In [28]:
```
!pip install pandas numpy matplotlib seaborn scikit-learn
```

```
Requirement already satisfied: pandas in /home/sargam/.conda/envs/notebook
s/lib/python3.10/site-packages (2.3.3)
Requirement already satisfied: numpy in /home/sargam/.conda/envs/notebook
s/lib/python3.10/site-packages (2.2.6)
Requirement already satisfied: matplotlib in /home/sargam/.conda/envs/note
books/lib/python3.10/site-packages (3.10.6)
Requirement already satisfied: seaborn in /home/sargam/.conda/envs/noteboo
ks/lib/python3.10/site-packages (0.13.2)
Requirement already satisfied: scikit-learn in /home/sargam/.conda/envs/no
tebooks/lib/python3.10/site-packages (1.7.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /home/sargam/.con
da/envs/notebooks/lib/python3.10/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /home/sargam/.conda/envs/no
tebooks/lib/python3.10/site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /home/sargam/.conda/envs/
notebooks/lib/python3.10/site-packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in /home/sargam/.conda/env
s/notebooks/lib/python3.10/site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /home/sargam/.conda/envs/no
tebooks/lib/python3.10/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /home/sargam/.conda/en
vs/notebooks/lib/python3.10/site-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /home/sargam/.conda/en
vs/notebooks/lib/python3.10/site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /home/sargam/.conda/env
s/notebooks/lib/python3.10/site-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /home/sargam/.conda/envs/noteb
ooks/lib/python3.10/site-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /home/sargam/.conda/env
s/notebooks/lib/python3.10/site-packages (from matplotlib) (3.2.5)
Requirement already satisfied: scipy>=1.8.0 in /home/sargam/.conda/envs/no
tebooks/lib/python3.10/site-packages (from scikit-learn) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in /home/sargam/.conda/envs/n
otebooks/lib/python3.10/site-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /home/sargam/.cond
a/envs/notebooks/lib/python3.10/site-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: six>=1.5 in /home/sargam/.conda/envs/notebo
oks/lib/python3.10/site-packages (from python-dateutil>=2.8.2->pandas) (1.
17.0)
```

In [29]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report, accu
```

In [30]:
```
data = pd.read_csv('/home/sargam/things/college/daa/lp3/ml/emails/emails.
```

In [31]:
```
print("Dataset shape:", data.shape)
print(data.info())
print(data.head())
```

```
Dataset shape: (5172, 3002)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5172 entries, 0 to 5171
Columns: 3002 entries, Email No. to Prediction
dtypes: int64(3001), object(1)
memory usage: 118.5+ MB
None
  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 [32]:
```
print("Missing values in each column:\n", data.isnull().sum())
```

```
Missing values in each column:
 Email No.     0
the            0
to             0
ect            0
and            0
              ..
military       0
allowing       0
ff             0
dry            0
Prediction     0
Length: 3002, dtype: int64
```

In [33]:
```
X = data.iloc[:, 1:-1].values  # all rows, all columns except first and l
y = data.iloc[:, -1].values    # last column: Prediction (1=spam, 0=not s
```

In [34]:
```
# Step 5: Train-test split (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
```

In [35]:
```
# Step 6: Scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [36]:
```
# Step 7: K-Nearest Neighbors (KNN) Classification
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)
y_pred_knn = knn.predict(X_test_scaled)
print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn))
print("KNN Classification Report:\n", classification_report(y_test, y_pre
print("KNN Confusion Matrix:\n", confusion_matrix(y_test, y_pred_knn))
```

```
KNN Accuracy: 0.821520618556701
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.78      0.86      1102
           1       0.63      0.93      0.75       450

    accuracy                           0.82      1552
   macro avg       0.80      0.85      0.81      1552
weighted avg       0.87      0.82      0.83      1552

KNN Confusion Matrix:
 [[856 246]
 [ 31 419]]
```

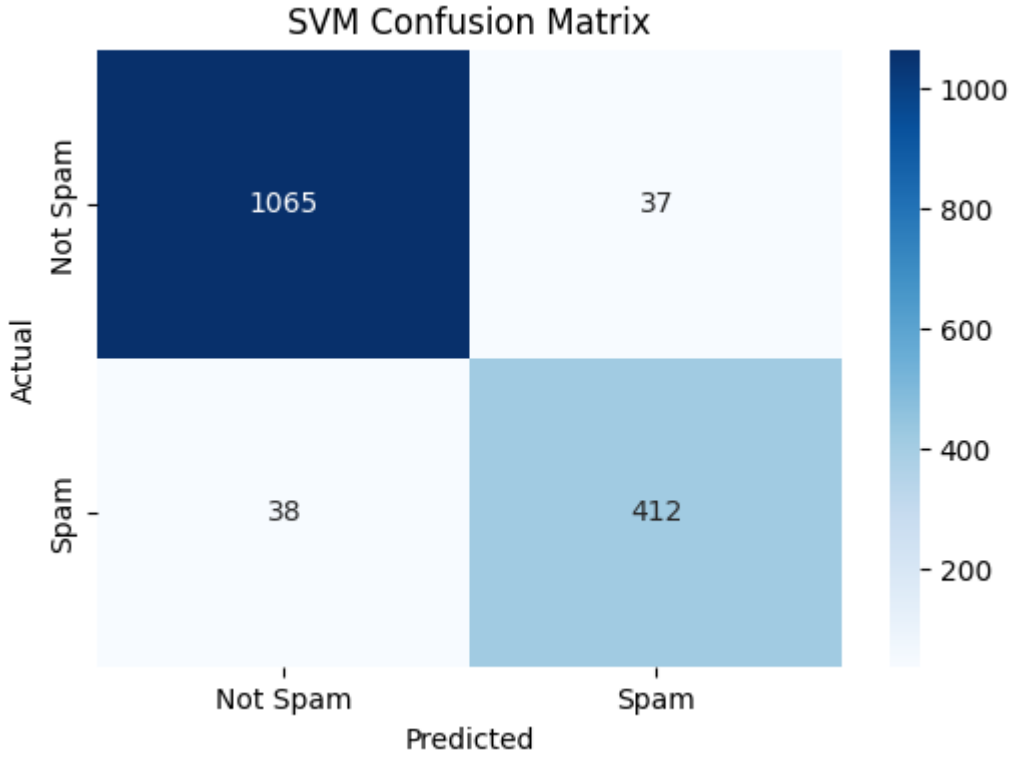
In [37]:
```python
# Step 8: SVM Classification
svm = SVC(kernel='linear', random_state=42)
svm.fit(X_train_scaled, y_train)
y_pred_svm = svm.predict(X_test_scaled)
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print("SVM Classification Report:\n", classification_report(y_test, y_pre
print("SVM Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svm))
```

```
SVM Accuracy: 0.9516752577319587
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      1102
           1       0.92      0.92      0.92       450

    accuracy                           0.95      1552
   macro avg       0.94      0.94      0.94      1552
weighted avg       0.95      0.95      0.95      1552

SVM Confusion Matrix:
 [[1065   37]
 [  38  412]]
```

In [38]:
```python
# Step 9: Compare performance
if accuracy_score(y_test, y_pred_svm) > accuracy_score(y_test, y_pred_knn
    print("SVM performed better.")
else:
    print("KNN performed better.")
```

```
SVM performed better.
```

In [39]:
```python
# Step 10: (Optional) Visualize confusion matrix for SVM
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred_svm), annot=True, fmt='d', cm
            xticklabels=['Not Spam', 'Spam'], yticklabels=['Not Spam', 'S
plt.title('SVM Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```


SVM Confusion Matrix
Not Spam
Spam
Actual
1065
37
38
412
Not Spam
Spam
Predicted
1000
800
600
400
200

In [ ]: