Gliezel Ann Pajarilla CMSC 197 - Machine Learning

```
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
```

(A) Importing pre-processed data

Data were preprocessed by removing numbers, punctuations, stop words etc and exported into a csv file

```
In [2]: #### improrting pre-processed data
from google.colab import drive
drive.mount('/content/drive', force_remount = True)

preprocessed_path = '/content/drive/My Drive/FOURTH YEAR/Subjects/CMSC 197/trec06/preprocessed
```

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

```
In [3]: #### Loading the data in a dataframe
data = pd.read_csv(preprocessed_path)
data.head()
```

Out[3]:

	folder	file	message	classification
0	0	0	mailing list queried weeks ago running set arc...	0
1	0	1	luxury watches buy rolex rolex cartier bvlgari...	1
2	0	2	academic qualifications prestigious nonacc red...	1
3	0	3	greetings verify subscription planfans list ch...	0
4	0	4	chauncey conferred luscious continued tonsillitis	1

(B) Splitting to train and test set

Folders less than or equal to 70 were assigned to train and 70 above are assigned to test. The train_df was further split into either 0 = 'ham' and 1 = 'spam'.

```
In [4]: #### splitting the train and the test set
train_df = data[data['folder'] <= 70]
test_df = data[data['folder'] > 70]
```

```
train_ham_df = train_df[train_df['classification'] == 0]
train_spam_df = train_df[train_df['classification'] == 1]
```

In [5]:
```
#### checking the test size of the train and test
print('Train dataset size:', len(train_df))
print('Test dataset size:', len(test_df))
print('Train ham dataset size:', len(train_ham_df))
print('Train spam dataset size:', len(train_spam_df))
```

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

In [6]:
```
data
```

Out[6]:

	folder	file	message	classification
0	0	0	mailing list queried weeks ago running set arc...	0
1	0	1	luxury watches buy rolex rolex cartier bvlgari...	1
2	0	2	academic qualifications prestigious nonacc red...	1
3	0	3	greetings verify subscription planfans list ch...	0
4	0	4	chauncey conferred luscious continued tonsillitis	1
...	...	...	...	...
35294	126	16	bla bla bla eee rererreerer er	1
35295	126	18	oil sector going crazy weekly gift kkpt thing ...	1
35296	126	19	suffering pain depression heartburn well help ...	1
35297	126	20	prosperous future increased money earning powe...	1
35298	126	21	moat coverall cytochemistry planeload salk	1

35299 rows × 4 columns

In [7]:
```
#### cleaned with nan (empty) in the message
data = data.dropna()
data
```

Out[7]:

	folder	file	message	classification
0	0	0	mailing list queried weeks ago running set arc...	0
1	0	1	luxury watches buy rolex rolex cartier bvlgari...	1
2	0	2	academic qualifications prestigious nonacc red...	1
3	0	3	greetings verify subscription planfans list ch...	0
4	0	4	chauncey conferred luscious continued tonsillitis	1
...	...	...	...	...
35294	126	16	bla bla bla eee rererreerer er	1
35295	126	18	oil sector going crazy weekly gift kkpt thing ...	1
35296	126	19	suffering pain depression heartburn well help ...	1
35297	126	20	prosperous future increased money earning powe...	1
35298	126	21	moat coverall cytochemistry planeload salk	1

33727 rows × 4 columns

In [8]:
```
#### checking test df
train_spam_df
```

Out[8]:

	folder	file	message	classification
1	0	1	luxury watches buy rolex rolex cartier bvlgari...	1
2	0	2	academic qualifications prestigious nonacc red...	1
4	0	4	chauncey conferred luscious continued tonsillitis	1
7	0	7	nbc today □ body diet beaches magazines hollyw...	1
8	0	8	oil sector going crazy weekly gift kkpt thing ...	1
...	...	...	...	...
19904	70	294	txtadd	1
19905	70	295	スピード！簡単！ 無料！ 今どきの出会いの仕方ですね。 問 xinwallacom	1
19906	70	296	special offer adobe video collection adobe pre...	1
19907	70	297	doctype html public wcdtd html transitionalen ...	1
19909	70	299	suffering pain depression heartburn well help ...	1

12460 rows × 4 columns

In [9]:
```
#### checking train df
train_ham_df
```

Out[9]:

	folder	file	message	classification
0	0	0	mailing list queried weeks ago running set arc...	0
3	0	3	greetings verify subscription planfans list ch...	0
5	0	5	quiet quiet well straw poll plan running	0
6	0	6	working departed totally bell labs recommended...	0
10	0	10	greetings mass acknowledgement signed planfans...	0
...	...	...	...	...
19883	70	270	equation generate prime numbers equation theor...	0
19884	70	271	equation generate prime numbers equation theor...	0
19899	70	288	dear dmdx users guidance generating dmdx item ...	0
19903	70	293	built handyboard works great testmotor passes ...	0
19908	70	298	mounted isu infrared demodulator hb realised r...	0

7450 rows × 4 columns

Traversing through each file to count the occurences in order to obtain the top 10,000 most frequent words

In [10]:
```
#### Counting top 10000 words from the training dataset
word_counts = {}

for index, row in 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(train_spam_df), 10000))

for index in range(len(train_spam_df)):
    for word in str(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 [11]:
```
test_df
```

Out[11]:

	folder	file	message	classification
19910	71	0	hesitantly derive perverse satisfaction clodho...	1
19911	71	1	things perform experiment display will remain ...	0
19912	71	2	best offer month viggra ci ialis vaiium xa naa...	1
19913	71	3	de ar wne cr doesnt matter ow real st mmed ia ...	1
19914	71	4	special offer adobe video collection adobe pre...	1
...	...	...	...	...
35294	126	16	bla bla bla eee rererreerer er	1
35295	126	18	oil sector going crazy weekly gift kkpt thing ...	1
35296	126	19	suffering pain depression heartburn well help ...	1
35297	126	20	prosperous future increased money earning powe...	1
35298	126	21	moat coverall cytochemistry planeload salk	1

15389 rows × 4 columns

Creating the feature matrices

Returns in the dictionary the word and its key based on its frequency arranged in descending order.

In [12]:
```
#### creating word counts dictionary and get the top 10,000 words
from collections import Counter

word_counts = Counter(word for message in 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[12]:  {'will': 10440,
           'board': 4904,
           'price': 4549,
           'company': 4224,
           'adobe': 3902,
           'nil': 3762,
           'time': 3716,
           'email': 3597,
           'list': 3447,
           'dont': 3201,
           'program': 3069,
           'и': 3058,
           'help': 2882,
           'windows': 2782,
           'professional': 2769,
           'message': 2655,
           'gold': 2624,
           'work': 2557,
           'ms': 2553,
           'wrote': 2471,
           'subject': 2360,
           'received': 2265,
           'good': 2256,
           'handyboard': 2240,
           'problem': 2225,
           'office': 2213,
           'hb': 2198,
           'well': 2159,
           'info': 2084,
           'microsoft': 2074,
           'university': 2027,
           'в': 2014,
           'pro': 1993,
           'add': 1991,
           'studies': 1949,
           'womens': 1886,
           'bit': 1883,
           'number': 1870,
           'file': 1849,
           'code': 1847,
           'find': 1747,
           'news': 1737,
           'stock': 1737,
           'motor': 1728,
           'save': 1726,
           'power': 1725,
           'ic': 1725,
           'corp': 1694,
           'read': 1691,
           'contenttype': 1690,
           'people': 1664,
           'big': 1660,
           'current': 1659,
           'call': 1648,
           'best': 1644,
           'system': 1631,
           'handy': 1630,
           'today': 1624,
           'great': 1576,
           'pt': 1552,
           'china': 1550,
           'address': 1544,
           'oil': 1521,
           'ive': 1517,
           'week': 1496,
           'send': 1476,
           'free': 1471,
           'data': 1457,
           '|': 1454,
           'xp': 1441,
           'de': 1437,
           'reviews': 1435,
           'mail': 1425,
           'development': 1414,
           'days': 1411,
           'site': 1409,
           'port': 1395,
           'retail': 1389,
           'offer': 1386,
           'campaign': 1380,
           'money': 1377,
           'gas': 1377,
           'day': 1375,
           'better': 1370,
           'set': 1364,
           'rating': 1356,
           'life': 1339,
           'cart': 1302,
           'website': 1300,
           'software': 1293,
           'robot': 1281,
           'technology': 1265,
           'computer': 1262,
           'energy': 1259,
           'web': 1253,
           'textplain': 1251,
           'fax': 1240,
           'ra': 1224,
           'going': 1218,
           'market': 1206,
           'service': 1192,
           'nbsp': 1182,
           'special': 1170,
           'product': 1168,
           'cantex': 1162,
           'control': 1142,
           'version': 1139,
           'motors': 1134,
           'years': 1131,
           'order': 1123,
           'contenttransferencoding': 1119,
           'currently': 1109,
           'pm': 1097,
           'report': 1087,
           'project': 1086,
           'color': 1078,
           'things': 1069,
           'font': 1067,
           'problems': 1055,
           'windowtext': 1040,
           'thing': 1029,
           'digital': 1028,
           'works': 1026,
           'doesnt': 1025,
           'sender': 1018,
           'download': 1014,
           'serial': 1013,
           'start': 1012,
           'check': 1007,
           'full': 994,
           'potential': 986,
           'provide': 981,
           'video': 980,
           'contact': 980,
           'forward': 978,
           'based': 976,
           'experience': 970,
           'question': 969,
           'support': 968,
           'mimeversion': 966,
           'txtadd': 965,
           'plan': 960,
           'small': 957,
           'space': 955,
           'short': 949,
           'href': 949,
           'width': 947,
           'cant': 945,
           'business': 943,
           'receive': 937,
           'fast': 933,
           'running': 932,
           'account': 932,
           'apple': 929,
           'working': 928,
           '∽∽∽☆': 927,
           'design': 926,
           'original': 926,
           'replyto': 926,
           'dragon': 920,
           'online': 919,
           'low': 919,
           'sep': 912,
           'women': 909,
           'error': 907,
           'на': 901,
           'side': 899,
           'output': 895,
           'golden': 893,
           'high': 887,
           'real': 877,
           'sensor': 876,
           'second': 866,
           'science': 866,
           'input': 866,
           'long': 863,
           'effects': 863,
           'fred': 863,
           'interested': 856,
           'aug': 851,
           'change': 847,
           'programs': 846,
           'department': 846,
           'expansion': 843,
           'c': 841,
           'cs': 836,
           'services': 835,
           'place': 835,
           'point': 834,
           'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa': 829,
           'students': 817,
           'battery': 815,
           'files': 814,
           'light': 813,
           'по': 812,
           'ir': 812,
           'size': 810,
           'hope': 804,
           'year': 804,
           'charsetusascii': 802,
           'thu': 800,
           '《《': 800,
           'approved': 799,
           'servo': 793,
           'pin': 792,
           'test': 790,
           'note': 789,
           'including': 781,
           'fine': 774,
           'three': 769,
           'pine': 768,
           'investors': 765,
           'future': 762,
           'questions': 760,
           'sonar': 759,
           'phone': 758,
           'memory': 753,
           'chip': 752,
           'trading': 751,
           'nan': 749,
           'height': 748,
           'st': 747,
           'hello': 747,
           'charsetiso': 745,
           'pc': 744,
           'border': 744,
           'statements': 743,
           'times': 741,
           'mode': 740,
           'internet': 739,
           'technologies': 733,
           'class': 731,
           'include': 731,
           'mon': 730,
           'voltage': 721,
           'committee': 719,
           'exploration': 717,
           'companies': 716,
           'starting': 716,
           'box': 714,
           'course': 711,
           'prospect': 711,
           'hours': 709,
           'buy': 708,
           'write': 706,
           'monday': 699,
           'dear': 695,
           'pleased': 694,
           'analog': 694,
           '問) ': 693,
           'kind': 691,
           'lcd': 690,
           'access': 689,
           'idea': 689,
           'seismic': 688,
           'watch': 686,
           'store': 684,
           'release': 683,
           'private': 682,
           'systems': 681,
           'case': 680,
           'shipping': 677,
           'longer': 677,
           'complete': 675,
           'application': 673,
           'expect': 667,
           'limited': 664,
           'production': 659,
           'companys': 659,
           'men': 658,
           'sensors': 658,
           'prices': 654,
           'investment': 653,
           'international': 649,
           'group': 649,
           'click': 647,
           'form': 646,
           'drive': 643,
           'visit': 641,
           'hc': 641,
           'process': 638,
           'photoshop': 638,
           'lot': 637,
           'tue': 637,
           'px': 636,
           'early': 633,
           'feel': 633,
           'simply': 633,
           'rolex': 632,
           'opportunity': 631,
           'update': 628,
           'td': 628,
           'launch': 627,
           'fri': 626,
           'wrong': 623,
           'int': 621,
           'additional': 619,
           'share': 615,
           'third': 615,
           'corporation': 611,
           'shares': 609,
           'book': 609,
           'properties': 606,
           'body': 605,
           'solution': 605,
           'canyon': 603,
           'apr': 601,
           'answer': 600,
           'example': 600,
           'engineering': 600,
           'connected': 598,
           'faceverdana': 598,
           'mailing': 597,
           'weeks': 596,
           '】': 596,
           'messageid': 595,
           'main': 593,
           'trade': 590,
           'source': 590,
           'large': 590,
           'texas': 590,
           'link': 590,
           'turn': 589,
           'src': 586,
           'products': 584,
           'john': 583,
           'friday': 581,
           'sincerely': 581,
           'connect': 579,
           'industry': 578,
           'format': 577,
           'student': 576,
           '↓': 576,
           'expected': 575,
           'acrobat': 575,
           'led': 575,
           'swath': 574,
           'edition': 573,
           'build': 573,
           'issue': 572,
           'signal': 572,
           'san': 571,
           'return': 567,
           'span': 567,
           'field': 566,
           '◇': 566,
           'как': 565,
           'interest': 560,
           'simple': 559,
           'function': 559,
           'youll': 558,
           'type': 557,
           'easy': 556,
           'numbers': 555,
           'thought': 555,
           'handyboardmediamitedu': 555,
           'school': 554,
           'fact': 553,
           'collection': 552,
           'matter': 551,
           'jan': 551,
           'april': 550,
           'directly': 549,
           'ago': 547,
           'top': 547,
           'dmdx': 547,
           'ill': 546,
           'increase': 546,
           'helvetica': 546,
           'arial': 546,
           'left': 544,
           'premiere': 542,
           'style': 542,
           'play': 540,
           'fontsize': 539,
           'machine': 538,
           'interface': 538,
           'circuit': 538,
           'server': 536,
           '· · · ...……━━━━━━━━━━━━━━━━━━━━━━……....· · ·': 536,
           'independent': 535,
           'degree': 535,
           'details': 534,
           'sun': 532,
           'reading': 532,
           'speed': 532,
           'lose': 530,
           'reply': 530,
           'management': 528,
           'manager': 528,
           'esmtp': 526,
           'building': 524,
           'cost': 523,
           'didnt': 522,
           'hard': 521,
           'starshipdesign': 521,
           'choice': 519,
           'advance': 519,
           'general': 519,
           'device': 519,
           'close': 518,
           'president': 515,
           'huge': 514,
           'credit': 514,
           'customer': 513,
           'mike': 512,
           '□': 512,
           'mac': 511,
           'conference': 511,
           'cgdc': 510,
           'servos': 508,
           'usa': 507,
           'word': 505,
           'cpu': 504,
           'common': 503,
           'html': 503,
           'turned': 502,
           'weight': 502,
           'library': 502,
           'west': 502,
           'pdt': 502,
           'press': 502,
           'deal': 502,
           'processing': 501,
           'response': 500,
           'te': 500,
           'effective': 499,
           'display': 498,
           'luck': 496,
           'tuesday': 494,
           'remember': 491,
           'open': 490,
           'normal': 490,
           'shareholders': 489,
           'nov': 489,
           'advice': 488,
           'making': 488,
           'load': 488,
           'sansserif': 488,
           'move': 487,
           'jul': 486,
           'pretty': 485,
           'watches': 483,
           'red': 483,
           'center': 482,
           'gt': 482,
           'для': 481,
           'standard': 480,
           'announced': 479,
           'quality': 478,
           'american': 478,
           'members': 478,
           'total': 478,
           'bad': 476,
           'cwtd': 476,
           'meeting': 474,
           '☆': 474,
           'ports': 473,
           'communication': 473,
           'bank': 471,
           'driver': 470,
           'result': 469,
           'lines': 467,
           'table': 467,
           'risk': 466,
           'focus': 465,
           'position': 464,
           'director': 462,
           'os': 460,
           'messages': 460,
           'network': 460,
           'greko': 460,
           'sell': 459,
           'widthd': 458,
           '０円': 457,
           'valigndtop': 456,
           'local': 455,
           'lego': 455,
           'supply': 454,
           'called': 451,
           'pay': 451,
           'major': 450,
           'happy': 448,
           'boards': 448,
           'orgasms': 447,
           'securities': 447,
           'havent': 446,
           'イリュージョン': 446,
           'parts': 446,
           'loss': 445,
           'drilling': 444,
           'pr': 443,
           'media': 443,
           'college': 442,
           'required': 442,
           'ross': 442,
           'copy': 441,
           'bestsellers': 441,
           'continue': 441,
           'loan': 441,
           'dvd': 438,
           'resources': 437,
           'range': 437,
           'mhz': 437,
           'person': 436,
           'appreciated': 436,
           'verified': 436,
           'fontfamily': 436,
           'request': 435,
           'discussion': 434,
           'encore': 433,
           'dc': 433,
           'mar': 433,
           'jonathan': 433,
           'term': 432,
           'audition': 432,
           'month': 432,
           'dec': 432,
           'pins': 432,
           'interactive': 432,
           'styledfontsizeptmsobidifontsizept': 432,
           'needed': 431,
           'exchange': 431,
           'wondering': 431,
           'img': 430,
           'coming': 429,
           'legal': 427,
           'correct': 427,
           'darkwinguoregonedu': 427,
           'antonio': 426,
           'clixme': 426,
           'nbspspan': 426,
           'cc': 425,
           'hardware': 423,
           'rate': 423,
           'paper': 421,
           'tel': 421,
           'started': 420,
           'issues': 419,
           'card': 418,
           'pcode': 417,
           'delivery': 416,
           'smtp': 416,
           'friend': 416,
           'unit': 416,
           'books': 414,
           'cash': 413,
           'screen': 413,
           'trace': 412,
           'august': 412,
           'reference': 411,
           'team': 411,
           'command': 409,
           'series': 409,
           'understand': 408,
           'stuff': 408,
           'val': 408,
           'programming': 408,
           'language': 407,
           'country': 406,
           'franklin': 405,
           'suggestions': 405,
           'cable': 405,
           'global': 404,
           'mime': 402,
           'unique': 402,
           'interesting': 401,
           'knowledge': 401,
           'winning': 401,
           'room': 401,
           'phd': 401,
           '∞───────────────────∞': 400,
           'allow': 399,
           'user': 398,
           'talk': 397,
           'inreplyto': 397,
           'text': 396,
           'growth': 396,
           'written': 396,
           'included': 395,
           'create': 393,
           'north': 392,
           'acres': 392,
           'letter': 392,
           'changes': 391,
           'ideas': 391,
           'green': 388,
           'highly': 387,
           'opportunities': 387,
           'formula': 387,
           'post': 386,
           'engaged': 386,
           'martin': 386,
           'key': 385,
           'manual': 385,
           'register': 384,
           '张': 384,
           'enjoy': 383,
           'true': 383,
           'hot': 382,
           'july': 381,
           'love': 379,
           'sex': 379,
           'interrog': 379,
           'item': 378,
           'lottery': 378,
           'не': 378,
           'charge': 378,
           'a': 377,
           'offers': 377,
           'health': 377,
           'producer': 376,
           'funds': 375,
           'что': 375,
           'jp': 375,
           'job': 374,
           'couple': 374,
           'inform': 374,
           'soft': 374,
           'south': 373,
           'held': 373,
           'ability': 373,
           'graduate': 373,
           'texthtml': 372,
           'mineral': 372,
           'water': 372,
           'writes': 372,
           'march': 371,
           'shareholder': 370,
           'ground': 370,
           'volume': 370,
           'cialis': 368,
           'record': 366,
           'multiple': 366,
           'claims': 366,
           'chips': 366,
           'penis': 365,
           'outputs': 364,
           'history': 363,
           'wont': 363,
           'announces': 362,
           'man': 362,
           'nice': 362,
           'difficult': 362,
           'la': 361,
           '■': 361,
           'headquartered': 360,
           'confirm': 359,
           'tests': 359,
           'degrees': 359,
           'purchase': 358,
           'area': 357,
           'max': 357,
           'routines': 357,
           'initial': 356,
           'reserves': 355,
           'isnt': 355,
           'house': 355,
           'ready': 355,
           'model': 354,
           'applications': 354,
           'licensed': 354,
           'prize': 353,
           'sat': 352,
           'radio': 352,
           'david': 351,
           'worked': 350,
           'returns': 350,
           'sound': 350,
           'undertaken': 348,
           'single': 348,
           'gapj': 348,
           'events': 348,
           'geophysical': 347,
           'fully': 347,
           'values': 347,
           'switch': 346,
           'membership': 346,
           'marketing': 346,
           'url': 346,
           'personal': 345,
           'void': 345,
           'addresses': 344,
           'minutes': 344,
           'methods': 344,
           'successful': 343,
           'split': 342,
           'names': 342,
           'natural': 342,
           'completed': 341,
           'profile': 341,
           'confidentiality': 340,
           'dr': 340,
           'exactly': 340,
           'cd': 340,
           'search': 340,
           'draw': 339,
           'edt': 339,
           'national': 339,
           'resistor': 339,
           'asked': 338,
           'ctxe': 338,
           'changed': 337,
           'feb': 336,
           'users': 336,
           'pounds': 336,
           'sharp': 336,
           'assembly': 335,
           'electronics': 335,
           'polaroid': 335,
           'advanced': 334,
           'inside': 334,
           'ram': 334,
           'xmailer': 334,
           'multipart': 332,
           'public': 331,
           'directory': 331,
           'designed': 331,
           'reason': 331,
           'built': 330,
           'avoid': 330,
           'head': 330,
           'respect': 330,
           'amount': 329,
           'writing': 328,
           'mobile': 328,
           'operations': 328,
           'located': 327,
           'option': 327,
           'pulse': 327,
           'consider': 326,
           'claim': 326,
           'todays': 326,
           'rest': 326,
           'chance': 326,
           'remove': 326,
           'security': 325,
           'papers': 324,
           'phase': 323,
           'june': 322,
           'worlds': 321,
           'bilbo': 321,
           'managed': 321,
           'moving': 321,
           'cheap': 321,
           'tools': 320,
           'verde': 320,
           'ー': 320,
           '写真有り': 320,
           'reset': 319,
           'linux': 318,
           'cubs': 318,
           'ma': 318,
           'statement': 316,
           'вы': 316,
           'robotics': 316,
           'picture': 315,
           'java': 315,
           'stepper': 315,
           'takes': 314,
           'visa': 314,
           'reported': 313,
           'terms': 313,
           'helps': 313,
           'agreement': 313,
           'comments': 313,
           'o': 312,
           'operating': 312,
           'functions': 312,
           'secure': 312,
           'care': 312,
           'bytes': 312,
           'category': 311,
           'providence': 311,
           'america': 311,
           'distance': 311,
           'pick': 310,
           'setting': 308,
           'connection': 308,
           'canadian': 308,
           'opt': 307,
           'youve': 307,
           'base': 307,
           'bulk': 307,
           'engine': 307,
           'clock': 307,
           'worldwide': 306,
           'extra': 306,
           'materials': 306,
           'rat': 306,
           'voice': 305,
           'batteries': 305,
           'guess': 304,
           'finally': 304,
           'tv': 304,
           'thursday': 303,
           'от': 303,
           'relief': 302,
           'sexual': 302,
           'routine': 302,
           'feminist': 302,
           'minerals': 301,
           'wide': 301,
           'arizona': 301,
           'level': 300,
           'sperm': 300,
           'ii': 299,
           'step': 299,
           'formatflowed': 298,
           'gpd': 298,
           'higher': 297,
           'interrupt': 297,
           'happened': 296,
           'fall': 296,
           'black': 296,
           'paddingin': 296,
           'ptheightin': 296,
           'advantage': 295,
           'ascii': 295,
           'inputs': 295,
           'article': 294,
           'downloaded': 294,
           '・・・・・・・・・・【無料】': 294,
           'car': 293,
           'monthly': 293,
           'material': 292,
           'wells': 292,
           'resrt': 292,
           'sector': 291,
           'figure': 291,
           'sales': 291,
           'bbb': 290,
           'social': 290,
           'unsubscribe': 290,
           '▼': 290,
           'wall': 289,
           'wait': 288,
           'header': 288,
           'difference': 288,
           'cut': 288,
           'styledwidthptbordertopnonebord': 288,
           'erleft': 288,
           'noneborderbottomsolid': 288,
           'ptborderrightsolid': 288,
           'msobordertopaltsolid': 288,
           'ptmsoborderleftaltsolid': 288,
           'windo': 288,
           'wtext': 288,
           'technical': 287,
           'target': 287,
           'registration': 287,
           'notice': 286,
           'attached': 286,
           'completion': 286,
           'groups': 285,
           'earning': 285,
           'structure': 285,
           'developing': 284,
           'financial': 284,
           'wanted': 284,
           'earth': 284,
           'ribbon': 284,
           'classes': 283,
           'tomorrow': 283,
           'annual': 283,
           'staff': 283,
           'ceo': 283,
           'meaning': 283,
           'wire': 283,
           'michael': 282,
           'rich': 282,
           'night': 282,
           'mention': 281,
           'fixed': 281,
           'external': 281,
           'rework': 280,
           'win': 280,
           'pack': 280,
           'sort': 279,
           'meds': 279,
           'appreciate': 279,
           'kit': 279,
           'за': 278,
           'worry': 278,
           'bill': 278,
           'cellspacing': 278,
           'analysis': 278,
           'remote': 277,
           'documentation': 277,
           'precedence': 277,
           'encoder': 277,
           'follow': 276,
           'dollars': 276,
           'bring': 276,
           'andor': 276,
           'method': 276,
           'module': 276,
           'specific': 276,
           'september': 275,
           'decision': 275,
           'star': 275,
           'mother': 274,
           'final': 274,
           'cellpadding': 274,
           'heard': 273,
           'pwm': 273,
           'view': 272,
           '━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━': 272,
           'lowest': 271,
           'suite': 271,
           'jump': 271,
           'institute': 271,
           'brought': 271,
           'mind': 270,
           'components': 270,
           'en': 270,
           'downloading': 270,
           'provided': 269,
           'joint': 269,
           'unix': 268,
           'party': 268,
           'ext': 268,
           'lost': 267,
           'review': 267,
           'learn': 267,
           'instructions': 267,
           'roman': 266,
           'trouble': 266,
           'foreign': 266,
           'customers': 266,
           'direction': 265,
           'placement': 265,
           'robots': 265,
           'symbol': 264,
           'generate': 264,
           'inches': 264,
           'controller': 264,
           'phoenix': 264,
           'clear': 264,
           'lower': 264,
           'projects': 263,
           'errors': 263,
           'assured': 262,
           'κ': 262,
           'jcf': 262,
           'thinking': 261,
           'pain': 261,
           'mine': 261,
           'community': 261,
           'largest': 260,
           'worldclass': 260,
           'listed': 260,
           'agent': 260,
           'premium': 260,
           'returnpath': 260,
           'direct': 260,
           'success': 260,
           'pill': 259,
           'white': 259,
           'contentlength': 259,
           '∽': 259,
           'underway': 258,
           'human': 258,
           'late': 258,
           'receiving': 258,
           '会': 258,
           'ephedra': 258,
           'measure': 258,
           'libraries': 257,
           'air': 257,
           'aa': 257,
           'masters': 257,
           'vm': 257,
           'kellystaolcom': 257,
           'hand': 256,
           'dept': 256,
           'earlier': 256,
           'commence': 255,
           'union': 255,
           'professor': 255,
           'increased': 255,
           'geological': 254,
           'purpose': 254,
           'mit': 254,
           'street': 254,
           'electronic': 254,
           'compiler': 253,
           'travel': 253,
           'io': 252,
           'food': 252,
           'gain': 252,
           'config': 252,
           'infinex': 252,
           'plasma': 252,
           'cm': 251,
           'basic': 251,
           'communications': 251,
           'tabs': 251,
           '⇒': 251,
           'eric': 250,
           ...}
```

Sparse matrix representation of the word displayed in its position

In [13]:
```python
## sparse matrix
messages_split = 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[13]:

	0	1	2	3	4	5	6	7	8
0	mailing	list	queried	weeks	ago	running	set	archive	server
1	luxury	watches	buy	rolex	rolex	cartier	bvlgari	frank	muller
2	academic	qualifications	prestigious	nonacc	redited	uni	versities	knowledge	experience
3	greetings	verify	subscription	planfans	list	charter	members	signed	day
4	chauncey	conferred	luscious	continued	tonsillitis	None	None	None	None

5 rows × 127 columns

(C) Feature Matrix Ham & Spam Messages

(a) ham with 10 000 columns that matches the 10 000 words where 0 and 1 refers to the presence or absence of the word

In [14]:
```python
#### initializing feature matrix for the ham
feature_matrix_ham = np.zeros((len(train_ham_df), len(top_10000_words)), dtype=int)
top_10000_words_list = list(top_10000_words.keys())

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

feature_matrix_ham
```

```
Out[14]: array([[1, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                ...,
                [1, 0, 0, ..., 0, 0, 0],
                [1, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0]])
```

(b) spam with 10 000 columns that matches the 10 000 words where 0 and 1 refers to the presence or absence of the word

```
In [15]: #### initializes feature matrix for spam
feature_matrix_spam = np.zeros((len(train_spam_df), len(top_10000_words)), dtype=int)
top_10000_words_list = list(top_10000_words.keys())

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

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


feature_matrix_spam
```

```
Out[15]: array([[0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                ...,
                [0, 0, 1, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0]])
```

(D) Computing the Priors

Where (1) train_spam_df is the number of spam emails in the training set, (2) train_ham_df is the number of ham emails in the training set,(3) train_df is the total number of emails (contains both ham and spam messages)

```
In [16]: #### Calculate prior probabilities for spam and ham
prior_spam = len(train_spam_df) / len(train_df)
prior_ham = len(train_ham_df) / len(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
```

(E) Computing the Likelihood of each word

applying to spam and ham classification with Laplace smoothing

```
In [19]: #### function for laplace smoothing
def laplace_smoothing(feature_matrix_spam, feature_matrix_ham, laplace_smoothing_val, num_clas
    prob_word_given_spam = np.zeros(len(top_10000_words))
    prob_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)):
        prob_word_given_spam[i] = (spam_word_count[i] + laplace_smoothing_val) / (total_spam_w
        prob_word_given_ham[i] = (ham_word_count[i] + laplace_smoothing_val) / (total_ham_word

    return prob_word_given_spam, prob_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(feature_matrix_spam, feature_matrix_ham, l
```

In [20]:
```
#### 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: [5.24104624e-03 5.39628174e-04 3.41025294e-03 ...
2.71046115e-05
 2.46405559e-06 2.46405559e-06]
Likelihood of a word being in a ham email: [6.56481019e-03 5.79756964e-03 3.69198464e-04 ... 2.
30749040e-05
 5.76872601e-05 4.03810820e-05]
```

In [21]:
```
## 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[21]:

	Word	P(Word|Spam)	P(Word|Ham)
0	will	0.005241	0.006565
1	board	0.000540	0.005798
2	price	0.003410	0.000369
3	company	0.003073	0.000436
4	adobe	0.001286	0.000017
5	nil	0.000002	0.000069
6	time	0.002306	0.003784
7	email	0.001252	0.004067
8	list	0.000310	0.002861
9	dont	0.002341	0.004214
10	program	0.001175	0.002870
11	и	0.000670	0.000003
12	help	0.001698	0.004364
13	windows	0.001333	0.001157
14	professional	0.001762	0.000101
15	message	0.001074	0.003779
16	gold	0.001042	0.000156
17	work	0.000468	0.004107
18	ms	0.001087	0.000666
19	wrote	0.000237	0.005241

Classifying the emails

In [25]:
```
#### classifying the emails using the computed probabilities
def classify_email(email, spam_word_probs, ham_word_probs, prob_spam, prob_ham):
    log_prob_spam = 0
    log_prob_ham = 0

    words = str(email).split()

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

    log_prob_spam += np.log(prob_spam)
    log_prob_ham += np.log(prob_ham)

    return 1 if log_prob_spam > log_prob_ham else 0
```

In [27]:
```
## tabular form the df in classifying email [actual and predicted]
test_df['predicted_classification'] = test_df['message'].apply(lambda x: classify_email(x, spa
classification_results_df = pd.DataFrame({
    'Message': test_df['message'],
    'Actual Classification': test_df['classification'],
    'Predicted Classification': test_df['predicted_classification']
})

classification_results_df.head(20)
```

```
<ipython-input-27-65cad345ddda>: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/i
ndexing.html#returning-a-view-versus-a-copy
  test_df['predicted_classification'] = test_df['message'].apply(lambda x: classify_email(x, sp
am_word_probs, ham_word_probs, prior_spam, prior_ham))
```

Out[27]:

	Message	Actual Classification	Predicted Classification
19910	hesitantly derive perverse satisfaction clodho...	1	1
19911	things perform experiment display will remain ...	0	0
19912	best offer month viggra ci ialis vaiium xa naa...	1	1
19913	de ar wne cr doesnt matter ow real st mmed ia ...	1	1
19914	special offer adobe video collection adobe pre...	1	1
19915	multipart message mime format dragon contentty...	1	1
19916	txtadd	1	1
19917	mistersporty incorporation rambrantplein ad de...	1	1
19918	choice best choice drugs viagra pill viagra so...	1	1
19919	ive changed dmdx listserv subject filter hopef...	0	0
19920	noticed documentation inputoutput pio possibil...	0	0
19921	putting nback experiment quick question list s...	0	0
19922	txtadd	1	1
19923	pm wrote noticed documentation inputoutput pio...	0	0
19924	pm wrote putting nback experiment quick questi...	0	0
19925	appears message cut email received list previo...	0	0
19926	input duplicate post john john curtin phd depa...	0	0
19927	set assistant prof weeks dont huge budget pay ...	0	0
19928	dear homeowner approved house loan fixed offer...	1	1
19929	discounted quality secure free gifts free worl...	1	0

Testing the classifier

Test Set

```
In [29]: #### classify the test emails
         test_df.loc[:,'predicted'] = test_df['message'].apply(lambda x: classify_email(x, spam_word_pr
         correct_test = (test_df['classification'] == test_df['predicted']).sum()
         incorrect_test = len(test_df) - correct_test

         print(f'Correctly classified emails ({correct_test / len(test_df) * 100}%)')
         print(f'Incorrectly classified emails ({incorrect_test / len(test_df) * 100}%)')
```

```
Correctly classified emails (93.42430149447694%)
Incorrectly classified emails (6.575698505523067%)
```

```
<ipython-input-29-1dba59752d99>: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/i
ndexing.html#returning-a-view-versus-a-copy
  test_df.loc[:,'predicted'] = test_df['message'].apply(lambda x: classify_email(x, spam_word_p
robs, ham_word_probs, prior_spam, prior_ham))
```

```
In [30]: #### correct_test in a dataframe
         correct_df = test_df[test_df['classification'] == 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	0.0	hesitantly derive perverse satisfaction clodho...	1.0	1.0	1
19911	71.0	1.0	things perform experiment display will remain ...	0.0	0.0	0
19912	71.0	2.0	best offer month viggra ci ialis vaiium xa naa...	1.0	1.0	1
19913	71.0	3.0	de ar wne cr doesnt matter ow real st mmed ia ...	1.0	1.0	1
19914	71.0	4.0	special offer adobe video collection adobe pre...	1.0	1.0	1
...	...	...	...	...	...	...
35294	126.0	16.0	bla bla bla eee rererreerer er	1.0	1.0	1
35295	126.0	18.0	oil sector going crazy weekly gift kkpt thing ...	1.0	1.0	1
35296	126.0	19.0	suffering pain depression heartburn well help ...	1.0	1.0	1
35297	126.0	20.0	prosperous future increased money earning powe...	1.0	1.0	1
35298	126.0	21.0	moat coverall cytochemistry planeload salk	1.0	1.0	1

14378 rows × 6 columns

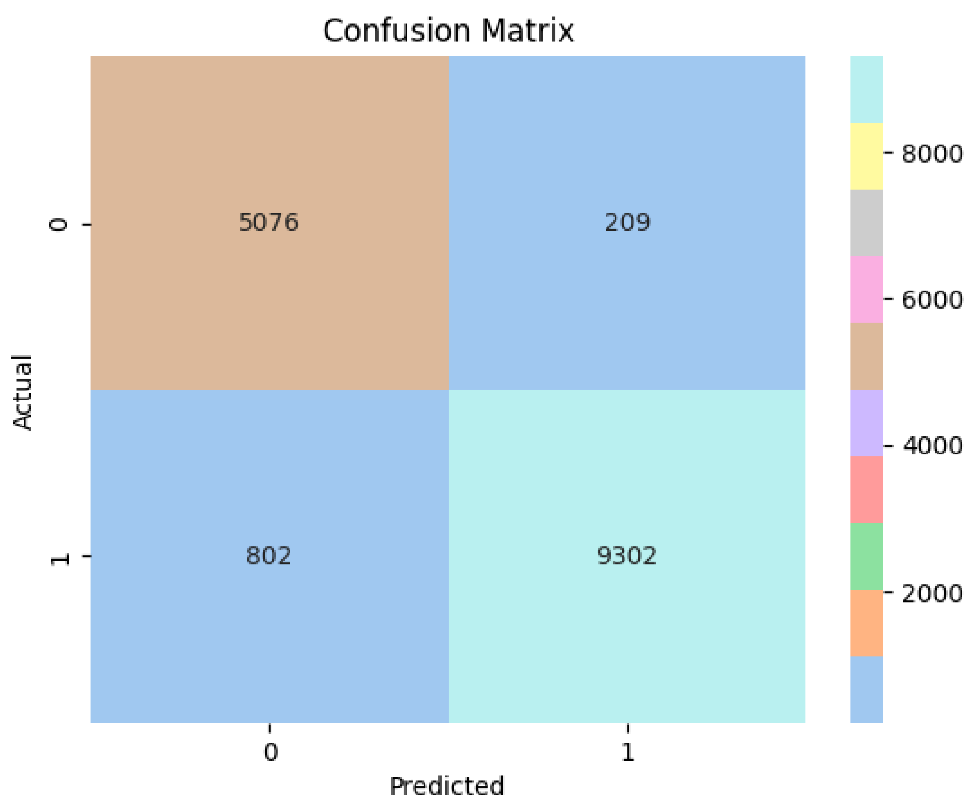
Performance evaluation

In [33]:
```
print(test_df['classification'].isnull().sum())
test_df = test_df.dropna(subset=['classification'])
```

```
1
```

In [34]:
```
#### creating array of the actual and predicted classifications
actual = test_df['classification'].to_numpy()
predicted = 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')
```

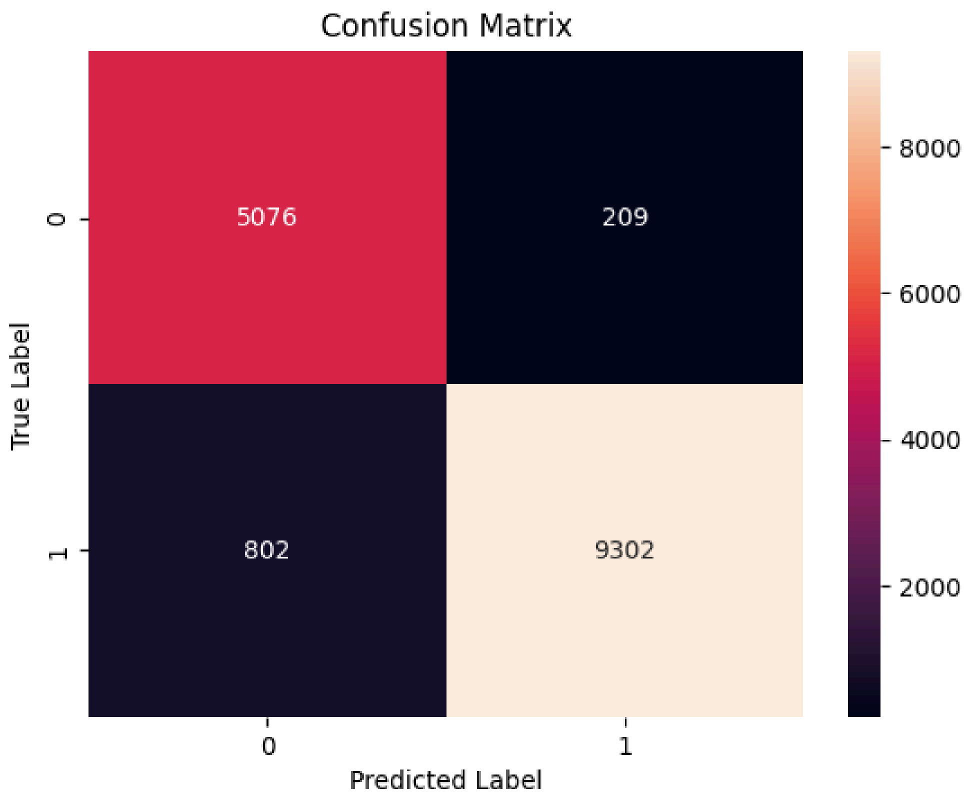
Out[34]:
```
Text(50.722222222222214, 0.5, 'Actual')
```



```
In [35]: #### 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 = 0.93430372343882
Precision = 0.9780254442224793
Recall = 0.9206254948535234
```

```
In [36]: #### false positives, false negative, true positive, true negative
         actual = np.array(test_df['classification'])
         predicted = np.array(test_df['predicted'])
         confusion_matrix = metrics.confusion_matrix(actual, predicted, labels = [0, 1])
         sns.heatmap(confusion_matrix, annot = True, fmt = 'd')
         plt.title('Confusion Matrix')
         plt.xlabel('Predicted Label')
         plt.ylabel('True Label')
         plt.show()

         print('True Negative Rate (TN) - {}'.format(confusion_matrix[0][0]))
         print('False Positive Rate (FP) - {}'.format(confusion_matrix[0][1]))
         print('False Negative Rate (FN) - {}'.format(confusion_matrix[1][0]))
         print('True Positive Rate (TP) - {}'.format(confusion_matrix[1][1]))
```



```
True Negative Rate (TN) - 5076
False Positive Rate (FP) - 209
False Negative Rate (FN) - 802
True Positive Rate (TP) - 9302
```

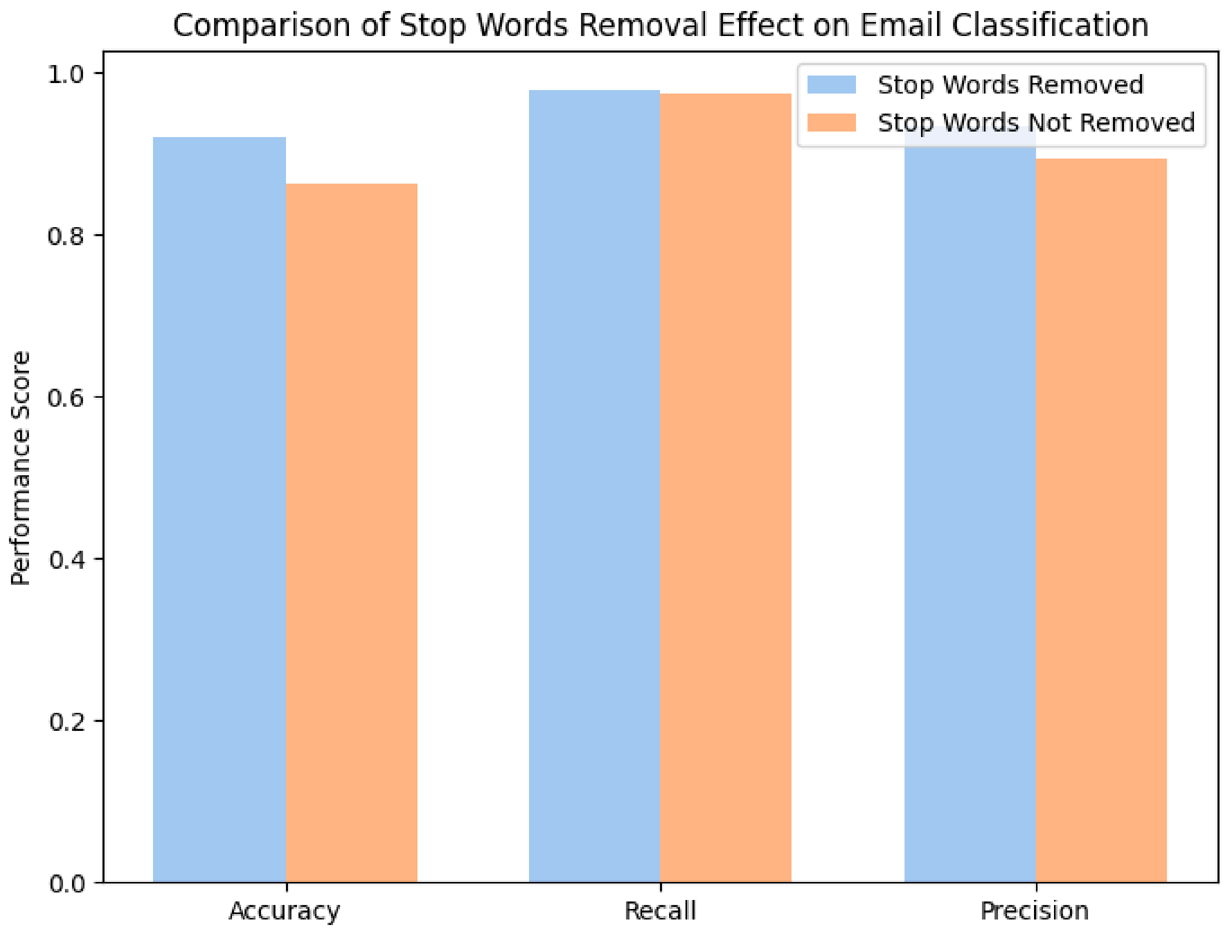
True positive rate was the highest, followed by true negative rates and then false negative rate and false positive rate. High TPR indicates that the model prioritizes classifying spam messages eventhough some ham emails may be misclassified.

(Q1). What is the effect of removing stop words in terms of precision, recall, and accuracy? Show a plot or a table of these results.

The csv obtained was from the original without using functions for cleaning such as removing the stop words. After, done the same method with the processes of obtaining accuracy, precision, and recall

In [37]:
```
#### utilized stop and those did not use the stop words
#Accuracy = 0.8947300019494444
#Precision = 0.9734375
#Recall = 0.8632224861441014

accuray_non_stopwords = 0.8947300019494444
recall_non_stopwords = 0.9734375
precision_non_stopwords = 0.8632224861441014

accuracy_stopwords = 0.93430372343882
recall_stopwords = 0.9780254442224793
precision_stopwords = 0.9206254948535234
```

```
In [38]: #### visualizing precision, recall and accuracy
precision_removed_stopwords = 0.93430372343882
recall_removed_stopwords = 0.9780254442224793
accuracy_removed_stopwords = 0.9206254948535234

precision_notremoved_stopwords = 0.8947300019494444
recall_notremoved_stopwords = 0.9734375
accuracy_notremoved_stopwords = 0.8632224861441014

stop_words_performance = {
    'Accuracy': accuracy_removed_stopwords,
    'Recall': recall_removed_stopwords,
    'Precision': precision_removed_stopwords
}

non_stop_words_performance = {
    'Accuracy': accuracy_notremoved_stopwords,
    'Recall': recall_notremoved_stopwords,
    'Precision': precision_notremoved_stopwords
}

metrics = ['Accuracy', 'Recall', 'Precision']
stop_words_values = [stop_words_performance[metric] for metric in metrics]
non_stop_words_values = [non_stop_words_performance[metric] for metric in metrics]

x = range(len(metrics))

plt.figure(figsize=(8, 6))

## plotting
plt.bar(x, stop_words_values, width=0.35, align='center', label='Stop Words Removed', color= s
plt.bar([i + 0.35 for i in x], non_stop_words_values, width=0.35, align='center', label='Stop

plt.xticks([i + 0.35 / 2 for i in x], metrics)
plt.ylabel('Performance Score')
plt.title('Comparison of Stop Words Removal Effect on Email Classification')
plt.legend()
plt.show()
```



Accuracy, recall and precision was highest when the stop words were removed than it was not removed. When stop words were not removed, there are possible noises in the data and would populate the feature matrix with the words like a, able, about which may not always carry significant meaning. Thus, removing it makes it much better for the model to focus more informative words that may contribute for it being a spam or ham

Q2. Experiment on the number of words used for training. Filter the dictionary to include only words occurring more than k times (1000 words, then k > 100, and k= 50 times). For example, the word “offer” appears 150 times, that means that it will be included in the dictionary.

In [39]:
```
#### experimenting and filtering for k = 50
k = 50
sorted_dict = dict(top_10000_words)
sorted_dict

filtered_dict_50 = {x: y for x, y in sorted_dict.items() if y > k}
filtered_dict_50_list = list(filtered_dict_50.keys())
print(f'Filtered Dictionary (k=50): {len(filtered_dict_50)}')
```

```
Filtered Dictionary (k=50): 4614
```

In [40]:
```
#### experimenting and filtering for k = 100
k = 100
sorted_dict = dict(top_10000_words)
sorted_dict

filtered_dict_100 = {x: y for x, y in sorted_dict.items() if y > k}
filtered_dict_100_list = list(filtered_dict_100.keys())
print(f'Filtered Dictionary (k=100): {len(filtered_dict_100)}')
```

```
Filtered Dictionary (k=100): 2580
```

As the k values (minimum frequency threshold) was increased, it would result to having fewer words that appeared more frequently. In order to determine the specific threshold to improve the performance, one must experiment in the k value to filter out most frequent words that appeared n times.

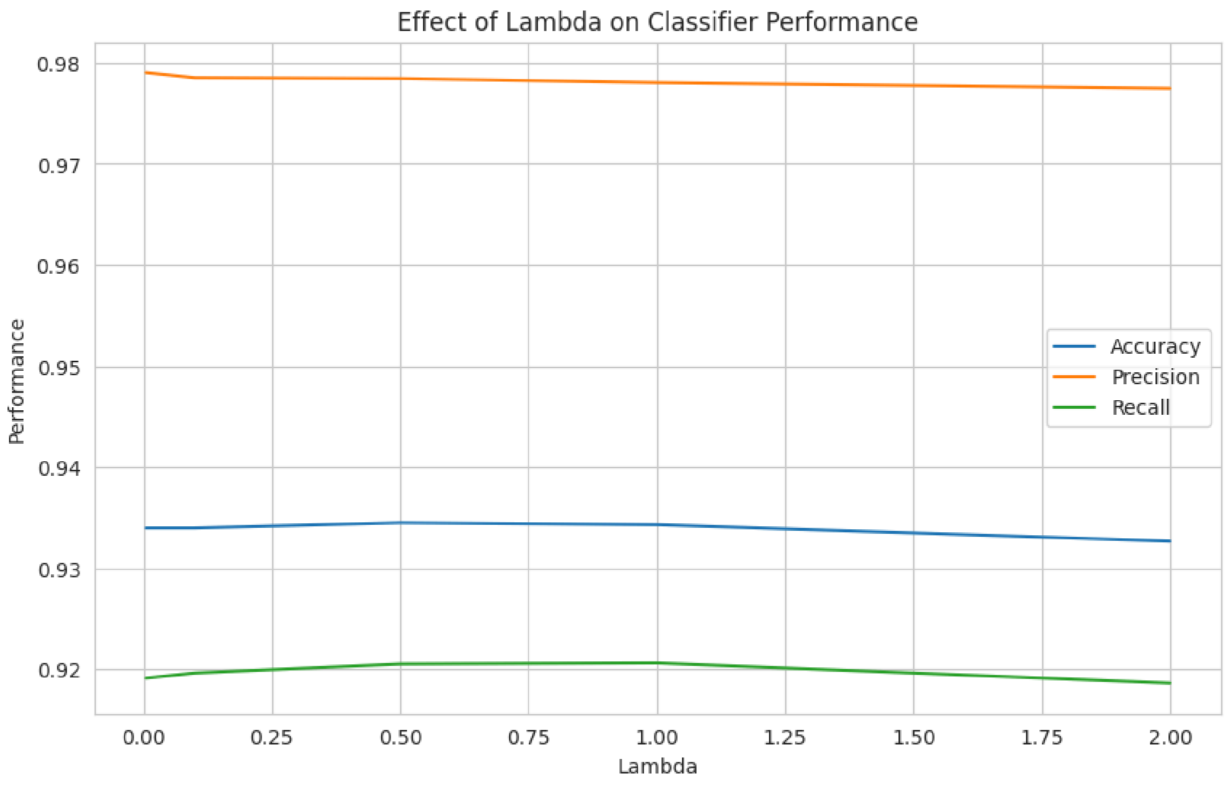
Q3. Discuss the results of the different parameters used for Lambda smoothing. Test it on 5 varying values of the λ (e.g. λ = 2.0, 1.0, 0.5, 0.1, 0.005), Evaluate performance metrics for each.

In [41]:
```
#### using the different lambda values
lambda_values = [2.0, 1.0, 0.5, 0.1, 0.005]

test_df_lambda = test_df.copy()
if 'predicted' in test_df_l0p1.columns:
  test_df_lambda = test_df_lambda.drop('predicted', axis=1)

results = []

## iterate through different lambda values
for laplace_smoothing_val in lambda_values:
  spam_word_probs, ham_word_probs = laplace_smoothing(feature_matrix_spam, feature_matrix_ham,
  test_df_lambda['predicted'] = test_df_lambda['message'].apply(lambda x: classify_email(x, sp

  ## calculating metrics
  actual = test_df_lambda['classification'].to_numpy()
  predicted = test_df_lambda['predicted'].to_numpy()

  accuracy = accuracy_score(actual, predicted)
  precision = precision_score(actual, predicted)
  recall = recall_score(actual, predicted)


  results.append({'lambda': laplace_smoothing_val, 'accuracy': accuracy, 'precision': precisio


results_df = pd.DataFrame(results)
results_df
```

Out[41]:

	lambda	accuracy	precision	recall
0	2.000	0.932679	0.977464	0.918646
1	1.000	0.934304	0.978025	0.920625
2	0.500	0.934499	0.978435	0.920527
3	0.100	0.933979	0.978517	0.919636
4	0.005	0.933979	0.979022	0.919141

In [50]:
```
#### plotting the relationship between lambda and performance metrics
plt.figure(figsize=(10, 6))
plt.plot(results_df['lambda'], results_df['accuracy'], label='Accuracy')
plt.plot(results_df['lambda'], results_df['precision'], label='Precision')
plt.plot(results_df['lambda'], results_df['recall'], label='Recall')
plt.xlabel('Lambda')
plt.ylabel('Performance')
plt.title('Effect of Lambda on Classifier Performance')
plt.legend()
plt.show()
```

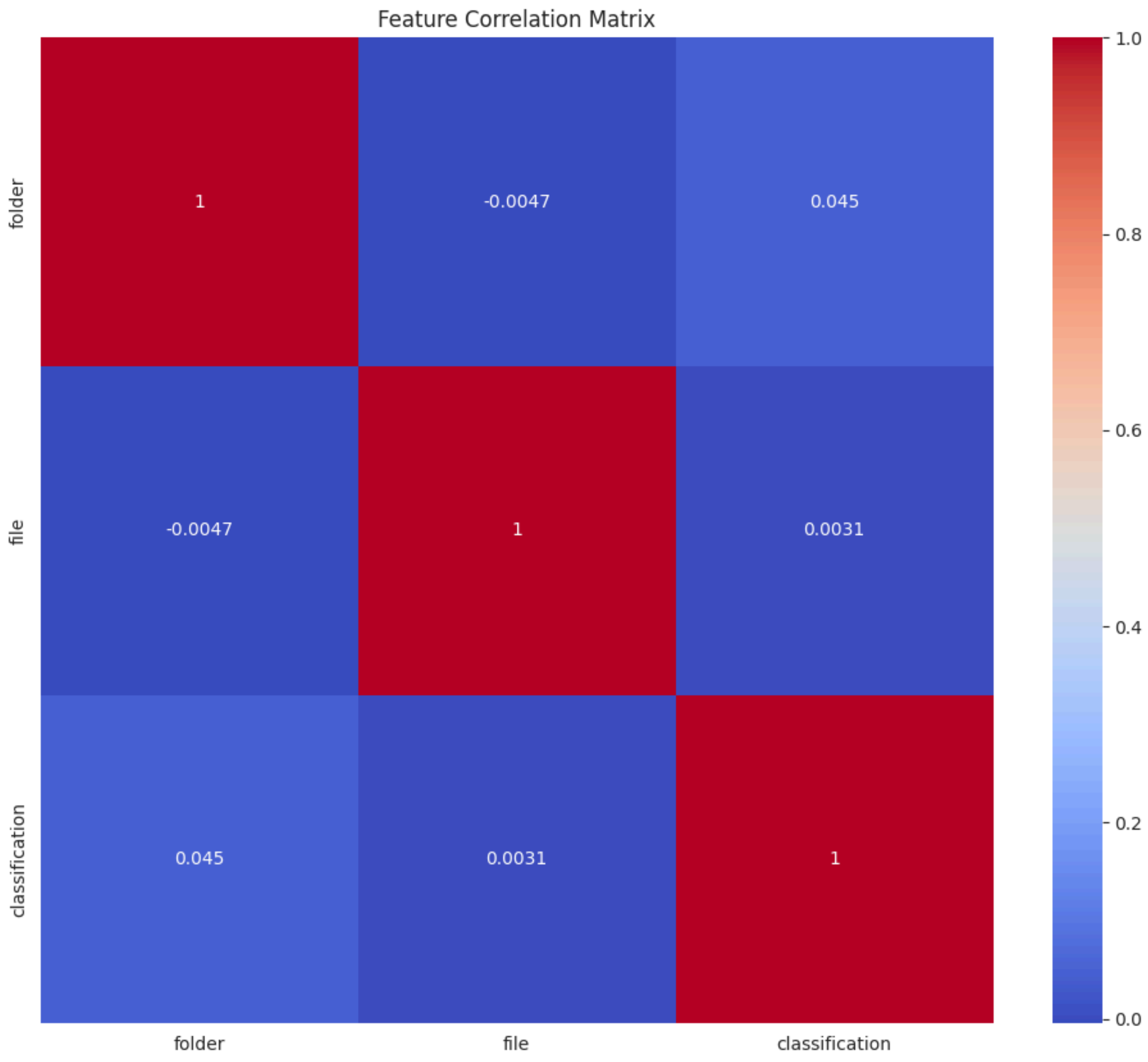


With these, the precision was the highest followed by accuracy and lastly recall. Thus, the model was able to determine a certain email or message as spam which means that there are few false positives or the ones that incorrectly classifies ham emails as spam.

Visualizations

Visualizing the features

```
In [53]: data_numeric = data.select_dtypes(include=[np.number])
         data_numeric = data_numeric.dropna()

         ## plot the heatmap
         plt.figure(figsize=(12, 10))
         sns.heatmap(data_numeric.corr(), annot=True, cmap='coolwarm')
         plt.title('Feature Correlation Matrix')
         plt.show()
         pd.set_option('display.max_rows', None, 'display.max_columns', None)
         print(correlation_matrix.to_string())
```



```
                  folder      file  classification
folder          1.000000 -0.004683        0.044978
file           -0.004683  1.000000        0.003106
classification  0.044978  0.003106        1.000000
```

Features are not highly correlated which supports the Naive Bayes Assumption that it treat features independently of each other. And it is not the main focus in terms of classifying spam and ham.

Where (1) train_spam_df is the number of spam emails in the training set, (2) train_ham_df is the number of ham emails in the training set,(3) train_df is the total number of emails (contains both ham and spam messages)

Q4. Recommendations to further improve the model

In order to further improve the importance, these are the things that are to be done based on conducting the activity:

1. Preprocessing should be done correctly since if words were not processed properly, it can influence the training and testing of the model and which can also contribute to noise.
2. Fo futher improve the model, it would be best to experiment on the k values and the lambda values
3. Identify features that are highly correlated since the assumption in the Naive Bayes is that they are independent so, it can be visualized through heatmap and clustermap.
4. Under pre-processing is stemming in which if you have words like cooking, it will be read as cook in which it will reduce the number of the unique words and may improve the model performance by joining related words.
5. There is also an imbalannce on the training and testing data sizes. One recommendation is to use the stratify parameter of the scikit-learn in order to maintain the class distribution to minimize the risk of being biased to a specific majority class.