Project on Sentimental Analysis

Table of Contents

1	Abstract
2	Introduction
3	Performance results of Naive Bayes model
4	Most useful words for predicting sentiment
5	Performance results of Rule-based model
6	Improvising Rule-based system
7	Error Analysis for rule-based system (test-dictionary method) for Nokia data
8	Conclusion
9	Appendices

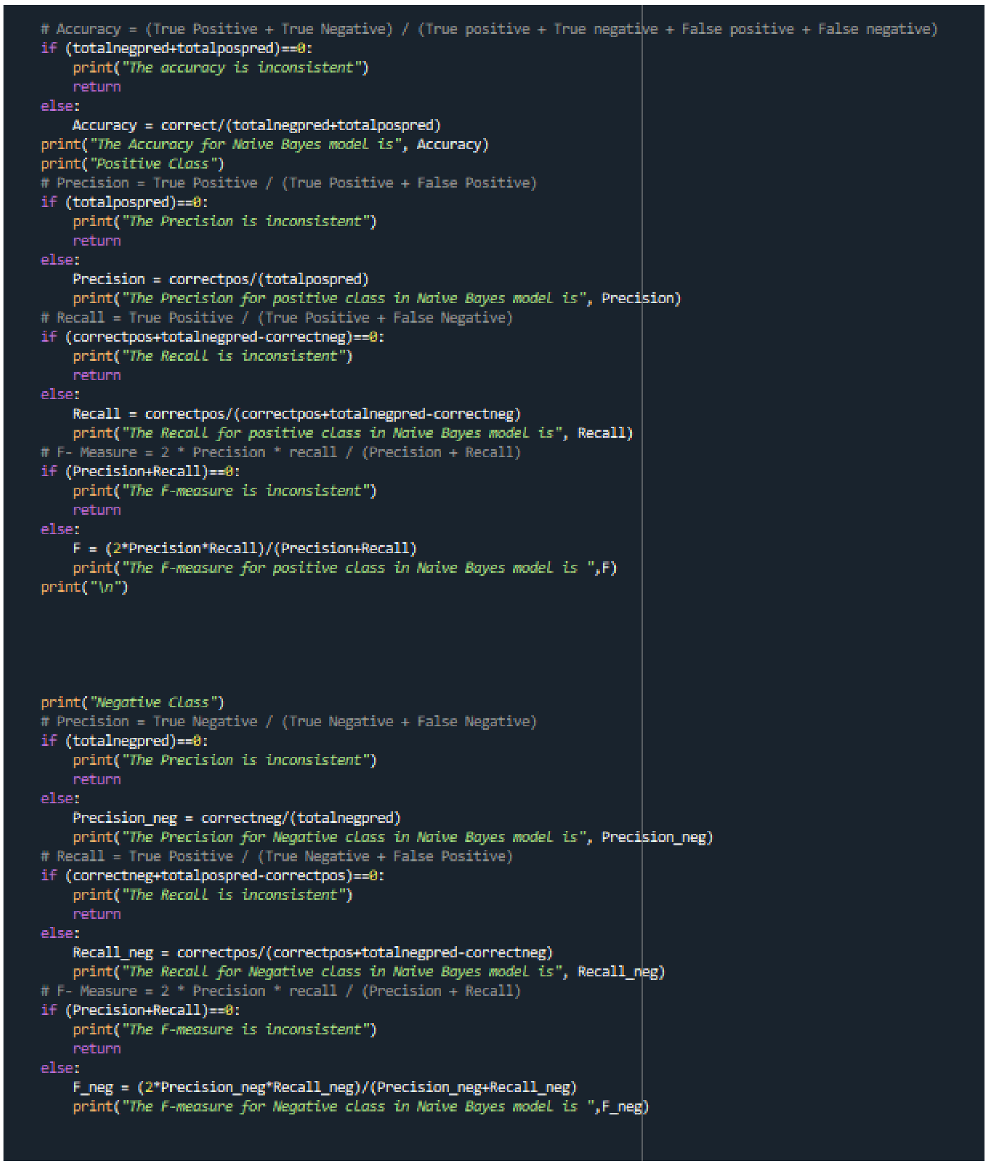
1. Abstract

The report corresponds to the statistical and rule-based approaches to text classification and the difference in results when improvising the rule-based approach. Further, all the most common errors in the sentiment prediction analysis are discussed to improve results.

2. Introduction:

Bayesian text classification is a commonly used text classification model which has its strengths and limitations. The rule-based approach is data-specific and cannot be applied to all types of data whereas the statistical approach is generic as it works based on probability, and the more versed the training, the more accurate the results.

3. The performance results of Naive bayes model



```python
# Accuracy = (True Positive + True Negative) / (True positive + True negative + False positive + False negative)
if (totalnegpred+totalpospred)==0:
    print("The accuracy is inconsistent")
    return
else:
    Accuracy = correct/(totalnegpred+totalpospred)
print("The Accuracy for Naive Bayes model is", Accuracy)
print("Positive Class")
# Precision = True Positive / (True Positive + False Positive)
if (totalpospred)==0:
    print("The Precision is inconsistent")
    return
else:
    Precision = correctpos/(totalpospred)
    print("The Precision for positive class in Naive Bayes model is", Precision)
# Recall = True Positive / (True Positive + False Negative)
if (correctpos+totalnegpred-correctneg)==0:
    print("The Recall is inconsistent")
    return
else:
    Recall = correctpos/(correctpos+totalnegpred-correctneg)
    print("The Recall for positive class in Naive Bayes model is", Recall)
# F- Measure = 2 * Precision * recall / (Precision + Recall)
if (Precision+Recall)==0:
    print("The F-measure is inconsistent")
    return
else:
    F = (2*Precision*Recall)/(Precision+Recall)
    print("The F-measure for positive class in Naive Bayes model is ",F)
print("\n")




print("Negative Class")
# Precision = True Negative / (True Negative + False Negative)
if (totalnegpred)==0:
    print("The Precision is inconsistent")
    return
else:
    Precision_neg = correctneg/(totalnegpred)
    print("The Precision for Negative class in Naive Bayes model is", Precision_neg)
# Recall = True Positive / (True Negative + False Positive)
if (correctneg+totalpospred-correctpos)==0:
    print("The Recall is inconsistent")
    return
else:
    Recall_neg = correctpos/(correctpos+totalnegpred-correctneg)
    print("The Recall for Negative class in Naive Bayes model is", Recall_neg)
# F- Measure = 2 * Precision * recall / (Precision + Recall)
if (Precision+Recall)==0:
    print("The F-measure is inconsistent")
    return
else:
    F_neg = (2*Precision_neg*Recall_neg)/(Precision_neg+Recall_neg)
    print("The F-measure for Negative class in Naive Bayes model is ",F_neg)
```

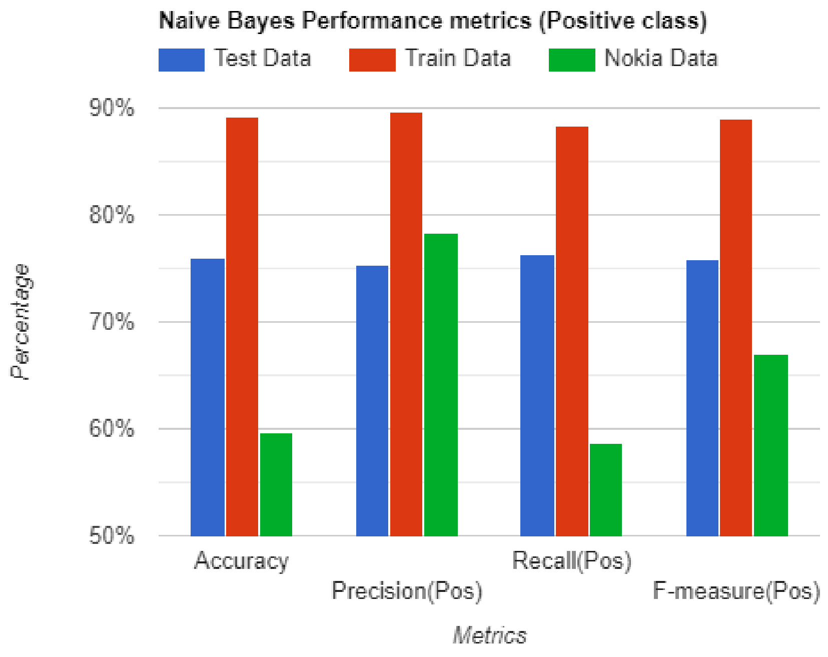
Results for Test Data

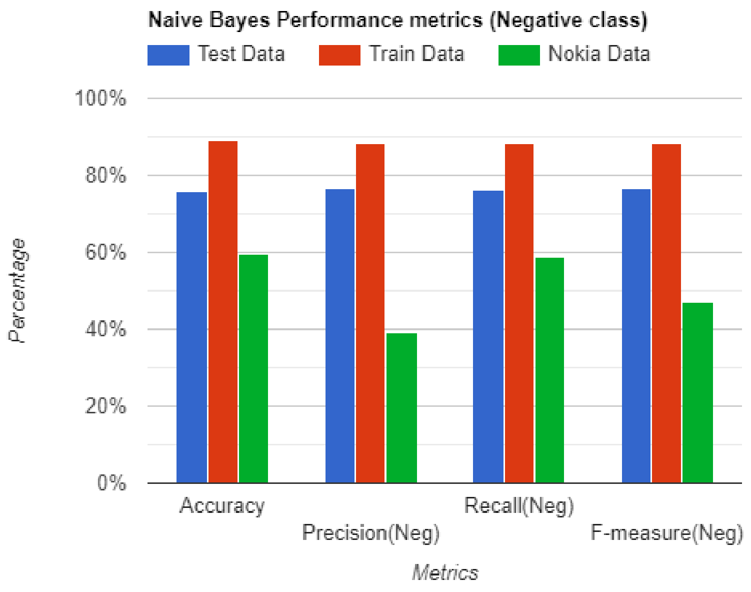
Metrics	Values
Accuracy	0.7602611940298507
Precision (Positive class)	0.7546468401486989
Recall (Positive class)	0.7645951035781544
F- measure (Positive class)	0.7595884003741816
Precision (Negative class)	0.7659176029962547
Recall (Negative class)	0.7645951035781544
F- measure (Negative class)	0.765255781908508

On applying the naive bayes theorem we can observe an accuracy of about 77% on test data, provided only 10% of overall data is considered.

3.1 Results of Naive bayes on Training and Nokia Data

Metrics	Train Data	Nokia Data
Accuracy	0.8916692732770305	0.5977443609022557
Precision (Positive class)	0.897989417989418	0.7841726618705036
Recall (Positive class)	0.8839583333333333	0.5860215053763441
F- measure (Positive class)	0.8909186351706037	0.6707692307692308
Precision (Negative class)	0.8855322646937936	0.3937007874015748
Recall (Negative class)	0.8839583333333333	0.5860215053763441
F- measure (Negative class)	0.8847445990211673	0.4709847470077345





There is a considerable amount of difference between the accuracy of the trained data and the Nokia product review data. This is due to the system being overfitted on train data given by rotten tomatoes, a movie review website and is then used to analyse the sentiments on mobile phones. Both are from different domains. The model could have worked properly if the training data was more generic or related to the field of study.

4. The Most Useful words for predicting sentiment

4.1 NEGATIVE:

'stupid', 'badly', 'mediocre', 'generic', 'unfunny', 'routine', 'poorly', 'waste', 'mindless', 'boring', 'pointless', 'disguise', 'tiresome', 'unless', 'stale', 'shoot', 'bore', 'meandering', 'annoying', 'product', 'plodding', 'dull', 'pass', 'apparently', 'chan', 'fatal', 'pinocchio', 'numbers', 'supposed', 'ill', 'junk', 'retread', 'wasted', 'offensive', 'ballistic', 'lousy', 'horrible', 'flat', 'inept', 'embarrassment', 'amateurish', 'banal', 'ludicrous', 'harvard', 'bag', 'kung', 'incoherent', 'stiff', 'crap', 'lifeless'

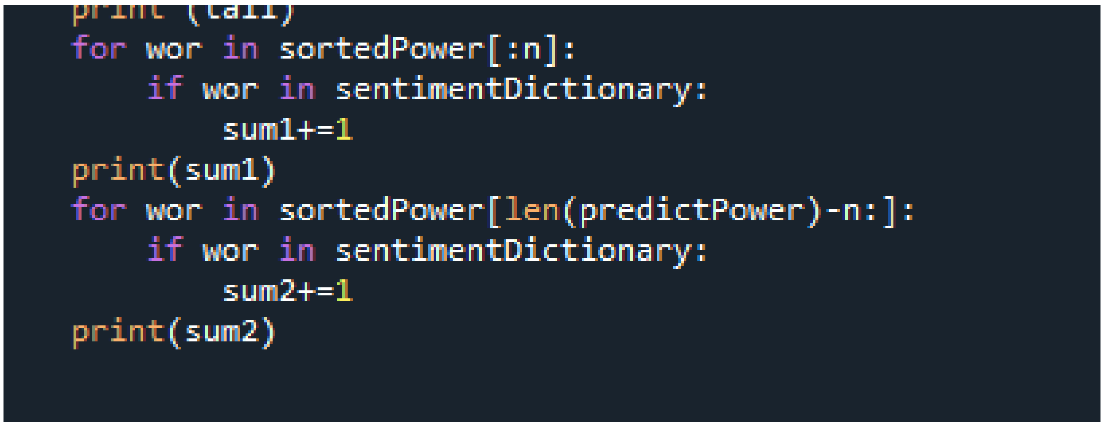
Only 54% (27/50 words) of negative useful words are found in Sentiment dictionary

4.2 POSITIVE:

'format', 'tour', 'intimate', 'stirring', 'transcends', 'breathtaking', 'sly', 'twisted', 'grown', 'ingenious', 'record', 'subversive', 'sadness', 'playful', 'startling', 'moviemaking', 'spare', 'timely', 'jealousy', 'smarter', 'warm', 'unexpected', 'pulls', 'tender', 'iranian', 'lively', 'captivating', 'sides', 'polished', 'respect', 'vividly', 'captures', 'detailed', 'wry', 'heartwarming', 'chilling', 'wonderfully', 'wonderful', 'powerful', 'realistic', 'imax', 'gem', 'mesmerizing', 'refreshingly', 'riveting', 'refreshing', 'delightful', 'inventive', 'provides', 'engrossing'

Only 60% (30/50 words) of positive useful words are found in Sentiment dictionary

Code snippet to find number of most useful words in sentiment dictionary.



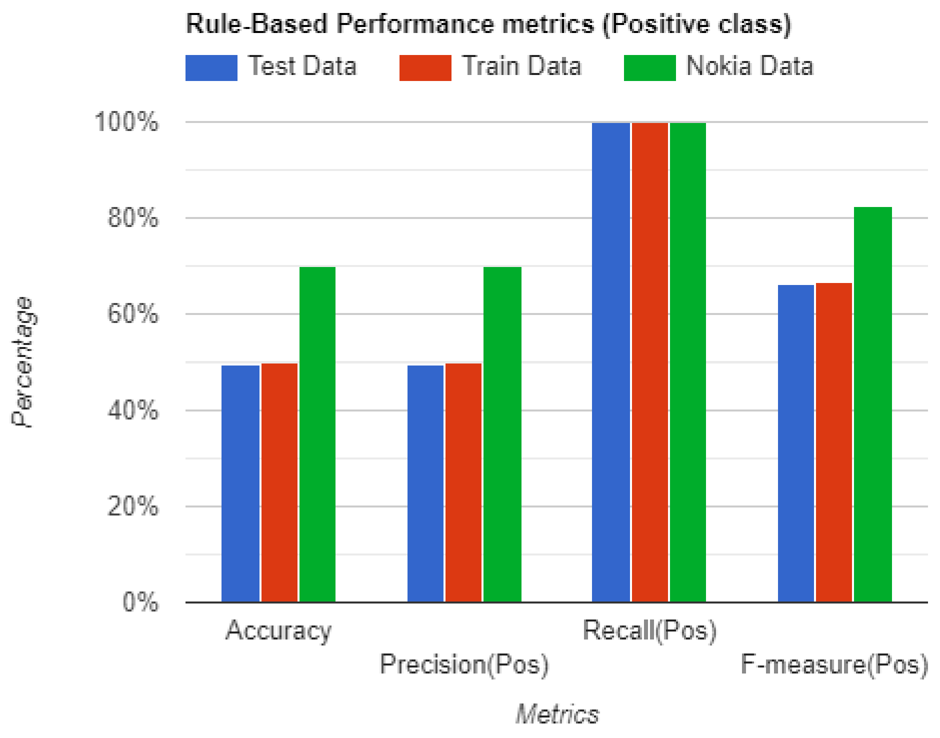
```
for wor in sortedPower[:n]:
    if wor in sentimentDictionary:
        sum1+=1
print(sum1)
for wor in sortedPower[len(predictPower)-n:]:
    if wor in sentimentDictionary:
        sum2+=1
print(sum2)
```

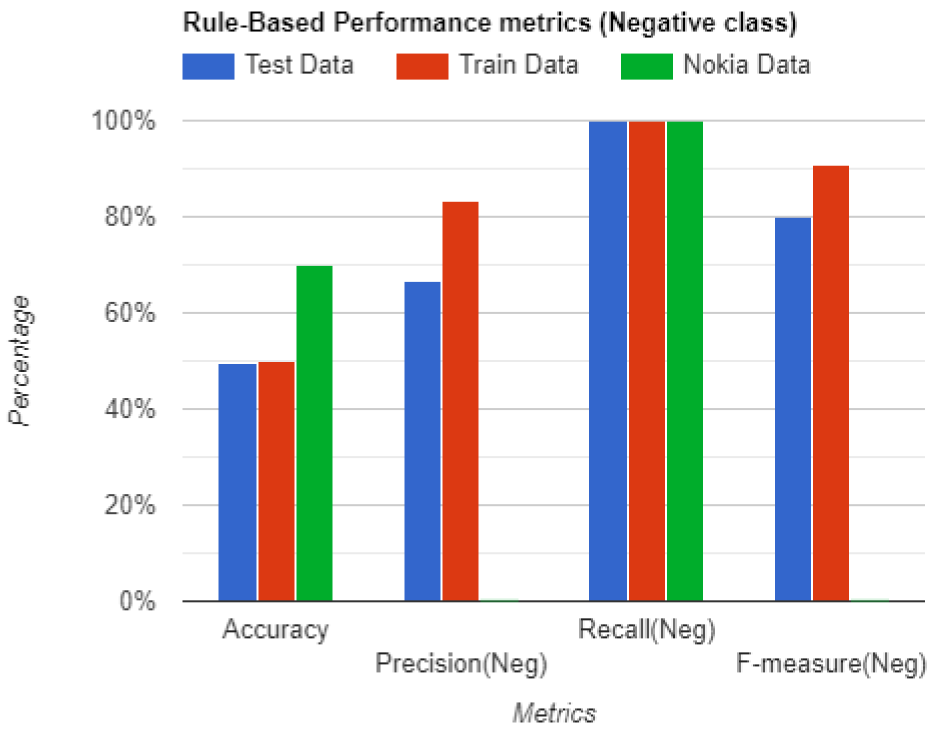
Not all the words selected by the model are good sentiment terms; some generic words are also considered negative or positive terms. This system can be further improved by handling the neutral words like “generic”, “routine”, “product”, “tour”, etc.

5. Performance results of Rule-based model

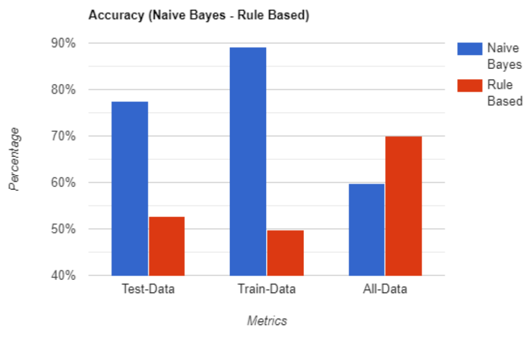
Metrics	Test data	Train Data	All/Nokia data
Accuracy	0.4962686567164179	0.502137420498384	0.6992481203007519
Precision (Positive class)	0.49579045837231056	0.5013065746838089	0.6992481203007519
Recall (Positive class)	0.9981167608286252	0.9991666666666666	1.0
F- measure (Positive class)	0.6625	0.6676411220157304	0.8230088495575222
Precision (Negative class)	0.6666666666666666	0.8333333333333334	Cannot be determined
Recall (Negative class)	0.9981167608286252	0.9991666666666666	1.0
F- measure (Negative class)	0.7993966817496229	0.9087463998787328	Cannot be determined

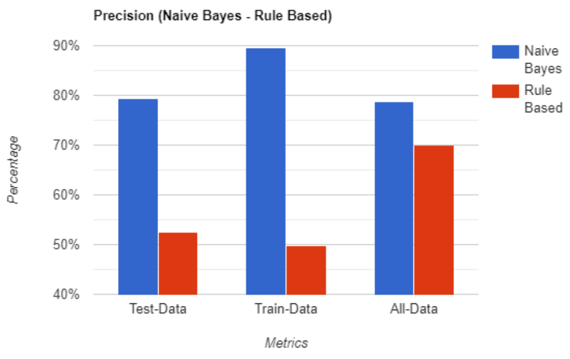
The Total Negative prediction for Nokia data for the above model Is 0, thus the precision and F-measure cannot be calculated. This inconsistency is due to the threshold value passed to the system for classification.

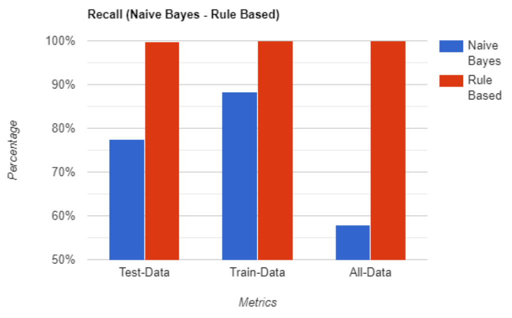


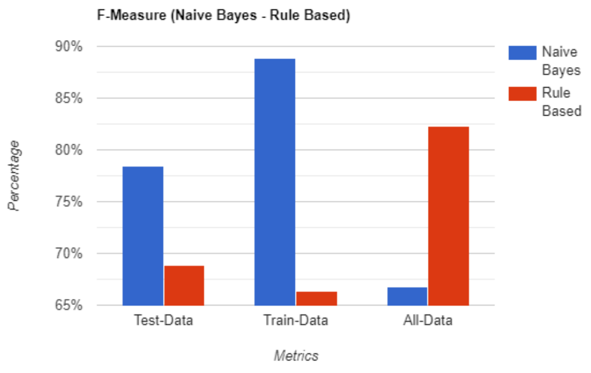


5.1 Naive Bayes Vs Rule based comparison









The above charts represent Naïve bayes vs Rule based comparison for positive class.

Under the observation between the rule-based and statistical approaches, we can see that the statistical approach seems to have better accuracy overall, but when it comes to Nokia Reviews data, the rule-based approach has better accuracy. This is because the dataset seems to have data that matches very well with specific positive and negative words, and also because the test data content trained is not enough for the Naive Bayes algorithm to achieve a higher accuracy under the Nokia data.

6 Improvising the Rule-based system

The rule-based system can be improved by adding cases to the train system based on rules like Negation, Diminisher, Intensifier, etc.

6.1 Implementation

1. I have created a list of negation, intensifier and diminisher words and modified the score value as follows.

If **Negation word + Positive word** then **Score = -3**

If **Negation word + Negative word** then **Score = +3**

2. Similarly I have created an Intensifier rule and Diminisher rule with the following formulae.

If **Intensifying word + Positive word** then **Score = +4**

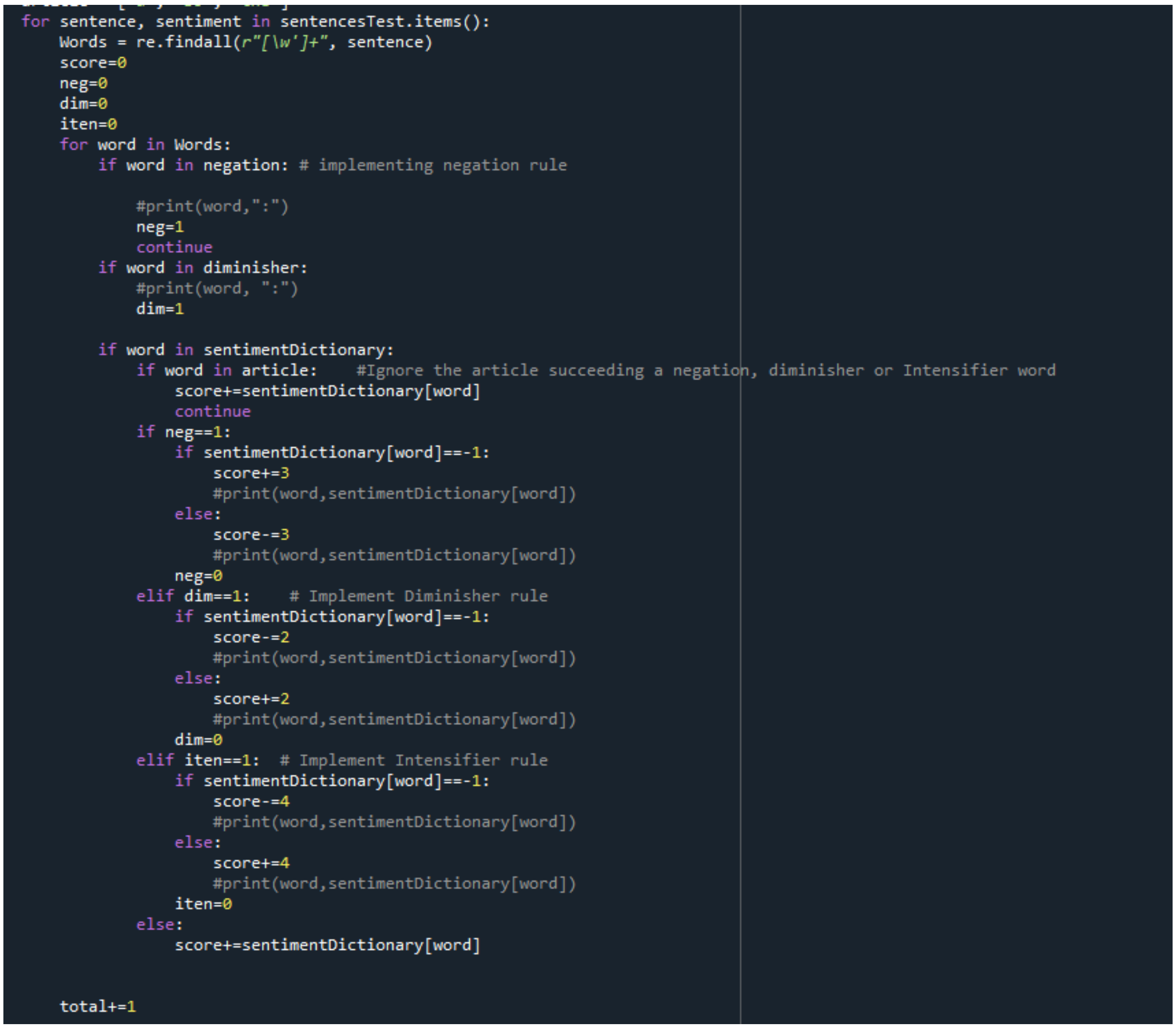
If **Intensifying word + Negative word** then **Score = -4**

If **Diminishing word + Positive word** then **Score = +2**

If **Diminishing word + Negative word** then **Score = -2**

3. I have also added a rule to ignore any articles following the Intensifier, Diminisher or Negation word.

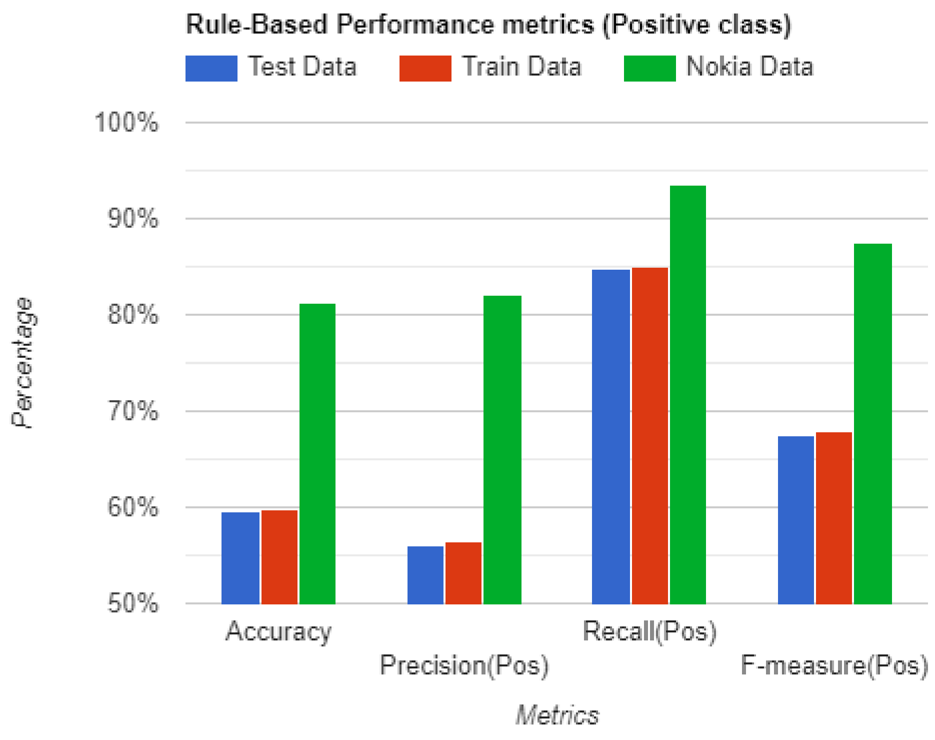
4. Improved the threshold value to 0 for better prediction.

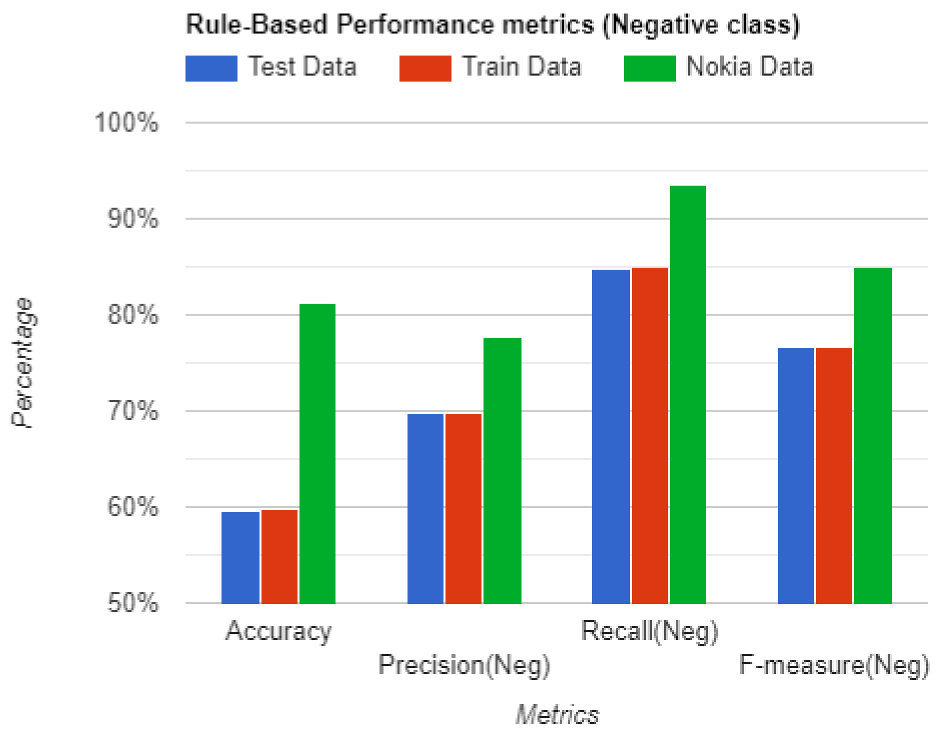


```
for sentence, sentiment in sentencesTest.items():
    Words = re.findall(r"[\w']+", sentence)
    score=0
    neg=0
    dim=0
    iten=0
    for word in Words:
        if word in negation: # implementing negation rule

            #print(word,":")
            neg=1
            continue
        if word in diminisher:
            #print(word, ":")
            dim=1

        if word in sentimentDictionary:
            if word in article:    #Ignore the article succeeding a negation, diminisher or Intensifier word
                score+=sentimentDictionary[word]
                continue
            if neg==1:
                if sentimentDictionary[word]==-1:
                    score+=3
                    #print(word,sentimentDictionary[word])
                else:
                    score-=3
                    #print(word,sentimentDictionary[word])
                neg=0
            elif dim==1:    # Implement Diminisher rule
                if sentimentDictionary[word]==-1:
                    score-=2
                    #print(word,sentimentDictionary[word])
                else:
                    score+=2
                    #print(word,sentimentDictionary[word])
                dim=0
            elif iten==1:  # Implement Intensifier rule
                if sentimentDictionary[word]==-1:
                    score-=4
                    #print(word,sentimentDictionary[word])
                else:
                    score+=4
                    #print(word,sentimentDictionary[word])
                iten=0
            else:
                score+=sentimentDictionary[word]


    total+=1
```

6.2 The Results of the rule-based system after modification is given by..

Metrics	Test data	Train data	All data
Accuracy	0.5951492537313433	0.5984777395474924	0.8120300751879699
Precision (Positive class)	0.5603985056039851	0.5657840011091085	0.8207547169811321
Recall (Positive class)	0.847457627118644	0.8502083333333333	0.9354838709677419
F- measure (Positive class)	0.6746626686656672	0.6794306168317655	0.8743718592964823
Precision (Negative class)	0.6988847583643123	0.6976450798990749	0.7777777777777778
Recall (Negative class)	0.847457627118644	0.8502083333333333	0.9354838709677419
F- measure (Negative class)	0.7660337380816559	0.7664080533317698	0.8493723849372385





After above implementation, the True positive and True negative count has increased resulting in better accuracy which is clearly reflected in the above graphs.

7. Error Analysis for rule-based system (test-dictionary method) for Nokia data

In most of the cases, a negative sentence is classified as positive.

1. Sentences with contradicting meaning are not classified perfectly.

Example : **“**......... , perfect size , **but** a tid bit quieter ...**”**

2. The system was not able to detect negation terms.

Example: **“**...... **not** wowable good”.

3. The Data-set is limited and does not cover all negative words.

Example : **“poor** visibility on the keys .**”**

“......... reception is **unpredictable”**

4. Some words mean differently in different contexts which are mispredicted.

Example : **“**...... setting is **loud** ! **”**

The word **“*loud*”** refers to negative sentiment but is classified as positive.

5. Incomplete or sentences with no sentiment terms are not predicted by the system

Example : “the volume.”

“its quiet.”

“some features in the user interface”

6. The sentences with a sarcastic meaning are difficult to classify.

Example: “it also has a stock-tracking app built in (also useless) and ……..”

7. The system does not handle the **intensifier** and **diminisher** words.

Example: “..... ***little*** inconvenient.”

“ ***huge*** disappointment …. ”

On addressing the above discussed scenarios, the system can be improved for better classification

8. Conclusion

The Statistical approach is better suitable for text classification than rule-based approach provided the training data is more generic, wide, and related to the domain.

9. Appendices

Error Analysis in section 7 is done using the following data

9.1 Nokia data errors

ERROR (neg classed as pos 0.00):the headset that comes with the phone has good sound volume but it hurts the ears like you cannot imagine !

ERROR (neg classed as pos 0.00):the phone comes with okay ringtones , some decent backgrounds / screensavers , but the phone has very little memory (mine had 230kb as it arrived from amazon , so you do n't have too many options on what you can put on there) .

ERROR (neg classed as pos 0.00):great battery life , perfect size , but a tid bit quieter than i would like .

ERROR (neg classed as pos 0.00):radio is awsome , but it does n't work unless you have the earpice in .

ERROR (neg classed as pos 0.00):this is a very nice phone , but there is no warranty on it .

ERROR (neg classed as pos 0.00):i thought this was the ultimate phone when it comes to basic features , but i was dissapointed when i saw that it was only a gsm comaptible phone .

ERROR (neg classed as pos 0.00):the gprs connection is sometimes slow , and writing instant messages with the included aol instant messenger software is a pain , but the other t-zones applications are quite useful .

ERROR (neg classed as pos 0.00):

ERROR (neg classed as pos 0.00):after several years of torture in the hands of at&t customer service i am delighted to drop them , and look forward to august 2004 when i will convert our other 3 family-phones from at&t to t-mobile !

ERROR (neg classed as pos 0.00):the day finally arrived when i was sure i 'd leave sprint .

ERROR (neg classed as pos 0.00):after years with that carrier 's expensive plans and horrible customer service , portability seemed heaven-sent .

ERROR (neg classed as pos 0.00):i spent hours setting up the stations (accepts about 13-14 , i believe) , though the reception is unpredictable .

ERROR (neg classed as pos 0.00):the colors on the screen are not as crisp as i 'd have liked them to be .

ERROR (neg classed as pos 0.00):the backlight on the phone goes off way too quickly (dangerous when you 're driving at night) , and there 's no way to change this --

ERROR (neg classed as pos 0.00):the buttons on the phone are small , even for my small fingertips , but you get used to them rather quickly .

ERROR (neg classed as pos 0.00):the menu options are uncreative , as you can 't see a full screen of menu items to pick from ; you have to scroll up and down to find what you 're looking for (yes , this is minor , but not when you 're trying to keep your eye on the road !) .

ERROR (neg classed as pos 0.00):the volume .

ERROR (neg classed as pos 0.00):i have excellent hearing but the volume level on this phone is especially quiet .

ERROR (neg classed as pos 0.00):i assumed they had exceptional service , but their reception in my area (los angeles) is horrendous .

ERROR (neg classed as pos 0.00):also , their t-zones , although cheap ($ 4.99 / mo.) never works .

ERROR (neg classed as pos 0.00):since i received the phone , i spent countless hours on the phone with customer service reps who promised t-zones would work " in 24 hours " .

ERROR (neg classed as pos 0.00):i must have heard this about a dozen times over the span of 2 weeks , when t-zones never worked .

ERROR (neg classed as pos 0.00):apparently , t-mobile is heavily back-logged and can 't keep up with demands .

ERROR (neg classed as pos 0.00):however , the calls constantly drop in my area and i experince mega-static , to the point where i 'd have to dial numbers 6-7 times to get a clear line .

ERROR (neg classed as pos 0.00):the volume level of the phone is not all that good .

ERROR (neg classed as pos 0.00):some of the higher pitched rings are very easy to hear , but not easy to listen to .

ERROR (neg classed as pos 0.00):the more subtle tones that were included with the phone are hard to hear at times .

ERROR (neg classed as pos 0.00):the vibration is not top .

ERROR (neg classed as pos 0.00):i am bored with the silver look .

ERROR (neg classed as pos 0.00):they are not wowable good .

ERROR (neg classed as pos 0.00):it also does n't have voice activated dialing . big minus ! !

ERROR (neg classed as pos 0.00):we got two phones for the t-mobile family plan (indeed very easy to switch to the family plan) and one came with a broken headphone (can hear but cannot be heard) and one died today after less than two weeks of use .

ERROR (neg classed as pos 0.00):when i got it home , i discovered that the menu options for sending messages via either text or email simply were n't there :

ERROR (neg classed as pos 0.00):negative : impossibly tiny and difficult to operate , barely visible , power button .

ERROR (neg classed as pos 0.00):poor visibility on the keys .

ERROR (neg classed as pos 0.00):only con i can think of is no camera ... .

ERROR (neg classed as pos 0.00):only one complaint about the speakerphone , you can only activate the speakerphone feature once the person you are calling answers the phone , not while it is ringing .

ERROR (neg classed as pos 0.00):the only problem i had was a small glitch with t-mobile .

ERROR (neg classed as pos 0.00):the phone has a few minor inconveniences , but only because it lacks those features , bluetooth and high spend internet , but there are very few problems with things that you expected this phone to do .

ERROR (neg classed as pos 0.00):ring tones only come with crazy songs and annoying rings , there is only one ring that sounds close to a regular ring .

ERROR (neg classed as pos 0.00):the screen is easily scratched but if you have the warranty you should be able to swap it out .

ERROR (neg classed as pos 0.00):games kind of stink and you cant download them you have to get the link cable to get additional games .

ERROR (neg classed as pos 0.00):buttons do seem a little sticky tho .. and the hands free kits ' connector might be a problem ..

ERROR (neg classed as pos 0.00):the keys are close together , and the layout is a bit funky in relation to standard rectangular layout keypads , but it 's not too weird .

ERROR (neg classed as pos 0.00):the fact that the " 0 " key is the space key for text input is a bit confusing , as many phones use the " # " key instead .

ERROR (neg classed as pos 0.00):there are some features in the user interface that i find a little inconvenient .

ERROR (neg classed as pos 0.00):the menu options appear one at a time , taking up the whole screen , and one has to scroll down one by one , or have memorized where they are in the menu order , to select them by using the number keypad .

ERROR (neg classed as pos 0.00):the vibrate setting is loud !

ERROR (neg classed as pos 0.00):so loud , really , that it does n't work terribly well as a silent ringer option .

ERROR (neg classed as pos 0.00):other things that i miss are voice-activated dialing and a standard 2.5 mm headset jack .

ERROR (neg classed as pos 0.00):- some features in the user interface

ERROR (neg classed as pos 0.00):- no voice activated dialing (what were they thinking

ERROR (neg classed as pos 0.00):- no option for caller-id pictures or individualized ringtones .

ERROR (neg classed as pos 0.00):- no propietary headset jack

ERROR (neg classed as pos 0.00):only if we could get one camera in it . .

ERROR (neg classed as pos 0.00):the volume key can be hard to press , but i think this may be by design rather than a flaw , perhaps to keep you from pressing it accidentally when you are in a call .

ERROR (neg classed as pos 0.00):pc cable is too expensive ... $ 50 for a usb cable ?

ERROR (neg classed as pos 0.00):the 6610 has the radio function which is utterly useless .

ERROR (neg classed as pos 0.00):it also has a stock tracking app built in (also useless) and a somewhat useful application which converts all types of metrics (currency , length , area , etc) .

ERROR (neg classed as pos 0.00):regarding pc software support nokia suite does not work with some versions of xp as i tried to connect my phone via infrared port on my laptop .

ERROR (neg classed as pos 0.00):its quiet .

ERROR (neg classed as pos 0.00):i do here a constant high-pitched distorted sound , not to mention the volume does not seem to be loud enough .

ERROR (neg classed as pos 0.00):although i find it more convenient to use 1-touch dialing , this phone does not have voice dialing .

ERROR (neg classed as pos 0.00):the keypad is a decent size , but the power on/off key is small and difficult to press .

ERROR (neg classed as pos 0.00):in my opinion the worst issue on this phone is the side-mounted volume control .

ERROR (neg classed as pos 0.00):one other issue is that the headphone jack is unique to nokia , so standard headphones will not work .

ERROR (neg classed as pos 0.00):one complaint ... the screen is too easily scratched !

ERROR (neg classed as pos 0.00):one more thing , the default ringtones that come with the phone are horrible .

ERROR (neg classed as pos 0.00):the only problem that i have found with the internet service is that it does not access mls (real estate broker software) very well at all .

ERROR (neg classed as pos 0.00):i find the lack of entertaining games on this phone quite disturbing .

ERROR (neg classed as pos 0.00):my only gripe about the hardware is the buttons .

ERROR (neg classed as pos 0.00):this model does have the traditional key arrangement , it 's just that they are really close to one another , and have unconventional shapes , so it takes a big getting used to for someone like me with big hands .

ERROR (neg classed as pos 0.00):finally , i reiterate my thumbs-down rating for t-mobile as a carrier .

ERROR (neg classed as pos 0.00):their network coverage is very sporadic , and the network always seems overloaded , resulting in very unpleasant calling experience .

ERROR (neg classed as pos 0.00):unfortunately , the 6610 does not offer voice dialing like my previous phone , but the other features it packs outweighs this shortcoming .

ERROR (neg classed as pos 0.00):one downside : as of this writing , t-mobile has n't updated their t-zone system to fully support the 6610 .

ERROR (neg classed as pos 0.00):the one huge disappointment is that the phones manufactured for t-mobile lack many of the menus and functions that a nokia straight from the manufacturer should have .

ERROR (neg classed as pos 0.00):the internet functions of the phone - wap and gprs - will only work through t-mobile 's services , because they have deleted the menu options that would enable you to configure the phone to be used on a different network .

ERROR (neg classed as pos 0.00):however , the main problem that i think is the with the sound quality .

ERROR (neg classed as pos 0.00):when talking the voice is not very clear .

9.2 The code Sentiment.py used for the entire sentimental analysis.

```
#!/usr/bin/env python

import re, random, math, collections, itertools


PRINT_ERRORS=0


#------------- Function Definitions ---------------------



def readFiles(sentimentDictionary,sentencesTrain,sentencesTest,sentencesNokia):


    #reading pre-labeled input and splitting into lines

    posSentences = open('rt-polarity.pos', 'r', encoding="ISO-8859-1")

    posSentences = re.split(r'\n', posSentences.read())


    negSentences = open('rt-polarity.neg', 'r', encoding="ISO-8859-1")

    negSentences = re.split(r'\n', negSentences.read())


    posSentencesNokia = open('nokia-pos.txt', 'r')

    posSentencesNokia = re.split(r'\n', posSentencesNokia.read())


    negSentencesNokia = open('nokia-neg.txt', 'r', encoding="ISO-8859-1")

    negSentencesNokia = re.split(r'\n', negSentencesNokia.read())


    posDictionary = open('positive-words.txt', 'r', encoding="ISO-8859-1")

    posWordList = re.findall(r"[a-z\-]+", posDictionary.read())


    negDictionary = open('negative-words.txt', 'r', encoding="ISO-8859-1")

    negWordList = re.findall(r"[a-z\-]+", negDictionary.read())
    for i in posWordList:

        sentimentDictionary[i] = 1

    for i in negWordList:

        sentimentDictionary[i] = -1



    #create Training and Test Datsets:

    #We want to test on sentences we haven't trained on, to see how well the model generalses to
previously unseen sentences



  #create 90-10 split of training and test data from movie reviews, with sentiment labels

    for i in posSentences:

        if random.randint(1,10)<2:

            sentencesTest[i]="positive"

        else:

            sentencesTrain[i]="positive"



    for i in negSentences:

        if random.randint(1,10)<2:

            sentencesTest[i]="negative"

        else:

            sentencesTrain[i]="negative"



    #create Nokia Datset:

    for i in posSentencesNokia:

            sentencesNokia[i]="positive"

    for i in negSentencesNokia:

            sentencesNokia[i]="negative"



#---------------------------End of data initialisation ----------------#
```

```
#calculates p(W|Positive), p(W|Negative) and p(W) for all words in training data

def trainBayes(sentencesTrain, pWordPos, pWordNeg, pWord):

    posFeatures = [] # [] initialises a list [array]

    negFeatures = []

    freqPositive = {} # {} initialises a dictionary [hash function]

    freqNegative = {}

    dictionary = {}

    posWordsTot = 0

    negWordsTot = 0

    allWordsTot = 0


    #iterate through each sentence/sentiment pair in the training data

    for sentence, sentiment in sentencesTrain.items():

        wordList = re.findall(r"[\w']+", sentence)


        for word in wordList: #calculate over unigrams

            allWordsTot += 1 # keeps count of total words in dataset

            if not (word in dictionary):

                dictionary[word] = 1

            if sentiment=="positive" :

                posWordsTot += 1 # keeps count of total words in positive class


                #keep count of each word in positive context

                if not (word in freqPositive):

                    freqPositive[word] = 1

                else:

                    freqPositive[word] += 1

            else:

                negWordsTot+=1# keeps count of total words in negative class
            #keep count of each word in positive context
            if not (word in freqNegative):
                freqNegative[word] = 1
            else:
                freqNegative[word] += 1

    for word in dictionary:
        #do some smoothing so that minimum count of a word is 1
        if not (word in freqNegative):
            freqNegative[word] = 1
        if not (word in freqPositive):
            freqPositive[word] = 1

        # Calculate p(word|positive)
        pWordPos[word] = freqPositive[word] / float(posWordsTot)

        # Calculate p(word|negative)
        pWordNeg[word] = freqNegative[word] / float(negWordsTot)

        # Calculate p(word)
        pWord[word] = (freqPositive[word] + freqNegative[word]) / float(allWordsTot)

#---------------------------End Training ----------------------------------

#implement naive bayes algorithm
#INPUTS:
#  sentencesTest is a dictonary with sentences associated with sentiment
#  dataName is a string (used only for printing output)
#  pWordPos is dictionary storing p(word|positive) for each word
#     i.e., pWordPos["apple"] will return a real value for p("apple"|positive)
#  pWordNeg is dictionary storing p(word|negative) for each word
#  pWord is dictionary storing p(word)
#  pPos is a real number containing the fraction of positive reviews in the dataset
def testBayes(sentencesTest, dataName, pWordPos, pWordNeg, pWord,pPos):
    pNeg=1-pPos

    #These variables will store results
    total=0
    correct=0
    totalpos=0
    totalpospred=0
    totalneg=0
    totalnegpred=0
    correctpos=0
    correctneg=0

    #for each sentence, sentiment pair in the dataset
    for sentence, sentiment in sentencesTest.items():
        wordList = re.findall(r"[\w']+", sentence)#collect all words

        pPosW=pPos
        pNegW=pNeg

        for word in wordList: #calculate over unigrams
            if word in pWord:
                if pWord[word]>0.00000001:
                    pPosW *=pWordPos[word]
                    pNegW *=pWordNeg[word]

        prob=0;
        if pPosW+pNegW >0:
            prob=pPosW/float(pPosW+pNegW)



        total+=1
        if sentiment=="positive":
            totalpos+=1
            if prob>0.5:
                correct+=1 # True positives + True Negatives
                correctpos+=1 # True Positives
                totalpospred+=1 # True Positive + False positive
            else:
                correct+=0
                totalnegpred+=1 #  + False Negative + True Negatives
                if PRINT_ERRORS:
                    print ("ERROR (pos classed as neg %0.2f):" %prob + sentence)
        else:
            totalneg+=1
            if prob<=0.5:
                correct+=1 # True positives + True Negatives
                correctneg+=1  # True negative
                totalnegpred+=1 # True Negatives + False Negatives
            else:
                correct+=0
                totalpospred+=1 #False positive + True Positive
                if PRINT_ERRORS:
                    print ("ERROR (neg classed as pos %0.2f):" %prob + sentence)


# TODO for Step 2: Add some code here to calculate and print: (1) accuracy; (2) precision and recall
for the positive class;
```

```
# (3) precision and recall for the negative class; (4) F1 score;


    # Accuracy = (True Positive + True Negative) / (True positive + True negative + False positive +
False negative)

    if (totalnegpred+totalpospred)==0:

        print("The accuracy is inconsistent")

        return

    else:

        Accuracy = correct/(totalnegpred+totalpospred)

    print("The Accuracy for Naive Bayes model is", Accuracy)

    print("Positive Class")

    # Precision = True Positive / (True Positive + False Positive)

    if (totalpospred)==0:

        print("The Precision is inconsistent")

        return

    else:

        Precision = correctpos/(totalpospred)

        print("The Precision for positive class in Naive Bayes model is", Precision)

    # Recall = True Positive / (True Positive + False Negative)

    if (correctpos+totalnegpred-correctneg)==0:

        print("The Recall is inconsistent")

        return

    else:

        Recall = correctpos/(correctpos+totalnegpred-correctneg)

        print("The Recall for positive class in Naive Bayes model is", Recall)

    # F- Measure = 2 * Precision * recall / (Precision + Recall)

    if (Precision+Recall)==0:

        print("The F-measure is inconsistent")

        return

    else:
```

```
    F = (2*Precision*Recall)/(Precision+Recall)
    print("The F-measure for positive class in Naive Bayes model is ",F)
print("\n")







print("Negative Class")
# Precision = True Negative / (True Negative + False Negative)
if (totalnegpred)==0:
    print("The Precision is inconsistent")
    return
else:
    Precision_neg = correctneg/(totalnegpred)
    print("The Precision for Negative class in Naive Bayes model is", Precision_neg)
# Recall = True Negative / (True Positive + False Negative)
if (correctneg+totalpospred-correctpos)==0:
    print("The Recall is inconsistent")
    return
else:
    Recall_neg = correctpos/(correctpos+totalnegpred-correctneg)
    print("The Recall for Negative class in Naive Bayes model is", Recall_neg)
# F- Measure = 2 * Precision * recall / (Precision + Recall)
if (Precision+Recall)==0:
    print("The F-measure is inconsistent")
    return
else:
    F_neg = (2*Precision_neg*Recall_neg)/(Precision_neg+Recall_neg)
    print("The F-measure for Negative class in Naive Bayes model is ",F_neg)
```

```
# This is a simple classifier that uses a sentiment dictionary to classify
# a sentence. For each word in the sentence, if the word is in the positive
# dictionary, it adds 1, if it is in the negative dictionary, it subtracts 1.
# If the final score is above a threshold, it classifies as "Positive",
# otherwise as "Negative"
def testDictionary(sentencesTest, dataName, sentimentDictionary, threshold):
    total=0
    correct=0
    totalpos=0
    totalneg=0
    totalpospred=0
    totalnegpred=0
    correctpos=0
    correctneg=0
    negation = ["not", "no" , "never"]
    for sentence, sentiment in sentencesTest.items():
        Words = re.findall(r"[\w']+", sentence)
        score=0
        for word in Words:

            if word in sentimentDictionary:
            score+=sentimentDictionary[word]

        total+=1
        if sentiment=="positive":

            totalpos+=1
            if score>=threshold:
                correct+=1
                correctpos+=1
                totalpospred+=1
            else:
                correct+=0
                totalnegpred+=1
        else:
            totalneg+=1
            if score<threshold:
                correct+=1
                correctneg+=1
                totalnegpred+=1
            else:
                correct+=0
                totalpospred+=1
```

```
# TODO for Step 5: Add some code here to calculate and print: (1) accuracy; (2) precision and recall for the positive class;
# (3) precision and recall for the negative class; (4) F1 score;

    # Accuracy = (True Positive + True Negative) / (True positive + True negative + False positive + False negative)
```

```
if (totalnegpred+totalpospred)==0:
    print("The accuracy is inconsistent")
    return
else:
    Accuracy = correct/(totalnegpred+totalpospred)
    print("The Accuracy for rule-based model is", Accuracy)
print("Positive Class")
# Precision = True Positive / (True Positive + False Positive)
if (totalpospred)==0:
    print("The Precision is inconsistent")
    return
else:
    Precision = correctpos/(totalpospred)
    print("The Precision for positive class in rule-based model is", Precision)
# Recall = True Positive / (True Positive + False Negative)
if (correctpos+totalnegpred-correctneg)==0:
    print("The Recall is inconsistent")
    return
else:
    Recall = correctpos/(correctpos+totalnegpred-correctneg)
    print("The Recall for positive class in rule-based model is", Recall)
# F- Measure = 2 * Precision * recall / (Precision + Recall)
if (Precision+Recall)==0:
    print("The F-measure is inconsistent")
    return
else:
    F = (2*Precision*Recall)/(Precision+Recall)
    print("The F-measure for positive class in rule-based model is ",F)
print("\n")

print("Negative Class")





print (totalnegpred)

# Precision = True Negative / (True Negative + False Negative)

if (totalnegpred)==0:

    print("The Precision is inconsistent")

    return


else:

    Precision_neg = correctneg/(totalnegpred)

    print("The Precision for Negative class in rule-based model is", Precision_neg)

# Recall = True Negative / (True positive + False Negative)

if (correctneg+totalpospred-correctpos)==0:

    print("The Recall is inconsistent")

    return

else:

    Recall_neg = correctpos/(correctpos+totalnegpred-correctneg)

    print("The Recall for Negative class in rule-based model is", Recall_neg)

# F- Measure = 2 * Precision * recall / (Precision + Recall)

if (Precision+Recall)==0:

    print("The F-measure is inconsistent")

    return

else:

    F_neg = (2*Precision_neg*Recall_neg)/(Precision_neg+Recall_neg)

    print("The F-measure for Negative class in rule-based model is ",F_neg)
```

```
#This is a dictionary-based classifier with negation, diminisher and intensifier
#rules to classify a sentence
#For each word in the sentence, if the word is in negation list and if the succeeding word is in
#positive dictionary, it adds the score to -3 and +3 if the word is in negative dictionary.
#For each word in the sentence, if the word is in Diminisher list and if the succeeding word is in
#positive dictionary, it adds the score to +2 and -2 if the word is in negative dictionary.
#For each word in the sentence, if the word is in Intensifier list and if the succeeding word is in
#positive dictionary, it adds the score to +4 and -4 if the word is in negative dictionary.
#also it ignores the article succeeding to any negation, intensifier and diminisher words.
#If the final score is above a threshold, it classifies as "Positive", otherwise as "Negative


def testDictionaryadvanced(sentencesTest, dataName, sentimentDictionary, threshold):
    total=0
    correct=0
    totalpos=0
    totalneg=0
    totalpospred=0
    totalnegpred=0
    correctpos=0
    correctneg=0

    # Creating a list of nagation words
    negation = ["not", "no" , "never", "cannot", "shouldn't", "wouldn't","doesn't", "n't"]
```

```
# Creating a list of Diminishing words

diminisher = ["somewhat", "rarely", "occasionally", "almost", "fairly", "kinda", "modicum", "partially",
"relatively", "some", "virtually", "barely", "few", "less", "mostly", "practically", "reasonably", "bit",
"fewer", "little", "nearly", "quite", "scanty", "sparsely", "exiguous", "hardly", "marginally", "negligibly",
"rather", "scarcely", "tolerably", "faintly", "insignificantly", "moderately", "partly", "slightly", "triflingly",
"virtually"]

# Creating a list of Intensifying words

intensifier = ["absolutely", "altogether", "amazingly", "astoundlingly", "awfully", "badly", "decidedly",
"bitterly", "bloody", "colossaly", "completely", "damn", "deeply", "drastically", "dreadfully", "entirely",
"enormously", "especially", "exceptionally", "excessively", "extraordinarily", "extremely", "fantastically",
"freaking", "frightfully", "fully", "greatly", "hella", "highly", "incredibly", "insanely", "intensly",
"immensely", "largely", "literally", "lot", "lots", "massive", "mightily", "more", "outrageously",
"particularly", "perfectly", "phenomenally", "pretty", "radically", "real", "really", "remarkably", "purely",
"so", "soo", "sooo", "super", "supremely", "surpassingly", "strikingly", "strongly", "terribly", "terrifically",
"totally", "thoroughly", "truly", "unusually", "utterly", "very", "wicked"]

# Creating a list of articles

article = ["a", "so", "the"]

for sentence, sentiment in sentencesTest.items():

    Words = re.findall(r"[\w']+", sentence)

    score=0

    neg=0

    dim=0

    iten=0

    for word in Words:

        if word in negation: # implementing negation rule



            #print(word,":")

            neg=1

            continue

        if word in diminisher:

            #print(word, ":")

            dim=1



        if word in sentimentDictionary:

            if word in article:    #Ignore the article succeeding a negation, diminisher or Intensifier word
    score+=sentimentDictionary[word]
    continue
if neg==1:
    if sentimentDictionary[word]==-1:
        score+=3
        #print(word,sentimentDictionary[word])
    else:
        score-=3
        #print(word,sentimentDictionary[word])
    neg=0
elif dim==1:    # Implement Diminisher rule
    if sentimentDictionary[word]==-1:
        score-=2
        #print(word,sentimentDictionary[word])
    else:
        score+=2
        #print(word,sentimentDictionary[word])
    dim=0
elif iten==1:  # Implement Intensifier rule
    if sentimentDictionary[word]==-1:
        score-=4
        #print(word,sentimentDictionary[word])
    else:
        score+=4
        #print(word,sentimentDictionary[word])
    iten=0
else:
    score+=sentimentDictionary[word]
```

```
        total+=1



        if sentiment=="positive":
            totalpos+=1
            if score>=threshold:
                correct+=1
                correctpos+=1
                totalpospred+=1
            else:
                correct+=0
                totalnegpred+=1
        else:
            totalneg+=1
            if score<threshold:
                correct+=1
                correctneg+=1
                totalnegpred+=1
            else:
                correct+=0
                totalpospred+=1



# TODO for Step 5: Add some code here to calculate and print: (1) accuracy; (2) precision and recall for the positive class;
# (3) precision and recall for the negative class; (4) F1 score;



    # Accuracy = (True Positive + True Negative) / (True positive + True negative + False positive + False negative)
    if (totalnegpred+totalpospred)==0:
        print("The accuracy is inconsistent")
    return
else:
    Accuracy = correct/(totalnegpred+totalpospred)
    print("The Accuracy for rule-based model is", Accuracy)
print("Positive Class")
# Precision = True Positive / (True Positive + False Positive)
if (totalpospred)==0:
    print("The Precision is inconsistent")
    return
else:
    Precision = correctpos/(totalpospred)
    print("The Precision for positive class in rule-based model is", Precision)
# Recall = True Positive / (True Positive + False Negative)
if (correctpos+totalnegpred-correctneg)==0:
    print("The Recall is inconsistent")
    return
else:
    Recall = correctpos/(correctpos+totalnegpred-correctneg)
    print("The Recall for positive class in rule-based model is", Recall)
# F- Measure = 2 * Precision * recall / (Precision + Recall)
if (Precision+Recall)==0:
    print("The F-measure is inconsistent")
    return
else:
    F = (2*Precision*Recall)/(Precision+Recall)
    print("The F-measure for positive class in rule-based model is ",F)



print("\n")
print("Negative Class")
```

```
print (totalnegpred)

# Precision = True Negative / (True Negative + False Negative)

if (totalnegpred)==0:

   print("The Precision is inconsistent")

   return


else:

   Precision_neg = correctneg/(totalnegpred)

   print("The Precision for Negative class in rule-based model is", Precision_neg)

# Recall = True Negative / (True Positive + False Negative)

if (correctneg+totalpospred-correctpos)==0:

   print("The Recall is inconsistent")

   return

else:

   Recall_neg = correctpos/(correctpos+totalnegpred-correctneg)

   print("The Recall for Negative class in rule-based model is", Recall_neg)

# F- Measure = 2 * Precision * recall / (Precision + Recall)

if (Precision+Recall)==0:

   print("The F-measure is inconsistent")

   return

else:

   F_neg = (2*Precision_neg*Recall_neg)/(Precision_neg+Recall_neg)

   print("The F-measure for Negative class in rule-based model is ",F_neg)
```

```
#Print out n most useful predictors

def mostUseful(pWordPos, pWordNeg, pWord, n):

    predictPower={}

    sum1 =0

    sum2=0

    for word in pWord:

        if pWordNeg[word]<0.0000001:

            predictPower=1000000000

        else:

            predictPower[word]=pWordPos[word] / (pWordPos[word] + pWordNeg[word])


    sortedPower = sorted(predictPower, key=predictPower.get)

    head, tail = sortedPower[:n], sortedPower[len(predictPower)-n:]

    print ("NEGATIVE:")

    print (head)

    print ("\nPOSITIVE:")

    print (tail)

    for wor in sortedPower[:n]:

        if wor in sentimentDictionary:

            sum1+=1

    print(sum1)

    for wor in sortedPower[len(predictPower)-n:]:

        if wor in sentimentDictionary:

            sum2+=1

    print(sum2)




#---------- Main Script --------------------------
```

```
sentimentDictionary={} # {} initialises a dictionary [hash function]

sentencesTrain={}

sentencesTest={}

sentencesNokia={}


#initialise datasets and dictionaries

readFiles(sentimentDictionary,sentencesTrain,sentencesTest,sentencesNokia)


pWordPos={} # p(W|Positive)

pWordNeg={} # p(W|Negative)

pWord={}    # p(W)


#build conditional probabilities using training data

trainBayes(sentencesTrain, pWordPos, pWordNeg, pWord)


#run naive bayes classifier on datasets

print ("Naive Bayes")

#testBayes(sentencesTrain,  "Films (Train Data, Naive Bayes)\t", pWordPos, pWordNeg, pWord,0.5)

print("\n")

#testBayes(sentencesTest,  "Films  (Test Data, Naive Bayes)\t", pWordPos, pWordNeg, pWord,0.5)

print("\n")

#testBayes(sentencesNokia, "Nokia   (All Data,  Naive Bayes)\t", pWordPos, pWordNeg, pWord,0.7)

print("\n")




#run sentiment dictionary based classifier on datasets

#testDictionary(sentencesTrain,  "Films (Train Data, Rule-Based)\t", sentimentDictionary, -4)
```

```
print("\n")

#testDictionary(sentencesTest,  "Films  (Test Data, Rule-Based)\t",  sentimentDictionary, -4)

print("\n")

testDictionary(sentencesNokia, "Nokia   (All Data, Rule-Based)\t",  sentimentDictionary, -3)

print("\n")


#testDictionaryadvanced(sentencesTest,  "Films  (Test Data, Rule-Based)\t",  sentimentDictionary, 0)

print("\n")

#testDictionaryadvanced(sentencesNokia, "Nokia   (All Data, Rule-Based)\t",  sentimentDictionary, 0)

print("\n")

#testDictionaryadvanced(sentencesTrain,  "Films (Train Data, Rule-Based)\t", sentimentDictionary, 0)


# print most useful words

#mostUseful(pWordPos, pWordNeg, pWord, 50)
```

Note: The 1000 words limitation exclude tables, graphs and appendices.