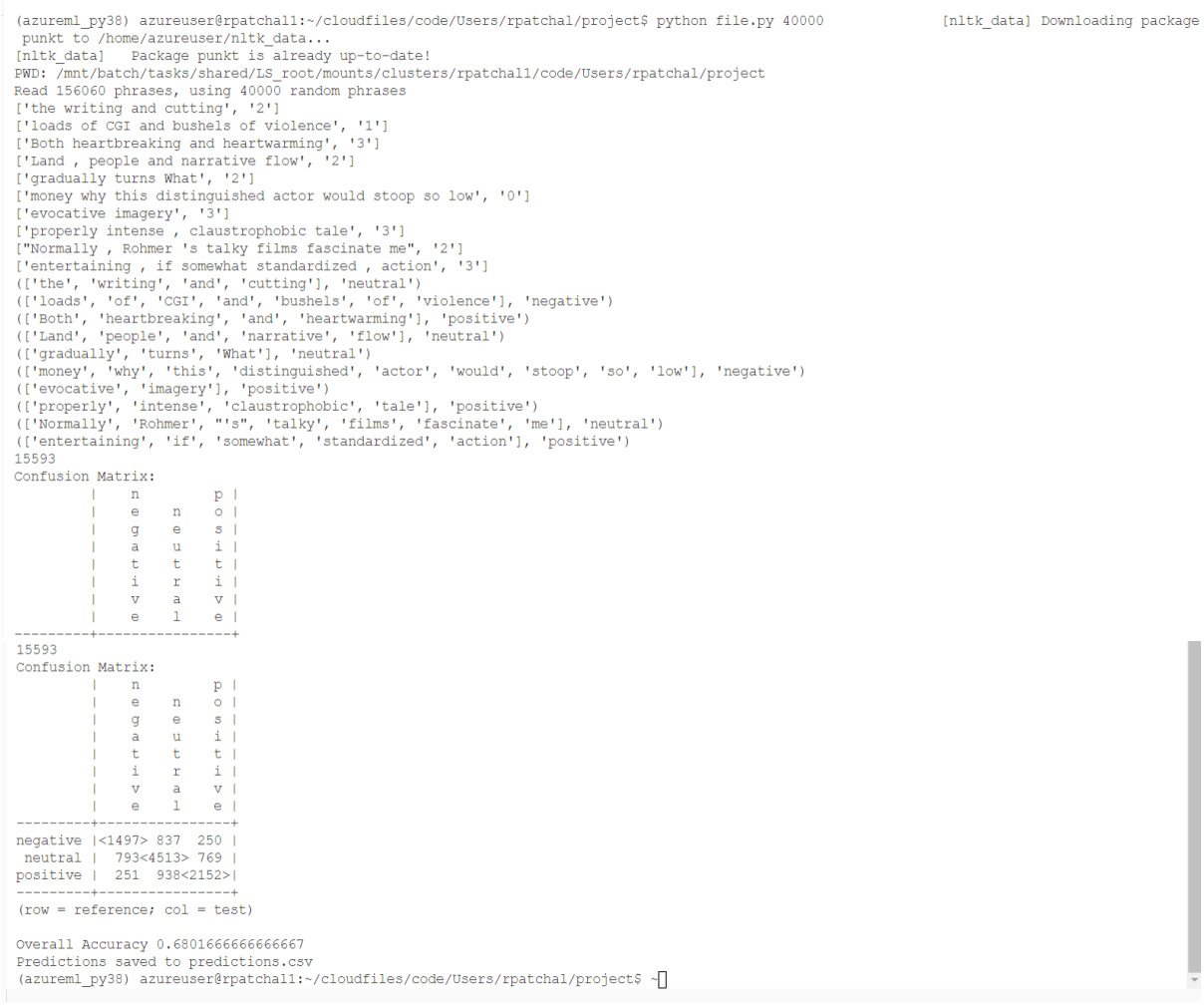
```
# function to extract comprehensive document features including sentiment, negation, bigrams and neutral word counts
def comprehensive_doc_features_neutral(document, word_features, SL, SL2, bigram_features, negationwords):
    document_words = set(document)
    document_bigrams = nltk.bigrams(document)
    features = {}
    # NEGATION WORDS: Generate V_ and V_NOT features based on negationwords
    for word in word_features:
        features['V_{}'.format(word)] = False
        features['V_NOT{}'.format(word)] = False
    for bigram in bigram_features:
        features['B_{}_{}'.format(bigram[0], bigram[1])] = (bigram in document_bigrams)
    # go through document words in order
    for i in range(0, len(document)):
        word = document[i]
        if ((i + 1) < len(document)) and ((word in negationwords) or (word.endswith("n't"))):
            i += 1
            features['V_NOT{}'.format(document[i])] = (document[i] in word_features)
        else:
            features['V_{}'.format(word)] = (word in word_features)
    # count variables for the 4 classes of subjectivity
    weakPos = 0
    strongPos = 0
    weakNeg = 0
    strongNeg = 0
    for word in document_words:
        if word in SL:
            strength, posTag, isStemmed, polarity = SL[word]
            if strength == 'weaksubj' and polarity == 'positive':
                weakPos += 1
            if strength == 'strongsubj' and polarity == 'positive':
                strongPos += 1
            if strength == 'weaksubj' and polarity == 'negative':
                weakNeg += 1
            if strength == 'strongsubj' and polarity == 'negative':
                strongNeg += 1
        features['positiveStrengthcount'] = weakPos + (5 * strongPos)
        features['negativeStrengthcount'] = weakNeg + (5 * strongNeg)

    posword = 0
    neutword = 0
    negword = 0
    for word in document_words:
        if word in SL2[0]:
            posword += 1
        if word in SL2[1]:
            neutword += 1
        if word in SL2[2]:
            negword += 1

    features['positivecount'] = posword
    features['neutralcount'] = neutword
    features['negativecount'] = negword

    return features
```

We have ran this feature function with **250 bigram features**. We have obtained the **accuracy of 68.02%** in classifying the reviews into positive, negative and neutral sentiments.



7. **Comprehensive Feature function with bigrams, negations and subjectivity with neutrals**: This function extracts features from a document, including word presence, negated instances, bigram occurrences, and counts of words categorized by their subjectivity and polarity, based on two subjectivity lexicons (SL and SL2). It combines these features into a single dictionary representation of the document.

   We have used same function as above, and used **1000 bigram features**, and obtained **accuracy of 67.21%** in classifying the reviews into positive, negative and neutral sentiments.

```
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ python file.py 40000
[nltk_data] Downloading package punkt to /home/azureuser/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
PWD: /mnt/batch/tasks/shared/LS_root/mounts/clusters/rpatchal1/code/Users/rpatchal/project
Read 156060 phrases, using 40000 random phrases
['frothing', '1']
["shadows Heidi 's trip", '2']
['works effortlessly at delivering genuine , acerbic laughs', '3']
['modestly', '3']
['of sex in a bid to hold our attention', '2']
['direct-to-video stuff', '1']
['sensitivities', '2']
["Sarah 's", '2']
['A modestly comic , modestly action-oriented World War II adventure that , in terms of authenticity', '3']
['remade for viewers who were in diapers when the original was released in 1987', '2']
(['frothing'], 'negative')
(['shadows', 'Heidi', "'s", 'trip'], 'neutral')
(['works', 'effortlessly', 'at', 'delivering', 'genuine', 'acerbic', 'laughs'], 'positive')
(['modestly'], 'positive')
(['of', 'sex', 'in', 'a', 'bid', 'to', 'hold', 'our', 'attention'], 'neutral')
(['direct-to-video', 'stuff'], 'negative')
(['sensitivities'], 'neutral')
(['Sarah', "'s"], 'neutral')
(['A', 'modestly', 'comic', 'modestly', 'action-oriented', 'World', 'War', 'II', 'adventure', 'that', 'in', 'te
rms', 'of', 'authenticity'], 'positive')
(['remade', 'for', 'viewers', 'who', 'were', 'in', 'diapers', 'when', 'the', 'original', 'was', 'released', 'in
', '1987'], 'neutral')
15565
Confusion Matrix:
          |    n         p |
          |    e    n    o |
```

```
['remade for viewers who were in diapers when the original was released in 1987', '2']
(['frothing'], 'negative')
(['shadows', 'Heidi', "'s", 'trip'], 'neutral')
(['works', 'effortlessly', 'at', 'delivering', 'genuine', 'acerbic', 'laughs'], 'positive')
(['modestly'], 'positive')
(['of', 'sex', 'in', 'a', 'bid', 'to', 'hold', 'our', 'attention'], 'neutral')
(['direct-to-video', 'stuff'], 'negative')
(['sensitivities'], 'neutral')
(['Sarah', "'s"], 'neutral')
(['A', 'modestly', 'comic', 'modestly', 'action-oriented', 'World', 'War', 'II', 'adventure', 'that', 'in', 'te
rms', 'of', 'authenticity'], 'positive')
(['remade', 'for', 'viewers', 'who', 'were', 'in', 'diapers', 'when', 'the', 'original', 'was', 'released', 'in
', '1987'], 'neutral')
15565
Confusion Matrix:
          |    n         p |
          |    e    n    o |
          |    g    e    s |
          |    a    u    i |
          |    t    t    t |
          |    i    r    i |
          |    v    a    v |
          |    e    l    e |
----------+-----------------+
negative |<1521> 920  250 |
 neutral |  793<4540> 799 |
positive |  254  918<2005>|
----------+-----------------+
(row = reference; col = test)

Overall Accuracy 0.6721666666666667
Predictions saved to predictions.csv
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$
```

8. **Comprehensive Feature function with bigrams, negations and subjectivity without neutrals:** This function creates a feature set for a document, capturing word presence, negated instances, bigram occurrences, and counts of words categorized by their subjectivity and polarity, based on a subjectivity lexicon (SL) and negation word list.

   The above function extends the first by including additional counts for subjectivity strength and introduces a new count for neutral words based on two separate lexicons (SL and SL2).
   This function extracts features focusing on word presence, negation, bigram occurrences, and counts of subjectivity classes (positive and negative).

```python
# function to extract comprehensive document features including sentiment and negation handling and bigrams
def comprehensive_doc_features(document, word_features, SL, bigram_features, negationwords):
    document_words = set(document)
    document_bigrams = nltk.bigrams(document)
    features = {}
    # NEGATION WORDS: Generate V_ and V_NOT features based on negationwords
    for word in word_features:
        features['V_{}'.format(word)] = False
        features['V_NOT{}'.format(word)] = False
    for bigram in bigram_features:
        features['B_{}_{}'.format(bigram[0], bigram[1])] = (bigram in document_bigrams)
    # go through document words in order
    for i in range(0, len(document)):
        word = document[i]
        if ((i + 1) < len(document)) and ((word in negationwords) or (word.endswith("n't"))):
            i += 1
            features['V_NOT{}'.format(document[i])] = (document[i] in word_features)
        else:
            features['V_{}'.format(word)] = (word in word_features)
    # count variables for the 4 classes of subjectivity
    weakPos = 0
    strongPos = 0
    weakNeg = 0
    strongNeg = 0

    for word in document_words:
        if word in SL:
            strength, posTag, isStemmed, polarity = SL[word]
            if strength == 'weaksubj' and polarity == 'positive':
                weakPos += 1
            if strength == 'strongsubj' and polarity == 'positive':
                strongPos += 1
            if strength == 'weaksubj' and polarity == 'negative':
                weakNeg += 1
            if strength == 'strongsubj' and polarity == 'negative':
                strongNeg += 1
        features['positivecount'] = weakPos + (5 * strongPos)
        features['negativecount'] = weakNeg + (5 * strongNeg)

    return features
```

We have ran this feature function and obtained the **accuracy of 81.67%** in classifying the reviews into positive and negative sentiments.

```
PWD: /mnt/batch/tasks/shared/LS_root/mounts/clusters/rpatchal1/code/Users/rpatchal/project
Read 156060 phrases, using 40000 random phrases
['on Elm Street', '2']
['the chemistry between Freeman and Judd', '2']
['than the first one', '2']
['orbit', '2']
['endeavour', '2']
['a taut contest of wills between Bacon and Theron', '2']
["alternate reality '", '2']
['old-time B movies', '2']
['High Crimes is a cinematic misdemeanor , a routine crime thriller remarkable only for its lack of logic and m
isuse of two fine actors , Morgan Freeman and Ashley Judd .', '0']
['is to imply terror by suggestion , rather than the overuse of special effects .', '3']
(['High', 'Crimes', 'is', 'a', 'cinematic', 'misdemeanor', 'a', 'routine', 'crime', 'thriller', 'remarkable', '
only', 'for', 'its', 'lack', 'of', 'logic', 'and', 'misuse', 'of', 'two', 'fine', 'actors', 'Morgan', 'Freeman'
, 'and', 'Ashley', 'Judd'], 'negative')
(['is', 'to', 'imply', 'terror', 'by', 'suggestion', 'rather', 'than', 'the', 'overuse', 'of', 'special', 'effe
cts'], 'positive')
(['that', 'educates', 'viewers', 'with', 'words', 'and', 'pictures', 'while', 'entertaining', 'them'], 'positiv
e')
(["'s", 'waltzed', 'itself', 'into', 'the', 'art', 'film', 'pantheon'], 'positive')
(['its', 'convolutions'], 'negative')
(['foreign', 'culture', 'only'], 'negative')
(['Brosnan', 'gives', 'a', 'portrayal', 'as', 'solid', 'and', 'as', 'perfect', 'as', 'his', 'outstanding', 'per
formance', 'as', 'Bond', 'in', 'Die', 'Another', 'Day'], 'positive')
(['is', 'sure', 'to', 'give', 'you', 'a', 'lot', 'of', 'laughs', 'in', 'this', 'simple', 'sweet', 'and', 'roman
tic', 'comedy'], 'positive')
(['such', 'subtlety', 'and', 'warmth'], 'positive')
(['there', 'was', 'any', 'doubt', 'that', 'Peter', "O'Fallon", 'did', "n't", 'have', 'an', 'original', 'bone',
'in', 'his', 'body'], 'negative')
13526
Confusion Matrix:
         |    n    p |
         |    e    o |
```

```
(["'s", 'waltzed', 'itself', 'into', 'the', 'art', 'film', 'pantheon'], 'positive')
(['its', 'convolutions'], 'negative')
(['foreign', 'culture', 'only'], 'negative')
(['Brosnan', 'gives', 'a', 'portrayal', 'as', 'solid', 'and', 'as', 'perfect', 'as', 'his', 'outstanding', 'per
formance', 'as', 'Bond', 'in', 'Die', 'Another', 'Day'], 'positive')
(['is', 'sure', 'to', 'give', 'you', 'a', 'lot', 'of', 'laughs', 'in', 'this', 'simple', 'sweet', 'and', 'roman
tic', 'comedy'], 'positive')
(['such', 'subtlety', 'and', 'warmth'], 'positive')
(['there', 'was', 'any', 'doubt', 'that', 'Peter', "O'Fallon", 'did', "n't", 'have', 'an', 'original', 'bone',
'in', 'his', 'body'], 'negative')
13526
Confusion Matrix:
         |    n    p |
         |    e    o |
         |    g    s |
         |    a    i |
         |    t    t |
         |    i    i |
         |    v    v |
         |    e    e |
---------+-----------+
negative |<4311>1112 |
positive | 1087<5490>|
---------+-----------+
(row = reference; col = test)

Overall Accuracy 0.81675
```

We have obtained **highest accuracy of 81.67%** till now by using **Naïve Bayes Algorithm**, and a **comprehensive** feature function which consists of **bigrams**, **negations**, and **subjectivity**. This model helps us classify movie reviews into positive and negative sentiments.

This code can be found in **classifyKaggle.py file**. It has various feature functions in it and commented lines of code which can be used to access these feature functions.

Cross-Validation:

As I have mentioned above that I have obtained highest accuracy using Comprehensive feature function, I performed Cross-Validation on the Naïve Bayes Model using that.

I wrote this code into **classifyKaggle.crossval.py file**. In this file, I have used just mentioned the **Comprehensive feature functions** with and without neutral words, and performed cross-validation.

Now lets look how I performed Cross-Validation for the above model!

1. **Model 1 – 5 Fold Cross Validation without neutrals:**
    - Naïve Bayes Algorithm
    - Feature function: Comprehensive Feature function with bigrams, negations and subjectivity without neutrals.
    - Bigram features = 1000

    We have obtained **average F-1 Score of 84.5%** by using 5-Fold Cross Validation.

```
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ python newfile.py 40000
[nltk_data] Downloading package punkt to /home/azureuser/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Read 156060 phrases, using 40000 random phrases
(['is', 'more', 'interesting', 'than', 'the', 'screenplay', 'which', 'lags', 'badly', 'in', 'the', 'middle', 'and', 'lurches', 'b
etween', 'not-very-funny', 'comedy', 'unconvincing', 'dramatics', 'and', 'some', 'last-minute', 'action', 'strongly', 'reminiscen
t', 'of', 'run', 'lola', 'run'], 'negative')
(['the', 'film', 'is', 'bright', 'and', 'flashy', 'in', 'all', 'the', 'right', 'ways'], 'positive')
(['both', 'a', 'beautifully'], 'positive')
(['does', "n't", 'really', 'care', 'about', 'the', 'thousands', 'of', 'americans', 'who', 'die', 'hideously'], 'negative')
(['a', 'powerful', 'sequel', 'and', 'one'], 'positive')
(['are', 'undeniably', 'touched'], 'positive')
(['a', 'serious', 'contender', 'for', 'the', 'title'], 'positive')
(['...', 'tara', 'reid', 'plays', 'a', 'college', 'journalist', 'but', 'she', 'looks', 'like', 'the', 'six-time', 'winner', 'of',
 'the', 'miss', 'hawaiian', 'tropic', 'pageant', 'so', 'i', 'do', "n't", 'know', 'what', 'she', "'s", 'doing', 'in', 'here', '...
'], 'negative')
(['what', 'it', 'promises', 'just', 'not', 'well', 'enough', 'to', 'recommend', 'it'], 'negative')
(['the', 'story', 'is', '--', 'forgive', 'me', '--', 'a', 'little', 'thin'], 'negative')
13429
Overall Accuracy 0.8120833333333334
Each fold size: 3920
Fold 0
Fold 1
Fold 2
Fold 3
Fold 4

Average Precision       Recall          F1      Per Label
negative              0.834      0.820      0.827
positive              0.850      0.862      0.856

Macro Average Precision Recall          F1      Over All Labels
              0.842      0.841      0.841

Label Counts {'negative': 8834, 'positive': 10770}
Micro Average Precision Recall          F1      Over All Labels
              0.843      0.843      0.843
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ [23~
```

```
(row = reference; col = test)

Each fold size: 3922
Fold 0
        Precision       Recall          F1
negative              0.845      0.812      0.828
positive              0.852      0.879      0.865
Fold 1
        Precision       Recall          F1
negative              0.848      0.815      0.831
positive              0.843      0.872      0.857
Fold 2
        Precision       Recall          F1
negative              0.805      0.813      0.809
positive              0.853      0.846      0.849
Fold 3
        Precision       Recall          F1
negative              0.836      0.816      0.826
positive              0.848      0.865      0.856
Fold 4
        Precision       Recall          F1
negative              0.847      0.832      0.840
positive              0.860      0.873      0.866

Average Precision       Recall          F1      Per Label
negative              0.836      0.818      0.827
positive              0.851      0.867      0.859

Macro Average Precision Recall          F1      Over All Labels
              0.844      0.842      0.843

Label Counts {'negative': 8711, 'positive': 10899}

Label weights [0.44421213666496684, 0.5557878633350332]
Micro Average Precision Recall          F1      Over All Labels
              0.845      0.845      0.845
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ ~
```

2. **Model 2: 5 Fold Cross Validation without neutrals:**
    - Naïve Bayes Algorithm
    - Feature function: Comprehensive Feature function with bigrams, negations and subjectivity without neutrals.
    - Bigram features = 250

We have obtained **average F-1 Score of 84.6%** by using 5-Fold Cross Validation.

```
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ python newfile.py 40000
[nltk_data] Downloading package punkt to /home/azureuser/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Read 156060 phrases, using 40000 random phrases
(['grand'], 'positive')
(['the', 'movie', 'tries', 'to', 'make', 'sense', 'of', 'its', 'title', 'character'], 'negative')
(['obscenely', 'bad'], 'negative')
(['to', 'its', 'own', 'detriment'], 'negative')
(['have', 'thought', 'possible'], 'positive')
(['we', 'hold', 'dear', 'about', 'cinema', 'only', 'now', 'it', "'s", 'begun', 'to', 'split', 'up', 'so', 'that
', 'it', 'can', 'do', 'even', 'more', 'damage'], 'negative')
(['some', 'very', 'good', 'acting', 'dialogue'], 'positive')
(['appearing', 'in', 'this', 'junk', 'that', "'s", 'tv', 'sitcom', 'material', 'at', 'best'], 'negative')
(['the', 'two', 'leads', 'delivering', 'oscar-caliber', 'performances'], 'positive')
(['better'], 'positive')
13411
Overall Accuracy 0.8205833333333333
Confusion Matrix:
          |    n    p |
          |    e    o |
          |    g    s |
          |    a    i |
          |    t    t |
          |    i    i |
          |    v    v |
          |    e    e |
----------+-----------+
negative |<4291>1093 |
positive | 1060<5556>|
----------+-----------+
(row = reference; col = test)

Each fold size: 3902
Fold 0
        Precision       Recall          F1
negative              0.852      0.818      0.835

(row = reference; col = test)

Each fold size: 3902
Fold 0
        Precision       Recall          F1
negative              0.852      0.818      0.835
positive              0.849      0.878      0.863
Fold 1
        Precision       Recall          F1
negative              0.828      0.812      0.820
positive              0.844      0.857      0.850
Fold 2
        Precision       Recall          F1
negative              0.848      0.835      0.841
positive              0.861      0.872      0.867
Fold 3
        Precision       Recall          F1
negative              0.818      0.819      0.818
positive              0.854      0.853      0.854
Fold 4
        Precision       Recall          F1
negative              0.838      0.838      0.838
positive              0.862      0.862      0.862

Average Precision       Recall          F1      Per Label
negative              0.837      0.824      0.830
positive              0.854      0.865      0.859

Macro Average Precision Recall          F1      Over All Labels
              0.845      0.845      0.845

Label Counts {'negative': 8787, 'positive': 10725}

Label weights [0.4503382533825338, 0.5496617466174661]
Micro Average Precision Recall          F1      Over All Labels
              0.846      0.847      0.846
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ ~
```

3. **Model 3: 10 Fold Cross Validation without neutrals:**
    - Naïve Bayes Algorithm
    - Feature function: Comprehensive Feature function with bigrams, negations and subjectivity without neutrals.
    - Bigram features = 250

We have obtained **average F-1 Score of 85.1%** by using 10-Fold Cross Validation.

```
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ python newfile.py 40000
[nltk_data] Downloading package punkt to /home/azureuser/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Read 156060 phrases, using 40000 random phrases
(['a', 'film', 'that', "'s", 'rarely', 'as', 'entertaining', 'as', 'it', 'could', 'have', 'been'], 'negative')
(['more', 'silly', 'than', 'scary'], 'negative')
(['decided', 'lack', 'of', 'spontaneity', 'in', 'its', 'execution', 'and', 'a', 'dearth', 'of', 'real', 'poigna
ncy', 'in', 'its', 'epiphanies'], 'negative')
(['covers', 'huge', 'heavy', 'topics', 'in', 'a', 'bland', 'surfacey', 'way', 'that', 'does', "n't", 'offer', '
any', 'insight', 'into', 'why', 'for', 'instance', 'good', 'things', 'happen', 'to', 'bad', 'people'], 'negativ
e')
(['shows', 'how', 'intolerance', 'has', 'the', 'power', 'to', 'deform', 'families', 'then', 'tear', 'them', 'ap
art'], 'positive')
(['ludicrous', 'attempt'], 'negative')
(['will', 'laugh', 'their', 'off', 'for', 'an', 'hour-and-a-half'], 'positive')
(['above', 'run-of-the-filth', 'gangster', 'flicks'], 'positive')
(['is', 'surprisingly', 'brilliant'], 'positive')
(['rather', 'dull', 'unimaginative'], 'negative')
13516
Overall Accuracy 0.8154166666666667
Confusion Matrix:
          |    n    p |
          |    e    o |
          |    g    s |
          |    a    i |
          |    t    t |
          |    i    i |
          |    v    v |
          |    e    e |
----------+-----------+
negative |<4228>1128 |
positive | 1087<5557>|
----------+-----------+
(row = reference; col = test)
```

```
13516
Overall Accuracy 0.8154166666666667
Confusion Matrix:
          |    n    p |
          |    e    o |
          |    g    s |
          |    a    i |
          |    t    t |
          |    i    i |
          |    v    v |
          |    e    e |
----------+-----------+
negative |<4228>1128 |
positive | 1087<5557>|
----------+-----------+
(row = reference; col = test)

Each fold size: 1954
Fold 0
          Precision       Recall          F1
negative              0.832      0.822      0.827
positive              0.846      0.855      0.851
Fold 1
          Precision       Recall          F1
negative              0.851      0.838      0.844
positive              0.865      0.875      0.870
Fold 2
          Precision       Recall          F1
negative              0.842      0.827      0.834
positive              0.864      0.877      0.871
Fold 3
          Precision       Recall          F1
negative              0.852      0.826      0.839
positive              0.853      0.876      0.864
Fold 4
          Precision       Recall          F1
negative              0.850      0.816      0.833
```

```
        Precision       Recall          F1
negative                0.850       0.816       0.833
positive                0.850       0.878       0.864
Fold 5
        Precision       Recall          F1
negative                0.866       0.837       0.851
positive                0.864       0.889       0.876
Fold 6
        Precision       Recall          F1
negative                0.851       0.825       0.838
positive                0.852       0.875       0.863
Fold 7
        Precision       Recall          F1
negative                0.827       0.796       0.811
positive                0.835       0.861       0.848
Fold 8
        Precision       Recall          F1
negative                0.841       0.825       0.833
positive                0.862       0.875       0.868
Fold 9
        Precision       Recall          F1
negative                0.833       0.839       0.836
positive                0.865       0.860       0.863

Average Precision       Recall          F1      Per Label
negative                0.844       0.825       0.835
positive                0.856       0.872       0.864

Macro Average Precision Recall          F1      Over All Labels
              0.850       0.849       0.849

Label Counts {'negative': 8726, 'positive': 10815}

Label weights [0.44654828309707795, 0.553451716902922]
Micro Average Precision Recall          F1      Over All Labels
              0.851       0.851       0.851
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$
```

By using **Cross-Validation**, I have obtained **highest F-1 Score** of **85.1%.**

I would like to mention that in my project, and after thorough analysis with various features and other factors, this is the highest F-1 Score that I have obtained. This is obtained by training **Naïve Bayes Algorithm** on 40000 reviews random sampled on train data. I have used a **comprehensive feature function** that consists of **bigrams, negations, and subjectivity**. This model is trained using **10-Fold Cross Validation**, and it classifies the movie reviews into positive and negative classes.

