In [1]:
```
import nltk
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer,WordNetLemmatizer
from nltk.tokenize import word_tokenize
import sklearn.metrics as m
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

In [2]:
```
dataset = pd.read_csv("SMSSpamCollection", sep='\t', names=['label', 'message'])
```

In [3]:
```
dataset
```

Out[3]:

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

```
In [7]: dataset
```

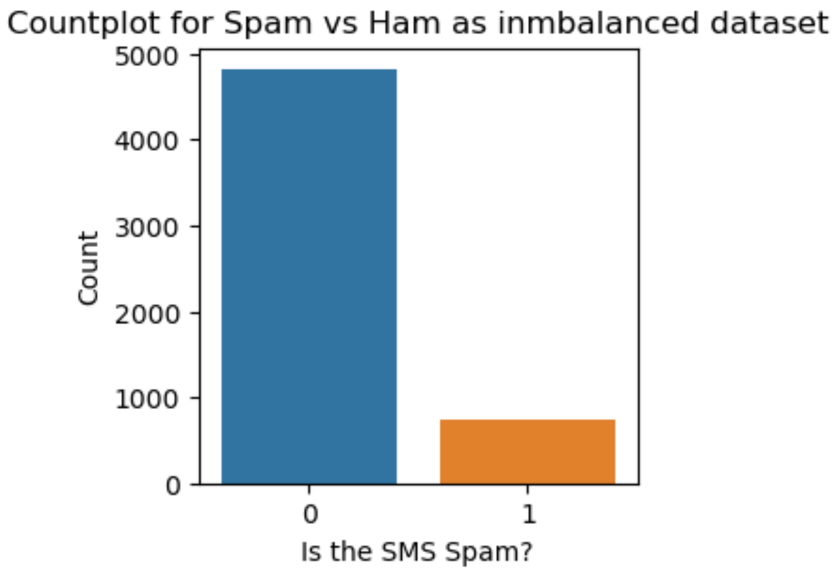
Out[7]:

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

```
In [9]: plt.figure(figsize=(3, 3))
        g = sns.countplot(x="label", data=dataset)
        p = plt.title('Countplot for Spam vs Ham as inmbalanced dataset')
        p = plt.xlabel('Is the SMS Spam?')
        p = plt.ylabel('Count')
```



```
In [10]: only_spam = dataset[dataset["label"] == 1]
In [11]: 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]: print("Number of Spam SMS:", len(only_spam))
         print("Number of Ham SMS:", len(dataset) - len(only_spam))
```

```
Number of Spam SMS: 747
Number of Ham SMS: 4825
```

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

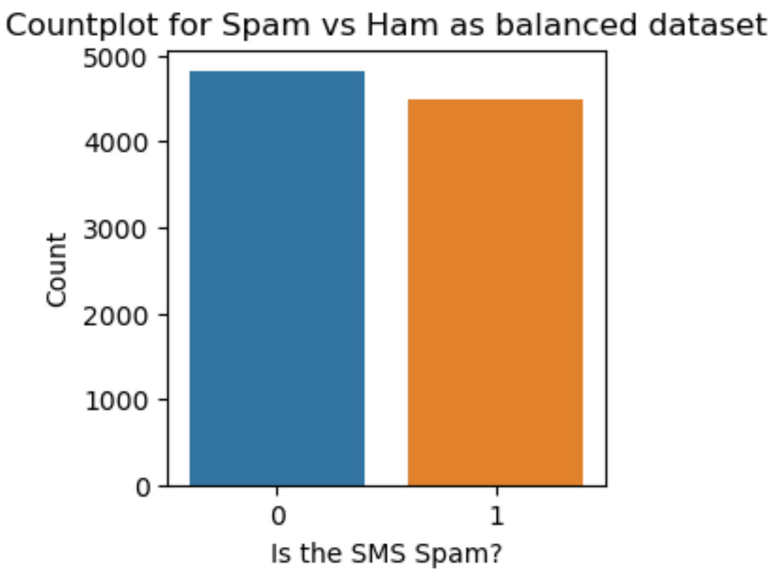
```
In [14]: count
```

```
Out[14]: 6
```

```
In [15]: for i in range(0, count-1):
             dataset = pd.concat([dataset, only_spam])

         dataset.shape
```

```
Out[15]: (9307, 2)
```

```
In [16]: plt.figure(figsize=(3, 3))
         g = sns.countplot(x="label", data=dataset)
         p = plt.title('Countplot for Spam vs Ham as balanced dataset')
         p = plt.xlabel('Is the SMS Spam?')
         p = plt.ylabel('Count')
```



```
In [17]: dataset['word_count'] = dataset['message'].apply(lambda x: len(x.split()))
```

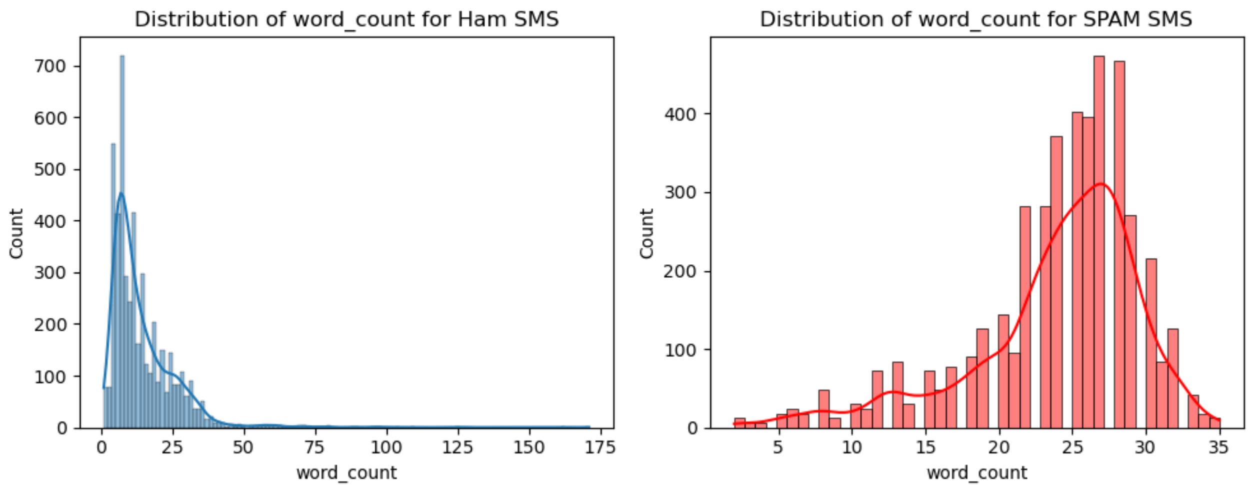
```
In [18]: dataset
```

Out[18]:

	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 [19]: plt.figure(figsize=(10,4))

         # (1,1)
         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')

         # (1,2)
         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 [20]: # Creating new feature of containing currency symbol
         def currency(data):
             currency_symbols = ['€', '$', '£', '¥', '₹']
             for i in currency_symbols:
                 if i in data:
                     return 1
             return 0
```

```
In [21]: dataset["contains_currency_symbols"] = dataset["message"].apply(currency)
```

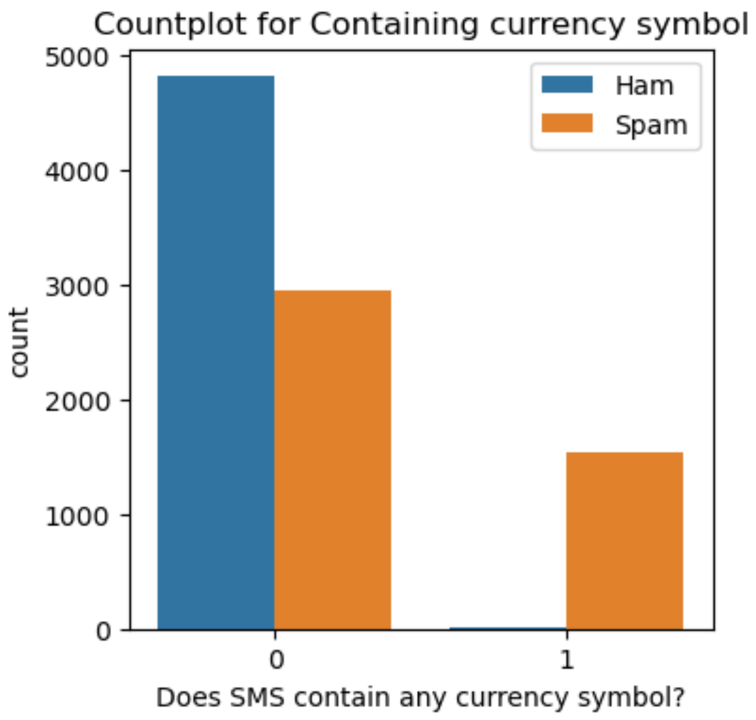
In [22]:
```
dataset
```

Out[22]:

	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

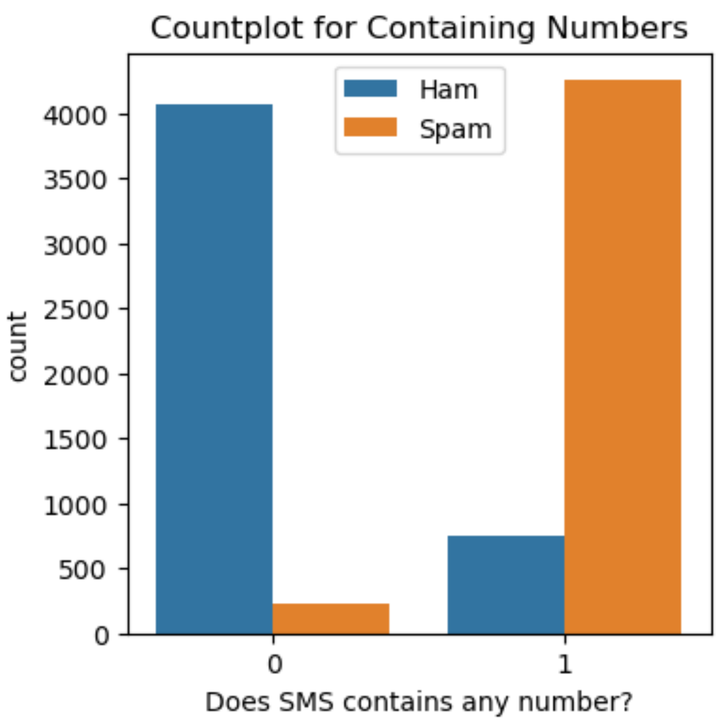
In [23]:
```
# Countplot for contains_currency_symbols
plt.figure(figsize=(4,4))
g = sns.countplot(x = 'contains_currency_symbols', data = dataset, hue = "label")
p = plt.title('Countplot for Containing currency symbol')
p = plt.xlabel('Does SMS contain any currency symbol?')
p = plt.ylabel('count')
p = plt.legend(labels=["Ham", "Spam"], loc=0)
```



In [24]:
```
# creating new feature of containing numbers
def number(data):
    for i in data:
        if ord(i) >= 48 and ord(i) <= 57:
            return 1
    return 0
```

In [25]:
```
dataset["contains_number"] = dataset["message"].apply(number)
```

In [26]:
```
# Countplot for containing numbers
plt.figure(figsize=(4,4))
g = sns.countplot(x = 'contains_number', data = dataset, hue = "label")
p = plt.title('Countplot for Containing Numbers')
p = plt.xlabel('Does SMS contains any number?')
p = plt.ylabel('count')
p = plt.legend(labels=["Ham", "Spam"], loc=0)
```



In [27]:
```
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
[nltk_data]     C:\Users\ahmed\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ahmed\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [28]:
```
corpus = []
wnl = WordNetLemmatizer()

for sms in list(dataset.message):
    message = re.sub(pattern='[^A-Za-z]', repl= ' ', string=sms) # Filtering out special
    message = message.lower()
    words = message.split() # Tokenizer
    filtered_words = [word for word in words if word not in set(stopwords.words("english
    lenm_words = [wnl.lemmatize(word) for word in filtered_words]
    message = ' '.join(lenm_words)

    corpus.append(message)
```

In [29]:
```python
corpus
```

Out[29]:
```
['go jurong point crazy available bugis n great world la e buffet cine got amore wat',
 'ok lar joking wif u oni',
 'free entry wkly comp win fa cup final tkts st may text fa receive entry question std
txt rate c apply',
 'u dun say early hor u c already say',
 'nah think go usf life around though',
 'freemsg hey darling week word back like fun still tb ok xxx std chgs send rcv',
 'even brother like speak treat like aid patent',
 'per request melle melle oru minnaminunginte nurungu vettam set callertune caller pre
ss copy friend callertune',
 'winner valued network customer selected receivea prize reward claim call claim code
kl valid hour',
 'mobile month u r entitled update latest colour mobile camera free call mobile update
co free',
 'gonna home soon want talk stuff anymore tonight k cried enough today',
 'six chance win cash pound txt csh send cost p day day tsandcs apply reply hl info',
 'urgent week free membership prize jackpot txt word claim c www dbuk net lccltd pobox
ldnw rw',
 'searching right word thank breather promise wont take help granted fulfil promise wo
```

In [30]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf=TfidfVectorizer(max_features=5000)
features=tfidf.fit_transform(corpus)
features=features.toarray()
features
```

Out[30]:
```
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 [31]:
```python
len(tfidf.get_feature_names_out())
```

Out[31]:
```
5000
```

In [32]:
```python
feature_names = tfidf.get_feature_names_out()
```

In [33]:
```python
x = pd.DataFrame(features, columns = feature_names)
y = dataset['label']
```

In [34]:
```python
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.metrics import classification_report, confusion_matrix
```

In [35]:
```python
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42
```

In [36]:
```
x_train
```

Out[36]:

	aa	aah	aathi	ab	aberdeen	abi	ability	abiola	abj	able	...	zebra	zed	zero	zf	zhong	zindgi
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
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
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
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
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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
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
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
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
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

7445 rows × 5000 columns

In [37]:
```
# Naive Bayes Model
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB()
cv = cross_val_score(mnb, x, y, scoring='f1', cv=10)
print(mnb)
print(cv)
```

```
MultinomialNB()
[0.97674419 0.97900552 0.9810901  0.98447894 0.98004435 0.98218263
 0.97900552 0.98113208 0.98342541 0.9844098 ]
```

In [38]:
```
print(cv.std)
```

```
<built-in method std of numpy.ndarray object at 0x000002D4FAD01C30>
```

In [39]:
```
print(round(cv.mean(),3))
print(round(cv.std(),3))
```

```
0.981
0.002
```

In [40]:
```
mnb.fit(x_train, y_train)
y_pred = mnb.predict(x_test)
```

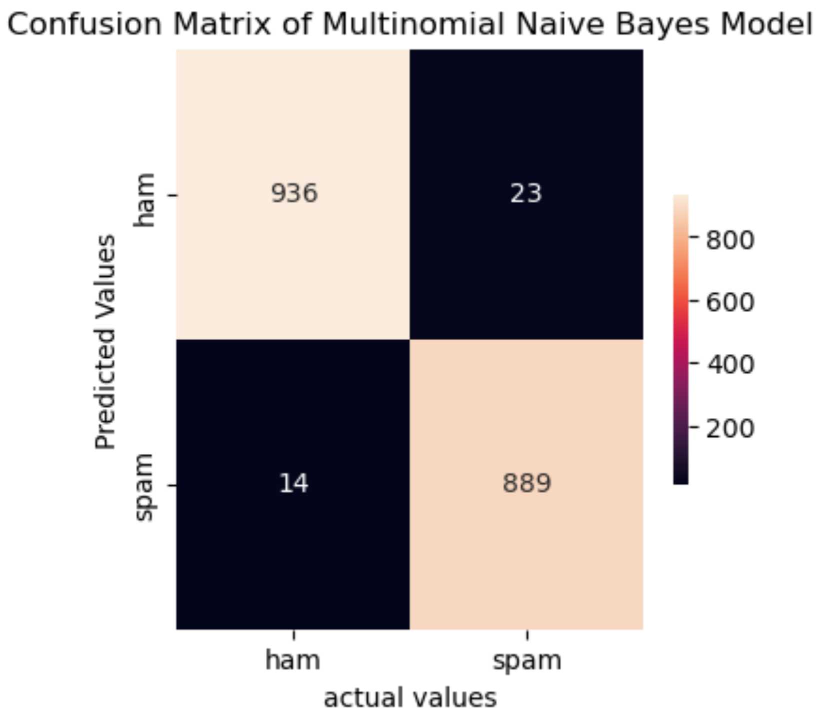
In [41]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       959
           1       0.97      0.98      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 [42]: cn = confusion_matrix(y_test, y_pred)
         cn
```

```
Out[42]: array([[936,  23],
                [ 14, 889]], dtype=int64)
```

```
In [43]: plt.figure(figsize=(4, 4))
         axis_labels = ['ham', 'spam']
         g = sns.heatmap(data=cn, xticklabels=axis_labels, yticklabels=axis_labels, annot = True,
         p = plt.title("Confusion Matrix of Multinomial Naive Bayes Model")
         p = plt.xlabel('actual values')
         p = plt.ylabel('Predicted Values')
```



```
In [44]: from sklearn.tree import DecisionTreeClassifier
         dt = DecisionTreeClassifier()
         cv1 = cross_val_score(dt, x, y, scoring='f1', cv=10)
         print(round(cv.mean(),3))
         print(round(cv.std(),3))
```

```
0.981
0.002
```

```
In [45]: dt.fit(x_train, y_train)
         y_pred1 = dt.predict(x_test)
```

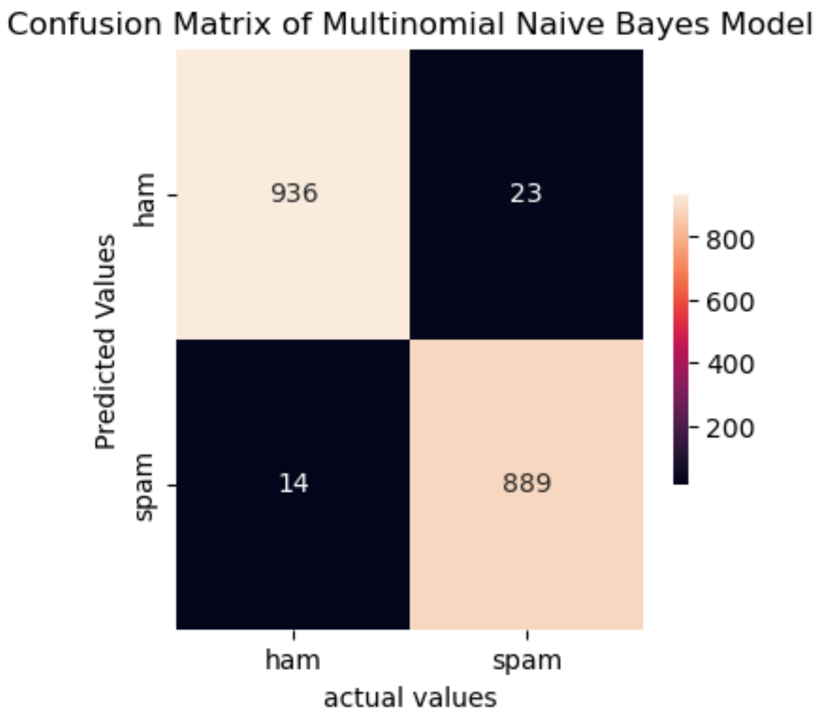
```
In [46]: print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       959
           1       0.97      0.98      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 [47]: cn = confusion_matrix(y_test, y_pred)
         cn
```

```
Out[47]: array([[936,  23],
                [ 14, 889]], dtype=int64)
```

```
In [48]: plt.figure(figsize=(4, 4))
         axis_labels = ['ham', 'spam']
         g = sns.heatmap(data=cn, xticklabels=axis_labels, yticklabels=axis_labels, annot = True,
         p = plt.title("Confusion Matrix of Multinomial Naive Bayes Model")
         p = plt.xlabel('actual values')
         p = plt.ylabel('Predicted Values')
```



```
In [49]: def predict_spam(sms):
             message = re.sub(pattern='[^A-Za-z]', repl= ' ', string=sms) # Filtering out special
             message = message.lower()
             words = message.split() # Tokenizer
             filtered_words = [word for word in words if word not in set(stopwords.words("english
             lenm_words = [wnl.lemmatize(word) for word in filtered_words]
             message = ' '.join(lenm_words)
             temp = tfidf.fit_transform([message]).toarray()
             return dt.predict(temp)
```