Capstone Project

Machine Learning Engineer Nanodegree

Bipin Sawant

July 13, 2020

Contents

I. Definition 2
Project Overview 2
Problem Statement 2
Metrics 3
II. Analysis 4
Data Exploration 4
Exploratory Visualization 7
Algorithms and Techniques 9
Benchmark 12
III. Methodology 13
Data Preprocessing 13
Implementation 14
Refinement 16
IV. Result 17
Model Evaluation and Validation 17
Justification 18
V. Conclusion 19
Free-From Visualization 19
Reflection 19
Improvement 20

I. Definition

Project Overview

Spam filtering is a binary classification task familiar to any user of email services. We will use machine learning classifiers to implement a similar spam filter.

The task is to distinguish between two types of emails, “spam” and “non-spam” often called “ham”. The machine learning classifier will detect that an email is spam if it is characterized by certain features. The textual content of the email – words like “Viagra” or “lottery” or phrases like “You've won 100,000,000 dollars! Click here!”, “Join now!” – is crucial in spam detection and offers some of the strongest clues.

To train the classifier, we need a representative dataset with both spam and ham emails. In this project, we will use Apache SpamAssasin public corpus, which contains about 6047 messages with approximate 30% spam ratio.

Problem Statement

Implement a Spam Filter with the help of a ML classifier which would classify a given mail as spam or ham.

The goal is to create a binary classifier which could mark an email as spam or ham with an acceptable accuracy. The tasks involved are as below:

- Download Spam and Ham mail data from Apache SpamAssasin public corpus.
- Pre-process email to normalize, lemmatize and tokenize the email worlds.
- Extract Feature from the spam and ham mail data.
- Train various classifiers and compare their performances.
- Fine tune the hyper parameters of best classifier model selected in above step.

The final classifier is expected to be useful in in classifying email as spam or ham and should perform better than sklearn dummy classifier.

Metrics

In real world scenario volume of ham males are generally higher that the spam mails. In the given dataset, also the spam ratio is 30% only. In this case Classification accuracy not enough since, it is the number of correct predictions made divided by the total number of predictions. So, a model that only predicted Ham mail in all case, would achieve an accuracy of approximately 70%. This is a high accuracy, but model is literally doing nothing and blindly classifying all mails as Ham. For such problem f1score is a better metric to analyze performance of a classifier. It considers both the precision and the recall of the test to compute the score. The solution will evaluate f1score of ML model against the benchmark f1 score

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

Where ***tp*** is true positive, ***fp*** is false positive and ***fn*** is false negative

II. Analysis

Data Exploration

Apache SpamAssasin public corpus has spam and ham mails data. Below is the overview of data
URL:

- http://spamassassin.apache.org/old/publiccorpus/

Content Folder:

- **spam:** 501 spam messages, all received from non-spam-trap sources.
- **easy_ham:** 2501 non-spam messages. These are typically quite easy to differentiate from spam, since they frequently do not contain any spammish signatures (like HTML etc).
- **hard_ham:** 251 non-spam messages which are closer in many respects to typical spam: use of HTML, unusual HTML markup, coloured text, "spammish-sounding" phrases etc.
- **easy_ham_2:** 1401 non-spam messages. A more recent addition to the set.
- **spam_2:** 1397 spam messages. Again, more recent.

Total count: **6051** messages, with about a **31.26%** spam ratio.

Spam Mail Example

```
From 12a1mailbot1@web.de  Thu Aug 22 13:17:22 2002
Return-Path: <12a1mailbot1@web.de>
Delivered-To: zzzz@localhost.spamassassin.taint.org
Received: from localhost (localhost [127.0.0.1])
     by phobos.labs.spamassassin.taint.org (Postfix) with ESMTP id
136B943C32
     for <zzzz@localhost>; Thu, 22 Aug 2002 08:17:21 -0400 (EDT)
Received: from mail.webnote.net [193.120.211.219]
     by localhost with POP3 (fetchmail-5.9.0)
     for zzzz@localhost (single-drop); Thu, 22 Aug 2002 13:17:21 +0100
(IST)
Received: from dd_it7 ([210.97.77.167])
     by webnote.net (8.9.3/8.9.3) with ESMTP id NAA04623
     for <zzzz@spamassassin.taint.org>; Thu, 22 Aug 2002 13:09:41
+0100
From: 12a1mailbot1@web.de
Received: from r-smtp.korea.com - 203.122.2.197 by dd_it7  with
Microsoft SMTPSVC(5.5.1775.675.6);
      Sat, 24 Aug 2002 09:42:10 +0900
To: <dcek1a1@netsgo.com>
Subject: Life Insurance - Why Pay More?
Date: Wed, 21 Aug 2002 20:31:57 -1600
MIME-Version: 1.0
Message-ID: <0103c1042001882DD_IT7@dd_it7>
Content-Type: text/html; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META content=3D"text/html; charset=3Dwindows-1252" http-
equiv=3DContent-T=
ype>
<META content=3D"MSHTML 5.00.2314.1000" name=3DGENERATOR></HEAD>
<BODY><!-- Inserted by Calypso -->
<TABLE border=3D0 cellPadding=3D0 cellSpacing=3D2
id=3D_CalyPrintHeader_ r=
ules=3Dnone
style=3D"COLOR: black; DISPLAY: none" width=3D"100%">
```

Ham Mail Example

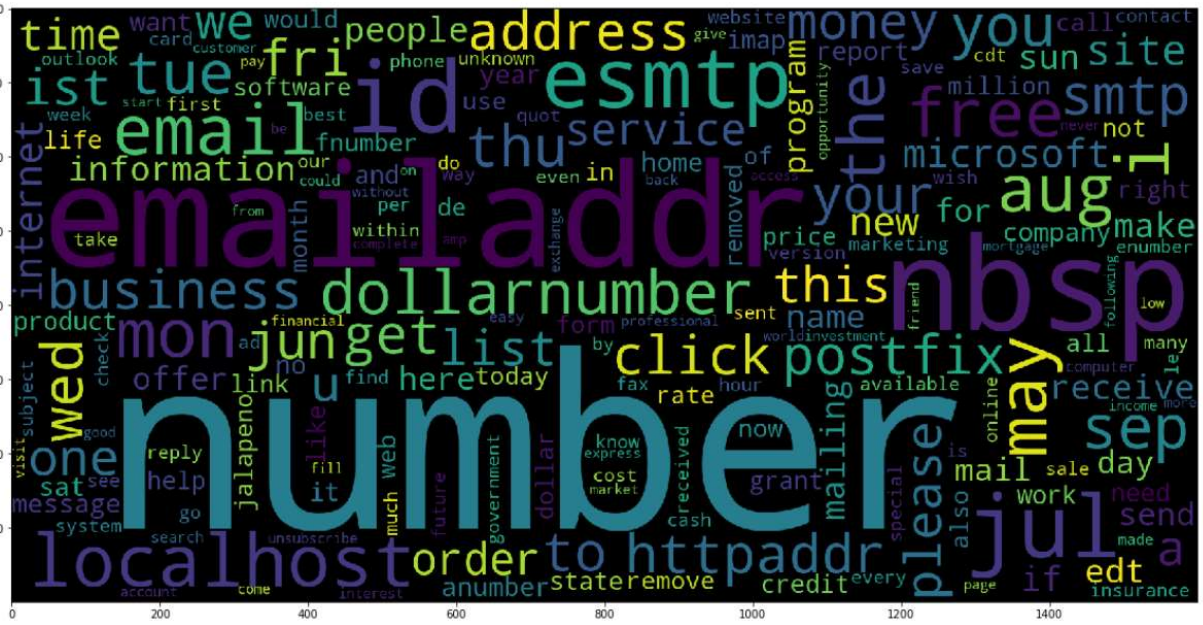
```
Return-Path: <malcolm-sweeps@mrichi.com>
Delivered-To: rod@arsecandle.org
Received: (qmail 16821 invoked by uid 505); 7 May 2002 14:37:01 -0000
Received: from malcolm-sweeps@mrichi.com by blazing.arsecandle.org
      by uid 500 with qmail-scanner-1.10 (F-PROT: 3.12. Clear:0.
Processed in 0.260914 secs); 07 May 2002 14:37:01 -0000
Delivered-To: rod-3ds@arsecandle.org
Received: (qmail 16811 invoked by uid 505); 7 May 2002 14:37:00 -0000
Received: from malcolm-sweeps@mrichi.com by blazing.arsecandle.org
      by uid 502 with qmail-scanner-1.10 (F-PROT: 3.12. Clear:0.
Processed in 0.250416 secs); 07 May 2002 14:37:00 -0000
Received: from bocelli.siteprotect.com (64.41.120.21)
  by h0090272a42db.ne.client2.attbi.com with SMTP; 7 May 2002 14:36:59
-0000
Received: from mail.mrichi.com ([208.33.95.187])
     by bocelli.siteprotect.com (8.9.3/8.9.3) with SMTP id JAA14328;
     Tue, 7 May 2002 09:37:01 -0500
From: malcolm-sweeps@mrichi.com
Message-Id: <200205071437.JAA14328@bocelli.siteprotect.com>
To: <rod-3ds@arsecandle.org>
Subject: Malcolm in the Middle Sweepstakes Prize Notification
Date: Tue, 7 May 2002 9:38:27 -0600
X-Mailer: sendEmail-v1.33

May 7, 2002
Dear rod-3ds@arsecandle.org:

Congratulations!  On behalf of Frito-Lay, Inc., we are pleased to
advise you
 that you've won Fourth Prize in the 3D's(R) Malcolm in the Middle(TM)
 Sweepstakes.   Fourth Prize consists of 1 manufacturer's coupon
redeemable at
 participating retailers for 1 free bag of 3D's(R) brand snacks (up to
7 oz.
 size), with an approximate retail value of $2.59 and an expiration
date of
 12/31/02.
```

Exploratory Visualization

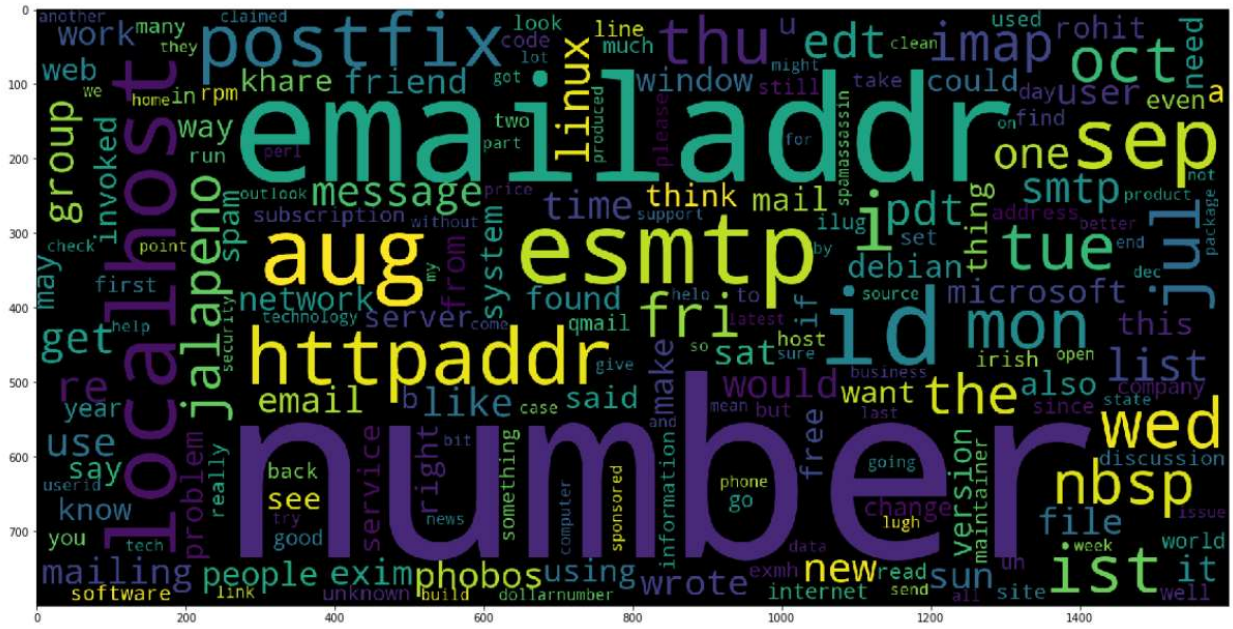
Emails are pre-processed to normalize, tokenize and lemmatize. After that, for each word, its frequency of appearance is calculated. This word and frequency data is used to create two type of plot for visualization.

1. Word cloud (tag cloud, or weighted list in visual design) is a visual representation of text data. Tags are usually single words, and the importance of each tag is shown with font size or color, which gives greater prominence to words that appear more frequently. Two separate word cloud map is created for spam and ham words. This visualization makes it easier to see relative frequency of words in spam and ham mails.

   Spam Mail Word Cloud

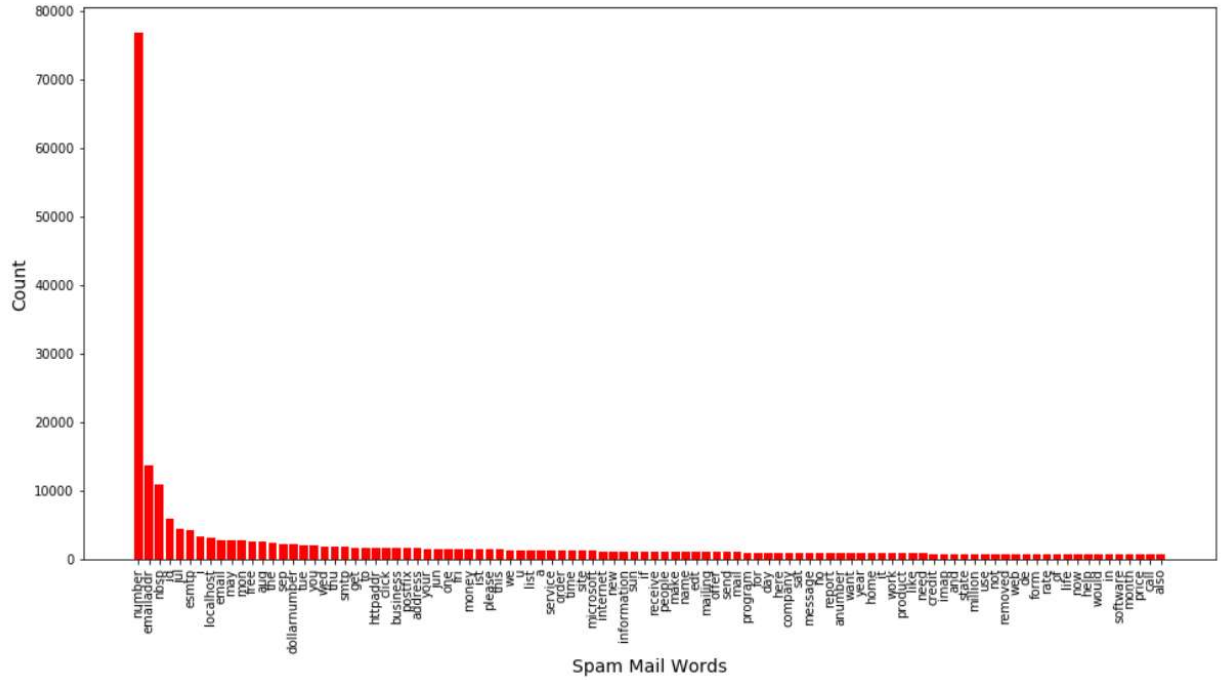


Ham Mail Word Cloud

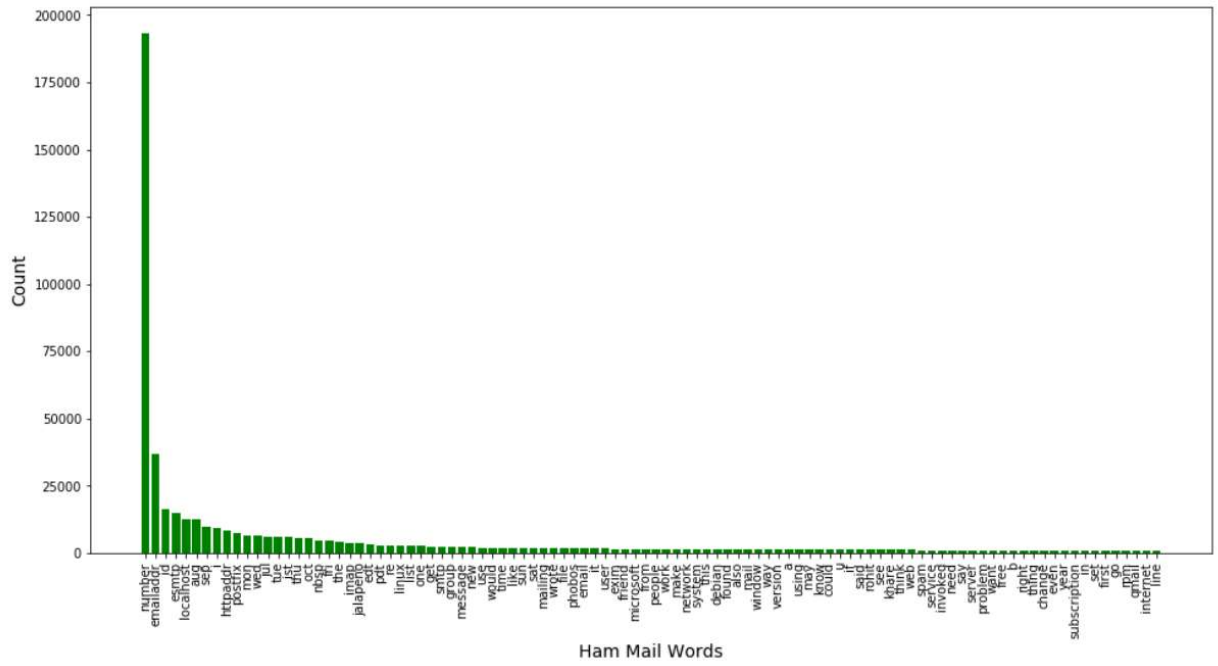


2. Frequency Histogram: 100 most frequent word and its frequency is plotted for spam email words and ham email words. Y axis has frequency count of top 100 high frequency words and X-axis has the corresponding word. This plot helps in visualizing the actual frequency of words in spam and ham mails.

Spam Mail Word Frequency Histogram (Top 100):



Ham Mail Word Frequency Histogram (Top 100):



Algorithms and Techniques

Sklearn Classifiers

Since the problem at hand is a typical binary classification problem. A bunch of classifiers is evaluated for the classification task. Following scikit-learn supervised learning models are selected

- Gaussian Naive Bayes (GaussianNB)
- Logistic Regression
- K-Nearest Neighbors (KNeighbors)
- Support Vector Machines (SVM)

Gaussian Naïve Bayes

Naive Bayes classifiers is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions between the features. It is a simple technique for constructing classifiers that assign class labels to problem instances, represented as vectors of feature values.

Why: Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. It only requires a small number of training data to estimate the parameters necessary

for classification. Also, the Naive Bayes has been used real Spam classification system.

Advantage: Naive Bayes classifiers are highly scalable, requiring parameters linear in the number of variables (features/predictors) in a learning problem. Maximumlikelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

Disadvantage: Due the feature independence assumption, it loses the ability to exploit interaction between features, however for classification task this is often is not a problem.

Source: https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Logistic Regression

Logistic Regression classification model is a probabilistic model which maximize the posterior class probability. It is used to model dichotomous (binary) outcome variables. In this model log odds of the outcome is modeled as a linear combination of the predictor variables

Why: Logistic Regression is also suitable for binomial classification problem like this. Also, as it is a high bias/low variance model, it Works well when we have large number of features in comparison to the number of data/sample. So, it is a good choice for the problem at hand where the number of data is low but the feature space is relatively large.

Advantage: If there's a lot of noise, logistic regression can handle it better. Logistic regression is intrinsically simple, it has low variance and so is less prone to overfitting.

Disadvantage: Unlike SVM, it considers all the points in the data set. This may be not being a preferred approach for certain problems

Source: https://en.wikipedia.org/wiki/Logistic_regression

K-Nearest Neighbours (KNeighbors)

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors

Why: Because the training set size is small so implementing instance based learning such as KNN classification can be a good candidate for such scenario. Also, we do not have to worry about the linear seaparability of the data and implanting the model is very easy as well.

Advantage: Makes no assumption about the data distribution. Learning time is very low. Immediately adapts to the new data not affected by linear separability of the data. An easy algorithm to explain and implement.

Disadvantage: Slow during prediction. So difficult to use this for prediction in real time. Storage space can be a challenge if working on more data. Performance impact is high if more features are added (Curse of Dimensionality)

Source:
http://www.dummies.com/programming/big-data/data-science/solving-realworld-problems-with-nearest-neighbor-algorithms/
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

SVM

A Support Vector Machine (SVM) is a classifier formally defined by a separating hyperplane. It maps the points in space, so that the separate categories are divided by a clear gap that is as wide as possible (maximizing the margin)

By implementing Kernel trick, we can perform nonlinear transformation in higher dimensional space. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space.

Why: SVM performs well for most of the cases. It can be used with various Kernel to fit the nonlinear decision boundary as well. Various parameters can be tuned to get a high level of accuracy.

Advantage High accuracy. By implementing Kernel Trick, we can inject the domain knowledge in the classifier. With appropriate kernel, it can work well even if the data is not linearly separable.

Disadvantage. Tuning the model for optimal parameters can be difficult. Memory intensive. Hard to interpret

Source: https://en.wikipedia.org/wiki/Support_vector_machine

Benchmark

In this project, we will use the sklearn dummy classifier as a simple baseline to compare with. DummyClassifier is a classifier that makes predictions using simple rules, so the goal is to perform better that this.

III. Methodology

Data Preprocessing

Reading Data

While reading the email from corpus file, opening the file with utf8 encoding. If UnicodeDecodeError occurs due to not supported character, we catch the exception and continue reading next file.

Pre-processing

In data pre-processing step we normalize, tokenize, lemmatize and filter out stop words and non-alphanumeric words. Below are the details of pre-processing. Lemmatize and change case to lower to reduce the words to their stemmed form.

- Clean the mail text by removing html tags
- Normalize numbers, Urls, email address and Dollar sign (Replace numeric values with ‘number’ , hyperlinks with 'httpaddr', any email address with 'emailaddr' and $ with 'dollar')
- Tokenize the words and take only alphanumeric words
- Filter stop words (words which do not have significance like : to, the, a etc)

Lemmatize and change case to lower to reduce the words to their stemmed form.

Processed Spam Mail

```
['emailaddr', 'tue', 'aug', 'number', 'number', 'number', 'number', 'email
addr', 'localhost', 'localhost', 'postfix', 'esmtp', 'id', 'tue', 'number'
, 'aug', 'number', 'number', 'number', 'number', 'edt', 'phobos', 'localho
st', 'imap', 'emailaddr', 'tue', 'number', 'aug', 'number', 'number', 'num
ber', 'number', 'ist', 'may', 'forged', 'esmtp', 'id', 'gnumberrdrwznumber
', 'tue', 'number', 'aug', 'number', 'number', 'number', 'emailaddr', 'waa
number', 'tue', 'number', 'aug', 'number', 'number', 'number', 'number', '
tue', 'number', 'aug', 'number', 'number', 'number', 'number', 'emailaddr'
, 'emailaddr', 'emailaddr', 'emailaddr', 'emailaddr', 'emailaddr', 'emaila
ddr', 'emailaddr', 'emailaddr', 'www', 'form', 'result', 'feedback', 'form
', 'it', 'submitted', 'emailaddr', 'tuesday', 'august', 'number', 'number'
, 'number', 'number', 'click', 'want', 'pay', 'porn', 'would', 'like', 'ge
t', 'free', 'the', 'honest', 'no', 'risk', 'number', 'free', 'way', 'if',
'take', 'couple', 'minute', 'read', 'simple', 'guide', 'able', 'get', 'fre
e', 'pass', 'top', 'paysites', 'online', 'click']
```

Processed Ham Mail

```
['emailaddr', 'thu', 'aug', 'number', 'number', 'number', 'number', 'email
addr', 'localhost', 'localhost', 'postfix', 'esmtp', 'id', 'thu', 'number'
, 'aug', 'number', 'number', 'number', 'number', 'edt', 'phobos', 'localho
st', 'imap', 'emailaddr', 'thu', 'number', 'aug', 'number', 'number', 'num
ber', 'number', 'ist', 'esmtp', 'gnumbermenlznumber', 'thu', 'number', 'au
g', 'number', 'number', 'number', 'number', 'esmtp', 'exim', 'number', 'de
bian', 'id', 'thu', 'number', 'aug', 'number', 'number', 'number', 'number
', 'esmtp', 'exim', 'number', 'debian', 'thu', 'number', 'aug', 'number',
'number', 'number', 'number', 'authenticated', 'esmtp', 'id', 'gnumbermein
umbervpnumber', 'thu', 'number', 'aug', 'number', 'number', 'number', 'dav
id', 'b', 'nothing', 'found', 'sadev', 'interesting', 'approach', 'spam',
'emailaddr', 'emailaddr', 'emailaddr', 'spamassassin', 'developer', 'thu',
'number', 'aug', 'number', 'number', 'number', 'number', 'thu', 'number',
'aug', 'number', 'number', 'number', 'number', 'seen', 'discussed', 'artic
le', 'approach', 'hell', 'rule', 'trying', 'accomplish', 'something', 'tho
mas', 'alva', 'email', 'sponsored', 'osdn', 'tired', 'phone', 'get', 'new'
, 'free', 'mailing', 'emailaddr']
```

Implementation

To create solution of the above problem we would be performing below steps:

- Pre-processing the mail - In this step we will process the mail content with following changes:
    - Remove html tags
    - Normalize numbers, Urls, email address and Dollar sign (Replace 0-9 with ‘number, hyperlinks with 'httpaddr', any email address with 'emailaddr' and $ with 'dollar')
    - Tokenize the words
    - Take only alphanumeric words
    - Lemmatize and change case to lower to reduce the words to their stemmed form.
    - Filter stop words (words which do not have significance like: to, the, a etc)
- Train test Split: - Create separate set for training and testing purpose.
- Extracting the features: Create a training set from above processed mail. From this set calculate frequency of each word and select high frequency word as feature for be used for learning.
- Training different classifier: Train the classifiers like Logistic Regression, K-Neighbour Classifier, SVM, Naïve Bayes etc

- Evaluating the classifiers: The above classifiers will be evaluated against the dummy classifier, by comparing their prediction time and f1 score.

Pre-processing helper methods:

- **preprocess_normalize:**
  This method is implemented to normalize the email data using regex rules, which removes html tags and replace numeric values with ‘number’ , hyperlinks with 'httpaddr', any email address with 'emailaddr' and $ with 'dollar'

- **preprocess_lemmatize_tokenize:**
  This method is using nltk word_tokenize to tokenize and nltk WordNetLemmatizer to lemmatize the email words. It also filters stopwords using nltk.corpus stopwords. This method also converts email words to lower case and takes only alpha-numeric words to return at the end.

- **LemmatizerTokenizer:**
  A custom tokenizer class is implemented which Lemmetize, Tokenize and perform other optimizations using the preprocess_lemmatize_tokenize method mentioned above

Sklearn Classifiers

Following scikit-learn supervised learning models are used to train the classifier.

- Gaussian Naive Bayes (GaussianNB)
- Logistic Regression
- K-Nearest Neighbors (KNeighbors)
- Support Vector Machines (SVM)

Training

A helper function **train_predict** is implemented for training and testing the four supervised learning models we have chosen above.

The function takes as input a classifier, and the training and testing data. It creates a sklearn pipeline using the vectorizer created in data processing step. Using this pipeline, the model is trained.

This function will report the training time, prediction time and f1score for both the training and testing data separately.

We import the dummy classifier and four other supervised learning models mentioned earlier, and run the **train_predict** function for each one. We will perform following action:

From the X_ train data set create four different training set of sizes: 25%, 50%, 75% and 100% of data.
Import the supervised learning models discussed in the previous section.
Initialize the three models and store them in clf_A, clf_B, clf_C and clf_D
Use a random_state for each model.
Use the default settings for each model — we will tune one specific model in a later section.
Fit each model with each training set size and make predictions on the test set.

Challenges and Observations:

I was facing memory exception on my laptop i5 processor and 8 GB RAM. Taking a lot of time for training the data.

In the input data, there are mails with nonstandard keyboard characters. These nonUnicode characters were causing issues in nltk tokenizers These data are filtered while reading by encoding as utf8 character.

Refinement

Initially I trained the classifier on sklearn.svm SVC which is not giving a meaningful result. F1_score is quite low in comparison to other classifiers. A regular SVC with default values uses a radial basis function as the kernel. This nonlinear basis function has higher variance. We can add regularization to the nonlinear model and we will probably see much better results. (This is the C parameter in scikit learn SVMs), however I decided to use a LinearSVC instead since we are training on default parameters and linear kernel has lower variance by default. LinearSVC seems a better choice because we have large feature space in comparison to the training dataset.

IV. Result

Model Evaluation and Validation

Below are the 16 different outputs 4 for each model using the varying training set sizes.

Classifier 1 - GaussianNB

Training Set Size	*Training Time*	*Prediction Time (test)*	*F1 Score (train)*	*F1 Score (test)*
1208	563.625	101.282	0.9973	0.8195
2216	745.116	101.775	0.9987	0.8432
3624	2129.144	101.640	0.9987	0.8796
4833	3317.674	102.033	0.9987	0.8926

Classifier 2 - LogisticRegression

Training Set Size	*Training Time*	*Prediction Time (test)*	*F1 Score (train)*	*F1 Score (test)*
1208	562.869	99.921	0.8915	0.8717
2216	730.136	100.318	0.9241	0.9220
3624	2121.438	100.204	0.9398	0.9276
4833	3235.752	101.185	0.9481	0.9348

Classifier 3 - KNeighborsClassifier

Training Set Size	Training Time	Prediction Time (test)	F1 Score (train)	F1 Score (test)
1208	562.466	145.419	0.8384	0.7569
2216	737.632	234.933	0.8823	0.8037
3624	2130.281	335.584	0.8828	0.8160
4833	3302.774	479.244	0.8117	0.7534

Classifier 4 - LinearSVC

Training Set Size	Training Time	Prediction Time (test)	F1 Score (train)	F1 Score (test)
1208	562.789	100.029	0.9946	0.9530
2216	728.364	100.347	0.9959	0.9730
3624	2119.104	101.484	0.9973	0.9795
4833	3244.224	101.099	0.9970	0.9808

In the above table gives we can see that KNeighborsClassifier is very slow. Gaussian Naïve Bayes and Logistic Regression Classifier are having similar training time and prediction time, but Logistic Regression Classifiers has higher score.

Justification

Overall Linear SVC model seems to be the best classifier, because the model is giving the highest testing f1 score among the other models. Training and prediction time is also comparable to Logistic Regression and Gaussian NB models in the experiment.

V. Conclusion

Free-From Visualization

From my Test data set
Content Folder:

- **spam:** 501 spam messages, all received from non-spam-trap sources.
- **easy_ham:** 2501 non-spam messages. These are typically quite easy to differentiate from spam, since they frequently do not contain any spammish signatures (like HTML etc).
- **hard_ham:** 251 non-spam messages which are closer in many respects to typical spam: use of HTML, unusual HTML markup, coloured text, "spammish-sounding" phrases etc.
- **easy_ham_2:** 1401 non-spam messages. A more recent addition to the set.
- **spam_2:** 1397 spam messages. Again, more recent.

Total count: **6051** messages, with about a **31.26%** spam ratio.

I would have loved to provide visualization such TF-IDF score for Spam and Ham mails. Currently my machine is running forever to fit the model I want. It has been a resource challenge but I will try to get this visualization eventually.

Source:
https://buhrmann.github.io/tfidf-analysis.htm

Reflection

Cleaning the data and preprocessing step to normalize, tokenize and lemmatize the input data is quite interesting. There is lot of preprocessing needed to clean and transform the data, to make it usable to work with feature extraction and model training. earlier I was relying on word frequency count for feature selection and relying on an empirical value to decide the selection cut-off.

Improvement

While researching about Email Spam classifiers, I tried to find out how this is implemented in the email providers like Gmail.
Gmail's engineering team added Deep Learning capabilities to the Gmail spam filtering system.

Source:
https://www.quora.com/How-does-the-Gmail-spam-filter-work
https://pepipost.com/blog/gmail-spam-filters-evolution/
https://www.sciencedirect.com/science/article/pii/S2405844018353404