In [1]:
```
import pandas as pd
import numpy as np
import warnings
import re
warnings.filterwarnings('ignore')
trainset=pd.read_csv('D:\PyCharm Community Edition 2020.1.3\datasets\\train.csv')
testset=pd.read_csv('D:\PyCharm Community Edition 2020.1.3\datasets\\test.csv')
dataset=pd.concat([trainset,testset],ignore_index=True)
dataset.head()
```

Out[1]:

| | text | sentiment |
|---|---|---|
| 0 | Now, I won't deny that when I purchased this o... | neg |
| 1 | The saddest thing about this "tribute" is that... | neg |
| 2 | Last night I decided to watch the prequel or s... | neg |
| 3 | I have to admit that i liked the first half of... | neg |
| 4 | I was not impressed about this film especially... | neg |

In [2]:
```
#split and extract words
data_senti=dataset.copy(deep=True)
for i,comments in enumerate(dataset['text']):
    data_senti['text'].iloc[i]=re.sub("[^a-zA-Z]"," ",comments)
    data_senti['sentiment'].iloc[i]=dataset['sentiment'].iloc[i]
data_senti.head()
```

Out[2]:

| | text | sentiment |
|---|---|---|
| 0 | Now I won t deny that when I purchased this o... | neg |
| 1 | The saddest thing about this tribute is that... | neg |
| 2 | Last night I decided to watch the prequel or s... | neg |
| 3 | I have to admit that i liked the first half of... | neg |
| 4 | I was not impressed about this film especially... | neg |

In [12]:
```
data_senti=data_senti.replace("neg",-1)
data_senti=data_senti.replace("pos",1)
```

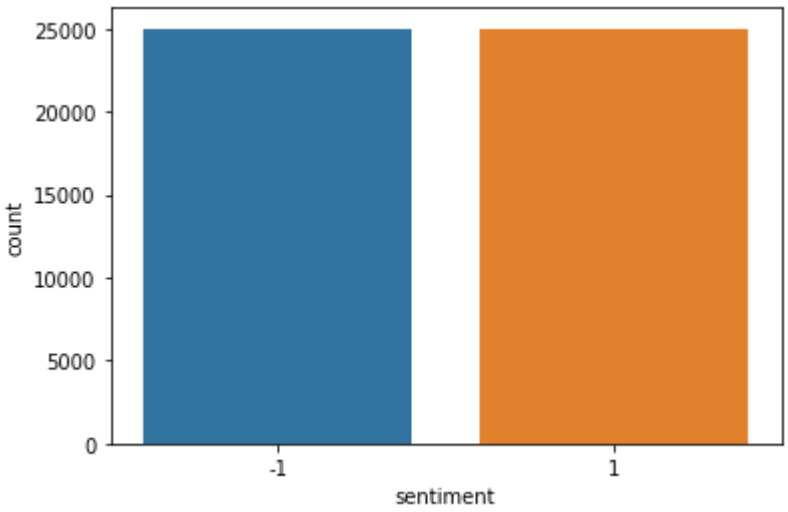
In [13]:
```
data=data_senti.copy(deep=True)
data.head()
```

Out[13]:

| | text | sentiment |
|---|---|---|
| 0 | Now I won t deny that when I purchased this o... | -1 |
| 1 | The saddest thing about this tribute is that... | -1 |
| 2 | Last night I decided to watch the prequel or s... | -1 |
| 3 | I have to admit that i liked the first half of... | -1 |
| 4 | I was not impressed about this film especially... | -1 |

In [14]:
```
##check if data is balanced

import seaborn as sns
sns.countplot(x='sentiment', data=data_senti)
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2cb17c89588>
```




In [15]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['sentim
ent'], test_size=0.2, random_state=0)
```

test_word_senti=testset.copy(deep=True) for i,comments in enumerate(testset['text']):
test_word_senti['text'].iloc[i]=re.sub("[^a-zA-Z]"," ",comments).split()
test_word_senti['sentiment'].iloc[i]=testset['sentiment'].iloc[i] test_word_senti.head()

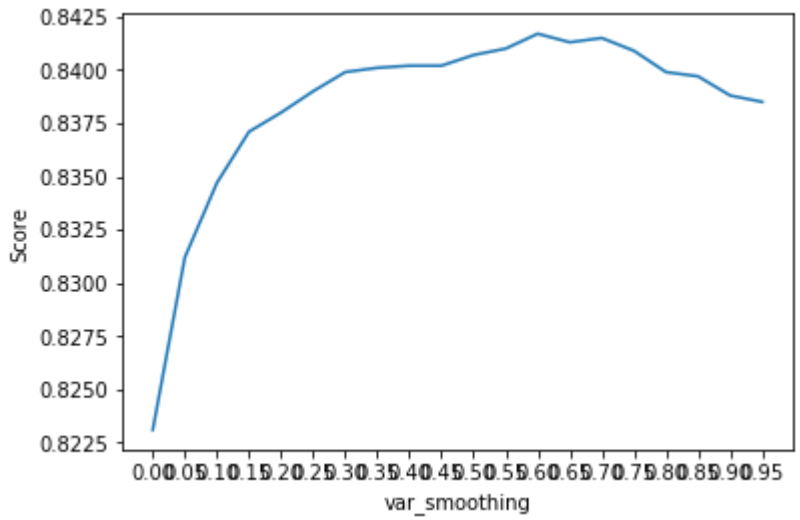
In [16]:
```
import nltk
#nltk.download('stopwords')
```

In [17]:
```
from nltk.corpus import stopwords
en_stops = set(stopwords.words('english'))
```

In [ ]:
```
def delet_stw(sets,n_ds):
    for i,strings in enumerate(sets['text']):
        n_string=[]
        for words in strings:
            if words not in en_stops:
                n_string.append(words)
        n_ds[i]=n_string
    return n_ds
```

In [18]:
```
##td-idf transformation
tf_x_train=X_train.copy()
tf_x_test=X_test.copy()
from sklearn.feature_extraction.text import TfidfVectorizer
tfidfconverter = TfidfVectorizer(sublinear_tf=True,max_features=2000, min_df=5
, max_df=0.7,stop_words=en_stops)
tf_x_train =tfidfconverter.fit_transform(tf_x_train)#train the vectorizer, bui
ld the vocabulary
tf_x_test=tfidfconverter.transform(tf_x_test)
```

In [19]:
```
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn import preprocessing
from sklearn.metrics import precision_recall_curve
from sklearn.naive_bayes import GaussianNB
lists=np.arange(0,1,0.05)
scoreList = []
accuracies = {}
for i in lists:
    clf = GaussianNB(var_smoothing=i)
    clf.fit(tf_x_train.toarray(),y_train)
    scoreList.append(clf.score(tf_x_test.toarray(),y_test))

plt.plot(np.arange(0,1,0.05), scoreList)
plt.xticks(np.arange(0,1,0.05))
plt.xlabel("var_smoothing")
plt.ylabel("Score")
plt.show()
```




In [20]:
```
from sklearn.metrics import classification_report
clf = GaussianNB(var_smoothing=0.01)
clf.fit(tf_x_train.toarray(),y_train)
predictions = clf.predict(tf_x_test.toarray())
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

          -1       0.81      0.84      0.83      4925
           1       0.84      0.81      0.82      5075

    accuracy                           0.82     10000
   macro avg       0.83      0.83      0.82     10000
weighted avg       0.83      0.82      0.82     10000
```

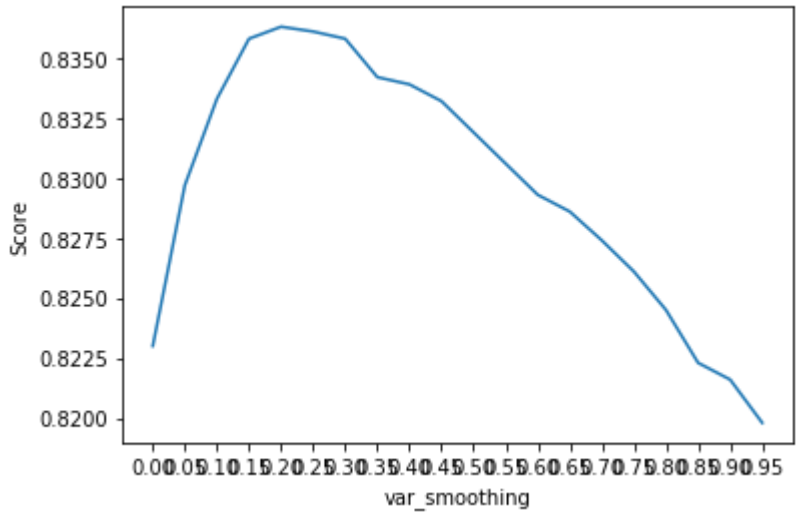
In [21]:
```
from sklearn.naive_bayes import MultinomialNB
model=MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)
model.fit(tf_x_train,y_train)
predictions = model.predict(tf_x_test)
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

          -1       0.84      0.86      0.85      4925
           1       0.86      0.84      0.85      5075

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000
```

In [22]:
```
#conda install -c glemaitre imbalanced-learn
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
#import RandomUnderSampler from imblearn
undersample = RandomUnderSampler(sampling_strategy='majority')
X_under, y_under = undersample.fit_resample(data, data['sentiment'])
```

In [24]:

```
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['sentim
ent'], test_size=0.2, random_state=0)
##td-idf transformation
tf_x_train=X_train.copy()
tf_x_test=X_test.copy()
tfidfconverter = TfidfVectorizer(sublinear_tf=True,max_features=2000, min_df=5
, max_df=0.7)
tf_x_train =tfidfconverter.fit_transform(tf_x_train)#train the vectorizer, bui
ld the vocabulary
tf_x_test=tfidfconverter.transform(tf_x_test)
lists=np.arange(0,1,0.05)
scoreList = []
accuracies = {}
for i in lists:
    clf = GaussianNB(var_smoothing=i)
    clf.fit(tf_x_train.toarray(),y_train)
    scoreList.append(clf.score(tf_x_test.toarray(),y_test))

plt.plot(np.arange(0,1,0.05), scoreList)
plt.xticks(np.arange(0,1,0.05))
plt.xlabel("var_smoothing")
plt.ylabel("Score")
plt.show()
clf = GaussianNB(var_smoothing=0.01)
clf.fit(tf_x_train.toarray(),y_train)
predictions = clf.predict(tf_x_test.toarray())
print(classification_report(y_test,predictions))
model=MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)
model.fit(tf_x_train,y_train)
predictions = model.predict(tf_x_test)
print(classification_report(y_test,predictions))
```




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| -1 | 0.81 | 0.84 | 0.83 | 4925 |
| 1 | 0.84 | 0.81 | 0.82 | 5075 |
| accuracy | | | 0.82 | 10000 |
| macro avg | 0.83 | 0.82 | 0.82 | 10000 |
| weighted avg | 0.83 | 0.82 | 0.82 | 10000 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| -1 | 0.84 | 0.85 | 0.85 | 4925 |
| 1 | 0.86 | 0.84 | 0.85 | 5075 |
| accuracy | | | 0.85 | 10000 |
| macro avg | 0.85 | 0.85 | 0.85 | 10000 |
| weighted avg | 0.85 | 0.85 | 0.85 | 10000 |