In [3]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
data = pd.read_csv('D:/bharat intern/task 1/mail_data.csv')
```

In [6]:
```
data
```

Out[6]:

| | Category | Message |
|---|---|---|
| **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 [7]:
```
data = data.where(pd.notnull(data),'')
```

In [8]:
```
data.head()
```

Out[8]:

| | Category | Message |
|---|---|---|
| **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... |

In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  5572 non-null   object
 1   Message   5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

In [11]:
```
data.shape
```

Out[11]:
```
(5572, 2)
```

In [14]:
```
data.loc[data['Category'] == 'spam', 'Category',] = 0
data.loc[data['Category'] == 'ham', 'Category',] = 1
```

In [15]:
```
X = data['Message']
Y = data['Category']
```

In [16]:
```
X
```

Out[16]:
```
0       Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
3       U dun say so early hor... U c already then say...
4       Nah I don't think he goes to usf, he lives aro...
                              ...
5567    This is the 2nd time we have tried 2 contact u...
5568                 Will ü b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                           Rofl. Its true to its name
Name: Message, Length: 5572, dtype: object
```

In [17]:
```
Y
```

Out[17]:
```
0       1
1       1
2       0
3       1
4       1
       ..
5567    0
5568    1
5569    1
5570    1
5571    1
Name: Category, Length: 5572, dtype: object
```

In [20]:
```
X_train, X_test, Y_train, Y_test= train_test_split(X,Y, train_size=0.2, random_stat
```

In [27]:
```
print(X.shape)
print(X_train.shape)
print(X_test.shape)
```

```
(5572,)
(1114,)
(4458,)
```

In [28]:
```
print(Y.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(5572,)
(1114,)
(4458,)
```

In [38]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
feature_extraction = TfidfVectorizer(min_df=1, stop_words='english', lowercase=True
X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)

Y_train = Y_train.astype('int')
Y_test = Y_test.astype('int')
```

In [37]:
```
print(X_train)
```

```
2309    Moby Pub Quiz.Win a £100 High Street prize if ...
3727                        No chikku nt yet.. Ya i'm free
385     Double mins and txts 4 6months FREE Bluetooth ...
2300    Congrats! 1 year special cinema pass for 2 is ...
4857         yes baby! I need to stretch open your pussy!
                              ...
789     5 Free Top Polyphonic Tones call 087018728737,...
968     What do u want when i come back?.a beautiful n...
1667    Guess who spent all last night phasing in and ...
3321    Eh sorry leh... I din c ur msg. Not sad alread...
1688    Free Top ringtone -sub to weekly ringtone-get ...
Name: Message, Length: 1114, dtype: object
```

In [40]:
```
print(X_train_features)
```

```
<Compressed Sparse Row sparse matrix of dtype 'float64'
        with 8568 stored elements and shape (1114, 3259)>
  Coords        Values
  (0, 1916)     0.261765485903701
  (0, 2275)     0.2290141352210295
  (0, 2296)     0.23699004553796602
  (0, 3158)     0.18974599130248632
  (0, 72)       0.21221460517223104
  (0, 1425)     0.24727278235089278
  (0, 2722)     0.23699004553796602
  (0, 2253)     0.19772190161942282
  (0, 1641)     0.1446291746277319
  (0, 1997)     0.1626637244407611
  (0, 1013)     0.261765485903701
  (0, 799)      0.261765485903701
  (0, 2963)     0.16049679403314557
  (0, 233)      0.24727278235089278
  (0, 3001)     0.23699004553796602
  (0, 2714)     0.15946068076413877
  (0, 186)      0.18527223439888055
  (0, 1)        0.261765485903701
  (0, 2658)     0.24727278235089278
  (0, 396)      0.261765485903701
  (1, 699)      0.5764566559616184
  (1, 2034)     0.5161926584979777
  (1, 3226)     0.49679198460421103
  (1, 1216)     0.3929892964742779
  (2, 1216)     0.17159758459319638
  :     :
  (1111, 1212)  0.4509504391514392
  (1111, 941)   0.4509504391514392
  (1112, 3009)  0.19840634450024627
  (1112, 2456)  0.34374499989110935
  (1112, 1950)  0.2508522295519489
  (1112, 3103)  0.2826382889934465
  (1112, 2647)  0.21942745688441115
  (1112, 2960)  0.2826382889934465
  (1112, 1665)  0.34374499989110935
  (1112, 1043)  0.33217624693018544
  (1112, 2054)  0.3227238875069444
  (1112, 942)   0.3586597164844422
  (1112, 1692)  0.33217624693018544
  (1113, 2714)  0.17105479399736778
  (1113, 1216)  0.34662174193772693
  (1113, 42)    0.2542211862175887
  (1113, 2516)  0.16057611350832152
  (1113, 3116)  0.39310295904162784
  (1113, 2624)  0.2090289055002543
  (1113, 3119)  0.24566536110968612
  (1113, 113)   0.2327642595844968
  (1113, 2414)  0.4773494859780158
  (1113, 2743)  0.2652515632286211
  (1113, 232)   0.2652515632286211
  (1113, 2741)  0.28079800645720193
```

In [41]:
```
model = LogisticRegression()
```

In [42]:
```
model.fit(X_train_features,Y_train)
```

Out[42]:
```
LogisticRegression()
```

In [46]:
```
prediction_on_training_data = model.predict(X_train_features)
accuracy_on_traning_data = accuracy_score(Y_train,prediction_on_training_data)
```

In [48]:
```
print('Accuracy on training data :', accuracy_on_traning_data)
```

```
Accuracy on training data : 0.9165170556552962
```

In [51]:
```
prediction_on_test_data = model.predict(X_test_features)
accuracy_on_test_data = accuracy_score(Y_test,prediction_on_test_data)
```

In [52]:
```
print('Accuracy on test data :', accuracy_on_test_data)
```

```
Accuracy on test data : 0.9039928218932256
```

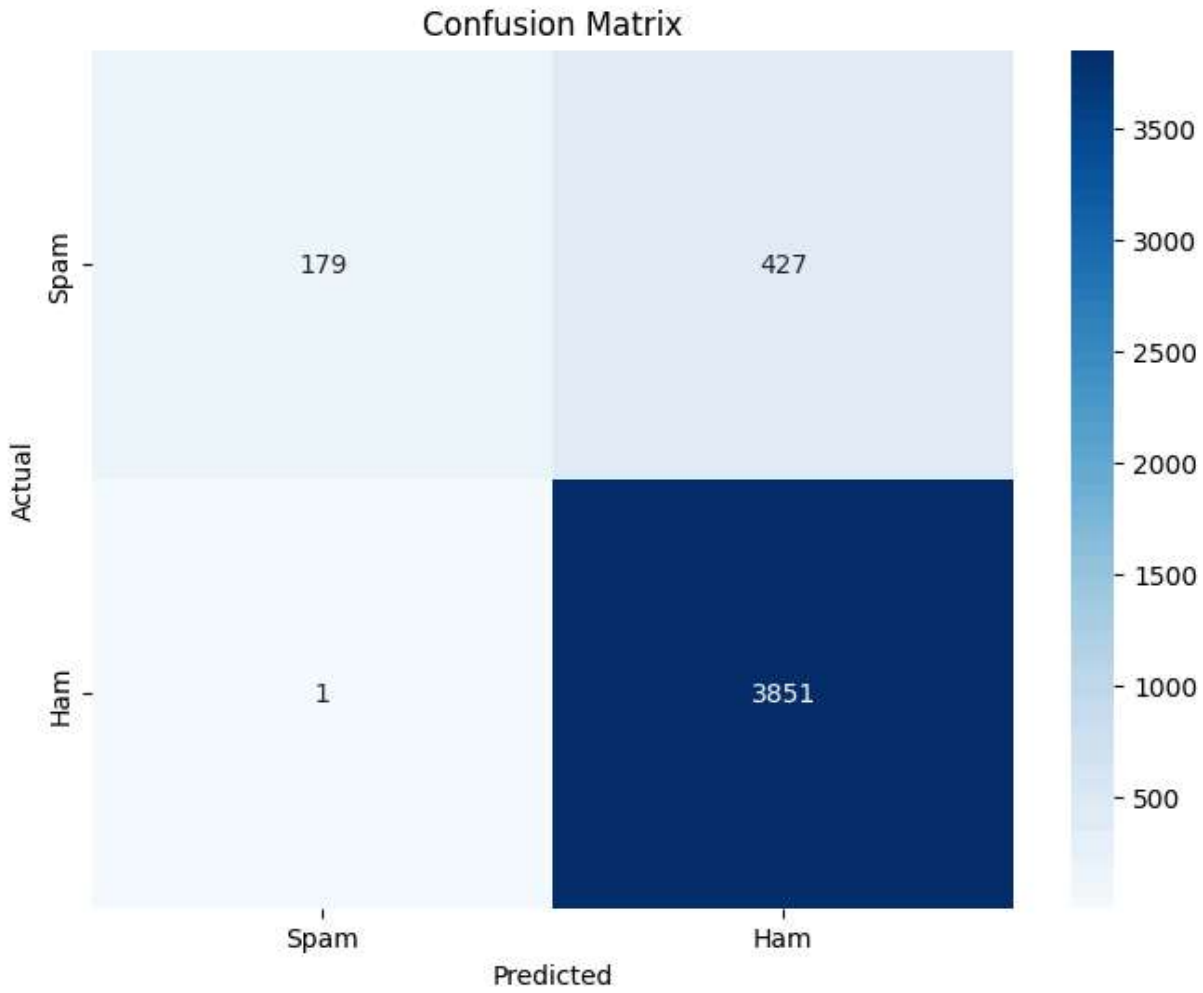
In [66]:
```
input_your_mail = ["I'm gonna be home soon and i don't want to talk about this stuf
input_data_features = feature_extraction.transform(input_your_mail)
prediction = model.predict(input_data_features)
print(prediction)

if(prediction[0]==1):
    print('ham mail')
else: print('spam mail')
```

```
[1]
ham mail
```

In [67]:
```
print('Confusion Matrix:\n', confusion_matrix(Y_test, prediction_on_test_data))
print('Precision:', precision_score(Y_test, prediction_on_test_data))
print('Recall:', recall_score(Y_test, prediction_on_test_data))
print('F1 Score:', f1_score(Y_test, prediction_on_test_data))
```

```
Confusion Matrix:
 [[ 179  427]
 [   1 3851]]
Precision: 0.9001870032725573
Recall: 0.9997403946002077
F1 Score: 0.9473554735547356
```

In [68]:
```
cm = confusion_matrix(Y_test, prediction_on_test_data)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Spam', 'Ham'], yti
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```




In [78]:
```
from sklearn.metrics import classification_report
print(classification_report(Y_test, prediction_on_test_data, target_names=['Spam',
```

```
              precision    recall  f1-score   support

        Spam       0.99      0.30      0.46       606
         Ham       0.90      1.00      0.95      3852

    accuracy                           0.90      4458
   macro avg       0.95      0.65      0.70      4458
weighted avg       0.91      0.90      0.88      4458
```

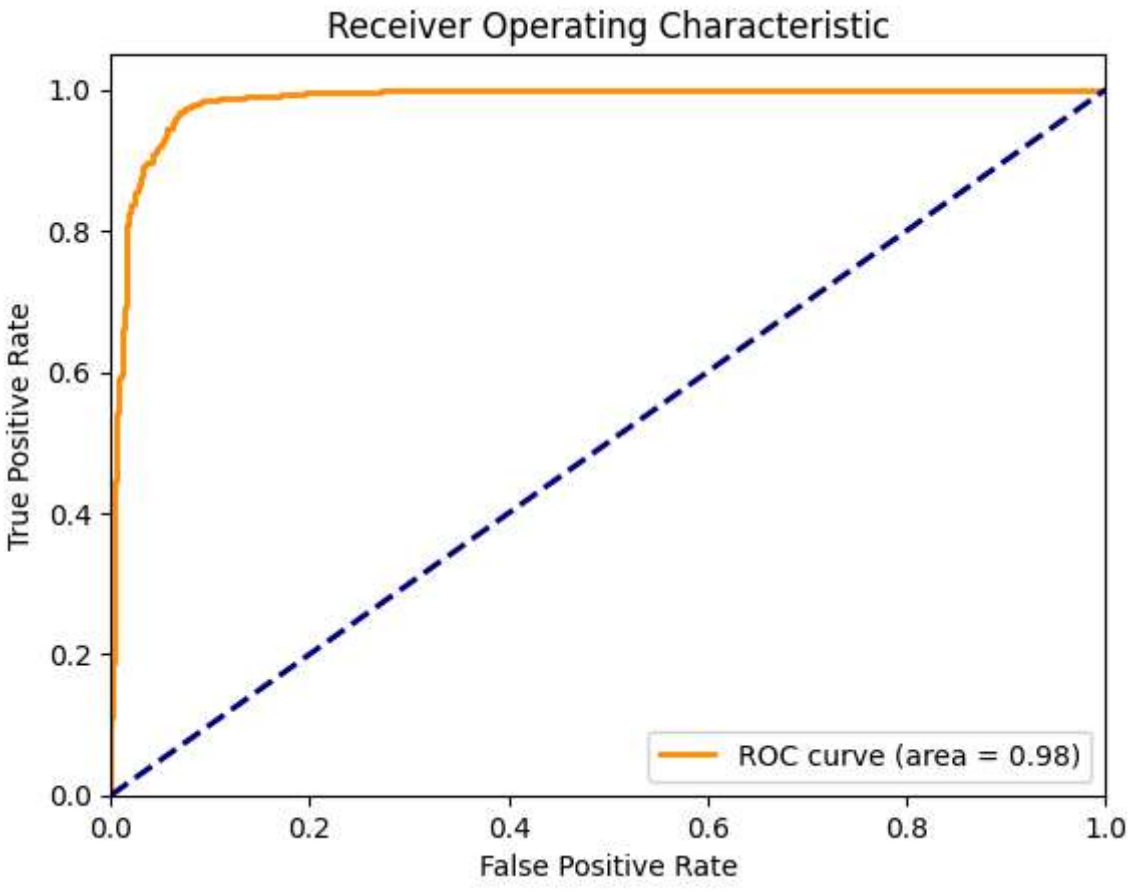
In [82]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
X_train_features = vectorizer.fit_transform(X_train)
X_test_features = vectorizer.transform(X_test)
```

