Lab 06: Exploratory Data Analysis (EDA) with Clustering

Author: Nigel Nelson

Introduction:

- This lab acts as an exercise in using clustering techniques in order to aid in the exploratory data analysis of a data set comprised of emails that are labeled as spam, or ham. This data set contains 65,542 emails, all of which are stored as separate JSON files stored in a central directory. Each one of these email JSON files contains 5 features, the category describing if the email was spam or not, the email address of the recipient, the email address of the sender, the subject line of the email, and the body of the email. Students are tasked with converting this data to a Pandas DataFrame for easier manipulation. Next, this DataFrame is converted into a binary feature matrix for each email and each vocabulary word that appears in the email data set, where a 1 indicates that the email contains a given word and a 0 indicates the word is not contained in the email. Following this, a second feature matrix is created from the binary feature matrix using SVD to distill all of the information of the native emails into 10 columns/components. The two most predictive of these components are then clustered into groups by a clustering algorithm chosen from the available Scikit Learn algorithms. The calculated clusters are then visually depicted using plotting libraries to evaluate if the groups formed make logical sense given domain knowledge of the underlying data set. These clustered groups are further analyzed using a confusion matrix to determine if the calculated clusters were able to correctly distinguish spam vs. ham emails. Next, the vocabulary of the two clusters are statistically analyzed using binomial tests to determine which words appear more in one cluster than the other at a statistically significant p-value. Finally, the two clusters are analyzed by reviewing the corresponding data set entries to determine if information such as the subject line, recipient, and sender can lend more information as to what trends the clustering algorithm may be grouping by.

Imports:

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import TruncatedSVD
from scipy.sparse import csr_matrix
import matplotlib.pyplot as plt
import glob
import os
import json
from sklearn.cluster import DBSCAN
from sklearn.metrics import confusion_matrix
from scipy.sparse import csc_matrix
from scipy import stats
```

Part I: Load and Transform the Data

Loading in the Email Data Set:

In [2]:
```
email_dicts = []
json_dir_name = 'email_json/email_json/email_json/email_json'

json_pattern = os.path.join(json_dir_name, '*.json')
file_list = glob.glob(json_pattern)

for file in file_list:
    with open(file) as json_file:
        dict_data = json.load(json_file)
        email_dicts.append(dict_data)
```

Creating an Emails DataFrame:

In [3]:
```
email_df = pd.DataFrame.from_dict(email_dicts)
email_df['category'] = email_df['category'].astype('category')
email_df.head()
```

Out[3]:

	body	category	from_address	subject	
0	\n\n\n\n\n\n\nDo you feel the pressure to perf...	spam	"Tomas Jacobs" <RickyAmes@aol.com>	Generic Cialis, branded quality@	the00@spe
1	Hi, i've just updated from the gulus and I che...	ham	Yan Morin <yan.morin@savoirfairelinux.com>	Typo in /debian/README	mirrors
2	authentic viagra\n\nMega authenticV I A G R A...	spam	"Sheila Crenshaw" <7stocknews@tractionmarketin...	authentic viagra	<the00@
3	\nHey Billy, \n\nit was really fun going out t...	spam	"Stormy Dempsey" <vqucsmdfgvsg@ruraltek.com>	Nice talking with ya	opt4@spe
4	\n\n\n\n\n\n\nsystem" of the home. It will ha...	spam	"Christi T. Jernigan" <dcube@totalink.net>	or trembling; stomach cramps; trouble in sleep...	ktwarwic@spe

In [4]:
```
email_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63542 entries, 0 to 63541
Data columns (total 5 columns):
body            63542 non-null object
category        63542 non-null category
from_address    63542 non-null object
subject         63410 non-null object
to_address      63141 non-null object
dtypes: category(1), object(4)
memory usage: 2.0+ MB
```

Using Scikit Learn's to Create Feature Matrix:

In [5]:
```
vectorizer = CountVectorizer(binary=True, min_df=10)
feat_matrix = vectorizer.fit_transform(email_df['body'])
```

In [6]:
```
print(f'Feature matrix # columns: {feat_matrix.shape[1]}')
print(f'Feature matrix # rows: {feat_matrix.shape[0]}')
print(f'Feature matrix # nonzero entries: {csr_matrix.count_nonzero(feat_matr
```

```
Feature matrix # columns: 32144
Feature matrix # rows: 63542
Feature matrix # nonzero entries: 6388795
```

Part II: Cluster the Emails

Using Scikit Learn's TruncatedSVD to Create Feature Matrix with Top 2 Variables/Components:

In [7]:
```
svd = TruncatedSVD(n_components=10, random_state=17)
svd_matrix = svd.fit_transform(feat_matrix)

top_components = np.vstack((svd_matrix[:,0], svd_matrix[:,1])).T
top_components.shape
```

Out[7]:
```
(63542, 2)
```

Clustering Emails Using DBSCAN:

- DBSCAN was chosen to cluster this data set for several reasons. The first is that DBSCAN is capable of clustering non-flat geometry, and uneven cluster sizes, which are important characteristics to clustering this data set after visualizing the same collection of emails in lab 5. In addition, DBSCAN is able to label data points as outliers. This is important to this data set because it is likely there are some emails that are empty, contain extremely long messages, or have other attributes that could add noise to the proper separation of spam emails from ham emails.

In [8]:
```
clustering = DBSCAN(eps=0.5, min_samples=10).fit(top_components)
print(f'Set of class labels produced: {set(clustering.labels_)}')
```

```
Set of class labels produced: {0, 1, -1}
```

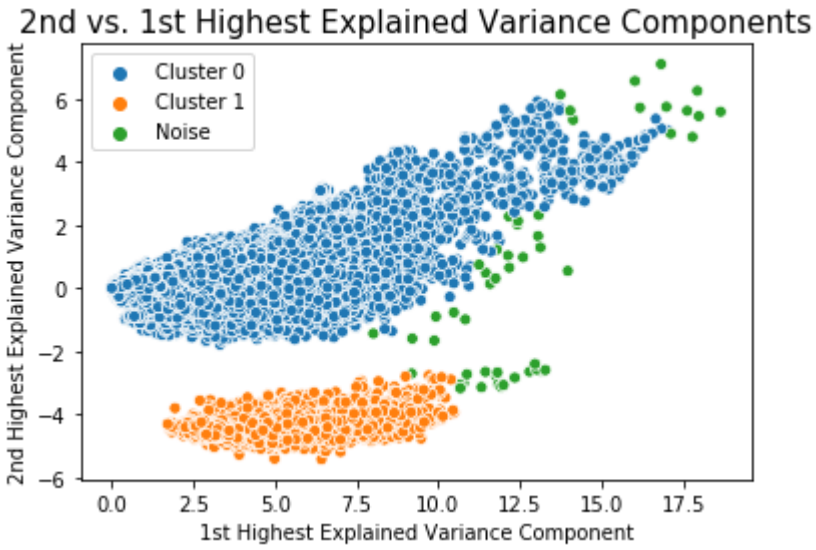
In [9]:
```
def label_to_string(label):
    if label == -1:
        return 'Noise'
    elif label == 0:
        return 'Cluster 0'
    else:
        return 'Cluster 1'

clusters = pd.Series(clustering.labels_)
cluster_labels = clusters.apply(label_to_string)
```

Plotting the Clusters Found with DBSCAN:

In [10]:
```
sns.scatterplot(x=svd_matrix[:,0], y=svd_matrix[:,1], hue=cluster_labels)
plt.ylabel('2nd Highest Explained Variance Component', fontsize=10)
plt.xlabel('1st Highest Explained Variance Component', fontsize=10)
plt.title('2nd vs. 1st Highest Explained Variance Components', fontsize=15)
```

Out[10]:
```
Text(0.5, 1.0, '2nd vs. 1st Highest Explained Variance Components')
```



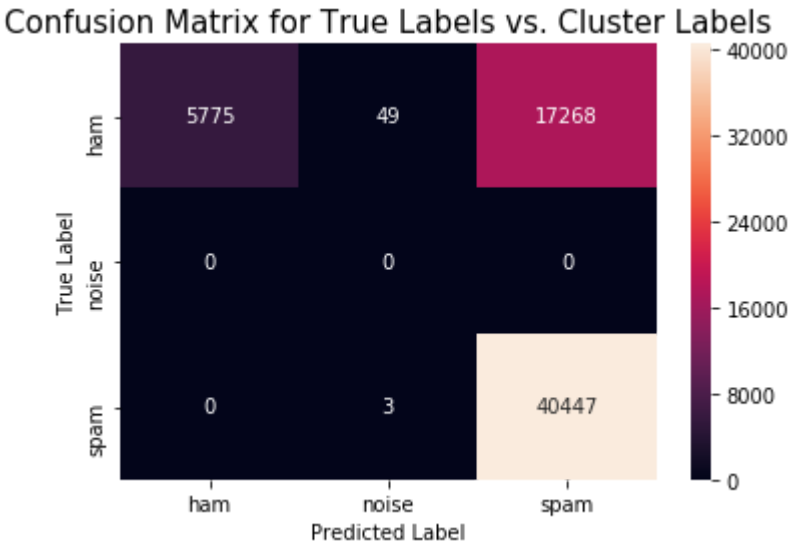
Creating Confusion Matrix for Clusters vs. Data Set Spam Labels:

In [11]:
```
def create_spam_labels(label):
    if label == -1:
        return 'noise'
    elif label == 0:
        return 'spam'
    else:
        return 'ham'

spam_labels = clusters.apply(create_spam_labels)
```

In [12]:
```
x = confusion_matrix(email_df['category'], spam_labels)
sns.heatmap(x, annot=True, fmt='g')
plt.yticks(np.arange(.5, 3.5, step=1), labels= ['ham', 'noise', 'spam'])
plt.xticks(np.arange(.5, 3.5, step=1), labels= ['ham', 'noise', 'spam'])
plt.ylabel('True Label', fontsize=10)
plt.xlabel('Predicted Label', fontsize=10)
plt.title('Confusion Matrix for True Labels vs. Cluster Labels', fontsize=15)
```

Out[12]: Text(0.5, 1, 'Confusion Matrix for True Labels vs. Cluster Labels')



Part III: Calculating Document Frequencies of Words

Removing Emails that were Labeled as Noise:

- The DBSCAN clustering algorithm labeled 52 emails as noise. This means that it was unable to place these emails into one of the two clusterers, meaning it didn't share enough sinmilarity with either to be considered apart of the cluster. In order to remove bias that these outliers may introduce, these noisey data points are removed from any further analysis.

In [13]:
```
cleaned_df = email_df[spam_labels != 'noise']
cleaned_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 63490 entries, 0 to 63541
Data columns (total 5 columns):
body            63490 non-null object
category        63490 non-null category
from_address    63490 non-null object
subject         63358 non-null object
to_address      63089 non-null object
dtypes: category(1), object(4)
memory usage: 2.5+ MB
```

Creating Separate Matrices for each Cluster:

In [14]:
```
cluster0_df = feat_matrix[(spam_labels == 'spam').values]
cluster1_df = feat_matrix[(spam_labels == 'ham').values]
```

Converting the Matrices to CSC Format:

In [15]:
```
cluster0_matrix = csc_matrix(cluster0_df)
cluster1_matrix = csc_matrix(cluster1_df)
```

Getting Document Frequency of each Word:

In [16]:
```
c0_doc_freq = cluster0_matrix.sum(0)
c1_doc_freq = cluster1_matrix.sum(0)

