In [1]:
```
import pandas as pd
```

In [4]:
```
df = pd.read_csv('SMSSpamCollection', sep= '\t',names= ['Label','Text'])
```

In [6]:
```
df
```

Out[6]:

	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

In [7]:
```
df.shape
```

Out[7]:
```
(5572, 2)
```

In [8]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\omnic\anaconda3\lib\site-packages (3.
7)
Requirement already satisfied: click in c:\users\omnic\anaconda3\lib\site-packages (f
rom nltk) (8.0.4)
Requirement already satisfied: joblib in c:\users\omnic\anaconda3\lib\site-packages
(from nltk) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\omnic\anaconda3\lib\site-p
ackages (from nltk) (2022.7.9)
Requirement already satisfied: tqdm in c:\users\omnic\anaconda3\lib\site-packages (fr
om nltk) (4.65.0)
Requirement already satisfied: colorama in c:\users\omnic\anaconda3\lib\site-packages
(from click->nltk) (0.4.6)
```

In [9]:
```
import nltk
```

In [18]:
```
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\omnic\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\omnic\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[18]:
```
True
```

In [19]:
```
sent = 'Hello Friends! How are you?'
```

In [20]:
```
from nltk.tokenize import word_tokenize
word_tokenize(sent)
```

Out[20]:
```
['Hello', 'Friends', '!', 'How', 'are', 'you', '?']
```

In [14]:
```
from nltk.corpus import stopwords
swords = stopwords.words('english')
swords
```

Out[14]:
```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
 'who',
 'whom',
 'this',
 'that',
 "that'll",
 'these',
 'those',
 'am',
 'is',
 'are',
 'was',
 'were',
 'be',
 'been',
 'being',
 'have',
 'has',
 'had',
 'having',
 'do',
 'does',
 'did',
 'doing',
```

```
                'a',
                'an',
                'the',
                'and',
                'but',
                'if',
                'or',
                'because',
                'as',
                'until',
                'while',
                'of',
                'at',
                'by',
                'for',
                'with',
                'about',
                'against',
                'between',
                'into',
                'through',
                'during',
                'before',
                'after',
                'above',
                'below',
                'to',
                'from',
                'up',
                'down',
                'in',
                'out',
                'on',
                'off',
                'over',
                'under',
                'again',
                'further',
                'then',
                'once',
                'here',
                'there',
                'when',
                'where',
                'why',
                'how',
                'all',
                'any',
                'both',
                'each',
                'few',
                'more',
                'most',
                'other',
                'some',
                'such',
                'no',
                'nor',
                'not',
                'only',
```

```
                'own',
                'same',
                'so',
                'than',
                'too',
                'very',
                's',
                't',
                'can',
                'will',
                'just',
                'don',
                "don't",
                'should',
                "should've",
                'now',
                'd',
                'll',
                'm',
                'o',
                're',
                've',
                'y',
                'ain',
                'aren',
                "aren't",
                'couldn',
                "couldn't",
                'didn',
                "didn't",
                'doesn',
                "doesn't",
                'hadn',
                "hadn't",
                'hasn',
                "hasn't",
                'haven',
                "haven't",
                'isn',
                "isn't",
                'ma',
                'mightn',
                "mightn't",
                'mustn',
                "mustn't",
                'needn',
                "needn't",
                'shan',
                "shan't",
                'shouldn',
                "shouldn't",
                'wasn',
                "wasn't",
                'weren',
                "weren't",
                'won',
                "won't",
                'wouldn',
                "wouldn't"]
```

In [21]:
```
clean = [word for word in word_tokenize(sent) if word not in swords]
clean
```

Out[21]:
```
['Hello', 'Friends', '!', 'How', '?']
```

In [22]:
```
from nltk.stem import PorterStemmer
ps = PorterStemmer()
clean = [ps.stem(word) for word in word_tokenize(sent) if word not in swords]
clean
```

Out[22]:
```
['hello', 'friend', '!', 'how', '?']
```

In [29]:
```
def clean_text(sent):
    tokens = word_tokenize(sent)
    clean = [word for word in tokens
             if word.isdigit() or word.isalpha()]
    clean = [ps.stem(word) for word in clean
             if word not in swords]
    return clean
```

In [30]:
```
sent = 'Hello Friends! How are you? We will learn Python Today!'
```

In [31]:
```
clean_text(sent)
```

Out[31]:
```
['hello', 'friend', 'how', 'we', 'learn', 'python', 'today']
```

In [32]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [33]:
```
tfidf = TfidfVectorizer(analyzer=clean_text)
```

In [34]:
```
x = df['Text']
y = df['Label']
```

In [35]:
```
x_new = tfidf.fit_transform(x)
```

In [36]:
```
x.shape
```

Out[36]:
```
(5572,)
```

In [37]:
```
x_new.shape
```

Out[37]:
```
(5572, 6513)
```

In [39]:
```
tfidf.get_feature_names_out()
```

Out[39]:
```
array(['0', '008704050406', '0089', ..., 'zyada', 'é', 'ü'], dtype=object)
```

In [40]:
```
y.value_counts()
```

Out[40]:
```
ham     4825
spam     747
Name: Label, dtype: int64
```

In [41]:
```
from sklearn.model_selection import train_test_split
```

In [42]:
```
x_train, x_test, y_train, y_test = train_test_split(x_new, y, random_state=0, test_siz
```

In [43]:
```
from sklearn.naive_bayes import GaussianNB
```

In [44]:
```
nb = GaussianNB()
```

In [45]:
```
nb.fit(x_train.toarray(), y_train)
```

Out[45]:
```
GaussianNB()
```

In [46]:
```
y_pred = nb.predict(x_test.toarray())
```

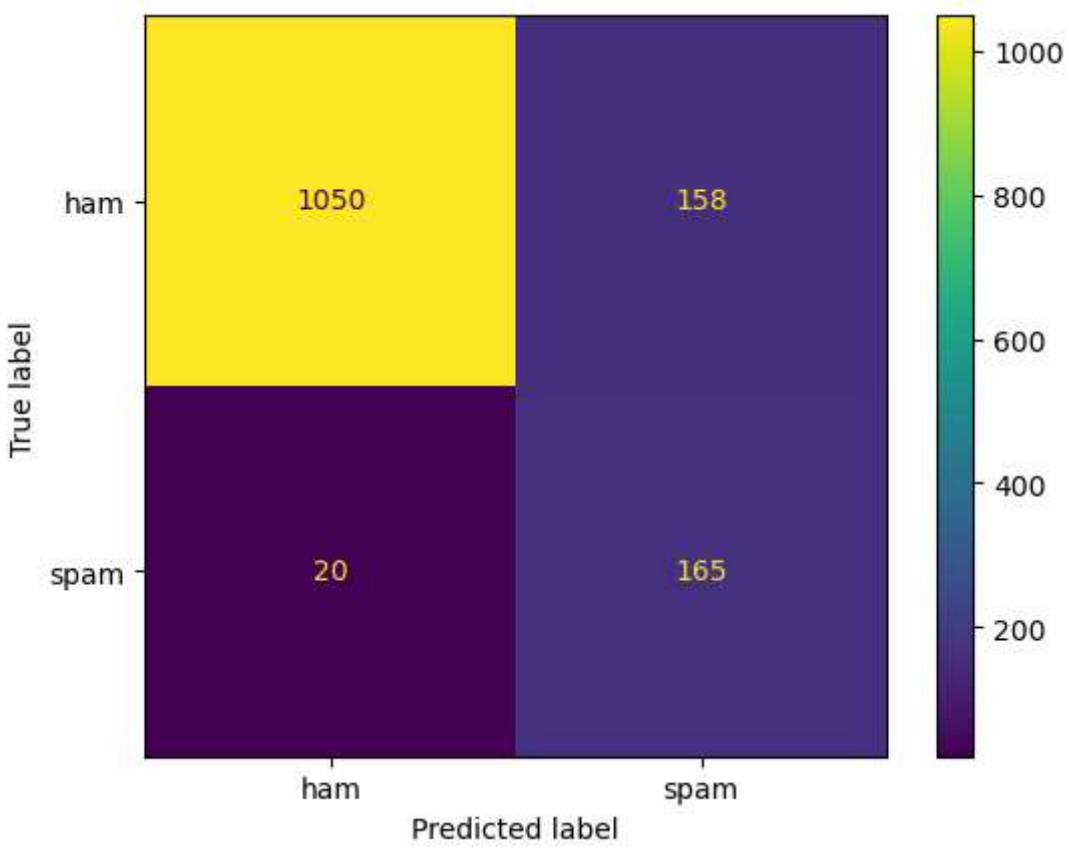
In [49]:
```
y_test.value_counts()
```

Out[49]:
```
ham     1208
spam     185
Name: Label, dtype: int64
```

In [50]:
```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score, classification_rep
```

In [51]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

Out[51]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x27e85d57e10>
```



In [53]:
```
print(classification_report(y_test, y_pred))
```

	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	0.75	0.88	0.79	1393
weighted avg	0.92	0.87	0.89	1393

In [54]:
```
accuracy_score(y_test, y_pred)
```

Out[54]:
```
0.8722182340272793
```

In [55]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [56]:
```
rf = RandomForestClassifier(random_state=0)
```

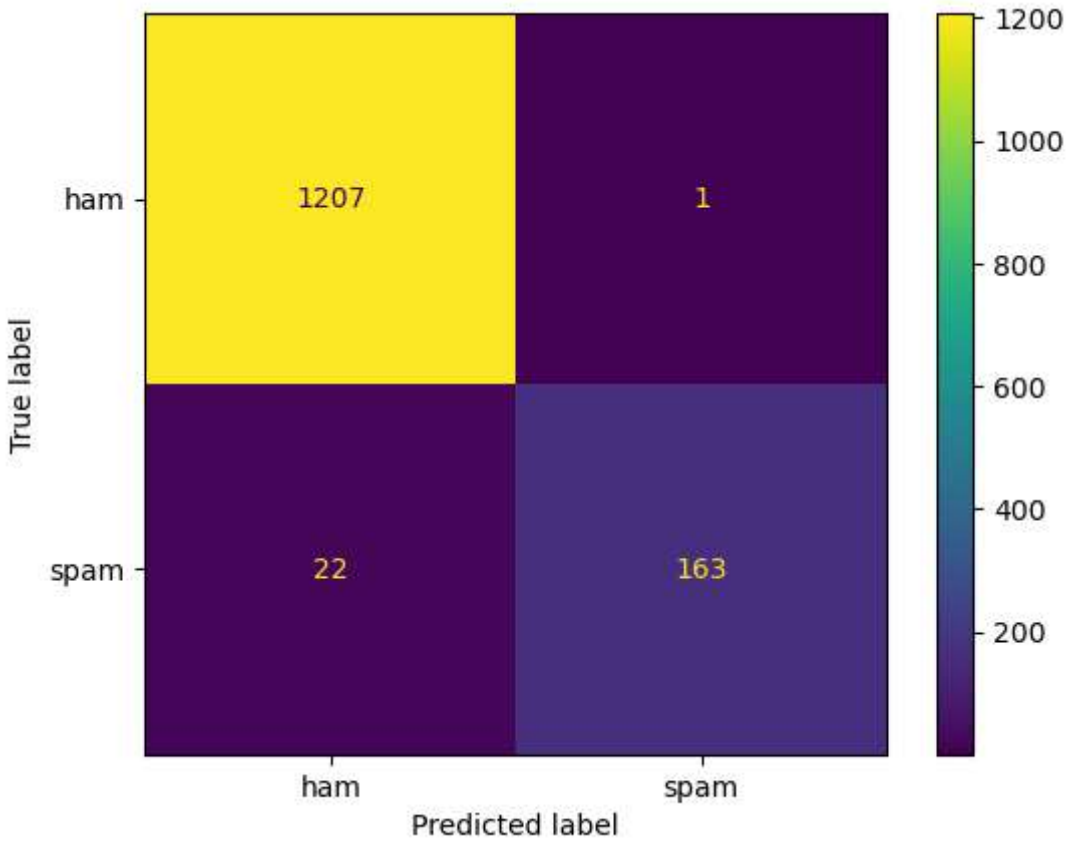
In [58]:
```
rf.fit(x_train, y_train)
```

Out[58]:
```
RandomForestClassifier
RandomForestClassifier(random_state=0)
```

In [59]:
```
y_pred = rf.predict(x_test)
```

In [60]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

Out[60]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x27e88d922d0>
```



In [61]:
```
accuracy_score(y_test, y_pred)
```

Out[61]:
```
0.9834888729361091
```

In [62]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99      1208
        spam       0.99      0.88      0.93       185

    accuracy                           0.98      1393
   macro avg       0.99      0.94      0.96      1393
weighted avg       0.98      0.98      0.98      1393
```

In [63]:
```
from sklearn.linear_model import LogisticRegression
log = LogisticRegression()
log.fit(x_train, y_train)
y_pred = log.predict(x_test)
accuracy_score(y_test, y_pred)
```

Out[63]:
```
0.9641062455132807
```

In [64]:
```
from sklearn.model_selection import GridSearchCV
```

In [65]:
```
params = {
    'criterion': ['gini','entropy'],
    'max_features': ['sqrt', 'log2'],
    'random_state': [0,1,2,3,4],
    'class_weight': ['balanced', 'balanced_subsample']
}
```

In [66]:
```
grid = GridSearchCV(rf, param_grid=params, cv=5, scoring='accuracy')
```

In [ ]:
```
grid.fit(x_train, y_train)
```

In [ ]:
```
rf = grid.best_estimator_
y_pred = rf.predict(x_test)
accuracy_score(y_test, y_pred)
```

In [ ]:
```

```