```
'''
Steps for your next Capstone:

1. Go out and find a dataset of interest. It could be from one of the recommended
resources or some other aggregation. Or it could be something that you scraped
yourself. Just make sure that it has lots of variables, including an outcome of
interest to you.

2. Explore the data. Get to know the data. Spend a lot of time going over its
quirks. You should understand how it was gathered, what's in it, and what the
variables look like.

3. Model your outcome of interest. You should try several different approaches
and really work to tune a variety of models before using the model evaluation
techniques to choose what you consider to be the best performer. Make sure to
think about explanatory versus predictive power, and experiment with both.


Execute the three tasks above in a Jupyter Notebook that you will submit to the
Thinkful team for grading.

PRESENTATION:
Next, to prepare for your presentation, create a slide deck and a 15-minute
presentation that guides viewers through your model. Be sure to cover a few
specific topics:

- A specified research question that your model addresses

- How you chose your model specification and what alternatives you compared it to

- The practical uses of your model for an audience of interest

- Any weak points or shortcomings of your model
'''
```

Spam Detector

Overview

Cyber-security is the practice of protecting systems and entities from outside forces designed to infiltrate, change, and gleam sensitive information from them. Today, companies have begun to practically invest in professionals and, in larger agencies, entire departments familiar with addressing, preventing, and managing these crises situations. Such professionals can include a team made up of security analysts, IT administrators, and/or a data scientists who work collaboratively to address problems like:

- system cleaning and updating
- problem cleaning, i.e. malware
- internal misuse(access privileges)
- and applying spam and/or phishing filters

PROPOSAL

The head of Cybersecurity at a prospective company is creating an innovative new software that identifies spam. How can you help them create an effective filter?

The Data

The data for this project consists of csv file of two folders; each folder containing emails of spam or ham(not spam). Each text file of the folders was iterated through to create a DataFrame, and written to the csv file.

Methods

For my research, I obtained a public dataset from Kaggle, "spam_ham_dataset.csv", which consisted of a combined csv file of two sets of data; one for spam-labeled email subjects and the other for non-spam-labeled email subjects. They were then written to one large csv file. This was for both ease of accessibility and analysis of both email types(https://raw.githubusercontent.com/mwarnsle1/Capstone_2_Spam_Detector/main/spam_ham_dataset.csv). The combined dataset catalogued the categorized emails received by users. Before the model creation, I initially applied a number of exploratory analyses; such as first applying a label to the user column, which - was previously unlabeled - then, the code was further cleaned and the column values were confirmed. I then used one-hot encoding and added a key for the new target variable, so that the variable - which previously held strings in its column - could be fed properly into the model.

I applied some preliminary visualizations to check the initial distribution of the variables; by themselves and in relation to each other. I then built two different models to observe their affect and accuracy on the dataset. The first was built was a Random Forest Classifier to the dataset. After observing the results, I applied some Natural Language Processing(NLP) techniques to the dataset. Specifically, word embedding and word vectorization were used as a better predictive model and to filter user emails. The results are discussed in the next section.

Results

The random forest ran successfully, but was not always able to detect spam emails. This may be chiefly due to the higher number of 'Ham' variables being in the dataset. A predictive model was able to be created using NLP techniques. Using word embedding, scatter plotting was used to visualize dots annotated with the words from the text. From the second NLP model, a predictive model was successfully created using word vectorization; where each categorization was used to detect and filter new emails.

Discussion & Recommendation

A closer look at the data indicates that Spam Detection/Spam Filters are best run on NLP word vectorization models. While random forests are very strong performers and work well on both classification and regression, as shown, they can't predict well outside the sample and will only return values within a range already seen before. Drawbacks of the dataset were having an uneven(higher) number of non-spam("Ham") emails; which could also negatively affect other kinds of predictive modeling. This imbalance was overcome for the NLP model, however. Future iterations would implement measures to balance out the dataset by under-sampling the "Ham" variable, and implement cross-validation.

Package Installation

```
!pip install -q kaggle
```

```
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving spam_mails_dataset.zip to spam_mails_dataset (1).zip
{'spam_mails_dataset.zip': b'PK\x03\x04-\x00\x00\x00\x08\x00\x85\x08RO_*\xaeL\xff\xff\x
```

```
#creating a kaggle folder/directory:

! mkdir ~/.kaggle
```

```
mkdir: cannot create directory ‘/root/.kaggle’: File exists
```

```
!touch ~/.kaggle/kaggle.json
```

```
api_token = {"username":"mwarnsle1","key":"91702627b2baaab089b5771b2bbbd38a"}

import json

with open('/root/.kaggle/kaggle.json', 'w') as file:
    json.dump(api_token, file)
```

```
! chmod 600 ~/.kaggle/kaggle.json
! kaggle datasets list
```

```
Warning: Looks like you're using an outdated API Version, please consider updating (serv
ref                                                          title
-----------------------------------------------------------  ----------------------------
gpreda/reddit-vaccine-myths                                  Reddit Vaccine Myths
crowww/a-large-scale-fish-dataset                            A Large Scale Fish Dataset
imsparsh/musicnet-dataset                                    MusicNet Dataset
promptcloud/careerbuilder-job-listing-2020                   Careerbuilder Job Listing 20
dhruvildave/wikibooks-dataset                                Wikibooks Dataset
mathurinache/twitter-edge-nodes                              Twitter Edge Nodes
fatiimaezzahra/famous-iconic-women                           Famous Iconic Women
nickuzmenkov/nih-chest-xrays-tfrecords                       NIH Chest X-rays TFRecords
alsgroup/end-als                                             End ALS Kaggle Challenge
simiotic/github-code-snippets                                GitHub Code Snippets
coloradokb/dandelionimages                                   DandelionImages
mathurinache/the-lj-speech-dataset                           The LJ Speech Dataset
imsparsh/accentdb-core-extended                              AccentDB - Core & Extended
stuartjames/lights                                           LightS: Light Specularity Da
nickuzmenkov/ranzcr-clip-kfold-tfrecords                     RANZCR CLiP KFold TFRecords
landrykezebou/lvzhdr-tone-mapping-benchmark-dataset-tmonet   LVZ-HDR Tone Mapping Benchma
datasnaek/youtube-new                                        Trending YouTube Video Stati
zynicide/wine-reviews                                        Wine Reviews
residentmario/ramen-ratings                                  Ramen Ratings
datasnaek/chess                                              Chess Game Dataset (Lichess)
```

```
#CHOOSING MY KAGGLE DATASET(API):

!kaggle datasets download -d venky73/spam-mails-dataset --force
```

```
Downloading spam-mails-dataset.zip to /content
  0% 0.00/1.86M [00:00<?, ?B/s]
100% 1.86M/1.86M [00:00<00:00, 61.1MB/s]
```

```
! unzip spam-mails-dataset.zip
```

```
Archive:  spam-mails-dataset.zip
replace spam_ham_dataset.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
```

```
#IMPORT PACKAGES:

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
from sklearn.decomposition import PCA
from sklearn import metrics
from sqlalchemy import create_engine
import warnings
warnings.filterwarnings("ignore")


#LOAD THE DATASET:

spam_df = pd.read_csv('spam_ham_dataset.csv')
```

Exploratory Data Analysis

```
#getting a look at the dataset:
spam_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5171 entries, 0 to 5170
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  5171 non-null   int64 
 1   label       5171 non-null   object
 2   text        5171 non-null   object
 3   label_num   5171 non-null   int64 
dtypes: int64(2), object(2)
memory usage: 161.7+ KB
```

```
spam_df.head()
```

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

```
spam_df.shape
```

```
(5171, 4)
```

```
#establishing the amount of null values in the dataset:
spam_df.isnull().sum()
```

```
Unnamed: 0    0
label         0
text          0
label_num     0
dtype: int64
```

```
#counting the percentage of null values in the dataset:
spam_df.isnull().sum()*100/spam_df.isnull().count()
```

```
Unnamed: 0    0.0
label         0.0
text          0.0
label_num     0.0
dtype: float64
```

```
#Giving a label to the first row('Unnamed: 0') which is uncallable > ('Client') first:

spam_df.columns = spam_df.columns.str.replace('Unnamed: 0', 'Client')
spam_df.head()
```

	Client	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

```
#The value_counts method() will be used to check the # of values (3672 ham/1499 spam; 5

spam_df.label.value_counts()
#there are no null values in the 'label' feature, b/c there are 5171 rows in all
```

```
ham     3672
spam    1499
Name: label, dtype: int64
```

```
spam_df.label_num.value_counts()

#there are no null values in this feature, either
```

```
0    3672
1    1499
Name: label_num, dtype: int64
```

```
spam_df.Client.value_counts()
spam_df.Client.value_counts()

#There don't appear to be any null values in this column, as there are 5171 rows in the
```

```
2047    1
537     1
4663    1
2616    1
569     1
       ..
1190    1
3239    1
1194    1
3243    1
0       1
Name: Client, Length: 5171, dtype: int64
```

```
spam_df.text.value_counts()

#since technically no null values exist when the method isnull() is run, this
#indicates that there is a different kind of value in the text column that is not
#visibly titled, but still has a minimal value of 'Subject: \r\n' when selected
#in the spreadsheet
```

```
Subject: calpine daily gas nomination\r\n>\r\nricky a . archer\r\nfuel supply\r\n700 lou
Subject: \r\n
Subject: we ' ve found a school for you !\r\n
Subject: \r\nthis week only : f . ree gen . erlc vlag . ra\r\ncover the shipping , and w
Subject: you can be smart !\r\n

Subject: just cents on the dollar : windows xp , office xp , norton system works , adobe
Subject: ena sales on hpl\r\njust to update you on this project ' s status :\r\nbased on
Subject: quick way to buy soft - ware\r\nvariety of top manufacturer software at wholesa
Subject: re : 1601\r\nyes , that sounds great ! !\r\n- - - - - original message - - - -
Subject: meter 6315 , purch from torch / rally , october\r\ndaren ,\r\ni show that you e
Name: text, Length: 4993, dtype: int64
```

```
#Creating a new column that tracks the number of characters per subject:

spam_df['text_length'] = spam_df['text'].str.len()
```

```
spam_df['text_length'].value_counts().sort_index()
#this gives further indication of the amount of empty-seeming cells, with 11 characters
```

```
11       16
15        1
18        1
19        1
25        1
         ..
14716     1
16312     1
```

```
16338     1
22073     1
32258     1
Name: text_length, Length: 2112, dtype: int64
```

```
#Finding the minimum character amount discoverable - of note, characters at length 11 i

spam_df[spam_df['text_length'] == 11]
```

	Client	label	text	label_num	text_length
154	4592	spam	Subject: \r\n	1	11
182	4727	spam	Subject: \r\n	1	11
296	4690	spam	Subject: \r\n	1	11
363	4682	spam	Subject: \r\n	1	11
1130	4136	spam	Subject: \r\n	1	11
1279	3749	spam	Subject: \r\n	1	11
1369	4282	spam	Subject: \r\n	1	11
2184	4600	spam	Subject: \r\n	1	11
2538	4705	spam	Subject: \r\n	1	11
2665	4062	spam	Subject: \r\n	1	11
2680	4254	spam	Subject: \r\n	1	11
2903	4141	spam	Subject: \r\n	1	11
3006	4962	spam	Subject: \r\n	1	11
3610	4595	spam	Subject: \r\n	1	11
4081	4153	spam	Subject: \r\n	1	11
4748	4683	spam	Subject: \r\n	1	11

```
spam_df[spam_df['text_length'] < 18]
```

	Client	label	text	label_num	text_length
154	4592	spam	Subject: \r\n	1	11
182	4727	spam	Subject: \r\n	1	11
296	4690	spam	Subject: \r\n	1	11
363	4682	spam	Subject: \r\n	1	11
1130	4136	spam	Subject: \r\n	1	11
1279	3749	spam	Subject: \r\n	1	11
1369	4282	spam	Subject: \r\n	1	11
1933	4829	spam	Subject: note\r\n	1	15
2184	4600	spam	Subject: \r\n	1	11
2538	4705	spam	Subject: \r\n	1	11
2665	4062	spam	Subject: \r\n	1	11

```
spam_df[spam_df['text_length'] < 50]
```

	Client	label	text	label_num	text_length
71	4261	spam	Subject: - get a dell laptop computer free !\r\n	1	46
144	4764	spam	Subject: penls enlarg 3 ment pllls\r\n	1	36
154	4592	spam	Subject: \r\n	1	11
162	3748	spam	Subject: holiday e - cards\r\ngbhzivjwl	1	37
180	4460	spam	Subject: \r\nwant a rolex watch ?\r\n	1	33
...	...	...	...	...	...
4748	4683	spam	Subject: \r\n	1	11
4827	4231	spam	Subject: - want a new laptop ? - get one free ...	1	49
4915	3899	spam	Subject: get your viagr $ a today !\r\n	1	37
5118	3814	spam	Subject: suprise your spouse ! vi [agra !\r\n	1	44
5134	3894	spam	Subject: private vl @ gra\r\nremove me	1	36

85 rows × 5 columns

```
#We now see that since adding the additional text-based column, the shape of the DataFr

spam_df.shape
```

```
    (5171, 5)
```

```
spam_df.head()
spam_df.head()
```

	Client	label	text	label_num	text_length
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0	327
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0	97
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0	2524
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1	414
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0	336

```
# Make sure the number of rows divides evenly into four samples.
col_names = ["Client", "text", "label_num", "text_length", "label"]
```

```
# Break into a set of features and a variable for the known outcome.
feature_cols = ["label_num", "text_length"]
#y = spam_df["Client", "text", "label_num", "text_length"] #put in actual label/feature
X = spam_df.label  #put in actual independent variable name
```

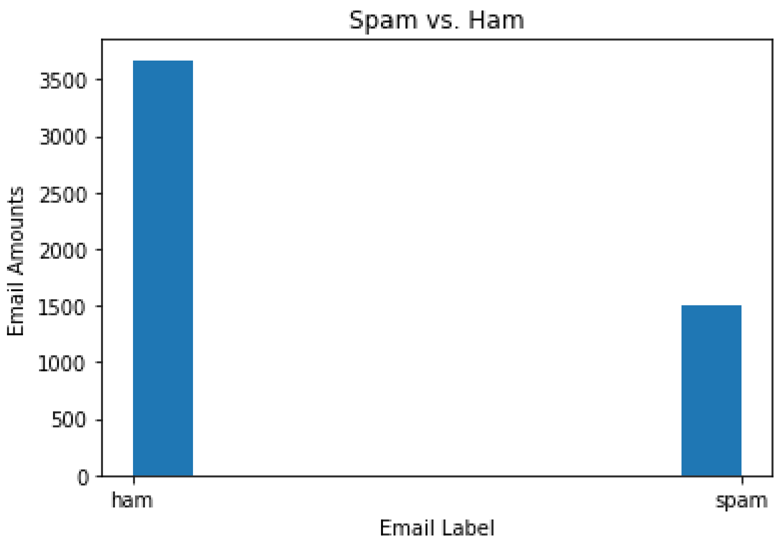
```
# Binarize y so that 1 means spam and 0 means no ham(i.e. not spam/regular mail).
#feature_cols = np.where(feature_cols > 0, 0, 1)

'''
Here, X will represent your features and 'feature_cols' will hold the labels. If
 'feature_cols' is equal to 1, that indicates that the corresponding email is
 spam. And if 'feature_cols' is equal to 0, then the email is not spam.
'''
```

```
'\nHere, X will represent your features and 'feature_cols' will hold the labels. If\n
'feature_cols' is equal to 1, that indicates that the corresponding email is \n spam. A
nd if 'feature_cols' is equal to 0, then the email is not spam.\n'
```

Preliminary Visualizations

Investigating how your target is distributed will help you understand the relationship between the target and the features. It's also useful for discovering some potential problems with the model.

```
#The below plot shows there's almost twice as many regular emails than Spam emails:

plt.hist(spam_df.label)
plt.title("Spam vs. Ham")
plt.xlabel("Email Label")
plt.ylabel("Email Amounts")
plt.show()
```



```
#One-hot encoding will be used by calling the get_dummy() function for the ham and spam
#only include one value, the get_dummies() function will create one dummy (indicator) v

pd.get_dummies(spam_df['label'].head())
```

	ham	spam
0	1	0
1	1	0
2	1	0
3	0	1
4	1	0

```
spam5_df = spam_df
```

```
#Now, we'll create another variable, 'spam_filter', that will hold the encoded target v

spam5_df["spam_filter"] = pd.get_dummies(spam_df.label, drop_first=True)
```

```
#showing how another column - spam_filter - was added, that has encoded the label colum

spam5_df.head()
```

	Client	label	text	label_num	text_length	spam_filter
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n	0	327	0

```
#showing the encoded column:

spam5_df["spam_filter"].head()
```

```
0    0
1    0
2    0
3    1
4    0
Name: spam_filter, dtype: uint8
```

```
#shape of the new DataFrame - now, with 6 columns(Client, text, label, label_num, text_

spam5_df.shape
```

```
(5171, 6)
```

DESCRIPTIVE STATS + PRELIMINARY VISUALIZATIONS

```
spam5_df.info()
```

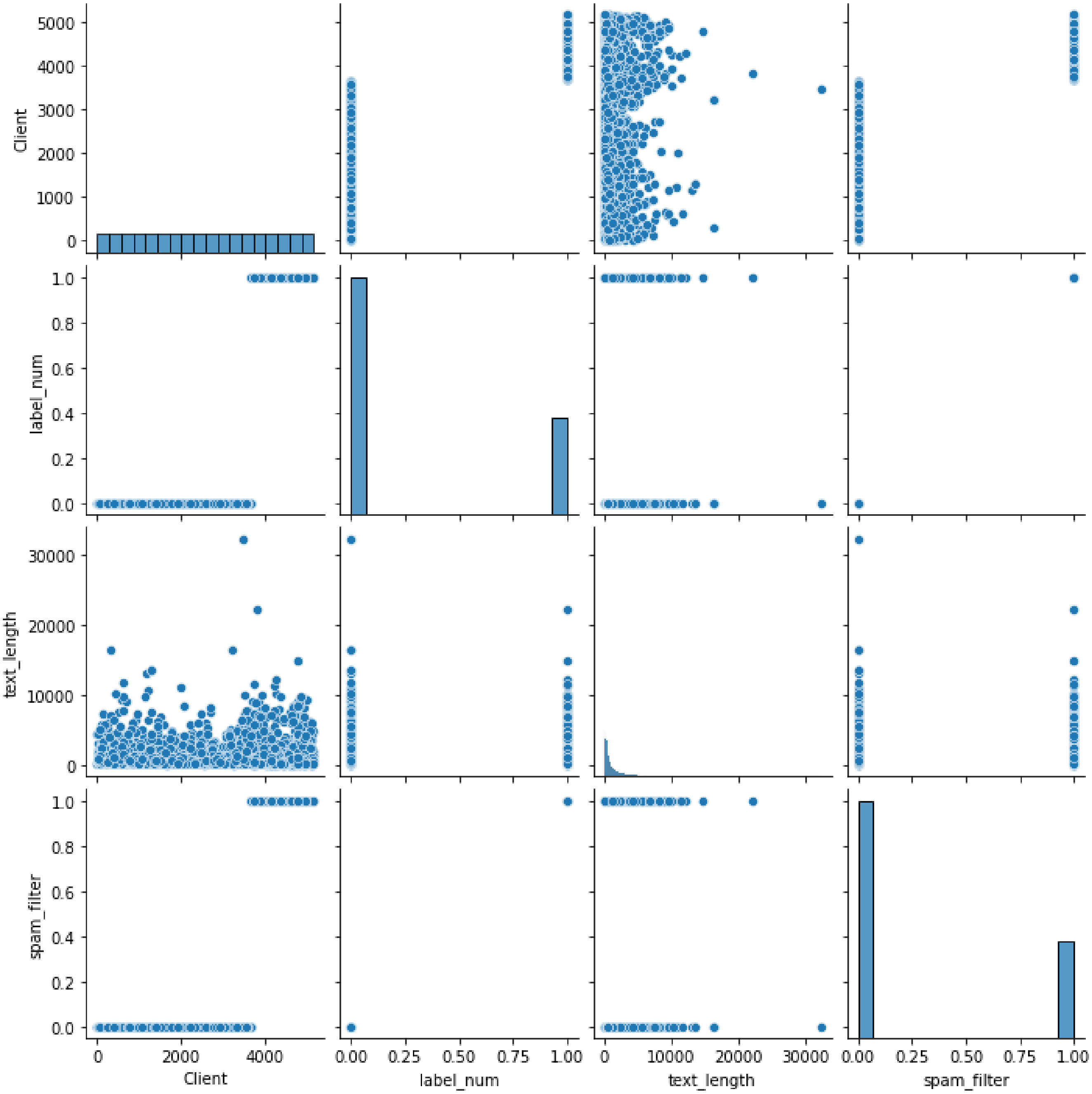
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5171 entries, 0 to 5170
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Client       5171 non-null   int64 
 1   label        5171 non-null   object
 2   text         5171 non-null   object
 3   label_num    5171 non-null   int64 
 4   text_length  5171 non-null   int64 
 5   spam_filter  5171 non-null   uint8 
dtypes: int64(3), object(2), uint8(1)
memory usage: 207.2+ KB
```

```
spam5_df.describe()
```

	Client	label_num	text_length	spam_filter
count	5171.000000	5171.000000	5171.000000	5171.000000
mean	2585.000000	0.289886	1048.391994	0.289886
std	1492.883452	0.453753	1528.513435	0.453753

```
sns.pairplot(spam5_df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f05dbaf3350>
```

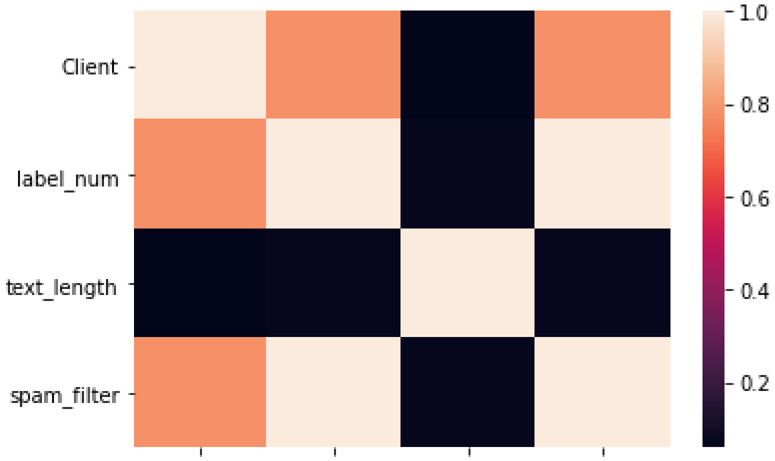


```
spam5_df.corr()
```

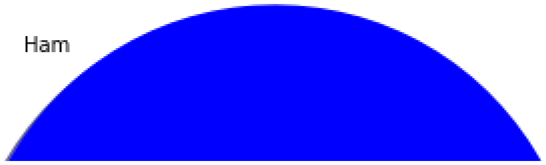
	Client	label_num	text_length	spam_filter
Client	1.000000	0.785847	0.060406	0.785847
label_num	0.785847	1.000000	0.073101	1.000000
text_length	0.060406	0.073101	1.000000	0.073101
spam_filter	0.785847	1.000000	0.073101	1.000000

```
sns.heatmap(spam5_df.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f05d478aa10>
```



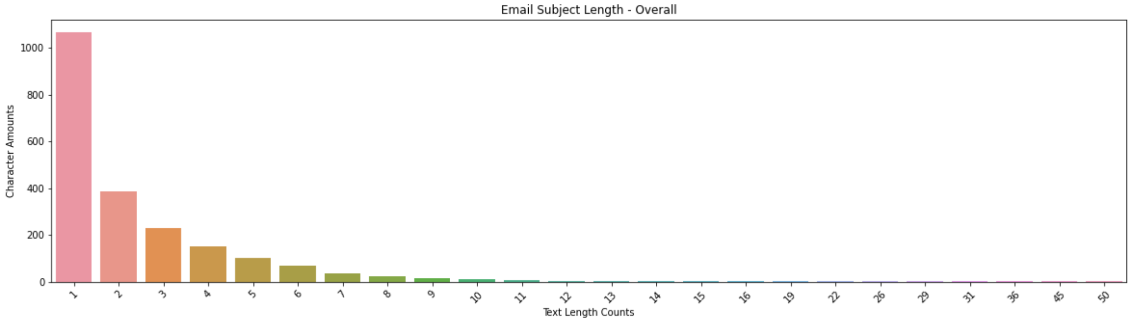
```
plt.figure(figsize=(7,7))
size=spam5_df['spam_filter'].value_counts()
label=['Ham','Spam']
color=['Blue','Pink']
explode=[0,0.1]
plt.pie(size,explode=explode,labels=label,colors=color,shadow=True)
plt.legend()
plt.show()
```



```
#the new independent/target variable's column is still 5171 rows long:

spam5_df['spam_filter'].shape
```

```
    (5171,)
```

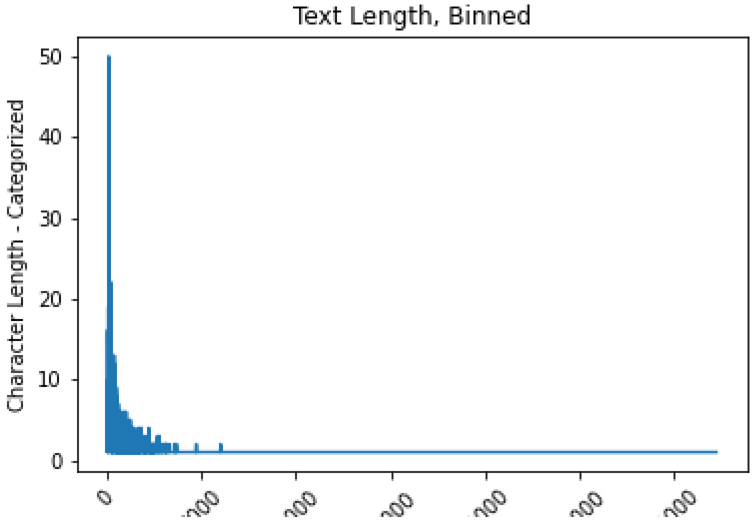
```
plt.figure(figsize=(20,5))

sns.countplot(spam5_df.groupby(['text_length']).count()['text'])

plt.title("Email Subject Length - Overall")
plt.xlabel("Text Length Counts")
plt.ylabel("Character Amounts")
plt.xticks(rotation=45)
plt.show()
```



```
#spam5_df.groupby('text_length').sum()

df = spam5_df.groupby(['text_length']).count()['text']
print(df)

# plot the result
df.plot()
```

```
plt.title("Text Length, Binned")
plt.xlabel("Text Length")
plt.ylabel("Character Length - Categorized")
plt.xticks(rotation=45)
plt.show()
```

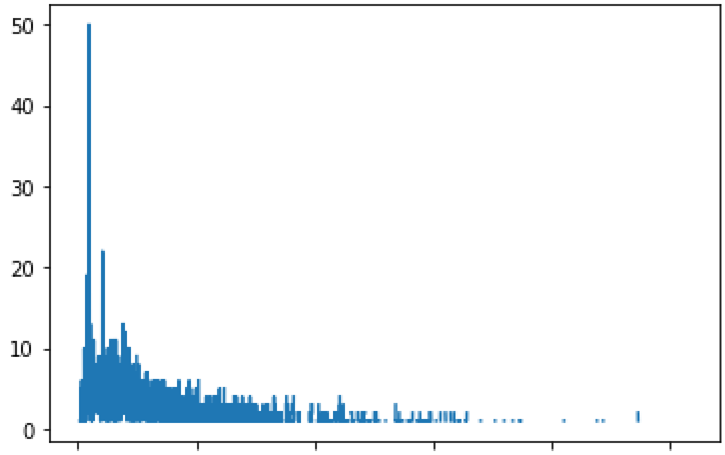
```
text_length
11       16
15        1
18        1
19        1
25        1
         ..
14716     1
16312     1
16338     1
22073     1
32258     1
Name: text, Length: 2112, dtype: int64
```



```
#Create another feature, based off the text length count:

spam5_df['text_counts'] = spam5_df.groupby(['text_length']).count()['text']
print(spam5_df['text_counts'])
```

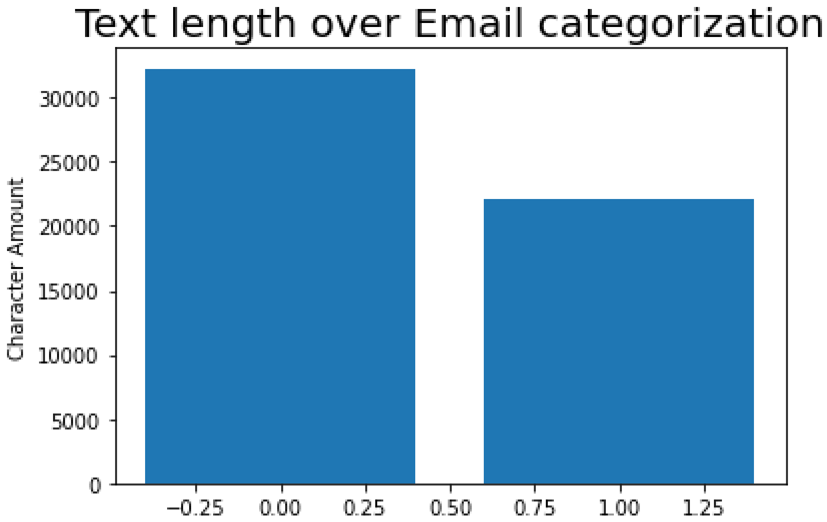
```
0      NaN
1      NaN
2      NaN
3      NaN
4      NaN
        ..
5166   NaN
5167   NaN
5168   NaN
5169   NaN
5170   NaN
Name: text_counts, Length: 5171, dtype: float64
```

```
df = spam5_df['text_counts']

# plot the result
df.plot()
plt.show()
```



```
plt.bar(spam5_df['spam_filter'],spam5_df['text_length'])
plt.title('Text length over Email categorization',fontsize=20)
plt.xlabel('Email Category')
plt.ylabel('Character Amount')
```

```
Text(0, 0.5, 'Character Amount')
```



Adding Features:

```
X = spam5_df.spam_filter
X = X.values.reshape(1, -1)

col_names = ["Client", "text", "label_num", "text_length", "label", "spam_filter"]
```

```
# Break into a set of features and a variable for the known outcome.
y = ["text_length", "text_counts", "text"]
#y = spam_df["Client", "text", "label_num", "text_length"] #put in actual label/feature

scaler = StandardScaler()
X_std = StandardScaler().fit_transform(X)

# Data frame to store features and predicted cluster memberships.
ypred = pd.DataFrame()


#the shape of the target variable is one column with 5171 rows:
X.shape
```

```
(1, 5171)
```

```
X_std.shape
```

```
(1, 5171)
```

Applying Random Forest technique to the dataset:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(spam5_df["text"],spam5_df["spam_filter
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer()
vect.fit(X_train)
X_train_df = vect.transform(X_train)
X_test_df = vect.transform(X_test)
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
model = RandomForestClassifier()
model.fit(X_train_df,y_train)

target = model.predict(X_test_df)
accuracy_score(y_test,target)
```

```
0.971042471042471
```

```
from joblib import dump, load
dump(model, 'model.joblib')
dump(vect, 'vector.joblib')
```

```
['vector.joblib']
```

```
model = load('model.joblib')
vect = load('vector.joblib')


def is_spam(inp = ["FREE FREE FREE FREE"]):
    if model.predict(vect.transform(inp))[0] == "spam":
        return True
    else:
        return False


print(is_spam(inp = [\
                     """Online Social Media platforms, such as Facebook and Twitter, en
```

```
False
```

```
print(is_spam(inp = ["""\
Congratulations You have won 10000$. FREE FREE FREE .Come and collect.
"""]))
```

```
False
```

```
print(is_spam(inp = ["""\
get a dell laptop computer free
"""]))
```

```
False
```

```
print(is_spam(inp = ["""\
neon week 8\r\n- experiencing god
"""]))
```

```
False
```

```
spam5_df.head()
```

	Client	label	text	label_num	text_length	spam_filter	text_
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0	327	0	
			Subject: hpl nom for				

Building NLP Models using Word Vectorization:

Creating a Natural Language Processing algorithm to further convert the text column:

```
!pip install top2vec
```

```
Requirement already satisfied: top2vec in /usr/local/lib/python3.7/dist-packages (1.0.26
Requirement already satisfied: numpy>=1.20.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: hdbscan>=0.8.27 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: umap-learn>=0.5.1 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: wordcloud in /usr/local/lib/python3.7/dist-packages (from
Collecting gensim<4.0.0
  Downloading gensim-3.8.3-cp37-cp37m-manylinux1_x86_64.whl (24.2 MB)
     |████████████████████████████████| 24.2 MB 101 kB/s
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from to
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: joblib>=1.0 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: scikit-learn>=0.20 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: cython>=0.27 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: pynndescent>=0.5 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: numba>=0.49 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: llvmlite<0.35,>=0.34.0.dev0 in /usr/local/lib/python3.7/d
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: pillow in /usr/local/lib/python3.7/dist-packages (from wo
Installing collected packages: gensim
  Attempting uninstall: gensim
    Found existing installation: gensim 4.0.1
    Uninstalling gensim-4.0.1:
      Successfully uninstalled gensim-4.0.1
Successfully installed gensim-3.8.3
```

```
!pip install --upgrade gensim
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.7/dist-packages (3.8.3)
Collecting gensim
  Using cached gensim-4.0.1-cp37-cp37m-manylinux1_x86_64.whl (23.9 MB)
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.7/dist-packag
Installing collected packages: gensim
  Attempting uninstall: gensim
    Found existing installation: gensim 3.8.3
    Uninstalling gensim-3.8.3:
      Successfully uninstalled gensim-3.8.3
ERROR: pip's dependency resolver does not currently take into account all the packages t
```

```
top2vec 1.0.26 requires gensim<4.0.0, but you have gensim 4.0.1 which is incompatible.
Successfully installed gensim-4.0.1
```

```
!pip install python-Levenshtein
```

```
Requirement already satisfied: python-Levenshtein in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (fro
```

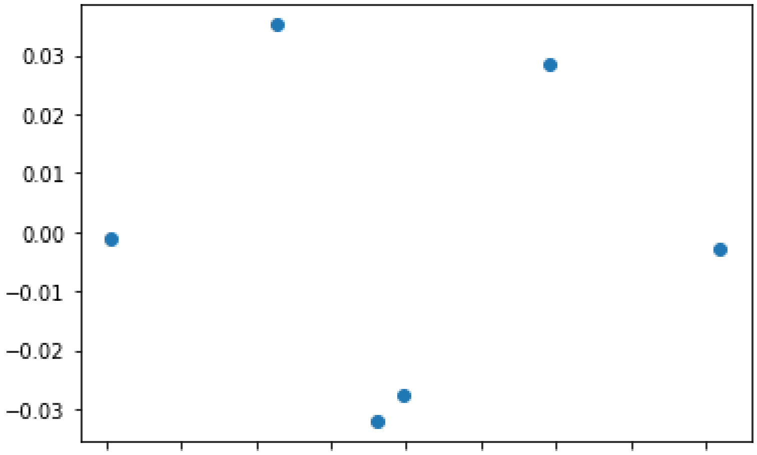
```
from gensim.test.utils import common_texts
from gensim.models import Word2Vec

model = Word2Vec(sentences=common_texts, vector_size=100, window=5, min_count=1, worker
model.save("word2vec.model")


# define training data
sentences = [['Subject: looking for medication ? we ` re the best source .', 'Subject:

# train model
#X = model[model.wv.vocab]
model = Word2Vec(sentences, min_count=1)

# summarize the loaded model
print(model)

# summarize vocabulary
words = list(model.wv.key_to_index)
print(words)

# access vector for one word
#print(model['sentences'])

# save model
model.save('model.bin')

# load model
new_model = Word2Vec.load('model.bin')
print(new_model)
```

```
Word2Vec(vocab=6, vector_size=100, alpha=0.025)
['Subject: expense report receipts not received', 'Subject: on - call notes', 'Subject:
Word2Vec(vocab=6, vector_size=100, alpha=0.025)
```

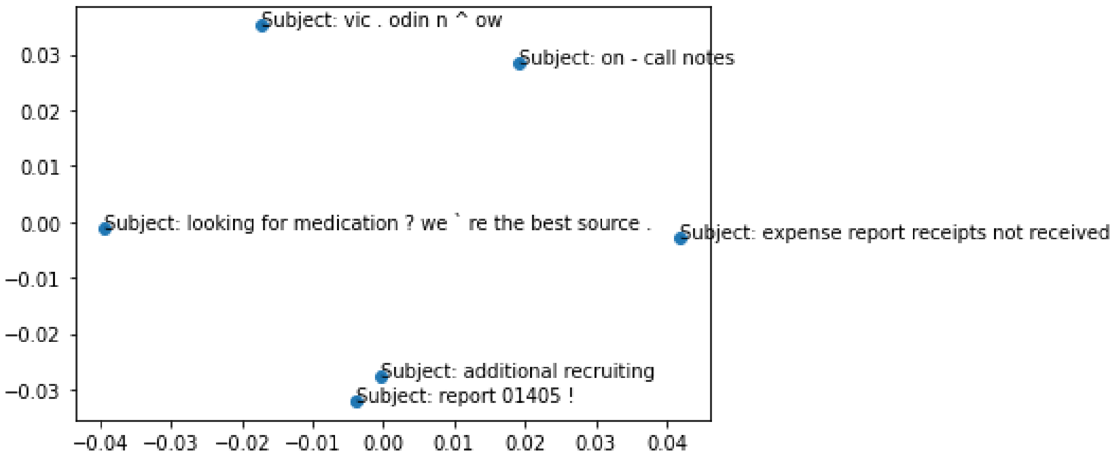
```
#retrieving all of the vectors from a trained modeL:
```

```
X = model.wv[model.wv.key_to_index]


#creating a 2-dimensional PCA model of the word vectors using the scikit-learn PCA clas
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
result = pca.fit_transform(X)


#plotting the projection using matplotlib, pulling out the two dimensions as x and y co
from matplotlib import pyplot

pyplot.scatter(result[:, 0], result[:, 1])
```

```
<matplotlib.collections.PathCollection at 0x7f05dbb646d0>
```



```
#Creating a scatter plot with the dots annotated with the Sentences:

# define training data

sentences = [['Subject: looking for medication ? we ` re the best source .', 'Subject:

# train model
model = Word2Vec(sentences, min_count=1)
# fit a 2d PCA model to the vectors
X = model.wv[model.wv.key_to_index]
pca = PCA(n_components=2)
result = pca.fit_transform(X)
# create a scatter plot of the projection
pyplot.scatter(result[:, 0], result[:, 1])
words = list(model.wv.key_to_index)
for i, word in enumerate(words):
  pyplot.annotate(word, xy=(result[i, 0], result[i, 1]))
pyplot.show()
```



```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (3.2.5)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from nltk)
```

```
# NORMALIZING:

X = spam5_df.spam_filter
X = X.values.reshape(1, -1)

col_names = ["Client", "text", "label_num", "text_length", "label", "spam_filter"]

# Break into a set of features and a variable for the known outcome.
y = ["text_length", "text"]
#y = spam_df["Client", "text", "label_num", "text_length"] #put in actual label/feature

scaler = StandardScaler()
X_std = StandardScaler().fit_transform(X)

# Data frame to store features and predicted cluster memberships.
ypred = pd.DataFrame()
```

```
spam5_df.head()
```

	Client	label	text	label_num	text_length	spam_filter	text_
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0	327	0	
			Subject: hpl nom for				

PADDING WITH ZERO(formatting the text_length column w/ zero-padding):

```
# importing pandas
import pandas as pd

# making data frame from csv at url
data = pd.read_csv("spam_ham_dataset.csv")

# converting to string dtype
data["text"] = spam5_df["text"].astype(str)

# width of output string
width = 55

# calling method and overwriting series
data["text"] = spam5_df["text"].str.zfill(width)

# display
data
```

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0
...	...	...	...	...
5166	1518	ham	Subject: put the 10 on the ft\r\nthe transport...	0
5167	404	ham	Subject: 3 / 4 / 2000 and following noms\r\nhp...	0

Word Vectorization

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(spam5_df['text'], spam5_df['spam_fi
# training the vectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()

X_train = vectorizer.fit_transform(X_train)
```

```
from sklearn import svm
svm = svm.SVC(C=1000)
svm.fit(X_train, y_train)
```

```
SVC(C=1000, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
from sklearn.metrics import confusion_matrix
X_test = vectorizer.transform(X_test)
y_pred = svm.predict(X_test)
print(confusion_matrix(y_test, y_pred))
```

```
[[371   6]
 [  0 141]]
```

- Actually/predicted to be Ham: 371
- Actually/predicted to be Spam: 141
- Predicted Spam/mistaken for Ham: 0
- Predicted Ham/mistaken for Spam: 6

```
#Testing it against a few new examples:

def pred(msg):
  msg = vectorizer.transform(['What is the Matter with you?'])
  prediction = svm.predict('What is the Matter with you?')
  return prediction[0]
```

```
def pred(msg):
  msg = vectorizer.transform(['WINNER$$$$$ SMS REPLY "WIN"'])
  prediction = svm.predict('WINNER$$$$$ SMS REPLY "WIN"')
  return prediction[0]
```