In [ ]:
```
#Name:Ankita
#OASIS INFOBYTE
#DATA SCIENCE TASK 2:EMAIL SPAM DETECTION
```

In [38]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [39]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [40]:
```
emails = pd.read_csv("emails.csv")
```

In [41]:
```
emails
```

Out[41]:

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1
...	...	...
5723	Subject: re : research and development charges...	0
5724	Subject: re : receipts from visit jim , than...	0
5725	Subject: re : enron case study update wow ! a...	0
5726	Subject: re : interest david , please , call...	0
5727	Subject: news : aurora 5 . 2 update aurora ve...	0

5728 rows × 2 columns

In [42]:
```
emails.describe()
```

Out[42]:

	spam
count	5728.000000
mean	0.238827
std	0.426404
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

In [43]:
```
emails.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5728 entries, 0 to 5727
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    5728 non-null   object
 1   spam    5728 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 89.6+ KB
```

In [44]:
```
emails['spam'].value_counts()
```

Out[44]:
```
0    4360
1    1368
Name: spam, dtype: int64
```

In [45]:
```
emails.notnull()
```

Out[45]:

	text	spam
0	True	True
1	True	True
2	True	True
3	True	True
4	True	True
...	...	...
5723	True	True
5724	True	True
5725	True	True
5726	True	True
5727	True	True

5728 rows × 2 columns

In [46]:
```
emails["text"] = emails["text"].str.lower()
emails.head()
```

Out[46]:

	text	spam
0	subject: naturally irresistible your corporate...	1
1	subject: the stock trading gunslinger fanny i...	1
2	subject: unbelievable new homes made easy im ...	1
3	subject: 4 color printing special request add...	1
4	subject: do not have money , get software cds ...	1

In [47]:
```
import string
PUNCT_TO_REMOVE = string.punctuation
def remove_punctuation(text):
    """custom function to remove the punctuation"""
    return text.translate(str.maketrans('', '', PUNCT_TO_REMOVE))
```

```
emails["text"] = emails["text"].apply(lambda text: remove_punctuation(text))
emails['text']
```

Out[47]:
```
0       subject naturally irresistible your corporate ...
1       subject the stock trading gunslinger  fanny is...
2       subject unbelievable new homes made easy  im w...
3       subject 4 color printing special  request addi...
4       subject do not have money  get software cds fr...
                              ...
5723    subject re  research and development charges t...
5724    subject re  receipts from visit  jim   thanks ...
5725    subject re  enron case study update  wow  all ...
5726    subject re  interest  david   please  call shi...
5727    subject news  aurora 5  2 update  aurora versi...
Name: text, Length: 5728, dtype: object
```

In [48]:
```
from nltk.corpus import stopwords
STOPWORDS = set(stopwords.words('english'))
STOPWORDS.add('subject')
def remove_stopwords(text):
    """custom function to remove the stopwords"""
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])

emails["text"] = emails["text"].apply(lambda text: remove_stopwords(text))
emails['text']
```

Out[48]:
```
0       naturally irresistible corporate identity lt r...
1       stock trading gunslinger fanny merrill muzo co...
2       unbelievable new homes made easy im wanting sh...
3       4 color printing special request additional in...
4       money get software cds software compatibility ...
                              ...
5723    research development charges gpg forwarded shi...
5724    receipts visit jim thanks invitation visit lsu...
5725    enron case study update wow day super thank mu...
5726    interest david please call shirley crenshaw as...
5727    news aurora 5 2 update aurora version 5 2 fast...
Name: text, Length: 5728, dtype: object
```

In [49]:
```
import nltk
```

In [50]:
```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ankii\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[50]:
```
True
```

In [51]:
```
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\ankii\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[51]:
```
True
```

In [52]:
```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
def lemmatize_words(text):
    return " ".join([lemmatizer.lemmatize(word) for word in text.split()])

emails["text"] = emails["text"].apply(lambda text: lemmatize_words(text))
emails.head()
```

Out[52]:

	text	spam
0	naturally irresistible corporate identity lt r...	1
1	stock trading gunslinger fanny merrill muzo co...	1
2	unbelievable new home made easy im wanting sho...	1
3	4 color printing special request additional in...	1
4	money get software cd software compatibility g...	1

In [53]:
```
X = emails['text']
y = emails['spam']
```

In [54]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
```

In [55]:
```
print(X_train.shape)
print(X_test.shape)
```

```
(4009,)
(1719,)
```

In [56]:
```
#Tfidf vectorization of data
```

In [57]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

In [58]:
```
print(X_train.shape)
print(X_test.shape)
```

```
(4009, 29542)
(1719, 29542)
```

In [59]:
```
#multinomial naive bayes for class
```

In [60]:
```
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(X_train, y_train)
```

Out[60]:
```
MultinomialNB()
```

In [61]:
```
y_pred = clf.predict(X_train)
```

In [62]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
trainacc = accuracy_score(y_train, y_pred)
trainf1 = f1_score(y_train, y_pred)
print(trainacc)
print(trainf1)
```

```
0.934647044150661
0.8427370948379351
```

In [63]:
```
y_pred_test = clf.predict(X_test)
```

In [64]:
```
testacc = accuracy_score(y_test, y_pred_test)
f1test = f1_score(y_test, y_pred_test)
```

```
print(testacc)
print(f1test)
```

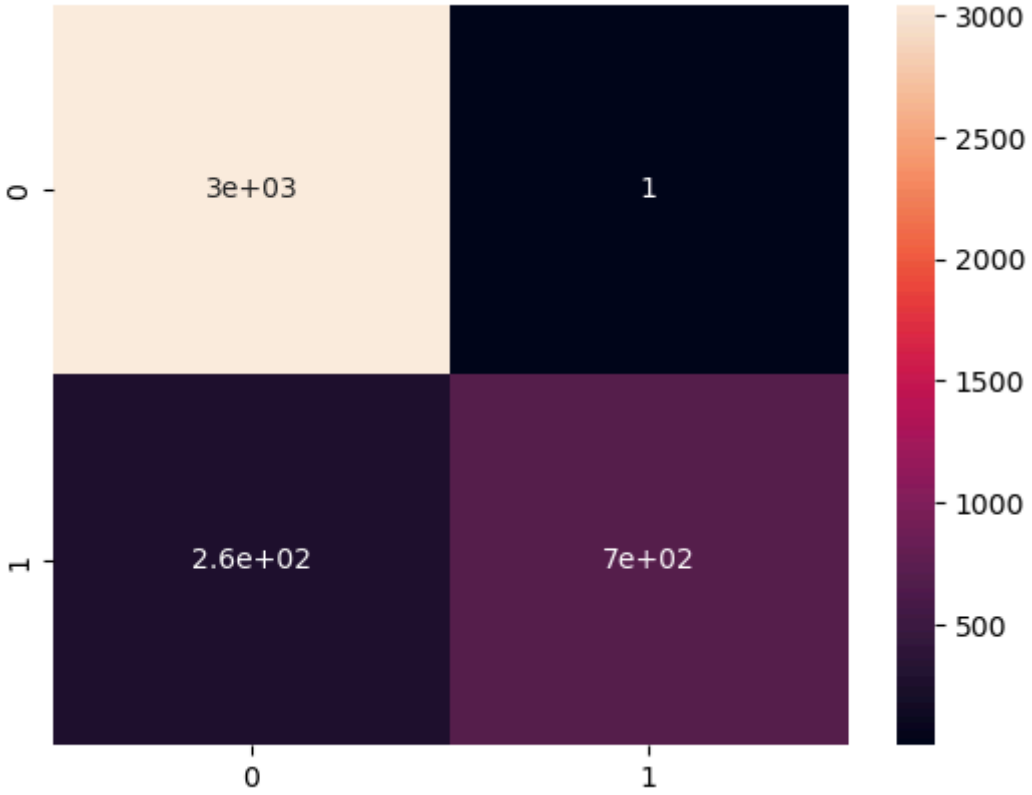
```
0.9028504944735312
0.7402799377916018
```

In [65]:
```
#Confusion matrix for train data
```

In [66]:
```
import seaborn as sn
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_train, y_pred)
sn.heatmap(cm, annot=True)
```

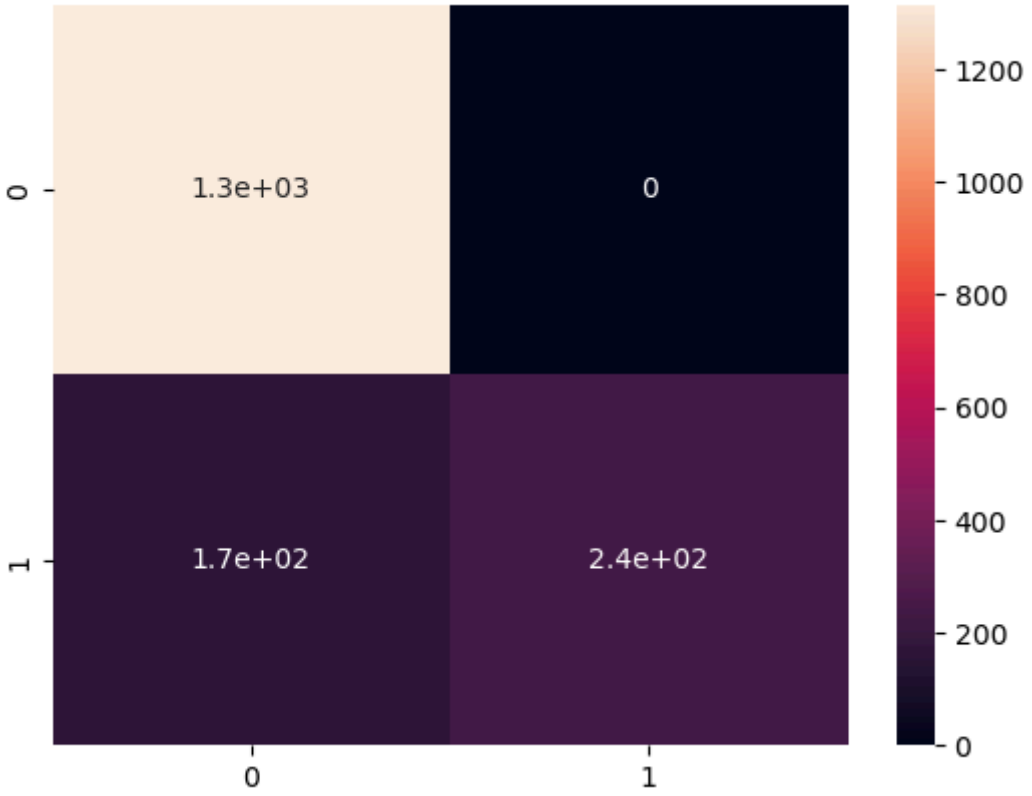
Out[66]:
```
<AxesSubplot:>
```



In [67]:
```
#Confusion matrix for test data
```

In [68]:
```
cm1 = confusion_matrix(y_test, y_pred_test)
sn.heatmap(cm1, annot=True)
```

Out[68]:
```
<AxesSubplot:>
```



In [69]:
```
#Hyper parameter tuning to find best
```

In [70]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB
params_KNN = {'alpha': [0.00000001,0.0000001, 0.000001,0.00001,0.0001,0.001,0.01,0.
gs_KNN = GridSearchCV(estimator=MultinomialNB(),
                      param_grid=params_KNN,
                      verbose=1,  # verbose: the higher, the more messages
                      scoring='f1',
                      return_train_score=True)
```

In [71]:
```
gs_KNN.fit(X_train, y_train)
best_parameters = gs_KNN.best_params_
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
```

In [72]:
```
best_parameters
```

Out[72]:
```
{'alpha': 0.01}
```

In [73]:
```
clf = MultinomialNB(alpha= 0.01)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_train)
```

In [74]:
```
from sklearn.metrics import accuracy_score
trainacc = accuracy_score(y_train, y_pred)
trainf1 = f1_score(y_train, y_pred)
print(trainacc)
print(trainf1)
```

```
0.9997505612372163
0.9994805194805195
```

In [75]:
```
y_pred_test = clf.predict(X_test)
testacc = accuracy_score(y_test, y_pred_test)
testf1 = f1_score(y_test, y_pred_test)
```

