Naive Bayes Model

In [8]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import BernoulliNB,GaussianNB,MultinomialNB
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [17]:

```
data = pd.read_csv('spamham.csv',encoding='latin1')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   type    5572 non-null   object
 1   email   5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

In [18]:

```
data.shape
```

Out[18]:

```
(5572, 2)
```

In [19]:

```
data.size
```

Out[19]:

```
11144
```

In [20]:

```
data.isnull().sum()
```

Out[20]:

```
type     0
email    0
dtype: int64
```

In [21]:

```
X=data.email
y=data.type
```

In [22]:

```
vector=TfidfVectorizer()
x=vector.fit_transform(X)
feature_vectors=vector.get_feature_names_out()
list(feature_vectors[2600:2700])
```

Out[22]:

```
['diesel',
```

'diet',
'dieting',
'diff',
'differ',
'difference',
'differences',
'different',
'difficult',
'difficulties',
'dificult',
'digi',
'digital',
'digits',
'dignity',
'dileep',
'dime',
'dimension',
'din',
'dine',
'dined',
'dinero',
'ding',
'dining',
'dinner',
'dino',
'dint',
'dip',
'dippeditinadew',
'direct',
'directly',
'director',
'directors',
'dirt',
'dirtiest',
'dirty',
'dis',
'disagreeable',
'disappeared',
'disappointment',
'disaster',
'disasters',
'disastrous',
'disc',
'disclose',
'disconnect',
'disconnected',
'discount',
'discreet',
'discuss',
'discussed',
'diseases',
'disk',
'dislikes',
'dismay',
'dismissial',
'display',
'distance',
'distract',
'disturb',
'disturbance',
'disturbing',
'ditto',
'divert',
'division',
'divorce',
'diwali',
'dizzamn',
'dizzee',
'dl',
'dled',
'dlf',
'dload',

```
 'dnot',
 'dnt',
 'do',
 'dob',
 'dobby',
 'doc',
 'dock',
 'docks',
 'docs',
 'doctor',
 'doctors',
 'documents',
 'dodda',
 'dodgey',
 'does',
 'doesdiscount',
 'doesn',
 'doesnt',
 'doesnåõt',
 'dog',
 'dogbreath',
 'dogg',
 'doggin',
 'dogging',
 'doggy',
 'dogs',
 'dogwood']
```

In [23]:

```
len(feature_vectors)
```

Out[23]:

```
8672
```

In [24]:

```
x.toarray()
```

Out[24]:

```
array([[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.]])
```

In [25]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

In [26]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4179, 8672)
(1393, 8672)
(4179,)
(1393,)
```

In [27]:

```
model=BernoulliNB()
model.fit(x_train,y_train)
predict=model.predict(x_test)
predict
```

Out[27]:

```
array(['ham', 'ham', 'ham', ..., 'spam', 'ham', 'ham'], dtype='<U4')
```

In [28]:

```
print(accuracy_score(y_test,predict))
print(model.score(x_train,y_train))
print(model.score(x_test,y_test))
```

```
0.9834888729361091
0.9856424982053122
0.9834888729361091
```

In [29]:

```
new_data=pd.Series("hello this is good and after few days it is good")
print(new_data)
new_data=vector.transform(new_data)
print(new_data)
```

```
0    hello this is good and after few days it is good
dtype: object
  (0, 7669) 0.23589858040024975
  (0, 4218) 0.19632669583559223
  (0, 4206) 0.36718517593367317
  (0, 3814) 0.3552869742023304
  (0, 3576) 0.5103008400069725
  (0, 3174) 0.3552869742023304
  (0, 2459) 0.3580627288559732
  (0, 1084) 0.17995345402398116
  (0, 963) 0.31229146008739916
```

In [30]:

```
new_data.shape
```

Out[30]:

```
(1, 8672)
```

In [31]:

```
model.predict(new_data)
```

Out[31]:

```
array(['ham'], dtype='<U4')
```

In [32]:

```
import pandas as pd
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [33]:

```
d = pd.read_csv('sampletext.csv',encoding='latin-1')
d.head()
```

Out[33]:

	t	msg
0	she is good	pos
1	not well	neg
2	not able to say good	neg
3	I think I am fine in some times	pos

4	too superb	pos

In [34]:

```
print(data.size)
print(data.info())
print(data.shape)
X=d.t
print(X)
print(d.msg)
```

```
11144
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   type    5572 non-null   object
 1   email   5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
None
(5572, 2)
0                          she is good
1                             not well
2                 not able to say good
3    I think I am fine in some times
4                           too superb
Name: t, dtype: object
0    pos
1    neg
2    neg
3    pos
4    pos
Name: msg, dtype: object
```

In [35]:

```
vect = TfidfVectorizer()
x = vect.fit_transform(X)
feature_name = vect.get_feature_names_out()
```

In [36]:

```
print(feature_name)
print(len(feature_name))
print(x)
print(x[1])
```

```
['able' 'am' 'fine' 'good' 'in' 'is' 'not' 'say' 'she' 'some' 'superb'
 'think' 'times' 'to' 'too' 'well']
16
  (0, 3) 0.49552379079705033
  (0, 5) 0.6141889663426562
  (0, 8) 0.6141889663426562
  (1, 15) 0.7782829228046183
  (1, 6) 0.6279137616509933
  (2, 7) 0.4821401170833009
  (2, 13) 0.4821401170833009
  (2, 0) 0.4821401170833009
  (2, 6) 0.3889876106617681
  (2, 3) 0.3889876106617681
  (3, 12) 0.40824829046386296
  (3, 9) 0.40824829046386296
  (3, 4) 0.40824829046386296
  (3, 2) 0.40824829046386296
  (3, 1) 0.40824829046386296
  (3, 11) 0.40824829046386296
  (4, 10) 0.7071067811865475
  (4, 14) 0.7071067811865475
  (0, 15) 0.7782829228046183
```

(0, 6) 0.6279137616509933

In [37]:

```
x=x.toarray()
x
```

Out[37]:

```
array([[0.        , 0.        , 0.        , 0.49552379, 0.        ,
        0.61418897, 0.        , 0.        , 0.61418897, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.62791376, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.77828292],
       [0.48214012, 0.        , 0.        , 0.38898761, 0.        ,
        0.        , 0.38898761, 0.48214012, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.48214012, 0.        ,
        0.        ],
       [0.        , 0.40824829, 0.40824829, 0.        , 0.40824829,
        0.        , 0.        , 0.        , 0.        , 0.40824829,
        0.        , 0.40824829, 0.40824829, 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.70710678, 0.        , 0.        , 0.        , 0.70710678,
        0.        ]])
```

In [38]:

```
model = BernoulliNB()
model.fit(x_train,y_train)
y_predict = model.predict(x_test)
accuracy_score(y_test,y_predict)
```

Out[38]:

```
0.9834888729361091
```

In [39]:

```
print(model.score(x_train,y_train))
```

```
0.9856424982053122
```

In [40]:

```
new_data = pd.Series('How good good get good good good good')
new_data
```

Out[40]:

```
0    How good good get good good good good
dtype: object
```

In [41]:

```
new_data.shape
```

Out[41]:

```
(1,)
```

In [42]:

```
data = pd.read_csv('spamham.csv',encoding='latin-1')
data.head()
```

Out[42]:

	type	email

	type	email
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...

In [43]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   type    5572 non-null   object
 1   email   5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

In [44]:

```
X = data.email
y = data.type
```

In [45]:

```
vect = TfidfVectorizer()
x = vect.fit_transform(X)
feature_name = vect.get_feature_names_out()
feature_name
```

Out[45]:

```
array(['00', '000', '000pes', ..., 'ûïharry', 'ûò', 'ûówell'],
      dtype=object)
```

In [46]:

```
x
```

Out[46]:

```
<5572x8672 sparse matrix of type '<class 'numpy.float64'>'
 with 73916 stored elements in Compressed Sparse Row format>
```

In [47]:

```
feature_name[2040:2050]
```

Out[47]:

```
array(['christ', 'christians', 'christmas', 'christmassy', 'chuck',
       'chuckin', 'church', 'ciao', 'cine', 'cinema'], dtype=object)
```

In [48]:

```
len(feature_name)
```

Out[48]:

```
8672
```

In [49]:

```
x=x.toarray()
x
```

Out[49]:

```
array([[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.]])
```

In [50]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
```

In [51]:

```
model = BernoulliNB()
model.fit(x_train,y_train)
y_predict = model.predict(x_test)
accuracy_score(y_test,y_predict)
```

Out[51]:

```
0.9820627802690582
```

In [52]:

```
model.score(x_train,y_train)
```

Out[52]:

```
0.9876598608929773
```

In [53]:

```
new_data = vect.transform(new_data)
new_data
```

Out[53]:

```
<1x8672 sparse matrix of type '<class 'numpy.float64'>'
 with 3 stored elements in Compressed Sparse Row format>
```

In [54]:

```
model.predict(new_data)
```

Out[54]:

```
array(['ham'], dtype='<U4')
```