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

In [8]:
```
data = pd.read_csv('SMSSpamCollection', sep = '\t', names=['label', 'text'])
```

In [9]:
```
data
```

Out[9]:

	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 [10]:
```
data.shape
```

Out[10]:
```
(5572, 2)
```

In [11]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\dell\anaconda3\lib\site-packages
(3.7)
Requirement already satisfied: click in c:\users\dell\anaconda3\lib\site-packages
(from nltk) (8.0.4)
Requirement already satisfied: joblib in c:\users\dell\anaconda3\lib\site-packages
(from nltk) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\dell\anaconda3\lib\site
-packages (from nltk) (2022.7.9)
Requirement already satisfied: tqdm in c:\users\dell\anaconda3\lib\site-packages
(from nltk) (4.65.0)
Requirement already satisfied: colorama in c:\users\dell\anaconda3\lib\site-packag
es (from click->nltk) (0.4.6)
```

In [12]:
```
import nltk
```

In [23]:
```
nltk.download('stopwords')
nltk.download('punkt')
```

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

Out[23]:
```
True
```

In [24]:
```
sent = 'Hello friends! How are you?'
```

In [25]:
```
from nltk.tokenize import word_tokenize
word_tokenize(sent)
```

Out[25]:
```
['Hello', 'friends', '!', 'How', 'are', 'you', '?']
```

In [29]:
```
from nltk.corpus import stopwords
swords = stopwords.words('english')
```

In [31]:
```
clean = [word for word in word_tokenize(sent) if word not in swords]
```

In [32]:
```
clean
```

Out[32]:
```
['Hello', 'friends', '!', 'How', '?']
```

In [33]:
```
from nltk.stem import PorterStemmer
ps = PorterStemmer()
clean = [ps.stem(word) for word in word_tokenize(sent) if word not in swords]
```

In [34]:
```
clean
```

Out[34]:
```
['hello', 'friend', '!', 'how', '?']
```

In [40]:
```
sent = 'Hello friends! How are you? We are doing the work today.'
```

In [43]:
```
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 [44]:
```
clean_text(sent)
```

Out[44]:
```
['hello', 'friend', 'how', 'we', 'work', 'today']
```

In [45]:
```
#Pre-processing
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [54]:
```
tfidf = TfidfVectorizer(analyzer=clean_text)
```

In [55]:
```
x = data['text']
y = data['label']
```

In [56]:
```
x_new = tfidf.fit_transform(x)
```

In [57]:
```
x.shape
```

Out[57]:
```
(5572,)
```

In [58]:
```
x_new.shape
```

Out[58]:
```
(5572, 6513)
```

In [59]:
```
 y.value_counts()
```

Out[59]:
```
ham     4825
spam     747
Name: label, dtype: int64
```

In [62]:
```
#Cross-Validation
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_si
```

In [63]:
```
x_train.shape
```

Out[63]:
```
(4179, 6513)
```

In [64]:
```
x_test.shape
```

Out[64]:
```
(1393, 6513)
```

In [65]:
```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
```

In [66]:
```
nb.fit(x_train.toarray(), y_train)
```

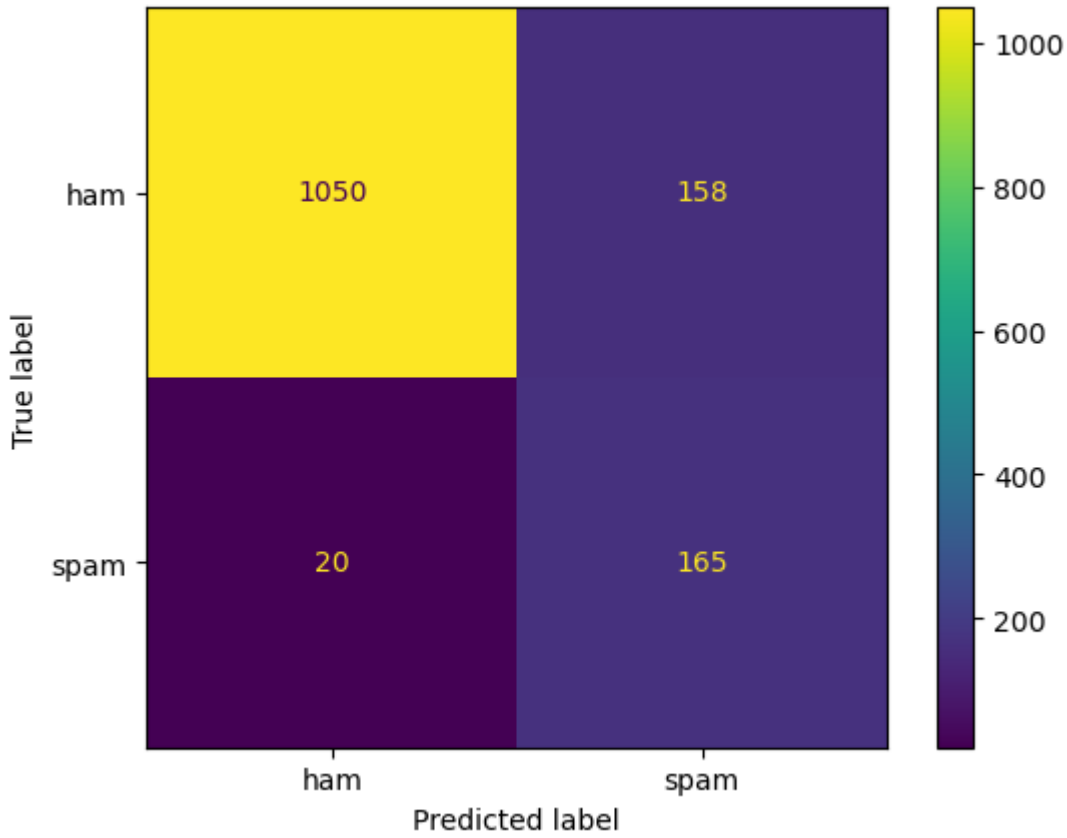
Out[66]:
```
▾ GaussianNB

GaussianNB()
```

In [68]:
```
y_pred = nb.predict(x_test.toarray())
```

In [69]:
```
y_test.value_counts()
```

Out[69]:
```
ham     1208
spam     185
Name: label, dtype: int64
```

In [70]:
```
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_predictions(y_test, y_pred);
```



In [71]:
```
from sklearn.metrics import accuracy_score, classification_report
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 [72]:
```
accuracy_score(y_test, y_pred)
```

Out[72]:
```
0.8722182340272793
```

In [73]:
```
from sklearn.ensemble import RandomForestClassifier
```

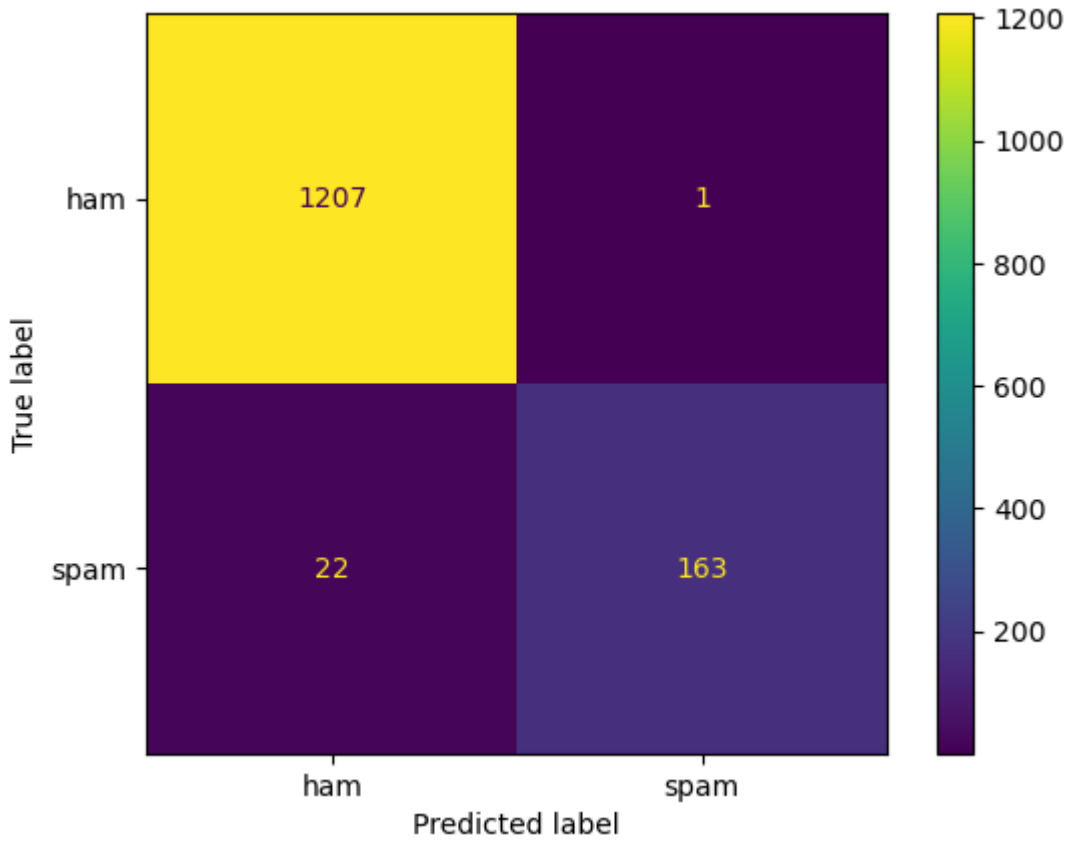
In [74]:
```
rf = RandomForestClassifier(random_state=0)
```

In [76]:
```
rf.fit(x_train, y_train)
```

Out[76]:
```
RandomForestClassifier
RandomForestClassifier(random_state=0)
```

In [77]:
```
y_pred = rf.predict(x_test)
```

In [78]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred);
```



In [79]:
```
from sklearn.metrics import accuracy_score, classification_report
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 [80]:
```
accuracy_score(y_test, y_pred)
```

Out[80]:
```
0.9834888729361091
```

In [83]:
```
#Logistic Regression
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[83]:
```
0.9641062455132807
```

In [84]:
```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
decision = DecisionTreeClassifier()
decision.fit(x_train, y_train)
y_pred = decision.predict(x_test)
accuracy_score(y_test, y_pred)
```

Out[84]:
```
0.9569274946159368
```