```
print(testacc)
print(testf1)
```

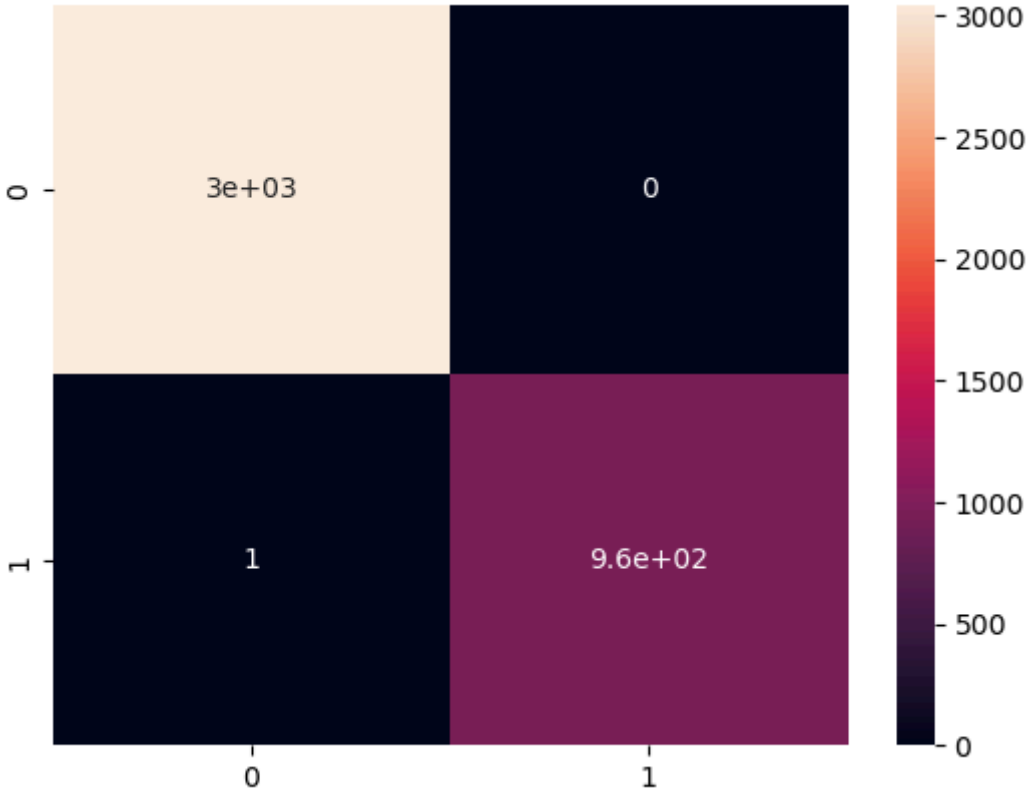
```
0.9860383944153578
0.9696202531645569
```

In [76]:
```
#Confusion matrix for test data
```

In [77]:
```
import seaborn as sn
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_train, y_pred)
sn.heatmap(cm, annot=True)
```

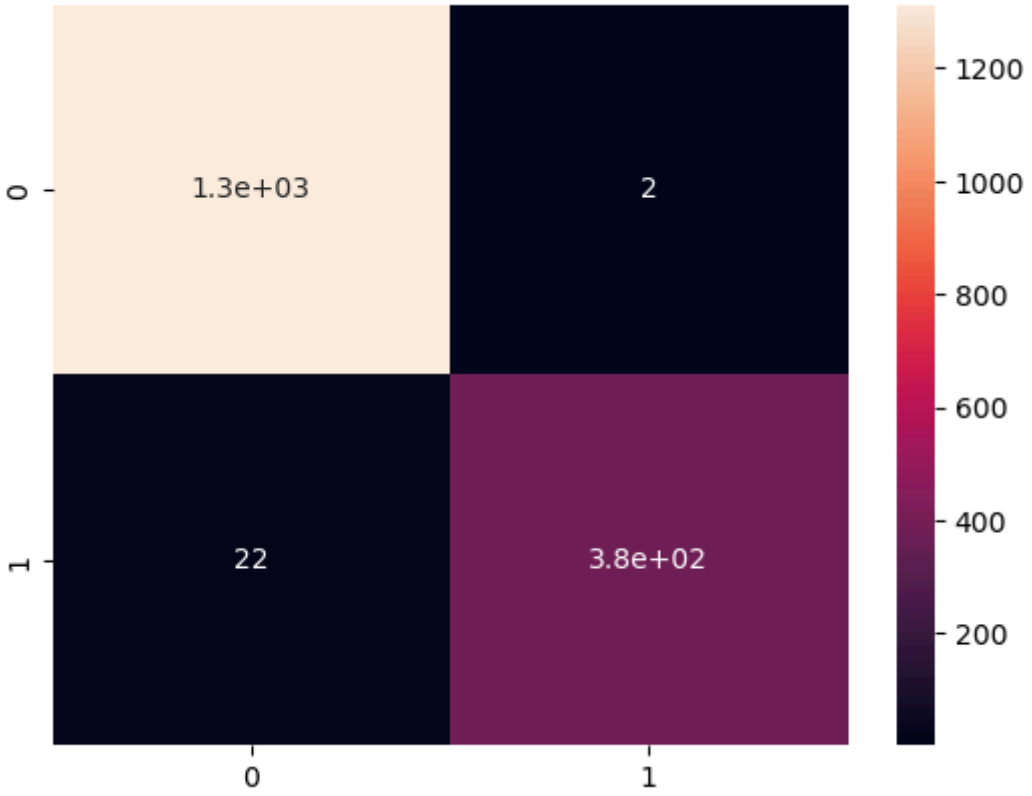
Out[77]:
```
<AxesSubplot:>
```



In [78]:
```
#Confusion matrix for test data
```

In [79]:
```
cm1 = confusion_matrix(y_test, y_pred_test)
sn.heatmap(cm1, annot=True)
```

Out[79]:
```
<AxesSubplot:>
```



In [ ]: