We decided to create a simple spam filtering script by using basic machine learning techniques. Spam filtering is basically a document classification task which involves classifying an email as spam or non-spam.

The dataset used is the enron corpus which contains a large number of spam and non-spam mails, which can be used to train and test our model.

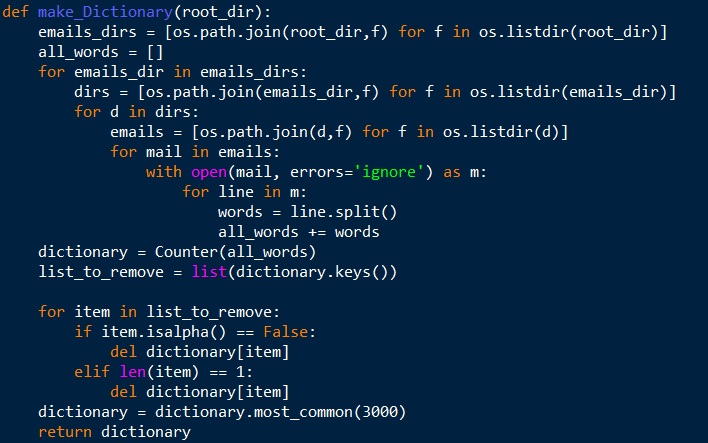
The main steps to create this application are:

1. **Preparing the text data.**
2. **Creating word dictionary.**
3. **Extract the relevant features.**
4. **Training the classifier.**

**Preparing the text data**:

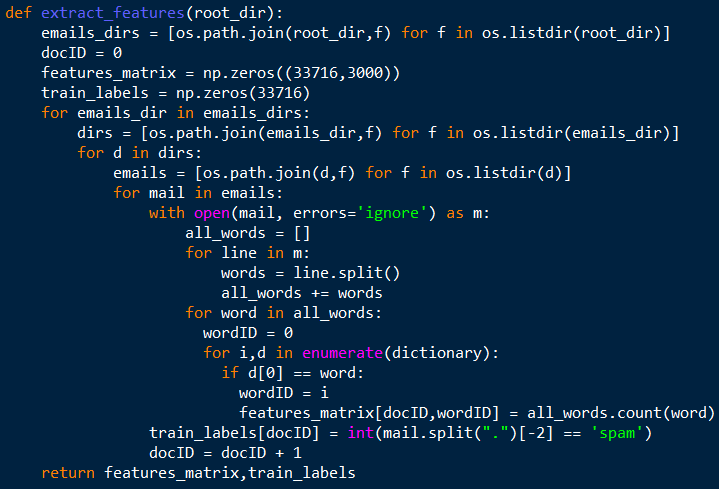
1. We perform text cleaning where words that do not contribute to the information we want to extract are removed. This includes undesirable characters like punctuation marks, stop words, digits which may not be helpful in detecting the spam email.
2. The enron corpus has already been cleaned in these ways:
   1. Removal of stop words like “and”, “the”, “of” which are not meaningful in classifying a email as spam or non-spam.
   2. Lemmatization where different inflected forms of the same word are grouped together so they can be analysed as a single item. For example, “exclude”, “excludes” and “excluded” would all be represented as “exclude”.

**Creating the word dictionary**:



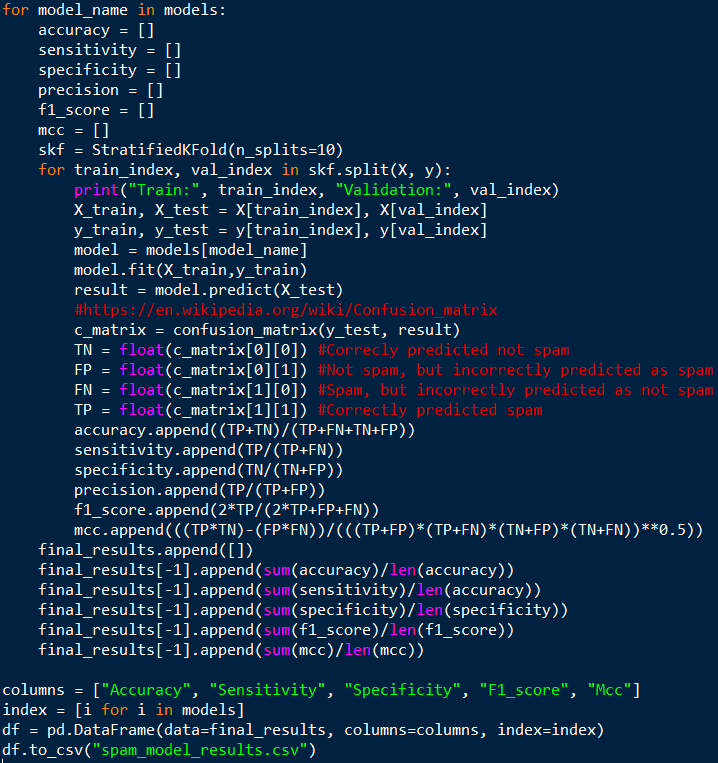
1. The code above depicts how the word dictionary is created.
2. We first generate the path for all the relevant directory. Next we open each individual mail, and add each individual word to a list of all words.
3. Next, using the “Counter” module, we create a dictionary, which the word as key and the frequency of occurrence of the word as value.
4. Next, we remove words from the dictionary if it consists of only alphabetic characters.
5. Lastly, we only keep the most common 3000 words.

**Extracting the relevant features**:

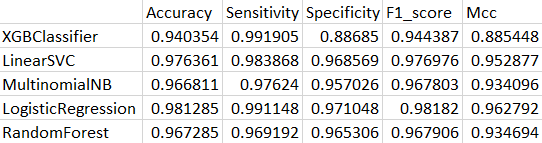


1. The code above depicts how we extract the features for training data.
2. After the dictionary is generated, we can extract the word count vector for each email of the training data. Each word count vector has the number of times each of the 3000 words appear in the training file.
3. We will create a features matrix which has the dimensions – (number of training samples, 3000). Since each row corresponds to one training sample, the number of rows is equal to the number of training samples. The number of columns is 3000 as we need to represent how many times each of the 3000 words appear within that training sample.
4. We will also create the training labels which will tell us whether each row in the features matrix is a spam or non-spam.

**Training the classifier:**



1. We tried several machine learning models, without tuning any parameters. The models are implemented with the python library – scikit-learn.
2. For each model, we used stratified 10-fold cross validation technique to measure how well each model performs. This is important to prevent overfitting and it is necessary in order to measure how well the model will generalise to unseen data. How this works is that the data is split into 10 partitions, in which 9 are used as training set to fit the model and the last partition is used as testing set to evaluate the model. This is repeated 10 times, with each partition being the test set each time. The average of the 10 times can then be taken. The stratified method is used to account for the different number of spam and non-spam emails. Stratified method will ensure each partition has the same ratio of spam to non-spam emails. The implementation for this part is achieved using the StratifiedKFold method from scikit-learn.
3. How each model is evaluated – we first calculate the confusion matrix using the scikit-learn library. This will give us the True Positives (emails correctly predicted spam), False Positives (emails incorrectly predicted as spam), True Negatives (emails correctly predicted as not spam) and False Negatives (emails incorrectly predicted as not spam).
4. Using the values in step 3, we can calculate various metrics to evaluate the model. The formulas for these metrics are taken from Wikipedia. These metrics are:
   1. Accuracy = (TP+TN)/(TP+FN+TN+FP)
   2. Sensitivity = TP/(TP+FN) which measures the proportion of actual positives that are correctly identified as such.
   3. Specificity = TN/(TN+FP) which measures the proportion of actual negatives that are correctly identified as such.
   4. F1\_Score which is another way to measure accuracy based on precision and recall.
   5. Matthew Correlation Coefficient is another to measure accuracy by considering true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. This is particularly relevant since we have a different number of non-spam and spam mails.
5. Lastly, we saved the evaluated metrics for each model into a neatly formatted csv.



**References**

<https://www.kdnuggets.com/2017/03/email-spam-filtering-an-implementation-with-python-and-scikit-learn.html>

<http://www.aueb.gr/users/ion/data/enron-spam>

<https://en.wikipedia.org/wiki/Confusion_matrix>

<https://scikit-learn.org/>