A. How to generate features and output files:

1. Groom original features
2. Generate 2 features ----- Emoticon and Bigram

Groom Original Features:

----

1. Use script to delete some of the possible unrelated features (delete them will increase accuracy).
   Each time just delete one feature to test.
2. Classify which features are unrelated to Naive Bayes and J48 respectively.
3. Use script in /p2-code/Code/1. DeleteOriFeature/deleteOriFeature.py to delete all the unrelated features to Naive Bayes and J48 respectively
4. Finally Naive Bayes gets 29 features and J48 gets 25 features from original features.
   and store the result into /p2-code/OutputFiles/DeleteUnrelatedFeature/test_J48.arff,
   /p2-code/OutputFiles/DeleteUnrelatedFeature/test_NB.arff,
   need manually delete head attributes line.

4.1 Format in test_J48.arff:

...(head attributes lines)

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,?

0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,?

0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,?

4.2 Format in test_NB.arff:

...(head attributes lines)

674277485807140864,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,?

629328699230306304,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,?

672700570013212672,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,?

--

Add Emoticon Features:

--

1. Set a positive emoticon set and a negative emoticon set.

negative_emoticons = {":-(", ":(", ":-|", ";-(", ";-<", "|-{"}

positive_emoticons = {":-)", ":)", ":o)", ":-}", ";-}", ":->", ";-)",":)"}

2. Use script in /p2-code/Code/2. AddEmoticonFeature/analysisTweetEmoticon.py to analyze each tweet:

2.1. If a tweet contains a postive emoticon --> set emoticon feature value as 2.

2.2. If a tweet contains a negative emoticon --> set emoticon feature value as 0.

2.3. else --> set emoticon feature value as 1.

3. Store the analysis result in a tempary file from test-tweets.txt:

/p2-code/OutputFiles/EmoticonFeature/test-tweets-Emo.txt

Format in tempary file is as follow :

1

1

1

1

1

....

4. Use script in /p2-code/Code/2. AddEmoticonFeature/addEmoticonFeature.py to add emoticon feature values from tempary files to

/p2-code/OutputFiles/EmoticonFeature/testEmo_J48.arff
/p2-code/OutputFiles/EmoticonFeature/testEmo_NB.arff

4.1 Format in testEmo_J48.arff is as follow:

...(head attributes lines)
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?
0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?
0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?

4.1 Format in testEmo_NB.arff is as follow:

...(head attributes lines)

674277485807140864,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?

629328699230306304,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?

672700570013212672,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?

Add Bigram Features:

----

1. Use script in /p2-code/Code/3. AddBigramFeature/generatePos_NegText.py to classify original tweet into different classes of sentiment.
   Generate three files ---- posTweets.txt, negTweets.txt, neuTweets.txt

2. Use above three files to calculate 100 highest freqency bigrams for each classes of sentiment.
   Then for each tweet in test-tweets.txt:

   2.1 Generate its own bigrams.

2.2 For each its own bigram, find if there is a class contains that bigram as well.

2.2.1 If only one sentiment class has that bigram ---- this tweet belongs to that sentiment class.

2.2.2 If multiple sentient classes have that bigram --- choose the class which has the higher frequency of that bigram.

2.3 If multiple its own bigram can be found in bigram sets, accumulate the frequency,
and then find which sentiment class have the highest
accumulated frequency ---- this tweet belongs to that sentiment class.

Note: If a tweet classify into positive sentiment from bigram sets --> set bigram feature value as 2.
If a tweet classify into negative sentiment from bigram sets --> set bigram feature value as 0.
else --> set bigram feature value as 1.

3. Store the analysis result in a tempary file test_bigram.txt:

/p2-code/OutputFiles/BigramFeature/testBigram.txt

Format in tempary file is as follow :

1

1

0

1

1

2

....

4. Use script in /p2-code/Code/3. AddBigramFeature/addBigramFeature.py to add bigram feature values from tempary files to

/p2-code/OutputFiles/BigramFeature/testEmo_J48.arff

/p2-code/OutputFiles/BigramFeature/testEmo_NB.arff

4.1 Format in testBigram_J48.arff is as follow:

...(head attributes lines)

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?

0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?

0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,?

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,2,?

4.1 Format in testBigram_NB.arff is as follow:

...(head attributes lines)

674277485807140864,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?

629328699230306304,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?

672700570013212672,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,?

802339460633559040,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?

802345702655344640,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,?

640741434589474816,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,?

Add Emoticon andBigram Features:

Combine the above two feature and generate the output file:

/p2-code/OutputSubmit/Emoticon_Bigram_J48.txt

/p2-code/OutputSubmit/Emoticon_Bigram_NB.txt

/p2-code/OutputSubmit/Emoticon_Bigram_Balanced_NB.txt ---> after using filter banlance data's result