

Τεχνητή Νοημοσύνη 2018-2019

Γιάννης Παπαγεωργίου 3160129

Δεύτερη Προγραμματιστική Εργασία:Υλοποίηση Bayes

Η κλάση naive με δοσμένο μονοπάτι ενός μέιλ αναγνωρίζει το μέιλ ως ham ή spam.

Η κλάση test παράγει τα σημεία για κάθε μια καμπύλη και με την βοήθεια του octave παράγονται τα ανάλογα διαγράμματα.

Αρχικά έχουμε την κλήση `ArrayList<String> seussham = seussre(hampath)`, όπου στο `arraylist<String>seussham`, μέσω της `seussre(String path)` μπαίνουν όλες οι λέξεις του εκάστοτε μέιλ (default delimiter το κενό). Έπειτα δημιουργείται `hashmap<String,Integer> seussCountH`, όπου `String` κάθε μοναδική λέξη του `seussham` και `Integer` ο αριθμός εμφανισής της στο `seussham`. Ομοίως για `spam`.

Εδώ λοιπόν έχουμε κάνει το train επί της ουσίας. Έπειτα με δοσμένο μέιλ, καλούμε την `seussone`, όπου δίνουμε σε `arraylist<String>`(με default delimiter) όλες τις λέξεις του μέιλ.

Μετά με την `cleararr(arraylist<String>)` “καθαρίζουμε” το `arraylist`, από λέξεις που μπορούν να μας αποπροσανατολίσουν από το όρθο αποτέλεσμα.Ύστερα μέσω των `hashmap` και του `arraylist` του μέιλ, φτιάχνουμε τους πίνακες `frequenciesham`, `frequenciesspam`, όπου αποθηκεύονται οι συχνότητες κάθε λέξης ανάλογα με το πόσες φορές κάθε λέξη του `arraylist` υπάρχει στα δεδομένα `ham` και `spam`.Τέλος γίνεται υπολογισμός και σύγκριση πιθανοτήτων `ham` or `spam`, με την βοήθεια των προηγούμενων πινάκων και της `Math.log1p`, ενώ παράλληλα γίνεται εφαρμογή Laplace.`Math.log1p` διότι έχει την καλύτερη δυνατή ακρίβεια για το υπολογισμό του $\ln(1+p)$ καθώς αντί για $P=p_1 \cdot p_2 \cdot \dots \cdot p_n$, υλοποιείται ο ισοδύναμος υπολογισμός $P=P+\text{math.log}(p_n)$, ο οποίος δίνει ακόμα μεγαλύτερη ακρίβεια και αποφυγή πιθανού underflow. Έπισης μπορεί κάποια πιθανότητα να υπολογιστεί μεγαλύτερη της μονάδας,πράγμα άτοπο για πιθανοτητα, αλλά εφόσον μιλάμε για το παραπάνω άθροισμα είναι απολύτως φυσιολογικό λόγω του οτι μιλάμε για άθροισμα και όχι γινόμενο.

Τώρα για την κλάση test έχουμε όλα τα παραπάνω, απλά την διαδικασία που ακολουθούμε για το ένα δοσμένο μέιλ το ακολουθούμε για όλα τα σετ των μέιλς to train, που δημιουργήσα manually(18%-mails1,36%-mails2,54%-mails3,72%-mails4,90%-mails5, 90-100%test).Για κάθε ένα τέτοιο σετ(mails1-mails5 υπολογίζω αρχικά την ακρίβεια προπόνησης με ευνόητους υπολογισμούς,ενώ παράλληλα δημιουργούνται confusion matrixes(με τους κατάλληλους ελέγχους) για κάθε τέτοιο σετ. Μετά υπολογίζεται το test error, ανά train set.Να σημειωθεί πως έχουν προκαθοριστεί πίνακες 5 θέσεων που αποθηκεύονται όλα τα αποτελέσματα(κάθε αποτέλεσμα για κάθε ένα από τα 5 σετ),πλην των 5 confusion matrixes που είναι μεγέθους 2X2.Τέλος εφαρμόζονται οι τύποι των precision,recall,f1 και επίσης μπαίνουν τα αποτελέσματα σε πίνακες μεγέθους 5, ώστε να έχουμε εύκολο print.

Κοινές μέθοδοι:

```
public static ArrayList<String> seussone(String path) throws  
FileNotFoundException
```

Διαβάζει το αρχείο από το path και επιστρέφει ένα arraylist<String>, όπου περιέχει όλες τις λέξεις του αρχείου χωρισμένες ανά κενό.










```
public static ArrayList<String> seussre(String path) throws  
FileNotFoundException
```

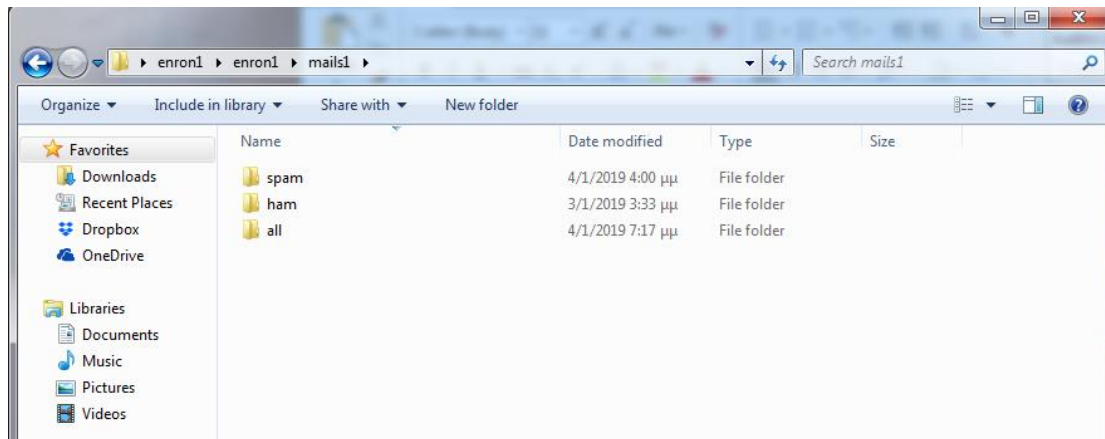
Από το δοσμένο path για κάθε αρχείο που περιέχει εντάσσει τις λέξεις(χωρισμένες ανά κενό) σε ένα arraylist<String> το οποίο και επιστρέφει.

```
public static void cleararr(ArrayList<String> arr)
```

Δέχεται ως όρισμα ένα ArrayList<String>, περιέχει έναν πίνακα από String (avoid) και μια λίστα list. Για κάθε στοιχείο του arraylist αν περιέχεται σε αυτό κάποιο string από τον πίνακα avoid, τότε εισέρχεται στην λίστα. Αφού τελειώσει αυτή η διαδικασία, απλά αφαιρούμε από το arraylist, όλα τα στοιχεία της λίστας, ώστε να επιτύχουμε τον επιθυμητό “καθαρισμό”.

Μορφή των αρχείων που δημιούργησα manually:

Name	Date modified	Type	Size
 ham	13/12/2018 4:57 μμ	File folder	
 mails1	4/1/2019 7:15 μμ	File folder	
 mails2	4/1/2019 7:44 μμ	File folder	
 mails3	4/1/2019 7:45 μμ	File folder	
 mails4	4/1/2019 7:46 μμ	File folder	
 mails5	4/1/2019 7:46 μμ	File folder	
 spam	13/12/2018 4:57 μμ	File folder	
 test	4/1/2019 8:09 μμ	File folder	
 Summary	29/10/2005 11:06 μμ	Text Document	1 KB



Οπού κάθε ένα από τα mails1-5 + test, περιέχει έναν φάκελο με τα σπάμ ανά σετ , τα χαμ και όλα μαζί.Τους δύο πρώτους φακέλους γιά την εισαγωγή και δημιουργία των ανάλογων hashmaps και το all για την διαδικασία του υπολογισμού των πιθανοτήτων.

Αρχικά βλέπουμε τα αποτελέσματα του κώδικα της κλάσης naïve.

```

46
47     String askedpath="C:\\Users\\John\\Desktop\\enron1\\enron1\\spam\\0153.2004-01-07.GP.spam.txt";
48     ArrayList<String> asked=seussone(askedpath);
49
50     frequenciesham = new double[asked.size()];
51     frequencyssпам = new double[asked.size()];
52     int ok=0;
53     String s1;
54     cleararr(asked);
55     for(int i=0; i<asked.size(); i++)
56     {
57         s1=asked.get(i);
58
59         if(seussCountH.containsKey(s1)) {
60             frequenciesham[ok]=seussCountH.get(s1);
61         }else {
62             frequenciesham[ok]=0.0;
63         }
64         if(seussCountS.containsKey(s1)) {
65             frequencyssпам[ok]=seussCountS.get(s1);
66         }else {
67             frequencyssпам[ok]=0.0;
68         }
69         ok++;
70     }
71
72
73
74     double sumham=0;
75     double sumspam=0;
76
77     for(int i=0; i<ok; i++) {
78
79         //calculating possibilities
80
81         if(frequenciesham[i]!=0) {

```

Problems @ Javadoc Declaration Console

<terminated> naïve [Java Application] C:\Program Files\Java\jre1.8.0_112\bin\javaw.exe (20 Ιαν 2019, 5:44:45 μ.μ.)

Its ham

Οπού για ένα δοσμένο μειλ αποφάνθηκε αν είναι ham or spam.

Στις παρακάτω εικόνες φαίνονται όλα τα αποτελέσματα που παράχθηκαν,για την δημιουργία των καμπυλών.

```

0.20860215053763442
0.17419354838709677
0.14229390681003584
0.0986559139784946
0.10774193548387101
===== Training error
0.25287356321839083
0.1954022988505747
0.17816091954022983
0.13793103448275867
0.22605363984674332
===== Testing error
659.0 1.0 191.0 79.0
1310.0 10.0 308.0 232.0
1960.0 20.0 367.0 443.0
2566.0 74.0 275.0 805.0
2959.0 341.0 140.0 1210.0
=====confusion matrixes
0.7752941176470588
0.8096415327564895
0.842286205414697
0.90320309750088
0.9548241368183285
=====precisionham
0.9875
0.9586776859504132
0.9568034557235421
0.9158134243458476
0.7801418439716312
=====precisionspam
0.9984848484848485
0.9924242424242424
0.98989898989899
0.9719696969696969
0.8966666666666666
=====recallham
0.29259259259259257
0.42962962962962964
0.5469135802469136
0.7453703703703703
0.8962962962962963
=====recallspam

```

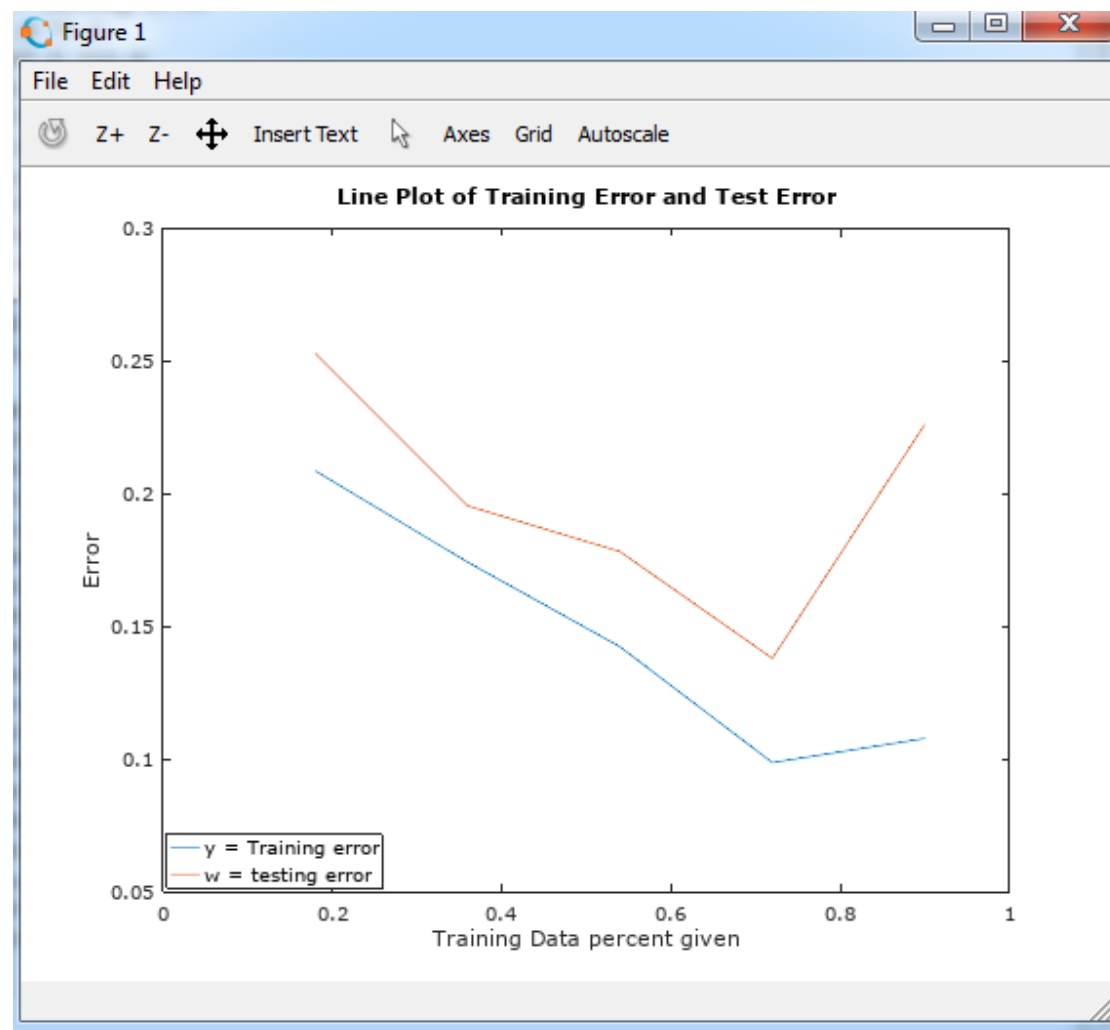
```

-----
=====recallspam
0.8728476821192052
0.8917631041524848
0.9101462735082424
0.936325488049626
0.9248320050007813
=====f1ham
0.45142857142857146
0.5933503836317136
0.6959937156323646
0.8218478815722307
0.8341951051361599
=====f1spam

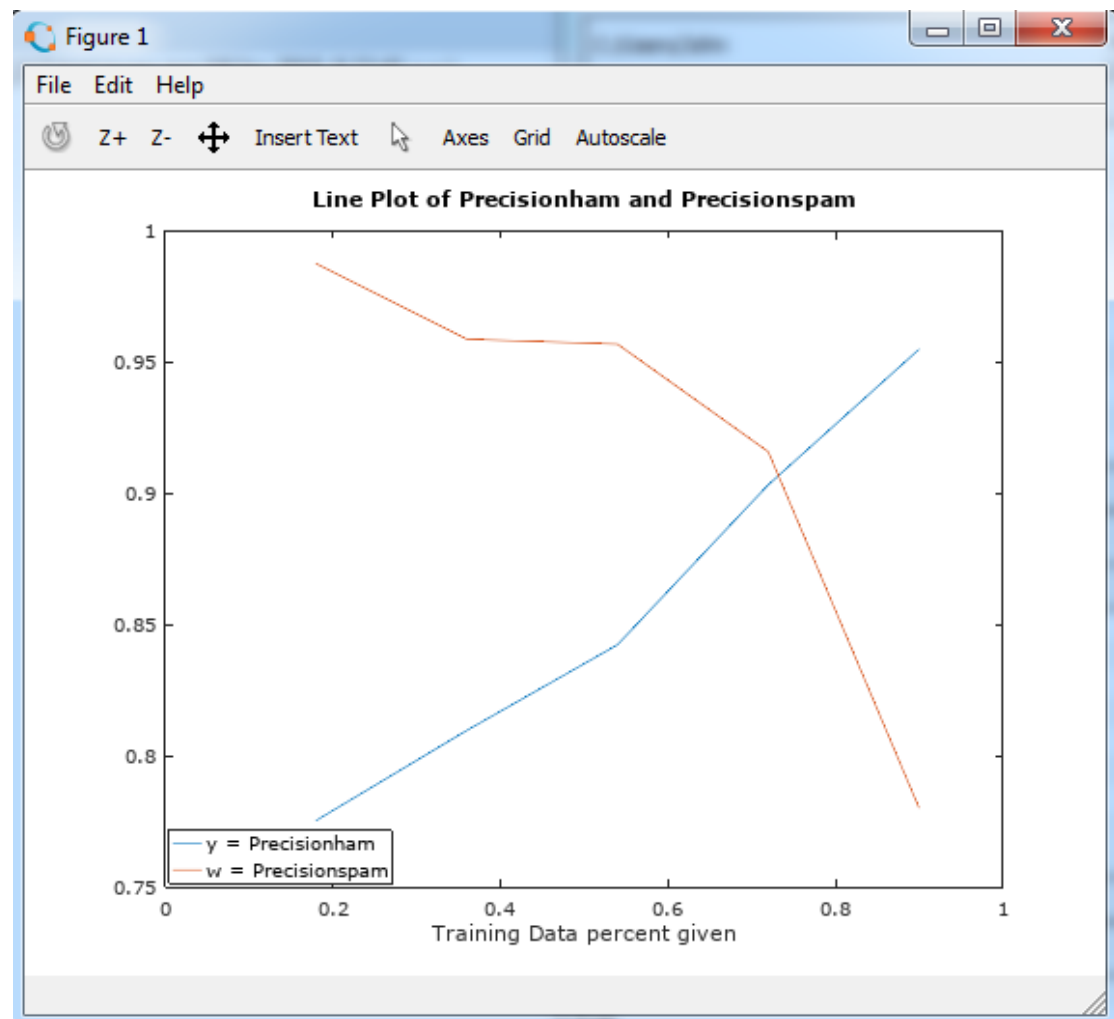
```

Οι καμπύλες:

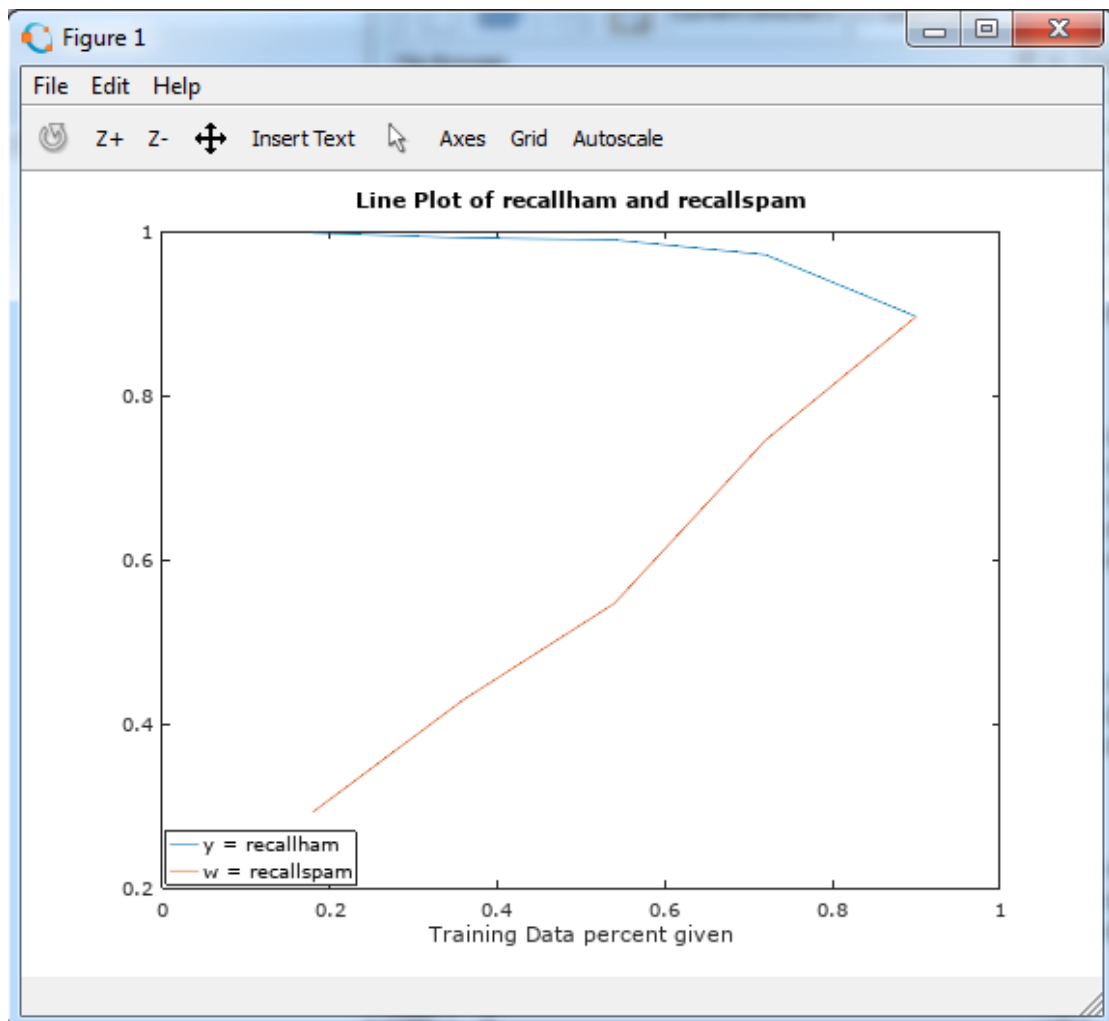
1) Training-Test error



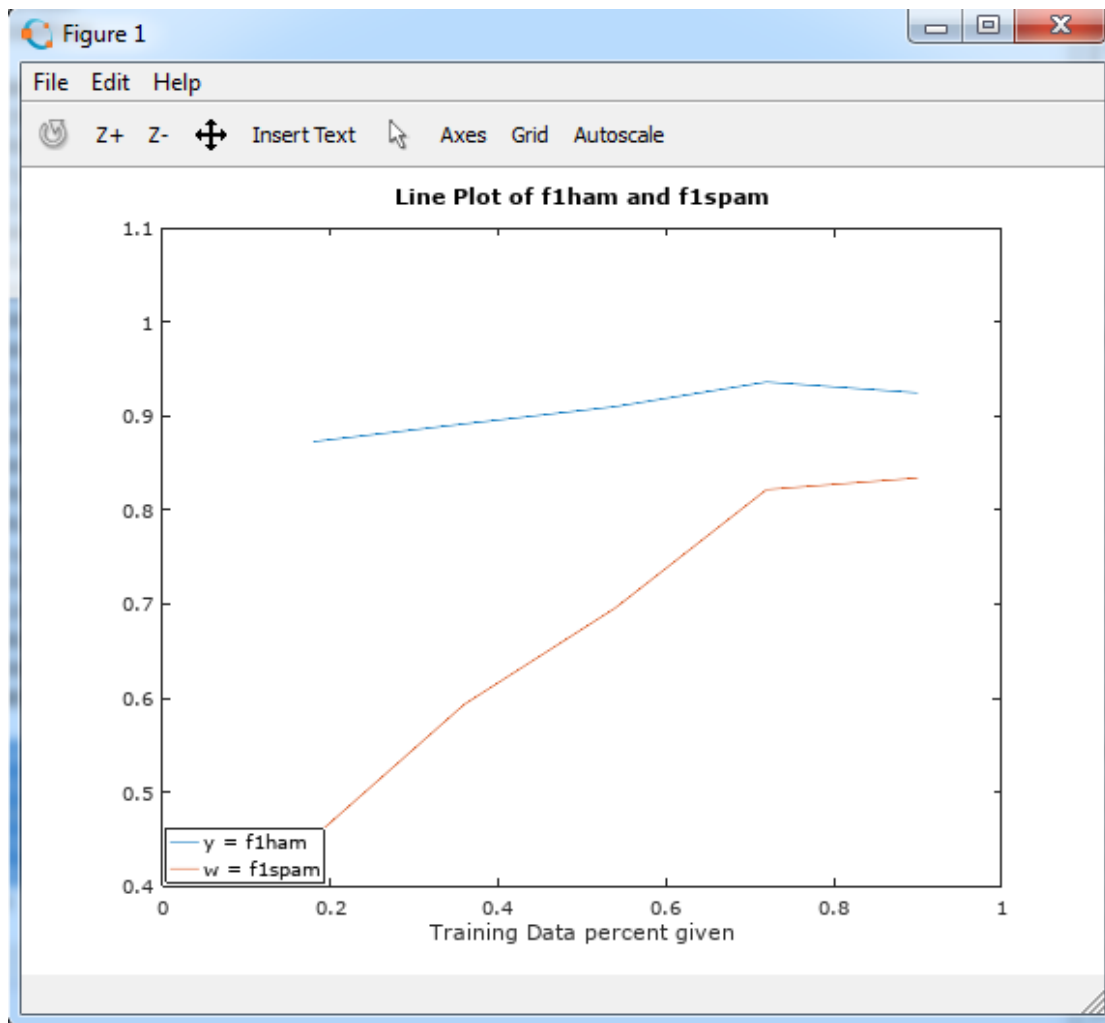
2) Precisionham - Precisionspam



3) Recallham-Recallspam



4)F1ham-F1spam



Ενδεικτικά και ο κώδικας octave για την δημιουργία των καμπυλών, αφού πάρθηκαν τα αποτελέσματα από την κλάση test.

