In [1]:
```
import pandas as pd
import numpy as np
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 classification_report, roc_auc_score, confusion_matr
import seaborn as sns
import matplotlib.pyplot as plt
from imblearn.over_sampling import SMOTE   # ← NEW

# --- 1. Data Loading and Cleaning ---
try:
    df = pd.read_csv("D:/Dalmia/Tri-5 2025/ML & DL/ML Internship/spam.csv",
                     encoding='latin-1', skiprows=96, header=None)
    if df.iloc[0, 0] == 'v1':
        df.columns = df.iloc[0]
        df = df[1:].reset_index(drop=True)
    else:
        df.columns = ['Label', 'Message', 'Col3', 'Col4', 'Col5']
    df = df[['Label', 'Message']]
except Exception:
    df = pd.read_csv("D:/Dalmia/Tri-5 2025/ML & DL/ML Internship/spam.csv",
                     encoding='latin-1')
    df = df.iloc[:, [0, 1]]
    df.columns = ['Label', 'Message']

# Map labels
df['Label'] = df['Label'].map({'ham': 0, 'spam': 1})

X = df['Message']
y = df['Label']

# Train-test split
X_train_raw, X_test_raw, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print("Original Training Label Distribution:")
print(y_train.value_counts())
print("-" * 40)

# --- 2. Text Vectorization BEFORE SMOTE ---
tfidf = TfidfVectorizer(stop_words='english', max_features=5000)

# Fit-transform on training text only
X_train_vectorized = tfidf.fit_transform(X_train_raw)
X_test_vectorized = tfidf.transform(X_test_raw)

# --- 3. Apply SMOTE on Vectorized Data ---
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_vectorized, y_

print("Resampled Training Label Distribution (SMOTE):")
print(pd.Series(y_train_resampled).value_counts())
print("-" * 40)

# --- 4. Train Model ---
model = LogisticRegression(solver='liblinear', random_state=42)
```

```
model.fit(X_train_resampled, y_train_resampled)

# Predict
y_pred = model.predict(X_test_vectorized)
y_proba = model.predict_proba(X_test_vectorized)[:, 1]

print("\n--- Final Model Evaluation ---")
print(f"Accuracy Score: {accuracy_score(y_test, y_pred):.4f}")
print(f"ROC-AUC Score: {roc_auc_score(y_test, y_proba):.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Reds',
            xticklabels=['Predicted Ham (0)', 'Predicted Spam (1)'],
            yticklabels=['Actual Ham (0)', 'Actual Spam (1)'])
plt.title('Final Spam Detection Confusion Matrix (SMOTE)')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.show()
```

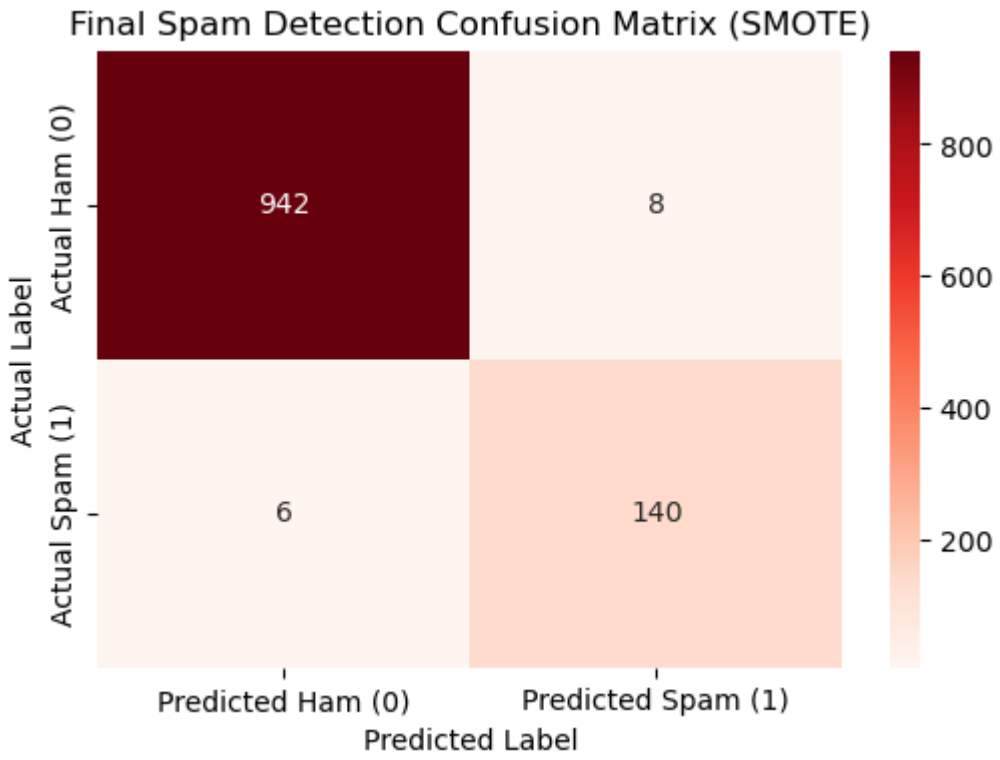
```
Original Training Label Distribution:
Label
0    3796
1     585
Name: count, dtype: int64
----------------------------------------
Resampled Training Label Distribution (SMOTE):
Label
1    3796
0    3796
Name: count, dtype: int64
----------------------------------------

--- Final Model Evaluation ---
Accuracy Score: 0.9872
ROC-AUC Score: 0.9928

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       950
           1       0.95      0.96      0.95       146

    accuracy                           0.99      1096
   macro avg       0.97      0.98      0.97      1096
weighted avg       0.99      0.99      0.99      1096
```


Final Spam Detection Confusion Matrix (SMOTE)
Actual Label
Actual Ham (0)
Actual Spam (1)
942
8
6
140
800
600
400
200
Predicted Ham (0)
Predicted Spam (1)
Predicted Label

In [ ]: