Classify the email using the binary classification method. Email Spam detection has two states: a) Normal State – Not Spam, b) Abnormal State – Spam. Use Support Vector Machine for classification. Analyze their performance. Dataset link: The emails.csv dataset on the Kaggle https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv

In [71]:
```
#Step1
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, roc_curve, auc, precision_recall_curve
```

In [72]:
```
#Step2: Load the dataset
df = pd.read_csv("C:/Users/samik/Downloads/archive (9)/emails.csv")

# Display the first few rows of the dataset to understand its structure
print(df.head())
```

```
  Email No.  the  to  ect  and  for  of    a  you  hou  ...  connevey  jay  \
0   Email 1    0   0    1    0    0   0    2    0    0  ...         0    0
1   Email 2    8  13   24    6    6   2  102    1   27  ...         0    0
2   Email 3    0   0    1    0    0   0    8    0    0  ...         0    0
3   Email 4    0   5   22    0    5   1   51    2   10  ...         0    0
4   Email 5    7   6   17    1    5   2   57    0    9  ...         0    0

   valued  lay  infrastructure  military  allowing  ff  dry  Prediction
0       0    0               0         0         0   0    0           0
1       0    0               0         0         0   1    0           0
2       0    0               0         0         0   0    0           0
3       0    0               0         0         0   0    0           0
4       0    0               0         0         0   1    0           0

[5 rows x 3002 columns]
```

In [73]:
```
#Step 3: Data Preprocessing
#Drop the 'Email' column
df = df.drop(columns=['Email'], errors='ignore')

# Set up features and target variable
X = df.drop(columns=['Prediction'])
y = df['Prediction']

# Convert feature data to numeric, handling any unexpected non-numeric values
X = X.apply(pd.to_numeric, errors='coerce').fillna(0)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [74]:
```
#Step 4: Train the Support Vector Machine Model

# Initialize and train the SVM model
svm = SVC(kernel='linear', C=1.0)
svm.fit(X_train, y_train)
```

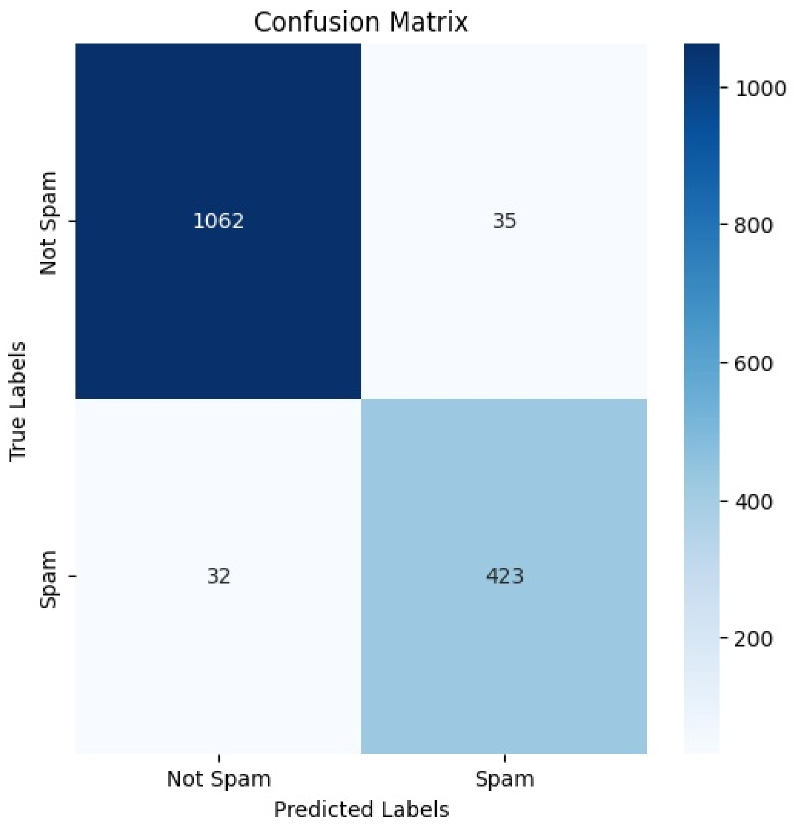
Out[74]:
```
▾ SVC
SVC(kernel='linear')
```

In [75]:
```
#Step 5: Predict and evaluate performance
y_pred = svm.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9568298969072165
Confusion Matrix:
 [[1062   35]
 [  32  423]]
Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.97      0.97      1097
           1       0.92      0.93      0.93       455

