Cork Institute of Technology

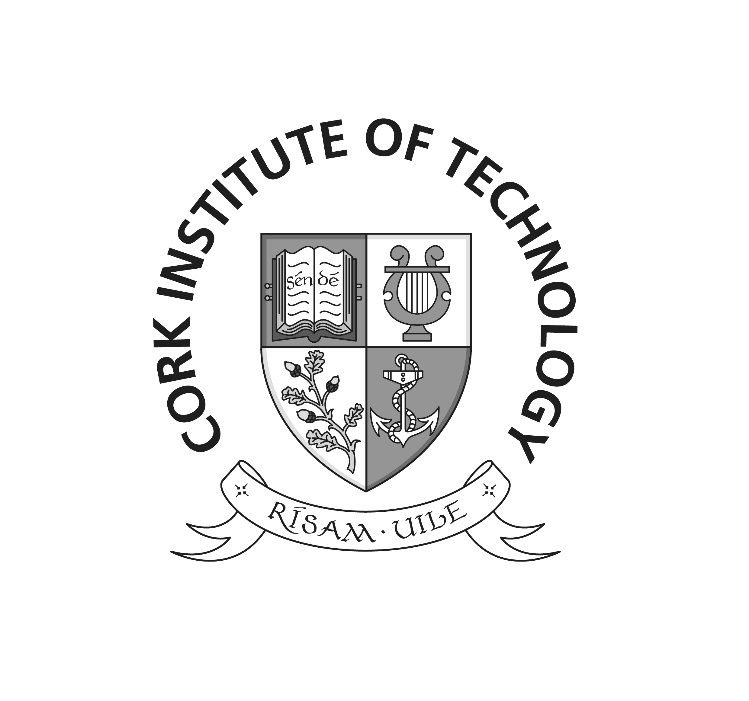
MSc in Data Science and Analytics

Applied Machine Learning

Project 1 Assignment

**Givaldo Silva Jr**

**R00182595**



Dr. Haithem Afli

4th April 2019

Sumário

[1. **Introduction** 3](#_Toc36913093)

[**2.** **Load dataset** 3](#_Toc36913094)

[**3.** **Preprocessing and clean the dataset** 3](#_Toc36913095)

[**4.** **Training and test splits** 6](#_Toc36913096)

[**5.** **Feature extraction** 8](#_Toc36913097)

[**6.** **Exploratory data analysis** 8](#_Toc36913098)

[**7.** **Supervised classification** 16](#_Toc36913099)

[**8.** **Model selection** 17](#_Toc36913100)

[**9.** **Model evaluation** 19](#_Toc36913101)

# **Introduction**

The dataset that we will use for this assignment is the Enron email dataset. You can find the full dataset on the web here: http://www.aueb.gr/users/ion/data/enron-spam/. The dataset is a collection of public domain emails from the Enron corporation. The emails have been manually classified as spam and non-spam. The primary goal of the assignment is to create a supervised classification pipeline to classify emails as spam or non-spam from the training data. You are free to use either the preprocessed emails or the raw emails for your analysis.

I did not put all information here in this report because you asked maximum 20 pages, and in the jupyter notebook it is not following the same sequence as your, then they complement each other.

# **Load dataset**

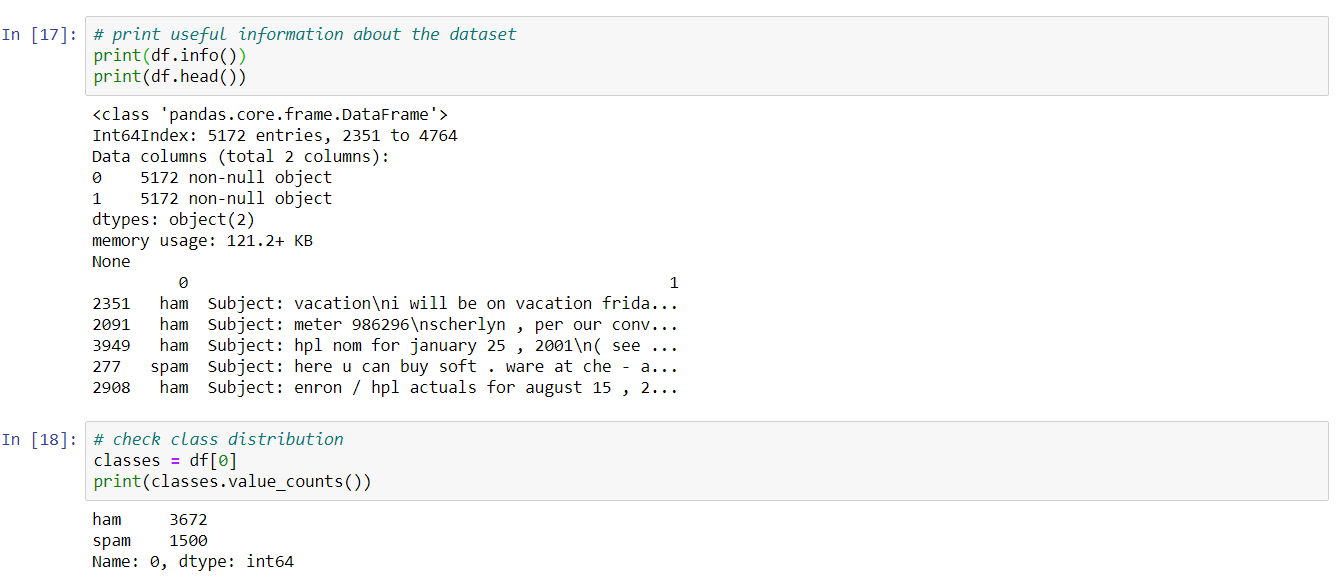
All emails are separated in different files, we need to load everything, and in this case, I have opted by working with one unique dataset.



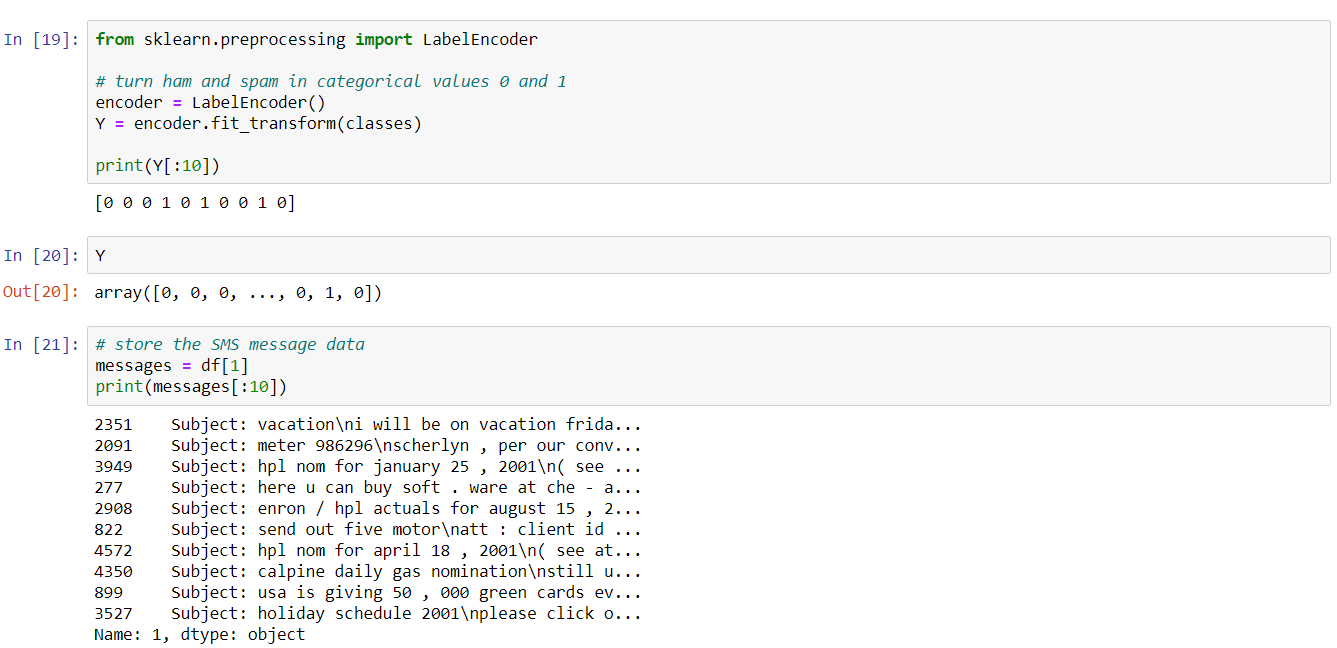
# **Preprocessing and clean the dataset**

Here we use the function shuffle to mix the emails, this will avoid some bias calculus. For example, when we split the data set into training and testing, we need to take care if the subsets are balanced, otherwise, we could have much more spam than ham in the training dataset for example.



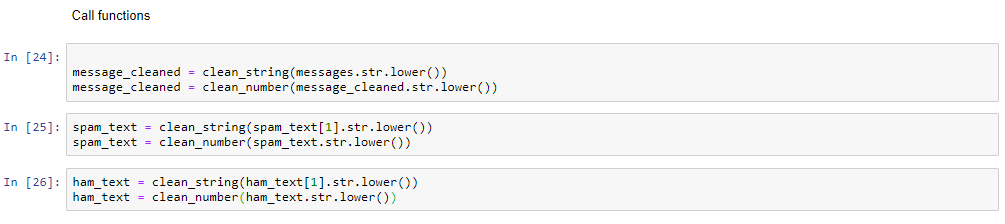


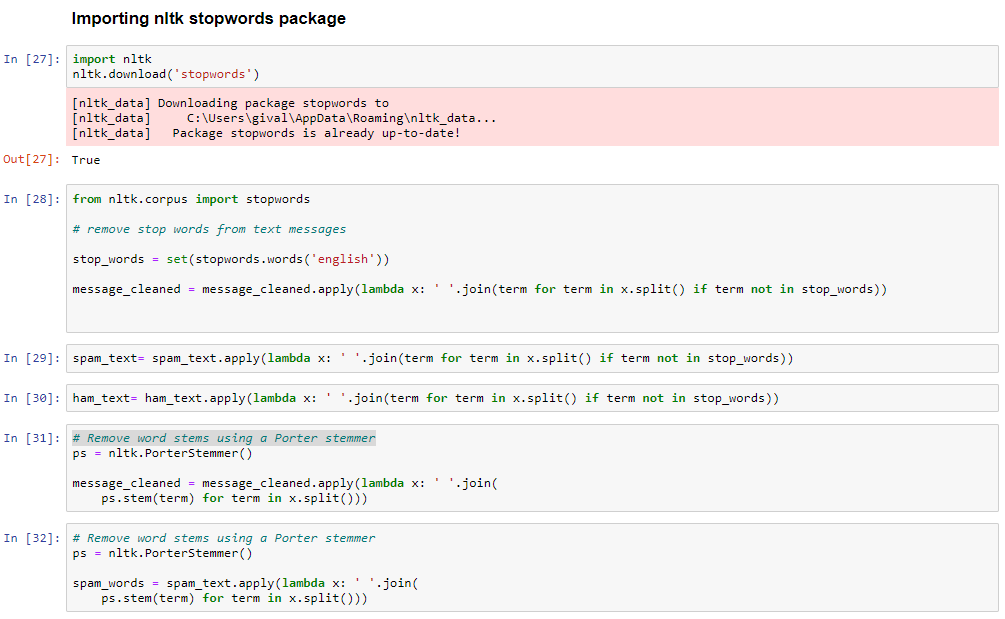
Here we are categorizing the dataset into 0 and 1 (ham and spam).



A lot of words, symbols and sometimes numbers are not good for prediction, actually it can have a negative effect in the algorithms, then, we need to clean as much as possible and try to get just the key words that really represent a Spam or ham email.









# **Training and test splits**

The very first thing that you will need to do is split the data into training and test sets. **Write a**

**Python script to perform the split: 70% of the data for training and the remainder for test**. Take

appropriate measures to ensure that the test set is not biased in any way. **There may be duplicate**

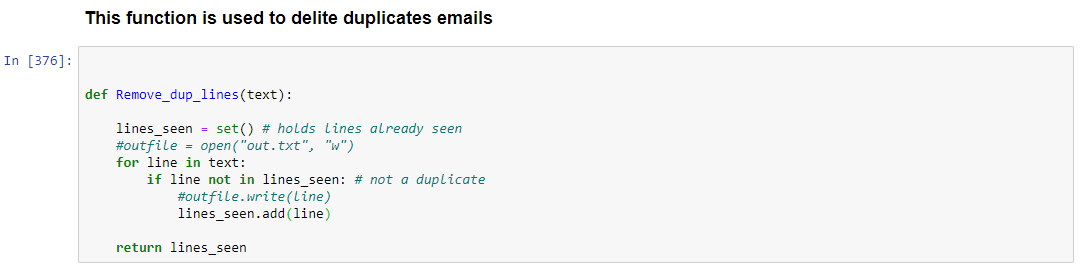
**and empty emails that need to be handled appropriately.** Store the resulting training and test

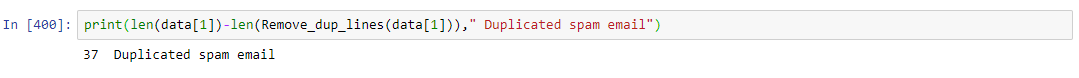
sets in files using any convenient data format that you like. Collect and record statistics on the

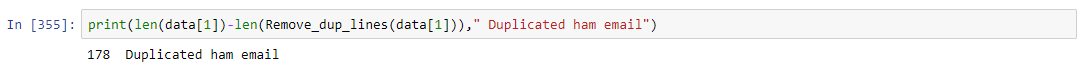
resulting training and test sets including total numbers of spam and non-spam emails in each

set.

I used the function train\_test\_split from sklean to split the dataset into train and test subsets.









# **Feature extraction**

The second part of preprocessing will be to extract the features you will need for the remainder

of the analysis. You may revisit this stage many times as you become more familiar with the

dataset and the kinds of features that may be useful for the classification task. You may want

to start with using a bag-of-words model here to transform the documents into a fixed length

representation suitable for classification. The sklearn.feature\_extraction.text package may be

useful here.

The features you choose will affect the performance of the final classifier, and there are many

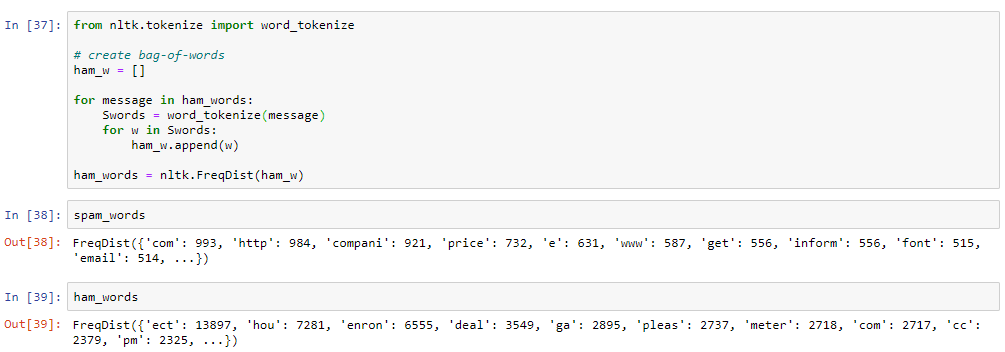
possibilities (e.g. stop word removal, TF-IDF encoding, infrequent word removal, etc.). Choose

something you think is reasonable to start with and later you can experiment with alternatives

on the validation set.

In order to run machine learning algorithms, we need to convert the text files into numerical feature vectors, I am using nltk.tokenize to do that.





# **Exploratory data analysis**

Use the training section of the dataset to perform some exploratory data analysis. The goal at

this stage is to become accustomed with the data and gain insights into the kinds of features

may be useful for classification.

Find the top-20 most frequently used words in spam and non-spam emails and use a bar plot

to show their relative frequencies. Compare the distribution of email lengths in spam and non-

Applied Machine Learning –

Project 1 Assignment

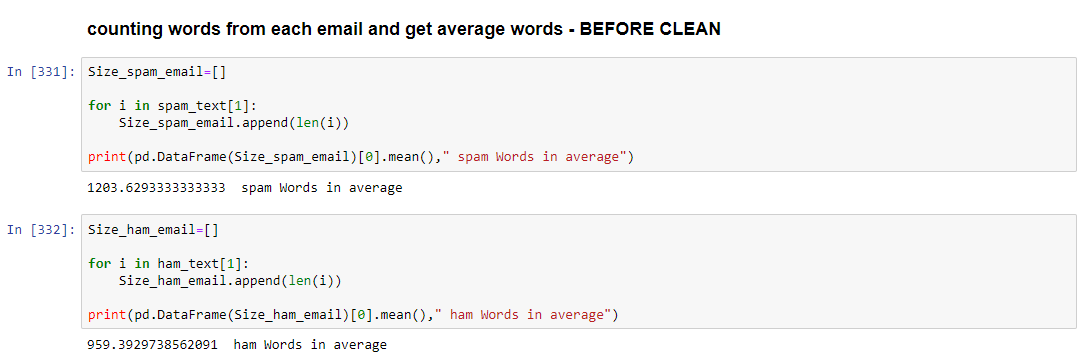
spam emails using appropriate plots (e.g. a boxplot). **Are spam emails usually shorter or longer?**

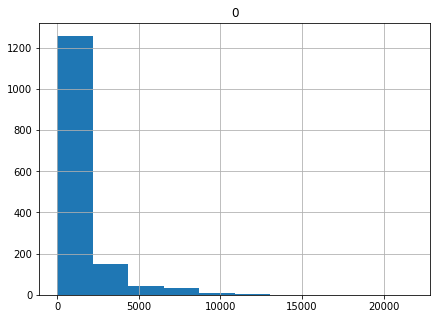
Look at the corresponding data points for the longest emails. Explain why these are so much

longer than most others. Document all your findings and any other interesting observations that

you can find.

Here we can see that spam emails have more words in average than ham emails.





Figure

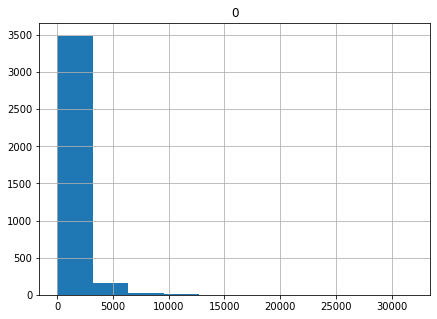


Figure 2

Here we have the average number of word (considering a 1000 most frequent words) after cleaning the emails. Ham emails has more words than spam emails in average.

In figure 3 we have the histogram from ham emails, the frequency of words in each email. Most part of emails have around 300 words and the maximum 3500 words.

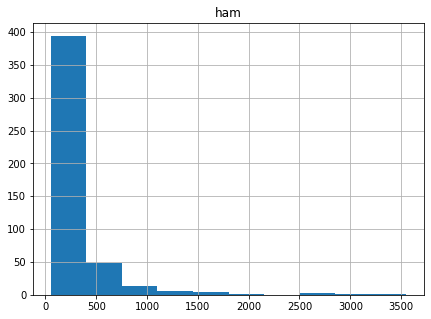


Figure 3

About spam e-mails, we can verify that most part of them have around 100 words and the maximum 1000 words.

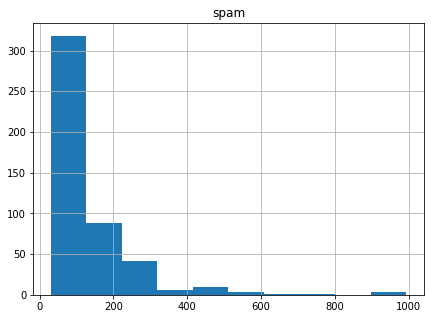


Figure 4

In this boxplot, figure 5, it is easy to see the concentration of the number of words between ham and spam, and their outliers.

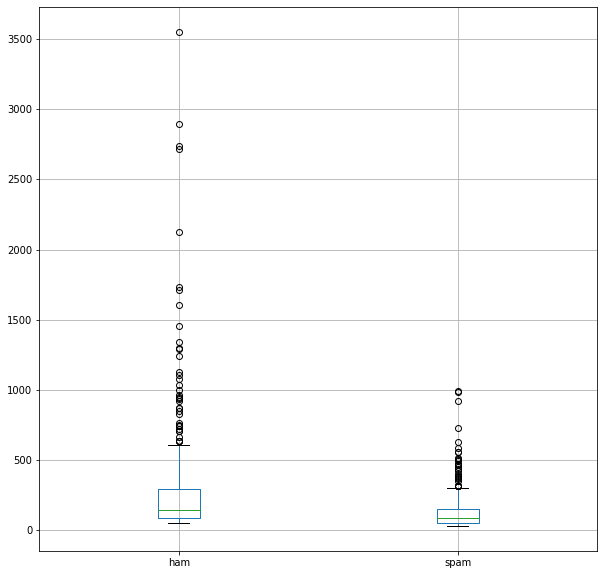


Figure 5

In this scatterplot (figure 6) I got similar words in both category, ham and spam, with their respective frequency. We can see that the words with higher frequency in spam axis could be considered a spam word like for example the word “http”,“compani” or “price” and the opposite is true the word “pleas”, “thank” and “deal” are more frequent in ham emails.

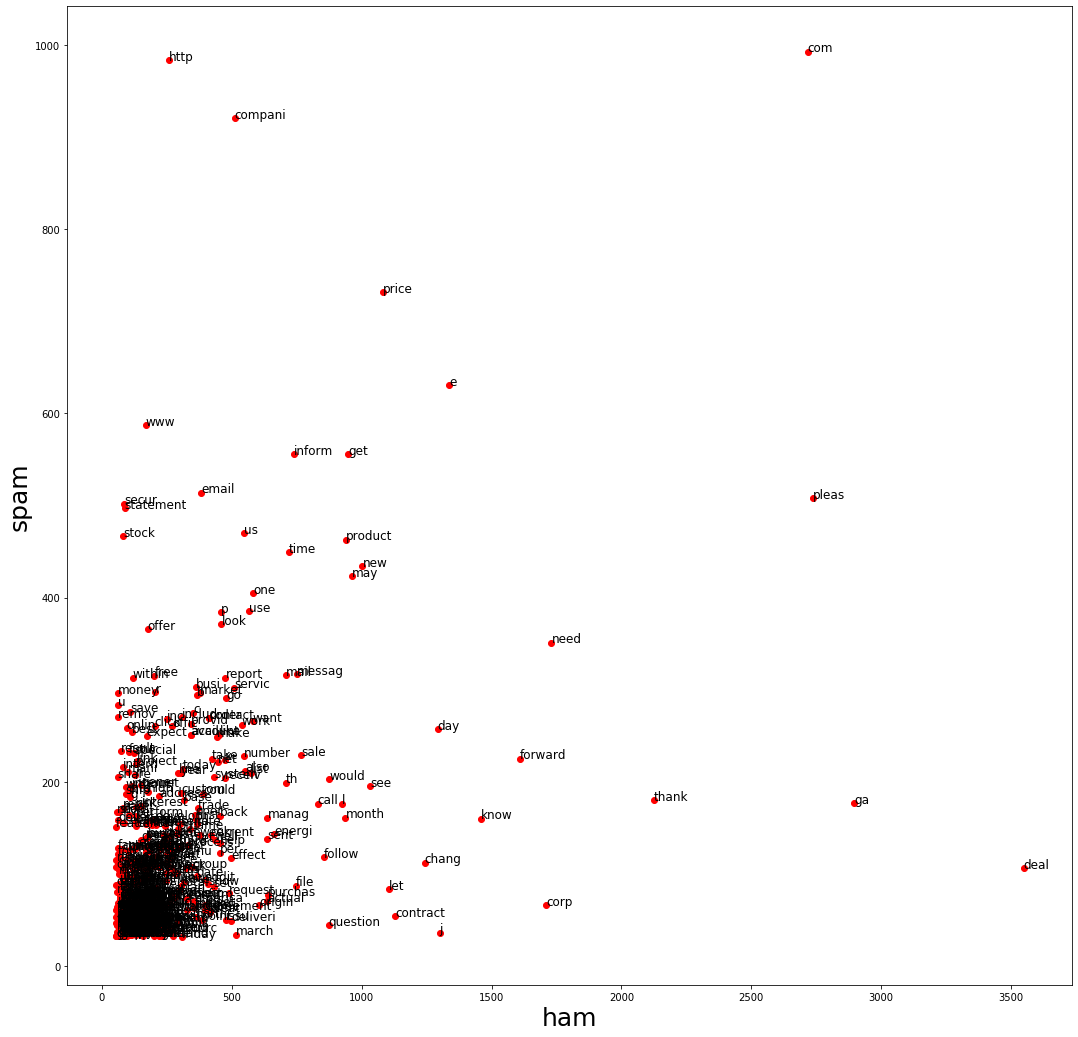


Figure 6

Considering the 30 most frequent words in the whole dataset in figure 7

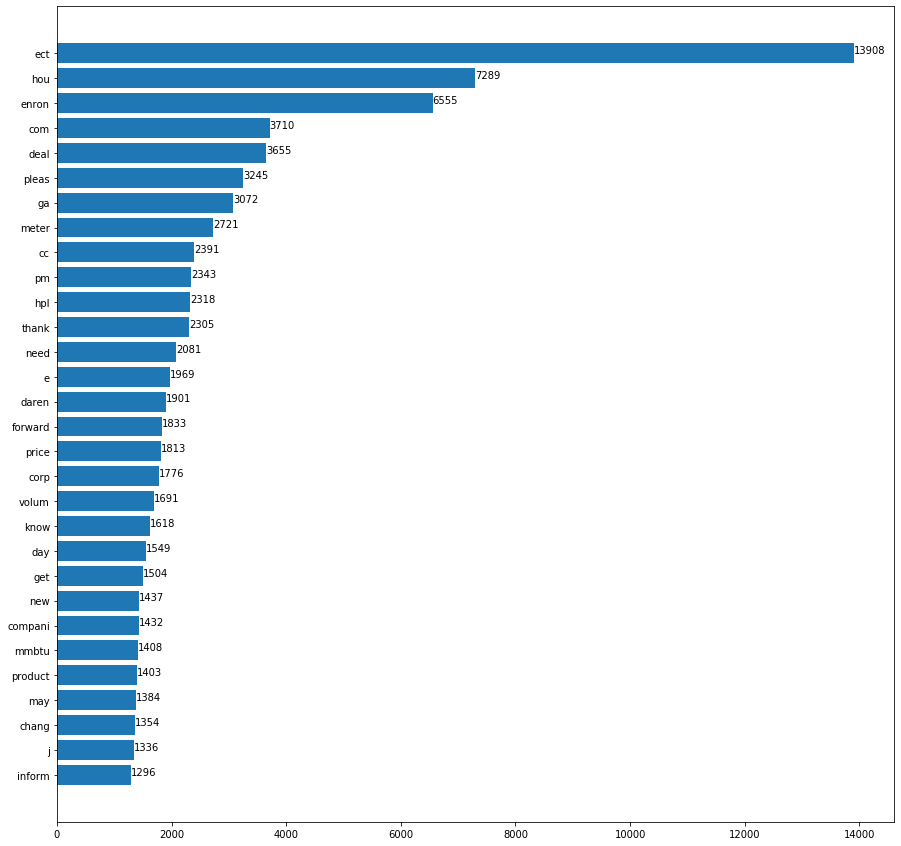
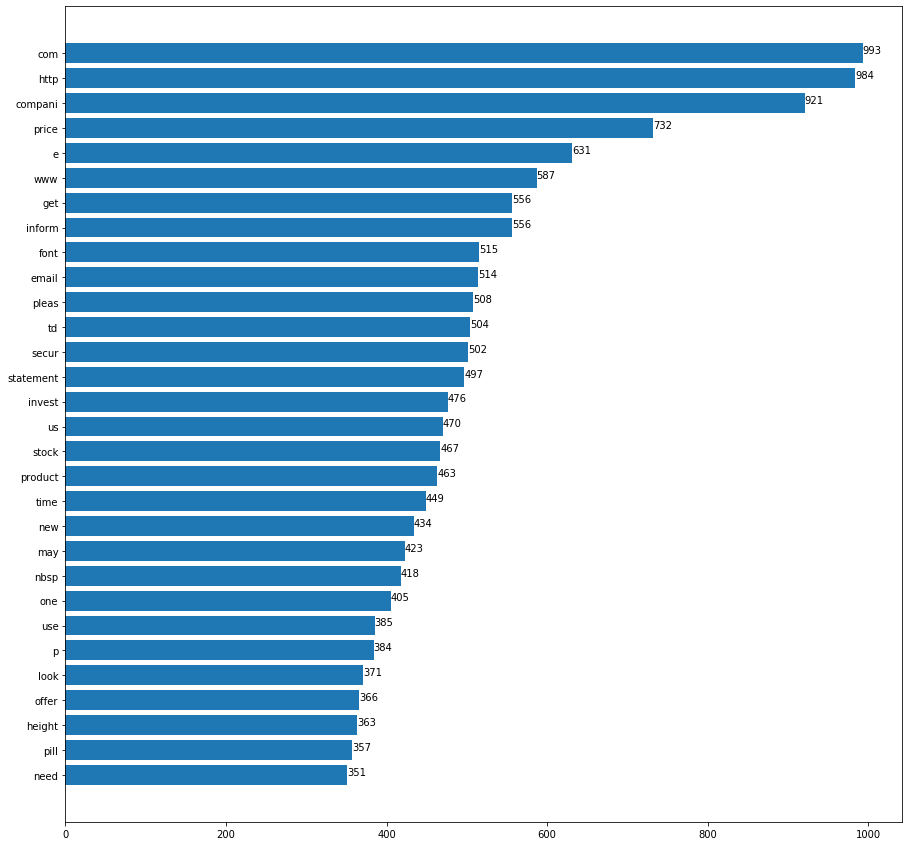


Figure 7

In figure 8 and 9 we can see the 30 most frequent words about spam and ham respectively. Spam words seem to have more homogeneity, differently from the entire dataset and ham subset, that have some words much more frequent than the rest of words.



Figure

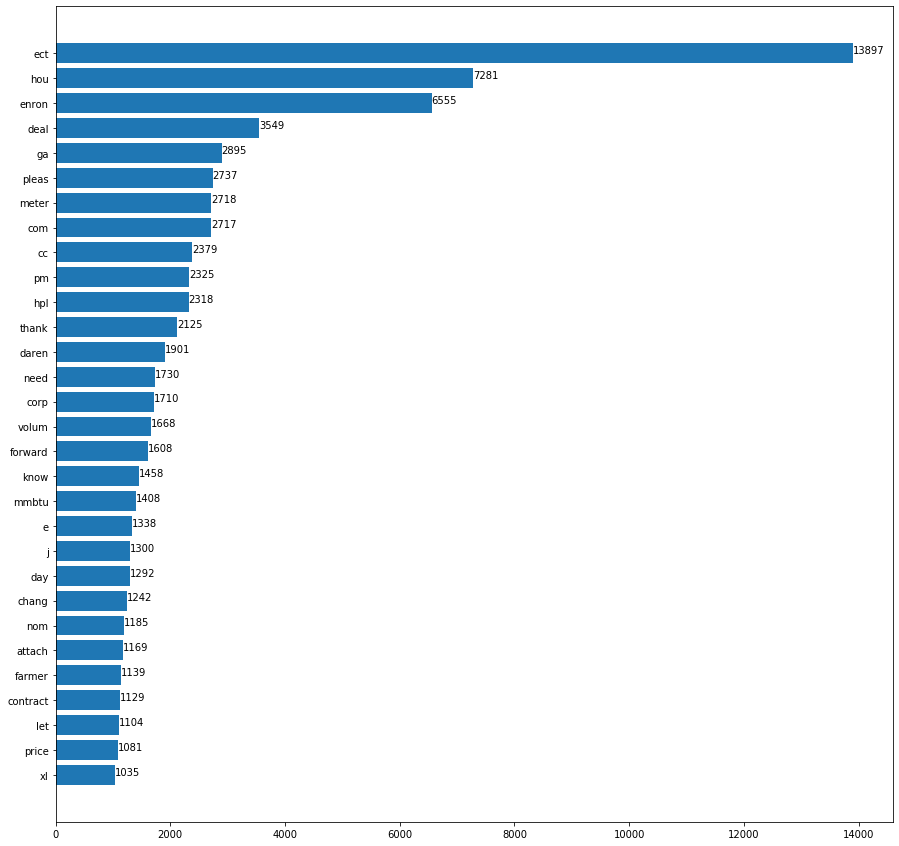
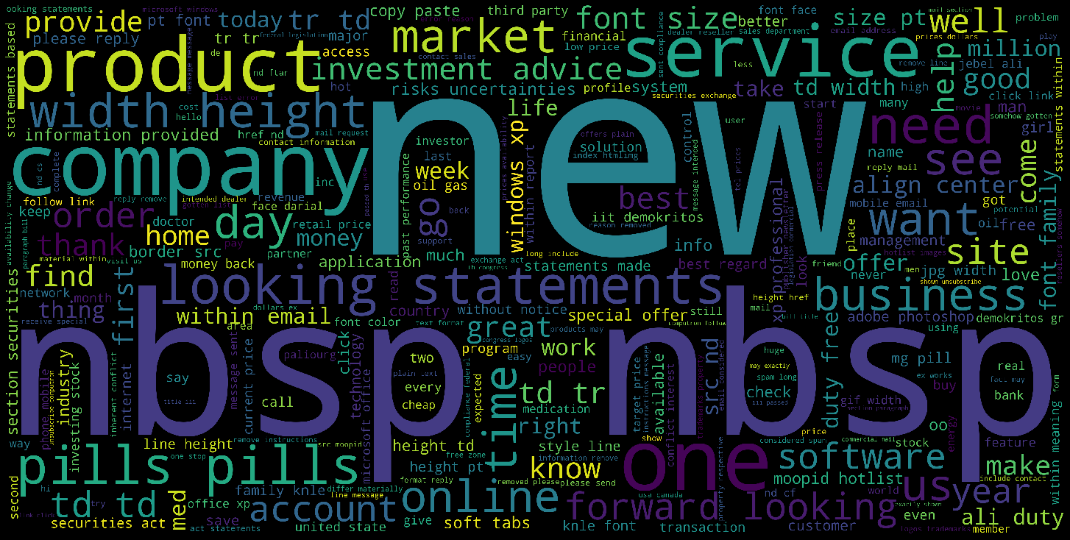


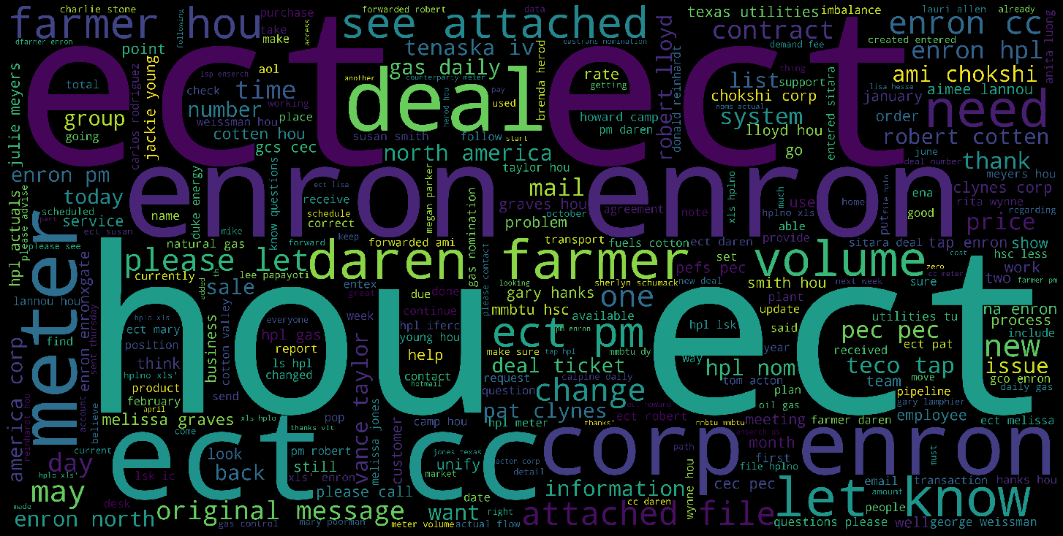
Figure 9

At figure 10 and 11 we can see a word cloud from Spam words and Ham words respectively.

These images were generated using the initial file emails all combined, this function gets the most frequent words in the text, it is good to have a first analysis about the text.

****

Figure

****

Figure

# **Supervised classification**

Train a supervised classification model on your features and calculate **validation accuracy**

either on a holdout validation set or using **cross validation**. Record the final accuracy of the

classifier. How many of the emails are correctly classified by the model? How many are

misclassified? Use the **sklearn.metrics** package to investigate the kinds of errors that are being

made. Document all findings. Use the **Python pickle** module to save the model to the disk.

Here I am using to train my first model.

I got an accuracy of 95.87 % with 42 errors ham predicted and 22 errors spam predicted.

Text classification can be described as assigning texts to an appropriate bucket. To train a text classifier, we need some annotated data. This training data can be obtained through several methods. considering this case, we label the email as Ham and SPAM.

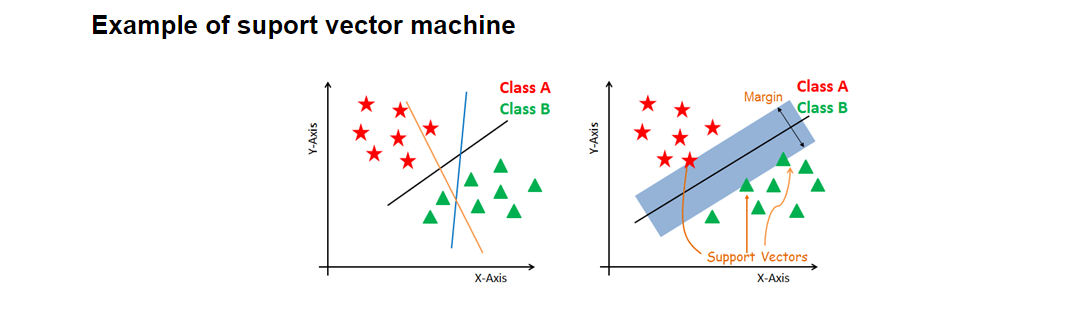


Figure 12

In this algorithm (SVM), we plot each data item (words) as a point in n-dimensional space (in this case n is the frequency of words as spam or ham), with the value of each feature being the value of a given coordinate. So, we performed the classification by finding the hyperplane that differentiates the two classes very well (figure 12).

Confusion matrix is used to measure the results, it will show us how good is the model to predict which email is a spam or ham, in that case we can see a high accuracy with some feel mistakes.

Something important to say is that we should expect some errors, it is normal, if we have something like 100% of accuracy it can be result of over fitting, when the model learn so much that it will not work so well it another dataset.



# **Model selection**

Select several candidate models that you want to compare. This could include different

classifiers (e.g. naive Bayes MultinomialNB), different hyperparameters, or different sets of

features (remember, hyperparameter selection is part of model selection!). Use a validation set

or cross-validation to compare the accuracy of **different models**. Create plots to compare a

subset of the models that you investigated during model selection. Retain the most effective

model for evaluation.

Here we are using different techniques (algorithms) to create different models for the same dataset and after we will measure the accuracy of each one to see which is the best.

**Supervised neighbors-based** learning has two methods: classification for data with discrete labels, and regression for data with continuous labels.

The nearest neighbor methods use a predefined number of training samples closest in distance to the new point and predict the label from these.

The distance can be any metric measure, like for example, standard Euclidean distance.

**Decision three classifier** broke the features into several nodes according with their proportions, it starts in the root node.

Entropy is a measure of uncertainty in data and information gain is expected reduction of entropy. The more balanced (spam and ham), closer to 1 the entropy will be. The algorithm splits the data to reduce entropy and increase the purity of the data, so classification will be easier. The highest info gain will be a good option to be the root node.

**Random forests** algorithm is an ensemble method, we don’t need split the dataset in train and test, because random forest will try lot of combination, it means lots of trees, and it will choose the best tree as result. Random forest can cover a much bigger space.

Advantages: Handle large dataset with higher dimensionality will not overfitting the model classification and regression task handle missing values and maintain accuracy for missing data.

Disadvantage: We don’t have much control on what the model does, just some parameters and set seeds regression is good but classification not so much.

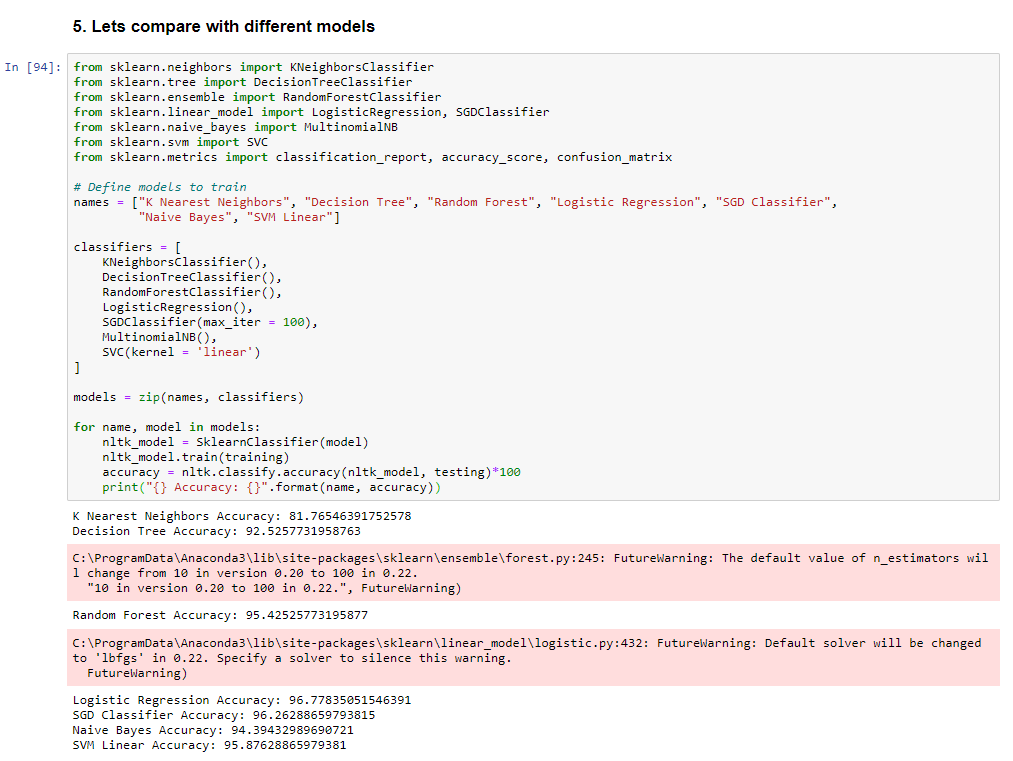
Hyperparameters the number of trees in the forest, it will affect the accuracy but also time to process it.

**Naive bayes** has a probabilistic approach, using conditional probability theory: P(A|B)= P(A&B)/P(A) probability of A given B divided by the individual probability of A

Random experiment: consider that each variable has an independent participation in some situations it is possible to achieve better results with naive bayes than with neural networks. there is no better algorithm than the other, there is the best one for a given problem good when the dataset is not too big.

* good for real time
* the result can be classified as a binary result from a treshoud
* three types of models in the Scikit-learn library:
  + Gaussian consider normal distribution
  + **Multinomial** count of discrete variables (example: count how many times a word appears in the text) as we are using in this case.
  + Bernoulli binary models (example: check if a word occurs in a text or not)

Example of use: multiclass forecast Span filter Mining of emotions feeling analyze Document Separation



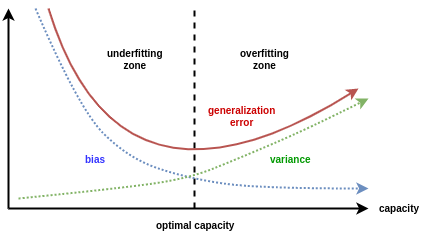
# **Model evaluation**

Estimate the **out-of-sample error** for the model that you found to be **most accurate** during model

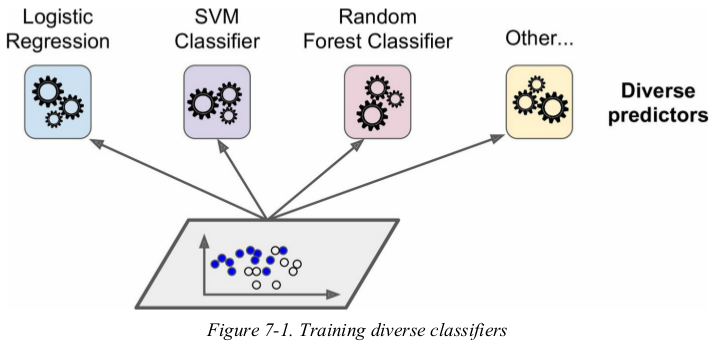
selection by evaluating it on the held-out test set. Use the sklearn.metrics package to benchmark

the model in several ways. **Create a graph plot for the model.** Compute the model's accuracy

on the test set. Comment on the implications of the resulting for a real production classifier.



**Out-of-sample error** is used to verify how accurately an algorithm is to predict values for unseen data.



**Voting** ensemble estimates multiple base models and uses voting to combine the individual predictions to arrive at the final ones, we can train different models, for example, a Decision Tree and a Logistic Regression, and then use the Voting Ensemble to combine the results.

We have wo types of voting:

**Hard** voting makes the final prediction by a simple majority vote for accuracy.

**Soft** Voting happens when all your classifiers can calculate probabilities for the outcomes. It arrives at the best result by averaging out the probabilities calculated by individual algorithms.

