TextClasificationusingNaiveBayes

May 16, 2025

[1]:
```
#importing required libraries
import numpy as np
import pandas as pd
import os
from collections import Counter
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
```

[2]:
```
# making stopwords array
stopwords = list(ENGLISH_STOP_WORDS);
print(len(stopwords));
```

```
318
```

[3]:
```
# data preprocessing
# making vocabulary dictionary
mainfolder = "./twenty+newsgroups/20_newsgroups";
vocabulary = Counter();
for root,directory,files in os.walk(mainfolder):
    for file in files:
        filepath = os.path.join(root,file)
        with open(filepath,"r",encoding = "latin-1") as f:
            content = f.read();
            content = ' '.join(word for word in content.split() if word not in␣
 ↪stopwords);
            content = content.split();
            vocabulary.update(content);
print(len(vocabulary));
```

```
463767
```

[4]:
```
# list of classes
yset = ["alt.atheism","comp.graphics","comp.os.ms-windows.misc","comp.sys.ibm.
 ↪pc.hardware","comp.sys.mac.hardware","comp.windows.x","misc.forsale",
       "rec.autos","rec.motorcycles",
       "rec.sport.baseball","rec.sport.hockey","sci.crypt","sci.
 ↪electronics","sci.med","sci.space","soc.religion.christian","talk.politics.
 ↪guns",
       "talk.politics.mideast","talk.politics.misc","talk.religion.misc"];
```

```
s = set(yset);
```

[5]:
```
# extracting top k words from vocabulary
k = 50000
vocabulary = dict(sorted(vocabulary.items(),key = lambda x : x[1],reverse =␣
 ↪True)[:k]);
print(len(vocabulary));
```

```
50000
```

[6]:
```
# creating dictionarylist for each document
dflist = [];
y = [];
for root,directory,files in os.walk(mainfolder):
    for file in files:
        filepath = os.path.join(root,file)
        found = False;
        for cls in s:
            if cls in root:
                y.append(cls);
                found = True;
                break;
        if (found == False):
            y.append("unknown");
        try :
            with open(filepath,"r",encoding = "latin-1") as f:
                content = f.read();
                content = ' '.join(word for word in content.split() if word not␣
 ↪in stopwords);
                content = content.split();
                wordcount = Counter(content);
                # print("wordcount",len(wordcount));
                row = {word : wordcount.get(word,0) for word in vocabulary.
 ↪keys()};
                # print("row",len(row));
                dflist.append(row);
                # print("row appended successfully",end = " ");
        except Exception as e:
            print("error",e);
```

[7]:
```
print(len(dflist));
```

```
20000
```

[8]:
```
keys = vocabulary.keys();
print(len(keys));
```

```
50000
```

[9]:
```
# creating dataframe
df = pd.DataFrame(dflist);
print(df.shape);
```

```
(20000, 50000)
```

[10]:
```
print(len(y));
s = set(y);
print(s);
```

```
20000
{'talk.politics.misc', 'sci.crypt', 'rec.autos', 'alt.atheism',
'talk.politics.mideast', 'rec.motorcycles', 'comp.graphics', 'sci.med',
'comp.windows.x', 'comp.sys.ibm.pc.hardware', 'soc.religion.christian',
'misc.forsale', 'sci.electronics', 'comp.sys.mac.hardware', 'unknown',
'comp.os.ms-windows.misc', 'rec.sport.baseball', 'talk.politics.guns',
'talk.religion.misc', 'rec.sport.hockey', 'sci.space'}
```

[11]:
```
print(df.head());
```

```
   I  >  The  Subject:  From:  Date:  Newsgroups:  Message-ID:  Lines:  Path:  \
0  0  0    0         0      0      0            0            0       0      0
1  2  0    0         1      1      1            1            1       1      1
2  3  7    1         1      1      1            1            1       1      1
3  2  8    1         1      1      1            1            1       1      1
4  2  0    1         1      1      1            1            1       1      1

   …  traffic?  >Nick  Lidstrom  Numminen  Ramage  $27  Comment  VOX:  84.0  \
0  …         0      0         0         0       0    0        0     0     0
1  …         0      0         0         0       0    0        0     0     0
2  …         0      0         0         0       0    0        0     0     0
3  …         0      0         0         0       0    0        0     0     0
4  …         0      0         0         0       0    0        0     0     0

   (Yadallee
0          0
1          0
2          0
3          0
4          0

[5 rows x 50000 columns]
```

[12]:
```
x = df;
y = np.array(y);
```

[13]:
```
print(x.shape,y.shape);
```

```
(20000, 50000) (20000,)
```

[14]:
```
print(df.head())
print(y[:5])
```

```
   I  >  The  Subject:  From:  Date:  Newsgroups:  Message-ID:  Lines:  Path:  \
0  0  0    0         0      0      0            0            0       0      0
1  2  0    0         1      1      1            1            1       1      1
2  3  7    1         1      1      1            1            1       1      1
3  2  8    1         1      1      1            1            1       1      1
4  2  0    1         1      1      1            1            1       1      1

   …  traffic?  >Nick  Lidstrom  Numminen  Ramage  $27  Comment  VOX:  84.0  \
0  …         0      0         0         0       0    0        0     0     0
1  …         0      0         0         0       0    0        0     0     0
2  …         0      0         0         0       0    0        0     0     0
3  …         0      0         0         0       0    0        0     0     0
4  …         0      0         0         0       0    0        0     0     0

   (Yadallee
0          0
1          0
2          0
3          0
4          0

[5 rows x 50000 columns]
['unknown' 'talk.politics.mideast' 'talk.politics.mideast'
 'talk.politics.mideast' 'talk.politics.mideast']
```

[15]:
```
# splitting data
from sklearn import model_selection as ms
xtrain,xtest,ytrain,ytest = ms.train_test_split(x,y,test_size = 0.
  ↪25,random_state = 0);
print(xtrain.shape,ytrain.shape,xtest.shape,ytest.shape);
```

```
(15000, 50000) (15000,) (5000, 50000) (5000,)
```

[16]:
```
# using naivebayes classifier from sklearn
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB();
clf.fit(xtrain,ytrain);
```

[17]:
```
ypred = clf.predict(xtest);
print(ypred.shape);
```

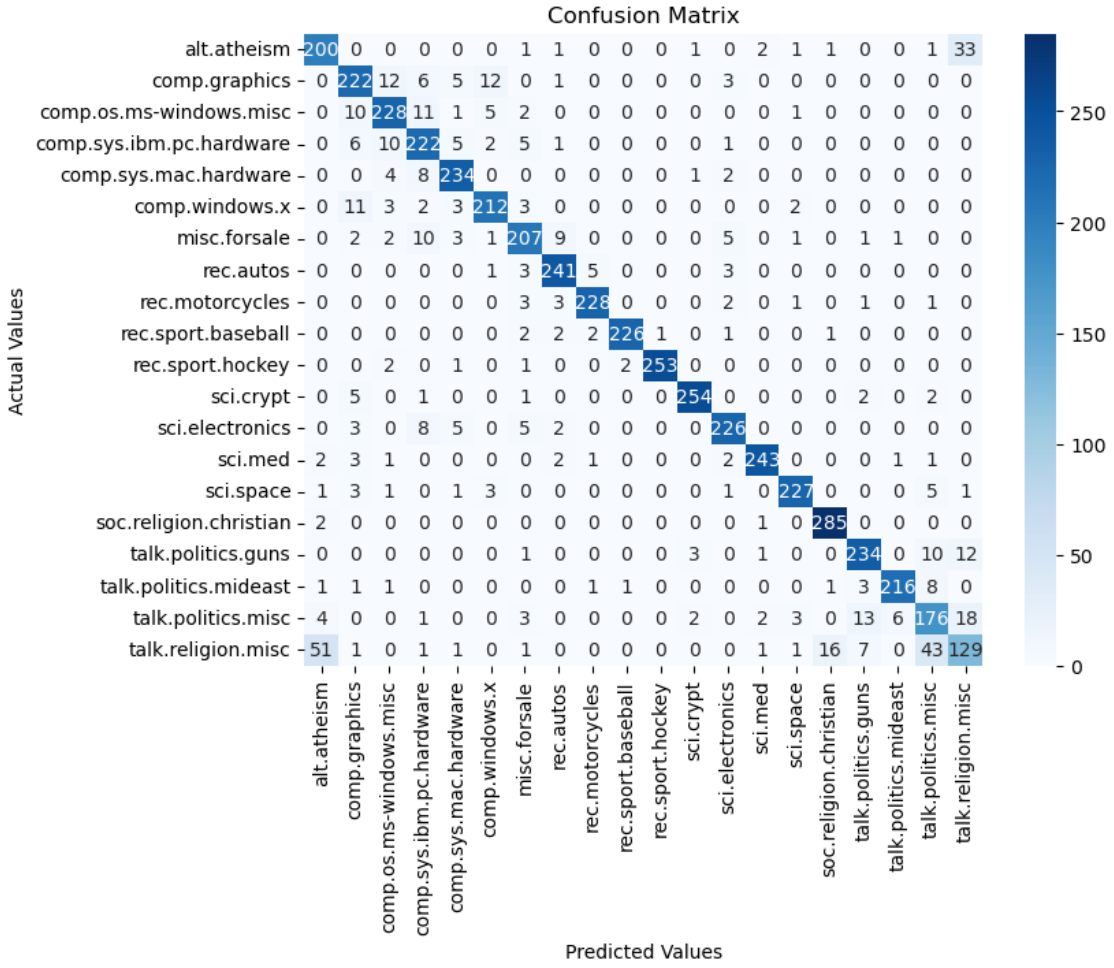
```
(5000,)
```

[18]:
```
# printing score
from sklearn.metrics import␣
  ↪accuracy_score,confusion_matrix,classification_report
```

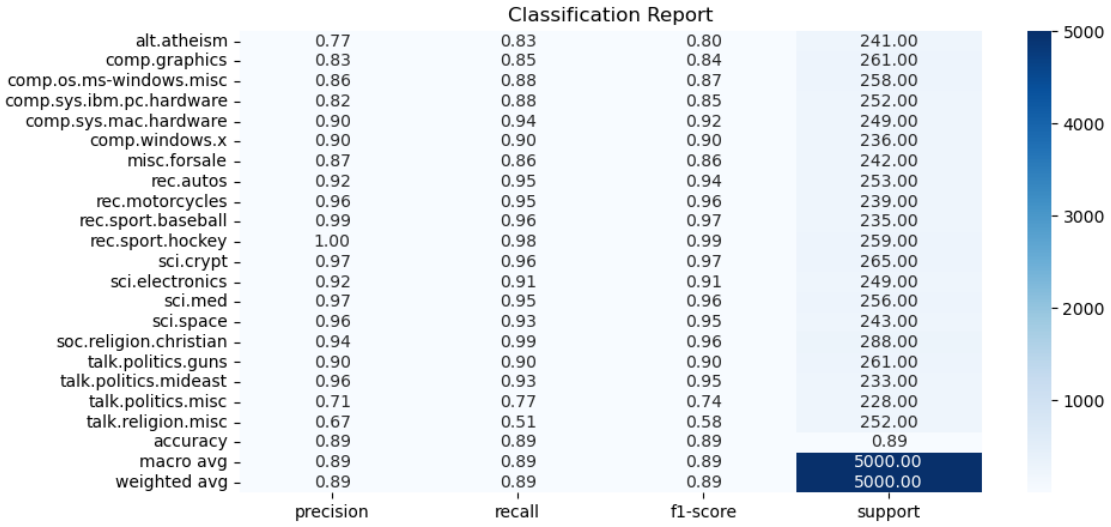
```
score = accuracy_score(ytest,ypred);
print(score);
```

0.8926

[19]:
```
# printing confusion matrix
import seaborn as sns
from matplotlib import pyplot as plt
cm = confusion_matrix(ytest,ypred);
plt.figure(figsize = (8,6));
sns.heatmap(cm,annot = True,fmt = "d",cmap = "Blues",xticklabels =␣
 ↪yset,yticklabels = yset);
plt.title("Confusion Matrix");
plt.xlabel("Predicted Values");
plt.ylabel("Actual Values");
```



[20]:
```
# printing classification report
report = classification_report(ytest,ypred,output_dict = True,target_names =␣
 ↪yset);
df = pd.DataFrame(report).transpose();
plt.figure(figsize = (10,5));
sns.heatmap(df,annot = True,fmt = ".2f",cmap = "Blues");
plt.title("Classification Report");
```



[21]:
```
# implementing Multinomial Naive Bayes from Scratch
class MultinomialNaiveBayes:
    def __init__(self,alpha = 1):
        self.alpha = alpha;
        self.classpriors = None;
        self.classes = None;
        self.featureprob = None;
    def fit(self,x,y):
        x = np.array(x);
        y = np.array(y);
        self.classes = np.unique(y);
        self.numclasses = len(self.classes);
        self.numfeatures = x.shape[1];
        self.classpriors = {};
        self.featureprob = {};
        for c in self.classes:
            xc = x[y == c];
            self.classpriors[c] = xc.shape[0]/x.shape[0];
            self.featureprob[c] = (xc.sum(axis = 0) + self.alpha)/(xc.sum() +␣
 ↪self.alpha*self.numfeatures);
    def likelyhood(self,x,Class):
```

```
        return np.prod(np.power(self.featureprob[Class],x)*np.power(1 - self.
↪featureprob[Class],1 - x));
    def predict(self,x):
        x = np.array(x);
        ypred = [];
        for xi in x:
            jointlikelyhood = [];
            for c in self.classes:
                likelyhood1 = self.classpriors[c]*self.likelyhood(xi,c);
                jointlikelyhood.append(likelyhood1);
            ypred.append(self.classes[np.argmax(jointlikelyhood)]);
        return np.array(ypred);
```

[22]:
```
clf = MultinomialNaiveBayes(alpha = 1);
clf.fit(xtrain,ytrain);
```

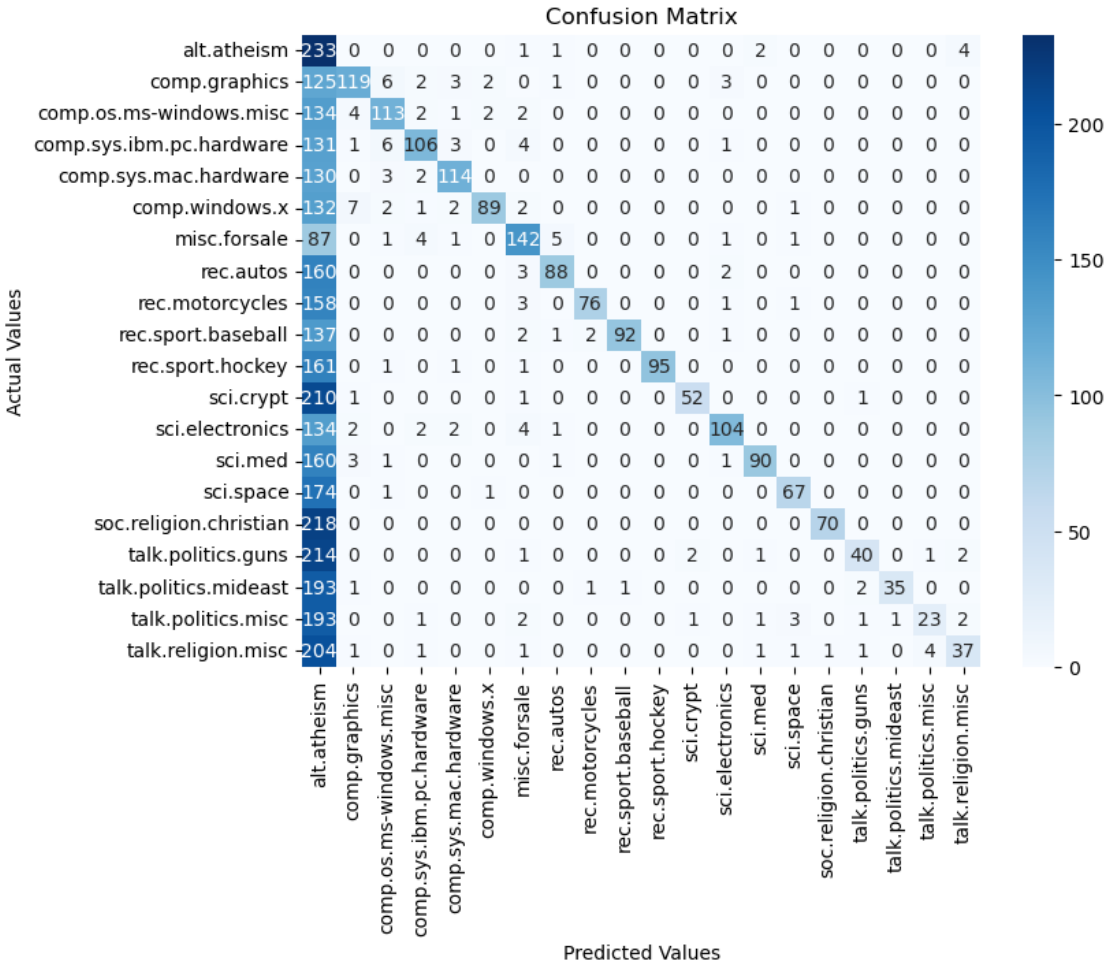
[23]:
```
ypred = clf.predict(xtest);
print(ypred.shape);
```

```
(5000,)
```

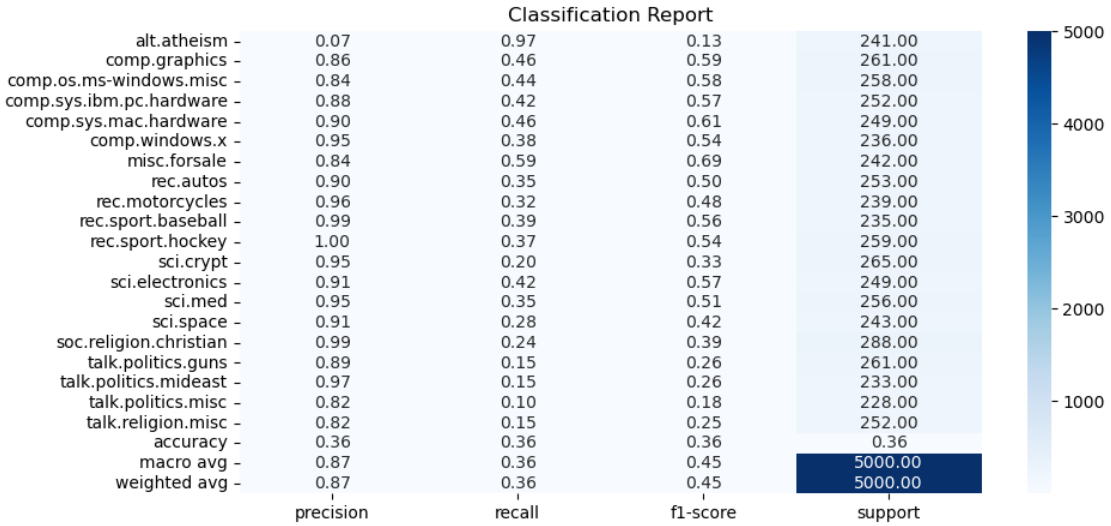
[24]:
```
# printing score
score = accuracy_score(ytest,ypred);
print(score);
```

```
0.357
```

[25]:
```
# printing confusion matrix
cm = confusion_matrix(ytest,ypred);
plt.figure(figsize = (8,6));
sns.heatmap(cm,annot = True,fmt = "d",cmap = "Blues",xticklabels =␣
↪yset,yticklabels = yset);
plt.title("Confusion Matrix");
plt.xlabel("Predicted Values");
plt.ylabel("Actual Values");
```



[26]:
```
# printing classification report
report = classification_report(ytest,ypred,output_dict = True,target_names =␣
 ↪yset);
df = pd.DataFrame(report).transpose();
plt.figure(figsize = (10,5));
sns.heatmap(df,annot = True,fmt = ".2f",cmap = "Blues");
plt.title("Classification Report");
```


Classification Report
alt.atheism 0.07 0.97 0.13 241.00
comp.graphics 0.86 0.46 0.59 261.00
comp.os.ms-windows.misc 0.84 0.44 0.58 258.00
comp.sys.ibm.pc.hardware 0.88 0.42 0.57 252.00
comp.sys.mac.hardware 0.90 0.46 0.61 249.00
comp.windows.x 0.95 0.38 0.54 236.00
misc.forsale 0.84 0.59 0.69 242.00
rec.autos 0.90 0.35 0.50 253.00
rec.motorcycles 0.96 0.32 0.48 239.00
rec.sport.baseball 0.99 0.39 0.56 235.00
rec.sport.hockey 1.00 0.37 0.54 259.00
sci.crypt 0.95 0.20 0.33 265.00
sci.electronics 0.91 0.42 0.57 249.00
sci.med 0.95 0.35 0.51 256.00
sci.space 0.91 0.28 0.42 243.00
soc.religion.christian 0.99 0.24 0.39 288.00
talk.politics.guns 0.89 0.15 0.26 261.00
talk.politics.mideast 0.97 0.15 0.26 233.00
talk.politics.misc 0.82 0.10 0.18 228.00
talk.religion.misc 0.82 0.15 0.25 252.00
accuracy 0.36 0.36 0.36 0.36
macro avg 0.87 0.36 0.45 5000.00
weighted avg 0.87 0.36 0.45 5000.00
precision
recall
f1-score
support
5000
4000
3000
2000
1000