In [1]:
```
##### Standard Libraries #####
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
##### For preprocessing #####
import os
import re
import email
import codecs
##### For performance evaluation #####
import seaborn as sns
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, recall_score, precision_score
```

In [2]:
```
#### uploaded the files in the google drive and located the path

from google.colab import drive
drive.mount('/content/drive', force_remount = True)

data_path = '/content/drive/My Drive/FOURTH YEAR/Subjects/CMSC 197/trec06/data/'
labels_path = '/content/drive/My Drive/FOURTH YEAR/Subjects/CMSC 197/trec06/labels.txt'
stop_data_path = '/content/drive/My Drive/FOURTH YEAR/Subjects/CMSC 197/trec06/stop_words.txt'
```

```
Mounted at /content/drive
```

In [3]:
```
#### verified the content of the data directory 0 - 127

if os.path.exists(data_path):
    folders = os.listdir(data_path)
    sorted_folders = sorted(folders, key=lambda x: int(x))
    print("Files in the data directory:")
    for folder in sorted_folders:
        print(folder)
else:
    print(f"Directory {data_path} not found.")
```

```
Files in the data directory:
000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
```

In [4]:
```
#### initializes the email dataframe

emails_df = pd.DataFrame(columns=['folder', 'file', 'message', 'classification'])
```

In [5]:
```
#### inititialize label dataframe
#### vectorized by turning ham = 0 and spam = 1

labels_df = pd.read_csv(labels_path, sep=' ', header=None, names=['classification', 'folder_file'])
labels_df['classification'] = labels_df['classification'].apply(lambda x: 0 if x == 'ham' else 1)
labels_df['folder'] = labels_df['folder_file'].apply(lambda x: x.replace("../data/", ""))
```

In [6]:
```
emails_df1 = labels_df[['classification', 'folder']]
emails_df1.head()
```

Out[6]:

	classification	folder
0	0	000/000
1	1	000/001
2	1	000/002
3	0	000/003
4	1	000/004

In [7]:
```
#### extracting original messages from the parsed email
def get_messages(parsed_email):
    message = ""
    if parsed_email.is_multipart():
        for part in parsed_email.walk():
            if part.get_content_type() == "text/plain":
                message = part.get_payload(decode=True).decode(part.get_content_charset() or 'utf-8')
                break
    else:
        message = parsed_email.get_payload(decode=True).decode(parsed_email.get_content_charset() or 'utf-8')
    return message.strip()
```

In [8]:
```
import chardet

def get_email_charset(email_path):
    """Detect the character encoding of the email content."""
    detector = chardet.UniversalDetector()
    with open(email_path, 'rb') as e_mail:
        for line in e_mail:
            detector.feed(line)
            if detector.done:  # Check if the detection is complete
                break
    detector.close()
    return detector.result['encoding']
```

In [9]:
```
emails_df_without_stopwords = pd.DataFrame(columns=['folder', 'file', 'message', 'classification'])

folders = os.listdir(data_path)
folders.sort(key=lambda x: int(x))
for folder in folders:
```

```
    files = os.listdir(os.path.join(data_path, folder))
    files.sort()
    for file in files:
        try:
            with open(os.path.join(data_path, folder, file), "r", encoding="ISO-8859-1") as e_mail:
                read_email = e_mail.read()
                parsed_email = email.message_from_string(read_email)
                message = get_messages(parsed_email)

                ## obtaining category based on df
                category_label = emails_df1[emails_df1["folder"] == f"{folder}/{file}"]["classification"].values[0]

                ## emails_df = pd.DataFrame(columns=['folder', 'file', 'message', 'classification'])

                ## concatenate the data to emails_df
                emails_df_without_stopwords = pd.concat([emails_df_without_stopwords, pd.DataFrame([[folder, file, message, category

        except Exception:
            continue

emails_df_without_stopwords.head()
```

Out[9]:

	folder	file	message	classification
0	000	000	The mailing list I queried about a few weeks a...	0
1	000	001	LUXURY WATCHES - BUY YOUR OWN ROLEX FOR ONLY...	1
2	000	002	Academic Qualifications available from prestig...	1
3	000	003	Greetings all. This is to verify your subscri...	0
4	000	004	try chauncey may conferred the luscious not co...	1

In [10]:

```
emails_df_without_stopwords.head(-20)
```

Out[10]:

	folder	file	message	classification
0	000	000	The mailing list I queried about a few weeks a...	0
1	000	001	LUXURY WATCHES - BUY YOUR OWN ROLEX FOR ONLY...	1
2	000	002	Academic Qualifications available from prestig...	1
3	000	003	Greetings all. This is to verify your subscri...	0
4	000	004	try chauncey may conferred the luscious not co...	1
...	...	...	...	...
35274	125	294	Hi,\n \nL \nC \nV \nX \nV \nA \nP \ne \nI \nI ...	1
35275	125	295		1
35276	125	297	i can show you how you can spruce up your educ...	1
35277	125	298	Hi\n \nX\nV\nV\nL\nP\nA\nC\na\nA\nI\ne\nr\nm\n...	1
35278	125	299	%TXT_ADD	1

35279 rows × 4 columns

In [16]:
```
#### preprocessed_emails.csv are exported inside a folder
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

not_preprocessed_folder = '/content/drive/My Drive/FOURTH YEAR/Subjects/CMSC 197/trec06/not_preprocessed_data'
if not os.path.exists(not_preprocessed_folder):
    os.makedirs(not_preprocessed_folder)



## save to csv
emails_df_without_stopwords.to_csv(os.path.join(not_preprocessed_folder, 'not_preprocessed_emails.csv'), index=False, escapechar='\\

print(f"Preprocessed emails path: {not_preprocessed_folder}/not_preprocessed_emails.csv")
```

```
Mounted at /content/drive
Preprocessed emails path: /content/drive/My Drive/FOURTH YEAR/Subjects/CMSC 197/trec06/not_preprocessed_data/not_preprocessed_email
s.csv
```

Subjecting the dataset (without removing the stop words) to the different tests

In [17]:
```
#### improrting pre-processed data
from google.colab import drive
drive.mount('/content/drive', force_remount = True)

not_preprocessed_path = '/content/drive/My Drive/FOURTH YEAR/Subjects/CMSC 197/trec06/not_preprocessed_data/not_preprocessed_emails.
```

```
Mounted at /content/drive
```

In [19]:
```
#### loading the data in a dataframe
no_data = pd.read_csv(not_preprocessed_path)
no_data.head()
```

Out[19]:

	folder	file	message	classification
0	0	0	The mailing list I queried about a few weeks a...	0
1	0	1	LUXURY WATCHES - BUY YOUR OWN ROLEX FOR ONLY...	1
2	0	2	Academic Qualifications available from prestig...	1
3	0	3	Greetings all. This is to verify your subscri...	0
4	0	4	try chauncey may conferred the luscious not co...	1

In [20]:
```
#### splitting the train and the test set
no_train_df = no_data[no_data['folder'] <= 70]
no_test_df = no_data[no_data['folder'] > 70]

no_train_ham_df = no_train_df[no_train_df['classification'] == 0]
no_train_spam_df = no_train_df[no_train_df['classification'] == 1]
```

In [21]:
```
#### checking the test size of the train and test
print("Train dataset size:", len(no_train_df))
print("Test dataset size:", len(no_test_df))
print("Train ham dataset size:", len(no_train_ham_df))
print("Train spam dataset size:", len(no_train_spam_df))
```

```
Train dataset size: 19910
Test dataset size: 15389
Train ham dataset size: 7450
Train spam dataset size: 12460
```

In [22]:
```
no_data
```

Out[22]:

	folder	file	message	classification
0	0	0	The mailing list I queried about a few weeks a...	0
1	0	1	LUXURY WATCHES - BUY YOUR OWN ROLEX FOR ONLY...	1
2	0	2	Academic Qualifications available from prestig...	1
3	0	3	Greetings all. This is to verify your subscri...	0
4	0	4	try chauncey may conferred the luscious not co...	1
...	...	...	...	...
35294	126	16	bla bla bla\neee\ne\n\n\n\n\n\n\n\nrererrreerer...	1
35295	126	18	The OIL sector is going crazy. This is our wee...	1
35296	126	19	http://vdtobj.docscan.info/?23759301\n\nSuffer...	1
35297	126	20	U N I V E R S I T Y D I P L O M A S\n\nDo you...	1
35298	126	21	but moat , coverall be cytochemistry be planel...	1

35299 rows × 4 columns

In [23]:
```
no_data = no_data.dropna()
no_data
```

Out[23]:

	folder	file	message	classification
0	0	0	The mailing list I queried about a few weeks a...	0
1	0	1	LUXURY WATCHES - BUY YOUR OWN ROLEX FOR ONLY...	1
2	0	2	Academic Qualifications available from prestig...	1
3	0	3	Greetings all. This is to verify your subscri...	0
4	0	4	try chauncey may conferred the luscious not co...	1
...	...	...	...	...
35294	126	16	bla bla bla\neee\ne\n\n\n\n\n\n\nrererreerer...	1
35295	126	18	The OIL sector is going crazy. This is our wee...	1
35296	126	19	http://vdtobj.docscan.info/?23759301\n\nSuffer...	1
35297	126	20	U N I V E R S I T Y D I P L O M A S\n\nDo you...	1
35298	126	21	but moat , coverall be cytochemistry be planel...	1

33803 rows × 4 columns

In [24]:
```
no_train_ham_df
```

Out[24]:

	folder	file	message	classification
0	0	0	The mailing list I queried about a few weeks a...	0
3	0	3	Greetings all. This is to verify your subscri...	0
5	0	5	It's quiet. Too quiet. Well, how about a str...	0
6	0	6	It's working here. I have departed almost tot...	0
10	0	10	Greetings all. This is a mass acknowledgement...	0
...	...	...	...	...
19883	70	270	Here is an equation that generate all prime nu...	0
19884	70	271	Here is an equation that generate all prime nu...	0
19899	70	288	Dear DMDX Users:\n\nI would like guidance in g...	0
19903	70	293	Hi,\n\nI built up a handyboard and most of it ...	0
19908	70	298	I have mounted the IS1U60 infrared demodulator...	0

7450 rows × 4 columns

In [25]:
```
no_train_spam_df
```

Out[25]:

	folder	file	message	classification
1	0	1	LUXURY WATCHES - BUY YOUR OWN ROLEX FOR ONLY...	1
2	0	2	Academic Qualifications available from prestig...	1
4	0	4	try chauncey may conferred the luscious not co...	1
7	0	7	From NBC Today Show:\n\nIt's the look everyone...	1
8	0	8	The OIL sector is going crazy. This is our wee...	1
...	...	...	...	...
19904	70	294	%TXT_ADD	1
19905	70	295	スピード！簡単！無料！\n今どきの出会いの仕方ですね。\nhttp://get-high.b...	1
19906	70	296	Special Offer\nAdobe Video Collection\nAdobe P...	1
19907	70	297	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 T...	1
19909	70	299	http://tmqmct.overpace.net/?23757781\n\nSuffer...	1

12460 rows × 4 columns

In [26]:
```
no_train_df
```

Out[26]:

	folder	file	message	classification
0	0	0	The mailing list I queried about a few weeks a...	0
1	0	1	LUXURY WATCHES - BUY YOUR OWN ROLEX FOR ONLY...	1
2	0	2	Academic Qualifications available from prestig...	1
3	0	3	Greetings all. This is to verify your subscri...	0
4	0	4	try chauncey may conferred the luscious not co...	1
...	...	...	...	...
19905	70	295	スピード！簡単！無料！\n今どきの出会いの仕方ですね。\nhttp://get-high.b...	1
19906	70	296	Special Offer\nAdobe Video Collection\nAdobe P...	1
19907	70	297	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 T...	1
19908	70	298	I have mounted the IS1U60 infrared demodulator...	0
19909	70	299	http://tmqmct.overpace.net/?23757781\n\nSuffer...	1

19910 rows × 4 columns

In [27]:
```
no_test_df
```

Out[27]:

	folder	file	message	classification
19910	71	0	Where we can hesitantly derive perverse satisf...	1
19911	71	1	There are several things you can use to perfor...	0
19912	71	2	Best offer of the month:\n\nViggra - $76.95\nC...	1
19913	71	3	De i ar Home O h wne n r , \n \nYour cr v ed ...	1
19914	71	4	Special Offer\nAdobe Video Collection\nAdobe P...	1
...	...	...	...	...
35294	126	16	bla bla bla\neee\ne\n\n\n\n\n\n\n\nrererreerer...	1
35295	126	18	The OIL sector is going crazy. This is our wee...	1
35296	126	19	http://vdtobj.docscan.info/?23759301\n\nSuffer...	1
35297	126	20	U N I V E R S I T Y D I P L O M A S\n\nDo you...	1
35298	126	21	but moat , coverall be cytochemistry be planel...	1

15389 rows × 4 columns

In [28]:
```
#### Counting top 10000 words from the training dataset
word_counts = {}

for index, row in no_train_df.iterrows():
    for word in str(row['message']).split():
        word_counts[word] = word_counts.get(word, 0) + 1

## getting 10000 words & corresponding frequency
sorted_words = sorted(word_counts.items(), key=lambda x: x[1], reverse=True)[:10000]
top_10000_words = dict(sorted_words)
top_10000_words_list = list(top_10000_words.keys())

feature_matrix_spam = np.zeros((len(no_train_spam_df), 10000))

for index in range(len(no_train_spam_df)):
    for word in str(no_train_spam_df.iloc[index]['message']).split():
        if word in top_10000_words:
            feature_matrix_spam[index][top_10000_words_list.index(word)] = 1
```

In [29]:
```
#### creating word counts dictionary and get the top 10,000 words
from collections import Counter
```

```
word_counts = Counter(word for message in no_train_df['message'] for word in str(message).split())
top_10000_words = dict(word_counts.most_common(10000))
top_10000_words_list = list(top_10000_words.keys())
top_10000_words
```

Out[29]:

```
{'the': 102563,
 'to': 63681,
 'and': 52098,
 '>': 52095,
 'a': 49652,
 'of': 47061,
 'I': 32805,
 'in': 29602,
 'is': 29076,
 'for': 24090,
 'you': 21010,
 'that': 19920,
 'on': 17705,
 'with': 16639,
 'be': 15122,
 'it': 14650,
 'have': 13392,
 'this': 13225,
 '-': 12673,
 'are': 11970,
 'your': 11169,
 'The': 11151,
 'from': 10800,
 'as': 10664,
 'or': 10227,
 'at': 9975,
 'will': 9641,
 'not': 9641,
 'A': 9629,
 'by': 9168,
 '<td': 9082,
 'can': 8004,
 'but': 7993,
 'an': 7356,
 'was': 6318,
 'we': 6285,
 'has': 6228,
 'my': 5772,
 'if': 5541,
 '>>': 5348,
 'all': 5300,
 'would': 5258,
 'our': 4992,
 'one': 4978,
 'i': 4811,
 'about': 4745,
```

```
'any': 4686,
'You': 4622,
'get': 4578,
'This': 4454,
'some': 4403,
'|': 4280,
'up': 4233,
'do': 4110,
'like': 4084,
'<br>': 4012,
'use': 3990,
'which': 3918,
'out': 3881,
'</tr>': 3808,
'more': 3740,
'there': 3636,
'they': 3597,
'what': 3589,
'We': 3544,
'<tr>': 3498,
'L': 3490,
'so': 3472,
'Adobe': 3465,
'been': 3459,
'e': 3451,
'If': 3442,
'am': 3370,
'when': 3308,
'me': 3287,
'only': 3277,
'V': 3234,
'its': 3165,
'know': 3106,
',': 3076,
'just': 3074,
'could': 3065,
'no': 3007,
'3': 2964,
'using': 2922,
'other': 2920,
'nil': 2899,
'и': 2894,
'*': 2852,
'1': 2819,
'how': 2748,
'very': 2735,
'2': 2624,
```

```
'price:': 2595,
'should': 2584,
'board': 2572,
'n': 2561,
'may': 2546,
'.': 2544,
'It': 2544,
'also': 2530,
'Professional': 2528,
'had': 2517,
'--': 2500,
'them': 2454,
'In': 2450,
'need': 2447,
'he': 2423,
'time': 2404,
'into': 2337,
'used': 2315,
'<p': 2294,
'wrote:': 2259,
'their': 2257,
'than': 2244,
'then': 2240,
'=': 2226,
'make': 2218,
'see': 2204,
'way': 2156,
'these': 2146,
"don't": 2138,
'company': 2126,
"I'm": 2120,
'were': 2108,
't': 2105,
'program': 2101,
'want': 2084,
'now': 2080,
'anyone': 2048,
'c': 2027,
'new': 2012,
'Our': 2011,
'two': 1992,
'Microsoft': 1985,
'From:': 1984,
'o': 1963,
'Hi': 1962,
'Subject:': 1952,
'r': 1941,
```

```
'who': 1919,
'where': 1917,
'C': 1913,
'X': 1876,
'Pro': 1858,
'help': 1827,
'MS': 1805,
'R': 1797,
'me,': 1789,
'message': 1772,
'1998': 1769,
'в': 1740,
'x': 1733,
'Windows': 1715,
'first': 1711,
'over': 1707,
'S': 1703,
'M': 1694,
'work': 1692,
'&': 1687,
"it's": 1673,
'v': 1671,
'$': 1671,
'information': 1667,
'us': 1665,
'/': 1662,
'find': 1661,
'<a': 1635,
'his': 1631,
'border="0"': 1619,
'go': 1604,
'Info': 1602,
'good': 1595,
'does': 1592,
'z': 1589,
'::': 1578,
'Date:': 1574,
'much': 1566,
'HB': 1548,
':': 1539,
'U': 1527,
'Office': 1523,
'b': 1520,
'Received:': 1506,
'Studies': 1502,
'gold': 1493,
'because': 1490,
```

```
'8': 1487,
'More': 1482,
'Content-Type:': 1481,
"Women's": 1480,
'give': 1478,
'G': 1477,
'think': 1473,
'|': 1452,
'University': 1448,
'2003': 1447,
'here': 1447,
'5': 1446,
'Inc.': 1446,
'XP': 1439,
'still': 1437,
'<TD': 1429,
'm': 1426,
'Is': 1425,
'id': 1415,
'being': 1409,
'}': 1400,
'New': 1397,
'Thanks': 1394,
'try': 1392,
'same': 1387,
'people': 1386,
'through': 1386,
'list': 1381,
'problem': 1370,
'please': 1365,
'motor': 1358,
'run': 1353,
'To:': 1351,
'file': 1339,
'1.5': 1329,
'/*': 1319,
'reviews': 1314,
'code': 1314,
'even': 1313,
'Rating:': 1305,
'Retail': 1301,
'*/': 1301,
'save:': 1296,
'[Add': 1296,
'cart]': 1296,
'number': 1289,
'{': 1287,
```

```
'last': 1284,
'Your': 1283,
'7': 1278,
'each': 1278,
'Please': 1276,
'it.': 1273,
'take': 1268,
'send': 1268,
'many': 1261,
"I've": 1259,
'Company': 1255,
'read': 1247,
'power': 1244,
'List': 1235,
'!': 1230,
'next': 1225,
'most': 1215,
'For': 1209,
'before': 1204,
'got': 1204,
'<p>': 1202,
'set': 1196,
'great': 1195,
'ra': 1195,
'P': 1193,
'following': 1182,
'might': 1179,
'All': 1174,
'within': 1168,
'available': 1164,
'sure': 1154,
'6': 1153,
'IC': 1150,
'<font': 1148,
'Big': 1147,
'email': 1145,
'really': 1142,
'To': 1135,
'1999': 1132,
'<WMST-L@UMDD.UMD.EDU>': 1132,
'What': 1128,
'off': 1122,
'without': 1106,
'such': 1097,
'On': 1096,
'Helvetica,': 1096,
'did': 1092,
```

```
'something': 1092,
'+': 1082,
'data': 1075,
'May': 1072,
'd': 1072,
'$69.95': 1070,
'going': 1063,
'already': 1061,
'4': 1060,
'money': 1049,
'after': 1046,
'looking': 1045,
'10': 1040,
'windowtext': 1040,
'There': 1032,
'both': 1026,
'trying': 1021,
'back': 1020,
'size=2': 1011,
'around': 1006,
'stock': 999,
'right': 996,
'Content-Transfer-Encoding:': 995,
'address': 992,
'No': 985,
'7bit': 984,
'At': 980,
'between': 978,
'<meta': 976,
'system': 976,
'different': 975,
'best': 973,
'well': 972,
'found': 968,
'<table': 967,
'\\\\': 967,
'And': 965,
'%TXT_ADD': 965,
'getting': 963,
'provide': 962,
'color=red': 962,
'while': 954,
'under': 950,
'tried': 947,
'width="50%"><font': 946,
'width="12%"><font': 946,
'1.0': 946,
```

```
'Handy': 943,
'able': 940,
'</td>': 940,
'Hi,': 939,
"doesn't": 936,
'2006': 936,
'part': 935,
'When': 930,
'development': 928,
'∽∽∽☆': 927,
'another': 926,
'must': 926,
'----------------------------------------------------------------': 923,
'current': 920,
'said': 919,
'My': 909,
'those': 906,
'robot': 904,
's': 903,
'Reply-To:': 903,
'THE': 899,
'0': 899,
'made': 898,
'text/plain;': 893,
'9': 891,
'Sender:': 890,
'They': 887,
'port': 886,
'look': 880,
'Corp.': 880,
'Arial,': 876,
'Sep': 875,
'currently': 873,
'1997': 873,
'Board': 872,
'$149.95': 868,
'better': 863,
'After': 860,
'000': 858,
'</html>': 857,
'receive': 856,
'N': 853,
'control': 852,
'<html>': 850,
'offer': 850,
'serial': 848,
'put': 847,
```

'Re:': 844,
'version': 840,
'site': 840,
'gas': 839,
'since': 835,
'software': 834,
'But': 832,
'low': 829,
'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA': 829,
'few': 827,
'Do': 820,
'life': 820,
'having': 817,
'download': 815,
'days': 814,
'Get': 813,
'mail': 813,
'p': 810,
'little': 805,
'</body>': 799,
'на': 798,
'y': 798,
'based': 797,
'Aug': 795,
'Now': 793,
"can't": 790,
'start': 789,
'let': 783,
'down': 777,
'own': 776,
'things': 776,
'forward': 776,
'free': 774,
'running': 772,
'Apple': 772,
'As': 771,
'Any': 769,
'price': 768,
'line': 768,
'always': 767,
'order': 767,
'report': 767,
'oil': 767,
'Does': 760,
'HTML': 758,
'He': 756,
'f': 756,

```
'web': 756,
'small': 752,
'working': 751,
'?': 751,
'c': 750,
'long': 750,
'motors': 746,
'no': 745,
'E': 744,
'l': 744,
'Energy': 740,
'-0500': 737,
'Golden': 736,
'width="14%">Windows</td>': 731,
'How': 725,
'works': 725,
'real': 724,
'face="Verdana,': 724,
'possible': 723,
'Thank': 720,
'digital': 720,
'handyboard': 716,
'never': 715,
'Wed,': 714,
'several': 712,
'contact': 711,
'someone': 706,
'nan': 706,
'problems': 705,
'tell': 705,
'(': 703,
'seems': 701,
'Thu,': 701,
'you.': 698,
'China': 697,
'IR': 697,
'world': 696,
'Act': 695,
'One': 694,
'問) ': 693,
'come': 691,
'de': 691,
'Can': 688,
'PM': 688,
'end': 687,
')': 685,
'known': 684,
```

```
'change': 684,
'too': 682,
'research': 682,
'g': 682,
'received': 682,
'1996': 681,
'until': 679,
'So': 679,
'pleased': 678,
'check': 678,
"It's": 674,
'Oil': 674,
'e-mail': 673,
'sent': 672,
'<TR>': 672,
'output': 671,
'Cantex': 670,
'Department': 667,
'years': 666,
'content="text/html;': 666,
'h': 666,
'<span': 665,
'q': 662,
'O': 661,
'per': 657,
'computer': 657,
'second': 655,
'three': 654,
'today': 653,
'it,': 652,
'campaign': 651,
'again': 650,
'thing': 650,
'interested': 649,
'market': 648,
'expansion': 644,
'full': 643,
'<head>': 642,
'technology': 642,
'potential': 638,
'Photoshop': 634,
'every': 634,
'business': 630,
'place': 630,
'battery': 630,
'sensor': 630,
'high': 628,
```

```
'#': 627,
'million': 626,
'kind': 625,
'handy': 624,
'keep': 623,
'point': 621,
'account': 619,
'light': 619,
'...': 617,
'news': 617,
'side': 617,
'Just': 616,
'service': 614,
'doing': 610,
'support': 609,
'<body>': 608,
'</table>': 606,
'shares': 605,
'servo': 605,
'De': 599,
'name': 598,
'fast': 595,
'day': 594,
'access': 594,
'================================================================================': 593,
'Mon,': 591,
'product': 590,
'Special': 589,
'-0700': 589,
'FOR': 587,
'u': 585,
'probably': 585,
'http-equiv="Content-Type"': 583,
'error': 580,
'T': 580,
'students': 579,
'voltage': 579,
'input': 578,
'Canyon': 578,
'including': 577,
'↓': 576,
'seen': 575,
'24': 574,
'why': 574,
'bit': 574,
'TO': 574,
'enough': 573,
```