c1_doc_freq.shape
```

Out[16]: (1, 32144)

Document Frequencies of "love", "works", and "different":

In [17]:
```
love_idx = vectorizer.vocabulary_.get('love')
works_idx = vectorizer.vocabulary_.get('works')
diff_idx = vectorizer.vocabulary_.get('different')

print(f'Frequency of "love" in cluster 0 emails: {c0_doc_freq[0, love_idx]}')
print(f'Frequency of "works" in cluster 0 emails: {c0_doc_freq[0, works_idx]}
print(f'Frequency of "different" in cluster 0 emails: {c0_doc_freq[0, diff_id
print()
print(f'Frequency of "love" in cluster 1 emails: {c1_doc_freq[0, love_idx]}')
print(f'Frequency of "works" in cluster 1 emails: {c1_doc_freq[0, works_idx]}
print(f'Frequency of "different" in cluster 1 emails: {c1_doc_freq[0, diff_id
```

```
Frequency of "love" in cluster 0 emails: 2012
Frequency of "works" in cluster 0 emails: 2361
Frequency of "different" in cluster 0 emails: 2083

Frequency of "love" in cluster 1 emails: 23
Frequency of "works" in cluster 1 emails: 628
Frequency of "different" in cluster 1 emails: 776
```

Part IV: Find Enriched Words with Statistical Testing

Testing if the Words "works" and "love" are Enriched in Cluster 1

In [18]:
```
cluster_0_expected_prob = c0_doc_freq[0, works_idx] / cluster0_df.shape[0]
cluster_1_expected_prob = c1_doc_freq[0, works_idx] / cluster1_df.shape[0]
pvalue = stats.binom_test(c0_doc_freq[0, works_idx], cluster0_df.shape[0],
cluster_1_expected_prob, alternative="greater")
print(f'Expected probabilty of "works" appearing in cluster 0: {cluster_0_exp
print(f'Observed probabilty of "works" appearing in cluster 1: {cluster_1_exp
print(f'Binomial test p-value: {pvalue}')
```

```
Expected probabilty of "works" appearing in cluster 0: 0.04090790955557481
Observed probabilty of "works" appearing in cluster 1: 0.10874458874458874
Binomial test p-value: 0.9999999999999999
```

- Interpretation: The observed frequency of the word "works" for cluster 0 is NOT greater than the frequency for cluster 1

In [19]:
```
cluster_0_expected_prob = c0_doc_freq[0, love_idx] / cluster0_df.shape[0]
cluster_1_expected_prob = c1_doc_freq[0, love_idx] / cluster1_df.shape[0]
pvalue = stats.binom_test(c0_doc_freq[0, love_idx], cluster0_df.shape[0],
cluster_1_expected_prob, alternative="greater")
print(f'Expected probabilty of "love" appearing in cluster 0: {cluster_0_expe
print(f'Observed probabilty of "love" appearing in cluster 1: {cluster_1_expe
print(f'Binomial test p-value: {pvalue}')
```

```
Expected probabilty of "love" appearing in cluster 0: 0.034860954691154813
Observed probabilty of "love" appearing in cluster 1: 0.003982683982683983
Binomial test p-value: 0.0
```

- Interpretation: The observed frequency of the word "love" for cluster 0 is greater than the frequency for cluster 1

Iterating Over all Vocab in the Emails to Find Enriched Words for Cluster 0:

- Note: A p-value of 0.05 is used to determine if the word's frequency is statistically significant between the two clusters

In [20]:
```
def find_enriched_words(cluster_tested, num_emails_tested,
                        cluster_compared, num_emails_compared, word, word_idx
    compared_expected_prob = cluster_compared[0, word_idx] / num_emails_compa
    pvalue = stats.binom_test(cluster_tested[0, word_idx], num_emails_tested,
    compared_expected_prob, alternative="greater")
    return pvalue, word, cluster_tested[0, word_idx]
```

In [21]:
```
c0_significant_vocab = []

for word, word_idx in vectorizer.vocabulary_.items():
    pvalue, word, c0_word_freq = find_enriched_words(c0_doc_freq, cluster0_df
                                                     c1_doc_freq, cluster1_df
                                                    word, word_idx)
    if pvalue < 0.05:
        c0_significant_vocab.append((pvalue, word, c0_word_freq))
```

Filtering Out Non-Alphabetic Vocabulary:

In [22]:
```
print(f'Length of enriched word for cluster 0 before filtering: {len(c0_signi

cleaned_c0_significant_vocab = []

for  pvalue, word, c0_word_freq in c0_significant_vocab:
    if word.isalpha():
        cleaned_c0_significant_vocab.append((pvalue, word, c0_word_freq))

print(f'Length of enriched word for cluster 0 after filtering: {len(cleaned_c
```

```
Length of enriched word for cluster 0 before filtering: 23977
Length of enriched word for cluster 0 after filtering: 20300
```

In [23]:
```
cleaned_c0_significant_vocab.sort(key= lambda enriched_tuple : (enriched_tupl
print(cleaned_c0_significant_vocab[:201])
```

```
[(0.0, 'your', 25822), (0.0, 'we', 17259), (0.0, 'here', 13601), (0.0, 'ou
r', 13368), (0.0, 'up', 10841), (0.0, 'us', 9444), (0.0, 'day', 8353), (0.
0, 'click', 7697), (0.0, 'their', 7415), (0.0, 'unsubscribe', 7378), (0.0,
'most', 7144), (0.0, 'price', 7069), (0.0, 'over', 6912), (0.0, 'who', 685
1), (0.0, 'he', 6800), (0.0, 'online', 6781), (0.0, 'news', 5992), (0.0, 'b
uy', 5842), (0.0, 'said', 5574), (0.0, 'free', 5570), (0.0, 'his', 5546),
(0.0, 'money', 5381), (0.0, 'visit', 5325), (0.0, 'high', 5261), (0.0, 'con
tact', 5235), (0.0, 'ca', 5225), (0.0, 'service', 5223), (0.0, 'life', 483
2), (0.0, 'world', 4784), (0.0, 'site', 4708), (0.0, 'viagra', 4675), (0.0,
'quality', 4674), (0.0, 'low', 4404), (0.0, 'every', 4401), (0.0, 'busines
s', 4185), (0.0, 'net', 4185), (0.0, 'men', 4128), (0.0, 'today', 4105),
(0.0, 'home', 4075), (0.0, 'special', 4063), (0.0, 'her', 4005), (0.0, 'com
pany', 3980), (0.0, 'week', 3966), (0.0, 'she', 3672), (0.0, 'great', 364
2), (0.0, 'full', 3553), (0.0, 'top', 3549), (0.0, 'receive', 3482), (0.0,
'alert', 3474), (0.0, 'inc', 3464), (0.0, 'offer', 3394), (0.0, 'never', 33
76), (0.0, 'learn', 3375), (0.0, 'samba', 3343), (0.0, 'rights', 3331), (0.
0, 'ready', 3323), (0.0, 'man', 3308), (0.0, 'fast', 3306), (0.0, 'revisio
n', 3262), (0.0, 'additional', 3216), (0.0, 'uwaterloo', 3214), (0.0, 'cial
is', 3181), (0.0, 'product', 3164), (0.0, 'him', 3051), (0.0, 'support', 30
28), (0.0, 'products', 2986), (0.0, 'prices', 2960), (0.0, 'send', 2948),
(0.0, 'hk', 2932), (0.0, 'yourself', 2927), (0.0, 'perl', 2890), (0.0, 'res
erved', 2855), (0.0, 'root', 2808), (0.0, 'store', 2804), (0.0, 'customer',
2801), (0.0, 'pills', 2779), (0.0, 'modified', 2716), (0.0, 'speedy', 267
1), (0.0, 'target', 2644), (0.0, 'ever', 2632), (0.0, 'safe', 2603), (0.0,
'hours', 2601), (0.0, 'office', 2580), (0.0, 'watch', 2578), (0.0, 'autho
r', 2574), (0.0, 'privacy', 2556), (0.0, 'future', 2531), (0.0, 'request',
2523), (0.0, 'rev', 2451), (0.0, 'cgi', 2429), (0.0, 'stop', 2396), (0.0,
'yours', 2365), (0.0, 'internet', 2360), (0.0, 'account', 2351), (0.0, 'sui
te', 2347), (0.0, 'offers', 2338), (0.0, 'lowest', 2337), (0.0, 'branches',
2330), (0.0, 'generic', 2257), (0.0, 'away', 2255), (0.0, 'shop', 2254),
(0.0, 'delivery', 2250), (0.0, 'canadian', 2234), (0.0, 'join', 2233), (0.
0, 'months', 2215), (0.0, 'daily', 2214), (0.0, 'pay', 2209), (0.0, 'pharma
cy', 2192), (0.0, 'network', 2169), (0.0, 'team', 2161), (0.0, 'half', 215
1), (0.0, 'feel', 2135), (0.0, 'services', 2134), (0.0, 'night', 2133), (0.
0, 'worldwide', 2105), (0.0, 'chance', 2091), (0.0, 'subscribed', 2082),
(0.0, 'forward', 2072), (0.0, 'credit', 2054), (0.0, 'beginners', 2053),
(0.0, 'customers', 2050), (0.0, 'united', 2044), (0.0, 'effective', 2038),
(0.0, 'country', 2038), (0.0, 'millions', 2037), (0.0, 'copyright', 2036),
(0.0, 'minutes', 2027), (0.0, 'manager', 2024), (0.0, 'soon', 2022), (0.0,
'love', 2012), (0.0, 'live', 2010), (0.0, 'sex', 2005), (0.0, 'market', 200
3), (0.0, 'friend', 1974), (0.0, 'professional', 1946), (0.0, 'soft', 193
3), (0.0, 'alerts', 1932), (0.0, 'become', 1925), (0.0, 'receiving', 1921),
(0.0, 'video', 1917), (0.0, 'came', 1916), (0.0, 'viewcvs', 1903), (0.0, 'w
ebsvn', 1893), (0.0, 'levitra', 1859), (0.0, 'shipping', 1844), (0.0, 'car
e', 1842), (0.0, 'growth', 1834), (0.0, 'php', 1828), (0.0, 'die', 1815),
(0.0, 'anywhere', 1811), (0.0, 'person', 1795), (0.0, 'sell', 1794), (0.0,
'loss', 1787), (0.0, 'against', 1785), (0.0, 'broker', 1766), (0.0, 'stron
g', 1757), (0.0, 'secure', 1757), (0.0, 'changeset', 1753), (0.0, 'human',
1743), (0.0, 'drug', 1723), (0.0, 'play', 1718), (0.0, 'erectile', 1707),
(0.0, 'approved', 1698), (0.0, 'utc', 1682), (0.0, 'established', 1671),
(0.0, 'told', 1670), (0.0, 'payments', 1656), (0.0, 'story', 1650), (0.0,
'huge', 1650), (0.0, 'speech', 1642), (0.0, 'summer', 1641), (0.0, 'upon',
1636), (0.0, 'purchase', 1634), (0.0, 'review', 1619), (0.0, 'anatrim', 160
8), (0.0, 'benefits', 1604), (0.0, 'profit', 1600), (0.0, 'energy', 1583),
```

```
(0.0, 'tabs', 1572), (0.0, 'five', 1570), (0.0, 'dysfunction', 1564), (0.0,
'powerful', 1562), (0.0, 'increase', 1541), (0.0, 'women', 1539), (0.0, 'te
chnology', 1535), (0.0, 'microsoft', 1534), (0.0, 'face', 1523), (0.0, 'fa
t', 1514), (0.0, 'lose', 1513), (0.0, 'erection', 1506), (0.0, 'private', 1
490), (0.0, 'advertisement', 1484), (0.0, 'penis', 1475), (0.0, 'security',
1469), (0.0, 'drugs', 1463), (0.0, 'media', 1458), (0.0, 'manage', 1456),
(0.0, 'pill', 1452), (0.0, 'patch', 1448), (0.0, 'bank', 1433), (0.0, 'keep
ing', 1431)]
```

Iterating Over all Vocab in the Emails to Find Enriched Words for Cluster 1:

- Note: A p-value of 0.05 is used to determine if the word's frequency is statistically significant between the two clusters

In [24]:
```
c1_significant_vocab = []

for word, word_idx in vectorizer.vocabulary_.items():
    pvalue, word, c1_word_freq = find_enriched_words(c1_doc_freq, cluster1_df
                                                     c0_doc_freq, cluster0_df
                                                    word, word_idx)
    if pvalue < 0.05:
        c1_significant_vocab.append((pvalue, word, c1_word_freq))
```

Filtering Out Non-Alphabetic Vocabulary:

In [25]:
```
print(f'Length of enriched word for cluster 1 before filtering: {len(c1_signi

cleaned_c1_significant_vocab = []

for  pvalue, word, c1_word_freq in c1_significant_vocab:
    if word.isalpha():
        cleaned_c1_significant_vocab.append((pvalue, word, c1_word_freq))

print(f'Length of enriched word for cluster 0 after filtering: {len(cleaned_c
```

```
Length of enriched word for cluster 1 before filtering: 5357
Length of enriched word for cluster 0 after filtering: 4517
```

In [26]:
```
cleaned_c1_significant_vocab.sort(key= lambda enriched_tuple : (enriched_tupl
print(cleaned_c1_significant_vocab[:201])
```

```
[(0.0, 'do', 5775), (0.0, 'the', 5775), (0.0, 'self', 5775), (0.0, 'htt
p', 5775), (0.0, 'org', 5775), (0.0, 'guide', 5775), (0.0, 'help', 5775),
(0.0, 'www', 5775), (0.0, 'mailing', 5775), (0.0, 'stat', 5775), (0.0, 'm
ath', 5775), (0.0, 'ethz', 5775), (0.0, 'ch', 5775), (0.0, 'mailman', 577
5), (0.0, 'listinfo', 5775), (0.0, 'read', 5775), (0.0, 'posting', 5775),
(0.0, 'project', 5775), (0.0, 'provide', 5775), (0.0, 'commented', 5775),
(0.0, 'minimal', 5775), (0.0, 'contained', 5775), (0.0, 'reproducible', 5
775), (0.0, 'code', 5775), (0.0, 'and', 5763), (0.0, 'list', 5718), (0.0,
'html', 5708), (0.0, 'please', 5704), (0.0, 'https', 5690), (0.0, 'to', 5
380), (0.0, 'in', 4835), (0.0, 'of', 4737), (0.0, 'is', 4725), (0.0, 'thi
s', 4086), (0.0, 'that', 3714), (0.0, 'but', 3394), (0.0, 'can', 3006),
(0.0, 'wrote', 3000), (0.0, 'thanks', 2786), (0.0, 'there', 2490), (0.0,
'data', 2452), (0.0, 'am', 2208), (0.0, 'use', 2186), (0.0, 'using', 217
6), (0.0, 'my', 2174), (0.0, 'how', 2163), (0.0, 'would', 2133), (0.0, 'a
ny', 2082), (0.0, 'which', 2058), (0.0, 'function', 2057), (0.0, 'versio
n', 2035), (0.0, 'hi', 1903), (0.0, 'want', 1697), (0.0, 'message', 163
2), (0.0, 'problem', 1508), (0.0, 'alternative', 1502), (0.0, 'deleted',
1426), (0.0, 'package', 1364), (0.0, 'example', 1321), (0.0, 'error', 128
6), (0.0, 'following', 1277), (0.0, 'university', 1173), (0.0, 'trying',
1116), (0.0, 'true', 986), (0.0, 'fax', 901), (0.0, 'matrix', 862), (0.0,
'values', 861), (0.0, 'library', 842), (0.0, 'tried', 837), (0.0, 'fram
e', 826), (0.0, 'plot', 805), (0.0, 'question', 796), (0.0, 'anyone', 75
4), (0.0, 'statistics', 677), (0.0, 'model', 642), (0.0, 'advance', 618),
(0.0, 'tel', 618), (0.0, 'professor', 594), (0.0, 'packages', 589), (0.0,
'mailto', 587), (0.0, 'vector', 567), (0.0, 'ac', 557), (0.0, 'edu', 55
5), (0.0, 'variable', 540), (0.0, 'bounces', 531), (0.0, 'stats', 525),
(0.0, 'variables', 498), (0.0, 'archive', 481), (0.0, 'column', 448), (0.
0, 'factor', 439), (0.0, 'brian', 437), (0.0, 'columns', 426), (0.0, 'rip
ley', 425), (0.0, 'ph', 407), (0.0, 'row', 400), (0.0, 'nabble', 398),
(0.0, 'rows', 369), (0.0, 'oxford', 350), (0.0, 'numeric', 343), (0.0, 'o
x', 326), (0.0, 'parks', 314), (0.0, 'cran', 295), (0.0, 'statistical', 2
91), (0.0, 'regression', 289), (0.0, 'subset', 281), (0.0, 'dataset', 27
7), (0.0, 'col', 275), (0.0, 'linear', 274), (0.0, 'df', 274), (0.0, 'rno
rm', 274), (0.0, 'sep', 272), (0.0, 'plots', 247), (0.0, 'axis', 240),
(0.0, 'cbind', 222), (0.0, 'sas', 210), (0.0, 'datasets', 209), (0.0, 'nr
ow', 197), (0.0, 'gabor', 188), (0.0, 'biostatistics', 177), (0.0, 'varia
nce', 176), (0.0, 'ncol', 174), (0.0, 'lattice', 170), (0.0, 'lapply', 16
4), (0.0, 'vectors', 164), (0.0, 'holtman', 149), (0.0, 'dataframe', 14
9), (0.0, 'rbind', 145), (0.0, 'htmland', 145), (0.0, 'deepayan', 145),
(0.0, 'biostat', 145), (0.0, 'coefficients', 142), (0.0, 'listhttps', 13
9), (0.0, 'helpplease', 139), (0.0, 'grothendieck', 137), (0.0, 'sapply',
132), (0.0, 'anova', 131), (0.0, 'matrices', 130), (0.0, 'xyplot', 129),
(0.0, 'colnames', 128), (0.0, 'runif', 127), (0.0, 'spss', 124), (0.0, 'r
ownames', 121), (0.0, 'grdevices', 114), (0.0, 'hmisc', 110), (0.0, 'harr
ell', 110), (0.0, 'xlab', 109), (0.0, 'ylab', 107), (0.0, 'glm', 101),
(0.0, 'ylim', 101), (0.0, 'lme', 95), (0.0, 'sqrt', 93), (0.0, 'ligges',
93), (0.0, 'dalgaard', 93), (0.0, 'optim', 89), (0.0, 'logistic', 88),
(0.0, 'sessioninfo', 87), (0.0, 'tapply', 85), (0.0, 'covariance', 79),
(0.0, 'weiwei', 77), (0.0, 'textconnection', 76), (0.0, 'lmer', 75), (0.
0, 'psych', 75), (0.0, 'xlim', 75), (0.0, 'coef', 72), (0.0, 'sundar', 7
1), (0.0, 'surv', 70), (0.0, 'gerontology', 68), (0.0, 'monteiro', 68),
(0.0, 'xfmail', 67), (0.0, 'nlme', 66), (0.0, 'adschai', 65), (0.0, 'inte
rmountainmail', 65), (0.0, 'abline', 65), (0.0, 'bioconductor', 63), (0.
0, 'genego', 62), (0.0, 'varadhan', 61), (0.0, 'sweave', 60), (0.0, 'trel
```

```
lis', 60), (0.0, 'cex', 59), (0.0, 'jhsph', 57), (0.0, 'gunter', 55), (0.
0, 'klasterecky', 55), (0.0, 'tcltk', 55), (0.0, 'boxplot', 54), (0.0, 'e
lyakhlifi', 54), (0.0, 'predictors', 53), (0.0, 'dimitris', 53), (0.0, 'm
ustapha', 53), (0.0, 'jhmi', 53), (0.0, 'rvaradhan', 52), (0.0, 'dieter',
52), (0.0, 'finzi', 51), (0.0, 'jfox', 50), (0.0, 'rodbc', 50), (0.0, 'no
nclinical', 50), (0.0, 'envir', 50), (0.0, 'kuleuven', 50), (0.0, 'cwis',
50), (0.0, 'agingandhealth', 50), (0.0, 'socserv', 49), (0.0, 'correlatio
ns', 49)]
```

Reflection Questions:

1. Make a guess as to why the emails might form two distinct clusters
    A. After skimming through the two clustered groups of email bodies, it appears that the reason for these two distinct groups is due to two different email body types, one which represents an initial email and one which represents a continuation of an email conversation. In many emails I saw, they began with a date, time, and what was last emailed in the conversation. The structure of this “reply” style email was likely picked up by the two most predictive principle components through common alpha-numeric tokens used. So, it is likely that the top, larger cluster seen in the plots represents initial emails, whereas the bottom, smaller cluster represents emails that were sent as a reply.
2. Compare the ham/spam labels to the cluster labels using the confusion matrix you generated. Are spam messages in both clusters or a single cluster? Are all of the messages in the clusters with spam labeled as spam?
    A. Spam emails are all in a single cluster. Howeveer, this cluster contains both spam and ham.
3. Skim through the top 200 words for each cluster. Can you identify any patterns for either of the clusters?
    A. One of the patterns that I noticed for the cluster containing spam is that the most common vocabulary included words that indicate requested action (“click”, “unsubscribe”, “buy”, and “request”) or vocabulary that is associated with products (“price”, “free”, “money”, “Viagra”). However, for the cluster that contained only ham emails, the vocabulary reflected a much more technical email body (“stat”, “math”, “project”, “reproducible”, and “code”).
4. Select the rows in the DataFrames for the emails in cluster 0. Print the top 25. Do the same for cluster 1. Do you the to and from addresses and subject lines provide additional help in identifying patterns?

In [27]:
```
email_df[(spam_labels == 'spam').values].head(25)
```

Out[27]:

	body	category	from_address	subject
0	\n\n\n\n\n\n\nDo you feel the pressure to perf...	spam	"Tomas Jacobs" <RickyAmes@aol.com>	Generic Cialis, branded quality@
1	Hi, i've just updated from the gulus and I che...	ham	Yan Morin <yan.morin@savoirfairelinux.com>	Typo in /debian/README
2	authentic viagra\n\nMega authenticV I A G R A...	spam	"Sheila Crenshaw" <7stocknews@tractionmarketin...	authentic viagra
3	\nHey Billy, \n\nit was really fun going out t...	spam	"Stormy Dempsey" <vqucsmdfgvsg@ruraltek.com>	Nice talking with ya
4	\n\n\n\n\n\n\nsystem" of the home. It will ha...	spam	"Christi T. Jernigan" <dcube@totalink.net>	or trembling; stomach cramps; trouble in sleep...
5	\n\n\n\n\n\n\nthe program and the creative abi...	spam	"Bobby L. Fleming" <zvyrepeated@liselebel.com>	Which is duty
6	\n\n\n\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t...	spam	"Esteban Everett" <dbmagyrln@dbmag.com>	Theorize get inside for local esc0rts who do i...
7	\n\n\n\n\n\n HoodiaLife - Start Losing Weight ...	spam	Real Loss <fibond@terra.com.do>	Losing Weight Quickly
9	\n\n\n\n\n\n\nGood day!\nVisit our new online ...	spam	"Billy Chin" <boxer@0733.com>	For Smile
10	\nCheck out the wonders of pound melting\n\n\n...	spam	"Josie Abernathy" <cebcalhounrennerhog@calhoun...	Less weight - more pleasure and joy
11	\n\n\n\n\n\n\nmovement on the tablet. I could...	spam	wing as <mstep@mavorama.com>	problems in animal studies using rats or rabbi...
12	\n\n\n\n\n\n\n\n	spam	"Geoffrey" <reginald@funeasy.biz>	We cure any desease!
13	\n\n\n\n\n\nWhat is HGH Life™? \n HGH Life™ is ...	spam	4ever Young <ualicia@netscape.net>	Human Growth Hormone
14	Hey Billy, \n\nit was really fun going out the...	spam	"Caroyln Herbert" <slqwil@planetonline.com>	hi man
15	\n\n\n\n\n\n\ncan assist advertising and marke...	spam	Domingo Bolden <lfriend@dryden.com>	when taken for a long time or in large doses, ...
17	\n\n\n\n\n\n\n\n	spam	"Henry" <gilbert@garageservices.biz>	We cure any desease!
18	\n\n\n\n\n\n\n\n	spam	"Simon" <walter@griffield.biz>	Need medicine? All here!
19	\n\n\n\n\n\n\n\n	spam	"Philip" <richard@expomedica.biz>	Need medicine? All here!
20	\n\n\n\n\n\n\nBecome Fit For Life!\n\nHGH is a...	spam	Young Future <kdemetrius@publicist.com>	Become Fit For Life!
21		spam	"Gena Blanco" <nytwdn@incamail.com>	Please Read This C

	body	category	from_address	subject
22	\nMake your fat friends envy you\n\n\n\n\n\n\n...	spam	"Carlene Campos" <cebcaligularhog@caligular.com>	Be leaner and slimmer by next week
23	\n\n\n\n\nRemember HANS and FIZ\nFire Mountain...	spam	"Chadwick Milesc" <ctuballerina@bb-autom.nl>	But serpentine
25	\n\n\n\n\n\n\n\n	spam	"Henry" <richard@expomedica.biz>	All products for your health!
26	\n\n\n\n\n\n\n\nSunday, 08 April, 2007, 18:00 ...	ham	"BBC daily email" <dailyemail@bbc.co.uk>	Your daily e-mail from the BBC
27	\n\n\n\n\n\nWhat is HGH Life™? \n HGH Life™ is ...	spam	4ever Young <doaraceli@altavista.nl>	HGH really helps!

In [28]:
```
email_df[(spam_labels == 'ham').values].head(25)
```

Out[28]:

	body	category	from_address	subject	
8	\nHi...\n\nI have to use R to find out the 90%...	ham	"Jochen.F" <jjfahr@ucalgary.ca>	[R] Confidence-Intervals.... help...	r-help(
16	Hm... sounds like a homework problem to me...\...	ham	"Sarah Goslee" <sarah.goslee@gmail.com>	Re: [R] Confidence-Intervals.... help...	<jj
24	Daer r-helpers,\n\nCan anyone help with the fo...	ham	Michael Kubovy <kubovy@virginia.edu>	[R] Failure of mcsamp() but not mcmcsamp()	r-help(
68	On 4/6/07, Wilfred Zegwaard wrote:\n\n> I'm n...	ham	"Johann Hibschman" <johannh@gmail.com>	Re: [R] Reasons to Use R	<wilfred.zegw
75	On 4/8/07, Johann Hibschman wrote:\n> R's pas...	ham	"Gabor Grothendieck" <ggrothendieck@gmail.com>	Re: [R] Reasons to Use R	"J <joh
112	I have a question to everybody.\n\nAfter log10...	ham	"Zia Uddin Ahmed" <zua3@cornell.edu>	[R] How do I back transforme ordinary log-krig...	r-help(
149	\nI am writing some code to obtain publication...	ham	"Cressoni, Massimo $NIH/NHLBI$ [F]" <cresson...	[R] Plot symbols dimensions	<r-help@
278	Dear Johann and Gabor,\n\nIt's what amounts to...	ham	Wilfred Zegwaard <wilfred.zegwaard@gmail.com>	Re: [R] Reasons to Use R	r-help(
307	Dear R-users,\n\nI would like to use "bruto" f...	ham	=?ISO-2022-JP?B?GyRCQG44fRsoQiAbJEI9JDwjGyhC?=...	[R] Could not fit correct values in discrimina...	r-help(
318	Dear R users,\n\nI am new to R. I would like t...	ham	"joey repice" <fireseedmusic@gmail.com>	[R] R:Maximum likelihood estimation using BHHH...	r-help(
323	Hi,\n\nI am using tsls function from sem packa...	ham	adschai@optonline.net	[R] Dealing with large nominal predictor in se...	r-help(
408	On Sun, 8 Apr 2007, hadley wickham wrote:\n\n>...	ham	Prof Brian Ripley <ripley@stats.ox.ac.uk>	Re: [R] data encapsulation with classes	<h.wicl
472	Hello,\n\nI want to know if there are some fun...	ham	Shao <xshining@gmail.com>	[R] How to solve differential and integral equ...	help@

	body	category	from_address	subject	
501	Joey,\n\nFirst of all, it is bad habit to call...	ham	chao gai <chaogai@duineveld.demon.nl>	Re: [R] R:Maximum likelihood estimation using ...	r-help(
534	Thanks,\nI have much to learn~~~\n\nShao chunx...	ham	Shao <xshining@gmail.com>	Re: [R] How to solve differential and integral...	help@
578	Dear adschai,\n\nIt's not possible to know fro...	ham	"John Fox" <jfox@mcmaster.ca>	Re: [R] Dealing with large nominal predictor i...	<adsch
582	Thanks anhnmncb\nI think the problem comes fro...	ham	fsando <fsando@fs-analyse.dk>	Re: [R] R 'could not find any X11 fonts'	r-help(
594	I am trying to install the gnomeGUI package\nI...	ham	fsando <fsando@fs-analyse.dk>	[R] Problem installing gnomeGUI in Ubuntu: "HA...	r-help(
622	Shuji,\n\nI suspect that bruto blows up becaus...	ham	"Kuhn, Max" <Max.Kuhn@pfizer.com>	Re: [R] Could not fit correct values in discri...	<kawagu
629	>>>>> "BDR" == Prof Brian Ripley \n>>>>> o...	ham	Martin Maechler <maechler@stat.math.ethz.ch>	Re: [R] data encapsulation with classes	<riple
630	Does anyone know of a package that includes th...	ham	"Chris Elsaesser" <chris.elsaesser@spadac.com>	[R] Modified Sims test	<r-help@
668	Gabor has already showed you one way to make y...	ham	"Greg Snow" <Greg.Snow@intermountainmail.org>	Re: [R] lm() intercept at the end, rather than...	<dimitrijoe
694	tha s9ze of db is an issue with R. We are stil...	ham	"Jorge Cornejo-Donoso" <jorgecornejo@uach.cl>	Re: [R] Reasons to Use R	'" <wilfred.ze
706	Have you tried 64 bit machines with larger mem...	ham	"Gabor Grothendieck" <ggrothendieck@gmail.com>	Re: [R] Reasons to Use R	"Jorg <jorge
715	I have a Dell with 2 Intel XEON 3.0 procesors ...	ham	"Jorge Cornejo-Donoso" <jorgecornejo@uach.cl>	Re: [R] Reasons to Use R	"'Ga <ggrothen

4. -

   A. By comparing email entries from both of the clusters, two attributes appear to separate the two groups. The first is that the subject lines for cluster 1, the smaller cluster that appears lower on the cluster plots, contain some indication that the emails were reply emails. This is indicated in the subject lines by the prefix “Re:”. Second, cluster 1 appears to have many more “common” email domains for the individual that sent the email than in cluster 1. This was observed because cluster 1 has several senders with “@gmail.com” and “.edu” email domains compared to cluster 0 which primarily has suspicious domains such as “@0733.com” and “@funeasy.biz”.
5. The clusters represent email from two separate mailing lists. One mailing list is for the R programming language, while the other mailing list is for a university. Which mailing list contained all of the spam?
   A. As mentioned earlier, cluster 1 contained many more technical terms such as “stat”, “math”, “project”, “reproducible”, and “code”. All of these terms are common when discussing programming languages, as such, it appears that cluster 1 was the R programming language mailing list. This conclusion is further supported by the fact that many of the subject lines contain references to 'R' and phrases such as 'Reasons to Use R', which are extremely likely to be references to the R progamming language. In addition, just about all of email recipients for cluster 0 had the domain “uwaterloo.ca”, further indicating that cluster 0 was likely the university mailing list.

Conclusion:

- This lab acted as an exercise in using clustering techniques in order to aid in the exploratory data analysis of a data set comprised of emails that are labeled as spam, or ham. This data set contains 65,542 emails, all of which are stored as separate JSON files stored in a central directory. In order to find the most accurate clusters that could be found using this data set, the DBSCAN algorithm was used from Scikit Learn due to its ability to cluster non-flat geometry, uneven clusters, and label data points as outliers. Using the body of the emails, DBSCAN was able to find two clusters within the data, and labeled 52 emails as outliers. These groups somewhat corresponded to whether the emails were spam or ham, as every email in cluster 1 was ham, but emails in cluster 0 were roughly 2/3 spam and 1/3 ham. By analyzing the vocabulary for statistically significant differences in word frequencies it was found that most of the spam emails contained vocabulary associated with action, “click”, “unsubscribe”, “buy”, and “request”, as well as products, “price”, “free”, “money”, “Viagra”. Whereas the ham emails contained vocabulary that was much more technical, “stat”, “math”, “project”, “reproducible”, and “code”. Lastly, after learning that this data set was comprised of a mailing list for the R programming language and a university mailing list, it became clear that this was likely the two groups that DBSCAN was clustering by. This was made apparent by observing that cluster 0 all had recipient domains of “uwaterloo.ca”, and that cluster 1 had subject lines containing indications of active replies to emails indicated by the “Re:” prefix and also the fact that the subject lines contained many uses of ‘R’, which is almost positively a reference to the R programming language.