```
import pandas as pd
import numpy as np
```

```
df=pd.read_csv("SMSSpamCollection",sep="\t",names=['label','text'])
```

```
df
```

	label	text
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

```
df.shape
```

```
(5572, 2)
```

```
!pip install nltk
#natural language toolkit
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
```

```
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
```

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
sent="Hello Friends..!How are you?We will learning python today"
```

```
ps=PorterStemmer()
swords=stopwords.words('english')
```

```
from nltk.corpus.reader.tagged import word_tokenize

def clean_text(sent):
  tokens=word_tokenize(sent)# means each word are seperated in quotes(eg='hii','how')
  clean=[word for word in tokens
         if word.isdigit()or word.isalpha()]
  clean=[ps.stem(word) for word in clean
         if word is not swords]
  return clean
```

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
clean_text(sent)

     ['hello',
      'friend',
      'how',
      'are',
      'you',
      'we',
      'will',
      'learn',
      'python',
      'today']
```

```
from sklearn.feature_extraction.text import TfidfVectorizer  #convert data into numbers
```

```
tfid=TfidfVectorizer(analyzer=clean_text) #analyzer function mule data aadhi clean hoyil mg convert honar
```

```
x=df['text'] #input
y=df['label'] #output
```

```
x_new=tfid.fit_transform(x)
```

```
before=x.shape
print("before converting data is:\n",before)
after=x_new.shape
print("after converting data is:\n",after)

     before converting data is:
      (5572,)
     after converting data is:
      (5572, 6531)
```

```
y.value_counts()

     ham     4825
     spam     747
     Name: label, dtype: int64
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_new,y,random_state=0,test_size=0.25)
```

```
traindata=x_train.shape
print("The training data:",traindata)
testdata=x_test.shape
print("The training data:",testdata)

     The training data: (4179, 6531)
     The training data: (1393, 6531)
```

```
ytraindata=y_train.shape
print("The training data:",ytraindata)
ytestdata=y_test.shape
print("The training data:",ytestdata)

     The training data: (4179,)
     The training data: (1393,)
```

```
from sklearn.naive_bayes import GaussianNB
```

```
nb=GaussianNB()
```

```
nb.fit(x_train.toarray(),y_train)

     ▾ GaussianNB
     GaussianNB()
```

```
y_pred=nb.predict(x_test.toarray())
```

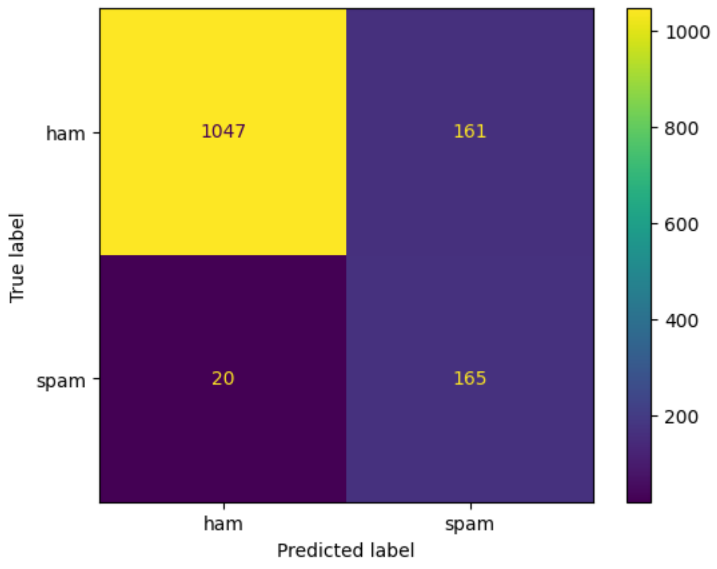
```
y_test.value_counts()

     ham     1208
     spam     185
     Name: label, dtype: int64
```

```
from sklearn.metrics import ConfusionMatrixDisplay


ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f8441359ab0>
```



```
from sklearn.metrics import classification_report,accuracy_score
print("The classification report is:\n",classification_report(y_test,y_pred))
```

```
The classification report is:
               precision    recall  f1-score   support

         ham       0.98      0.87      0.92      1208
        spam       0.51      0.89      0.65       185

    accuracy                           0.87      1393
   macro avg       0.74      0.88      0.78      1393
weighted avg       0.92      0.87      0.88      1393
```

```
accuracy=accuracy_score(y_test,y_pred)
print("The accuracy is:\n",accuracy)
```

```
The accuracy is:
 0.8700646087580761
```

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(random_state=0)
rf.fit(x_train,y_train)
y_pred=rf.predict(x_test)
```

```
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

```
     <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f843ee67040>
```

```
from sklearn.metrics import classification_report,accuracy_score
print("The classification report is:\n",classification_report(y_test,y_pred))
```

```
     The classification report is:
                   precision    recall  f1-score   support

             ham       0.98      1.00      0.99      1208
            spam       0.99      0.86      0.92       185

        accuracy                           0.98      1393
       macro avg       0.99      0.93      0.96      1393
    weighted avg       0.98      0.98      0.98      1393
```

```
accuracy=accuracy_score(y_test,y_pred)
print("The accuracy is:\n",accuracy)
```

```
     The accuracy is:
      0.9813352476669059
```

```
#Hyper Paramter tuning
from sklearn.model_selection import GridSearchCV


params={
    'criterion':['gini','entropy'],
    'max_features':['sqrt','log2'],
    'random_state':[0,1,2,3,4],
    'class_weight':['balanced','balanced_subsample']

}


grid=GridSearchCV(rf,param_grid=params,cv=5,scoring='accuracy')


grid.fit(x_train,y_train)
```

```
     ▸            GridSearchCV
     ▸ estimator: RandomForestClassifier
          ▸ RandomForestClassifier
```

```
rf=grid.best_estimator_


y_pred=rf.predict(x_test)


accuracy_score(y_test,y_pred)
```

```
     0.9784637473079684
```