**A NAÏVE BAYES EMAIL SPAM CLASSIFIER**

**Introduction:**

The emergence of the electronic mail or e-mail facility has proven to be of immense help to organizations wanting to reach out to a large audience all over the internet, in attempts to increase their sales. However, many of these mails are in fact not genuine, and are called junk mail or **spam** mail. Spam mail causes a lot of trouble for the internet community: large amount of spam traffic between servers causes delays in transmission of legitimate email. Sorting out the spam email from the legitimate mail is a time consuming process and constantly involves the risk of deleting useful mail. The goal of this project is to implement a simple Naïve Bayes Email Spam Filter in Python.

**Spam Filtering:**

Spam Filtering is a binary classification task. Given an email, the goal is to classify an email into any one of the two categories: spam(junk) mail or ham(legitimate) mail. The textual content of the mail: words like “free”, “Viagra”, “lottery”, “Congratulations! You have won a 1,000,000 dollars! Click here to claim your price”: is crucial in spam detection and offers some of the strongest clues.

We will be using a Naïve Bayes Classifier for the purpose. The machine learning classifier that we will implement will detect that an email is spam or not based on some features that it will extract from the email, given the fact that it has been trained on a significant amount of labelled, training data. The performance of the email spam classifier depends considerably on the type of the classifier used, the size of the training data, and the quality of the features learned by the classifier. Moreover, the user also has the option to feed an input mail to the classifier through the code, and the result of the classification can be viewed.

The sample dataset that we will be using is the **Ernon Dataset**, which contains about half a million labelled emails, separately categorized into ham and spam emails. The dataset can be obtained from [here](https://labs-repos.iit.demokritos.gr/skel/i-config/downloads/enron-spam/preprocessed/).

The spam detection process involves five steps, which are as follows:

1. Loading the data
2. Pre-processing of the data
3. Extraction of the features
4. Training the Classifier
5. Evaluating the Classifier

**Step 1: Loading the data**

We need to access the dataset to be able to read the emails in the ham and spam subfolders and to load them into two separate lists. The two lists contain the words from the ham and spam emails respectively.

**Step 2: Pre-processing the data**

Currently, the data is stored in the form of sentences. However, to be able to use the words in these texts as features for the Naïve Bayes Classifier, we need to pre-process the data and normalize it, so that different forms of the same word are treated as the same word. Moreover, we need to tokenize the data, i.e. split the text by white spaces and punctuation marks using a tokenizer. This step is crucial because the Naïve Bayes Classifier expects the input text to be of a particular format, which is the main purpose of this step.

**Step 3: Extraction of the features**

Once the text is pre-processed, we can extract the features characteristic of ham and spam mails. Removal of stopwords, i.e. words like ‘a’, ‘an’, ‘the’, ‘this’ etc. is optional. The feature extraction step involves registering the fact that each word that was read appears in the email, and also assigning a label to the read data, i.e. ham or spam.

An alternative approach to this is the Bag of Words(BOW) approach, which simply removes the stopwords from the corpus and calculates the frequency of occurrence of each word in the text.

**Step 4: Training the Classifier**

Now that the data is in a format appropriate to be fed to the classifier, we can split the data into a training set, that will be used to train the classifier, and a test set which will be used to evaluate the classifier. Here we have split the original dataset into 80% for training and 20% for testing.

**Naïve Bayes Classifier:**

The Naïve Bayes Algorithm is a simple yet powerful classification algorithm which has been applied widely to the problem of spam filtering. The basis for this algorithm is the direct application of the Bayes’ Theorem, which states that:

The classifier tries to choose the most probable class for an email among ham or spam, based on what it has learned about the features (presence or the frequency of the words in the corpus). More precisely, it tries to assign a class c to an email such that

This statement means that the classifier will try to assign a class to the given email based on which of the two probabilities is higher- implies spam class will be assigned or implies ham class will be assigned. These probabilities can be calculated using the Bayes Theorem. Thus,

Now the classifier needs to compare these fractions for the two classes for every single email. However, the denominator remains the same in both the classes, and hence only the numerators need to be compared. We now have

The Naïve Bayes Classifier assumes that each word in the corpus occurs independently of all other words, so we can multiply the conditional probabilities for each of the words directly. Therefore, the algorithm will classify an email as spam if,

and ham otherwise. We now have a working spam filter which assigns labels ham or spam to emails depending upon the features it has learned from the input corpus of emails.

**Step 5: Evaluating the classifier**

After the classifier has been trained, we now need to evaluate the performance of the classifier. This is usually done by writing code to check the accuracy of the classifier on the testing set. Accuracy on the training set tells us how good the classifier is in learning appropriate features of the data it has been trained upon, and accuracy on the test set tells us how well the classifier generalises this information learnt to predict classes of emails it has not seen before. Another way to evaluate the classifier is to write a line of code to find out the features which are most significant while performing the classification.

**Conclusion:**

In this project, we have successfully implemented a Naïve Bayes Email Spam Filter in Python. The dataset used is the Ernon Dataset. Results show that the classifier has an accuracy of more than 90%.

The source code for the same can be found here.