'Call': 572,
'done': 571,
'third': 571,
'call': 570,
'wish': 569,
'sonar': 568,
'additional': 567,
'design': 567,
'◇': 566,
'question': 564,
'AND': 563,
'Acrobat': 563,
'k': 563,
'Has': 562,
'test': 561,
'companies': 559,
'came': 558,
'value': 558,
'else': 556,
'connect': 556,
'1.': 555,
'Dear': 553,
'Fri,': 553,
'production': 552,
'CS2': 552,
'Tue,': 552,
'expect': 550,
'idea': 549,
'her': 549,
'pin': 548,
'>I': 547,
'w': 546,
'simply': 546,
'BY': 542,
'Premiere': 540,
'seem': 538,
'<img': 538,
'believe': 538,
'Science': 537,
'week': 537,
'j': 537,
'Apr': 537,
'15': 537,
'· · · ...……━━━━━━━━━━━━━━━━━━━━━━……....· · ·': 536,
'page': 536,
'int': 536,
'MIME-Version:': 535,

```
'That': 534,
'write': 534,
'at:': 532,
'100%': 531,
'】': 531,
'Edition': 530,
'include': 530,
'Information': 530,
'online': 529,
'cellspacing="0"': 529,
'files': 528,
'say': 527,
'Prospect': 527,
'connected': 527,
'prices': 526,
'various': 525,
'Effects': 524,
'memory': 524,
'large': 524,
'ever': 523,
'Video': 522,
'recent': 522,
'special': 522,
'25': 520,
'Some': 520,
'quite': 520,
'</head>': 519,
"I'd": 519,
'seismic': 516,
'chip': 516,
'mailing': 515,
'</font>': 513,
'□----------------------------------------------------------------': 512,
'handyboard@media.mit.edu': 512,
'away': 511,
'These': 511,
'<div': 511,
'once': 510,
'analog': 509,
'starting': 508,
'far': 504,
'Gas,': 504,
'turn': 504,
'width="100%">': 503,
'lose': 503,
'four': 502,
'investment': 502,
```

```
'all,': 502,
'class="text"': 500,
'least': 499,
'lot': 498,
'anything': 497,
'Price:': 497,
'past': 497,
'soon': 497,
'drive': 495,
'turned': 494,
'Current': 492,
'LCD': 492,
'2.': 491,
"Don't": 491,
'ESMTP': 490,
'class': 488,
'A.': 487,
'April': 487,
'6.5': 486,
'return': 484,
'Are': 483,
'shareholders': 482,
'close': 482,
'Jan': 482,
'face=Verdana': 482,
'experience': 481,
'further': 480,
'launch': 480,
'⟪': 480,
'Best': 479,
'John': 478,
'(PDT)': 478,
'build': 477,
'space': 477,
'7.0': 476,
'buy': 476,
'Jul': 475,
'16': 475,
'Product': 475,
'questions': 474,
'Texas.': 474,
'☆': 474,
'-0400': 473,
'However,': 473,
'AM': 473,
'hope': 472,
'old': 472,
```

```
'OF': 469,
'during': 469,
'board.': 469,
'75>;': 469,
'Engineering': 468,
'pretty': 468,
'office': 468,
'future': 467,
'12': 467,
'$49.95': 466,
'times': 466,
'show': 465,
'less': 465,
'longer': 464,
'Handyboard': 464,
'0in': 464,
'Gold': 463,
'add': 463,
'application': 463,
'hard': 460,
'With': 459,
'signal': 459,
'website': 459,
'approved': 458,
'deal': 457,
'Fred': 457,
'０円': 457,
'type': 456,
'width=3D238': 456,
'valign=3Dtop': 456,
'.5pt;': 456,
'hours': 455,
'directly': 455,
'$99.95': 453,
'source': 452,
'short': 452,
'shipping': 451,
'thought': 450,
'common': 449,
'properties': 448,
'Why': 448,
'store': 448,
'making': 448,
'left': 448,
'products': 448,
'http://love-match.bz/pc/?06': 448,
'simple': 447,
```

```
'West': 447,
'(from': 447,
'イリュージョン': 446,
'link': 446,
'process': 445,
'effective': 445,
'solution': 445,
'Sincerely,': 444,
'matter': 444,
"didn't": 442,
'industry': 442,
'B': 441,
'drilling': 439,
'feel': 439,
'Email': 439,
'via': 439,
'Offer': 438,
'DVD': 438,
'sell': 438,
'women': 438,
'+1': 437,
'Collection': 437,
'this.': 437,
'focus': 437,
'anybody': 437,
'Ross': 437,
'reading': 435,
'Securities': 435,
'American': 434,
'plan': 434,
'9.0': 434,
'early': 433,
'$200.00': 433,
"I'll": 433,
'either': 433,
'easy': 433,
'Audition': 432,
'Encore': 432,
'Bestsellers': 432,
'$550.00': 432,
'$480.05': 432,
'(87%)': 432,
'$150.05': 432,
'(75%)': 432,
'$599.00': 432,
'$529.05': 432,
'(88%)': 432,
```

```
'By': 431,
'help.': 431,
'IN': 430,
'complete': 429,
'Then': 427,
'sensors': 427,
'darkwing.uoregon.edu': 427,
'limited': 426,
'date': 424,
'Thanks,': 424,
'continue': 424,
'fact': 424,
'Nov': 424,
'class=3DMsoNormal><![if': 424,
'!supportEmptyParas]> <![endif]><span': 424,
"style=3D'font-size:14.0pt;mso-bidi-font-size:12.0pt'><o:p></o:p></span><=": 424,
'/p>': 424,
'30': 422,
'yet': 421,
'cost': 420,
'Watch': 419,
'nil]': 418,
'Timeout': 417,
'DMDX': 417,
'weight': 416,
'form': 416,
'him': 416,
'instead': 416,
'load': 416,
'sans-serif"': 416,
'building': 415,
'exploration': 414,
'you,': 414,
'main': 414,
'No.:': 414,
'vAlign="top"': 414,
'US': 413,
'YOUR': 412,
'CORP': 412,
'20': 412,
'PC': 411,
'D': 411,
'Will': 411,
'week.': 408,
'Trade': 408,
'thanks': 408,
'top': 408,
```

```
'CWTD': 408,
'align="left"': 408,
'для': 407,
'processing': 407,
'etc.': 407,
'200>"+"/*<': 407,
'opportunity': 405,
'21,': 405,
'book': 405,
'means': 405,
'video': 405,
'called': 404,
'case': 404,
'project': 404,
'year': 403,
'rather': 401,
'Program': 401,
'&#8729;': 400,
'wondering': 400,
'∞————————————————∞': 400,
'cannot': 399,
'reply': 399,
'expected': 398,
"haven't": 397,
'Mike': 396,
'Good': 396,
'color=#808080': 396,
'Computer': 394,
'pay': 394,
'ms': 394,
'whether': 393,
'align="center"><b><font': 393,
'Manager': 393,
'circuit': 393,
'acres': 392,
'(or': 392,
'11': 392,
'unique': 392,
'color="white"': 391,
'Franklin': 391,
'format.': 390,
'allow': 390,
"Company's": 390,
'News': 390,
'te': 390,
'Message-----': 390,
'loss': 387,
```

```
'color=gray><b>Adobe': 387,
'engaged': 386,
'Campaign': 385,
'announce': 385,
'details': 385,
'home': 384,
'function': 384,
'device': 384,
'numbers': 384,
'/<msfd': 384,
'open': 383,
'highly': 382,
'cash': 382,
'weeks': 381,
'share': 381,
'present': 381,
'credit': 381,
'local': 380,
'announced': 380,
'World': 380,
'statements': 380,
'<Delay': 380,
'private': 379,
'above': 379,
'interface': 379,
'/*<Timeout': 379,
'2000><bmp>"interrog"<Delay': 379,
'ask': 378,
'http://hipergirl.com/': 378,
'fine': 377,
'THIS': 377,
'Free': 377,
'Internet': 376,
'problem.': 376,
'now,': 375,
'later': 375,
'display': 375,
'>From': 374,
'field': 374,
'related': 374,
'-----Original': 374,
'nothing': 373,
'big': 373,
'similar': 373,
'choice': 373,
'Research': 373,
'coming': 373,
```

```
'In-Reply-To:': 373,
'below': 372,
'original': 372,
'makes': 372,
'3.': 371,
'move': 371,
'course': 371,
'Dec': 371,
'given': 370,
'quality': 370,
'producer': 370,
'Message-ID:': 369,
'update': 368,
'(I': 367,
'MIME': 366,
'text/html;': 366,
'standard': 365,
'driver': 365,
'PR': 364,
'orgasms': 364,
'went': 362,
'inform': 362,
'size': 362,
'size="2"': 362,
'</TR>': 362,
'held': 361,
'comes': 361,
'due': 361,
'looks': 361,
'written': 361,
'speed': 361,
'■': 361,
'though': 360,
'huge': 360,
'now.': 360,
'Greko': 360,
'14': 359,
'board,': 359,
'ability': 358,
'everything': 358,
'across': 357,
'San': 356,
'happy': 356,
'time.': 356,
'couple': 356,
'stop': 355,
'them.': 355,
```

```
'(and': 355,
'mode': 355,
'interest': 354,
'cause': 353,
'shall': 353,
'confirm': 352,
'major': 352,
'Time': 351,
'is,': 351,
'men': 350,
'started': 350,
'Y': 350,
'actually': 350,
'starship-design:': 350,
...}
```

In [30]:
```
## sparse matrix
messages_split = no_train_df['message'].apply(lambda x: str(x).split())
max_words = max(messages_split.apply(len))
max_columns = 127

df_words = pd.DataFrame(np.full((len(messages_split), max_columns), None))
for i, words in enumerate(messages_split):
    for j, word in enumerate(words[:max_columns]):
        df_words.iloc[i, j] = word

df_words.head()
```

Out[30]:

	0	1	2	3	4	5	6	7	8	9	...	117	118	119	120
0	The	mailing	list	I	queried	about	a	few	weeks	ago	...	Catholics	on	the	net
1	LUXURY	WATCHES	-	BUY	YOUR	OWN	ROLEX	FOR	ONLY	$219!	...	None	None	None	None
2	Academic	Qualifications	available	from	prestigious	NON-ACC	REDITED	uni	versities.	Do	...	None	None	None	None
3	Greetings	all.	This	is	to	verify	your	subscription	to	the	...	None	None	None	None
4	try	chauncey	may	conferred	the	luscious	not	continued	a	tonsillitis	...	None	None	None	None

5 rows × 127 columns

In [31]:
```
#### initializes feature matrix for spam
featurematrix_spam = np.zeros((len(no_train_spam_df), len(top_10000_words)), dtype=int)
```

```
top_10000_words_list = list(top_10000_words.keys())

for index in range(len(no_train_spam_df)):
    words = str(no_train_spam_df.iloc[index]['message']).split()
    for word in words:
        if word in top_10000_words:

            featurematrix_spam[index][top_10000_words_list.index(word)] = 1


featurematrix_spam
```

Out[31]:
```
array([[1, 0, 0, ..., 0, 0, 0],
       [1, 1, 1, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 1, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 1, 1, ..., 0, 0, 0]])
```

In [33]:
```
#### initializing feature matrix for the ham
featurematrix_ham = np.zeros((len(no_train_ham_df), len(top_10000_words)), dtype=int)
top_10000_words_list = list(top_10000_words.keys())

for index in range(len(no_train_ham_df)):
    words = str(no_train_ham_df.iloc[index]['message']).split()
    for word in words:
        if word in top_10000_words:
            featurematrix_ham[index][top_10000_words_list.index(word)] = 1

featurematrix_ham
```

Out[33]:
```
array([[1, 1, 1, ..., 0, 0, 0],
       [1, 1, 1, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [1, 1, 1, ..., 0, 0, 0],
       [1, 1, 1, ..., 0, 0, 0],
       [1, 1, 1, ..., 0, 0, 0]])
```

In [34]:
```
#### Calculate prior probabilities for spam and ham
prior_spam = len(no_train_spam_df) / len(no_train_df)
prior_ham = len(no_train_ham_df) / len(no_train_df)

print(f"Prior probability of spam: {prior_spam}")
print(f"Prior probability of ham: {prior_ham}")
```

```
Prior probability of spam: 0.6258161727774988
Prior probability of ham: 0.37418382722250126
```

In [35]:
```python
#### function for laplace smoothing
def laplace_smoothing(feature_matrix_spam, feature_matrix_ham, laplace_smoothing_val, num_classes):
    p_word_given_spam = np.zeros(len(top_10000_words))
    p_word_given_ham = np.zeros(len(top_10000_words))

    spam_word_count = np.sum(feature_matrix_spam, axis=0)
    ham_word_count = np.sum(feature_matrix_ham, axis=0)

    total_spam_words = np.sum(spam_word_count)
    total_ham_words = np.sum(ham_word_count)


    for i in range(len(top_10000_words)):
        p_word_given_spam[i] = (spam_word_count[i] + laplace_smoothing_val) / (total_spam_words + laplace_smoothing_val * num_classe
        p_word_given_ham[i] = (ham_word_count[i] + laplace_smoothing_val) / (total_ham_words + laplace_smoothing_val * num_classes)

    return p_word_given_spam, p_word_given_ham

## initializing laplace smoothing parameter and number of classes
laplace_smoothing_val = 1
num_classes = 2
spam_word_probs, ham_word_probs = laplace_smoothing(featurematrix_spam, featurematrix_ham, laplace_smoothing_val, num_classes)
```

In [36]:
```python
#### print likelihood of being spam or ham

print(f"Likelihood of a word being in a spam email: {spam_word_probs}")
print(f"Likelihood of a word being in a ham email: {ham_word_probs}")
```

```
Likelihood of a word being in a spam email: [6.96215484e-03 7.24301209e-03 7.54080545e-03 ... 4.23402889e-06
 4.23402889e-06 3.52835741e-05]
Likelihood of a word being in a ham email: [1.02863276e-02 9.72670474e-03 8.48940166e-03 ... 3.06642646e-05
 2.45314117e-05 7.66606616e-06]
```

In [37]:
```python
## table form of the likelihood

likelihood_df = pd.DataFrame({
    'Word': top_10000_words_list,
    'P(Word|Spam)': spam_word_probs,
    'P(Word|Ham)': ham_word_probs
})

likelihood_df.head(20)
```

Out[37]:

	Word	P(Word|Spam)	P(Word|Ham)
0	the	0.006962	0.010286
1	to	0.007243	0.009727
2	and	0.007541	0.008489
3	>	0.000210	0.004255
4	a	0.007096	0.009031
5	of	0.005957	0.008091
6	I	0.002593	0.008335
7	in	0.006738	0.007008
8	is	0.005082	0.007579
9	for	0.004681	0.007215
10	you	0.004934	0.005691
11	that	0.003321	0.006449
12	on	0.004039	0.006211
13	with	0.003856	0.005889
14	be	0.003579	0.005490
15	it	0.003228	0.006225
16	have	0.003319	0.006141
17	this	0.003144	0.005014
18	-	0.002921	0.002160
19	are	0.003153	0.004296

In [38]:

```
#### classifying the emails using the computed probabilities
def classify_email(email, spam_word_probs, ham_word_probs, p_spam, p_ham):
    log_p_spam = 0
    log_p_ham = 0

    words = str(email).split()
```

```
    for word in words:
        if word in top_10000_words:
            log_p_spam += np.log(spam_word_probs[top_10000_words_list.index(word)])
            log_p_ham += np.log(ham_word_probs[top_10000_words_list.index(word)])

    log_p_spam += np.log(p_spam)
    log_p_ham += np.log(p_ham)

    return 1 if log_p_spam > log_p_ham else 0
```

In [43]:
```
# prompt: table form the df in classifying email

no_test_df['predicted_classification'] = no_test_df['message'].apply(lambda x: classify_email(x, spam_word_probs, ham_word_probs, pr

classification_results_df = pd.DataFrame({
    'Message': no_test_df['message'],
    'Actual Classification': no_test_df['classification'],
    'Predicted Classification': no_test_df['predicted_classification']
})

classification_results_df.head(20)
```

```
<ipython-input-43-90081e54f90e>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  no_test_df['predicted_classification'] = no_test_df['message'].apply(lambda x: classify_email(x, spam_word_probs, ham_word_probs, prior_spam, prior_ham))
```

Out[43]:

	Message	Actual Classification	Predicted Classification
19910	Where we can hesitantly derive perverse satisf...	1	1
19911	There are several things you can use to perfor...	0	0
19912	Best offer of the month:\n\nViggra - $76.95\nC...	1	1
19913	De i ar Home O h wne n r , \n \nYour cr v ed ...	1	1
19914	Special Offer\nAdobe Video Collection\nAdobe P...	1	1
19915	This is a multi-part message in MIME format.\n...	1	1
19916	%TXT_ADD	1	1
19917	The Mistersporty Incorporation\nRambrantplein ...	1	1
19918	ED Choice, your best choice for ED drugs\n\nVi...	1	1
19919	I've changed the DMDX listserv subject filter,...	0	0
19920	I noticed in documentation on input/output wit...	0	0
19921	I am putting together an n-back experiment and...	0	0
19922	%TXT_ADD	1	1
19923	At 04:41 PM 8/30/00 -0500, you wrote:\n>I noti...	0	0
19924	At 04:50 PM 8/30/00 -0500, you wrote:\n>I am p...	0	0
19925	It appears that my last message was cut off (a...	0	0
19926	Thanks for the input. Sorry for the duplicate...	0	0
19927	Just getting set up (been an assistant prof fo...	0	0
19928	Dear Homeowner, \n\nhttp://usmortz.com\n\nYou...	1	1
19929	DISCOUNTED! QUALITY! SECURE!\n\nAre you lookin...	1	1

In [44]:
```
#### classify the test emails
no_test_df.loc[:,'predicted'] = no_test_df['message'].apply(lambda x: classify_email(x, spam_word_probs, ham_word_probs, prior_spam
correct_test = (no_test_df['classification'] == no_test_df['predicted']).sum()

print(f"Correctly classified {correct_test} out of {len(no_test_df)} test emails ({correct_test / len(no_test_df) * 100}%)")
```

```
Correctly classified 13769 out of 15389 test emails (89.47300019494445%)
```

```
<ipython-input-44-14914959a31b>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  no_test_df.loc[:,'predicted'] = no_test_df['message'].apply(lambda x: classify_email(x, spam_word_probs, ham_word_probs, prior_spam, prior_ham))
```

In [45]:
```
#### correct_test in a dataframe
correct_df = no_test_df[no_test_df['classification'] == no_test_df['predicted']]
print("DataFrame of Correctly Classified Emails")
display(correct_df)
```

DataFrame of Correctly Classified Emails

	folder	file	message	classification	predicted_classification	predicted
19910	71	0	Where we can hesitantly derive perverse satisf...	1	1	1
19911	71	1	There are several things you can use to perfor...	0	0	0
19912	71	2	Best offer of the month:\n\nViggra - $76.95\nC...	1	1	1
19913	71	3	De i ar Home O h wne n r , \n \nYour cr v ed ...	1	1	1
19914	71	4	Special Offer\nAdobe Video Collection\nAdobe P...	1	1	1
...	...	...	...	...	...	...
35293	126	15	Genuine College Degree in 2 Weeks! \n\nHave yo...	1	1	1
35294	126	16	bla bla bla\neee\ne\n\n\n\n\n\n\nrererreerer...	1	1	1
35295	126	18	The OIL sector is going crazy. This is our wee...	1	1	1
35296	126	19	http://vdtobj.docscan.info/?23759301\n\nSuffer...	1	1	1
35297	126	20	U N I V E R S I T Y D I P L O M A S\n\nDo you...	1	1	1

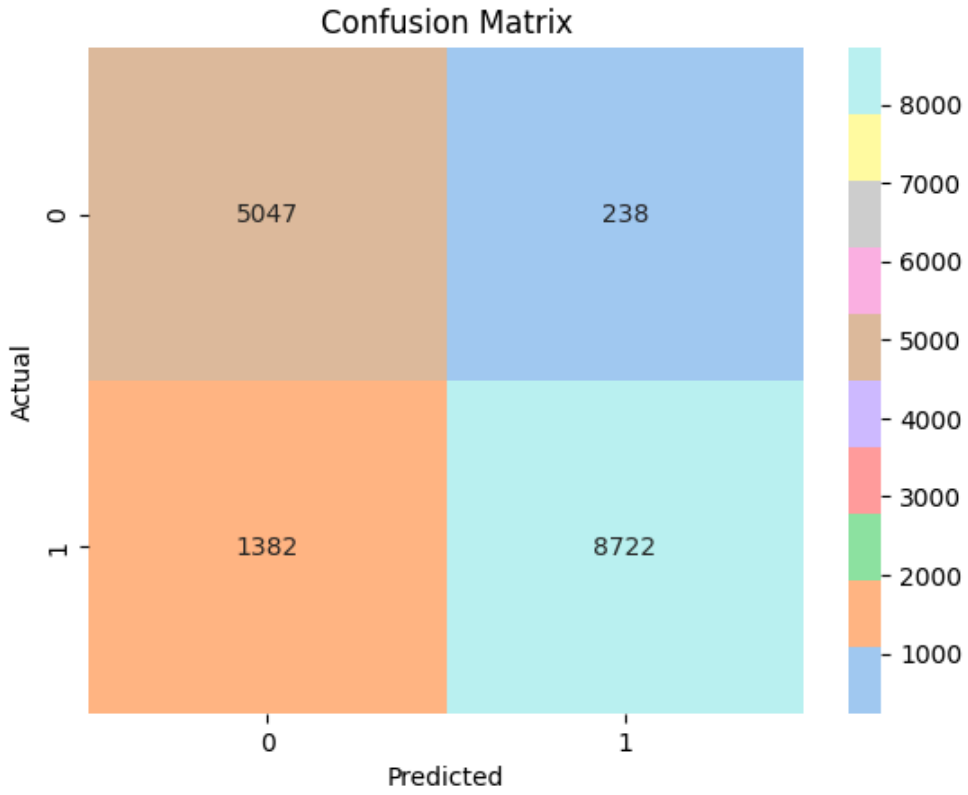
13769 rows × 6 columns

In [46]:
```
#### creating array of the actual and predicted classifications
actual = no_test_df['classification'].to_numpy()
predicted = no_test_df['predicted'].to_numpy()

from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(actual, predicted)
```

```
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap=sns.color_palette("pastel"))
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
```

Out[46]: Text(50.722222222222214, 0.5, 'Actual')



In [47]:
```
#### calculating accuracy, precision, recall
accuracy = accuracy_score(actual, predicted)
precision = precision_score(actual, predicted)
recall = recall_score(actual, predicted)

print(f"Accuracy = {accuracy}")
print(f"Precision = {precision}")
print(f"Recall = {recall}")
```

```
Accuracy = 0.8947300019494444
Precision = 0.9734375
Recall = 0.8632224861441014
```