In [3]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
```

In [4]:

```python
data = pd.read_csv("iris.csv")
data.head()
```

Out[4]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [5]:

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [6]:

```python
data.describe()
```

Out[6]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [7]:

```python
data['species'].value_counts()
```

```
Out[7]:  species
         Iris-setosa        50
         Iris-versicolor    50
         Iris-virginica     50
         Name: count, dtype: int64
```

```
In [8]:  X = data.drop('species', axis=1)
         y = data['species']
```

```
In [9]:  X_train, X_test, y_train, y_test = train_test_split(
             X, y, test_size=0.3, random_state=42
         )
```

```
In [10]: nb = GaussianNB()
         nb.fit(X_train, y_train)
```

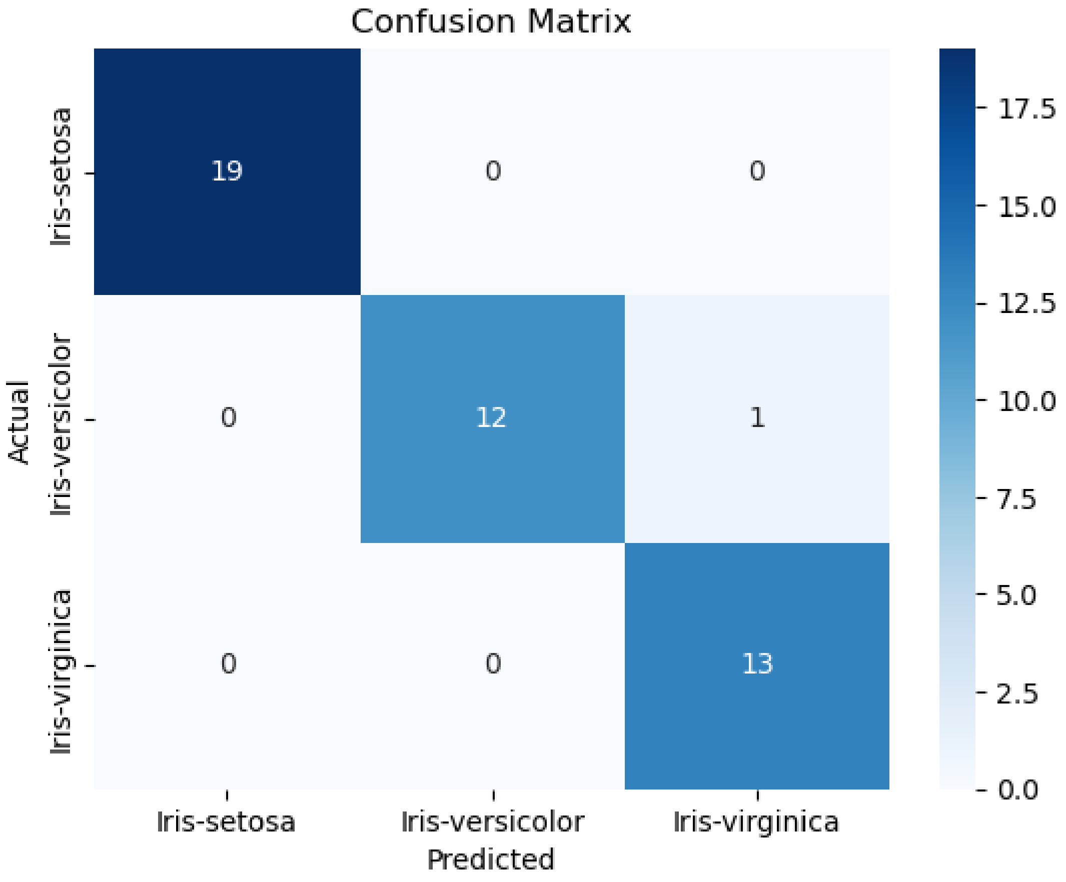
```
Out[10]: ▾ GaussianNB  i ?

         GaussianNB()
```

```
In [11]: y_pred = nb.predict(X_test)
```

```
In [12]: cm = confusion_matrix(y_test, y_pred)
         cm
```

```
Out[12]: array([[19,  0,  0],
                [ 0, 12,  1],
                [ 0,  0, 13]])
```

```
In [13]: sns.heatmap(cm, annot=True, cmap='Blues', fmt='d',
                     xticklabels=nb.classes_,
                     yticklabels=nb.classes_)
         plt.xlabel("Predicted")
         plt.ylabel("Actual")
         plt.title("Confusion Matrix")
         plt.show()
```



In [14]:
```
for i, label in enumerate(nb.classes_):
    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = cm.sum() - (TP + FP + FN)

    print(f"\nClass: {label}")
    print("TP:", TP)
    print("FP:", FP)
    print("FN:", FN)
    print("TN:", TN)
```

```
Class: Iris-setosa
TP: 19
FP: 0
FN: 0
TN: 26

Class: Iris-versicolor
TP: 12
FP: 0
FN: 1
TN: 32

Class: Iris-virginica
TP: 13
FP: 1
FN: 0
TN: 31
```

In [15]:
```
accuracy = accuracy_score(y_test, y_pred)
accuracy
```

Out[15]:
```
0.9777777777777777
```

In [16]:
```
error_rate = 1 - accuracy
error_rate
```

Out[16]:
```
0.022222222222222254
```

In [17]:
```
precision = precision_score(y_test, y_pred, average='macro')
precision
```

Out[17]:
```
0.9761904761904763
```

In [18]:
```
recall = recall_score(y_test, y_pred, average='macro')
recall
```

Out[18]:
```
0.9743589743589745
```

In [19]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        19
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.93      1.00      0.96        13

       accuracy                           0.98        45
      macro avg       0.98      0.97      0.97        45
   weighted avg       0.98      0.98      0.98        45
```

In [20]:

```
cv_scores = cross_val_score(nb, X, y, cv=5)
cv_scores
```

Out[20]:

```
array([0.93333333, 0.96666667, 0.93333333, 0.93333333, 1.        ])
```

In [41]:

```
nb.predict_proba(X_test[:5])
```

Out[41]:

```
array([[4.15880005e-088, 9.95527834e-001, 4.47216606e-003],
       [1.00000000e+000, 1.31031235e-013, 2.21772205e-020],
       [9.83170191e-285, 2.70138564e-012, 1.00000000e+000],
       [9.54745274e-092, 9.74861431e-001, 2.51385686e-002],
       [1.08679560e-103, 8.31910700e-001, 1.68089300e-001]])
```

In [43]:

```
data = pd.read_csv("spam.csv", encoding='latin-1')
data.head()
```

Out[43]:

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

In [47]:

```
data.info()
data['label'].value_counts()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5171 entries, 0 to 5170
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  5171 non-null   int64 
 1   label       5171 non-null   object
 2   text        5171 non-null   object
 3   label_num   5171 non-null   int64 
dtypes: int64(2), object(2)
memory usage: 161.7+ KB
```

Out[47]:

```
label
ham     3672
spam    1499
Name: count, dtype: int64
```

In [49]:

```
X = data['text']
y = data['label_num']
```

In [51]:

```
vectorizer = TfidfVectorizer(stop_words='english')

X = vectorizer.fit_transform(X)
```

In [53]:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
```

In [55]:

```
nb = MultinomialNB()
nb.fit(X_train, y_train)
```

Out[55]:

▾ MultinomialNB i ?

```
MultinomialNB()
```

In [59]:

```
y_pred = nb.predict(X_test)
y_pred
```

Out[59]:

```
array([0, 1, 0, ..., 0, 0, 0])
```

In [61]:

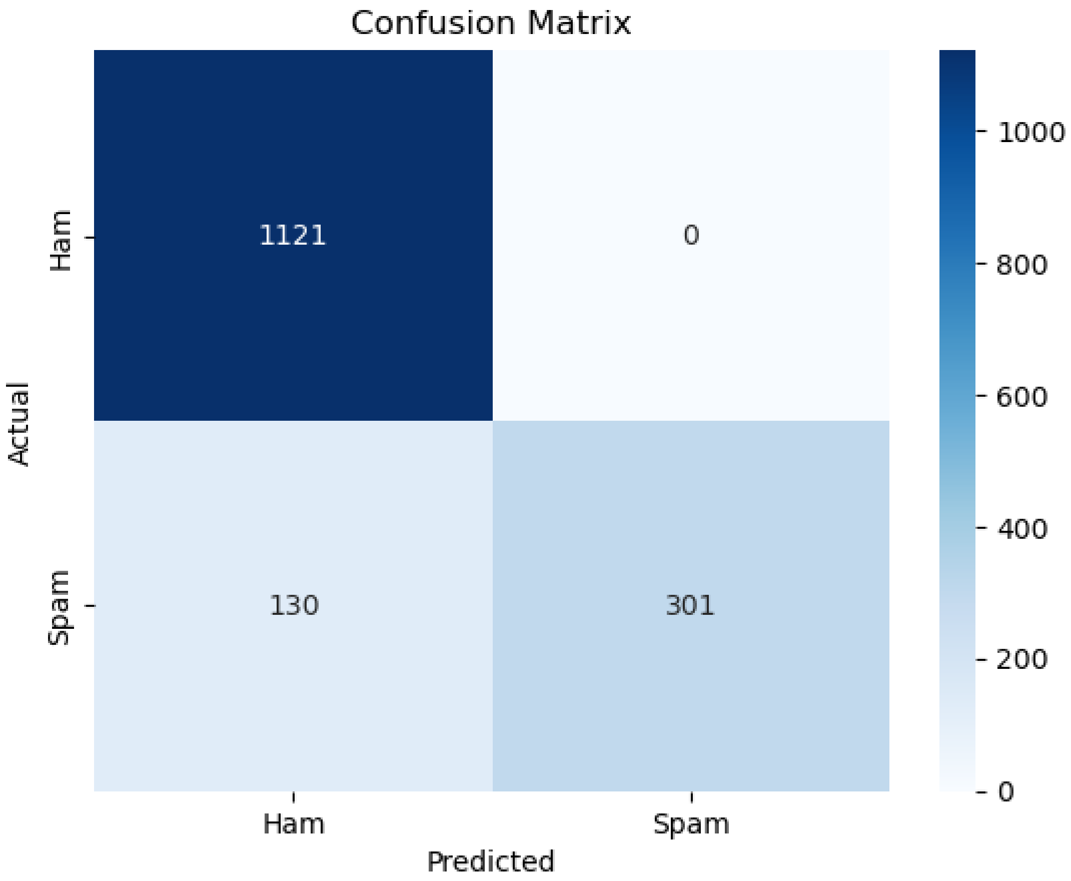
```
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[61]:

```
array([[1121,    0],
       [ 130,  301]])
```

In [63]:

```
sns.heatmap(cm, annot=True, cmap='Blues', fmt='d',
            xticklabels=['Ham', 'Spam'],
            yticklabels=['Ham', 'Spam'])

plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



In [65]:

```
TP = cm[1,1]
FP = cm[0,1]
FN = cm[1,0]
TN = cm[0,0]

print("TP:", TP)
print("FP:", FP)
print("FN:", FN)
print("TN:", TN)
```

```
TP: 301
FP: 0
FN: 130
TN: 1121
```

In [67]:

```
accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy
```

```
print("Accuracy:", accuracy)
print("Error Rate:", error_rate)
```

```
Accuracy: 0.9162371134020618
Error Rate: 0.08376288659793818
```

In [69]:
```
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("Precision:", precision)
print("Recall:", recall)
```

```
Precision: 1.0
Recall: 0.6983758700696056
```

In [71]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      1121
           1       1.00      0.70      0.82       431

    accuracy                           0.92      1552
   macro avg       0.95      0.85      0.88      1552
weighted avg       0.92      0.92      0.91      1552
```

In [73]:
```
cv_scores = cross_val_score(nb, X, y, cv=5)

print("Cross-validation scores:", cv_scores)
print("Mean CV accuracy:", cv_scores.mean())
```

```
Cross-validation scores: [0.91111111 0.93230174 0.92843327 0.92166344 0.91489362]
Mean CV accuracy: 0.9216806361487212
```

In [75]:
```
nb.predict_proba(X_test[:5])
```

Out[75]:
```
array([[9.99790505e-01, 2.09494966e-04],
       [2.20075521e-01, 7.79924479e-01],
       [9.99998514e-01, 1.48588305e-06],
       [9.99999886e-01, 1.14285146e-07],
       [9.99999879e-01, 1.21435114e-07]])
```

In [ ]: