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

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
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)
```

```
import nltk
```

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

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

```
sent = 'How are you friends?'
```

```
from nltk.tokenize import word_tokenize
word_tokenize(sent)
```

```
['How', 'are', 'you', 'friends', '?']
```

```
from nltk.corpus import stopwords
swords = stopwords.words('english')
```

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

```
['How', 'friends', '?']
```

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

```
['how', 'friend', '?']
```

```
sent = 'Hello friends! How are you? We will learning python today'
```

```
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
```

```
clean_text(sent)
```

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

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf = TfidfVectorizer(analyzer=clean_text)
x = df['text']
y = df['label']
```

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

```
x.shape
```

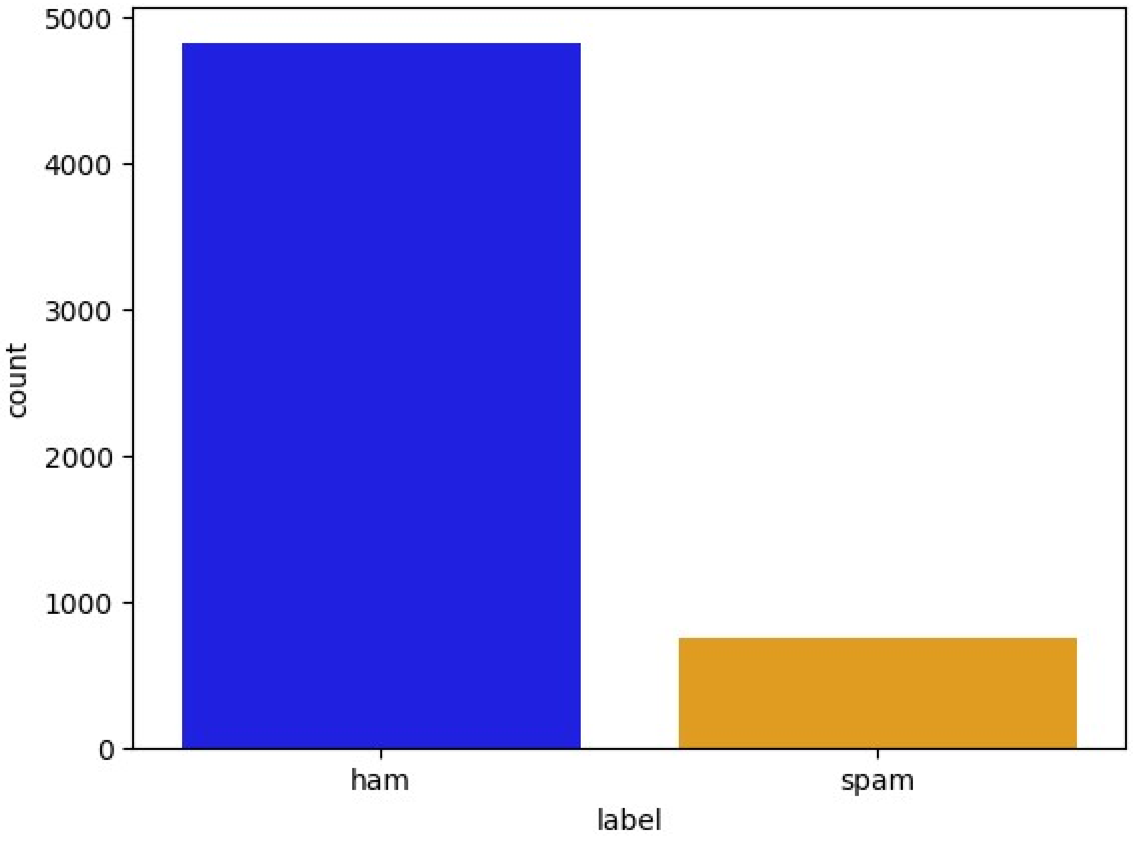
```
(5572,)
```

```
x_new.shape
```

```
(5572, 6513)
```

```
import seaborn as sns
sns.countplot(x=y, palette=['blue','orange'])
```

```
<ipython-input-72-2b8a1a03d0dd>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.countplot(x=y, palette=['blue','orange'])
<Axes: xlabel='label', ylabel='count'>
```



```
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,random_state=1)
```

```
print(f"Size of splitted data")
print(f"x_train {x_train.shape}")
print(f"y_train {y_train.shape}")
print(f"y_test {x_test.shape}")
print(f"y_test {y_test.shape}")
```

```
Size of splitted data
x_train (4179, 6513)
y_train (4179,)
y_test (1393, 6513)
y_test (1393,)
```

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

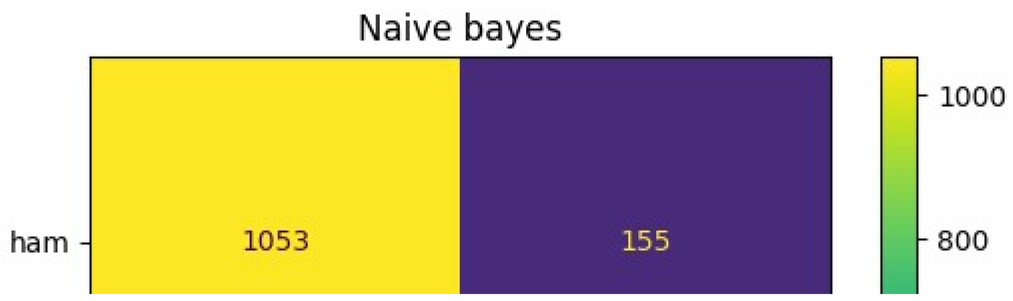
```
nb=GaussianNB()
nb.fit(x_train.toarray(),y_train)
y_pred_nb=nb.predict(x_test.toarray())
```

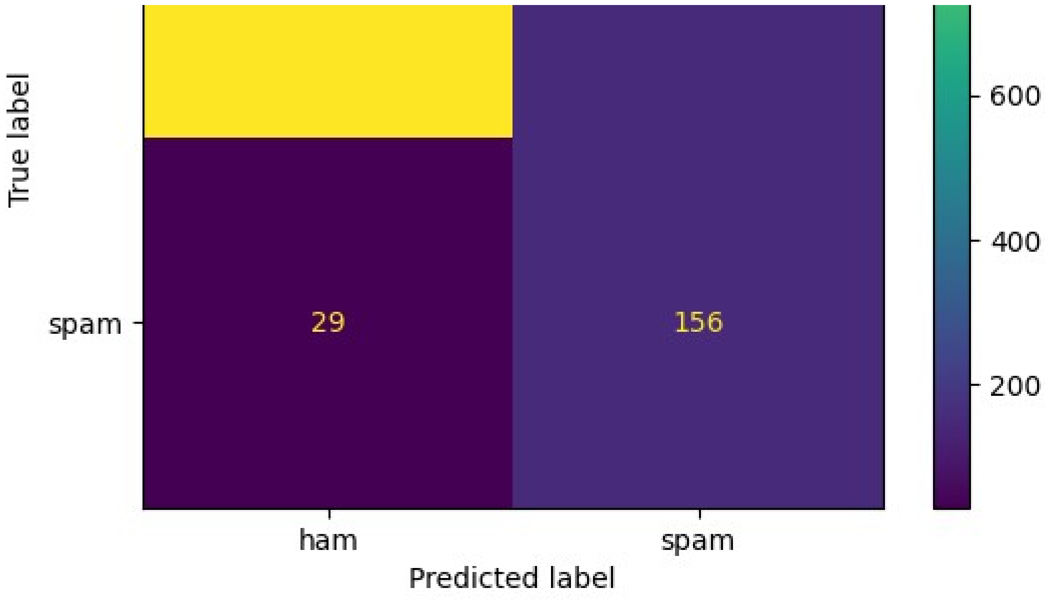
```
y_test.value_counts()
```

label	count
ham	1208
spam	185

dtype: int64

```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_nb)
plt.title('Naive bayes')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_nb)}")
print(classification_report(y_test,y_pred_nb))
```

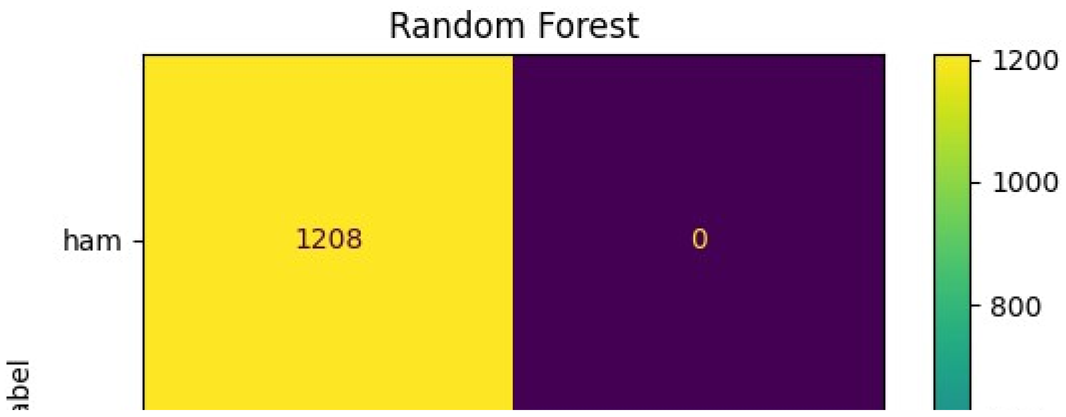


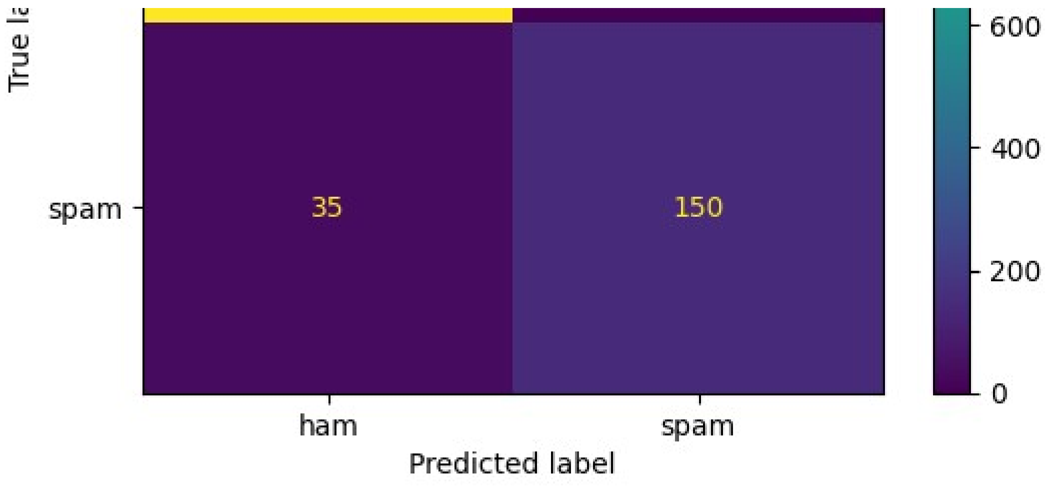


```
 Accuracy is 0.867910983488873
              precision    recall  f1-score   support

         ham       0.97      0.87      0.92      1208
        spam       0.50      0.84      0.63       185

    accuracy                           0.87      1393
   macro avg       0.74      0.86      0.77      1393
weighted avg       0.91      0.87      0.88      1393
```

```
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier(random_state=1)
model_rf.fit(x_train,y_train)
RandomForestClassifier(random_state=1)
y_pred_rf = model_rf.predict(x_test)
```

```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_rf)
plt.title('Random Forest')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_rf)}")
print(classification_report(y_test,y_pred_rf))
```

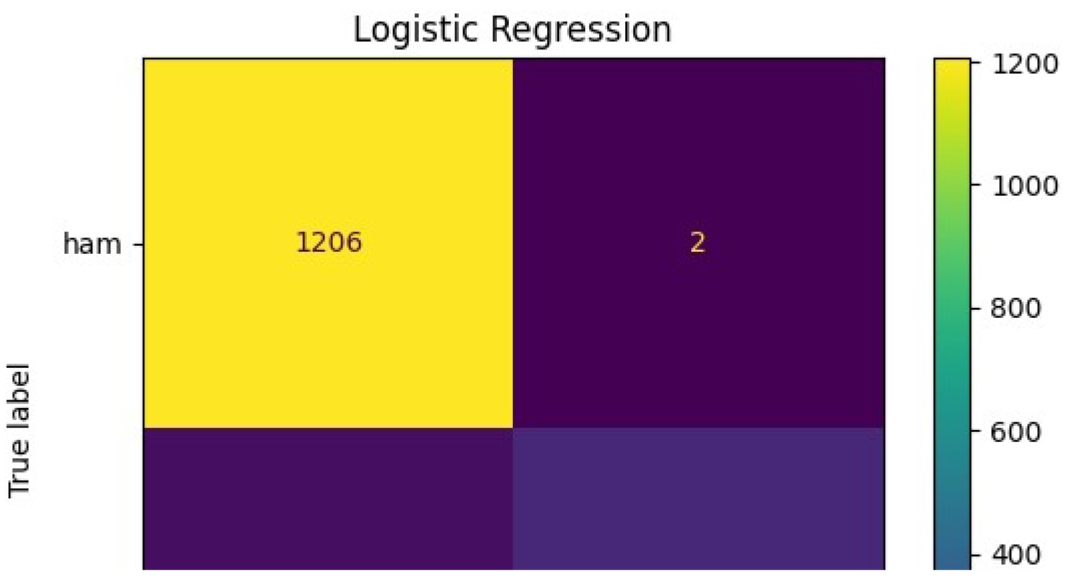


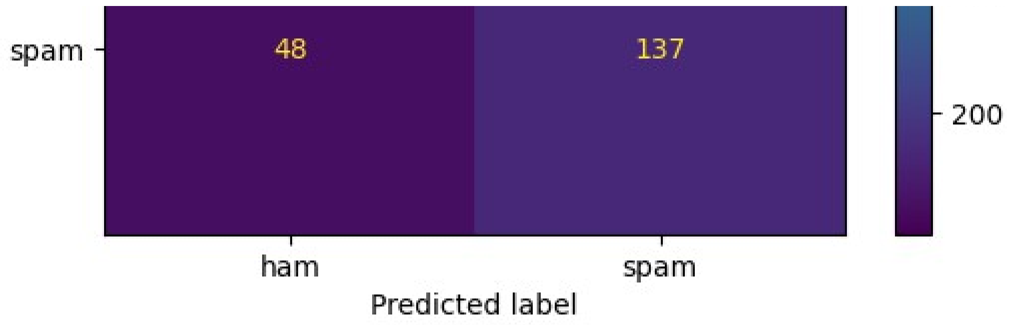


```
 Accuracy is 0.9748743718592965
              precision    recall  f1-score   support

         ham       0.97      1.00      0.99      1208
        spam       1.00      0.81      0.90       185

    accuracy                           0.97      1393
   macro avg       0.99      0.91      0.94      1393
weighted avg       0.98      0.97      0.97      1393
```

```
from sklearn.linear_model import LogisticRegression
model_lr=LogisticRegression(random_state=1)
model_lr.fit(x_train,y_train)
y_pred_lr=model_lr.predict(x_test)
```

```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_lr)
plt.title('Logistic Regression')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_lr)}")
print(classification_report(y_test,y_pred_lr))
```





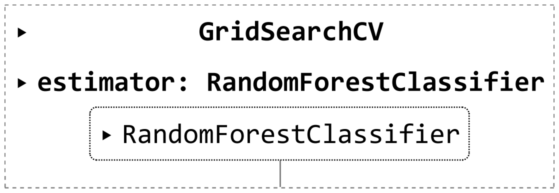
```
 Accuracy is 0.9641062455132807
              precision    recall  f1-score   support

         ham       0.96      1.00      0.98      1208
        spam       0.99      0.74      0.85       185

    accuracy                           0.96      1393
   macro avg       0.97      0.87      0.91      1393
weighted avg       0.96      0.96      0.96      1393
```

```
from sklearn.model_selection import GridSearchCV
```

```
para={

      'criterion':['gini','entropy','log_loss'],
      'class_weight':['balanced','balanced_subsample']
}
```

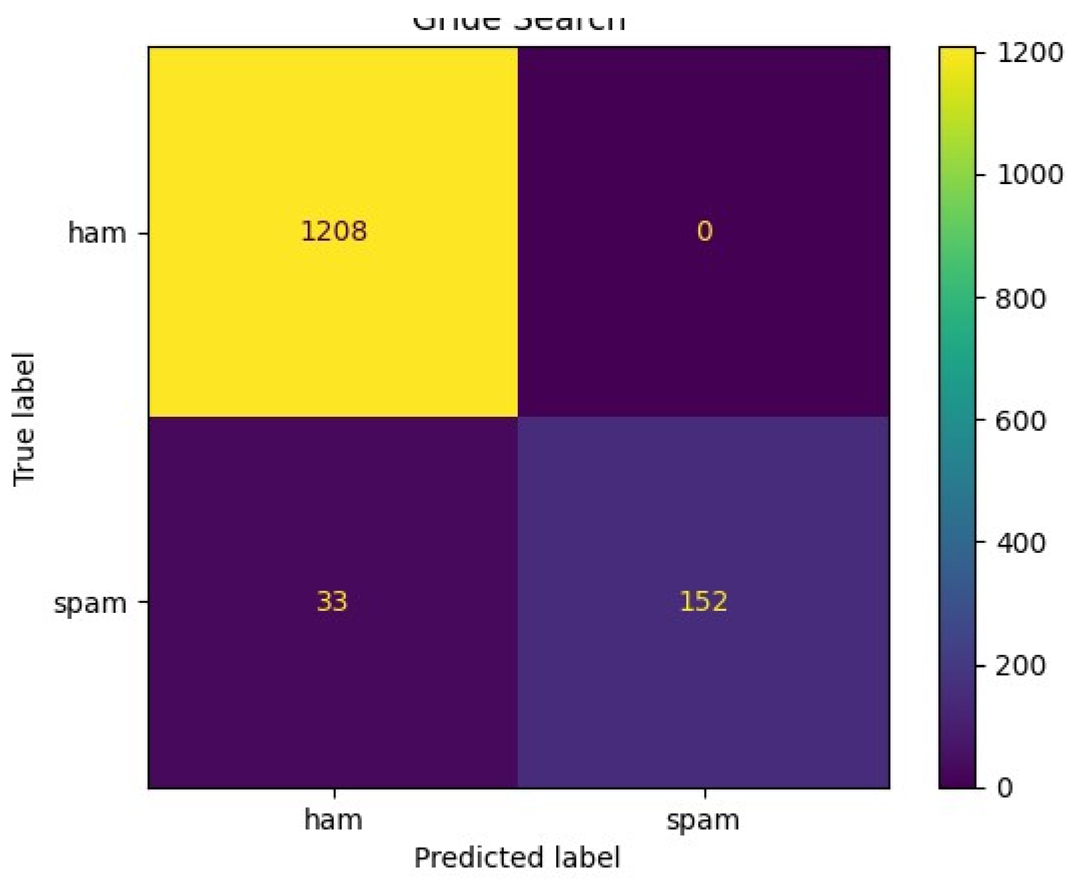
```
grid=GridSearchCV(model_rf,param_grid=para,cv=5, scoring='accuracy')
```

```
grid.fit(x_train,y_train)
```



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

```
rf = grid.best_estimator_
y_pred_grid = rf.predict(x_test)
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_grid)
plt.title('Gride Search')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_grid)}")
print(classification_report(y_test,y_pred_grid))
```



```
Accuracy is 0.9763101220387652
              precision    recall  f1-score   support

         ham       0.97      1.00      0.99      1208
        spam       1.00      0.82      0.90       185

    accuracy                           0.98      1393
   macro avg       0.99      0.91      0.94      1393
weighted avg       0.98      0.98      0.98      1393
```