Experiments:

1. **Trigrams:**
    - 5-Fold Cross Validation
    - Naïve Bayes Algorithm
    - Feature function: Comprehensive Feature function with trigrams, negations and subjectivity without neutrals.
    - Trigram features = 250

We have obtained average **F-1 Score of 85%** by using 5-Fold Cross Validation.

```
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ python newfile.py 40000
[nltk_data] Downloading package punkt to /home/azureuser/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Read 156060 phrases, using 40000 random phrases
(['it', 'makes', 'you', 'feel', 'like', 'a', 'chump'], 'negative')
(['in', 'its', 'place', 'a', 'sweetness', 'clarity', 'and', 'emotional', 'openness', 'that', 'recalls', 'the', 'classics', 'of', 'early', 'ita
lian', 'neorealism'], 'positive')
(['full', 'of', 'grace'], 'positive')
(['is', 'the', 'edge', 'of', 'wild', 'lunatic', 'invention', 'that', 'we', 'associate', 'with', 'cage', "'s", 'best', 'acting'], 'positive')
(["'s", 'a', 'funny', 'little', 'movie', 'with', 'clever', 'dialogue', 'and', 'likeable', 'characters'], 'positive')
(['unsettling', 'to', 'watch', 'as', 'an', 'exploratory', 'medical', 'procedure', 'or', 'an', 'autopsy'], 'negative')
(['her', 'material', 'is', 'not', 'first-rate'], 'negative')
(['of', 'cookie-cutter', 'action', 'scenes'], 'negative')
(['modern-day', 'comedies'], 'positive')
(['if', 'there', 'was', 'ever', 'a', 'movie', 'where', 'the', 'upbeat', 'ending', 'feels', 'like', 'a', 'copout'], 'negative')
13353
Overall Accuracy 0.8221666666666667
Confusion Matrix:
          |    n    p |
          |    e    o |
          |    g    s |
          |    a    i |
          |    t    t |
          |    i    i |
          |    v    v |
          |    e    e |
----------+-----------+
negative |<4320>1078 |
positive | 1056<5546>|
----------+-----------+
(row = reference; col = test)

Each fold size: 3871
Fold 0
       Precision       Recall          F1
```

```
(row = reference; col = test)

Each fold size: 3871
Fold 0
        Precision       Recall          F1
negative              0.844      0.822      0.833
positive              0.850      0.869      0.860
Fold 1
        Precision       Recall          F1
negative              0.841      0.829      0.835
positive              0.862      0.871      0.867
Fold 2
        Precision       Recall          F1
negative              0.840      0.827      0.833
positive              0.853      0.865      0.859
Fold 3
        Precision       Recall          F1
negative              0.853      0.836      0.844
positive              0.857      0.872      0.865
Fold 4
        Precision       Recall          F1
negative              0.837      0.824      0.831
positive              0.857      0.868      0.862

Average Precision       Recall          F1      Per Label
negative              0.843      0.828      0.835
positive              0.856      0.869      0.862

Macro Average Precision Recall          F1      Over All Labels
              0.849      0.848      0.849

Label Counts {'negative': 8725, 'positive': 10632}

Label weights [0.4507413338843829, 0.5492586661156171]
Micro Average Precision Recall          F1      Over All Labels
              0.850      0.850      0.850
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ [reconnecting terminal]
```

2. **Decision Tree:**
    - I have trained this model upon 20000 movie reviews using **Decision Tree Classifier**, and obtained accuracy of **70.75%**

```
(azureml_py38) azureuser@rpatchal1:~/cloudfiles/code/Users/rpatchal/project$ python svmfile.py 20000
[nltk_data] Downloading package punkt to /home/azureuser/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Read 156060 phrases, using 20000 random phrases
(['a', 'film', 'that', 'loses', 'sight', 'of', 'its', 'own', 'story'], 'negative')
(['take', 'us', 'to', 'that', 'elusive', 'lovely', 'place', 'where', 'we', 'suspend', 'our', 'disbelief'], 'positive')
(['racist', 'portraits'], 'negative')
(['with', 'humor', 'warmth', 'and', 'intelligence', 'captures', 'a', 'life', 'interestingly', 'lived'], 'positive')
(['a', 'mesmerizing', 'cinematic', 'poem', 'from', 'the', 'first', 'frame', 'to', 'the', 'last'], 'positive')
(['that', "'s", 'critic-proof', 'simply', 'because', 'it', 'aims', 'so', 'low'], 'negative')
(["'s", 'a', 'smartly', 'directed', 'grown-up', 'film', 'of', 'ideas'], 'positive')
(['moments', 'of', 'hilarity'], 'positive')
(['banged', 'their', 'brains', 'into', 'the', 'ground', 'so', 'frequently'], 'negative')
(['big', 'splash'], 'positive')
10537
Overall Accuracy 0.7075
Confusion Matrix:
          |    n    p |
          |    e    o |
          |    g    s |
          |    a    i |
          |    t    t |
          |    i    i |
          |    v    v |
          |    e    e |
----------+-----------+
negative |<1549>1083 |
positive |  672<2696>|
----------+-----------+
(row = reference; col = test)
```

I have performed the subsequent experiments upon **20000 movie reviews**.

3. **Classification into three sentiments (pos, neg, neu) for 20000 movie reviews :** I considered all three sentiments in classification task, achieving an accuracy of **63.3%** when train-test split is 70%. Subsequently, when the train-test split is 90%, the accuracy slightly improved to **63.5%.**

4. **Classification into two sentiments (pos, neg) for 20000 movie reviews :** We have achieved an accuracy of **76.5%**

5. **Including stop words:** By including stop words (common words like "the", "is", "and", etc.), the accuracy increased to **77.9%.** Stop words can contain important contextual information and their inclusion might aid in improving classification performance.

6. **Using negation feature function and removed stop words:** Despite removing stop words, the addition of negation features contributed to maintaining a relatively high accuracy of **77.1%.** Negation features help capture changes in sentiment caused by phrases like "not good" or "didn't like".

7. **Used negation feature function and kept stop words:** Keeping stop words while including negation features yielded an accuracy of **80%.** This suggests that stop words might provide valuable contextual information that enhances sentiment classification.

8. **Used bigram feature function with PMI:** Introducing bigram features with PMI, which capture word pairs, led to an accuracy of **77.55%.** Bigrams can capture more complex relationships between words and might provide additional context for sentiment analysis.

9. **Used bigrams with chi-squared measure:** Employing chi-squared measure with bigrams resulted in an accuracy of **77.05%.** This statistical measure helps identify significant associations between bigrams and sentiments.

10. **Used bigrams with raw frequency measure:** Utilizing raw frequency measure with bigrams achieved an accuracy of **76.65%.** Raw frequency measures the frequency of occurrence of bigrams without considering their association with sentiments.

11. **Included Subjectivity Lexicon (SL) Features without removing stopwords:** Incorporating Subjectivity Lexicon (SL) features without removing stopwords yielded an accuracy of **80.15%.** SL features capture the subjectivity and polarity of words, enhancing sentiment analysis.

12. **Included SL Features after removing stopwords:** After removing stopwords, the accuracy slightly increased to **80.2%.** This suggests that removing stopwords did not significantly impact the effectiveness of SL features.

13. **Using Negations and bigrams features together without removing stopwords:** We have achieved an accuracy of **78.85%.** This combination helps capture negated instances and complex word relationships simultaneously.

14. **Using Negations and bigrams features together keeping stopwords:** Keeping stopwords we achieved accuracy of **80.6%.** This indicates that stopwords might contain valuable information for sentiment analysis, even in the presence of negation features.

15. **Using Transformers – BERT:** Upon using BERT Model to fine-tune it on entire train dataset, I have obtained F-1 Score of **67.11%**

16. **Using Transformers – RoBERTa:** Upon using RoBERTa Model to fine-tune it on entire train dataset, I have obtained F-1 Score of **68.56%**

17. **Using Transformers – DistilBERT:** Upon using DistilBERT Model to fine-tune it on entire train dataset, I have obtained F-1 Score of **62.67%**

Finally, I would like to conclude my report by saying that the best model that I have trained is Naïve Bayes model using comprehensive feature functions upon 40000 movie reviews by 10-Fold Cross Validation, which achieved highest accuracy (F-1) of 85.01%

I have stored the predictions into predictions.csv file in corpus folder.