    accuracy                           0.96      1552
   macro avg       0.95      0.95      0.95      1552
weighted avg       0.96      0.96      0.96      1552
```

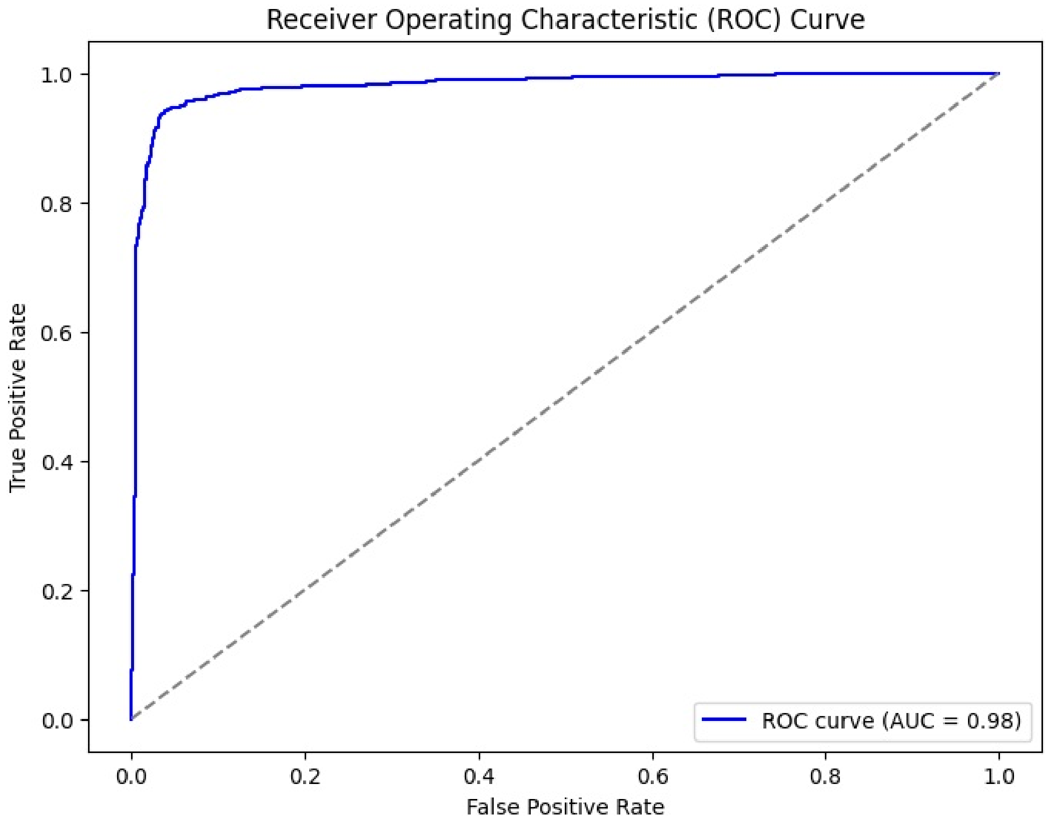
Confusion Matrix Heatmap - Displays counts of true positives, false positives, true negatives, and false negatives, which help you understand misclassifications.

In [76]:
```
#Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=['Not Spam', 'Spam'], yticklabels=['Not
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```



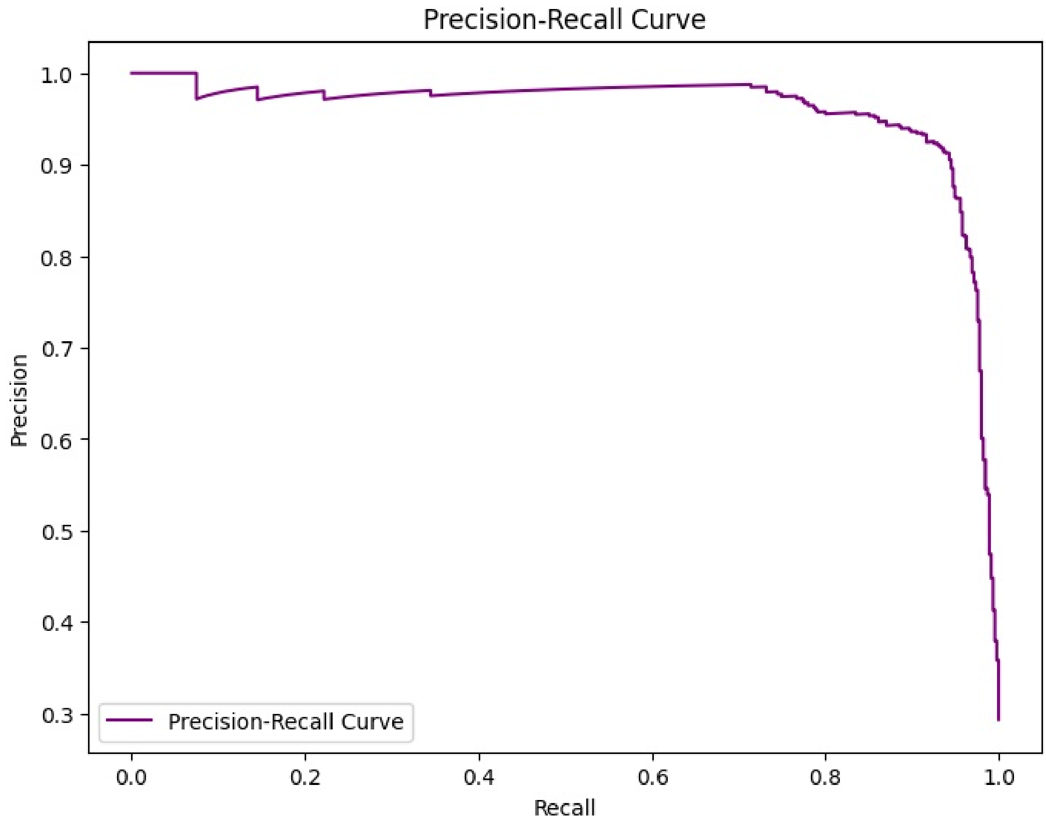
ROC Curve - Plots the true positive rate (TPR) against the false positive rate (FPR), showing how well the classifier distinguishes between classes. The Area Under the Curve (AUC) summarizes the classifier's ability to differentiate between spam and non-spam.

In [77]:
```
# ROC Curve
y_pred_prob = svm.decision_function(X_test)  # Get decision function scores for ROC
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color="blue", label=f"ROC curve (AUC = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], color="gray", linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic (ROC) Curve")
plt.legend(loc="lower right")
plt.show()
```



Precision-Recall Curve- Particularly useful for imbalanced datasets, it illustrates the trade-off between precision and recall. A high area under the precision-recall curve indicates good classification performance.

In [78]:
```
# Precision-Recall Curve
precision, recall, _ = precision_recall_curve(y_test, y_pred_prob)

plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color="purple", label="Precision-Recall Curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve")
plt.legend()
plt.show()
```



In [ ]: