Load spam data from a CSV file. Train a logistic regression model on this data. Manually specify features extracted from the content of three emails (this part is simplified for illustration; real-world applications would require automated feature extraction based on the actual email content). Use the trained model to predict whether each email is spam or not. Employ Recursive Feature Elimination (RFE) to determine the importance of each feature in the dataset for predicting spam, helping identify which features may be less important.

In [7]:
```
import pandas as pd
import re
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.feature_selection import RFE

# Load the spam data
spam_data = pd.read_csv('spam-data.csv')  # Adjust path as necessary

# Separating features and target variable
X = spam_data.drop('Class', axis=1)
y = spam_data['Class']

# Building and training the logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(X, y)

# Function to extract email features based on provided criteria
def extract_email_features(email_text):
    num_words = len(email_text.split())
    num_links = len(re.findall(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F])
    num_capitalized_words = len(re.findall(r'\b[A-Z][A-Z]+\b', email_text))
    spam_words = ['win', 'prize', 'lottery', 'offer', 'discount', 'free', 'promotion', 'opportunity']
    num_spam_words = sum(word.lower() in email_text.lower() for word in spam_words)
    return [num_words, num_links, num_capitalized_words, num_spam_words]

# Reading and processing emails
emails_path = 'emails.txt'  # Adjust path as necessary
with open(emails_path, 'r') as file:
    emails_text = file.read().split('----------------')

emails_features = [extract_email_features(email) for email in emails_text]

# Predicting spam status for the extracted email features
email_predictions = model.predict(emails_features)
for i, prediction in enumerate(email_predictions, start=1):
    print(f"Email {i} is {'spam' if prediction == 1 else 'not spam'}.")

# Analyzing feature importance for spam detection using RFE
selector = RFE(model, n_features_to_select=1)
selector = selector.fit(X, y)
ranking = selector.ranking_

# Printing the feature importance ranking
print("Feature importance ranking (1 is most important):")
for i, rank in enumerate(ranking, start=1):
    print(f"Feature {i}: Rank {rank}")
```

```
Email 1 is spam.
Email 2 is not spam.
Email 3 is spam.
Feature importance ranking (1 is most important):
Feature 1: Rank 4
Feature 2: Rank 1
Feature 3: Rank 3
Feature 4: Rank 2
```

```
/opt/anaconda3/lib/python3.11/site-packages/sklearn/base.py:439: UserWarning: X does not have valid feature name
s, but LogisticRegression was fitted with feature names
  warnings.warn(
```

In [ ]: