Case Study 3 Report:

Spam Classifier

Alexy Morris and Domicia Herring

October 3, 2022

Abstract

The team has been provided emails from the IT department to create an algorithm of whether an email is spam or not.

1. Introduction

1.1 Business Understanding

Spam emails are massive amounts of unwanted or unsolicited emails that are mostly sent for commercial purposes but can also be for nefarious reasons. While the former is just annoying to an email user, the latter can be harmful phishing attempts to gain access to your computer, funds, or information. Spam filtering is an important feature for business and personal use. A person may receive thousands of emails per day with up to 85% being spam mail. Therefore, ensuring that you are finding the important mail most email services have created spam filters.

Spam prevention is of high importance to businesses, due to the significant amount they receive. This case study's purpose is to build a spam classifier using Naive Bayes and clustering to predict which email is spam and what is not.

1.2 Data Meaning Type

To engineer an algorithm that predicts the factors for whether an email is spam or not, five datasets from The Sam Assassin Dataset will be utilized. The dataset the five datasets of 9,353 emails are as follows:

Spam emails

- The “spam” file contained 1,001 spam messages.
- The “spam_2” file contained 1,398 spam messages from more current sources.

Non-spam emails

- The “easy_ham” file contained 5,052 non-spam messages.
- The “easy_ham_2” file contained 1,401 non-spam messages from more current sources.
- The “hard_ham” file contained 501 non-spam messages which contains spam like language.

2. Methods

2.1 Data Preprocessing

The original dataset consisted of five folders with several emails in each folder. To create a cohesive dataset, the data was read into the notebook using the os packages listdir function and a for loop which appended our emails into a list with the correct encoding. The team created a target list to match our emails and then was stored together into a sub-dataset. This process was repeated for each folder and was finally stacked into a final dataset. To differentiate, the spam was marked as '1' while ham was '0'.

2.2 Exploratory Data Analysis

After combining the data for a concise dataset, the team began by ensuring the data was categorized correctly. Figure 2.2a and Figure 2.2b shows that the dataset consists of 6,954 ham emails and 2,399 spam emails. This confirmed the team's suspicion that there is a large inequality in the data that would cause an unbalanced effect on the models. Thus, the team chose to down sample the ham emails to match the number of spam emails.

```
0    6954
1    2399
Name: target, dtype: int64
```

Figure 2.2a- Dataset breakdown

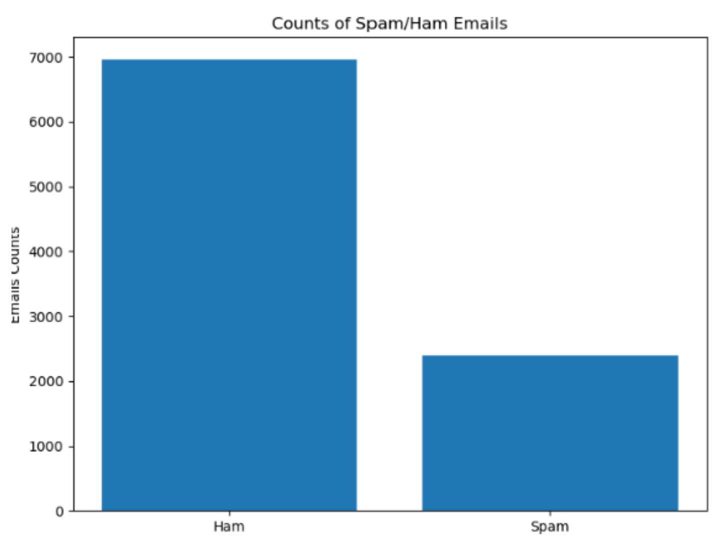


Figure 2.2b- Bar Graph of Dataset

Since the dataset is in a non-traditional html format, the team set out to view samples of the emails. Both samples of the emails with (Figure 2.2c), and without formatting (Figure 2.2d) were obtained.

```
also has sex with the other 499,999 women.

499,999 women have had more than one partner. 499,999 men have
only had one partner. It is now "perfectly obvious" that in the
common meaning of the term, among this contrived population,
that women are "more promiscuous" than men -- even though the
single "most promiscuous" person, Wilt, is a man.

"Promiscuity" is not "exactly, perfectly identical between males
and females", except under a degenerate custom definition of
"promiscuity".

> >unless "promiscuity" is defined uselessly.
>
> Ain't nothin' useless about averages.

Averages are useful, sure -- but much more so if called by their
actual name, rather than conflated with another concept.

- Gordo
```

Figure 2.2c-Sample of Formatted Email

```
['From fork-admin@xent.com  Sun Sep  8 23:50:39 2002\n', 'Return-Path: <fork-admin@xent.com>\n', 'Delivered-To: yyyy@localhost.spama
```

Figure 2.2d-Sample of Unformatted Email

Satisfied with the dataset, the team split our dataset into data/target train and test subsets using the train_test_split function from the sklearn model_selection package, this function contains an inherent row split randomizer which was greatly important as the created dataset was neatly stacks by spam and non-spam. Then, the “\n”s was replaced in the data test and train sets to standardize the model. Finally, to prepare for the two models the data was transformed and stored in different variables. For the Naive Bayes, a count vectorizer was applied. For the k-means model, both a count vectorizer and tf-idf transformation was applied. At this point, the models were prepared to run.

2.2 Models

2.2.1 Naive Bayes Model

To create our initial classifier, the team chose the Multinomial Naive Bayes Classifier which is used with discrete features and thus suitable for text classification. A variety of parameters were tested; however it was found that the baseline model gave the best results.

2.2.2 K-Means Model

Due to K-means clustering only taking numeric data, the data was transformed with a tf-idf vectorizer. The team chose k-means specifically because it allows for the clustering size to be set. Also, it was decided that a flat geometry metric works best for the studies purposes.

3. Results

3.1 Naive Bayes Model

For the Naive Bayes model, the confusion matrix (Figure 3.1a) shows a precision of 99% and a recall of 93% for the spam emails. Meaning the spam emails were classified correctly 99% with a capture rate of 93%. In contrast, the non-spam emails show a precision of 93% and a recall of 99%. Meaning the non-spam emails were classified correctly 93% with a capture rate of 99%. Figure 3.1b visually depicts the model's accuracy.

	precision	recall	f1-score	support
0	0.93	0.99	0.96	807
1	0.99	0.93	0.96	777
accuracy			0.96	1584
macro avg	0.96	0.96	0.96	1584
weighted avg	0.96	0.96	0.96	1584

Figure 3.1a-Navie Bayes Confusion Matrix

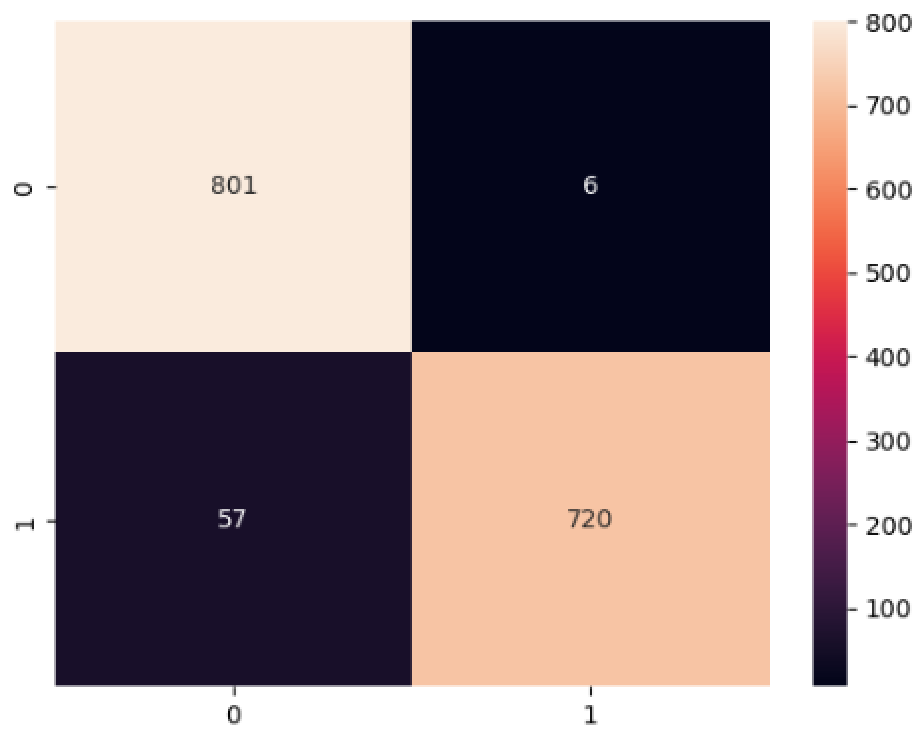


Figure 3.1b-Navie Bayes Heat Map

3.2 K-Means Model

For the K-Means model, there are five evaluation points the team chooses to evaluate shown in Figure 3.2a and 3.2b. They are described as follows:

- The Homogeneity score is a 0 to 1 measurement of degree where the cluster contains only assignments of the class it was given.
- The Completeness score is a measurement where all members of the same class are assigned to the same cluster.
- The V-measure score is a score from 0 to 1 that measures the homogeneity and completeness.
- The Adjusted Rand-Index is the Rand Index computing the similarities between two

clustering by all sample pairs and counting pairs adjusted for chance.

- The Silhouette score is used to measure distance between clusters.

Analyzing the measurements detailed above, the team sees that the Silhouette score is fairly low at 0.057, the V-measure is also fairly low with a low homogeneity score of 0.166 and the Adjusted Rand- Index is low at 0.140. All these things combined lead the team to believe the model is not the best and would need to be enhanced in some way.

```
clustering done in 1.04 ± 0.12 s
Homogeneity: 0.166 ± 0.000
Completeness: 0.217 ± 0.000
V-measure: 0.188 ± 0.000
Adjusted Rand-Index: 0.140 ± 0.000
Silhouette Coefficient: 0.057 ± 0.002
```

Figure 3.2a-K-Means Model

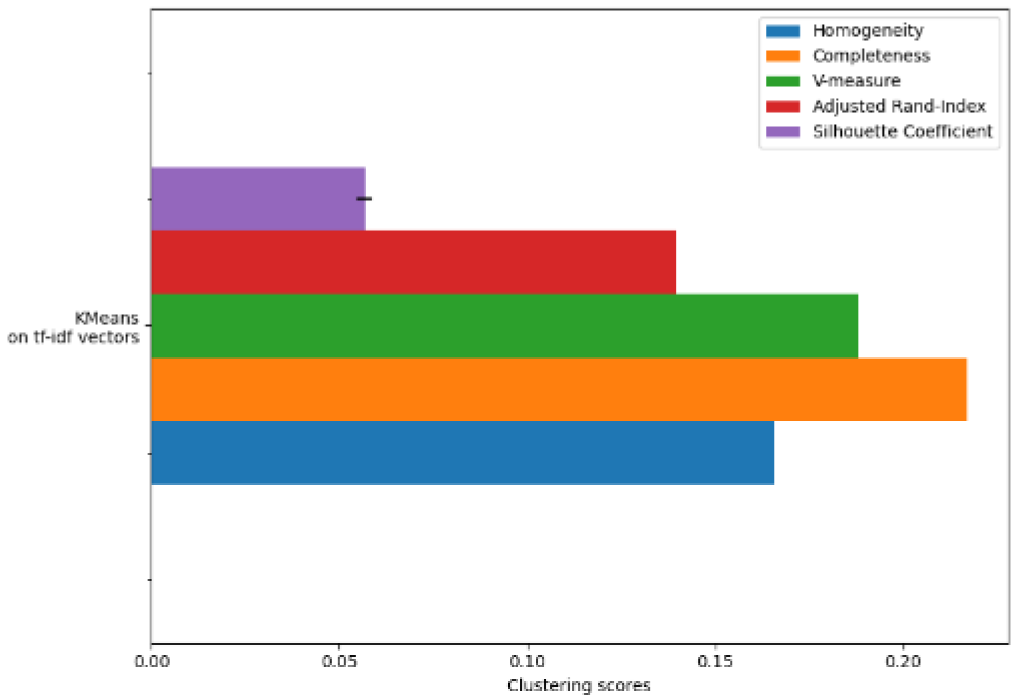


Figure 3.2b-K-Means Model Bar Graph

4. Conclusion

With 85% of incoming emails being spam, anyone can make the argument that removing all those emails would be for the best interest of the users. While it's unlikely an algorithm can remove all the spam email, there are ways to build highly efficient models to remove most of the risky emails.

The team was able to classify emails based on various factors. The clustering model yielded an adjusted rand-index of 14%. Also, the Naive Bayes model yielded an accuracy of 99%. Therefore, the team can conclude that the Naive Bayes model has a higher accuracy rate of classifying whether an email is spam.

5. Citation

5.1. Cveticanin, Nikolina. "What's on the Other Side of Your Inbox - 20 SPAM Statistics for 2022." Dataprot, 20 July 2022, dataprot.net/statistics/spam-statistics.

5.2. Zuccarelli, Eugenio. "Performance Metrics in Machine Learning—Part 3: Clustering." Medium, 31 Jan. 2021, https://towardsdatascience.com/performance-metrics-in-machine-learning-part-3-clustering-d69550662dc6.

6. Code

The code is on the following page.

In [1]:

```
import pandas as pd
import os
from collections import Counter, defaultdict
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.utils import resample
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisp
lay
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn import metrics
from time import time
```

tmp = os.listdir("C:/Users/Alex M/Downloads/SpamAssassinMessages/easy_ham")

ignore warning, code runs properly x = []

for i in tmp: with open(os.path.join("C:/Users/Alex M/Downloads/SpamAssassinMessages/easy_ham", i), 'r', encoding = "cp437") as f: x.append(f.read()) print(x)

y = [0]*len(x)

df1 = pd.DataFrame(list(zip(x, y)), columns =['text', 'target']) df1.head()

tmp = os.listdir("C:/Users/Alex M/Downloads/SpamAssassinMessages/easy_ham_2")

x = []

for i in tmp: with open(os.path.join("C:/Users/Alex M/Downloads/SpamAssassinMessages/easy_ham_2", i), 'r', encoding = "cp437") as f: x.append(f.read()) print(x)

y = [0]*len(x)

df2 = pd.DataFrame(list(zip(x, y)), columns =['text', 'target']) df2.head()

tmp = os.listdir("C:/Users/Alex M/Downloads/SpamAssassinMessages/hard_ham")

x = []

for i in tmp: with open(os.path.join("C:/Users/Alex M/Downloads/SpamAssassinMessages/hard_ham", i), 'r', encoding = "cp437") as f: x.append(f.read()) print(x) y = [0]*len(x)

df3 = pd.DataFrame(list(zip(x, y)), columns =['text', 'target']) df3.head()

tmp = os.listdir("C:/Users/Alex M/Downloads/SpamAssassinMessages/spam")

x = []

for i in tmp: with open(os.path.join("C:/Users/Alex M/Downloads/SpamAssassinMessages/spam", i), 'r', encoding = "cp437") as f: x.append(f.read()) print(x)

y = [1]*len(x)

df4 = pd.DataFrame(list(zip(x, y)), columns =['text', 'target']) df4.head()

tmp = os.listdir("C:/Users/Alex M/Downloads/SpamAssassinMessages/spam_2")

x = []

for i in tmp: with open(os.path.join("C:/Users/Alex M/Downloads/SpamAssassinMessages/spam_2", i), 'r', encoding = "cp437") as f: x.append(f.read()) print(x)

y = [1]*len(x)

df5 = pd.DataFrame(list(zip(x, y)), columns =['text', 'target']) df5.head()

frames = [df1, df2, df3, df4, df5]

final = pd.concat(frames)

final.shape

os.makedirs("C:/Users/Alex M/Downloads/SpamAssassinMessages", exist_ok=True)
final.to_csv("C:/Users/Alex M/Downloads/SpamAssassinMessages/final.csv")

In [71]:
```
tmp = os.listdir("C:/Users/Alex M/Downloads/SpamAssassinMessages/easy_ham")
```

In [75]:

```
#sample for proper formatting
with open(os.path.join("C:/Users/Alex M/Downloads/SpamAssassinMessages/easy_ham", tmp
[3]),errors='ignore') as myfile:
    for line in myfile.readlines():
        print(line.strip('n\n'))
```

From irregulars-admin@tb.tf Thu Aug 22 14:23:39 2002
Return-Path: <irregulars-admin@tb.tf>
Delivered-To: zzzz@localhost.netnoteinc.com
Received: from localhost (localhost [127.0.0.1])
	by phobos.labs.netnoteinc.com (Postfix) with ESMTP id 9DAE147C66
	for <zzzz@localhost>; Thu, 22 Aug 2002 09:23:38 -0400 (EDT)
Received: from phobos [127.0.0.1]
	by localhost with IMAP (fetchmail-5.9.0)
	for zzzz@localhost (single-drop); Thu, 22 Aug 2002 14:23:38 +0100 (IST)
Received: from web.tb.tf (route-64-131-126-36.telocity.com
    [64.131.126.36]) by dogma.slashnull.org (8.11.6/8.11.6) with ESMTP id
    g7MDGOZ07922 for <zzzz-irr@spamassassin.taint.org>; Thu, 22 Aug 2002 14:16:24 +0100
Received: from web.tb.tf (localhost.localdomain [127.0.0.1]) by web.tb.tf
    (8.11.6/8.11.6) with ESMTP id g7MDP9I16418; Thu, 22 Aug 2002 09:25:09
    -0400
Received: from red.harvee.home (red [192.168.25.1] (may be forged)) by
    web.tb.tf (8.11.6/8.11.6) with ESMTP id g7MDO4I16408 for
    <irregulars@tb.tf>; Thu, 22 Aug 2002 09:24:04 -0400
Received: from prserv.net (out4.prserv.net [32.97.166.34]) by
    red.harvee.home (8.11.6/8.11.6) with ESMTP id g7MDFBD29237 for
    <irregulars@tb.tf>; Thu, 22 Aug 2002 09:15:12 -0400
Received: from [209.202.248.109]
    (slip-32-103-249-10.ma.us.prserv.net[32.103.249.10]) by prserv.net (out4)
    with ESMTP id <2002082213150220405qu8jce>; Thu, 22 Aug 2002 13:15:07 +0000
MIME-Version: 1.0
X-Sender: @ (Unverified)
Message-Id: <p04330137b98a941c58a8@[209.202.248.109]>
To: undisclosed-recipient: ;
From: Monty Solomon <monty@roscom.com>
Content-Type: text/plain; charset="us-ascii"
Subject: [IRR] Klez: The Virus That Won't Die
Sender: irregulars-admin@tb.tf
Errors-To: irregulars-admin@tb.tf
X-Beenthere: irregulars@tb.tf
X-Mailman-Version: 2.0.6
Precedence: bulk
List-Help: <mailto:irregulars-request@tb.tf?subject=help>
List-Post: <mailto:irregulars@tb.tf>
List-Subscribe: <http://tb.tf/mailman/listinfo/irregulars>,
    <mailto:irregulars-request@tb.tf?subject=subscribe>
List-Id: New home of the TBTF Irregulars mailing list <irregulars.tb.tf>
List-Unsubscribe: <http://tb.tf/mailman/listinfo/irregulars>,
    <mailto:irregulars-request@tb.tf?subject=unsubscribe>
List-Archive: <http://tb.tf/mailman/private/irregulars/>
Date: Thu, 22 Aug 2002 09:15:25 -0400

Klez: The Virus That Won't Die

Already the most prolific virus ever, Klez continues to wreak havoc.

Andrew Brandt
>>From the September 2002 issue of PC World magazine
Posted Thursday, August 01, 2002

The Klez worm is approaching its seventh month of wriggling across the Web, making it one of the most persistent viruses ever. And experts warn that it may be a harbinger of new viruses that use a combination of pernicious approaches to go from PC to PC.

Antivirus software makers Symantec and McAfee both report more than

2000 new infections daily, with no sign of letup at press time. The British security firm MessageLabs estimates that 1 in every 300 e-mail messages holds a variation of the Klez virus, and says that Klez has already surpassed last summer's SirCam as the most prolific virus ever.

And some newer Klez variants aren't merely nuisances--they can carry other viruses in them that corrupt your data.

...

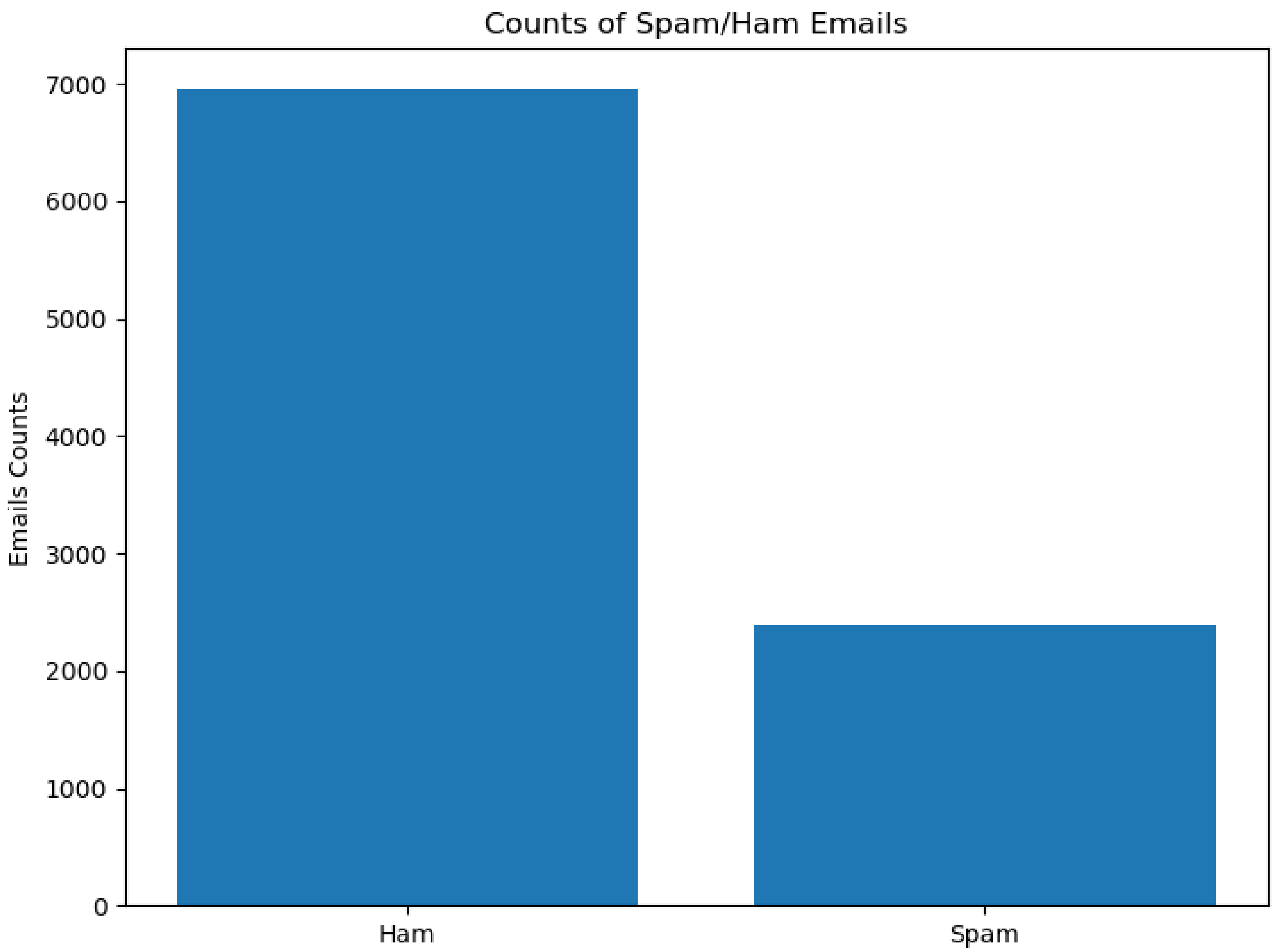
http://www.pcworld.com/news/article/0,aid,103259,00.asp

In [77]:
```
#sample for without formatting
with open(os.path.join("C:/Users/Alex M/Downloads/SpamAssassinMessages/easy_ham", tmp
[3]),errors='ignore') as myfile:
    lines = myfile.readlines()
print(lines)
```

```
['From irregulars-admin@tb.tf  Thu Aug 22 14:23:39 2002\n', 'Return-Path: <irregulars-a
dmin@tb.tf>\n', 'Delivered-To: zzzz@localhost.netnoteinc.com\n', 'Received: from localh
ost (localhost [127.0.0.1])\n', '\tby phobos.labs.netnoteinc.com (Postfix) with ESMTP i
d 9DAE147C66\n', '\tfor <zzzz@localhost>; Thu, 22 Aug 2002 09:23:38 -0400 (EDT)\n', 'Re
ceived: from phobos [127.0.0.1]\n', '\tby localhost with IMAP (fetchmail-5.9.0)\n', '\t
for zzzz@localhost (single-drop); Thu, 22 Aug 2002 14:23:38 +0100 (IST)\n', 'Received:
from web.tb.tf (route-64-131-126-36.telocity.com\n', '    [64.131.126.36]) by dogma.sla
shnull.org (8.11.6/8.11.6) with ESMTP id\n', '    g7MDGOZ07922 for <zzzz-irr@spamassass
in.taint.org>; Thu, 22 Aug 2002 14:16:24 +0100\n', 'Received: from web.tb.tf (localhos
t.localdomain [127.0.0.1]) by web.tb.tf\n', '    (8.11.6/8.11.6) with ESMTP id g7MDP9I1
6418; Thu, 22 Aug 2002 09:25:09\n', '    -0400\n', 'Received: from red.harvee.home (red
[192.168.25.1] (may be forged)) by\n', '    web.tb.tf (8.11.6/8.11.6) with ESMTP id g7M
DO4I16408 for\n', '    <irregulars@tb.tf>; Thu, 22 Aug 2002 09:24:04 -0400\n', 'Receive
d: from prserv.net (out4.prserv.net [32.97.166.34]) by\n', '    red.harvee.home (8.11.6
/8.11.6) with ESMTP id g7MDFBD29237 for\n', '    <irregulars@tb.tf>; Thu, 22 Aug 2002 0
9:15:12 -0400\n', 'Received: from [209.202.248.109]\n', '    (slip-32-103-249-10.ma.us.
prserv.net[32.103.249.10]) by prserv.net (out4)\n', '    with ESMTP id <200208221315022
0405qu8jce>; Thu, 22 Aug 2002 13:15:07 +0000\n', 'MIME-Version: 1.0\n', 'X-Sender: @ (U
nverified)\n', 'Message-Id: <p04330137b98a941c58a8@[209.202.248.109]>\n', 'To: undisclo
sed-recipient: ;\n', 'From: Monty Solomon <monty@roscom.com>\n', 'Content-Type: text/pl
ain; charset="us-ascii"\n', "Subject: [IRR] Klez: The Virus That  Won't Die\n", 'Sende
r: irregulars-admin@tb.tf\n', 'Errors-To: irregulars-admin@tb.tf\n', 'X-Beenthere: irre
gulars@tb.tf\n', 'X-Mailman-Version: 2.0.6\n', 'Precedence: bulk\n', 'List-Help: <mailt
o:irregulars-request@tb.tf?subject=help>\n', 'List-Post: <mailto:irregulars@tb.tf>\n',
'List-Subscribe: <http://tb.tf/mailman/listinfo/irregulars>,\n', '    <mailto:irregular
s-request@tb.tf?subject=subscribe>\n', 'List-Id: New home of the TBTF Irregulars mailin
g list <irregulars.tb.tf>\n', 'List-Unsubscribe: <http://tb.tf/mailman/listinfo/irregul
ars>,\n', '    <mailto:irregulars-request@tb.tf?subject=unsubscribe>\n', 'List-Archive:
<http://tb.tf/mailman/private/irregulars/>\n', 'Date: Thu, 22 Aug 2002 09:15:25 -0400\n
', '\n', "Klez: The Virus That Won't Die\n", ' \n', 'Already the most prolific virus ev
er, Klez continues to wreak havoc.\n', '\n', 'Andrew Brandt\n', '>>From the September 2
002 issue of PC World magazine\n', 'Posted Thursday, August 01, 2002\n', '\n', '\n', 'T
he Klez worm is approaching its seventh month of wriggling across \n', 'the Web, making
it one of the most persistent viruses ever. And \n', 'experts warn that it may be a har
binger of new viruses that use a \n', 'combination of pernicious approaches to go from
PC to PC.\n', '\n', 'Antivirus software makers Symantec and McAfee both report more tha
n \n', '2000 new infections daily, with no sign of letup at press time. The \n', 'Briti
sh security firm MessageLabs estimates that 1 in every 300 \n', 'e-mail messages holds
a variation of the Klez virus, and says that \n', "Klez has already surpassed last summ
er's SirCam as the most prolific \n", 'virus ever.\n', '\n', "And some newer Klez varia
nts aren't merely nuisances--they can carry \n", 'other viruses in them that corrupt yo
ur data.\n', '\n', '...\n', '\n', 'http://www.pcworld.com/news/article/0,aid,103259,00.
asp\n', '_______________________________________________\n', 'Irregulars mailing list\n
', 'Irregulars@tb.tf\n', 'http://tb.tf/mailman/listinfo/irregulars\n', '\n']
```

In [56]:
```
final = pd.read_csv(r'C:/Users/Alex M/Downloads/SpamAssassinMessages/final.csv',low_memo
ry = False)
```

In [57]:
```
fig = plt.figure()
# plt.figure(figsize=(1,1))
ax = fig.add_axes([0,0,1,1])
labels = ['Ham', 'Spam']
ax.bar(labels,final['target'].value_counts())
plt.ylabel('Emails Counts')
plt.title('Counts of Spam/Ham Emails')
plt.show()
```



In [31]:
```
final['target'].value_counts()
```

Out[31]:
```
0    6954
1    2399
Name: target, dtype: int64
```

In [32]:
```
ham_messages = final[final["target"] == 0]
spam_messages  = final[final["target"] == 1]
```

In [34]:
```
print(ham_messages.shape)
print(spam_messages.shape)
```

```
(6954, 3)
(2399, 3)
```

In [35]:
```
ham_downsample = resample(ham_messages,
             replace=True,
             n_samples=len(spam_messages),
             random_state=42)

print(ham_downsample.shape)
```

```
(2399, 3)
```

In [33]:
```
final.head
```

Out[33]:
```
<bound method NDFrame.head of       Unnamed: 0
text  target
0              0  From exmh-workers-admin@redhat.com  Thu Aug 22...      0
1              1  From Steve_Burt@cursor-system.com  Thu Aug 22 ...      0
2              2  From timc@2ubh.com  Thu Aug 22 13:52:59 2002\n...      0
3              3  From irregulars-admin@tb.tf  Thu Aug 22 14:23:...      0
4              4  From Stewart.Smith@ee.ed.ac.uk  Thu Aug 22 14:...      0
...          ...                                                ...    ...
9348        1393  From tba@insiq.us  Wed Dec  4 11:46:34 2002\nR...      1
9349        1394  Return-Path: <raye@yahoo.lv>\nReceived: from u...      1
9350        1395  From cweqx@dialix.oz.au  Tue Aug  6 11:03:54 2...      1
9351        1396  From ilug-admin@linux.ie  Wed Dec  4 11:52:36 ...      1
9352        1397  mv 00001.317e78fa8ee2f54cd4890fdc09ba8176 0000...      1

[9353 rows x 3 columns]>
```

In [38]:
```
final = pd.concat([ham_downsample, spam_messages])
print(final.shape)
```

```
(4798, 3)
```

In [39]:
```
final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4798 entries, 860 to 9352
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  4798 non-null   int64
 1   text        4798 non-null   object
 2   target      4798 non-null   int64
dtypes: int64(2), object(1)
memory usage: 149.9+ KB
```

In [ ]:
```
# Naive Bayes
```

In [41]:
```
x = final['text']
y = final['target']
```

In [42]:
```
x.head()
```

Out[42]:
```
860     From fork-admin@xent.com  Thu Sep 26 11:04:45 ...
5390    From ilug-admin@linux.ie  Mon Aug 12 11:07:18 ...
5226    From ilug-admin@linux.ie  Wed Jul 31 12:31:04 ...
5191    From ilug-admin@linux.ie  Mon Jul 29 11:28:30 ...
3772    From rpm-list-admin@freshrpms.net  Mon Sep 23 ...
Name: text, dtype: object
```

In [43]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=4
2)
```

In [44]:
```
X_train = X_train.str.replace("\n"," ")
X_test = X_test.str.replace("\n"," ")
```

In [45]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
the_count = CountVectorizer()
X = the_count.fit_transform(X_train)
```

In [46]:
```
vocab = the_count.vocabulary_
rev = {j:i for i,j in vocab.items()}
```

In [47]:
```
nb = MultinomialNB()
nb.fit(X.toarray(),y_train)
```

Out[47]:
```
▾ MultinomialNB

MultinomialNB()
```

In [48]:
```
x2 = the_count.transform(X_test)
prednb = nb.predict(x2.toarray())
```

In [49]:
```
print(classification_report(y_test,prednb))
```

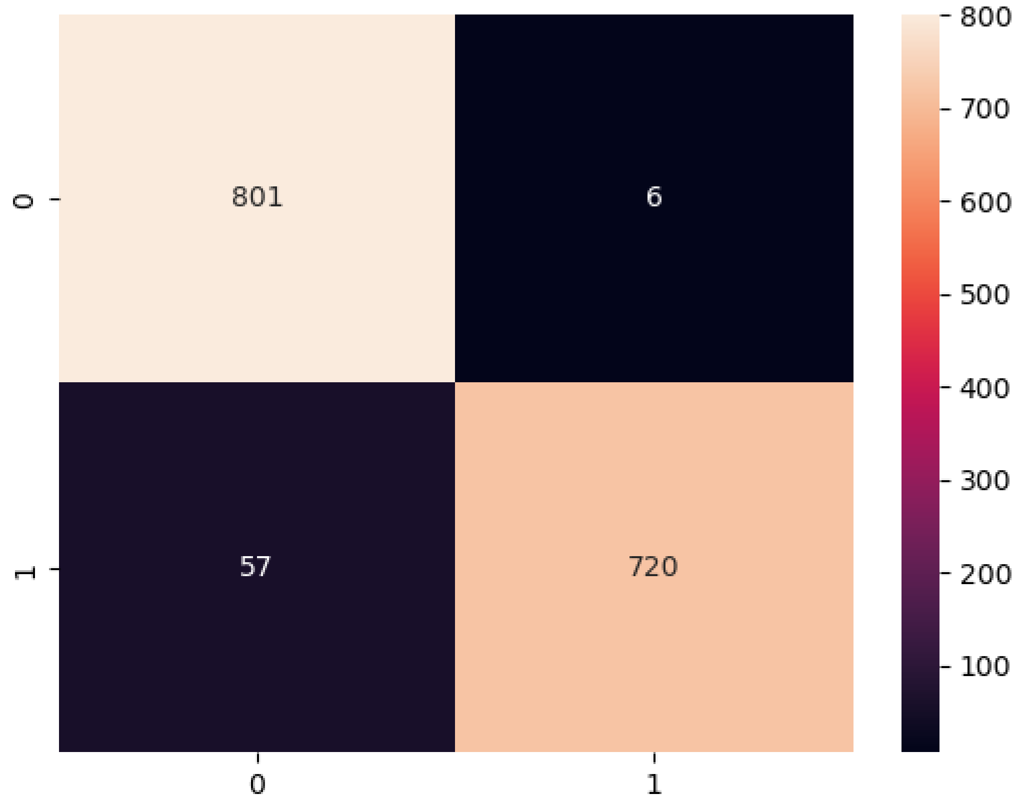
```
              precision    recall  f1-score   support

           0       0.93      0.99      0.96       807
           1       0.99      0.93      0.96       777

    accuracy                           0.96      1584
   macro avg       0.96      0.96      0.96      1584
weighted avg       0.96      0.96      0.96      1584
```

In [50]:
```
cm = confusion_matrix(y_test,prednb)
sns.heatmap(cm, annot=True, fmt="d")
```

Out[50]: <AxesSubplot:>



In [ ]:
```
# Kmeans Cluster
```

In [52]:

```python
#from documentation https://scikit-learn.org/stable/auto_examples/text/plot_document_clu
stering.html#sphx-glr-auto-examples-text-plot-document-clustering-py
#fits and produces various metrics shown below

labels = y
unique_labels, category_sizes = np.unique(labels, return_counts=True)
true_k = unique_labels.shape[0]

evaluations = []
evaluations_std = []


def fit_and_evaluate(km, X, name=None, n_runs=5):
    name = km.__class__.__name__ if name is None else name

    train_times = []
    scores = defaultdict(list)
    for seed in range(n_runs):
        km.set_params(random_state=seed)
        t0 = time()
        km.fit(X)
        train_times.append(time() - t0)
        scores["Homogeneity"].append(metrics.homogeneity_score(labels, km.labels_))
        scores["Completeness"].append(metrics.completeness_score(labels, km.labels_))
        scores["V-measure"].append(metrics.v_measure_score(labels, km.labels_))
        scores["Adjusted Rand-Index"].append(
            metrics.adjusted_rand_score(labels, km.labels_)
        )
        scores["Silhouette Coefficient"].append(
            metrics.silhouette_score(X, km.labels_, sample_size=2000)
        )
    train_times = np.asarray(train_times)

    print(f"clustering done in {train_times.mean():.2f} ± {train_times.std():.2f} s ")
    evaluation = {
        "estimator": name,
        "train_time": train_times.mean(),
    }
    evaluation_std = {
        "estimator": name,
        "train_time": train_times.std(),
    }
    for score_name, score_values in scores.items():
        mean_score, std_score = np.mean(score_values), np.std(score_values)
        print(f"{score_name}: {mean_score:.3f} ± {std_score:.3f}")
        evaluation[score_name] = mean_score
        evaluation_std[score_name] = std_score
    evaluations.append(evaluation)
    evaluations_std.append(evaluation_std)
```

In [53]:
```python
vectorizer = TfidfVectorizer(
    max_df=0.5,
    min_df=5,
    stop_words="english",
)

X_tfidf = vectorizer.fit_transform(x)

print(f"n_samples: {X_tfidf.shape[0]}, n_features: {X_tfidf.shape[1]}")
```

```
n_samples: 4798, n_features: 16942
```

In [54]:
```python
kmeans = KMeans(
    n_clusters=2,
    max_iter=100,
    n_init=5,
)
```

In [55]:
```python
fit_and_evaluate(kmeans, X_tfidf, name="KMeans\non tf-idf vectors")
```

```
clustering done in 1.04 ± 0.12 s
Homogeneity: 0.166 ± 0.000
Completeness: 0.217 ± 0.000
V-measure: 0.188 ± 0.000
Adjusted Rand-Index: 0.140 ± 0.000
Silhouette Coefficient: 0.057 ± 0.002
```

In [62]:
```python
fig, (ax0, ax1) = plt.subplots(ncols=2, figsize=(16, 6), sharey=True)

df = pd.DataFrame(evaluations[::-1]).set_index("estimator")
df_std = pd.DataFrame(evaluations_std[::-1]).set_index("estimator")

df.drop(
    ["train_time"],
    axis="columns",
).plot.barh(ax=ax0, xerr=df_std)
ax0.set_xlabel("Clustering scores")
ax0.set_ylabel("")

df["train_time"].plot.barh(ax=ax1, xerr=df_std["train_time"])
ax1.set_xlabel("Clustering time (s)")
plt.tight_layout()
```