In [1]:
```
#importing the required libraries
import numpy as np
import pandas as pd
```

In [2]:
```
#import data set
dataset = pd.read_csv("smsspam",sep='\t',names=['label','message'])
#sep because the dataset data ,is not seperated by comma whereas its seperated by tabs
dataset
```

Out[2]:

	label	message
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...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

In [3]:
```
dataset.isnull()
```

Out[3]:

	label	message
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
...	...	...
5567	False	False
5568	False	False
5569	False	False
5570	False	False
5571	False	False

5572 rows × 2 columns

In [4]:
```
dataset.info()
```

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

In [5]:
```
dataset.describe()
```

Out[5]:

	label	message
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

In [6]:
```
dataset = dataset.fillna(method='bfill')
dataset
```

Out[6]:

	label	message
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...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

In [7]:
```
dataset["label"] = dataset["label"].map({"ham":0,"spam":1})
```

In [8]:
```
dataset
```

Out[8]:

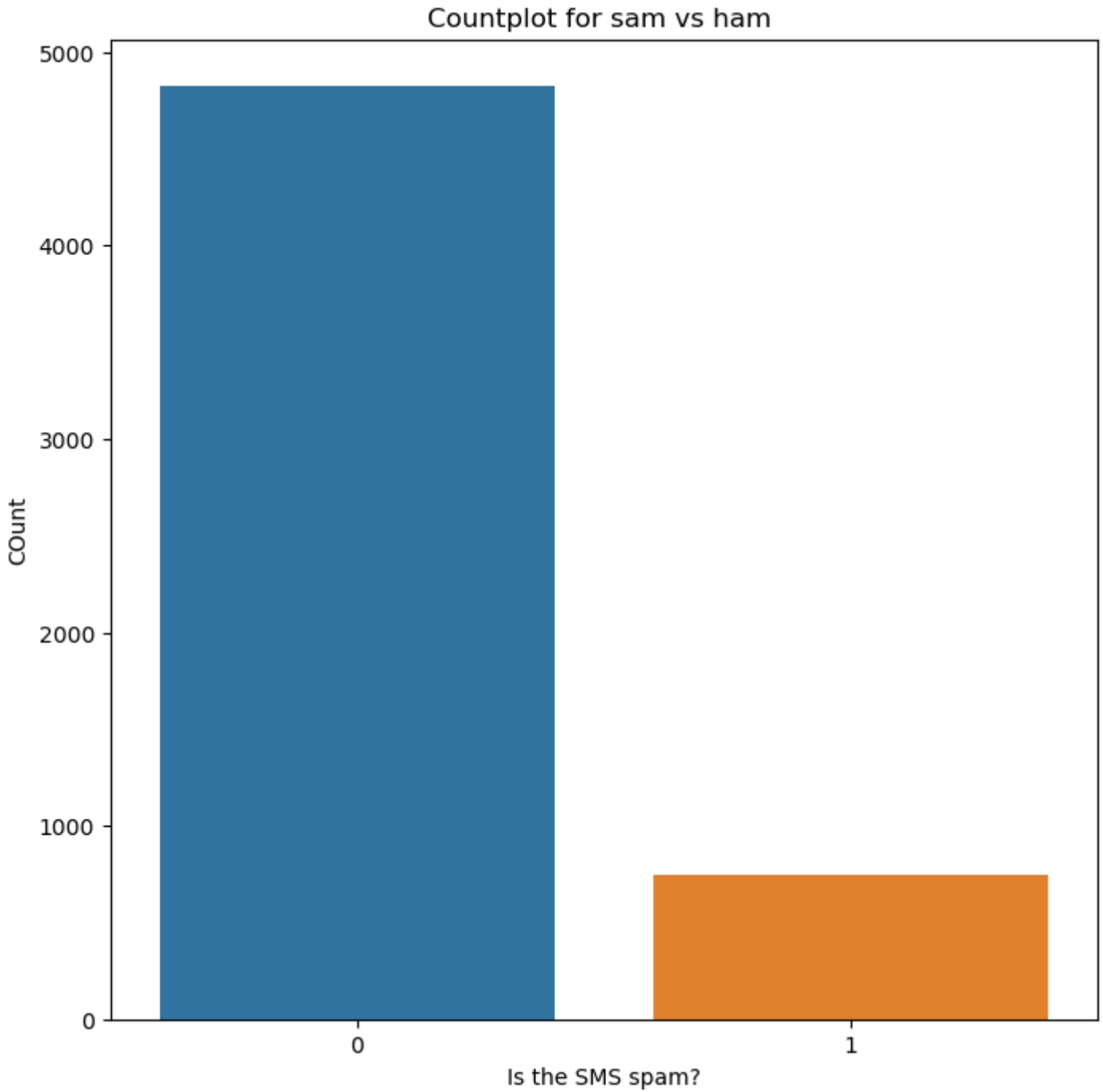
	label	message
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	1	This is the 2nd time we have tried 2 contact u...
5568	0	Will ü b going to esplanade fr home?
5569	0	Pity, * was in mood for that. So...any other s...
5570	0	The guy did some bitching but I acted like i'd...
5571	0	Rofl. Its true to its name

5572 rows × 2 columns

In [9]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [10]:
```
plt.figure(figsize=(8,8))
g = sns.countplot(x="label",data=dataset)
plt.title("Countplot for sam vs ham ")
plt.xlabel('Is the SMS spam?')
plt.ylabel('COunt')
```

Out[10]:
```
Text(0, 0.5, 'COunt')
```


Countplot for sam vs ham
5000
4000
3000
2000
1000
0
COunt
0
1
Is the SMS spam?

In [11]:
```
#Handling the imbalamced data set usng Oversampling
only_spam= dataset[dataset['label']==1]
only_spam
```

Out[11]:

	label	message
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
5	1	FreeMsg Hey there darling it's been 3 week's n...
8	1	WINNER!! As a valued network customer you have...
9	1	Had your mobile 11 months or more? U R entitle...
11	1	SIX chances to win CASH! From 100 to 20,000 po...
...	...	...
5537	1	Want explicit SEX in 30 secs? Ring 02073162414...
5540	1	ASKED 3MOBILE IF 0870 CHATLINES INCLU IN FREE ...
5547	1	Had your contract mobile 11 Mnths? Latest Moto...
5566	1	REMINDER FROM O2: To get 2.50 pounds free call...
5567	1	This is the 2nd time we have tried 2 contact u...

747 rows × 2 columns

In [12]:
```
#no.of spam sms
print(len(only_spam))
#no.of ham sms
print(len(dataset)-len(only_spam))
```

```
747
4825
```

In [13]:
```
count = int((dataset.shape[0]-only_spam.shape[0])/only_spam.shape[0])
count
```

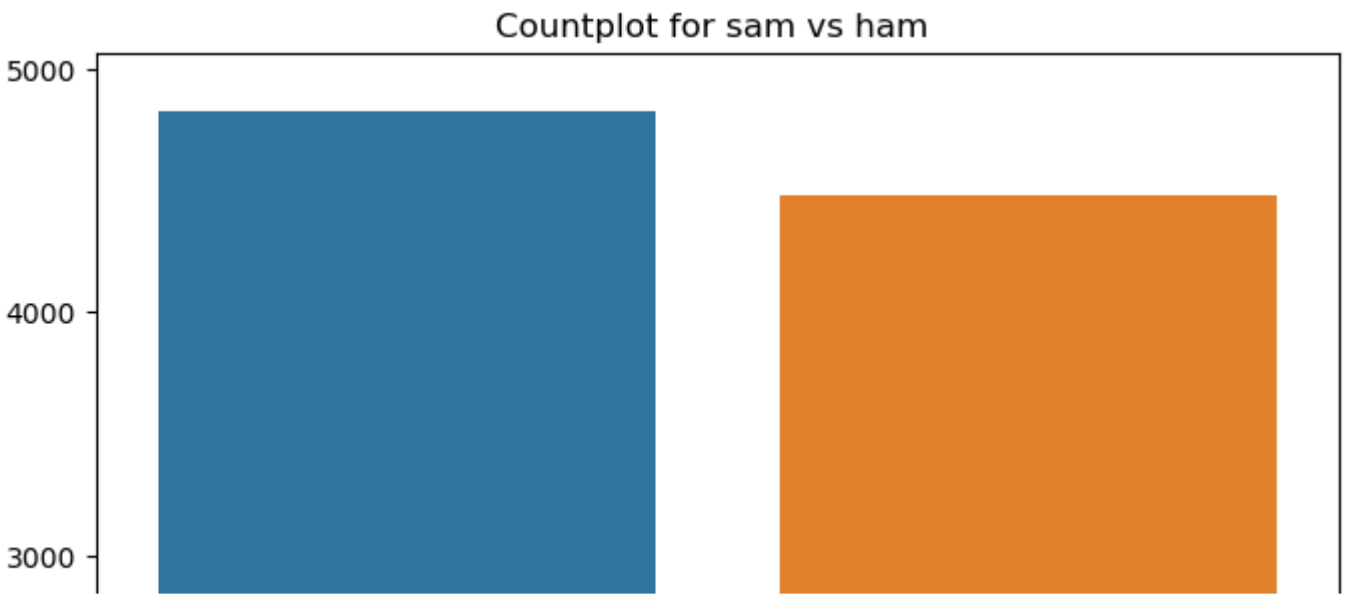
Out[13]: 6

In [14]:
```
#to balance the dataset
for i in range (0,count-1):
    dataset=pd.concat([dataset,only_spam])

dataset.shape
```

Out[14]: (9307, 2)

In [15]:
```
#balanced dataset
plt.figure(figsize=(8,8))
g = sns.countplot(x="label",data=dataset)
plt.title("Countplot for sam vs ham ")
plt.xlabel('Is the SMS spam?')
plt.ylabel('COunt')
```

Out[15]: Text(0, 0.5, 'COunt')



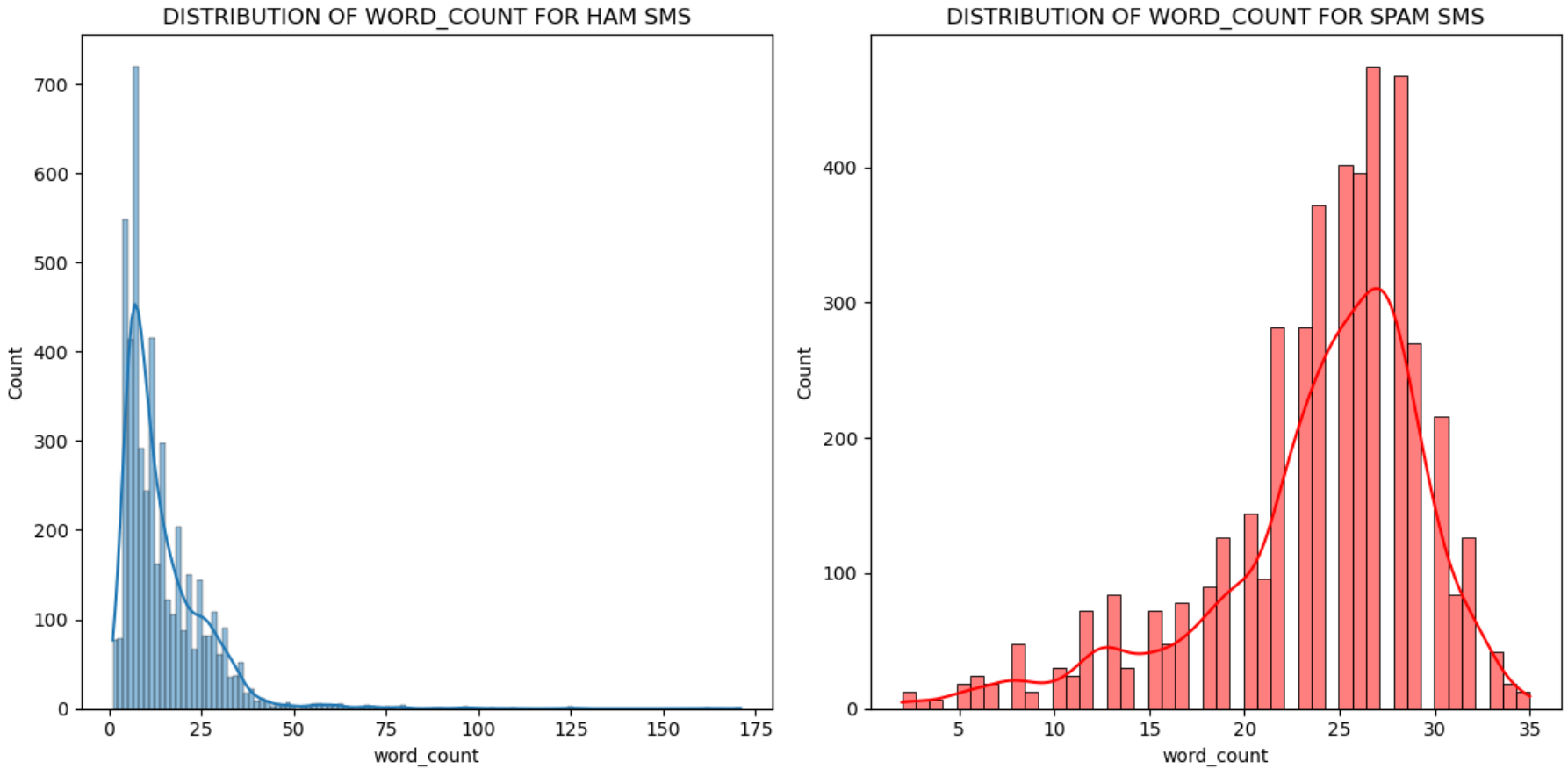
In [16]:
```
dataset['word_count'] = dataset['message'].apply(lambda x:len(x.split()))
dataset
```

Out[16]:

	label	message	word_count
0	0	Go until jurong point, crazy.. Available only ...	20
1	0	Ok lar... Joking wif u oni...	6
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	28
3	0	U dun say so early hor... U c already then say...	11
4	0	Nah I don't think he goes to usf, he lives aro...	13
...	...	...	...
5537	1	Want explicit SEX in 30 secs? Ring 02073162414...	16
5540	1	ASKED 3MOBILE IF 0870 CHATLINES INCLU IN FREE ...	33
5547	1	Had your contract mobile 11 Mnths? Latest Moto...	28
5566	1	REMINDER FROM O2: To get 2.50 pounds free call...	28
5567	1	This is the 2nd time we have tried 2 contact u...	30

9307 rows × 3 columns

In [17]:
```
plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
g=sns.histplot(dataset[dataset['label']==0].word_count,kde=True)
p=plt.title("DISTRIBUTION OF WORD_COUNT FOR HAM SMS")

plt.subplot(1,2,2)
g=sns.histplot(dataset[dataset['label']==1].word_count,color="red",kde=True)
p=plt.title("DISTRIBUTION OF WORD_COUNT FOR SPAM SMS")

plt.tight_layout()
plt.show()
```



In [18]:
```
#creating a new feature to check whether it has curreny symbols
def currency(dataset):
    currency_symbols = ['$','£','€','¥','₹']
    for i in currency_symbols:
        if i in dataset:
            return 1
    return 0
```

In [19]:
```
dataset["Contains_currency_symbols"]=dataset["message"].apply(currency)
dataset
```

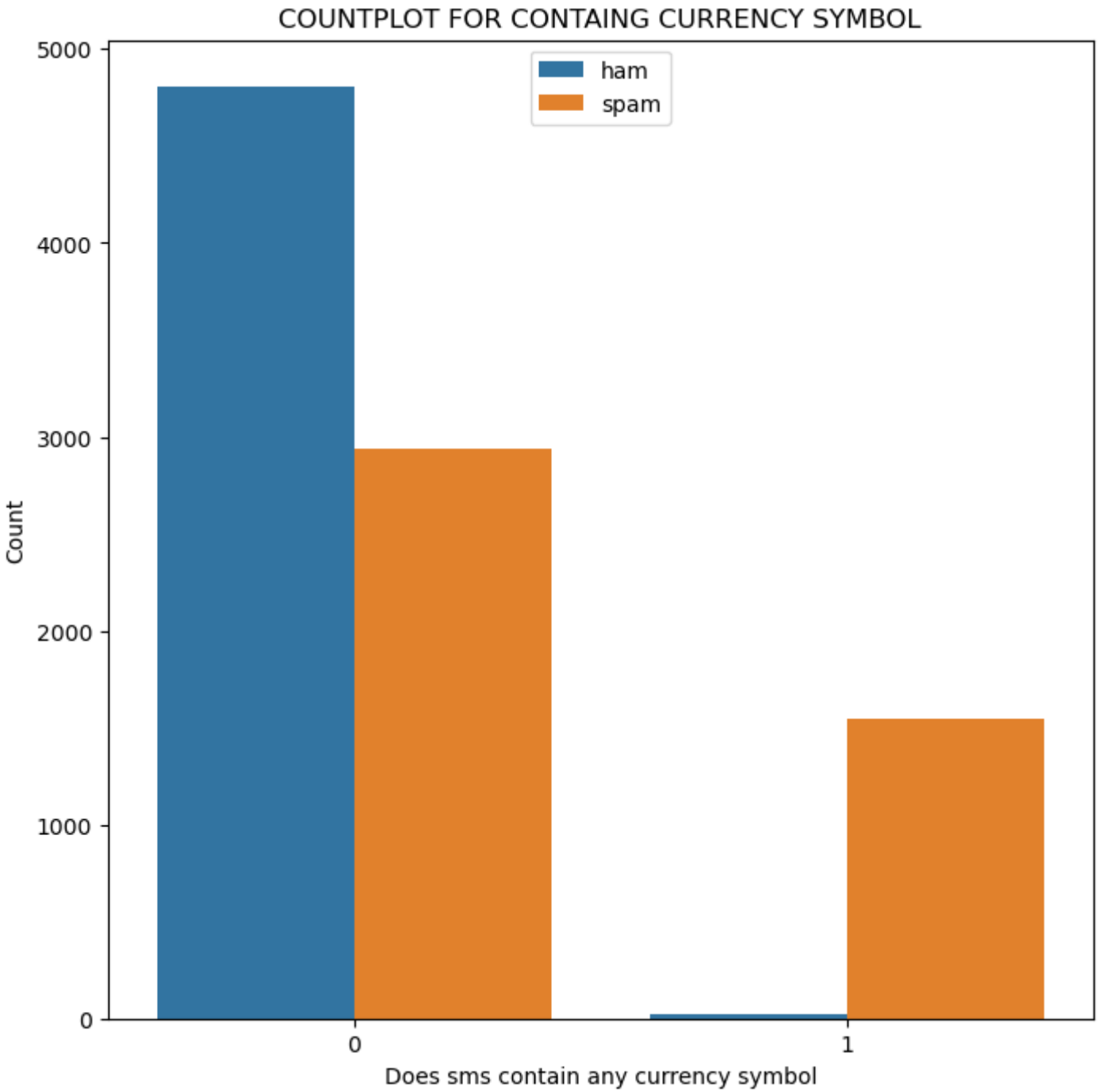
Out[19]:

	label	message	word_count	Contains_currency_symbols
0	0	Go until jurong point, crazy.. Available only ...	20	0
1	0	Ok lar... Joking wif u oni...	6	0
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	28	0
3	0	U dun say so early hor... U c already then say...	11	0
4	0	Nah I don't think he goes to usf, he lives aro...	13	0
...	...	...	...	...
5537	1	Want explicit SEX in 30 secs? Ring 02073162414...	16	0
5540	1	ASKED 3MOBILE IF 0870 CHATLINES INCLU IN FREE ...	33	1
5547	1	Had your contract mobile 11 Mnths? Latest Moto...	28	0
5566	1	REMINDER FROM O2: To get 2.50 pounds free call...	28	0
5567	1	This is the 2nd time we have tried 2 contact u...	30	1

9307 rows × 4 columns

In [20]:
```
plt.figure(figsize=(8,8))
g=sns.countplot(x="Contains_currency_symbols",data=dataset,hue="label")
plt.title("COUNTPLOT FOR CONTAING CURRENCY SYMBOL")
plt.xlabel("Does sms contain any currency symbol")
plt.ylabel("Count")
plt.legend(labels=["ham","spam"],loc=9)
```

Out[20]:
```
<matplotlib.legend.Legend at 0x1b0af8af3a0>
```


COUNTPLOT FOR CONTAING CURRENCY SYMBOL
ham
spam
5000
4000
3000
2000
1000
0
Count
0
1
Does sms contain any currency symbol

In [21]:
```
#creating new feature of containing numbers
def number (dataset):
    for i in dataset:
        if ord(i)>=48 and ord(i)<=57:
            return 1
    return 0
```

In [22]:
```
dataset["contains_number"]=dataset["message"].apply(number)
```

In [23]:
```
dataset
```

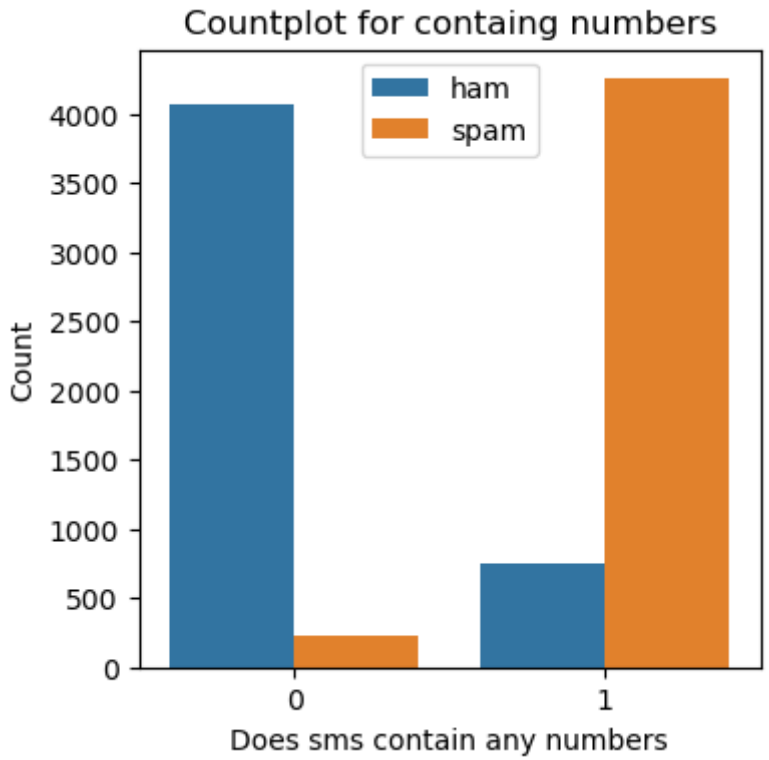
Out[23]:

	label	message	word_count	Contains_currency_symbols	contains_number
0	0	Go until jurong point, crazy.. Available only ...	20	0	0
1	0	Ok lar... Joking wif u oni...	6	0	0
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	28	0	1
3	0	U dun say so early hor... U c already then say...	11	0	0
4	0	Nah I don't think he goes to usf, he lives aro...	13	0	0
...	...	...	...	...	...
5537	1	Want explicit SEX in 30 secs? Ring 02073162414...	16	0	1
5540	1	ASKED 3MOBILE IF 0870 CHATLINES INCLU IN FREE ...	33	1	1
5547	1	Had your contract mobile 11 Mnths? Latest Moto...	28	0	1
5566	1	REMINDER FROM O2: To get 2.50 pounds free call...	28	0	1
5567	1	This is the 2nd time we have tried 2 contact u...	30	1	1

9307 rows × 5 columns

In [24]:
```
plt.figure(figsize=(4,4))
g=sns.countplot(x="contains_number",data=dataset,hue="label")
plt.title("Countplot for containg numbers")
plt.xlabel("Does sms contain any numbers")
plt.ylabel("Count")
plt.legend(labels=["ham","spam"],loc=9)
```

Out[24]:
```
<matplotlib.legend.Legend at 0x1b0afbbbaf0>
```



In [25]:
```
#Data cleaning
import nltk
import re
nltk.download("stopwords")
nltk.download("wordnet")
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

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

In [26]:
```
corpus = []
wnl = WordNetLemmatizer()

for sms in list(dataset.message):
    message = re.sub(pattern='[^a-zA-Z]',repl= ' ',string = sms)
    message = message.lower()
    words = message.split()#tokenization
    filtered_words = [word for word in message if word not in set(stopwords.words('english'))]
    lem_words = [wnl.lemmatize(word) for word in filtered_words]
    message = ''.join(lem_words)


    corpus.append(message)
```

In [27]:
```
corpus
```

Out[27]:
```
['g unl jurng pn  crz   vlble nl n bug n gre wrl l e buffe    cne here g re w   ',
 'k lr    jkng wf u n   ',
 'free enr n    wkl cp  wn f cup fnl k          ex f         receve enr quen  x re  c  ppl           ver   ',
 'u un   erl hr    u c lre hen    ',
 'nh  n  hnk he ge  uf  he lve run here hugh',
 'freeg he here rlng   been   week  nw n n wr bck    lke e fun u up fr  ll  b k  xxx  chg  en        rcv',
 'even  brher  n lke  pek wh e  he re e lke  pen ',
 ' per ur reque  elle elle  ru nnnungne nurungu ve   h been e  ur cllerune fr ll cller  pre     cp ur fren cllerun
e',
 'wnner     vlue newrk cuer u hve been elece  receve      prze rewr   cl cll             cl ce kl     vl    hur n
l ',
 'h ur ble    nh r re  u r enle  upe  he le clur ble wh cer fr free  cll he ble upe c free n           ',
 '  gnn be he n n  n  wn  lk bu h uff nre ngh  k   ve cre enugh  ',
 'x chnce  wn ch  fr            pun x  ch   n en         c    p          nc ppl repl hl   nf',
 'urgen  u hve wn    week free eberhp n ur          prze jckp  x he wr  cl  n         c www buk ne lccl pbx     ln
w  rw  ',
 ' ve been erchng fr he rgh wr  hnk u fr h breher   pre  wn ke ur help fr grne n wll fulfl  pre  u hve been wnerfu
l n  bleng  ll e ',
 ' hve  e n un wh wll  ',
```

In [28]:
```
#building the bag of words model
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(max_features = 500)
vectors = tfidf.fit_transform(corpus).toarray()
feature_names = tfidf.get_feature_names_out()
```

In [29]:
```
x = pd.DataFrame(vectors,columns = feature_names)
y = dataset['label']
```

In [30]:
```
from sklearn.model_selection import cross_val_score,train_test_split
from sklearn.metrics import classification_report,confusion_matrix
```

In [31]:
```
X_train,X_test,Y_train,Y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

In [32]:
```
X_train
```

Out[32]:

	bb	bbe	bc	bck	be	becue	been	beer	befre	beleve	...	wx	xch	xh	xn	xng	xucn	xx	xxx	xze	ze
3533	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	0.0	0.0
2592	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	0.0	0.0
4253	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	0.0	0.0
6976	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	0.0	0.0
7191	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	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5734	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	0.0	0.0
5191	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	0.0	0.0
5390	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	0.0	0.0
860	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	0.0	0.0
7270	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	0.0	0.0

7445 rows × 500 columns

In [33]:
```
X_test
```

Out[33]:

	bb	bbe	bc	bck	be	becue	been	beer	befre	beleve	...	wx	xch	xh	xn	xng	xucn	xx	xxx	xze	ze
1155	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1790	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3003	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6489	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
592	0.0	0.0	0.0	0.0	0.0	0.0	0.23912	0.0	0.313684	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4147	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
274	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1345	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8891	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4031	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

1862 rows × 500 columns

In [34]:
```
Y_train
```

Out[34]:
```
3533    0
2592    0
4253    0
4896    1
833     1
       ..
1072    1
5191    0
5390    0
860     0
1423    1
Name: label, Length: 7445, dtype: int64
```

In [35]:
```
Y_test
```

Out[35]:
```
1155    0
1790    0
3003    0
1122    1
592     1
       ..
4147    0
274     0
1345    0
2367    1
4031    0
Name: label, Length: 1862, dtype: int64
```

In [36]:
```
#building naive bayes model
from sklearn.naive_bayes import MultinomialNB
mnb= MultinomialNB()
cv = cross_val_score(mnb,x,y,scoring='f1',cv=10)
print(round(cv.mean(),3))
print(round(cv.std(),3))
```

```
0.923
0.008
```

In [37]:
```
mnb.fit(X_train,Y_train)
Y_pred = mnb.predict(X_test)
```

In [38]:
```
print(classification_report(Y_test,Y_pred))
```

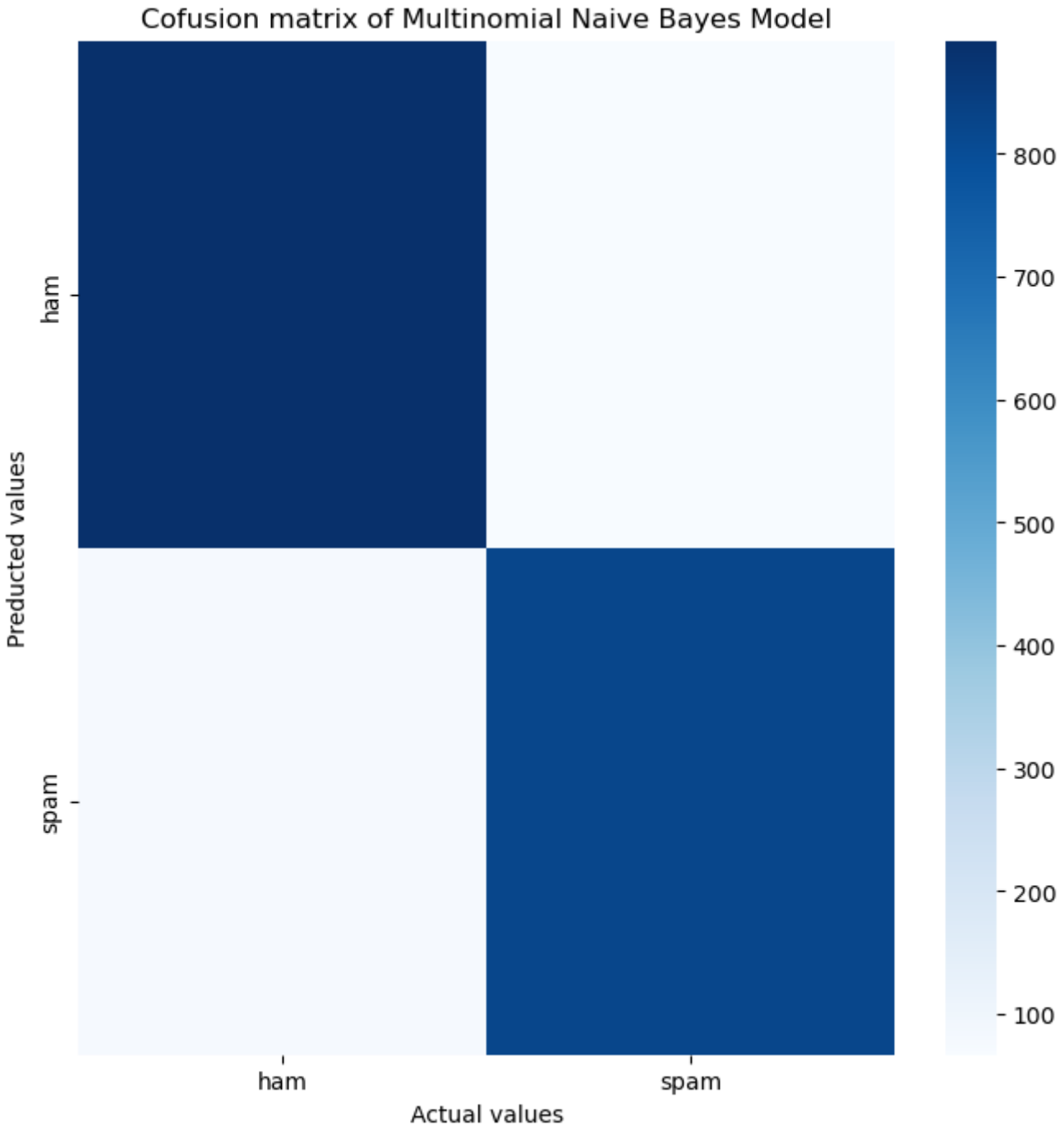
```
              precision    recall  f1-score   support

           0       0.92      0.93      0.92       959
           1       0.92      0.91      0.92       903

    accuracy                           0.92      1862
   macro avg       0.92      0.92      0.92      1862
weighted avg       0.92      0.92      0.92      1862
```

In [39]:
```
cm = confusion_matrix(Y_test,Y_pred)
cm
```

Out[39]:
```
array([[892,  67],
       [ 79, 824]], dtype=int64)
```

In [40]:
```
plt.figure(figsize=(8,8))
axis_labels= ['ham','spam']
g= sns.heatmap(data=cm,xticklabels=axis_labels,yticklabels=axis_labels,cmap="Blues")
p=plt.title("Cofusion matrix of Multinomial Naive Bayes Model")
plt.xlabel("Actual values")
plt.ylabel("Predicted values")
```

Out[40]:
```
Text(70.72222222222221, 0.5, 'Predicted values')
```


Cofusion matrix of Multinomial Naive Bayes Model
Predicted values
ham
spam
Actual values
ham
spam
800
700
600
500
400
300
200
100

In [41]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
cv1=cross_val_score(dt,x,y,scoring='f1',cv=10)
print(round(cv1.mean(),3))
print(round(cv1.std(),3))
```

