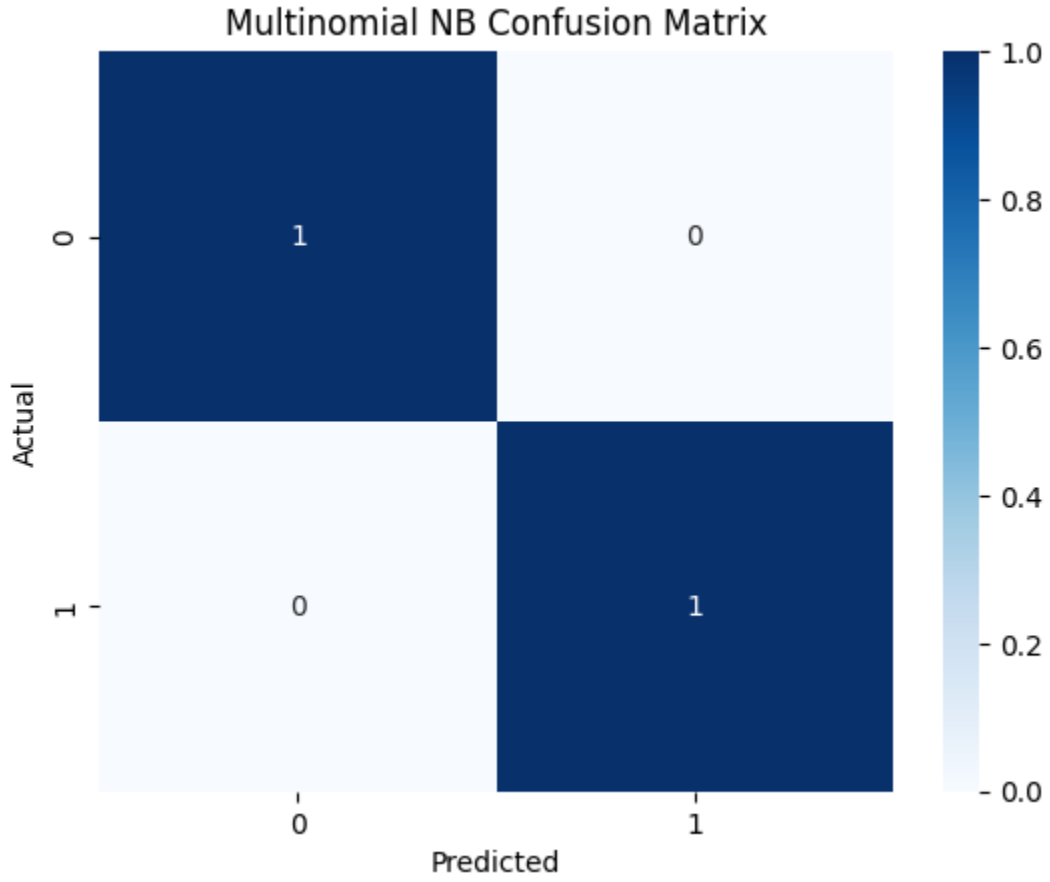
In [3]:

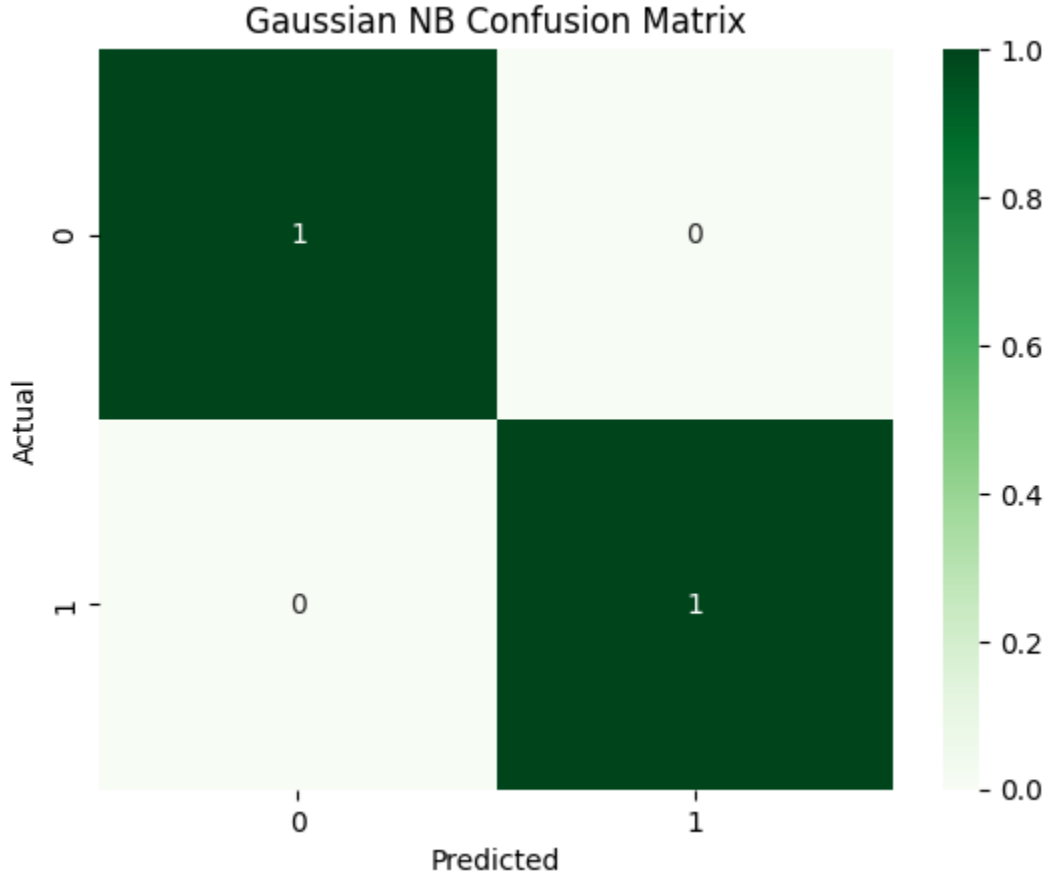
```
# Spam Email Detection using Naive Bayes

# Import libraries
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB, GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, con
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load CSV data
df = pd.read_csv('spam_emails.csv')

# Inspect data
print(df.head())

# Feature extraction using TF-IDF
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(df['Email'])
y = df['Label']

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# --------------------------------
# Multinomial Naive Bayes (for text)
# --------------------------------
mnb = MultinomialNB()
mnb.fit(X_train, y_train)
y_pred_mnb = mnb.predict(X_test)

# Evaluation
print("----- Multinomial Naive Bayes -----")
print("Accuracy:", accuracy_score(y_test, y_pred_mnb))
print("Precision:", precision_score(y_test, y_pred_mnb))
print("Recall:", recall_score(y_test, y_pred_mnb))
print("F1 Score:", f1_score(y_test, y_pred_mnb))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred_mnb)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Multinomial NB Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# --------------------------------
# Gaussian Naive Bayes (requires dense array)
# --------------------------------
gnb = GaussianNB()
# Convert sparse matrix to dense
X_train_dense = X_train.toarray()
X_test_dense = X_test.toarray()

gnb.fit(X_train_dense, y_train)
```

```python
y_pred_gnb = gnb.predict(X_test_dense)

# Evaluation
print("----- Gaussian Naive Bayes -----")
print("Accuracy:", accuracy_score(y_test, y_pred_gnb))
print("Precision:", precision_score(y_test, y_pred_gnb))
print("Recall:", recall_score(y_test, y_pred_gnb))
print("F1 Score:", f1_score(y_test, y_pred_gnb))

# Confusion matrix
cm_gnb = confusion_matrix(y_test, y_pred_gnb)
sns.heatmap(cm_gnb, annot=True, fmt='d', cmap='Greens')
plt.title("Gaussian NB Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# ------------------------------
# Real-world testing: classify new emails
# ------------------------------
new_emails = [
    "Win a brand new car now",
    "Please review the meeting notes",
    "Urgent! Claim your free vacation"
]

X_new = vectorizer.transform(new_emails)
predictions = mnb.predict(X_new)

for email, pred in zip(new_emails, predictions):
    label = "Spam" if pred == 1 else "Not Spam"
    print(f"Email: '{email}' --> Prediction: {label}")
```

```
                                    Email  Label
0                     Win a free iPhone now      1
1                 Meeting at 10 am tomorrow      0
2  Congratulations! You have won a lottery       1
3           Please find the attached report      0
4                 Claim your free prize now      1
----- Multinomial Naive Bayes -----
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```




```
----- Gaussian Naive Bayes -----
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```



```
Email: 'Win a brand new car now' --> Prediction: Spam
Email: 'Please review the meeting notes' --> Prediction: Not Spam
Email: 'Urgent! Claim your free vacation' --> Prediction: Spam
```

In [ ]:

In [ ]: