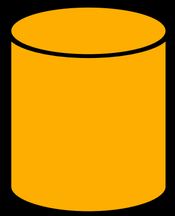
Filtering Spam E-Mails

Using Naive Bayes Classifier

Pavan Chauhan

Initial Dataset

Complete Spam Assassin
Enron Spam Subset
Ling Spam



- Exploratory Data Analysis
- Importing Libraries

Mails

- Spam
- Ham

We have three datasets that contains spam and ham mails.

Let's clear and concatenate them.

Cleaning & Preparing text

Now let's clean and prepare the text in order to use in Naive Bayes.

Cleaning links

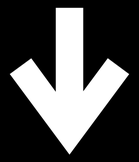
Cleaning digits except alphabetical and numerical characters.

Lowering

Tokenizing

Lemmatizing and Removing Stop words

Bag of Words



Pre-Processed Data

now the dataset is ready for modeling

Pre-Processed Data

data splitting

↓

fit the model

↓

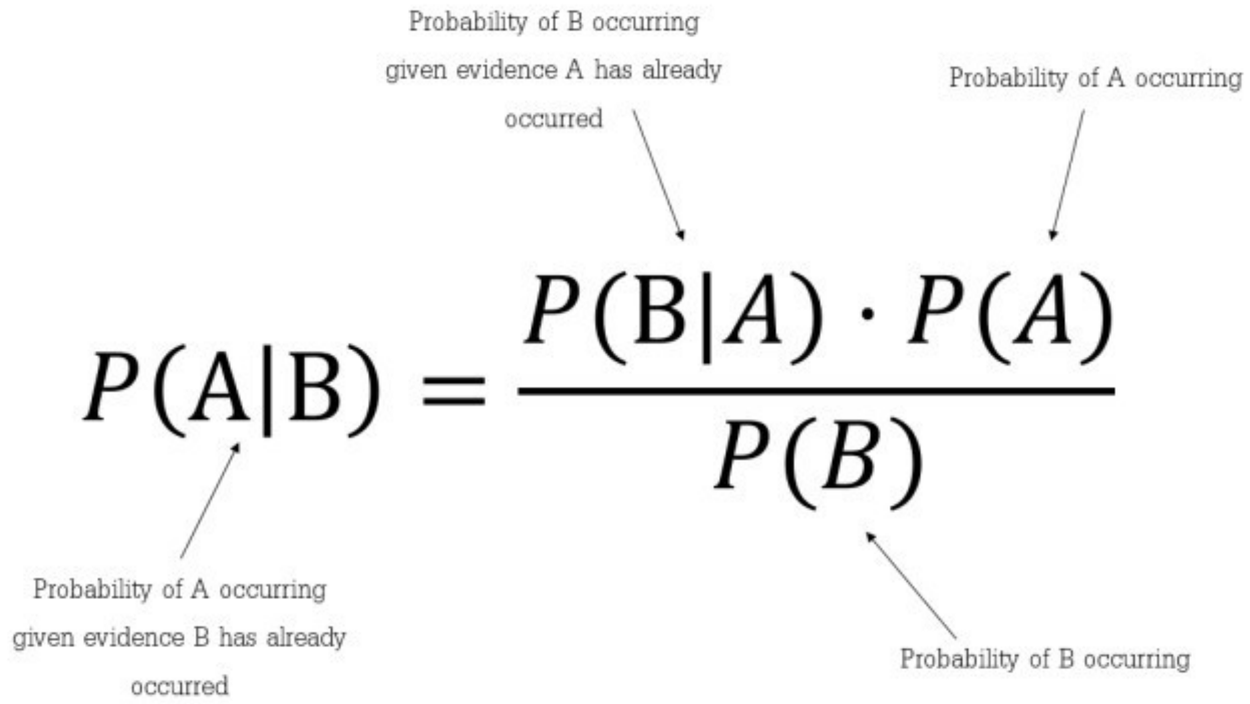
Evaluate

with

Accuracy & Confusion Matrix

About Bayes Theorem & Naive bayes

Bayes Theorem is a probability theorem that explained first by Thomas Bayes It has a simple formula, you can understand if you know basic probability.



Cleaning Links

- Cleaning Links

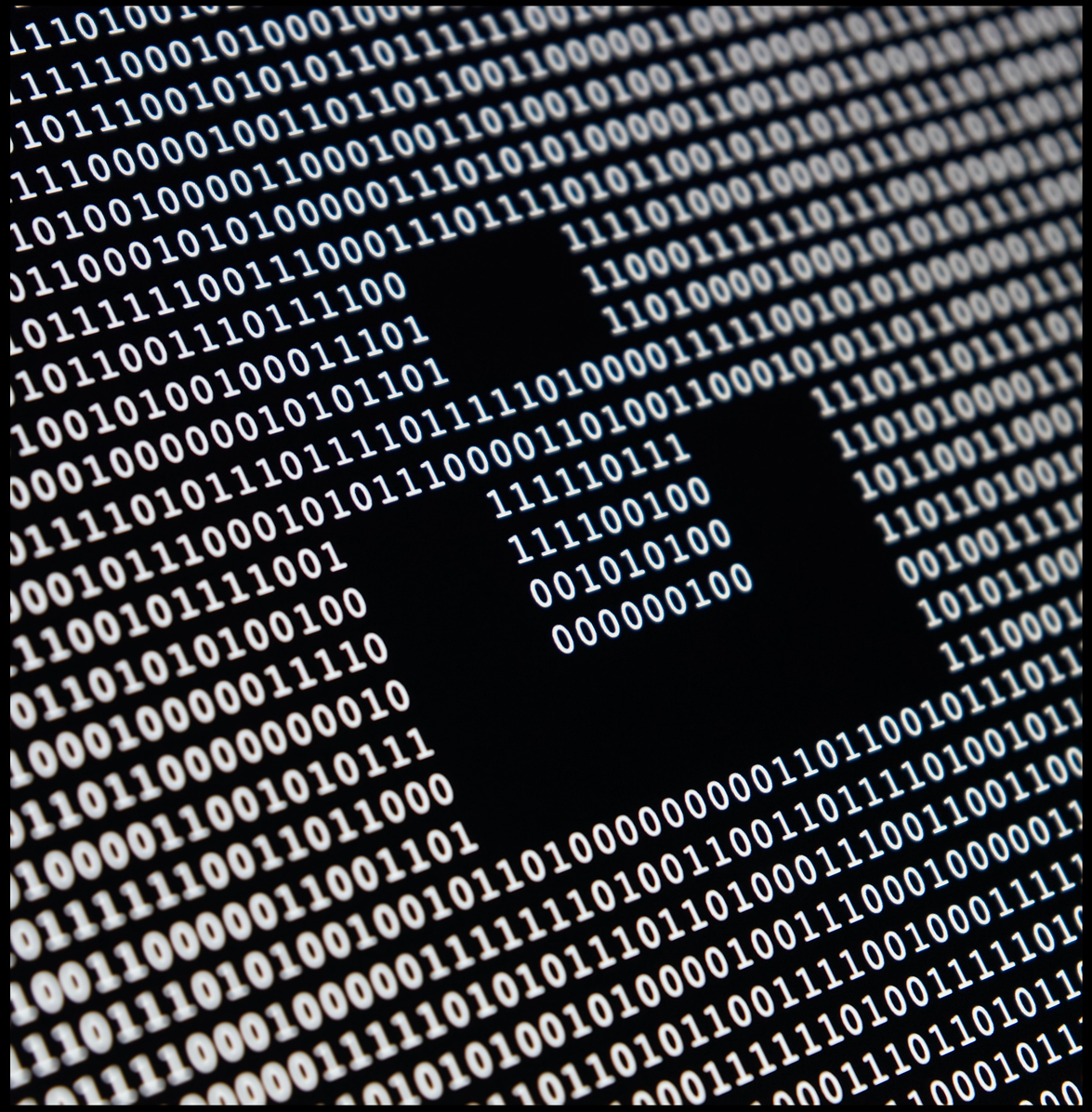
- As you can predict, there are links in a mails such as: https://google.com.tr If we don't remove them they can cause problems (problems might be small but we don't want problems in here :D)

- In order to clean links we will use regex (regular expressions) I will explain it.

Cleaning Digits Except Alphabetical and Numerical Characters

- As you can see from the text above, there are a lot of digits such as * and : They don't have a meaning, so we should remove them from the texts.

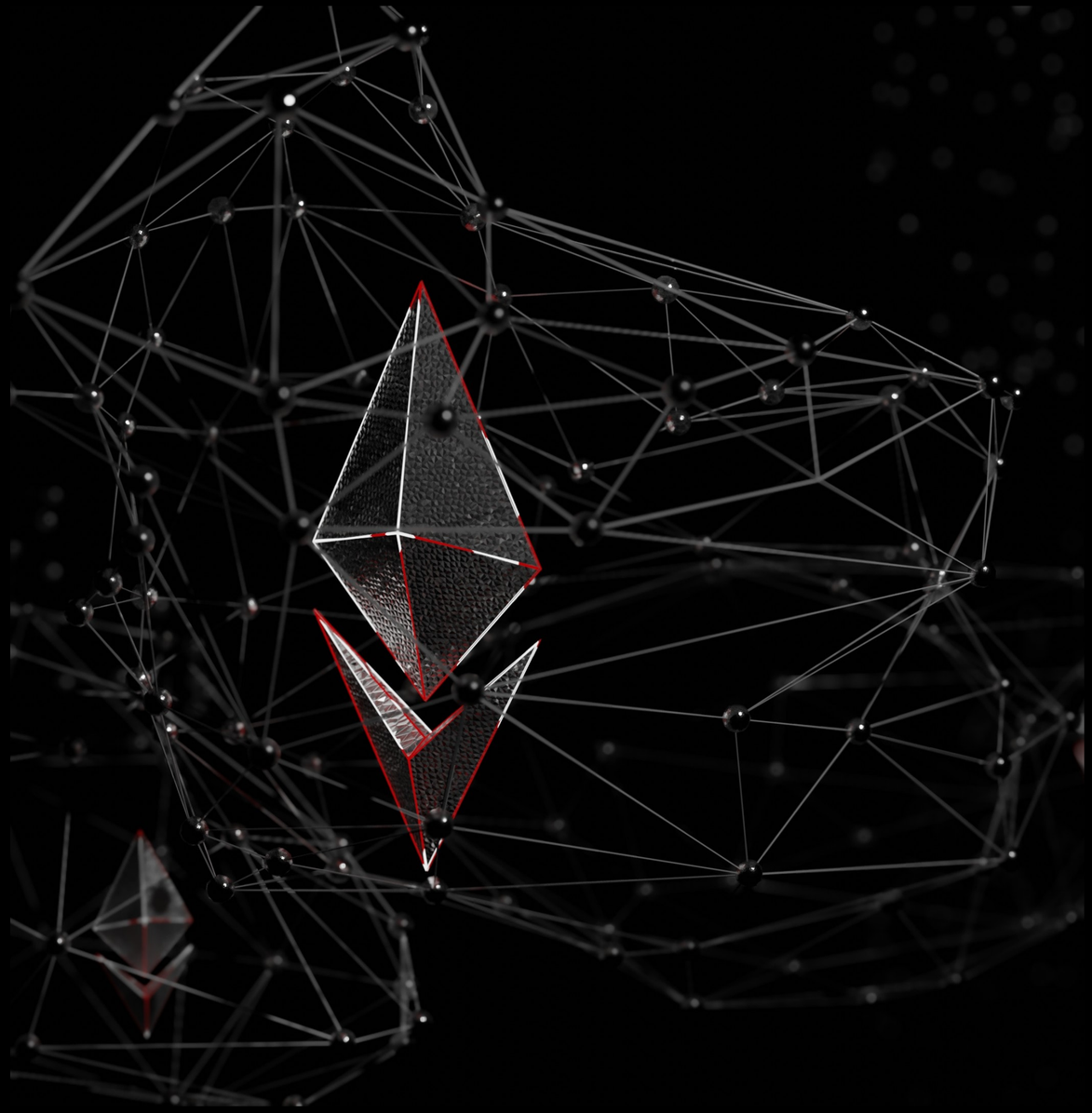
- In order to clean unrelevant digits we'll use regex again.



Tokenizing

In order to create a feature that shows whether the text includes the word or not, we need to split words into lists, we can do this using pythonString.split() but there is a better function to do this in NLTK.

Let's tokenize the texts.



Lemmatizing and Removing Stop words

In natural languages, words can get additional so each word can have a lot of versions, sometimes these additional may give tips to us but in filtering spams, we don't need them

There are two ways to remove additional: Stemmers and Lemmatizers

- **Stemmers** are rule based weak tools, they remove additional using rules but in natural languages everything does not follow the rules. Also It cant change tenses, for instance lemmatizers convert learnt into learn, stemmers don't touch them. Although stemmers are weak they are fast and although so many natural language do not have lemmatizers most of them have stemmers.

- **Lemmatizers** uses dictionaries to remove additionals and change tenses. They work good but developing a lemmatizer is hard and needs a lot of resource, so they are rare. Also lemmatizers use dictionaries, and that causes lemmatizers being slow.

Removing Stop-words

In natural languages there are words that not have a special meaning such as will, it is always a tense and such as and,or

In order to win from time and improve the model we should remove them. There are several ways to remove them but in this kernel we'll use stopwords corpora of NLTK. There are stopwords of 11 natural language in there.

Bag of words

And we came to the final process of this section: Bag of Words. Bag of Words is an easy approach to make sense of texts.

Bag of Words

And we came to the final process of this section: Bag of Words. Bag of Words is an easy approach to make sense of texts. In order to explain it I'll give an example

=======TEXTS======	WE	HAVE	SOME	WORD	HELLO	WORLD	FROM	PYTHON	I	APPLE	LOVE
We have some words	1	1	1	1	0	0	0	0	0	0	0
Hello world from Python	0	0	0	0	1	1	1	1	0	0	0
Hello I have some apples	0	1	1	0	1	0	0	0	1	1	0
I love the world	0	0	0	0	0	1	0	0	1	0	1