```
0.977
0.005
```

In [42]:
```
dt.fit(X_train,Y_train)
y_pred1=dt.predict(X_test)
```

In [43]:
```
print(classification_report(Y_test,y_pred1))
```

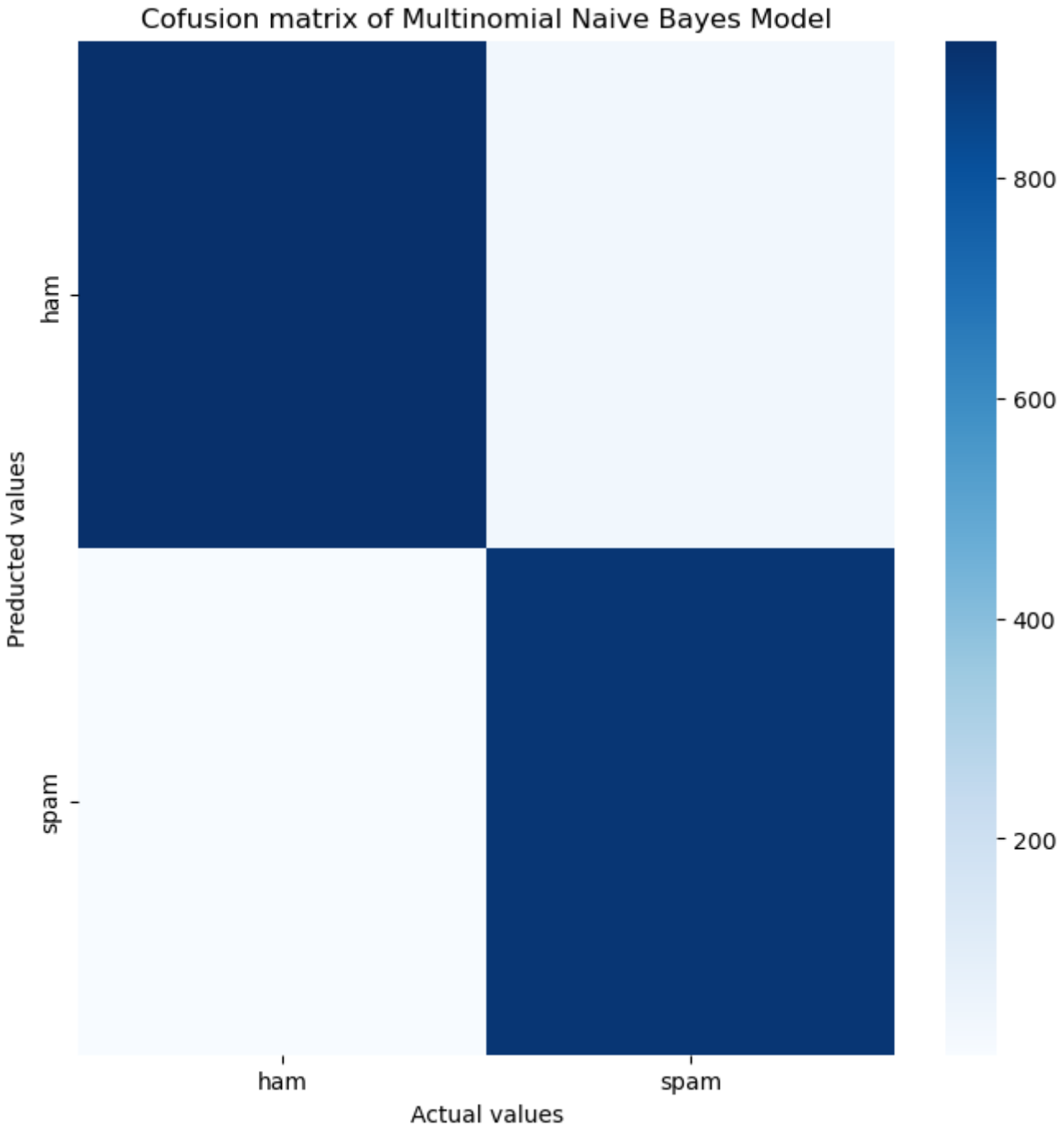
```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98       959
           1       0.96      1.00      0.98       903

    accuracy                           0.98      1862
   macro avg       0.98      0.98      0.98      1862
weighted avg       0.98      0.98      0.98      1862
```

In [44]:
```
cm = confusion_matrix(Y_test,y_pred1)
cm
```

Out[44]:
```
array([[924,  35],
       [  4, 899]], dtype=int64)
```

In [45]:
```
plt.figure(figsize=(8,8))
axis_labels= ['ham','spam']
g= sns.heatmap(data=cm,xticklabels=axis_labels,yticklabels=axis_labels,cmap="Blues")
p=plt.title("Cofusion matrix of Multinomial Naive Bayes Model")
plt.xlabel("Actual values")
plt.ylabel("Predicted values")
```

Out[45]:
```
Text(70.72222222222221, 0.5, 'Predicted values')
```


Cofusion matrix of Multinomial Naive Bayes Model
Predicted values
ham
spam
Actual values
ham
spam
800
600
400
200

In [46]:
```
def predict_spam(sms):
    message = re.sub(pattern='[^a-zA-Z]',repl= ' ',string = sms)
    message = message.lower()
    words = message.split()#tokenization
    filtered_words = [word for word in message if word not in set(stopwords.words('english'))]
    lem_words = [wnl.lemmatize(word) for word in filtered_words]
    message = ''.join(lem_words)
    temp = tfidf.transform([message]).toarray()
    return mnb.predict(temp)
```

In [47]:
```
#prediction
sample_message='IMPORTANT - you have a chance to n lottery trip to paris for 4D3N'
if predict_spam(sample_message):
    print("THIS IS SPAM")
else :
    print("THIS IS HAM")
```

```
THIS IS SPAM

C:\Users\Saketh Nandan\Downloads\Anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid
feature names, but MultinomialNB was fitted with feature names
  warnings.warn(
```

In [48]:
```
#prediction2
sample2 = "I had never got any spam messages in my whole life"
if predict_spam(sample2):
    print("THIS IS SPAM")
else:
    print("THIS IS HAM")
```

```
THIS IS HAM

C:\Users\Saketh Nandan\Downloads\Anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid
feature names, but MultinomialNB was fitted with feature names
  warnings.warn(
```