In [1]:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing the Dataset

In [2]:

```
file = 'spambase.data'
data = pd.read_csv(file, header=None)
print(data.head())
```

```
     0     1     2    3     4     5     6     7     8     9  ...    48  \
0  0.00  0.64  0.64  0.0  0.32  0.00  0.00  0.00  0.00  0.00  ...  0.00   
1  0.21  0.28  0.50  0.0  0.14  0.28  0.21  0.07  0.00  0.94  ...  0.00   
2  0.06  0.00  0.71  0.0  1.23  0.19  0.19  0.12  0.64  0.25  ...  0.01   
3  0.00  0.00  0.00  0.0  0.63  0.00  0.31  0.63  0.31  0.63  ...  0.00   
4  0.00  0.00  0.00  0.0  0.63  0.00  0.31  0.63  0.31  0.63  ...  0.00   

      49   50     51     52     53     54   55    56  57  
0  0.000  0.0  0.778  0.000  0.000  3.756   61   278   1  
1  0.132  0.0  0.372  0.180  0.048  5.114  101  1028   1  
2  0.143  0.0  0.276  0.184  0.010  9.821  485  2259   1  
3  0.137  0.0  0.137  0.000  0.000  3.537   40   191   1  
4  0.135  0.0  0.135  0.000  0.000  3.537   40   191   1  

[5 rows x 58 columns]
```

In [3]:

```
data.columns = [
    'word_freq_make',
    'word_freq_address',
    'word_freq_all',
    'word_freq_3d',
    'word_freq_our',
    'word_freq_over',
    'word_freq_remove',
    'word_freq_internet',
    'word_freq_order',
    'word_freq_mail',
    'word_freq_receive',
    'word_freq_will',
    'word_freq_people',
    'word_freq_report',
    'word_freq_addresses',
    'word_freq_free',
    'word_freq_business',
    'word_freq_email',
    'word_freq_you',
    'word_freq_credit',
    'word_freq_your',
    'word_freq_font',
    'word_freq_000',
    'word_freq_money',
    'word_freq_hp',
    'word_freq_hpl',
    'word_freq_george',
    'word_freq_650',
    'word_freq_lab',
    'word_freq_labs',
    'word_freq_telnet',
    'word_freq_857',
    'word_freq_data',
    'word_freq_415',
    'word_freq_85',
    'word_freq_technology',
    'word_freq_1999',
    'word_freq_parts',
    'word_freq_pm',
    'word_freq_direct',
    'word_freq_cs',
    'word_freq_meeting',
    'word_freq_original',
    'word_freq_project',
    'word_freq_re',
    'word_freq_edu',
    'word_freq_table',
    'word_freq_conference',
    'char_freq_;',
    'char_freq_(',
    'char_freq_[',
    'char_freq_!',
    'char_freq_$',
    'char_freq_#',
    'capital_run_length_average',
    'capital_run_length_longest',
    'capital_run_length_total',
    'spam'
]
```

In [4]:
```
data.head()
```

Out[4]:

	word_freq_make	word_freq_address	word_freq_all	word_freq_3d	word_freq_our	word_freq_over	word_freq_remove	word_freq_internet
0	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00
1	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07
2	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63
4	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63

5 rows × 58 columns

Pre-process the dataset

In [5]:
```
duplicated_rows = data.duplicated()
sum_duplicate = sum(duplicated_rows)
print(f"Number of duplicate rows: {sum_duplicate}")
```

```
Number of duplicate rows: 391
```

In [6]:
```
data = data.drop_duplicates()
```

In [7]:
```
data.loc[(data['word_freq_george'] > 0) | (data['word_freq_650'] > 0), 'spam'] = 0
```

In [8]:
```
columns_to_display = ['word_freq_george', 'word_freq_650', 'spam']
df_subset = data.loc[:, columns_to_display]
df_subset
```

Out[8]:

	word_freq_george	word_freq_650	spam
0	0.0	0.0	1
1	0.0	0.0	1
2	0.0	0.0	1
3	0.0	0.0	1
4	0.0	0.0	1
...	...	...	...
4596	0.0	0.0	0
4597	0.0	0.0	0
4598	0.0	0.0	0
4599	0.0	0.0	0
4600	0.0	0.0	0

4210 rows × 3 columns

In [9]:

```
data.to_csv('spambase.csv', index=False)
new_data = pd.read_csv('spambase.csv', header=0)
new_data.head()
```

Out[9]:

	word_freq_make	word_freq_address	word_freq_all	word_freq_3d	word_freq_our	word_freq_over	word_freq_remove	word_freq_internet	w
0	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00	
1	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	
2	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	
4	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	

5 rows × 58 columns

Splitting the Dataset and Training the Model

In [10]:

```
X = new_data.iloc[:, 0:57]
y = new_data.iloc[:, 57]
```

In [11]:

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

Out[11]:

```
array([[-0.34792164,  1.16102457,  0.67588944, ..., -0.04911671,
         0.04439849, -0.02130997],
       [ 0.3521501 ,  0.3684328 ,  0.40439129, ..., -0.00814327,
         0.24484101,  1.1911417 ],
       [-0.14790114, -0.24802746,  0.81163851, ...,  0.13387587,
         2.16908914,  3.18117903],
       ...,
       [ 0.65218084, -0.24802746,  0.01653679, ..., -0.12008102,
        -0.23120996, -0.27996632],
       [ 2.85240628, -0.24802746, -0.56524496, ..., -0.12783519,
        -0.23622103, -0.34463041],
       [-0.34792164, -0.24802746,  0.69528216, ..., -0.12472749,
        -0.23622103, -0.40606129]])
```

In [12]:

```
pca = PCA(0.95)
X_pca = pca.fit_transform(X_scaled)
X.shape,X_pca.shape
```

Out[12]:

```
((4210, 57), (4210, 49))
```

In [13]:

```
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=42)
```

In [14]:

```
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

Out[14]:

```
▾ GaussianNB
GaussianNB()
```

In [15]:

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

Model results and Confusion Matrix

In [16]:

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.86
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.86      0.88       490
           1       0.82      0.87      0.84       352

    accuracy                           0.86       842
   macro avg       0.86      0.87      0.86       842
weighted avg       0.87      0.86      0.87       842
```

In [17]:

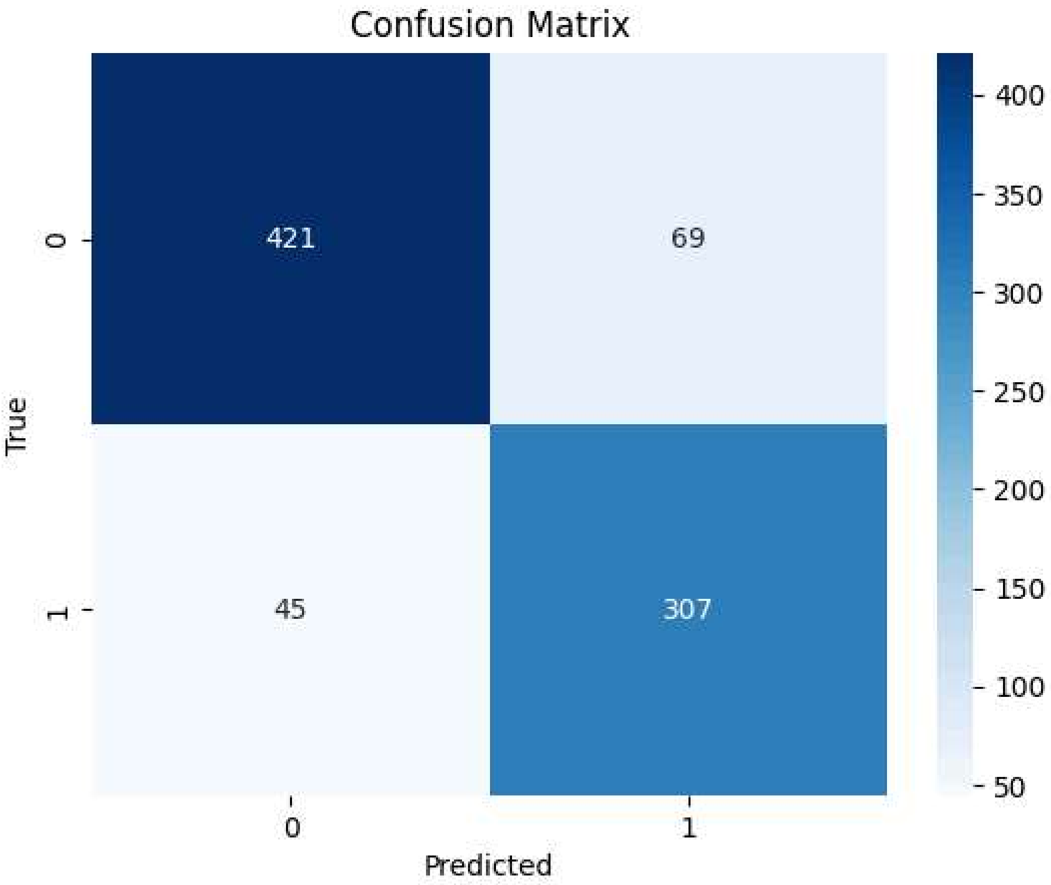
```
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[17]:

```
array([[421,  69],
       [ 45, 307]])
```

In [18]:

```
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```


Confusion Matrix
421
69
45
307
0
1
True
0
1
Predicted
400
350
300
250
200
150
100
50