

**Contents Page**

**Contents**

[Introduction 2](#_30j0zll)

[Specification 3](#_1fob9te)

[Preparing the text data 3](#_3znysh7)

[Stop words 3](#_2et92p0)

[The: 3 Of: 1 For: 1 I: 1 3](#_tyjcwt)

[The: 5 Of: 3 To: 2 Or: 1 I: 2 A: 4 4](#_3dy6vkm)

[Lemmatization 4](#_1t3h5sf)

[Creating word dictionary 5](#_4d34og8)

[Feature extraction process 5](#_2s8eyo1)

[Training the classifier 6](#_17dp8vu)

[Criteria for Categories 6](#_3rdcrjn)

[Keyword sets 6](#_26in1rg)

[Flowcharts 7](#_lnxbz9)

[Pseudo Code 8](#_35nkun2)

[Milestone 2 8](#_1ksv4uv)

[SVM 8](#_44sinio)

[Dataset 9](#_2jxsxqh)

[Description of Fields 10](#_z337ya)

[Measurement metric (F1 Score) 10](#_3j2qqm3)

[Creating the machine 12](#_1y810tw)

[Results 13](#_4i7ojhp)

[Evaluation 13](#_2xcytpi)

[Conclusion 14](#_1ci93xb)

[Future works 15](#_3whwml4)

# Introduction

Data Mining is an incredibly valuable asset within investigative forensics; effectively it is the process of analysing massive amounts of already existing information and then using it to predict certain occurrences or situations that could arise in the future. One of the best examples of this is email filtering, there is a multitude of different types of emails that come through every day, some are just normal everyday emails known as ham whereas others could be spam, phishing emails or even contain malicious items such as malware. Email filtering can be used to classify these types of mail and only give you the types of emails the user would want to see; this is just a basic example of data mining but it is the foundation for it as a forensic technique.

The idea of this project was to create a programme powered by a support vector machine that analyses a set of emails and further classifies them into their different variations such as spam or ham. An ideal result will be to ensure all the emails are appropriately split up and put into their different sections using a python script.

This investigative report will visit various different areas, such as how to properly prepare the text for analysis using techniques such as lemmatization and the use of stop words; this will be performed by utilising feature extraction and training the classifier and demonstrating how integral they are to the overall process. This will be followed by talking about different criteria for categories of email, how a support Vector machine works (SVM) before moving on to actual measurement and analysis methods such as the F1 score before displaying our finished script. Effectively walking you through all the basic processes needed in order to create a complex machine that can learn by itself.

This is something that is unknowingly used in everyday life and understanding the basics is essential whether you’re a forensics analyst or an average user viewing their emails.

# Specification

To start we will split up all the different types of possible emails into more general titles; from here we will further define what makes up these different categories. After which we will take these qualities and feed them throughout machine in order to split incoming mail into the relevant areas.

## Preparing the text data

Before any data can be inserted into the model it needs to be properly prepared to give the model the best chance of differentiating different types of email; there are a few different methods that can be used in preparation.

## Stop words

To start, within an email there are bound to be a lot of common words that don’t help the classification of the email but rather hinder it due to their constant use in the English language. These words are called “stop words” and include things such as “and”, “of” and “the”. These words clutter the sentences and don’t offer much help; so the first step in preparation would be to remove these types of words. For Example in the Ling-spam corpus one of the emails read as follows:

*”*

*Subject: re : 3 . 403 human subjects*

***i*** *am not clear about* ***the*** *laws outside* ***the*** *uk but surely* ***the*** *problem* ***for*** *using human subjects speaking is copyright ? unless you have clearance fronm them use* ***of*** *theri words in an y ' published ' form might contravene their rights . certaibly authors of ' authetntic ' , aterr / materials have faced this issue for soime time .”*

**The: 3 Of: 1 For: 1 I: 1**

A second email reads as follows:

*“****a*** *bibliography* ***of*** *work on language and power has been placed on* ***the*** *linguist server . this bibliography combines contributions from* ***a*** *number* ***of*** *respondents to* ***a*** *query* ***i*** *placed in linguist* ***a*** *while back . most* ***of*** *those respondents are listed in my summary ( linguist 3-366 ) . some items may be inappropriate - -* ***i*** *was unable to verify all* ***of******the*** *suggestions . this bibliography includes* ***a*** *wide range* ***of*** *items which relate in one degree* ***or*** *another to* ***the*** *topic - - from* ***the*** *highly theoretical to* ***the*** *highly particularistic to* ***the*** *applied . apologies for any incorrect or incomplete references”*

**The: 5 Of: 3 To: 2 Or: 1 I: 2 A: 4**

Within these 2 emails we can see there are multiple examples of stop words such as “the” “of” “am” “at” “on” and “but” , and the word “the” is used 3 times in one email and 5 times in the next, this shows how common place they are as words as these 2 emails are both about completely different subjects yet they still contain a lot of the same words; these words don’t help to indicate the intention of the email whatsoever and can therefore be overlooked when inserted into the model. This saves large amounts of time as it cuts out a good proportion of the words within an email that aren’t good indicators and thus only leaves the words that have some potential value.

## Lemmatization

The process of stemming is a basic version of lemmatization that just chops off the ends of words with the hope of achieving the goal correctly most of the time; whereas real lemmatization does a more thorough job that uses a knowledge of vocabulary and morphological analysis of words, this aims to remove only inflectional endings and returns the word to its base dictionary form of a word which is known as the lemma. For example if we’re doing regular stemming a word such as “saw” could be cut to just “s” however with lemmatization would try and return a variation e.g. “see” or “saw” depending on whether its context is a vowel or a noun.

This basically puts together different variations of the same word e.g. “read” “reading” “reader” these would all be represented as “read” this process saves a massive amount of time as it allows these words to be analysed as just one word rather than constantly having to pick out each individual variation of it. Within the Enron data set there are a lot of spam related emails that we are trying to differentiate from normal emails, the text in these emails is often very similar from email to email, this is where lemmatization steps in, for example in one spam email words such as “Loan” are used; rather than search for every variation of “Loan” we can group “Loans” and “Loan” together in order to cut down the time used in the search for them.

One email within the ling-spam corpus starts with:

*“ Subject: 6th manchester phonology meet -* ***programme***

*Provisional* ***programmes*** *the north”*

Within this small introduction we can see that there are 2 variations of the stem word “programme” on top of this outside of this email we have other variations such as “programming” “programmer” and “programmed”; all of these words when lemmatized will be reduced to “programme” and grouped as so to improve on the efficiency of the analysis and keep the amount of content having to be analysed to a minimum.

## Creating word dictionary

Using a word dictionary helps cut out the clutter as well, it firstly is used to add a load of particular words and their frequency, this helps better define the type of email as for instance phishing emails more often than not ask for things such as credentials, so if we add words such as “password” “username” and “Login” into our dictionary with an appropriate frequency it can then be used to highlight phishing emails.

A dictionary can be created using python and you can check what words are included via the command “print dictionary”; as the script analyses the dataset (set of emails) it will take count of the frequency of words and add them to the dictionary, as more and more emails are read and more words are recorded the script can then start to differentiate between the types of email due to certain words as explained above.

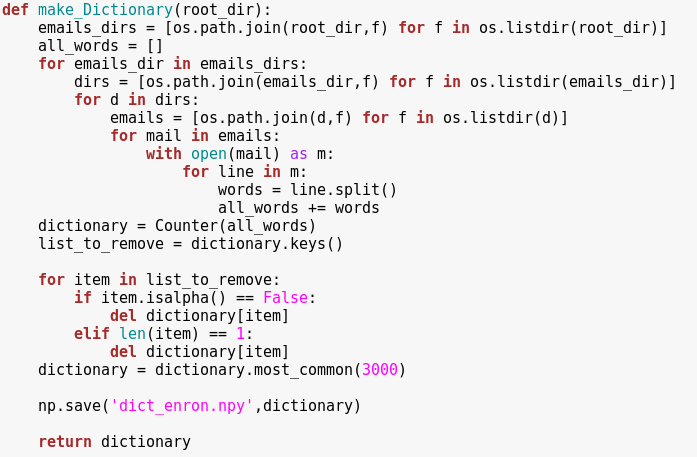
Another great use for word dictionary’s is to get rid of special characters, through the use of Python a short script can be written that makes it so special characters such as “!” or “$” are ignored and not analysed as again they don’t aid the classification of emails and can often confuse things as they cannot be shortened and are used in multiple lines of contexts with a lot of different ramifications. For instance in the below email taken from the ling-spam corpus data set:

*Subject: language teacher list*

*hi* ***,*** *an acquaintance of mine who be on the net* ***,*** *but not on the list would like to know whether there be a list that discuss foreign language teach* ***(*** *like didactic hint* ***,*** *textbook* ***,*** *classroom material* ***,*** *job* ***. . -*** *much like linguist*  ***).*** *if there be* ***,*** *send the relevant info to* ***:*** *birgit roller bowl green state university e-mail* ***:*** *birgitr* ***@*** *bgnet* ***.*** *bgsu* ***.*** *edu thank achim stenzel*

Within this email we can see multiple examples of special characters from **@** to **(** all of these characters can be dismissed by the algorithm leading to it only reading through credible and useful characters.

The python code below is used to create the dictionary of words which the program will use during the email classifying process:

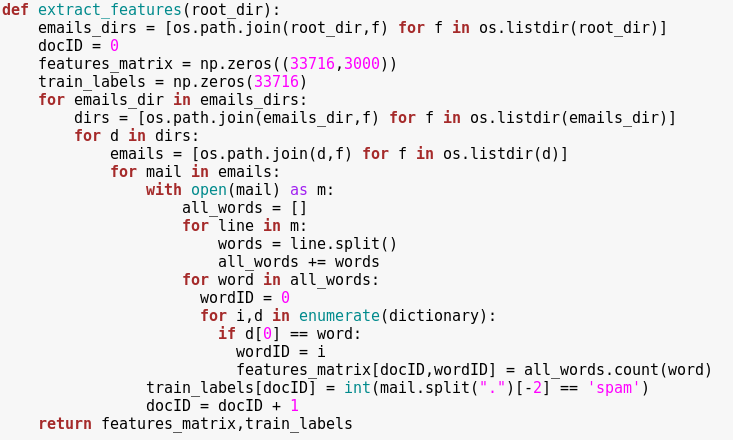


## Feature extraction process

When working with machine learning, image processing, and pattern recognition the feature extraction process can be utilised in order to reduce how many resources are required in order to present a large set of data. This is done by including and excluding attributes that are present in the data provided without changing any aspects of the data. This enables the user to identify or remove any data that is irrelevant or does not contribute to the reliability of the data.

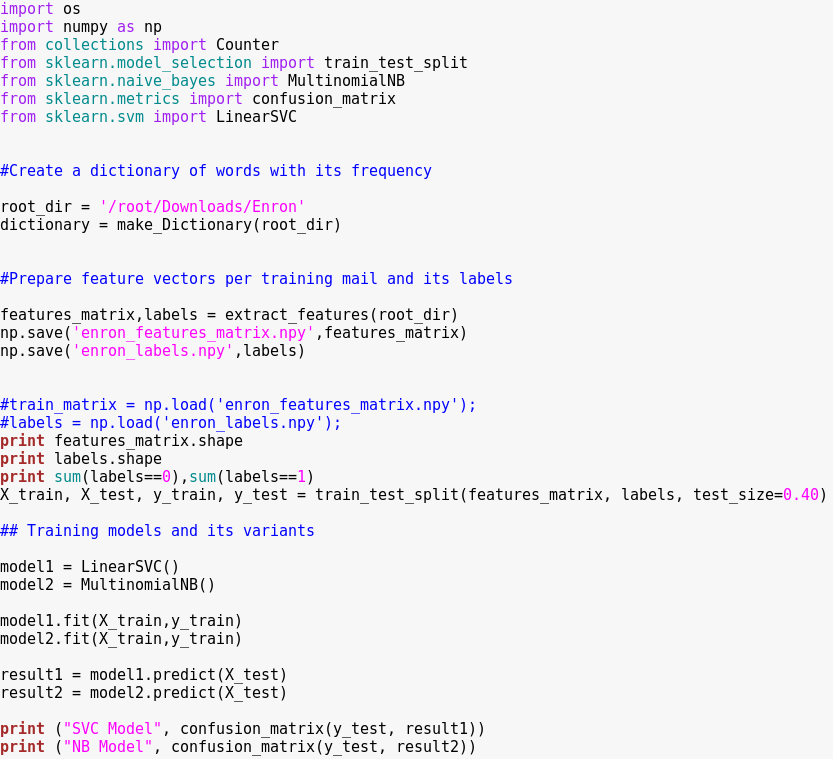
When attempting to perform analysis of large collections of data having many different variables can cause issues. This in turn leads to the requirement for large amounts of computing power. In general feature extraction is a strong methodology for being able to observe and organise the data sets variables whilst being able to describe the data with a certain level of accuracy.

An example of using the feature extraction process lies within the email classification script. Emails may have many unique words which represent having a large amount of variables in a data set. By using the feature extraction process it will be possible to identify how many unique words have been used and then following this the sort of email that has been sent whether it is spam, malware or genuine.



## Training the classifier

SVM’s are supervised binary classifiers meaning they’re very effective when there are a high number of features in use. In terms of training its classifiers the SVM, when given a set of training data (large quantities of data are used for best results), will form the word dictionary as mentioned previously and from this gather a subset of data which will become the support vectors (boundaries for separating the hyperplane). From these support vectors the SVM is able to begin assigning classification to each instance of data and continually learn which instance belongs to which classification the more training data it receives. Once the classifiers have been trained it will then be time to move onto using test data to see how well the classifiers work.



## Criteria for Categories

Spam - unrecognised sender address, shortened URL link, Misspelling, credentials, big blocks of text

Virus- abnormal file size in header, unrecognised email, misspelled names

Malware- .exe files, compressed files

Bots- a huge amount of recipients, shortened text

Normal- recognised email, past responses

Adverts- HTML, cookie history, keywords (deals etc.), Large or Bold text

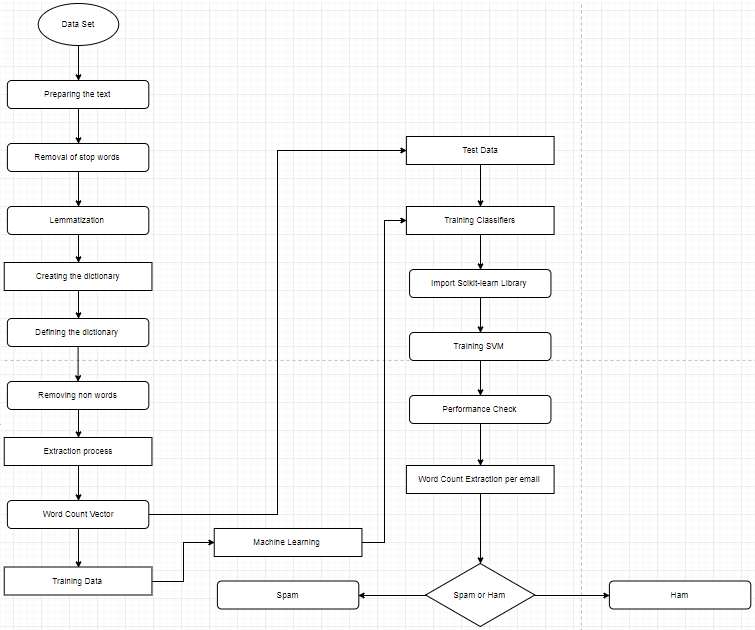
Phishing - links, misspell, login page, images

## Keyword sets

Phishing - Login, Credentials, Verify Account, Password, Client, Member, Secure, chosen, money, cash, click here, link

Spam - Deals, Bargains, off, %, Price, pay, unsubscribe, free, save, sale, meet singles, Earn, extra cash, discount, join, hot, gambling, cheap, win, credit

## Flowchart



## Pseudo Code

FOR EVERY EMAIL RECEIVED

IF KEYWORD IN KEYWORD DICTIONARY

THEN IF SPAM WORD COUNT > 200

AND IF TEXT WORD COUNT < 200

THEN EMAIL IS SPAM

ELSE IF TEXT WORD COUNT > 200

THEN EMAIL IS NOT SPAM

ELSE IF SPAM WORD COUNT < 200

AND IF TEXT WORD COUNT < 200

THEN EMAIL IS SPAM

ELSE IF TEXT WORD COUNT > 200

THEN EMAIL IS NOT SPAM

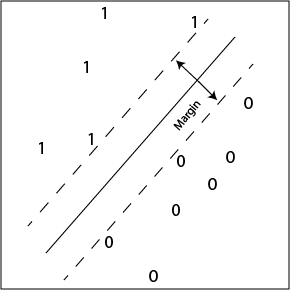
# 

# Milestone 2

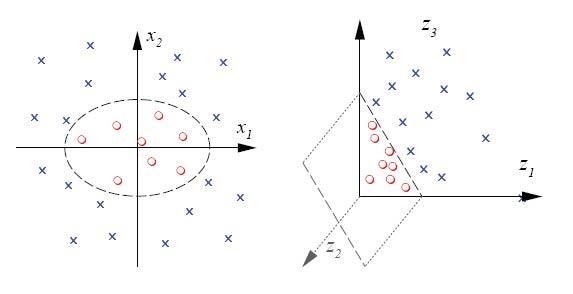
## SVM

An SVM also known as a support vector machine is an integral part of machine learning; they are supervised learning models which means they take in inputs from a source and produce an output based on already existing examples of input and output pairs. Their main purpose is to help in the classification and analysis of data. For instance in the demonstrated email classification there are 7 potential areas that an email could be placed into, the SVM algorithm will construct a model that will assign every email to one of those classifications based on aspects in the email such as senders, wording and other areas, however each attribute will be individually assessed through different graphs and hyperplanes.

This can be achieved via the use of hyperplanes, hyperplanes are a line on a graph that accurately splits up data into certain sections ready for classification; for instance if the emails were on a graph showing the amount of senders, the hyperplane could intersect the graph splitting it into under 100 senders and over 100 senders and defining the email as either being part of a bot or not being part of a bot. Just using a simple line hyperplane to split up the data is known as a linear hyperplane, however in a lot of areas a line won’t do and doesn’t appropriately split up the data, this is where the Kernel trick comes into play, this is a non-linear method of splitting up and defining data. It does this through some very complex data transformations, and then finds out the process in which to better separate the data based on certain attributes or labels that you have defined.

****

**Fig 1 Linear hyperplane: Chris Thornton Sussex University**



**Fig 2 Hyperplane Kernel trick: Chris Thornton Sussex University**

## Dataset

A dataset, is a massive collection of data such as emails, images, text and more, and is a critical part of machine learning as it provides the user with a lot of information designed to help train a computer program to recognise key information such as spam and malicious programs. By using a dataset within machine learning it allows for a more accurate discovery of files. In the case relating to the application currently being developed by the team, we are using the Lingspam dataset, which contains several thousand emails, allowing for the classification of them into either spam or not spam.

However, there is also the Enron dataset, which is a collection of approximately 600000 real emails between employees from a now bankrupt corporation and is used to help within data analysis to help train machine learning applications. There is also the sentence sentiments dataset, although smaller than the Enron dataset, it contains positive and negative text which can allow for the training of a classification application. When training a machine learning program, it requires two types of data, one to train the system and one to test it, however when training the MLP (machine learning program), the dataset used will be larger, so to test how accurate the program is at identifying different types of data in a larger body, whereas the testing data will be smaller. With regards to training the email classification program we are developing, we will be using approximately 300000 emails from the Enron dataset as the training data and 200000 for the testing, this is to ensure that our application can handle large amounts of data effectively whilst also being able to categorise smaller sets within a given time.

## Description of Fields

The different fields used in our algorithm and flowcharts are what we’re using to help discover which of our classifications an email in the Enron dataset falls into. Each field requires a yes or no answer which will in turn follow on to the next part of the flow chart ultimately coming to a final classification decision as to what the email is.

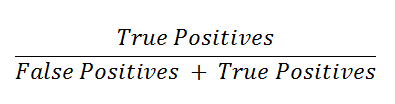
For example, the first question asked by the algorithm is whether the address the email was sent from is known, if the outcome is yes the email is checked to see how many people it is sent to, if this number is above 200 the email reaches the end of the flow chart and is categorized as a bot (refer to flow chart diagrams for visual representation).

The only time this example isn’t used is when the email is passed through our key word set function whereby instead of a yes or no outcome, a predetermined set of words are searched for to help determine whether an email is a phishing email or an advert.

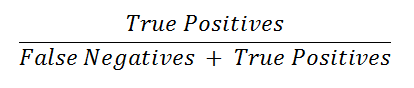
## Measurement metric (F1 Score)

When performing a statistical analysis with binary classifications users can utilise the F1 score, also commonly known as a F-Score or F-Measure. The F1 Score is used to take measurement of a tests accuracy to provide the investigator with useful information such as the reliability of their data.

In order to do this the F1 score uses both the precision (p) and recall (r) of a test and then calculates the mean from these two values. Precision is the defined as the number of correct positive results divided by the number of all positive results leading to the equation:

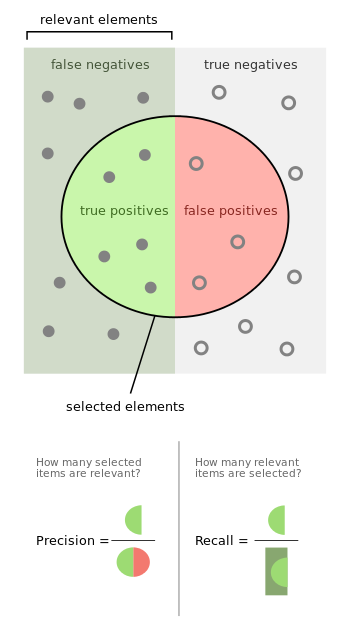


Whereas the recall can be calculated by taking the number of current positive results which is then divided by the total number of relevant samples:



Refer to Figure 3 which demonstrates precision and recall.

**Figure 3 - Precision and Recall visualised**

****

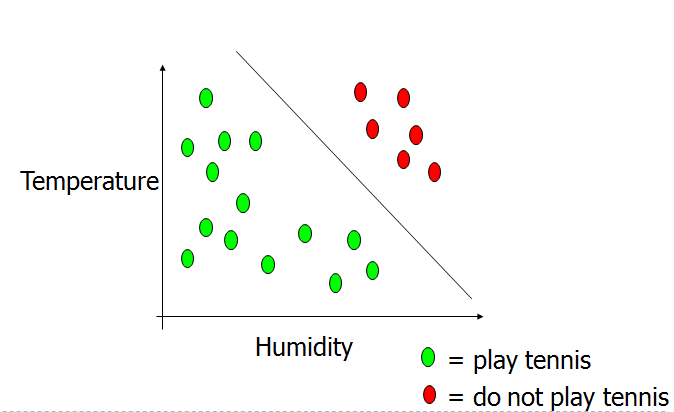
**Figure 3 Image Source:** [**https://en.wikipedia.org/wiki/F1\_score#/media/File:Precisionrecall.svg**](https://en.wikipedia.org/wiki/F1_score#/media/File:Precisionrecall.svg)

This then leads on to being able to perform the calculation to retrieve the F1 score which is:

****

As an example by using the F1 score an investigator could calculate from a large set of patients how many patients are infected with a virus compared to the patients who are healthy, Likewise it could also be used to calculate the likelihood of a group of people who will still play tennis depending on the temperature and humidity. Consult Figure 4 which demonstrates the tennis players being categorised.

**Figure 4 - Tennis players who will or won't play due to weather conditions**

****

**Figure 4 Image Source: Dr Deepayan Bhowmik, Digital Forensics and Machine Learning, Page 12**

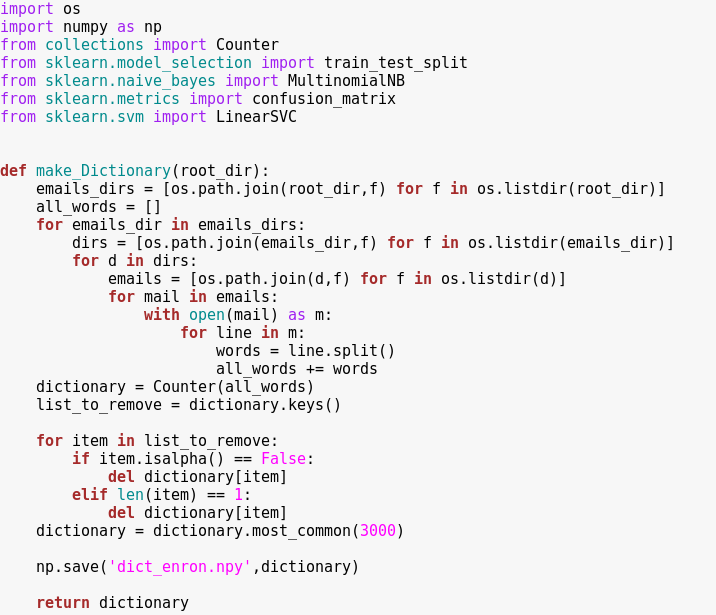
# Creating the machine

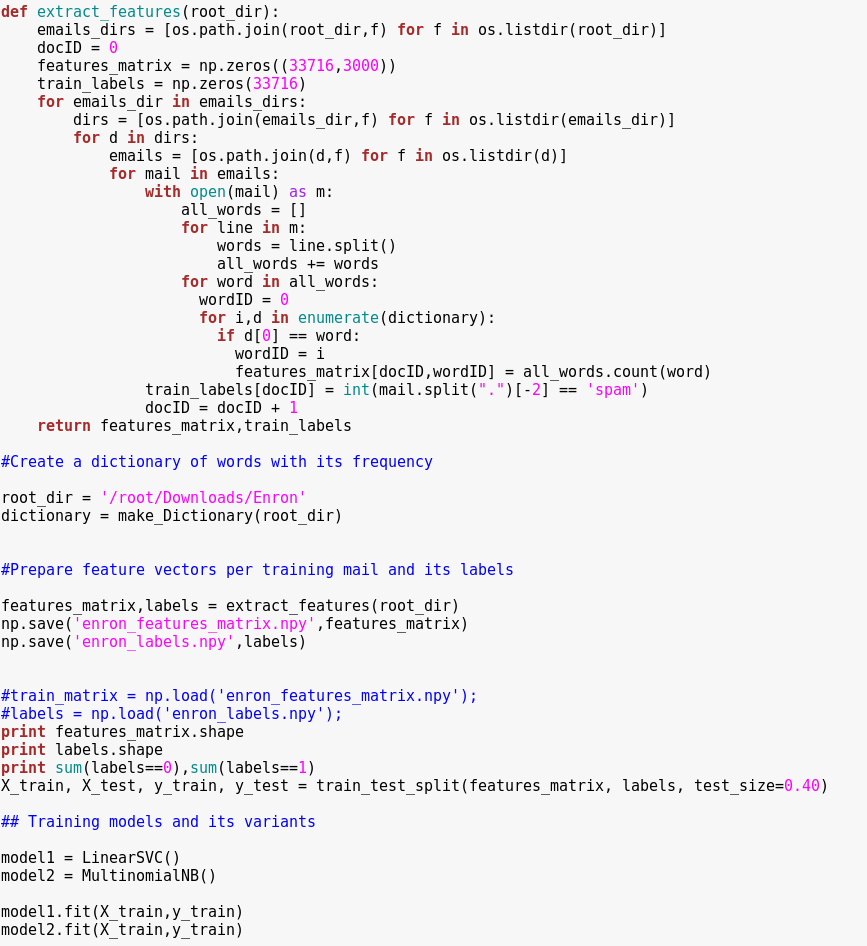
As the project required a machine be built to classify emails into either spam or not spam, we needed to acquire some training data on which the machine can use. The training and test data both came from the ling-spam corpus, as it is a recognised email classification training set within machine learning. After we had acquired this data we began to build the machine in the python programming language including downloading several libraries from the scikit learn library as well as numpy library. This is due to the scikit learn library allowing for machine learning and the numpy library allowing for measurement and metrics. Once the libraries had been successfully imported into python, the development of the machine could commence.

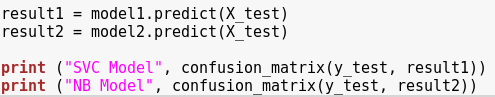
As the machine needed to be trained to search for spam emails, it first needed a dictionary so it could match keywords with that in the emails, however for it to be successful, it also needed to remove non dictionary words such as full stops, commas, spaces and more including removing common words used within the English language, as spam emails tend not to have very long text bodies. This is known as text cleaning and is an essential part of the classification process. Once the dictionary had been created, the machine needed a feature vector matrix, which was designed to use the training data with the dictionary and output how many times a word appears within the email. Once all the code was assembled, and the training and test folders had been specified, the program gave out 8 numbers representing what was spam and what wasn’t for each type of machine learning such as SVM and Naive Bayes.

# Algorithm

## Enron Code







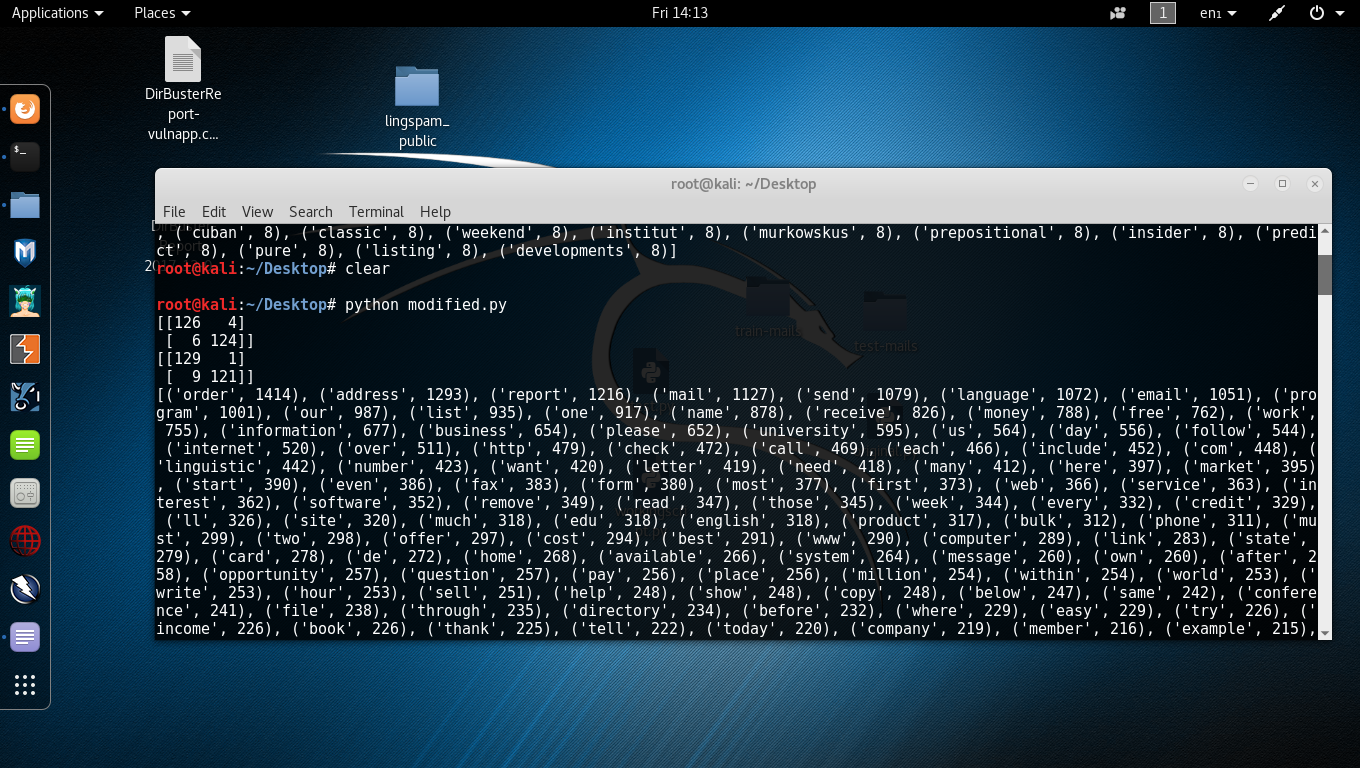
## Ling-spam Code



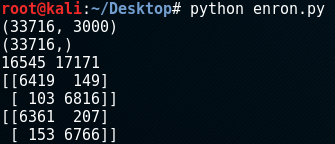
# 

# Results

After the machine had been created and was fully functioning, it started to successfully differentiate the data and outputted 8 different values relevant to each different machine learning method, such as SVM when supervised and Naive Bayes (NB). From here we were able to work out that each value was relating to whether the email corresponded to either not spam (ham) or into spam. As each dataset used contained a specific set of spam messages, we can work out how accurate the machine is for each method. When using the SVM method, the machine produced an accuracy rating of 95.38% when searching the test mail folder, whereas the NB produced an accuracy rating of 99.28%. The screenshots below provide the results from the ling-spam dataset and the Enron Corpus.

****

**Ling-spam**

****

**Enron Corpus**

# Evaluation

At the start of the project we had an extremely large and thorough flowchart, it encompassed 7 different types of email and had criteria such as whether the address was recognised or not, the number of recipients, multiple keyword readers, if there was attached files, whether there were executables and whether it was compressed or not. However this was not what was specified in the brief, we were told to make a machine that can differentiate between spam and ham emails; this lead us to having to drastically shorten our email criteria and streamline the flowchart to show this. In the end our flowchart only tested for key words known to be in spam emails and then further on from this it tested the word count in the email to define the email as either spam or ham.

When creating the actual script for the email filtering we had a lot of issues with personalising it to make it fit our data set and our criteria, at the start the script we made was coming up with a plethora of errors all down to improper placement and calling of variables; this issue was a simple one to solve but it took a while to find the exact area of issue. In the end we found that the problem was that the section that alienates special characters was in the wrong place so it was calling variables which weren’t defined yet which threw up errors. After we moved it down and into the correct area this problem was solved.  
  
The next issue that arose in code was that when python tried to initiate certain areas of the script it couldn’t find the parts that it needed to run even though they were there, this turned out to be a formatting issue and was down to poor indenting. The error came about because the improper indenting was pushing parts into different sections so it wasn’t being read and associated with the correct areas thus making the script unable to get past it. After reading through the script we found where the parts with incorrect indentations and moved them into the correct areas leading to the script reading through and completing itself.

# Conclusion

The main aim of this project was the script a machine that could split up a specific dataset of emails into the categories of spam and ham, whilst also gaining an understanding of machine learning and its many areas of characterization that come into play in order to move on to splitting them up into different categories.

To start the machine we created successfully completed the brief and was able to differentiate between the 2 different types of emails, it was overall a well worked project in which we overcame multiple challenges in order to reach our end goal. In the future we would maybe have used some type of script analyser to read through our work and pick out any little errors, this would have sped up the process massively and saved a lot of time which was spent on trying to source the errors and fix them.

With regards to learning about machine learning as a whole we believe we have picked up a great deal of knowledge in the subject due to detailed research and other methods, we have been able to pick up an understanding of areas within the subject such as lemmatization, stop words, SVM’s and the actual creation of a functioning machine learning script. At the beginning of the project we had a severe lack of knowledge in these areas but after its completion we all agree that we have a much stronger understanding of it and are much more competent in the area.

However machine learning isn’t just for filtering emails, it’s used every day in computer forensics in order to split up evidence and define a lot of different evidence in a lot of different categories. It is an incredibly beneficial area to have knowledge in and will definitely come into play in the future in forensic careers; a lot of the time the machines are pre-programmed and quite general, so having the knowledge on how to personalise them will be indispensable in future careers. Machine learning is getting more and more powerful every day and the need to learn it is ever growing.

# Future works

For future research we would like to see someone build on this and make a more thorough and advanced machine that can differentiate between even more types of email and move them into different mail boxes within a user's email. Also moving on from this it would be good progression to have image analysing incorporated into the script that could identify specific images and flag them up or use them to define an email with even greater detail, a lot of the time especially in emails containing adverts there are certain images that are common place and used to catch your eye; if someone was able to create a script to pick out these images it could help in moving the emails to the appropriate mailbox and not get mixed in with important mail.

It would be beneficial as a whole to try and incorporate machine learning into a lot of other different areas outside of emails, it could even be used in things such as psychiatric evaluation, a lot of the time the way a person writes can be an indicator of certain mental health implications and if a machine can be taught to pick these out and learn from past peoples indicators it could massively help in the diagnosis of issues and speed up the process in which help is given. Machine learning has a lot of potential and will hopefully be used a lot more all over the world in years to come.