```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from imblearn.over_sampling import SMOTE
```

## Data Loading

```
df = pd.read_csv(r"C:\Users\arjun\Downloads\archive (3)\spam.csv",
encoding="ISO-8859-1")  # Try this first
df.head()

     v1                                                 v2 Unnamed: 2  \
0   ham  Go until jurong point, crazy.. Available only ...        NaN

1   ham                      Ok lar... Joking wif u oni...        NaN

2  spam  Free entry in 2 a wkly comp to win FA Cup fina...        NaN

3   ham  U dun say so early hor... U c already then say...        NaN

4   ham  Nah I don't think he goes to usf, he lives aro...        NaN


  Unnamed: 3 Unnamed: 4
0        NaN        NaN
1        NaN        NaN
2        NaN        NaN
3        NaN        NaN
4        NaN        NaN
```

## Data Preprocessing

```
df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1,inplace=True)
df.head()
```

```
     v1                                                 v2
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
```

```
df['v1'].value_counts()
```

```
v1
ham     4825
spam     747
Name: count, dtype: int64
```

```
df['v1']=df['v1'].map({'ham': 0, 'spam': 1})
```

```
df.head()
```

```
   v1                                                 v2
0   0  Go until jurong point, crazy.. Available only ...
1   0                      Ok lar... Joking wif u oni...
2   1  Free entry in 2 a wkly comp to win FA Cup fina...
3   0  U dun say so early hor... U c already then say...
4   0  Nah I don't think he goes to usf, he lives aro...
```

## Model Training

```
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(df['v2'])  # Replace 'message' with your
text column
y = df['v1']

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

# Apply SMOTE to balance classes
smote = SMOTE(random_state=42)
X_train_bal, y_train_bal = smote.fit_resample(X_train, y_train)

# Train Naïve Bayes classifier
nb = MultinomialNB()
nb.fit(X_train_bal, y_train_bal)

# Make predictions
y_pred = nb.predict(X_test)
```

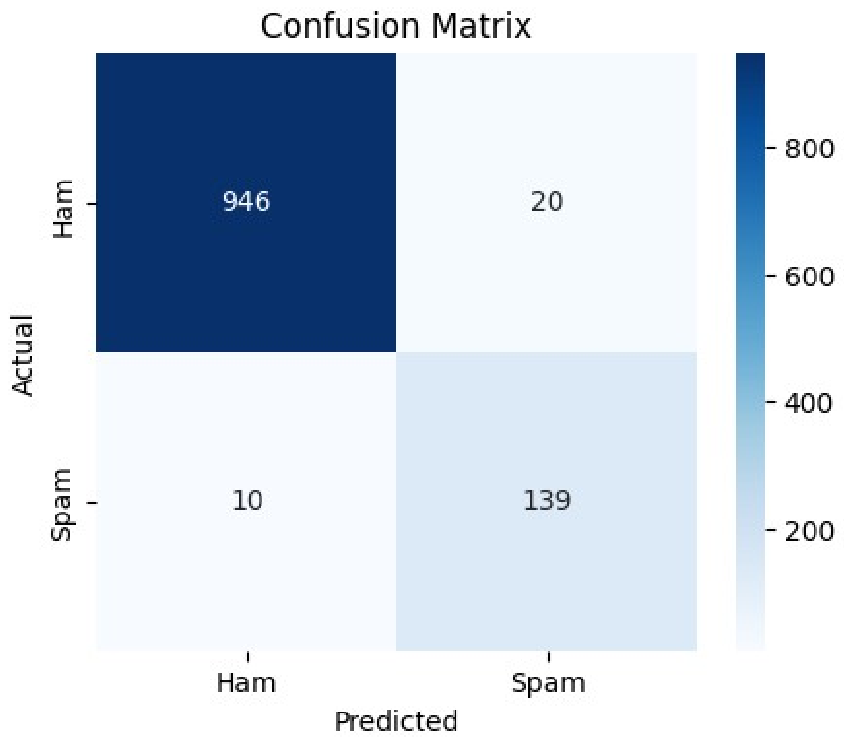
```python
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_report(y_test,
y_pred))

# Confusion matrix visualization
plt.figure(figsize=(5, 4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d',
cmap='Blues', xticklabels=['Ham', 'Spam'], yticklabels=['Ham',
'Spam'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
Accuracy: 0.97
Classification Report:
               precision    recall  f1-score   support

           0       0.99      0.98      0.98       966
           1       0.87      0.93      0.90       149

    accuracy                           0.97      1115
   macro avg       0.93      0.96      0.94      1115
weighted avg       0.97      0.97      0.97      1115
```



```python
# Training Accuracy
train_accuracy = nb.score(X_train_bal, y_train_bal)
print(f"Training Accuracy: {train_accuracy:.2f}")

# Testing Accuracy
test_accuracy = nb.score(X_test, y_test)
print(f"Testing Accuracy: {test_accuracy:.2f}")

# Compare Train & Test Accuracy
if train_accuracy - test_accuracy > 0.1:  # 10% difference is a sign
of overfitting
    print("Potential Overfitting Detected!")
else:
    print("No Overfitting Detected.")
```

```
Training Accuracy: 0.99
Testing Accuracy: 0.97
No Overfitting Detected.
```

## Testing Model

```python
sample_text = ["Congratulations! ▯ You have won a FREE iPhone 15!
Click the link below to claim your prize now. Hurry, limited time
offer! ▯ http://fake-prize.com"]

# Convert to numerical features (Assuming you used TF-IDF or
CountVectorizer)
sample_text_vectorized = vectorizer.transform(sample_text)

# Predict
prediction = nb.predict(sample_text_vectorized)

# Output Result
print("Spam" if prediction[0] == 1 else "Not Spam")
```

```
Spam
```

```python
# Example message to test (Non-Spam)
test_message = ["Hey, are we still meeting for lunch at 1 PM? Let me
know if you’re running late!"]

# Convert text into numerical features (assuming you used TF-IDF or
CountVectorizer)
test_message_transformed = vectorizer.transform(test_message)  # Use
the same vectorizer from training

# Predict using the trained model
prediction = nb.predict(test_message_transformed)
```

```python
# Output result
print("Spam" if prediction[0] == 1 else "Not Spam")
```

```
Not Spam
```

## Detailed Report