In [1]:
```
import pandas as p
import numpy as n
import seaborn as s
import matplotlib.pyplot as plt
df = p.read_csv("SMSSpamCollection",sep='\t',names=['label','text'])
df
```

Out[1]:

| | 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 [2]:
```
df.shape
```

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

In [3]:
```
#!pip install nltk
import nltk
```

In [5]:
```
#nltk.download("stopwords")
#nltk.download("punkt")
from nltk.corpus import stopwords
sword = stopwords.words("english")
sword
```

Out[5]:
```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
```

In [10]:
```
from nltk.tokenize import word_tokenize
```

In [17]:
```
from nltk.stem import PorterStemmer
ps = PorterStemmer()
```

In [27]:
```
def clean_text(sent):
    clean = word_tokenize(sent)
    clean = [word for word in clean if word.isdigit() or word.isalpha()]
    clean = [ps.stem(word) for word in clean if word not in sword]
    return clean
```

In [28]:
```
clean_text("how are yours dad > : going to homes")
```

Out[28]:
```
['dad', 'go', 'home']
```

In [32]:
```
print("hi")
```

```
hi
```

In [33]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [34]:
```
tf = TfidfVectorizer(analyzer = clean_text)
```

In [35]:
```
x = df['text']
y = df['label']
x_new = tf.fit_transform(x)
x_new.shape
```

Out[35]:
```
(5572, 6513)
```

In [37]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_new,y,test_size=0.25,rand
```

In [38]:
```
from sklearn.naive_bayes import GaussianNB
```

In [40]:
```
nb = GaussianNB()
nb.fit(x_train.toarray(),y_train)
```

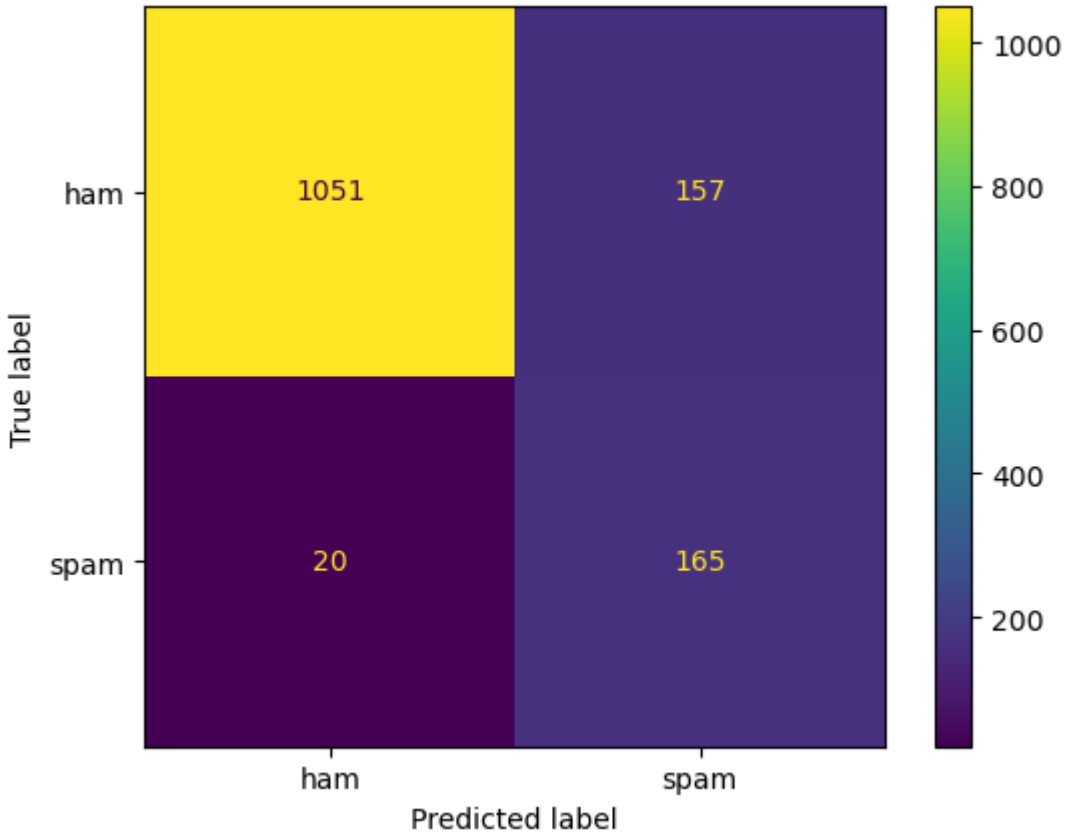
Out[40]:
```
GaussianNB()
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [41]:
```
y_predict = nb.predict(x_test.toarray())
```

In [44]:
```
from sklearn.metrics import ConfusionMatrixDisplay,classification_report
ConfusionMatrixDisplay.from_predictions(y_test,y_predict)
```

Out[44]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x22d48d
80fd0>
```



In [45]:
```
print(classification_report(y_test,y_predict))
```

```
              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
```

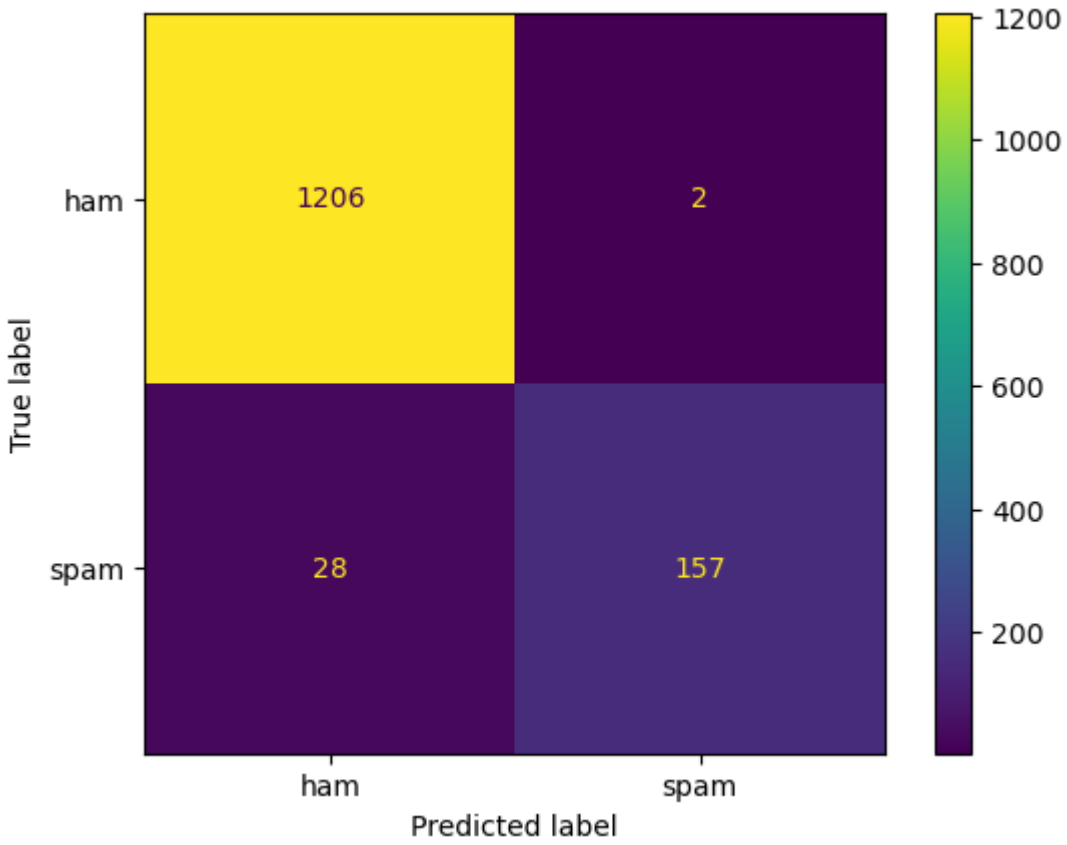
In [47]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [48]:
```
RFC = RandomForestClassifier()
```

In [49]:
```
RFC.fit(x_train,y_train)
y_predict_RFC = RFC.predict(x_test)
```

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

Out[51]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x22d47e
42110>
```



In [52]:
```
print(classification_report(y_test,y_predict_RFC))
```

```
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99      1208
        spam       0.99      0.85      0.91       185

    accuracy                           0.98      1393
   macro avg       0.98      0.92      0.95      1393
weighted avg       0.98      0.98      0.98      1393
```

In [53]:
```
from sklearn.model_selection import GridSearchCV
```

In [56]:
```
para = {
    'criterion':['gini','entropy','log_loss'],
    'class_weight':['balanced','balanced_subsample']
}

grid = GridSearchCV(RFC,param_grid = para,cv=5,scoring='accuracy')
```

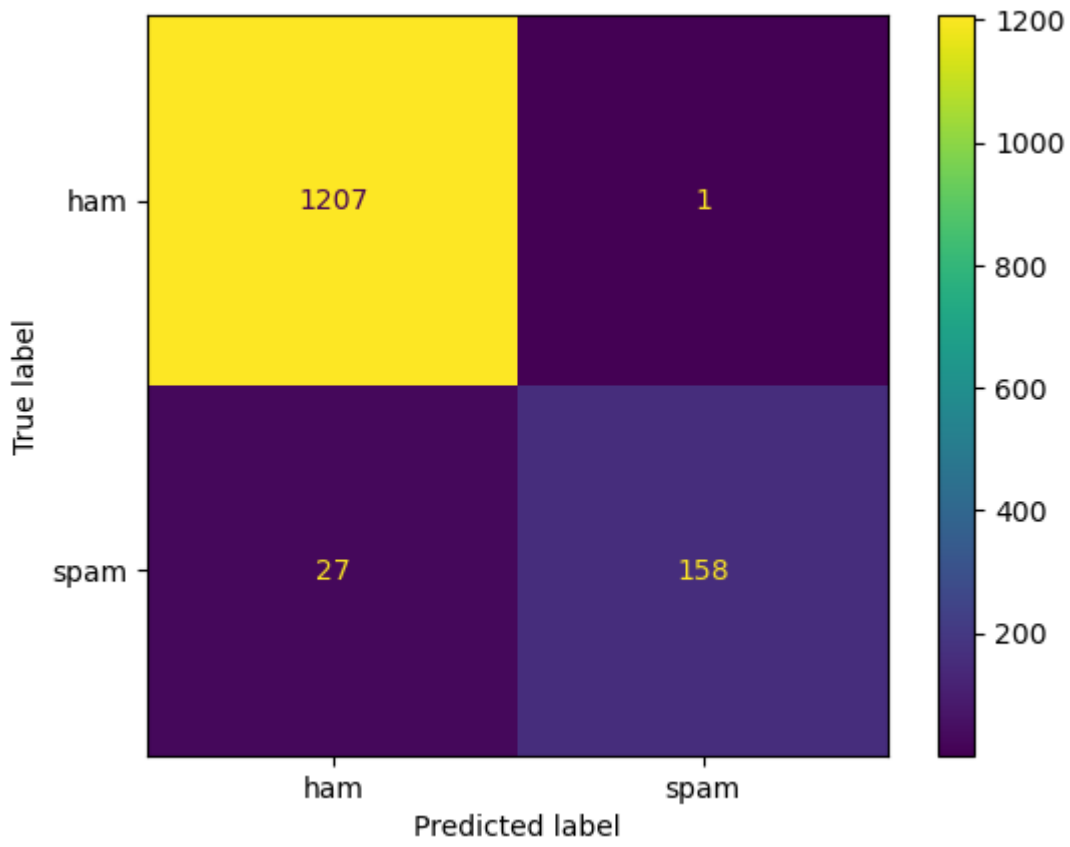
In [57]:
```
grid.fit(x_train,y_train)
```

Out[57]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'class_weight': ['balanced', 'balanced_subsampl
e'],
                         'criterion': ['gini', 'entropy', 'log_loss']},
             scoring='accuracy')
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [59]:
```
y_predict_grid = grid.predict(x_test)
ConfusionMatrixDisplay.from_predictions(y_test,y_predict_grid)
```

Out[59]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x22d4b1
d1f50>
```




In [60]:
```
print(classification_report(y_test,y_predict_grid))
```

```
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99      1208
        spam       0.99      0.85      0.92       185

    accuracy                           0.98      1393
   macro avg       0.99      0.93      0.95      1393
weighted avg       0.98      0.98      0.98      1393
```

In [ ]:
```

```