In [83]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train_features, Y_train)
```

Out[83]:
```
LogisticRegression
LogisticRegression()
```

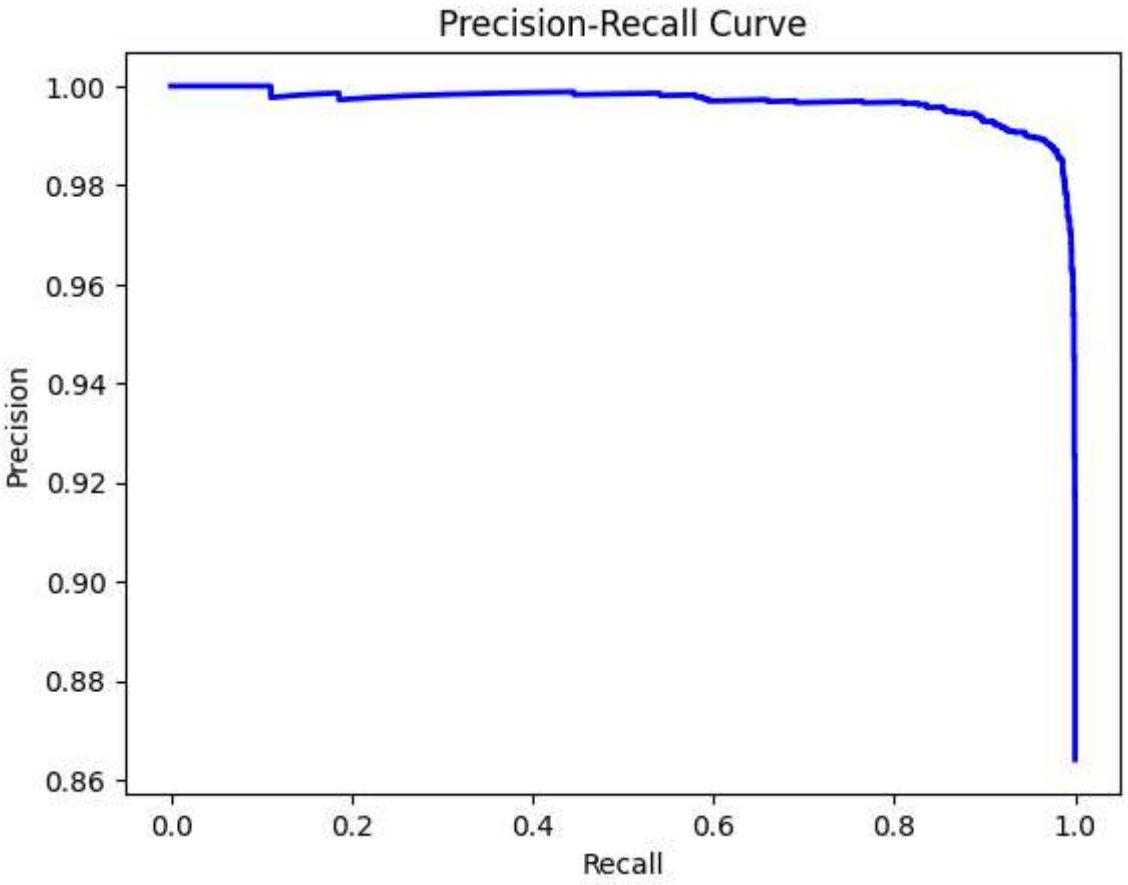
In [84]:
```
y_probs = model.predict_proba(X_test_features)[:, 1]
```

In [85]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
fpr, tpr, _ = roc_curve(Y_test, y_probs)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.show()
```




In [86]:
```
from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt
precision, recall, _ = precision_recall_curve(Y_test, y_probs)
plt.figure()
plt.plot(recall, precision, color='blue', lw=2)
```

```
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.show()
```




In [81]:
```
from wordcloud import WordCloud
spam_messages = data[data['Category'] == 0]['Message']
ham_messages = data[data['Category'] == 1]['Message']
spam_wordcloud = WordCloud(width=800, height=400, background_color='white').generat
ham_wordcloud = WordCloud(width=800, height=400, background_color='white').generate
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
plt.imshow(spam_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Spam Messages')
plt.subplot(1, 2, 2)
plt.imshow(ham_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Ham Messages')
plt.show()
```



In [96]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
param_grid = {
    'C': [0.1, 1, 10],
    'max_iter': [100, 200, 300]
}
grid_search = GridSearchCV(LogisticRegression(solver='liblinear'), param_grid, cv=5
grid_search.fit(X_train_features, Y_train)
print(f'Best Parameters: {grid_search.best_params_}')
print(f'Best Score: {grid_search.best_score_}')
```

```
Best Parameters: {'C': 10, 'max_iter': 100}
Best Score: 0.9721649901022099
```

In [97]:
```
from sklearn.linear_model import LogisticRegression
final_model = LogisticRegression(C=10, max_iter=300, solver='liblinear')
final_model.fit(X_train_features, Y_train)
from sklearn.metrics import accuracy_score
Y_pred = final_model.predict(X_test_features)
accuracy = accuracy_score(Y_test, Y_pred)
print(f'Final Model Accuracy: {accuracy}')
```

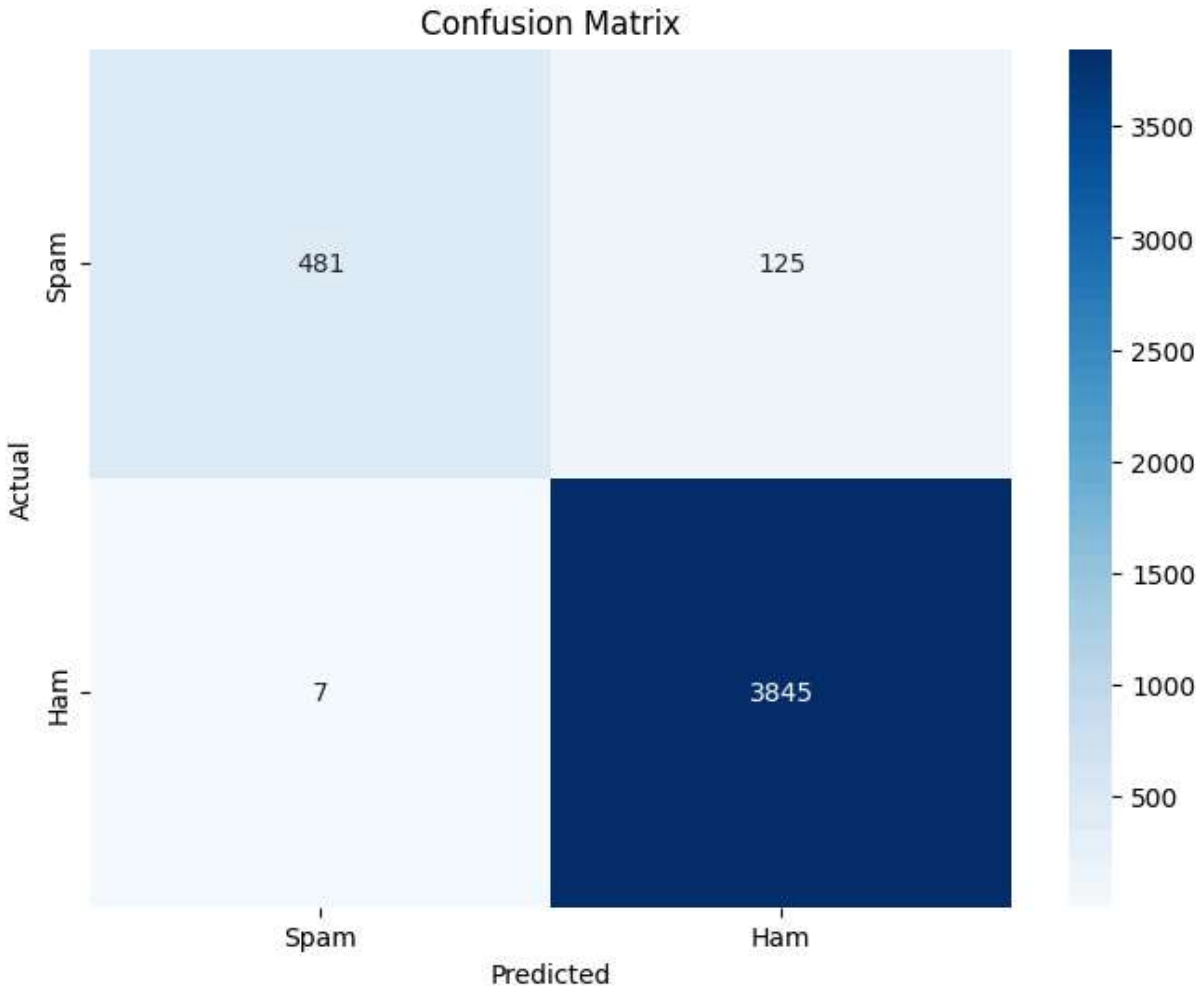
```
Final Model Accuracy: 0.9703903095558546
```

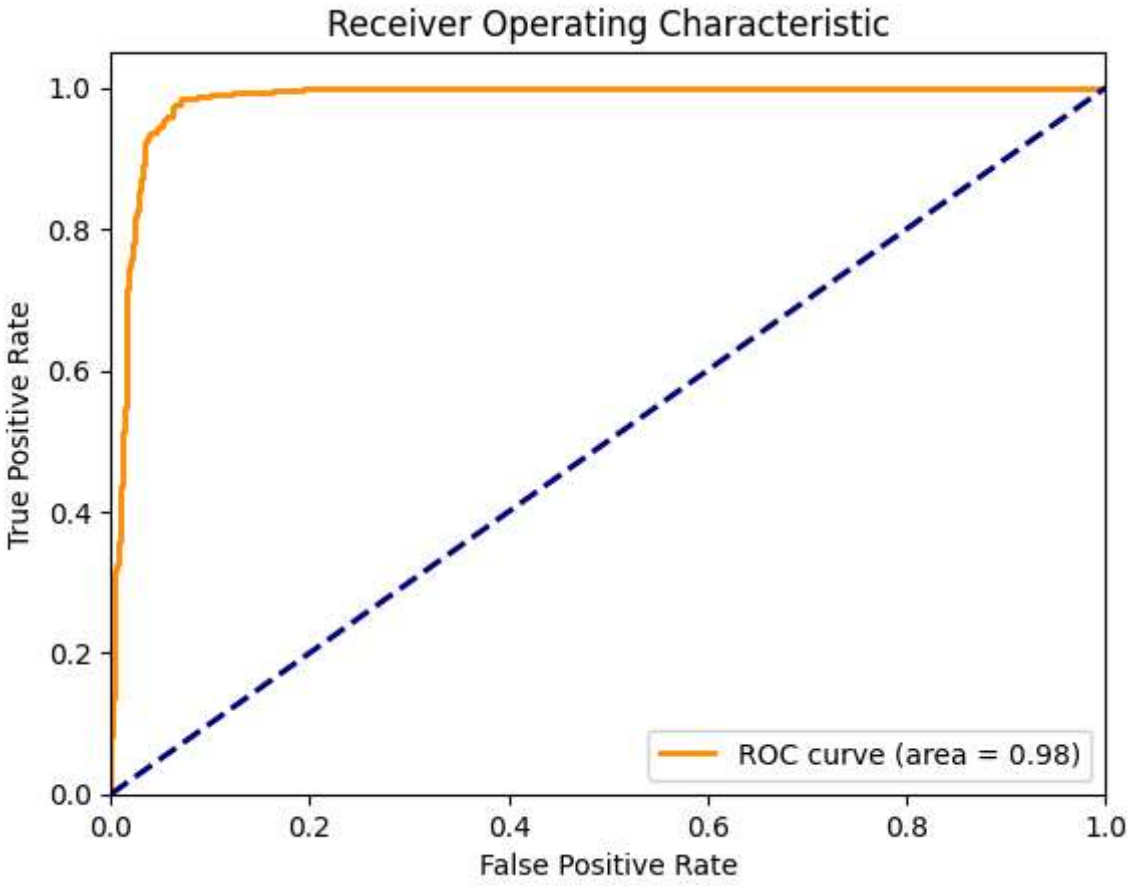
In [91]:
```
best_model = grid_search.best_estimator_
prediction_on_test_data = best_model.predict(X_test_features)
accuracy_on_test_data = accuracy_score(Y_test, prediction_on_test_data)
print('Accuracy on test data:', accuracy_on_test_data)
```

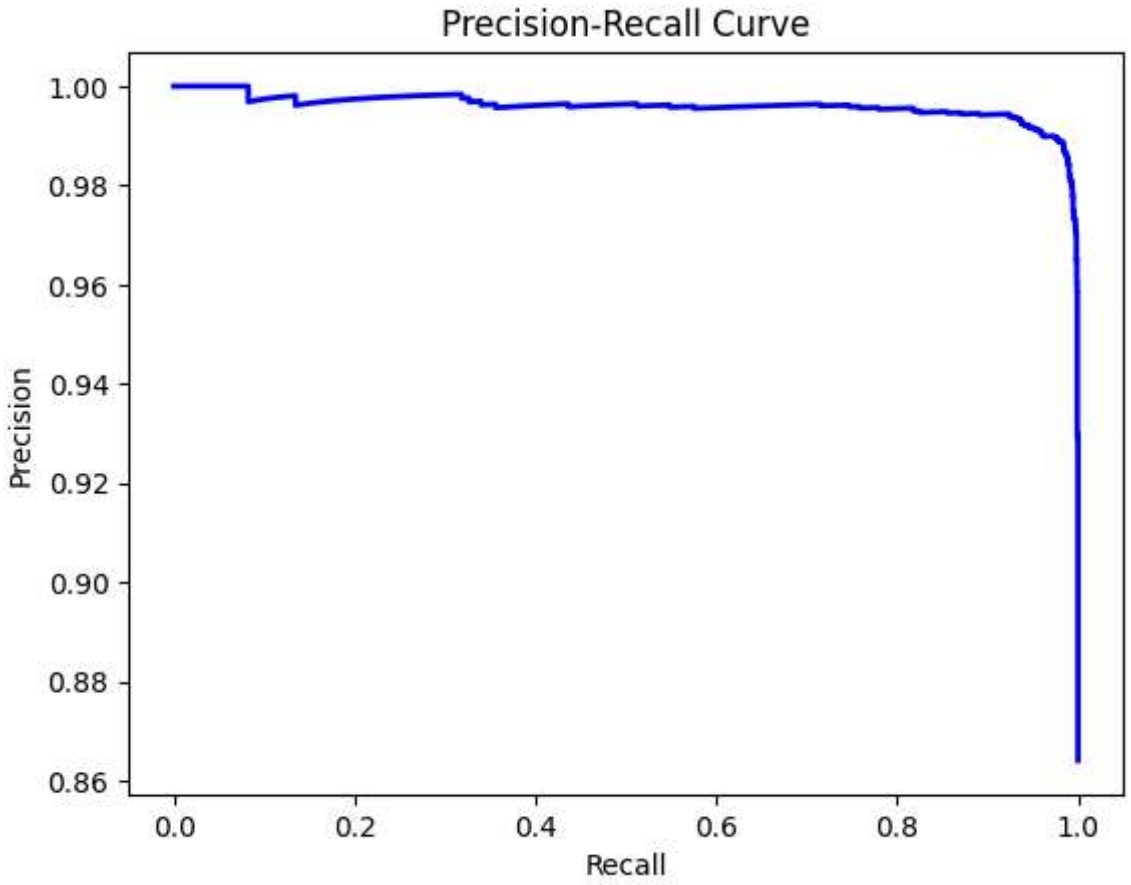
```
Accuracy on test data: 0.9703903095558546
```

In [92]:
```
from sklearn.metrics import confusion_matrix, roc_curve, auc, precision_recall_curv
import matplotlib.pyplot as plt
import seaborn as sns
cm = confusion_matrix(Y_test, prediction_on_test_data)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Spam', 'Ham'], yti
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
y_probs = best_model.predict_proba(X_test_features)[:, 1]
fpr, tpr, _ = roc_curve(Y_test, y_probs)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
```

```
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.show()
precision, recall, _ = precision_recall_curve(Y_test, y_probs)
plt.figure()
plt.plot(recall, precision, color='blue', lw=2)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.show()
```




Receiver Operating Characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.98)
0.0
0.2
0.4
0.6
0.8
1.0


Precision-Recall Curve
Precision
Recall
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.0
0.2
0.4
0.6
0.8
1.0

In [98]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
rf_model = RandomForestClassifier(n_estimators=100, random_state=3)
rf_model.fit(X_train_features, Y_train)
rf_predictions = rf_model.predict(X_test_features)
print('Random Forest Accuracy:', accuracy_score(Y_test, rf_predictions))
print('Classification Report:\n', classification_report(Y_test, rf_predictions, tar
```

```
Random Forest Accuracy: 0.9499775684163302
Classification Report:
              precision    recall  f1-score   support

        Spam       0.99      0.64      0.78       606
         Ham       0.95      1.00      0.97      3852

    accuracy                           0.95      4458
   macro avg       0.97      0.82      0.87      4458
weighted avg       0.95      0.95      0.95      4458
```

In [99]:
```
from sklearn.svm import SVC
svm_model = SVC(kernel='linear', probability=True, random_state=3)
svm_model.fit(X_train_features, Y_train)
svm_predictions = svm_model.predict(X_test_features)
print('SVM Accuracy:', accuracy_score(Y_test, svm_predictions))
print('Classification Report:\n', classification_report(Y_test, svm_predictions, ta
```

```
SVM Accuracy: 0.9708389412292507
Classification Report:
              precision    recall  f1-score   support

        Spam       0.99      0.80      0.88       606
         Ham       0.97      1.00      0.98      3852

    accuracy                           0.97      4458
   macro avg       0.98      0.90      0.93      4458
weighted avg       0.97      0.97      0.97      4458
```

In [100…
```
from sklearn.ensemble import GradientBoostingClassifier
gb_model = GradientBoostingClassifier(n_estimators=100, random_state=3)
gb_model.fit(X_train_features, Y_train)
gb_predictions = gb_model.predict(X_test_features)
print('Gradient Boosting Accuracy:', accuracy_score(Y_test, gb_predictions))
print('Classification Report:\n', classification_report(Y_test, gb_predictions, tar
```

```
Gradient Boosting Accuracy: 0.9537909376401974
Classification Report:
              precision    recall  f1-score   support

        Spam       0.96      0.69      0.80       606
         Ham       0.95      1.00      0.97      3852

    accuracy                           0.95      4458
   macro avg       0.96      0.84      0.89      4458
weighted avg       0.95      0.95      0.95      4458
```

In [101…

```
from sklearn.naive_bayes import MultinomialNB
nb_model = MultinomialNB()
nb_model.fit(X_train_features, Y_train)
nb_predictions = nb_model.predict(X_test_features)
print('Naive Bayes Accuracy:', accuracy_score(Y_test, nb_predictions))
print('Classification Report:\n', classification_report(Y_test, nb_predictions, tar
```

```
Naive Bayes Accuracy: 0.9804845222072678
Classification Report:
              precision    recall  f1-score   support

        Spam       0.97      0.88      0.92       606
         Ham       0.98      1.00      0.99      3852

    accuracy                           0.98      4458
   macro avg       0.98      0.94      0.96      4458
weighted avg       0.98      0.98      0.98      4458
```

In [102…

```
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train_features, Y_train)
knn_predictions = knn_model.predict(X_test_features)
print('KNN Accuracy:', accuracy_score(Y_test, knn_predictions))
print('Classification Report:\n', classification_report(Y_test, knn_predictions, ta
```

```
KNN Accuracy: 0.8712427097353073
Classification Report:
              precision    recall  f1-score   support

        Spam       1.00      0.05      0.10       606
         Ham       0.87      1.00      0.93      3852

    accuracy                           0.87      4458
   macro avg       0.94      0.53      0.52      4458
weighted avg       0.89      0.87      0.82      4458
```

