In [1]:

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

In [2]:

```
df = pd.read_csv(r"E:\Academics\Sem 05\ML LAB\Practical No. 01 (Regression Techniqu
                 sep='\t',names=['label','text'])
```

In [3]:

```
df
```

Out[3]:

	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 [4]:

```
df.shape
```

Out[4]:

```
(5572, 2)
```

In [3]:

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

In [5]:

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

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

Out[5]:

```
True
```

In [8]:

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

In [9]:

```
from nltk.stem import PorterStemmer
ps = PorterStemmer()
from nltk.corpus import stopwords
swords = stopwords.words('english')
from nltk.tokenize import word_tokenize
word_tokenize(sent)
```

Out[9]:

```
['Hello',
 'friends',
 '!',
 'How',
 'are',
 'you',
 '?',
 'We',
 'will',
 'learning',
 'python',
 'today']
```

In [10]:

```python
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 [11]:

```python
clean_text(sent)
```

Out[11]:

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

In [12]:

```python
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(analyzer=clean_text)
x = df['text']
y = df['label']
```

In [13]:

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

In [14]:

```python
x.shape
```

Out[14]:

```
(5572,)
```

In [15]:

```python
x_new.shape
```

Out[15]:

```
(5572, 6513)
```

In [16]:

```python
x_new
```

Out[16]:

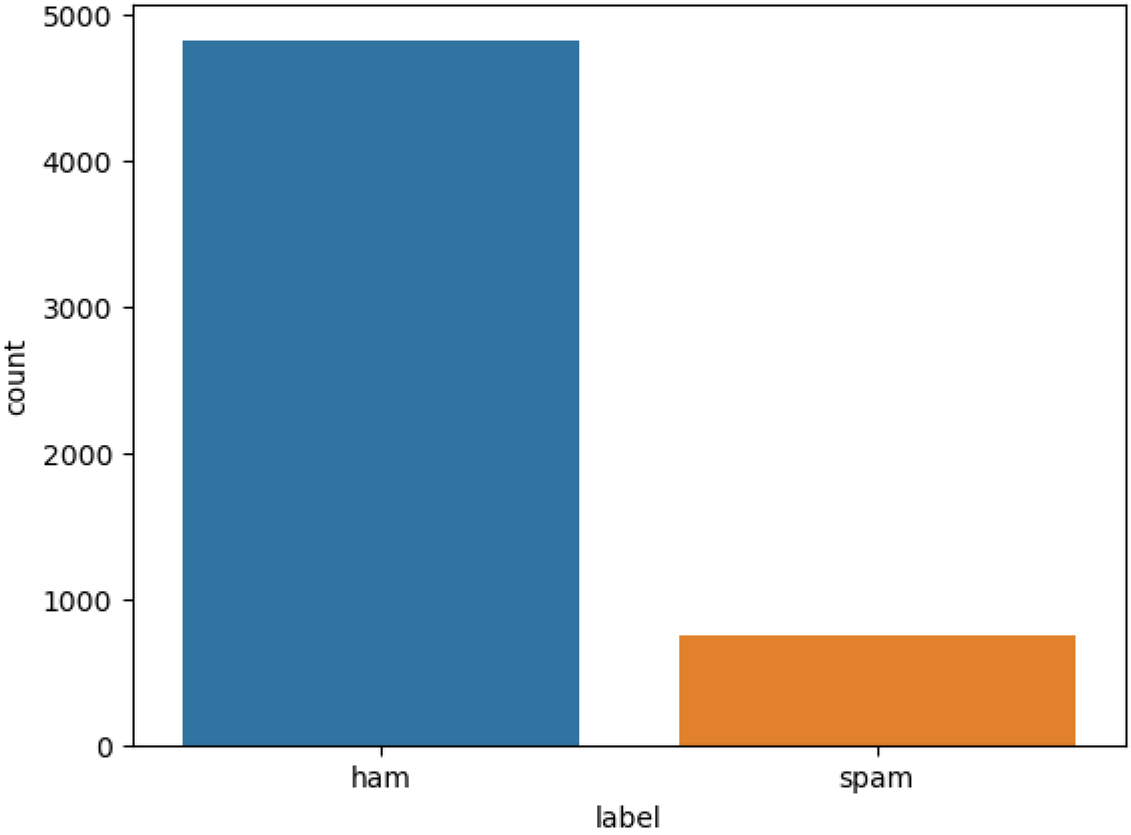
```
<5572x6513 sparse matrix of type '<class 'numpy.float64'>'
	with 52578 stored elements in Compressed Sparse Row format>
```

In [17]:

```python
import seaborn as sns
sns.countplot(x=y)
```

Out[17]:

```
<Axes: xlabel='label', ylabel='count'>
```



In [18]:
```
#cross validation
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)
```

In [19]:
```
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,)
```

In [20]:
```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train.toarray(),y_train)
y_pred_nb = nb.predict(x_test.toarray())
```

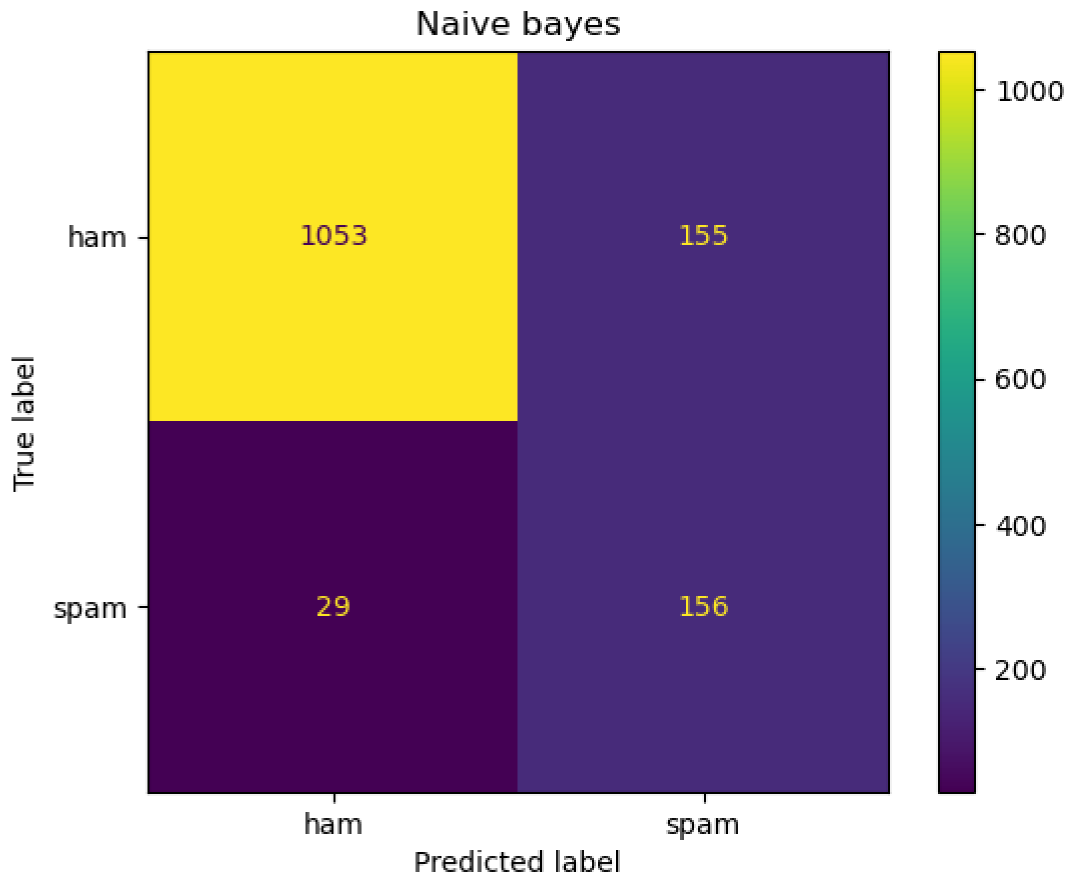
In [21]:
```
y_test.value_counts()
```

Out[21]:
```
label
ham     1208
spam     185
Name: count, dtype: int64
```

In [22]:
```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
```

In [23]:
```
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))
```



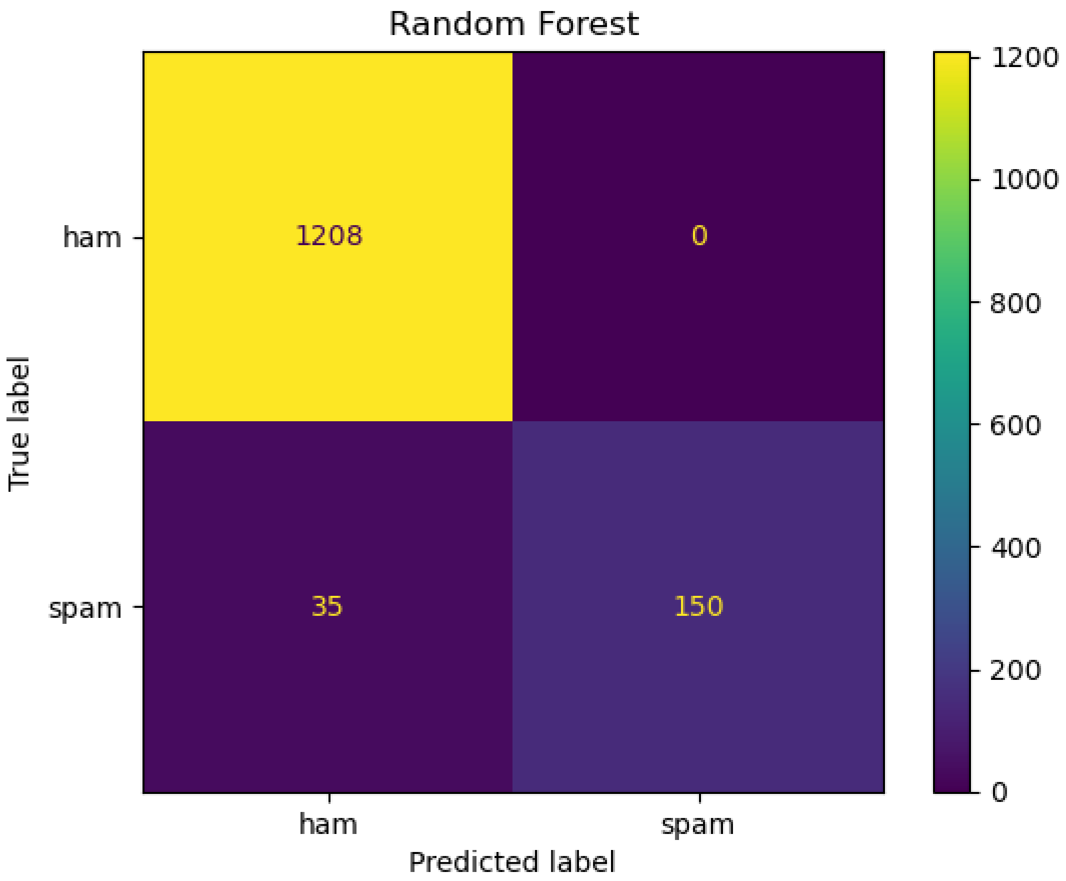
```
 Accuracy is 0.867910983488873
              precision    recall  f1-score   support

         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
```

In [24]:
```
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier(random_state=1)
model_rf.fit(x_train,y_train)
```

Out[24]:
```
▾ RandomForestClassifier
RandomForestClassifier(random_state=1)
```

In [25]:
```
y_pred_rf = model_rf.predict(x_test) #float
```

In [26]:
```
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
```

In [27]:
```
from sklearn.model_selection import GridSearchCV
```

In [28]:
```
para = {

    'criterion':['gini', 'entropy','log_loss'],
  # 'max_features': ['sqrt','log2'],
    #'random_state': [0,1,2,3,4],
    'class_weight':['balanced','balanced_subsample']
}
```

In [29]:
```
grid = GridSearchCV(model_rf, param_grid=para, cv=5, scoring='accuracy')
```

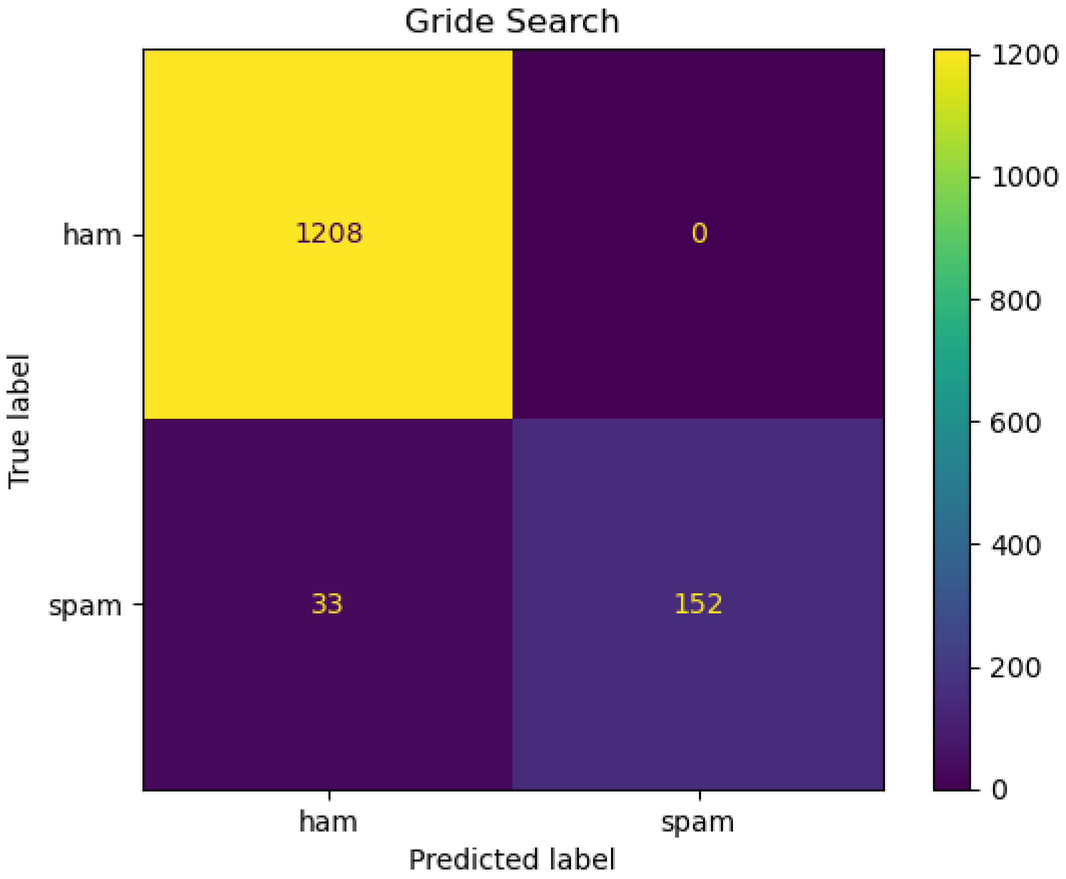
In [30]:
```
grid.fit(x_train,y_train)
```

Out[30]:
```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [31]:
```
rf = grid.best_estimator_
```

In [32]:
```
y_pred_grid = rf.predict(x_test)
```

In [33]:
```
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
```

In [ ]: