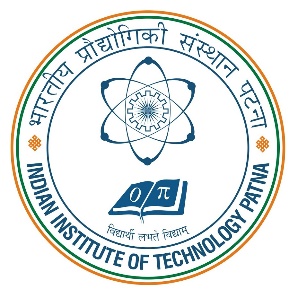
**Foundation of Machine Learning**

**(CS564)**

Assignment 4 :Twitter Sentiment Analysis using Naïve Bayes Classifier

Anindya Sundar Das (1811MC02)



Introduction:

The purpose of this assignment was to do sentiment analysis using Naïve Bayes Classifier on Twitter dataset. There are two class labels are 0: negative 1: positive. Tweets should brought in usable format before feeding it to classifier hence we have to do the pre-processing.

**Step A**. Collect Data from Twitter

**Step B**. Do pre-processing to clean the data.

**Step C.** Apply Naive Bayes Classification algorithm on Cleaned data to classify positive and negative classes and apply trained model on test data and find accuracy, F1 Score Precision and Recall.

**Step A : Data Collection**

Dataset : Twitter Sentiment Analysis Dataset

Total number of Tweet : 7086

Number of Classes for prediction : 0 (Negative Sentiment) 1 (Positive Sentiment)

Split data into two part 1) Training Set 2) Testing Set

**1.** Training Set contains 75% of data.

**2.** Testset contains 25% of data.

**Step B : Pre-Processing**

Step 1 : Remove hash-tag ,URL and Mentions from the dataset.

data=re.sub(' +', ' ', data)

text = re.sub(r"(^https?:\/\/.\*[\r\n]\*)|(?:\#+[\w\_]+[\w\'\_\-]\*[\w\_]+)|(?:@[\w\_]+)", '', data, flags=re.MULTILINE)

Step 2 : Replace apexes by 'not'

text = re.sub(r"\b(\w+n't)\b", 'not', text)

Step 3.1 : Replace Positive smileys by 'smile\_positive'

text = re.sub(r":\)|:‑\)|:-]|:-3|:->|8-\)|:-}|:o\)|:c\)|:^\)|=]|=\)|:\)|:]|:3|:>|8\)|:}|:‑D|:D|8‑D|8D|x‑D|xD|X‑D|XD|=D|=3|B^D|:-\)\)|:'‑\)|:'\):‑O|:O|:‑o|:o|:-0|8-0|>:O|:-\\*|:\\*|:×|;‑\)|;\)|\\*-\)|\\*\|;‑]|;]|;^\)|:‑,|;D\|:‑P|:P|X‑P|XP|x‑p|xp|:‑p|:p|:‑Þ|:‑Þ|:‑þ|:þ|:Þ|:Þ|:‑b|:b|d:|=p|>:P\|O:‑\)|O:\)|0:‑3|0:3|0:‑\)|0:\)|0;^\)|;‑\)|:‑J|#‑\)|%‑\)|%\)|<3|@};-|@}->--|@}‑;‑'‑‑‑|@>‑‑>‑‑", "smilepositive", text)

Step 3.2 : Replace Negative smileys by 'smile\_negative'

text = re.sub(r":‑\(|:\(|:‑c|:c|:‑<|:<|:‑\[|:\[|:-\|\||>:\[|:{|:@|>:\(|:'‑\(|:'\(|D‑':|D:<|D:|D8|D;|D=|DX|:‑/|:/|:‑.|>:\|:L|=L|:S|:‑\||:\||:‑X|:X|:‑#|:#|:‑&|:&|>:‑\)|>:\)|}:‑\)|}:\)|3:‑\)|3:\)|>;\)|',:-l|',:-\||>\_>|<\_<|<\\3|</3", "smilenegative", text)

Step 4 : Remove punctuations from the text

translator = str.maketrans('', '', string.punctuation)# retrun a table which maps each punctuation to NONE

text=text.translate(translator)

Step 5 : Converting to lowercase and save it to sentences.txt

text=text.lower()

f2 = open("sentences.txt","w")

print(text,file=f2)

Step 6 : Apply Stemming on the data sentence wise

f3 = open("sentences.txt",'r')

text=f3.readlines()

ps = PorterStemmer()

stems1=[]

for line in text:

tokens = word\_tokenize(line)

stems=[]

for word in tokens:

stems.append(ps.stem(word))

stems1.append(stems)

Step 7 : Removal of Stopwords

f2 = open("sentences.txt")

text=f2.readlines()

stop\_words = set(stopwords.words('english'))

stop1 = []

for line in text:

word\_tokens = word\_tokenize(line)

filtered\_sentence = []

for w in word\_tokens:

if w not in stop\_words:

filtered\_sentence.append(w)

stop1.append(filtered\_sentence)

Step 8 : Save the preprocessed data to a ‘.txt’ file for further use.

**Step C : Implement Naive Bayes Classifier**

Naive Bayes classifiers are a collection of classification algorithms based on Bayes’ Theorem. This type of classifiers are called Probabilistic Classifier. Naive Bayes is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Here we have used Bayes Classifier to classify text. The procedure as follows :

*Step 1:* Read pre-processed Twitter Data

*Step 2:* Binary Classification where tweets are only labelled “Positive” denoted by “1” and “Negative” denoted by “0”.

*Step 3:*Divide into trainset(75%) and test set(25%)

*Step 4:* Define multinomial Naive Bayes Classifier which allows us to fit the classifier on our training data, predict the sentiment (positive or negative) of each tweet, and score our results (i.e., determine how many tweets we've classified correctly.

*Step 5.1 :* The classifier is fitted to the training data unct to compute naive Bayes classification probabilities.

*Step 5.2 :* Apply the classifier on the test set and predict the class

*Step 5.3 :* Find Accuracy, Precession, Recall, F-measure of the Classifier

*Step 5.4 :* Prepare the confusion matrix and produce the classification report.

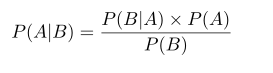
*Step 6 :* End

**Discussion on how we built our Classifier**

In the first step, we focus on extracting features of text. We need numerical features as input for our classifier. So an intuitive choice would be word frequencies, i.e., counting the occurrence of every word in the document.

Then, we need to convert the probability that we wish to calculate into a form that can be calculated using word frequencies. Here, we adopt the properties of possibilities and Bayes’ Theorem to do the conversion.

Bayes’ Theorem is useful for dealing with conditional probabilities, since it provides a way for us to reverse them.

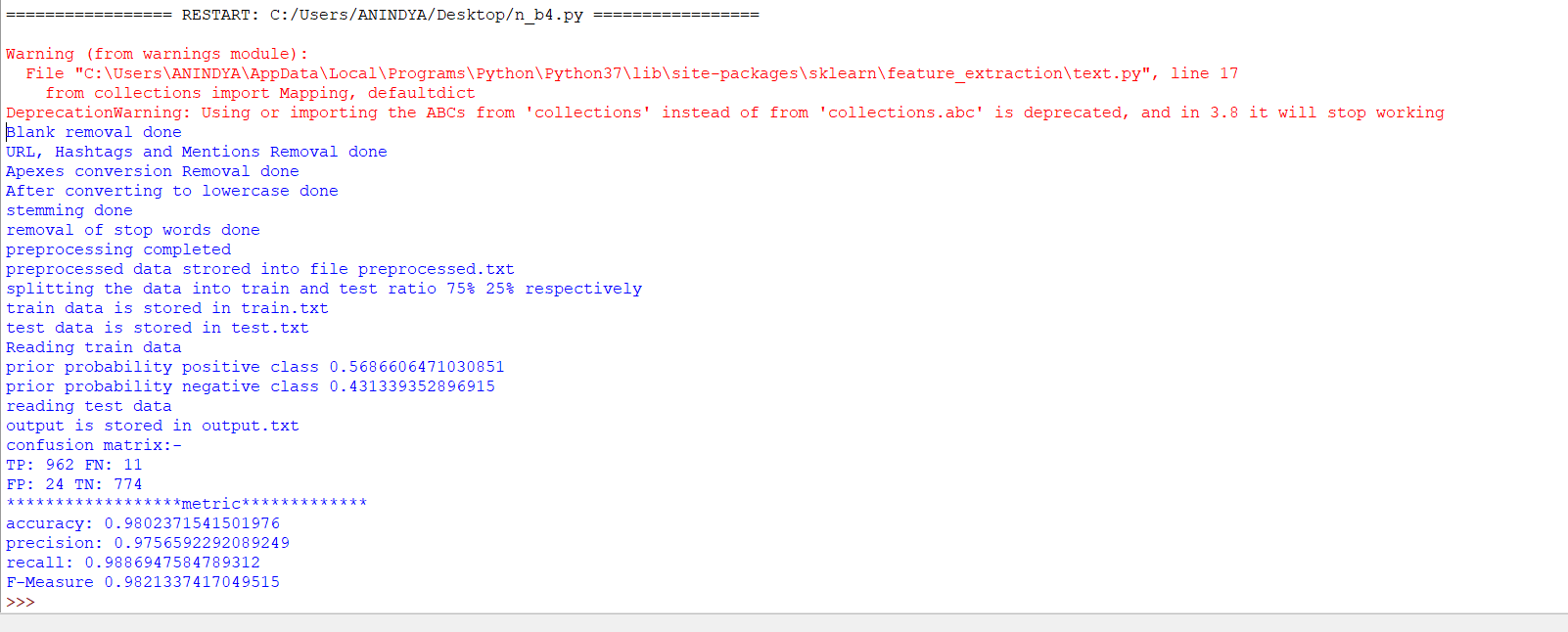


In Naive Bayes, there is an assumption that every word is independent of one another. Now, we look at individual words in a sentence, instead of the entire sentence.

In the final step, we simply calculate the probability of each category using Bayes theorem, and the category with the highest probability is the output.

It is problematic when a frequency-based probability is zero, because it will wipe out all the information in the other probabilities, and we need to find a solution for this. A solution would be Laplace smoothing, which is a technique for smoothing categorical data. A small-sample correction, or pseudo-count, will be incorporated in every probability estimate. Consequently, no probability will be zero. this is a way of regularizing Naive Bayes, and when the pseudo-count is zero, it is called Laplace smoothing.

Results:



prior probability positive class 0.5686606471030851

prior probability negative class 0.431339352896915

reading test data

output is stored in output.txt

confusion matrix:-

TP: 962 FN: 11

FP: 24 TN: 774

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*metric\*\*\*\*\*\*\*\*\*\*\*\*\*

accuracy: 0.9802371541501976

precision: 0.9756592292089249

recall: 0.9886947584789312

F-Measure 0.9821337417049515