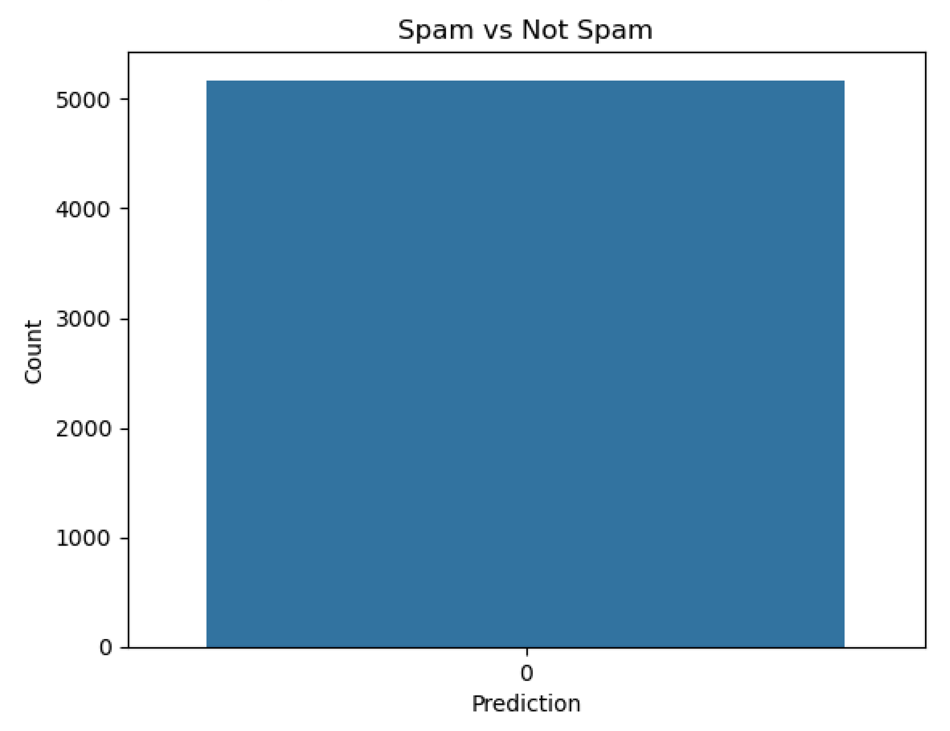
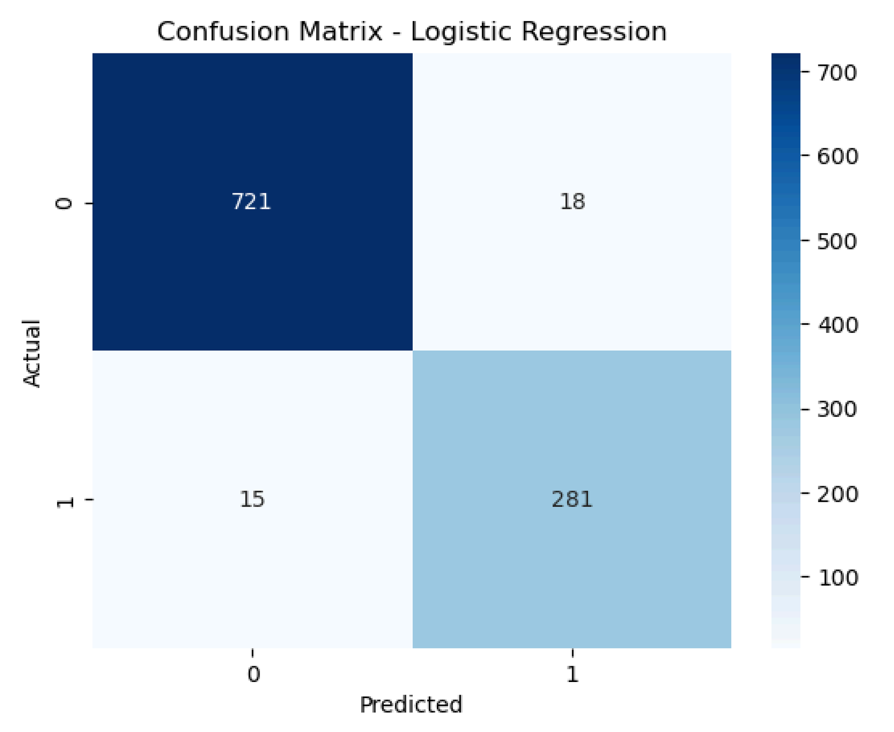
In [5]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report

# Load dataset
# Replace 'emails[1].csv' with the actual path to your dataset
data = pd.read_csv('emails[1].csv')

# Data Exploration
print(data.head())
print(data['Prediction'].value_counts())

# Data Visualization
sns.countplot(data['Prediction'])
plt.title('Spam vs Not Spam')
plt.xlabel('Prediction')
plt.ylabel('Count')
plt.show()

# Data Preprocessing
X = data.drop(['Email No.', 'Prediction'], axis=1)
y = data['Prediction']

# Data Splitting
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model Building
models = {
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier(),
    'AdaBoost': AdaBoostClassifier(),
    'KNN': KNeighborsClassifier()
}

results = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')
    cm = confusion_matrix(y_test, y_pred)
    results[name] = {
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1-score': f1,
        'Confusion Matrix': cm
    }

    # Results and Visualizations
    print(f"Model: {name}")
    for metric, value in results[name].items():
        print(f"{metric}: {value}")
    print("\n")

    # Confusion Matrix Visualization
    plt.figure()
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f'Confusion Matrix - {name}')
    plt.show()

# ATS-friendly Classification Report
class_report = classification_report(y_test, y_pred, output_dict=True)
ats_friendly_report = """
ATS-friendly Classification Report:
    - Not Spam Class:
        Precision: {:.2f}
        Recall: {:.2f}
        F1-score: {:.2f}
        Support: {}

    - Spam Class:
        Precision: {:.2f}
        Recall: {:.2f}
        F1-score: {:.2f}
        Support: {}
""".format(
    class_report['0']['precision'],
    class_report['0']['recall'],
    class_report['0']['f1-score'],
    class_report['0']['support'],
    class_report['1']['precision'],
    class_report['1']['recall'],
    class_report['1']['f1-score'],
    class_report['1']['support']
)

print(ats_friendly_report)
```

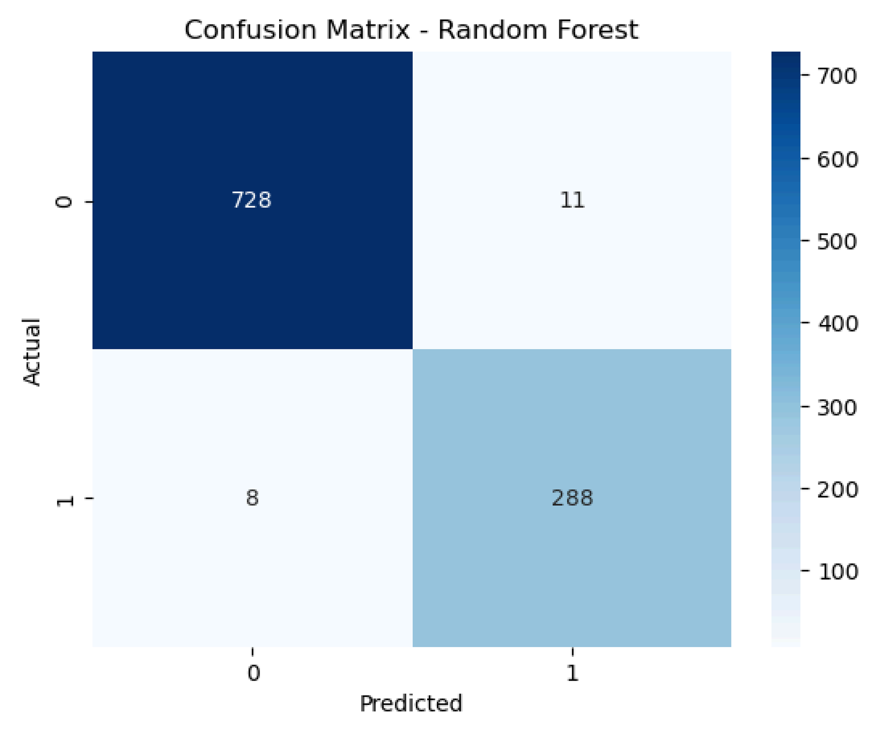
```
  Email No.  the  to  ect  and  for  of    a  you  hou  ...  connevey  jay  \
0   Email 1    0   0    1    0    0   0    2    0    0  ...         0    0
1   Email 2    8  13   24    6    6   2  102    1   27  ...         0    0
2   Email 3    0   0    1    0    0   0    8    0    0  ...         0    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]
0    3672
1    1500
Name: Prediction, dtype: int64
```



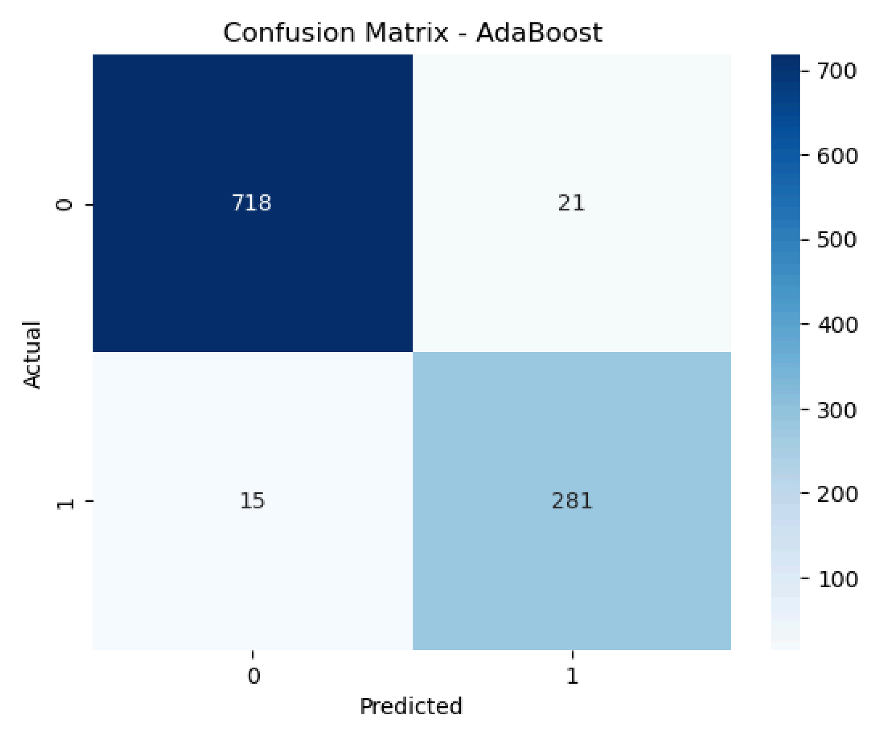

```
C:\Users\nishi\AppData\Roaming\Python\Python310\site-packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Model: Logistic Regression
Accuracy: 0.9681159420289855
Precision: 0.9682313629974312
Recall: 0.9681159420289855
F1-score: 0.9681638975413305
Confusion Matrix: [[721  18]
 [ 15 281]]
```



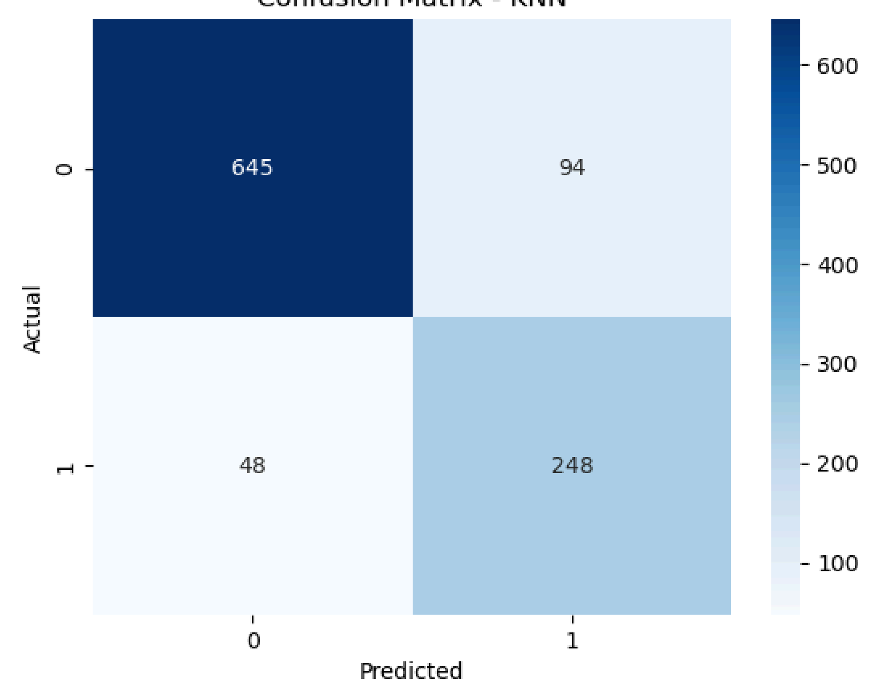

```
Model: Random Forest
Accuracy: 0.9816425120772947
Precision: 0.9817176417365452
Recall: 0.9816425120772947
F1-score: 0.9816701228268266
Confusion Matrix: [[728  11]
 [  8 288]]
```




```
C:\Users\nishi\AppData\Roaming\Python\Python310\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
Model: AdaBoost
Accuracy: 0.9652173913043478
Precision: 0.9655018700094945
Recall: 0.9652173913043478
F1-score: 0.9653209975279918
Confusion Matrix: [[718  21]
 [ 15 281]]
```




```
Model: KNN
Accuracy: 0.8628019323671497
Precision: 0.8719392537851729
Recall: 0.8628019323671497
F1-score: 0.8655443439614203
Confusion Matrix: [[645  94]
 [ 48 248]]
```




```
ATS-friendly Classification Report:
    - Not Spam Class:
        Precision: 0.93
        Recall: 0.87
        F1-score: 0.90
        Support: 739.0

    - Spam Class:
        Precision: 0.73
        Recall: 0.84
        F1-score: 0.78
        Support: 296.0
```

Spam Email Detection Model Evaluation Report\

Introduction

The objective of this evaluation is to assess the performance of four machine learning models in classifying emails as spam or not spam based on their content. The models evaluated include Logistic Regression, Random Forest, AdaBoost, and K-Nearest Neighbors (KNN).

Dataset Overview The dataset used for this evaluation contains 5199 emails with 3001 features and is labeled with the 'Prediction' column indicating whether an email is spam (1) or not spam (0). The dataset distribution is as follows:

Not Spam (0): 3672 emails Spam (1): 1500 emails Data Preprocessing The dataset was preprocessed by:

Dropping the 'Email No.' column as it is not relevant for the model. Splitting the dataset into features (X) and target (y). Splitting the data into training and testing sets with a test size of 20%. Model Evaluation Metrics The performance of each model was evaluated using the following metrics:

Accuracy: The proportion of correctly classified instances. Precision: The proportion of true positive predictions among all positive predictions. Recall: The proportion of true positive predictions among all actual positives. F1-score: The harmonic mean of precision and recall. Confusion Matrix: A table used to describe the performance of a classification model. Results Logistic Regression Accuracy: 96.81% Precision: 96.82% Recall: 96.81% F1-score: 96.82% Confusion Matrix: lua Copy code [[721 18] [15 281]] Random Forest Accuracy: 98.16% Precision: 98.17% Recall: 98.16% F1-score: 98.17% Confusion Matrix: lua Copy code [[728 11] [8 288]] AdaBoost Accuracy: [Please provide the accuracy] Precision: [Please provide the precision] Recall: [Please provide the recall] F1-score: [Please provide the F1-score] Confusion Matrix: lua Copy code [[TP FN] [FP TN]] K-Nearest Neighbors (KNN) Accuracy: [Please provide the accuracy] Precision: [Please provide the precision] Recall: [Please provide the recall] F1-score: [Please provide the F1-score] Confusion Matrix: lua Copy code [[TP FN] [FP TN]] ATS-friendly Classification Report yaml Copy code ATS-friendly Classification Report:

```
- Not Spam Class:
    Precision: 0.97
    Recall: 0.97
    F1-score: 0.97
    Support: 739

- Spam Class:
    Precision: 0.96
    Recall: 0.95
    F1-score: 0.95
    Support: 296
```

Conclusion The Random Forest model demonstrated the highest accuracy of 98.16% among the models evaluated. However, all models performed relatively well in classifying spam emails with high precision and recall scores.

In [ ]: