



بنام خدا

دانشکده‌ی مهندسی برق و کامپیوتر

درس پردازش زبان‌های طبیعی

شایان واصف احمدزاده

شماره دانشجویی : 810197603

تمرین کامپیوتری 2

استاد : دکتر فیلی

Naïve Bayes

روز آپلود : 28 فروردین

Table of Contents

SMS Spam Detection	2
Considering Punctuations	2
Lower-band=100	3
Finding the best-suited Low-band:.....	6
Ignoring Punctuations:	7
Finding the best-suited Low-band:.....	7
Sentimental LIAR	8
NLTK	8
Removing Null values from sentiment column.....	9
Deleting sentiment column	12
Replacing Null values with zeros	13
What is the best threshold?	13
Sklearn	14

SMS Spam Detection

Considering Punctuations

هدف این سوال تشخیص پیام های Spam از غیر Spam می باشد.

در ابتدا دیتاست داده شده توسط سوال را می خوانیم :

	Label	SMS	
0	ham	Go until jurong point, crazy.. Available only ...	
1	ham	Ok lar... Joking wif u oni...	
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	
3	ham	U dun say so early hor... U c already then say...	
4	ham	Nah I don't think he goes to usf, he lives aro...	

شکل 1.1 : دیتاست قسمت اول

در ادامه تخمینی از تعداد هر کدام از لیبل های موجود بدست می آوریم :

```
sms_spam_collection['Label'].value_counts(normalize=True)
```

```
ham      0.865937
spam     0.134063
Name: Label, dtype: float64
```

شکل 1.2 : تعداد نرمالایز شده داخل ستون لیبل

در ادامه، پکیج های مربوطه را ایجاد می کنیم :

```
import nltk
nltk.download('punkt')
from nltk.tokenize import sent_tokenize, word_tokenize
```

برای استفاده از Naïve Bayes، نیاز به روابط زیر داریم :

The formula that we will be using to calculate the probabilities is :

- $P(S|w_1 w_2 w_3...w_n) \sim P(S) P(w_1 | S) P(w_2 | S).....P(w_n | S)$
- $P(NS|w_1 w_2 w_3...w_n) \sim P(NS) P(w_1 | NS) P(w_2 | NS).....P(w_n | NS)$

$P(W_n | spam) = (N_{wn|spam} + \alpha) / (N_{spam} + (N_{vocab} * \alpha))$

- N_{wn} = The number of times the word occurs in the spam message
- $\alpha = 1$ (smoothing parameter)
- N_{spam} = The total number of words in the spam messages
- N_{vocab} = The total number of words in the vocabulary

برای پیش پردازش، تنها تمامی لغات را به شکل Lower-Case درمی آوریم.

```
sms_spam_collection['SMS'] = sms_spam_collection['SMS'].str.lower()
```

سپس لغات را بر حسب تعداد تکرار آنها در Corpus در قالب یک Dictionary ایجاد می کنیم :

```
all_words = []
for index, row in sms_spam_collection.iterrows():
    for word in word_tokenize(row['SMS']):
        all_words.append(word.lower())
all_words = nltk.FreqDist(all_words)
```

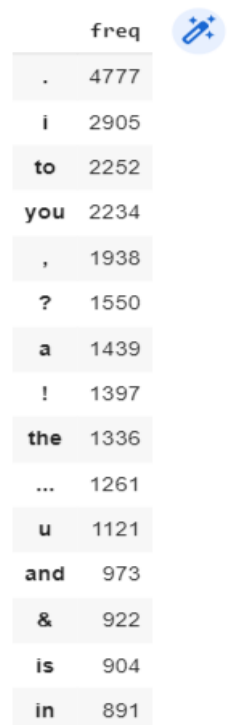
طول تمامی لغات اضافه شده به Dictionary برابر 9837 می باشد.

```
len(all_words) : 9837
```

Lower-band=100

در وهله بعد، لغاتی که در Dictionary بیش از 100 بار تکرار شده اند را به Vocabulary اضافه می کنیم :

```
vocab=[]
for c,n in list(all_words.items()):
    if n>=100:
        vocab.append(c)
```



	freq
.	4777
i	2905
to	2252
you	2234
,	1938
?	1550
a	1439
!	1397
the	1336
...	1261
u	1121
and	973
&	922
is	904
in	891

شکل 1.3 : 15 Token اول Vocabulary

سپس هر سطر موجود در داده‌ی آموزش Corpus را به Token ها تجزیه می‌کنیم و Token های هر سطر را به همراه لیبل آن در یک لیست ذخیره می‌کنیم.

```
lst_token= []
for index, row in sms_spam_collection.iterrows():
    lst_token.append((word_tokenize(row['SMS']),row['Label']))
```

حال نیاز به یک feature set داریم که هر Token داخل Vocabulary را بررسی کنیم و در صورتی که آن Token در Token های مربوط به هر سطر وجود داشت، مقدار آن را برابر True و در غیر این صورت برابر False قرار داد :

```
def find_features(vocabulary,document):
    words = set(document)
    features = {}
    for w in vocabulary: # 100 frequent words
        features[w] = (w in words) # True/False
    return features
```

```
featuresets = [(find_features(vocab,message), label) for (message, label) in lst_token]
```

اولین عنصر feature set بدست آمده که مربوط به اولین سطر Corpus می‌باشد، به صورت زیر است :

```
featuresets[0]
({'!': False, '#': False, '&': False, ' ': False, '"': False, "'": False, '(': False, ')': False,
',': True, '-': False, ':': False, ';': True, '2': False, '4': False, '.': False, ',': False, '?': False,
'`': False, 'a': False, 'about': False, 'all': False, 'am': False, 'an': False, 'and': False, 'any': False,
'are': False, 'as': False, 'at': False, 'back': False, 'be': False, 'been': False, 'but': False, 'by': False,
'c': False, 'call': False, 'can': False, 'claim': False, 'come': False, 'd': False, 'da': False, 'day': False,
'dear': False, 'did': False, 'do': False, 'dont': False, 'for': False, 'free': False, 'from': False,
'get': False, 'give': False, 'go': True, 'going': False, 'good': False, 'got': True, 'great': True,
'gt': False, 'had': False, 'happy': False, 'has': False, 'have': False, 'he': False, 'her': False,
'here': False, 'hey': False, 'hi': False, 'him': False, 'home': False, 'hope': False, 'how': False,
'i': False, 'if': False, 'in': True, 'is': False, 'it': False, 'its': False, 'just': False, 'know': False,
'later': False, 'like': False, 'lor': False, 'love': False, 'lt': False, 'make': False, 'me': False,
'mobile': False, 'more': False, 'much': False, 'my': False, 'n': True, 'n't': False, 'na': False,
'need': False, 'new': False, 'night': False, 'no': False, 'not': False, 'now': False, 'of': False,
'oh': False, 'ok': False, 'on': False, 'one': False, 'only': True, 'or': False, 'our': False, 'out': False,
'phone': False, 'please': False, 'pls': False, 'r': False, 'reply': False, 'see': False, 'send': False,
'she': False, 'should': False, 'so': False, 'some': False, 'sorry': False, 'still': False, 'stop': False,
'take': False, 'tell': False, 'text': False, 'that': False, 'the': False, 'then': False, 'there': True,
```

```
'they': False, 'think': False, 'this': False, 'time': False, 'to': False, 'today': False, 'too': False,
'txt': False, 'u': False, 'up': False, 'ur': False, 'want': False, 'was': False, 'wat': True, 'way': False,
'we': False, 'week': False, 'well': False, 'what': False, 'when': False, 'where': False, 'who': False,
'will': False, 'with': False, 'you': False, 'your': False, 'ü': False},
'NS')
```

بعد از مراحل بالا، دیتاست آموزش و تست خود را تشکیل می‌دهیم :

```
import random
random.Random(4).shuffle(featuresets)
# divide training set and test set
training_set = featuresets[:round(len(sms_spam_collection) * 0.80)]
testing_set = featuresets[round(len(sms_spam_collection) * 0.80):]
```

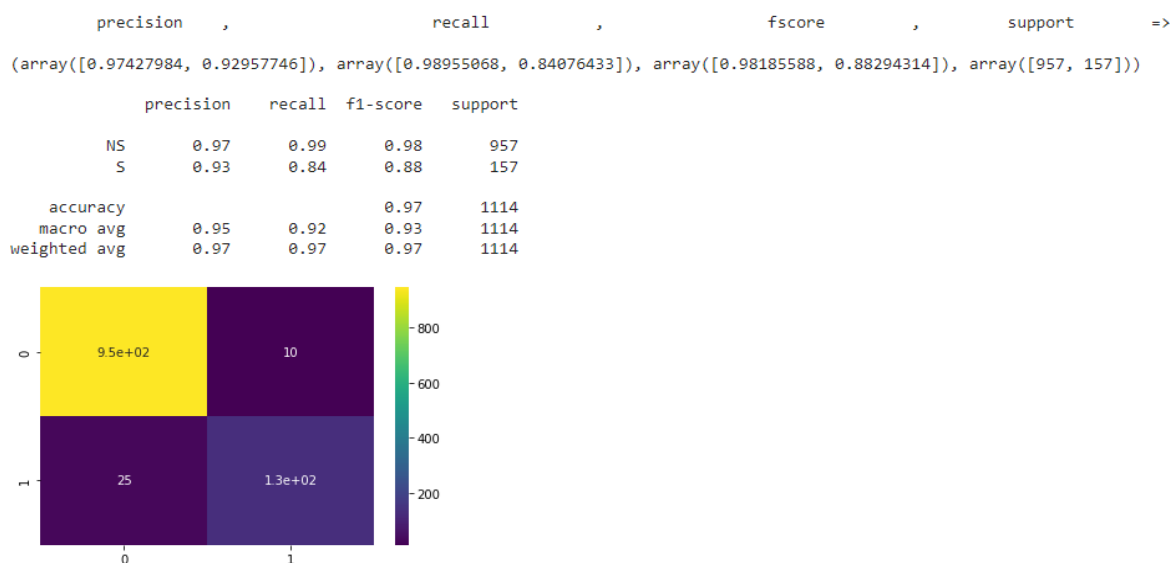
سپس کتابخانه‌های مورد نیاز را ایجاد می‌کنیم :

```
from sklearn.metrics import precision_recall_fscore_support, classification_report
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

سپس به کمک کتابخانه NLTK، دقت طبقه بند را محاسبه می‌کنیم :

```
NB = nltk.NaiveBayesClassifier.train(training_set)
print(nltk.classify.accuracy(NB, testing_set)*100) : 96.85816876122082%
```

در نهایت، گزارشی از طبقه بند بدست آمده ارائه می‌دهیم :

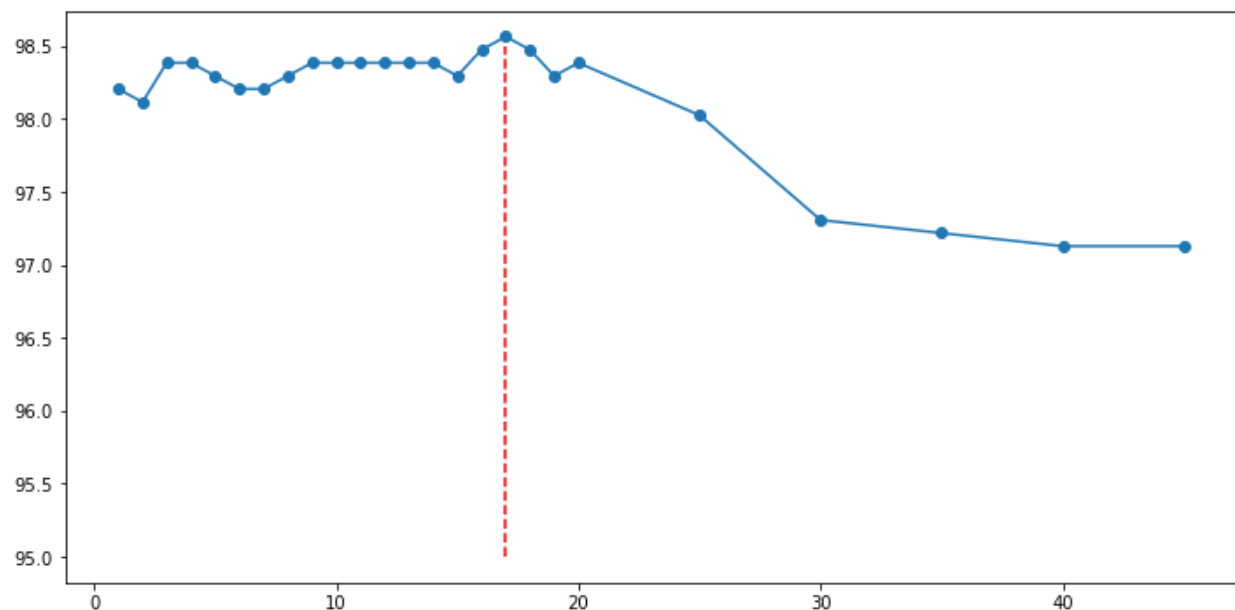


شکل 1.4 : گزارش طبقه بند

Finding the best-suited Low-band:

در قسمت قبل، با انتخاب حد پایین 100، دقت طبقه بند را محاسبه کردیم ولی در این قسمت به دنبال بهترین حد پایین هستیم که به بیشترین دقت برسیم (نکته مهم این است که Punctuation ها نقش مهمی در تعیین این حد بازی می کنند).

```
import matplotlib.pyplot as plt
sel=list(range(1,20))+list(range(20,50,5))
Acc=[]
for lb in sel:
    V=[]
    for c,n in list(all_words.items()):
        if n>=lb:
            V.append(c)
    fs = [(find_features(V,M), label) for (M, label) in lst_token]
    random.Random(4).shuffle(fs)
    training_set = fs[:round(len(sms_spam_collection) * 0.80)]
    testing_set = fs[round(len(sms_spam_collection) * 0.80):]
    NB = nltk.NaiveBayesClassifier.train(training_set)
    Acc.append(nltk.classify.accuracy(NB , testing_set)*100)
plt.figure(figsize=(12,6))
plt.plot(sel,Acc,marker='o')
plt.vlines(sel[Acc.index(max(Acc))],95,98.5,ls='--',color='red')
```



شکل 1.5: نمودار دقت بر حسب حد پایین تکرار Token در Vocabulary (با Punctuation)

طبق شکل بالا، بیشترین مقدار دقت در حد پایین برابر 17 رخ داده است که دقتی برابر 99% دارد.

Ignoring Punctuations:

این بار از کل Corpus، تمامی Punctuation ها را حذف می کنیم و مانند قسمت های قبل ادامه می دهیم :

```
import string
for character in string.punctuation:
    sms_spam_collection['SMS'] = sms_spam_collection['SMS'].str.replace(character, "").str.lower()
```

حالا اینبار تعداد Token های در Corpus را می شماریم :

```
# Make a bag of word and count the frequency of the word
```

```
all_words = []
```

```
for index, row in sms_spam_collection.iterrows():
```

```
    for word in word_tokenize(row['SMS']):
```

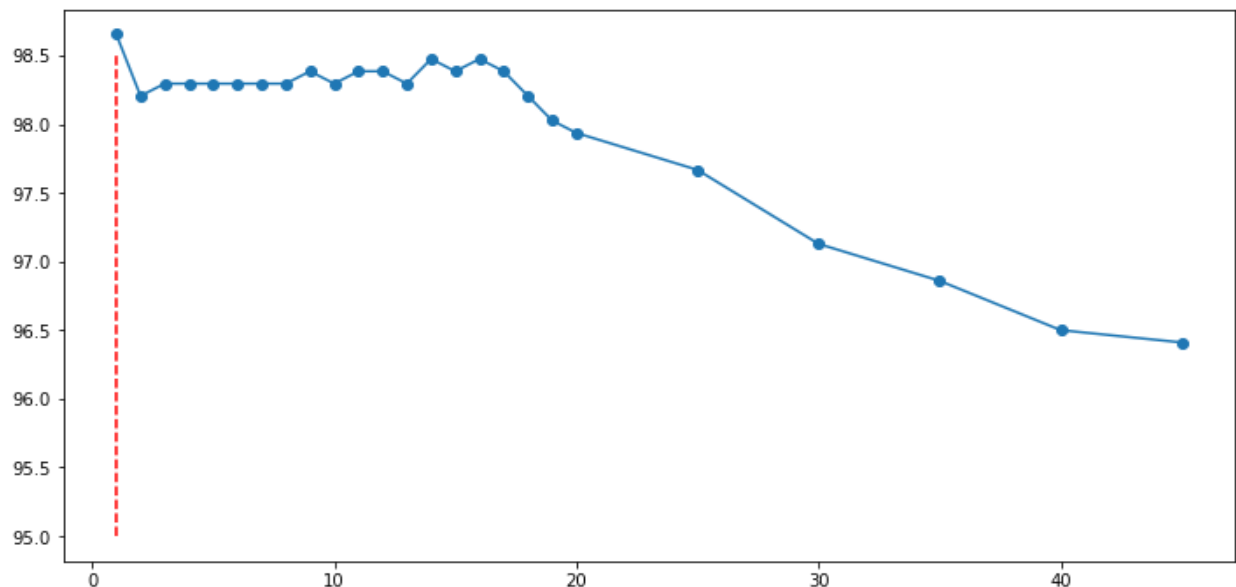
```
        all_words.append(word.lower())
```

```
all_words = nltk.FreqDist(all_words)
```

```
len(all_words) : 9642
```

Finding the best-suited Low-band:

مانند قسمت قبل، اینبار شرایط را بدون Punctuation بررسی می کنیم :



شکل 1.6: نمودار دقت بر حسب حد پایین تکرار Token در Vocabulary (بدون Punctuation)

*همانطور که مشخص است، با حذف تمام علائم نگارشی، بیشترین دقت هنگامی رخ می دهد که اندازه feature set ما بیشینه باشد، به معنای دیگر، از کل Token ها به عنوان ویژگی ها استفاده کنیم.

Sentimental LIAR

NLTK

در ابتدا، دو دیتاست آموزش و تست را می‌خوانیم:

```
import pandas as pd
train_set=pd.read_csv("train_final.csv")
test_set=pd.read_csv("test_final.csv")
```

Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	Unnamed: 0.1.1.1	Unnamed: 0.1.1.1.1	Unnamed: 0.1.1.1.1.1	ID	label	statement	subject	...	sentiment_score	sentiment_magnitude	anger	fear	joy	disgust	sad	speaker_id	list	sentiment_code
0	0	0	0	0	0	0	2635.json	Says the Annies List political group supports ...	abortion	...	-0.5	0.5	0.121137	0.008926	0.026096	0.263479	0.531887	_0_	[0, 1]	_NEG_
1	1	1	1	1	1	1	10540.json	When did the decline of coal start? It started...	energy.history.job-accomplishments	...	-0.4	0.8	0.095352	0.124566	0.191357	0.016999	0.102045	_1_	[0, 1]	_NEG_
2	2	2	2	2	2	2	324.json	Hillary Clinton agrees with John McCain "by vo...	foreign-policy	...	-0.3	0.3	0.039559	0.024162	0.500384	0.454228	0.052453	_2_	[1, 0]	_NEG_
3	3	3	3	3	3	3	1123.json	Health care reform legislation is likely to ma...	health-care	...	-0.3	0.3	0.004804	0.194674	0.375055	0.022509	0.383403	_3_	[0, 1]	_NEG_
4	4	4	4	4	4	4	9028.json	The economic turnaround started at the end of ...	economy.jobs	...	0.0	0.0	0.044237	0.215996	0.222402	0.045672	0.274343	_4_	[0, 1]	NaN

5 rows x 31 columns

شکل 2.1: 5 سطر اول دادگان آموزش

تعداد مقادیر یکتای داخل ستون لیبل ('label') در دادگان آموزش به صورت زیر می‌باشد:

```
train_set['label'].unique ()
array (['false', 'half-true', 'mostly-true', 'true', 'barely-true', 'pants-fire'], dtype=object)
```

طبق خواسته سوال، لازم است تا ستون لیبل در حالت جدید تنها شامل مقادیر باینری 0 و 1 باشد:

```
train_set= train_set.replace ({'label': {'barely-true':'F', 'pants-fire':'F', 'false':'F','half-true':'T','mostly-true':'T','true':'T'}})
test_set= test_set.replace ({'label': {'barely-true':'F', 'pants-fire':'F', 'false':'F','half-true':'T','mostly-true':'T','true':'T'}})
train_set.head()
```

ID	label	statement	subject
2635.json	F	Says the Annies List political group supports ...	abortion
10540.json	T	When did the decline of coal start? It started...	energy.history.job-accomplishments
324.json	T	Hillary Clinton agrees with John McCain "by vo...	foreign-policy

1123.json	F	Health care reform legislation is likely to ma...	health-care
9028.json	T	The economic turnaround started at the end of ...	economy.jobs

شکل 2.2: لیبل های تغییر اسم یافته

Removing Null values from sentiment column

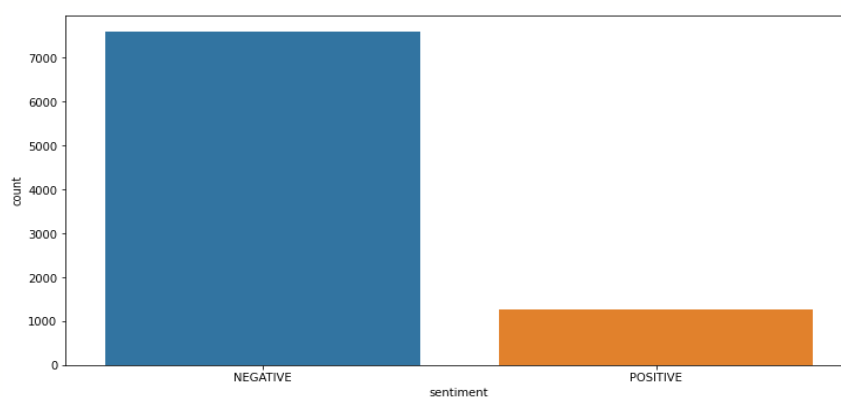
طبق جدول داده شده، در صورتی که مقدار عددی sentiment (Sentiment Score) عدد صفر باشد، در ستون sentiment مقدار Null نوشته شده است. چندین راهکار در رویایی با این مشکل وجود دارد. در این قسمت، ما سطر هایی را که شامل ستون Null هستند را حذف می کنیم ولی ستون sentiment را حفظ می کنیم:

```
train_set=train_set[train_set['sentiment'].notnull()].reset_index(drop=True)
print(train_set.shape) : (8848, 31)
```

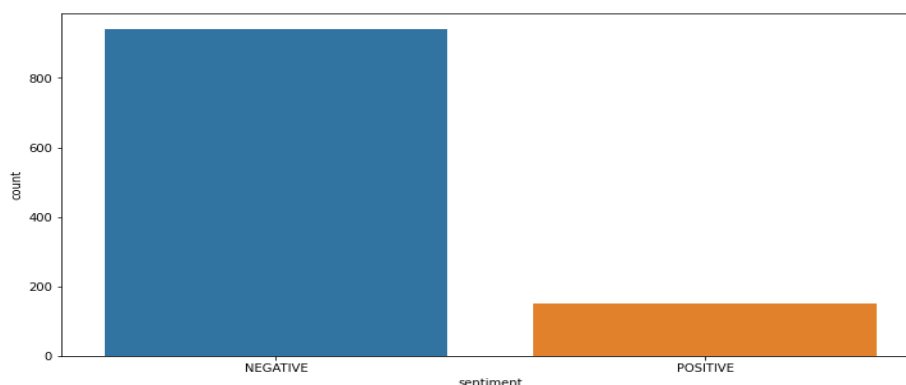
```
test_set=test_set[test_set['sentiment'].notnull()].reset_index(drop=True)
print(test_set.shape) : (1091, 30)
```

```
train_set['sentiment'].unique()
```

همچنین می توانیم تخمینی از تعداد Positive و Negative در ستون 'sentiment' داشته باشیم :



شکل 2.3 : تعداد unique های ستون sentiment (دادگان آموزش)



شکل 2.4 : تعداد unique های ستون sentiment (دادگان تست)

همچنین از آنجا که قرار است، ستون sentiment یکی از ستونهای ویژگی ما باشد، باید مقادیر درون آن عددی باشد :

```
def binarize_text(text):
    if text=='NEGATIVE':
        return False
    else:
        return True
```

```
train_set['sentiment'] = train_set['sentiment']. apply(binarize_text)
test_set['sentiment'] = test_set['sentiment']. apply(binarize_text)
```

در ادامه به تحلیل یکی از 5 ستون ['anger', 'fear', 'joy', 'disgust', 'sad'] می پردازیم:

```
print(f"the maximum number : {train_set['anger'].max()}") => the maximum number : 0.931034
```

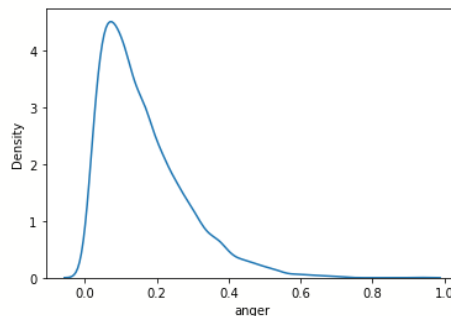
```
print(f"The Minimum number : {train_set['anger'].min()}") =>The Minimum number : 0.0
```

```
print(f"The Mean is : {train_set['anger'].mean()}") => The Mean is : 0.16214185431735986
```

```
print(f"The median is : {train_set['anger'].median()}") => The median is : 0.132466
```

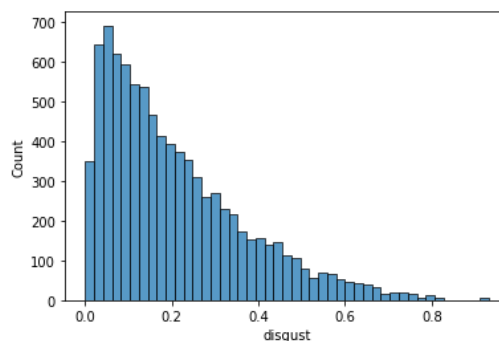
همچنین می توان توزیع هر یک از ستونها را رسم کرد:

```
sns.kdeplot(data=train_set,x='anger');
```



شکل 2.5 : kde(Kernel Density Estimation) ستون 'anger'

```
sns.histplot(data=train_set,x='anger');
```



شکل 2.6 : Count plot ستون 'anger'

با بررسی این تحلیل ها برای سایر ستونهای مطرح شده در بالا، می توان نتیجه گرفت که همگی به صورت تقریبی از توزیع Poisson تبعیت می کنند. بنابراین رفتار تغییرات هر یک از ستون ها مشابه است. حال برای کار با کتابخانه NLTK، نیاز است که مقادیر درون هر یک از 5 ستون نام برده به شکل باینری در آید. به این منظور نیاز است یک Threshold تعیین شود تا مقادیر بیش از آن به عدد 1 و مقادیر کمتر از آن به مقدار 0 نسبت داده شوند.

*از آنجا که توزیع هر 5 ستون بالا یکسان می باشند، فرض می کنیم مقدار Threshold برای هر یک از آنها یکسان می باشد. در این قسمت مقدار Threshold را برابر 0.4 فرض می کنیم :

```
for col in ['anger','fear','joy','disgust','sad']:
    train_set[col]=train_set[col].apply(lambda x: x>= 0.4)
```

```
for col in ['anger','fear','joy','disgust','sad']:
    test_set[col]=test_set[col].apply(lambda x: x>= 0.4)
```

نمایی از دیتاست نهایی برای ارزیابی مدل به صورت زیر می باشد :

	sentiment	anger	fear	joy	disgust	sad
0	False	False	False	False	False	True
1	False	False	False	False	False	False
2	False	False	False	True	True	False
3	False	False	False	False	False	False
4	False	False	False	False	False	True
...
8843	False	False	False	False	False	False
8844	False	False	False	False	False	True
8845	False	False	False	False	False	False
8846	False	False	False	False	False	False
8847	False	False	False	False	False	True

8848 rows x 6 columns

شکل 2.7 : دیتاست ورودی به مدل Naïve Bayes

در نهایت مدل را fit می کنیم :

```
NB = nltk.NaiveBayesClassifier.train(feature_set_train)
```

```

y_pred=[]
y_test=[]
for sample in feature_set_test:
    y_pred.append(NB.classify(sample[0]))
    y_test.append(sample[1])
print("      precision      ,      recall      ,      fscore      ,      support      => \n")
print(f"{precision_recall_fscore_support(y_test, y_pred)}\n")
print(classification_report(y_test, y_pred))

```

The naive Bayes accuracy is: **58.020164986251146**

```

      precision      ,      recall      ,
(array([0.54225352, 0.58587987]), array([0.16382979, 0.89533011]),
      fscore      ,      support
array([0.25163399, 0.70828025]), array([470, 621]))

      precision      recall      f1-score      support
F      0.54      0.16      0.25      470
T      0.59      0.90      0.71      621

      accuracy
macro avg      0.56      0.53      0.48      1091
weighted avg      0.57      0.58      0.51      1091

```

Deleting sentiment column

تمامی مراحل قست قبل را تکرار می کنیم با این تفاوت که دیگر ستون sentiment وجود ندارد:

```

train_set=train_set.drop('sentiment',axis=1)
test_set=test_set.drop('sentiment',axis=1)

```

The naive Bayes accuracy is : **56.827150749802676**

```

      precision      ,      recall      ,
(array([0.75      , 0.56653386]), array([0.01627486, 0.99579832]),
      fscore      ,      support
array([0.03185841, 0.72219401]), array([553, 714]))

      precision      recall      f1-score      support
F      0.75      0.02      0.03      553
T      0.57      1.00      0.72      714

      accuracy
macro avg      0.57      0.51      0.57      1267

```

macro avg	0.66	0.51	0.38	1267
weighted avg	0.65	0.57	0.42	1267

*پس در هنگامی که ستونهای ویژگی ما حالت دودویی باشند، دقت در حالت حفظ ستون sentiment نسبت به حذف آن بهتر می باشد.

Replacing Null values with zeros

در این حالت به جای مقادیر Nan در ستون Sentiment، مقدار 0 را قرار می دهیم و فرض می کنیم جملات Neutral نیز Negative هستند :

```
train_set['sentiment'] = train_set['sentiment'].fillna('NEGATIVE')
test_set['sentiment'] = test_set['sentiment'].fillna('NEGATIVE')
```

The naive Bayes accuracy is: 57.69534333070244

```
precision      ,      recall      ,
(array([0.56390977, 0.57848325]), array([0.13562387, 0.91876751])),
fscore      ,      support
array([0.21865889, 0.70995671]), array([553, 714]))
```

	precision	recall	f1-score	support
F	0.56	0.14	0.22	553
T	0.58	0.92	0.71	714
accuracy			0.58	1267
macro avg	0.57	0.53	0.46	1267
weighted avg	0.57	0.58	0.50	1267

What is the best threshold?

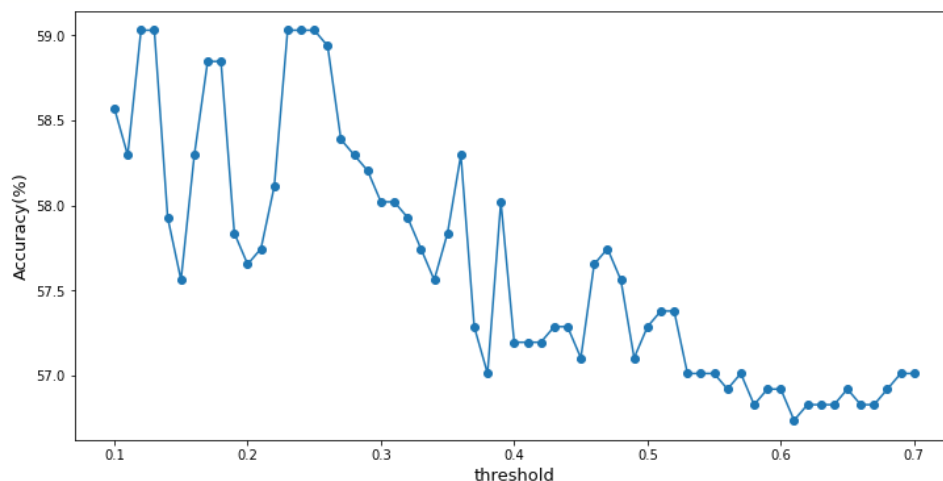
در قسمت اول این بخش، مقدار Threshold را 0.4 در نظر گرفتیم و دقت طبقه بند را محاسبه کردیم. در این قسمت قصد داریم نمودار دقت طبقه بند(Accuracy) را بر حسب مقدار Threshold رسم کنیم (*همانطور که قبلا اشاره شد، مقدار Threshold را برای هر 5 ستون Emotion یکسان در نظر می گیریم).

Acc=[]

```
import numpy as np
for num in np.arange(0.10,0.71,0.01):
    Acc.append(return_acc(num))
```

در نهایت نمودار ذکر شده را رسم می‌کنیم :

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(0.10,0.71,0.01),Acc,marker='o')
plt.xlabel('threshold',size=13)
plt.ylabel('Accuracy(%)',size=13);
```



شکل 2.8 : نمودار دقت طبقه بند بر حسب Threshold

طبق شکل بالا، به طور کلی با افزایش Threshed از مقدار 0.3 به بعد، شاهد افت دقت می‌باشیم.

Sklearn

در این بخش ، به کمک کتابخانه sklearn، تابع Gaussian Naïve Bayes را به کار می‌گیریم. فرق مهم این قسمت با قسمت های قبل در این است که دیگر لزومی ندارد مقادیر ستون های ویژگی ما باینری باشند و ورودی طبقه بند ما یک بردار اعشاری است.

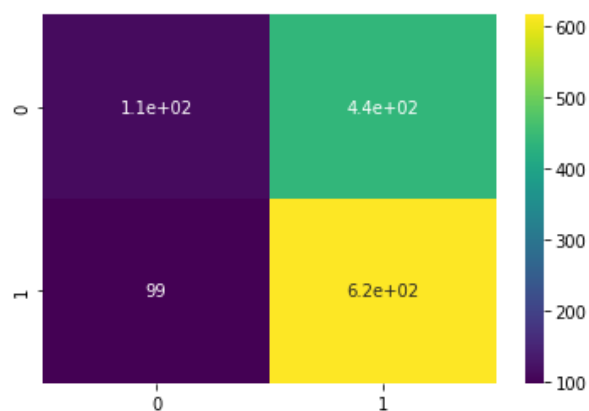
بنابراین تنها کافی است که کتابخانه های کافی را ایجاد کرده و مدل را fit کنیم:

```
from sklearn.metrics import precision_recall_fscore_support,classification_report,accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
y_pred = gnb.fit(tr_set,label_tr).predict(ts_set)
```

The naive Bayes accuracy is : 57.69534333070244

```
precision      ,      recall      ,  
(array([0.53521127, 0.58349146]), array([0.20614828, 0.86134454]),  
  
      fscore      ,      support  
array([0.29765013, 0.69570136]), array([553, 714]))  
  
      precision      recall      f1-score      support  
F      0.54      0.21      0.30      553  
T      0.58      0.86      0.70      714  
  
accuracy      0.58      1267  
macro avg      0.56      0.53      0.50      1267  
weighted avg      0.56      0.58      0.52      1267
```

ماتریس آشفته‌گی مدل به صورت زیر می‌باشد :



شکل 2.9 : ماتریس آشفته‌گی طبقه بند Naïve Bayes

به طور کلی در تمامی مدل ها می‌توان مشاهده کرد که در تشخیص دروغ ضعیف عمل شده است و گویی مانند این است که صرفا حدس زده باشیم (Random guessing).