In [104…

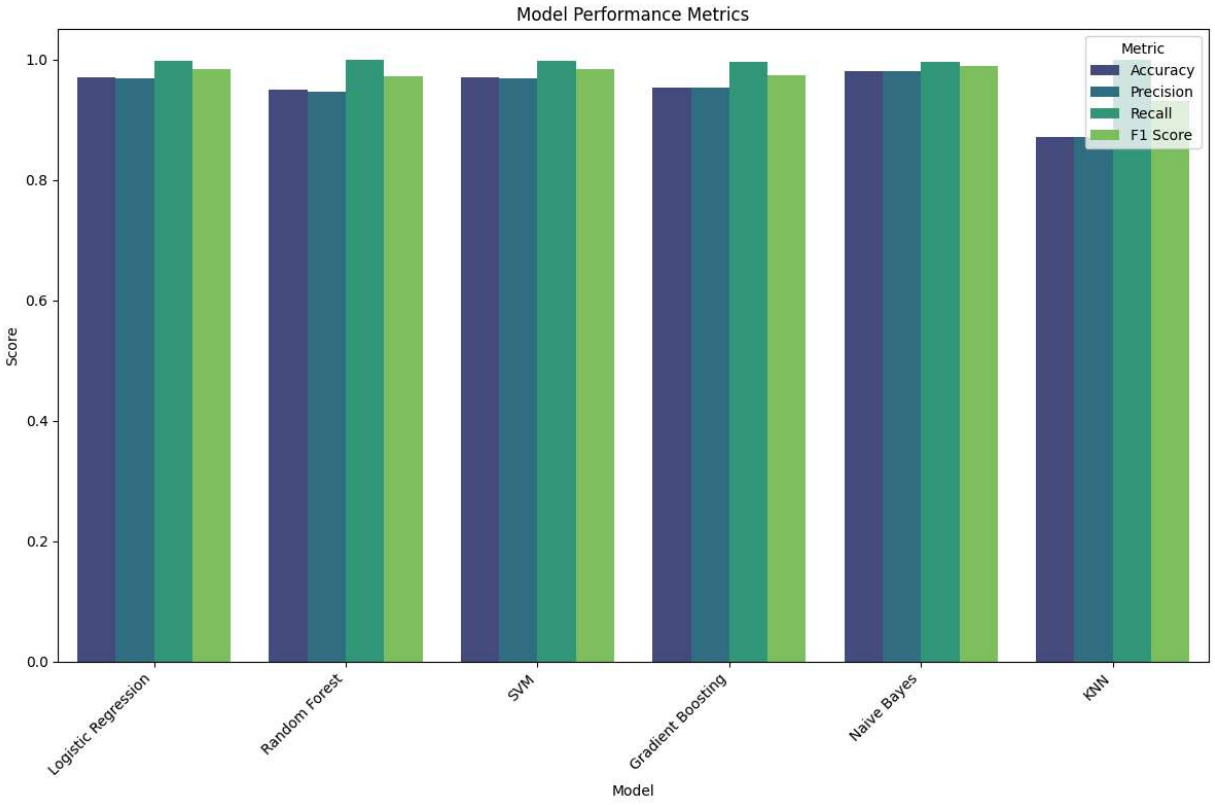
```
import pandas as pd
results = pd.DataFrame({
    'Model': ['Logistic Regression', 'Random Forest', 'SVM', 'Gradient Boosting', '
    'Accuracy': [accuracy_on_test_data, accuracy_score(Y_test, rf_predictions), acc
    'Precision': [precision_score(Y_test, prediction_on_test_data), precision_score
    'Recall': [recall_score(Y_test, prediction_on_test_data), recall_score(Y_test,
    'F1 Score': [f1_score(Y_test, prediction_on_test_data), f1_score(Y_test, rf_pre
```

```
})

print(results)
```

```
                Model  Accuracy  Precision    Recall  F1 Score
0  Logistic Regression  0.970390   0.968514  0.998183  0.983125
1        Random Forest  0.949978   0.945714  0.999481  0.971854
2                  SVM  0.970839   0.968766  0.998442  0.983380
3    Gradient Boosting  0.953791   0.953032  0.995587  0.973845
4          Naive Bayes  0.980485   0.981335  0.996366  0.988793
5                  KNN  0.871243   0.870312  1.000000  0.930660
```

In [106…

```
results = pd.DataFrame({
    'Model': ['Logistic Regression', 'Random Forest', 'SVM', 'Gradient Boosting', '
    'Accuracy': [0.9704, 0.9500, 0.9708, 0.9538, 0.9805, 0.8712],
    'Precision': [0.9685, 0.9457, 0.9688, 0.9530, 0.9813, 0.8703],
    'Recall': [0.9982, 0.9995, 0.9984, 0.9956, 0.9964, 1.0000],
    'F1 Score': [0.9831, 0.9719, 0.9834, 0.9738, 0.9888, 0.9307]
})
results_melted = results.melt(id_vars='Model', var_name='Metric', value_name='Score
plt.figure(figsize=(12, 8))
sns.barplot(data=results_melted, x='Model', y='Score', hue='Metric', palette='virid
plt.title('Model Performance Metrics')
plt.xlabel('Model')
plt.ylabel('Score')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Metric')
plt.tight_layout()
plt.show()
```




In [ ]: