
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ


# Ανάλυση και κατηγοριοποίηση χρηστών Twitter

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της


**ΔΑΝΑΗΣ ΠΛΑ ΚΑΡΥΔΗ**



**Επιβλέπων :** Ιωάννης Βασιλείου
Καθηγητής Ε.Μ.Π.



Αθήνα, Απρίλιος 2014

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

# Ανάλυση και κατηγοριοποίηση χρηστών Twitter

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

**ΔΑΝΑΗΣ ΠΛΑ ΚΑΡΥΔΗ**

**Επιβλέπων :** Ιωάννης Βασιλείου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11η Απριλίου 2014.

| *(Υπογραφή)* | *(Υπογραφή)* | *(Υπογραφή)* |
|---|---|---|
| ..... | ..... | ..... |
| Ιωάννης Βασιλείου<br>Καθηγητής Ε.Μ.Π. | Ιωάννης Σταύρακας<br>Ερευνητής ΙΠΣΥ | Κώστας Κοντογιάννης<br>Αναπληρωτής Καθηγητής<br>Ε.Μ.Π. |

Αθήνα, Απρίλιος 2014

*(Υπογραφή)*

.....

**ΔΑΝΑΗ ΠΛΑ ΚΑΡΥΔΗ**

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

# Περίληψη

Σε αυτή την εργασία θα προσπαθήσουμε να αναλύσουμε τα χαρακτηριστικά των spam bots και στη συνέχεια χρησιμοποιώντας κάποιον αλγόριθμο μηχανικής μάθησης να κατασκευάσουμε ένα μοντέλο που θα κατηγοριοποιεί λογαριασμούς του Twitter σε spammer και μη spammer. Η κατηγοριοποίηση αυτή βασίζεται σε χαρακτηριστικά που εξάγονται από το προφίλ των χρηστών και το περιεχόμενο των tweets που δημοσιεύουν. Στα πλαίσια της εκπαίδευσης του μοντέλου κατηγοριοποίησης, δοκιμάζονται αρκετοί αλγόριθμοι και επιλέγεται ο καταλληλότερος. Επιπλέον διαμορφώνεται μία blacklist με spammer λογαριασμούς. Επιπλέον σχεδιάστηκε και υλοποιήθηκε web εφαρμογή που: κατηγοριοποιεί λογαριασμούς Twitter, ανιχνεύει spammers σε friends και followers του χρήστη και τέλος ο χρήστης μπορεί να χαρακτηρίζει λογαριασμούς, ώστε τα δεδομένα αυτά να χρησιμοποιούνται για την περεταίρω εκπαίδευση του μοντέλου και για τον καθορισμό ενός συνόλου λογαριασμών που είναι spammer για να εμπλουτιστεί η blacklist.



**Λέξεις Κλειδιά:** <<Twitter, κακόβουλο, κατηγοριοποίηση, ανίχνευση, ανάλυση, ρομπότ, μαύρη λίστα, μηχανική μάθηση>>

## Abstract

The scope of this thesis was the development of a model able to detect spammer Twitter accounts. In order to accomplish this we analyze the characteristics of spam bots that will help us build a classification machine learning model. These characteristics are extracted from account's profile and user's tweet content. In order to choose the optimal machine learning algorithm we test a variety of them. Furthermore we form a blacklist containing spammer accounts. Finally, we developed a web application that gives the user the opportunity to categorize Twitter accounts, detect spammers in his friends or followers and mark accounts as spammers or not spammers. This way both the classifier and the blacklist are updated with fresh data.



**Keywords:** <<Twitter, spammer, bots, classification, detection, machine learning, blacklist>>

# Πίνακας περιεχομένων

# 1

# *Εισαγωγή*

## *1.1 Spam Bots στο Twitter*

Το Twitter είναι ένα μέσο κοινωνικής δικτύωσης που επιτρέπει στους χρήστες του να στέλνουν και να διαβάζουν σύντομα μηνύματα (μέχρι 140 χαρακτήρες), τα οποία ονομάζονται tweets. Τα hashtags, δηλαδή λέξεις ή φράσεις που ξεκινούν με το σύμβολο #, χρησιμοποιούνται για την ομαδοποίηση τέτοιων tweets με βάση το θέμα τους. Το σύμβολο @ ακολουθούμενο από ένα όνομα χρήστη σε κάποιο tweet στέλνει το συγκεκριμένο tweet κατευθείαν σε αυτόν το χρήστη. Σε αντίθεση με τα υπόλοιπα social media οι σχέσεις μεταξύ των χρηστών του Twitter είναι 2 , friend ή follower. Εάν ένας χρήστης προσθέσει έναν άλλον ως friend τότε γίνεται follower του, ενώ ο δεύτερος θα παραμείνει friend του πρώτου. Τέλος όταν ένας χρήστης δημοσιεύει ένα tweet, τότε αυτό αυτόματα εμφανίζεται στην αρχική σελίδα τόσο του ίδιου όσο και στις σελίδες των followers του.

Το Twitter σήμερα έχει φθάσει να έχει 200 εκατομμύρια ενεργούς χρήστες και να βρίσκεται μέσα στις 10 πρώτες πιο δημοφιλείς ιστοσελίδες. Φυσικό λοιπόν είναι να προσελκύσει πληθώρα από spammers οι οποίοι δημιουργούν προβλήματα στους χρήστες του. Οι spammers αυτοί έχουν διάφορους σκοπούς : είτε να κλέψουν λογαριασμούς, είτε να

βομβαρδίσουν με άσχετες διαφημίσεις τους χρήστες, είτε να τους ωθήσουν να χρησιμοποιήσουν κακόβουλο λογισμικό και ιστοσελίδες. Στην προσπάθειά τους οι spammers αξιοποιούν λογισμικό, δηλαδή αυτοματοποιημένα προγράμματα που λέγονται bots. Τα spam bots δημιουργούν λογαριασμούς, δημοσιεύουν tweets συνήθως με links σε κακόβουλες ιστοσελίδες, προσθέτουν φίλους, «μαντεύουν» αδύναμους κωδικούς με σκοπό να μπορούν να κλέβουν λογαριασμούς, χρησιμοποιώντας το Twitter API. Ταυτόχρονα και ιδιαίτερα τα τελευταία χρόνια έχουν αναπτύξει «έξυπνες» μεθόδους ώστε να καταφέρνουν να περνούν απαρατήρητα και να μην ανιχνεύονται από τα εργαλεία και τους μηχανισμούς του Twitter.

Το Twitter σήμερα υστερεί σχετικά στην ανάπτυξη επαρκούς τεχνολογίας για την έγκυρη και έγκαιρη ανίχνευσή τους. Ο μηχανισμός που χρησιμοποιεί μπλοκάρει τους λογαριασμούς που δημοσιεύουν links που χαρακτηρίζονται κακόβουλα με βάση το Google's Safebrowsing API. Επιπλέον χρησιμοποιεί την τεχνική του «mark as spam», δηλαδή τη δυνατότητα που έχουν οι χρήστες να αναφέρουν κάποιον άλλο χρήστη ως spammer.

Υπάρχει όμως ένα πρόβλημα: Είναι όλα τα bots ταυτόχρονα και spammers? Η απάντηση είναι όχι, αφού η χρήση τους δεν γίνεται μόνο από spammers. Υπάρχουν πολλές επιχειρήσεις πχ στο χώρο της ενημέρωσης που χρησιμοποιούν bots για τη συνεχή ροή ειδήσεων στο λογαριασμό τους με σκοπό την ενημέρωση των followers τους. Αυτό εξυπηρετεί το στόχο του Twitter να γίνει ενημερωτικό δίκτυο, ενώ από την άλλη δυσχεραίνει την απομόνωση των spam bots αφού πρέπει να αναπτυχθεί τρόπος που να ξεχωρίζει τα «κακά» από τα «καλά» bots.

## *1.2 Αντικείμενο διπλωματικής*

Σε αυτή την εργασία θα προσπαθήσουμε να αναλύσουμε τα χαρακτηριστικά των spam bots και στη συνέχεια χρησιμοποιώντας κάποιον αλγόριθμο μηχανικής εκμάθησης να κατασκευάσουμε ένα μοντέλο που θα κατηγοριοποιεί λογαριασμούς σε spam και human. Η κατηγοριοποίηση θα βασίζεται σε χαρακτηριστικά που εξάγονται από το προφίλ των χρηστών και το περιεχόμενο των tweets που δημοσιεύουν. Στα πλαίσια της εκπαίδευσης

μοντέλου κατηγοριοποίησης, δοκιμάζονται αρκετοί αλγόριθμοι και επιλέγεται ο καταλληλότερος.

### *1.2.1 Συνεισφορά*

Στη διπλωματική εργασία σχεδιάστηκε και υλοποιήθηκε σύστημα που:

- Χρησιμοποιεί το API του Twitter για την πρόσβαση στη ροή των tweets και στα χαρακτηριστικά των χρηστών
- Αναλύει το περιεχόμενο των tweets και τα χαρακτηριστικά των χρηστών
- Εκπαιδεύει ένα μοντέλο κατηγοριοποίησης των χρηστών, με βάση κάποιον αλγόριθμο μηχανικής μάθησης
- Κατηγοριοποιεί λογαριασμούς χρηστών στο twitter

Επιπλέον για να είναι η πληροφορία αυτή προσβάσιμη αναπτύσσεται web εφαρμογή που επιτρέπει:

- Ο χρήστης να χαρακτηρίζει λογαριασμούς, ώστε τα δεδομένα αυτά να χρησιμοποιούνται για την περεταίρω εκπαίδευση του μοντέλου.
- Να καθοριστεί ένα σύνολο λογαριασμών που είναι spam για να διαμορφωθεί μία black list.

# 2

## *Σχετικές εργασίες*

Γενικά και στο βαθμό που το Twitter αναπτυσσόταν και μέχρι να φθάσει στο σημερινό επίπεδο πολλές μελέτες έχουν γίνει για την ταχεία του επέκταση, αλλά και τη χρήση του καθαυτή [1,2,3]. Πολύ ενδιαφέρουσα είναι η μελέτη που έγινε πάνω σε 100000 χρήστες με σκοπό την κατηγοριοποίησή τους [5]. Το μόνο κριτήριο που χρησιμοποιήθηκε ήταν η αναλογία friends και followers και το αποτέλεσμα ήταν να χωριστούν σε τρεις κατηγορίες: broadcasters, acquaintances και miscreants που είναι και οι πιο πιθανοί spammers. Σε επόμενη μελέτη [6] διαπιστώθηκε ότι οι spammers δημοσιεύουν περισσότερα tweets και συχνά αναπτύσσουν σχέσεις μεταξύ τους. Επίσης η μεγάλη αναλογία followers προς following είναι ένδειξη ότι ο χρήστης ενδέχεται να είναι spammer. Αργότερα [7] χρησιμοποιούνται και άλλες μετρήσεις όπως η εντροπία των διαστημάτων μεταξύ tweets, που δεν περιλαμβάνονται σε προηγούμενες έρευνες. Οι προσπάθειες κατηγοριοποίησης ξεκίνησαν από τους spammers στις υπηρεσίες email [8] οι οποίες εισήγαγαν τη χρήση του Bayes κατηγοριοποιητή. Τέλος έγινε μία μελέτη που προσεγγίζει το σκοπό της εργασίας [9]. Σε αυτήν γίνεται κατηγοριοποίηση με βάση χαρακτηριστικά των χρηστών και του περιεχομένου των tweets τους. Η μελέτη αυτή βασίζεται σε 500 χιλιάδες tweets και η κατηγοριοποίηση για την εκπαίδευση έγινε manually. Σε αντίθεση με αυτήν την προσέγγιση εμείς χρησιμοποιήσαμε περίπου 10 εκατομμύρια tweets (7 εκατομμύρια διεθνών και 3

εκατομμύρια ελληνικών) και η κατηγοριοποίηση για την εκπαίδευση έγινε με τη χρήση των ευρέως χρησιμοποιούμενων και αναγνωρισμένων url blacklists.

# 3

# *Θεωρητικό υπόβαθρο*

## *3.1 Χαρακτηριστικά spam bots λογαριασμών*

Για την εκπαίδευση του μοντέλου θα χρειαστεί απαραίτητα ένα σύνολο δεδομένων με γνωστή την ιδιότητά τους ως spammers ή όχι. Τα spam bots όσο κι αν εξελίσσονται με την πάροδο του χρόνου και όσο κι αν έχουν βελτιωθεί στον τρόπο που κρύβονται, δεν παύουν να αποτελούν προγράμματα που προσπαθούν, αλλά είναι αδύνατο να καταφέρουν η συμπεριφορά τους στα social media να προσομοιάσει με αυτή του ανθρώπου. Ζητούμενο είναι λοιπόν να κατανοήσουμε ποια χαρακτηριστικά τους θα κρατήσουμε για την εκπαίδευση του μοντέλου. Αυτά τα χαρακτηριστικά θα πρέπει σε γενικές γραμμές, περισσότερο ή λιγότερο να διαφοροποιούνται μεταξύ των spammers και των υπολοίπων. Τα χαρακτηριστικά λοιπόν που εξήχθησαν για την εκπαίδευση και την κατηγοριοποίηση αποτελούνται αφενός από χαρακτηριστικά που έχουν να κάνουν με το προφίλ και τους φίλους του χρήστη, τη συμπεριφορά του στο χρόνο και αφετέρου από χαρακτηριστικά που έχουν να κάνουν με το περιεχόμενο των tweets που δημοσιεύει.

### *3.1.1 Αναλογία friends και followers*

Τα tweets ενός χρήστη γίνονται ορατά μόνο από όσους είναι followers του. Τα spam bots γίνονται followers σε πολλούς χρήστες με την προοπτική οι χρήστες αυτοί να γίνουν followers του spammer. Η αναλογία υπολογίζεται ως ο λόγος των followers προς το άθροισμα friends και followers. Εάν ο αριθμός των followers ενός χρήστη είναι σχετικά μικρός σε σχέση με τους friends του, ο λόγος είναι μικρός. Αυτοί οι χρήστες είναι θεωρητικά πιθανότερο να είναι spam. Ωστόσο τα σύγχρονα spam bots έχουν ξεπεράσει αυτό το εμπόδιο και έτσι εάν σε ορισμένο χρονικό διάστημα που έγιναν followers ενός χρήστη αυτός δεν έγινε δικός τους, τότε σταματούν να τον ακολουθούν και ο λόγος αλλοιώνεται. Αυτό έγινε φανερό και στα αποτελέσματα αυτής της εργασίας, όπου τελικά ο αλγόριθμος κατηγοριοποίησης δεν έδωσε τόσο βάρος στο λόγο αυτόν κατά την εκπαίδευση.

### *3.1.2 Αριθμός url ανά tweet*

Οι spammers συνήθως έχουν ως στόχο να οδηγήσουν τους χρήστες να ανοίξουν κάποια σελίδα κακόβουλη, διαφημιστική, κλπ. Έτσι δημοσιεύουν συνήθως τα αντίστοιχα links και μάλιστα σε συνεπτυγμένη μορφή, λόγω του περιορισμού των 140 χαρακτήρων ανά tweet. Έτσι φαίνεται ότι όσο μεγαλύτερος είναι αυτός ο μέσος όρος url ανά tweet τόσο μεγαλύτερες είναι οι πιθανότητες ο λογαριασμός να είναι spam.

### *3.1.3 Απαντήσεις (@ replies) ανά tweet*

Οι spammers συχνά θέλουν να έχουν πρόσβαση σε λογαριασμούς με τους οποίους δεν είναι friends ή followers. Ένας τρόπος να το πετύχουν αυτό είναι να στείλουν μία απάντηση ή αλλιώς μία αναφορά (@reply/mention). Με άλλα λόγια μετά το @ τοποθετούν ένα όνομα χρήστη και οποιοδήποτε μήνυμα και αυτό εμφανίζεται πλέον στο χρήστη. Οι spammers συνήθως χρησιμοποιούν πληθώρα τέτοιων αναφορών με σκοπό τα μηνύματά τους να πάνε σε όσο το δυνατό περισσότερους χρήστες. Έτσι γίνεται αντιληπτό ότι ένας λογαριασμός με πολύ μεγάλο αριθμό αναφορών ανά μήνυμα είναι πολύ πιθανό να είναι spam.

### *3.1.4 Ομοιότητα μεταξύ tweets*

Μία πολύ ενδιαφέρουσα παράμετρος που πρέπει να λάβουμε υπόψιν είναι το κατά πόσο τα tweets που δημοσιεύει ένας χρήστης μοιάζουν μεταξύ τους. Εάν το κείμενο δηλαδή αλλάζει από tweet σε tweet και αν ναι κατά πόσο. Είναι πιθανό τα spam bots προκειμένου να αποφύγουν τον εντοπισμό τους, να καταφεύγουν σε μικρές αλλαγές στα κείμενα τα οποία δημοσιεύουν. Όμως είναι γεγονός πως αδιαμφησβήτητη παράμετρος που ξεχωρίζει ένα spam bot από έναν απλό χρήστη είναι η ομοιότητα των tweets.

#### *3.1.4.1 Απόσταση Levenshtein*

Η απόσταση Levenshtein μεταξύ δύο συμβολοσειρών δίνεται από τον ελάχιστο αριθμό διαδικασιών που απαιτούνται για να μετασχηματιστεί μια σειρά σε κάποια άλλη. Με τον όρο διαδικασία εννοούμε μια εισαγωγή, διαγραφή, ή αντικατάσταση χαρακτήρα. Πήρε το όνομά της από τον Vladimir Levenshtein, ο οποίος εξέτασε αυτήν την απόσταση το 1965.

Παραδείγματος χάριν, η απόσταση Levenshtein μεταξύ της λέξης "kitten" και της "sitting" είναι 3, αφού με τις παρακάτω τρεις αλλαγές μετασηματίζεται η μία στην άλλη χωρίς αυτό να είναι δυνατό με λιγότερες αλλαγές:

1. kitten → sitten (αντικατάσταση του "s" με "k")
2. sitten → sittin (αντικατάσταση του "i" με "e")
3. sittin → sitting (εισαγωγή του "g" στο τέλος).

Ο ψευδοκώδικας που αντιστοιχεί στην υλοποίησης του αλγορίθμου φαίνεται παρακάτω και αφορά την συνάρτηση LevenshteinDistance που παίρνει δύο συμβολοσειρές την s μήκους m και την t μήκους n, και επιστρέφει την Levenshtein απόσταση μεταξύ τους:

```
int LevenshteinDistance(char s[1..m], char t[1..n])
{
/* for all i and j, d[i,j] will hold the Levenshtein distance between the first i characters of s and the first j characters of t; note that d has (m+1)*(n+1) values
```

```
*/
  declare int d[0..m, 0..n]

  clear all elements in d // set each element to zero
   // source prefixes can be transformed into empty string by
  // dropping all characters
  for i from 1 to m{
      d[i, 0] := i
  }
   // target prefixes can be reached from empty source prefix
  // by inserting every characters
  for j from 1 to n   {
      d[0, j] := j
    }
   for j from 1 to n    {
      for i from 1 to m        {
          if s[i] = t[j] then
            d[i, j] := d[i-1, j-1]       // no operation required
          else
            d[i, j] := minimum
                    (
                      d[i-1, j] + 1,  // a deletion
                      d[i, j-1] + 1,  // an insertion
                      d[i-1, j-1] + 1 // a substitution
                    )
        }
     }
    return d[m, n]
}
```

### *3.1.5 Χρονικό διάστημα μεταξύ των δημοσιεύσεων*

Τα spam bots όπως καθετί αυτοματοποιημένο παρουσιάζουν μια περιοδικότητα ως προς τη συμπεριφορά τους. Με άλλα λόγια μπορεί να δημοσιεύουν tweets σε σταθερά χρονικά διαστήματα. Για έναν άνθρωπο αυτό είναι σχεδόν αδύνατο. Άρα είναι σαφές ότι λογαριασμοί που στέλνουν με σταθερή συχνότητα tweets έχουν αυξημένες πιθανότητες να είναι spam.

### *3.1.6 Το μέσο από το οποίο έγινε η δημοσίευση*

Ένας χρήστης του Twitter μπορεί να χρησιμοποιεί την πλατφόρμα από πληθώρα συσκευών, προγραμμάτων, applications, κλπ. Μερικά παραδείγματα είναι web, Tweet Button, Posterous, foursquare, tnd, API, HootSuite, YoruFukurou, Twitter for iPhone, twitterfeed, Twitter for Android, TweetDeck, Twitpic, Keitai Web, Mobile Web, txt, Pinterest, Google, Twitter for Windows Phone, LinkedIn, Facebook, Amazon, Instagram, Snaptu, Twitter for iPad, Mobile Web (M2), Twitter for Nokia S40, TwitBird, Samsung Mobile, Tweetbot for Mac, bitly, Twittascope, κλπ. Συνήθως οι απλοί χρήστες χρησιμοποιούν το Twitter από το web ή από κάποιο κινητό smartphone ή application. Από την άλλη μεριά τα spam bots, ακριβώς επειδή πρόκειται για προγράμματα χρησιμοποιούν συνήθως το API του Twitter.

## *3.2 Αλγόριθμοι κατηγοριοποίησης για μηχανική μάθηση*

### *3.2.1 Μηχανική μάθηση*

Η μηχανική μάθηση έχει ως σκοπό τη δημιουργία μηχανών ικανών να μαθαίνουν, δηλαδή ικανών να βελτιώνουν την απόδοση τους σε κάποιους τομείς μέσω της αξιοποίησης προηγούμενης γνώσης και εμπειρίας και με βάση αυτήν την εμπειρία να μπορούν να παίρνουν όσο το δυνατό σωστότερες αποφάσεις.

Ένας αρκετά γενικός ορισμός που θα μπορούσε να δοθεί για τη μηχανική μάθηση δίνεται στο Mitchell 1997 [10]: «Ένα πρόγραμμα υπολογιστή λέμε ότι μαθαίνει από την εμπειρία Ε ως

προς κάποια κλάση εργασιών T και μέτρο απόδοσης P, αν η απόδοση του σε εργασίες από το T, όπως μετριέται από το P, βελτιώνεται μέσω της εμπειρίας E.»

Για παράδειγμα, το πρόβλημα της κατηγοριοποίησης χρηστών Twitter θα μπορούσε να προσδιοριστεί σύμφωνα με τον παραπάνω ορισμό ως εξής:

- Έργο T: Η κατάταξη λογαριασμών στις δύο κατηγορίες, spammer και not spammer.
- Μέτρο απόδοσης P: Το ποσοστό των χρηστών που ταξινομήθηκαν σωστά
- Εμπειρία E: Ένα σύνολο από χρήστες με γνωστή κατηγοριοποίηση.

Αρχικά πριν κιόλας από την εκπαίδευση του μοντέλου θα πρέπει να έχουμε επιλέξει τα χαρακτηριστικά που θα χρησιμοποιήσουμε. Στην περίπτωση αυτής της εργασίας τα χαρακτηριστικά αυτά είναι αυτά που αναφέρθηκαν προηγουμένως (παράγραφος 3.1). Για κάθε αντικείμενο προς κατηγοριοποίηση εξάγεται ένα διάνυσμα με τα χαρακτηριστικά του. Ανάλογα με το είδος της πληροφορίας που αντιπροσωπεύει ένα χαρακτηριστικό, αυτό μπορεί να είναι συνεχές (continuous) – π.χ. ένας πραγματικός ή ακέραιος αριθμός, ή ονομαστικό (nominal), το οποίο λαμβάνει ένα προκαθορισμένο σύνολο διακριτών τιμών, αριθμητικών ή συμβολικών. Με αυτόν τον τρόπο, έχοντας επιλέξει ν χαρακτηριστικά, απεικονίζουμε το χώρο του προβλήματος μας σε έναν ν- διάστατο χώρο, το χώρο των στιγμιοτύπων (instance space), αντιστοιχώντας κάθε στιγμιότυπο εκπαίδευσης σε ένα διάνυσμα ν διαστάσεων.

Στην επιβλεπόμενη μηχανική μάθηση κατά την εκπαίδευση χρησιμοποιείται ένας αριθμός στιγμιότυπων με γνωστό από πριν το αποτέλεσμα της απόφασης. Έτσι κατά την εκπαίδευση, ο αλγόριθμος αναζητά μια συνάρτηση η οποία προσεγγίζει όσο το δυνατόν περισσότερο μια ιδανική συνάρτηση με την οποία δύναται να μοντελοποιηθεί το πρόβλημα, τη συνάρτηση στόχο, δηλαδή μία συνάρτηση που να επαληθεύει το ήδη γνωστό αποτέλεσμα. Οι δύο συναρτήσεις έχουν ελεύθερη μεταβλητή ένα τυχαίο διάνυσμα x , πεδίο ορισμού το χώρο των στιγμιοτύπων και σύνολο τιμών το οποίο καθορίζεται από την εκάστοτε εφαρμογή. Έτσι, η επίλυση ενός προβλήματος μηχανικής μάθησης ανάγεται στην επίλυση ενός προβλήματος προσέγγισης των τιμών μιας συνάρτησης.

Γενικά ο στόχος της επιβλεπόμενης μηχανικής μάθησης είναι να κατασκευαστεί ένα μοντέλο που να αναπαριστά τη γνώση που παρέχεται μέσω της εμπειρίας Ε και το οποίο στη συνέχεια πρόκειται να χρησιμοποιηθεί για την αξιολόγηση νέων στιγμιότυπων.

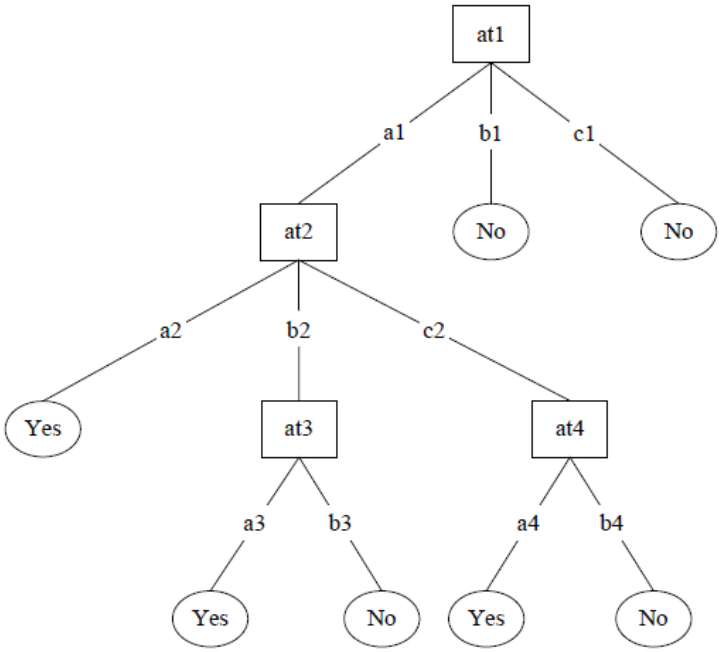
Υπάρχουν πολλά είδη επιβλεπόμενης μάθησης. Συγκεκριμένα χρησιμοποιούνται ανάλογα με την περίπτωση αλγόριθμοι κατηγοριοποίησης, ομαδοποίησης, δημιουργίας προτάσεων, κλπ.

### *3.2.2 Αλγόριθμοι κατηγοριοποίησης*

Για το πρόβλημα της συγκεκριμένης εργασίας θα χρειαστούμε ένα μοντέλο κατηγοριοποίησης. Συγκεκριμένα το μοντέλο θα εκπαιδευτεί κατάλληλα ώστε να μπορεί να κατηγοριοποιεί χρήστες του Twitter σε spammers και όχι. Επομένως πρέπει να γνωρίσουμε ορισμένους αλγόριθμους κατηγοριοποίησης και σε επόμενη ενότητα να τους χρησιμοποιήσουμε δοκιμαστικά, ώστε να επιλέξουμε τον καταλληλότερο για την υλοποίηση της εργασίας. Παρακάτω παρουσιάζονται συνοπτικά οι βασικότεροι αλγόριθμοι κατηγοριοποίησης.

#### *3.2.2.1 Δέντρα αποφάσεων*

Τα δέντρα αποφάσεων είναι δέντρα που κατηγοριοποιούν στιγμιότυπα ταξινομώντας τα με βάση τις τιμές των χαρακτηριστικών τους. Κάθε κόμβος σε ένα τέτοιο δέντρο αναπαριστά ένα χαρακτηριστικό σε ένα στιγμιότυπο προς κατηγοριοποίηση και κάθε κλάδος αναπαριστά μία τιμή που ο κόμβος (το χαρακτηριστικό του στιγμιότυπου) έχει. Τα στιγμιότυπα κατηγοριοποιούνται ξεκινώντας από τη ρίζα του δέντρου και ταξινομούνται με βάση τις τιμές των χαρακτηριστικών τους. Στο σχήμα 3.2.2.1.α φαίνεται ένα παράδειγμα ενός τέτοιου δέντρου για το σύνολο στιγμιότυπων εκπαίδευσης του πίνακα 3.2.2.1.α.




Σχήμα 3.2.2.1.α

| at1 | at2 | at3 | at4 | Class |
|---|---|---|---|---|
| a1 | a2 | a3 | a4 | Yes |
| a1 | a2 | a3 | b4 | Yes |
| a1 | b2 | a3 | a4 | Yes |
| a1 | b2 | b3 | b4 | No |
| a1 | c2 | a3 | a4 | Yes |
| a1 | c2 | a3 | b4 | No |
| b1 | b2 | b3 | b4 | No |
| c1 | b2 | b3 | b4 | No |

Πίνακας 3.2.2.1.α

Η κατασκευή ενός βέλτιστου δυαδικού δέντρου απόφασης είναι NP-πλήρες πρόβλημα και για αυτό οι θεωρητικοί ψάχνουν μια υλοποίηση που θα προσεγγίζει το βέλτιστο δέντρο. Το χαρακτηριστικό που θα χωρίζει καλύτερα τα δεδομένα εκπαίδευσης θα είναι ο κόμβος ρίζα του δέντρου.

Στο σχήμα 3.2.2.1.β φαίνεται ένας ψευδοκώδικας για την κατασκευή δέντρων αποφάσεων.

```
Check for base cases
 For each attribute a
    Find the feature that best
    divides the training data such
    as information gain from
    splitting on a
Let a best be the attribute with the
highest normalized information gain
   Create a decision node node that
   splits on a_best
Recurse on the sub-lists obtained by
splitting on a best and add those
nodes as children of node
```

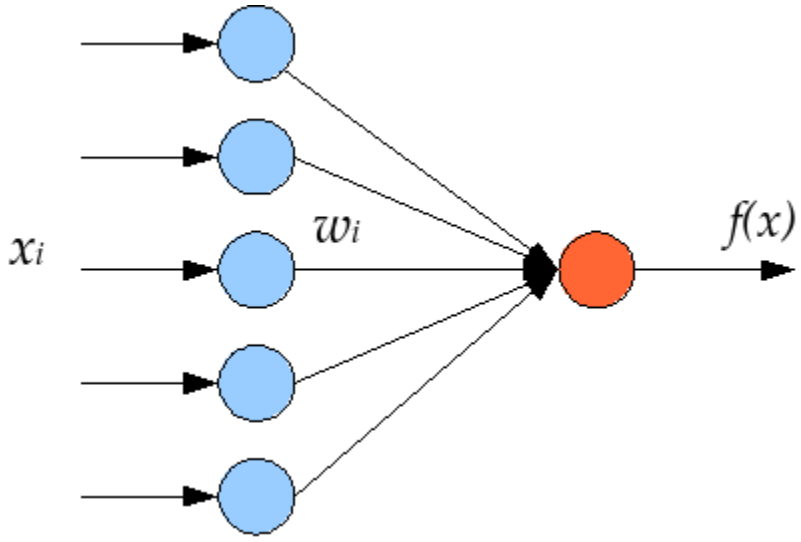
Σχήμα 3.2.2.1.β

### *3.2.2.2 Αλγόριθμοι βασισμένοι σε perceptron*

Τα perceptron είναι βασισμένα στον τρόπο λειτουργίας του εγκεφάλου, προσομοιώνουν δηλαδή τη δομή και λειτουργία των νευρώνων του.

#### *3.2.2.2.1 Perceptron ενός στρώματος*

Ένα perceptron ενός στρώματος (Σχήμα 3.2.2.2.1.α) περιγράφεται συνοπτικά ως εξής. Έστω το διάνυσμα τιμών χαρακτηριστικών εισόδου $x_1 \dots x_n$ και $w_1 \dots w_n$ είναι το διάνυσμα βαρών των συνδέσεων των χαρακτηριστικών (νευρώνων) εισόδου, τότε το perceptron υπολογίζει το άθροισμα $\sum_i x_i \cdot w_i$ και η έξοδος αυτή περνά μέσα από ένα ρυθμιζόμενο κατώφλι: εάν το άθροισμα είναι πάνω από το κατώφλι η έξοδος (απόφαση) είναι 1, αλλιώς 0. Συνήθως ο αλγόριθμος αυτός χρησιμοποιείται για εκμάθηση από ένα σύνολο εκπαίδευσης με συνεχείς επαναλήψεις του πάνω στο σύνολο αυτό μέχρι να βρει ένα διάνυσμα πρόβλεψης το οποίο επαληθεύεται για όλο το σύνολο εκπαίδευσης.

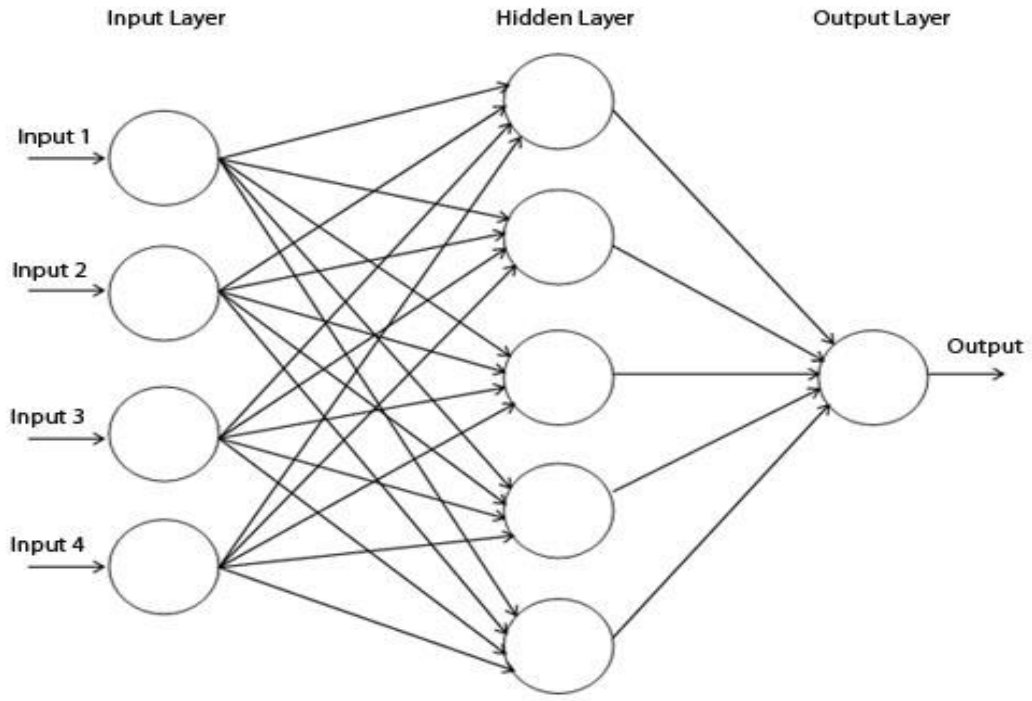



Σχήμα 3.2.2.2.1.α

### *3.2.2.2.2 Perceptron πολλαπλών στρωμάτων*

Τα perceptron μπορούν να κατηγοριοποιήσουν μόνο γραμμικά διαχωρίσιμα σύνολα στιγμιότυπων. Αν μία ευθεία γραμμή μπορεί να τα διαχωρίσει στις σωστές κατηγορίες, τότε λέμε ότι είναι γραμμικά διαχωρίσιμες και το perceptron θα βρει τη σωστή λύση. Εάν όμως δεν είναι, τότε το perceptron δε θα φθάσει ποτέ στο σημείο να κατηγοριοποιήσει όλα τα instances σωστά. Τη λύση στο πρόβλημα αυτό έρχονται να λύσουν τα perceptron πολλαπλών επιπέδων ή αλλιώς νευρωνικά δίκτυα.

Τα νευρωνικά δίκτυα αποτελούνται από μεγάλο αριθμό μονάδων (νευρώνες) που ενώνονται με συνδέσεις (Σχήμα 3.2.2.2.2.α). Οι ομάδες νευρώνων συγκροτούν ορισμένα στρώματα: εισόδου, τα οποία δέχονται τα δεδομένα εκπαίδευσης, εξόδου, τα οποία έχουν τα δεδομένα εξόδου και ενδιάμεσα στρώματα. Τα feed forward νευρωνικά επιτρέπουν τα μηνύματα από την είσοδο προς την έξοδο μόνο.




Σχήμα 3.2.2.2.2.α

Αρχικά το δίκτυο εκπαιδεύεται σε ένα σύνολο από ζεύγη στιγμιότυπων (εισόδου – εξόδου). Τα βάρη των συνδέσεων μεταξύ των νευρώνων είναι προσδιορισμένα και το νευρωνικό μπορεί να χρησιμοποιηθεί για την κατηγοριοποίηση άλλων στιγμιότυπων. Κατά την κατηγοριοποίηση το σήμα στην είσοδο διαπερνά όλο το δίκτυο για να καθοριστούν οι τιμές ενεργοποίησης σε όλους τους νευρώνες εξόδου. Κάθε νευρώνας εισόδου έχει μία τιμή ενεργοποίησης που αντιπροσωπεύει κάποιο χαρακτηριστικό. Στη συνέχεια, κάθε νευρώνας στέλνει την τιμή ενεργοποίησής του σε κάθε ένα νευρώνα του ενδιάμεσου στρώματος, ο οποίος με τη σειρά του υπολογίζει τη δική του τιμή ενεργοποίησης και προωθεί το σήμα στους νευρώνες εξόδου. Όλες οι τιμές ενεργοποίησης υπολογίζονται με βάση μία συνάρτηση ενεργοποίησης. Η συνάρτηση αθροίζει όλες τις συνδέσεις που φτάνουν στο νευρώνα, δηλαδή όλα τα βάρη των συνδέσεων επί την τιμή ενεργοποίησης του νευρώνα από όπου προέρχονται. Αυτό το άθροισμα τροποποιείται περεταίρω ώστε η τιμή αυτή να είναι μεταξύ 0 και 1. Πολύ σημαντικός είναι ο προσδιορισμός του αριθμού των νευρώνων του ενδιάμεσου στρώματος. Εάν αυτός ο αριθμός είναι πολύ μικρός ή πολύ μεγάλος το αποτέλεσμα μπορεί να είναι ένα ανεπαρκές, λόγω υπερκάλυψης ή εκφυλισμένο, λόγω υπερεκπαίδευσης μοντέλο αντίστοιχα και η κατηγοριοποίηση άστοχη. Τα βάρη είναι αρχικοποιημένα σε τυχαίες τιμές και μετά τα στιγμιότυπα περνούν επαναλαμβανόμενα μέσα από το δίκτυο. Οι τιμές εισόδου

ενός στιγμιότυπου τοποθετούνται στους νευρώνες εισόδου και η έξοδος συγκρίνεται κάθε φορά με την επιθυμητή έξοδο για κάθε στιγμιότυπο. Τα βάρη ρυθμίζονται κατάλληλα με ένα συγκεκριμένο ρυθμό με τέτοιο τρόπο ώστε να προσεγγίζουν αυτά που θα είχαν φέρει το επιθυμητό αποτέλεσμα.

#### *3.2.2.2.3 RBF νευρωνικό δίκτυο*

Το RBF νευρωνικό δίκτυο είναι δίκτυο τριών στρωμάτων όπου κάθε ενδιάμεσος νευρώνας εφαρμόζει μία συνάρτηση ενεργοποίησης και κάθε νευρώνας εξόδου ένα σταθμισμένο άθροισμα από τις εξόδους των ενδιάμεσων νευρώνων. Η διαδικασία εκπαίδευσης χωρίζεται συνήθως σε δύο μέρη: Πρώτα οι ενδιάμεσοι νευρώνες καθορίζονται με αλγόριθμους ομαδοποίησης και μετά τα βάρη που συνδέουν τους ενδιάμεσους με τους νευρώνες εξόδου καθορίζονται με αλγόριθμους, συνήθως , ελαχίστων τετραγώνων.

### *3.2.3 Στατιστικοί αλγόριθμοι*

Οι αλγόριθμοι αυτοί χαρακτηρίζονται από τη χρήση μοντέλων πιθανοτήτων που υποδηλώνουν την πιθανότητα ένα στιγμιότυπο να ανήκει στη μία ή την άλλη κατηγορία. Εδώ υπάρχουν πολλοί διαφορετικοί αλγόριθμοι (LDA, maximum entropy, Bayesian Networks). Θα σταθούμε ωστόσο μόνο σε μία κατηγορία μόνο.

#### *3.2.3.1 Αλγόριθμος Naive Bayes*

Τα Naive Bayesian δίκτυα είναι απλά Bayesian δίκτυα που συντίθενται από κατευθυνόμενους ακυκλικούς γράφους με έναν μόνο πρόγονο και πολλούς απογόνους. Υποθέτει ότι έχει ένα σύνολο δεδομένων S και ότι κάθε δείγμα δεδομένων $X=(x_1,x_2,...,x_v)$ με μ κατηγορίες $C_1,C_2,..,C_\mu$. Δεδομένου ενός αγνώστου δείγματος δεδομένων X, ο κατηγοριοποιητής θα προβλέψει ότι το X ανήκει στην κατηγορία C που έχει την μέγιστη εκ των υστέρων πιθανότητα με βάση το X. Ο στόχος, λοιπόν, είναι να βρούμε την μέγιστη εκ των υστέρων πιθανότητα για κάθε κλάση, με αποτέλεσμα ο Naive Bayes κατηγοριοποιητής να έχει υψηλή απόδοση. Η απόδοση του συγκρίνεται με αυτή των δέντρων απόφασης και κάποιους κατηγοριοποιητές που στηρίζονται σε νευρωνικά δίκτυα σε ορισμένες εφαρμογές.

Υπάρχει πληθώρα άλλων αλγόριθμων –SVM’s κλπ. - ωστόσο για τη συγκεκριμένη εργασία αρκεί η μελέτη των παραπάνω.

# 4

# *Ανάλυση Συστήματος*

## *4.1 Συλλογή δεδομένων*

Για την ανάπτυξη και εκπαίδευση του μοντέλου ταξινόμησης των χρηστών του Twitter θα χρειαστούμε ένα σύνολο εκπαίδευσης με τα δεδομένα που αναφέραμε προηγουμένως. Με άλλα λόγια χρειαζόμαστε ένα σύνολο με διανύσματα χαρακτηριστικών, ενώ για κάθε διάνυσμα θα πρέπει να γνωρίζουμε και αν είναι spammer ή όχι. Αυτό θα εξεταστεί σε επόμενη ενότητα.

Για την συγκρότηση του συνόλου εκπαίδευσης χρησιμοποιήθηκαν τρεις μέθοδοι:

- η πρώτη λάμβανε απευθείας tweets που δημοσιεύονταν εκείνη τη στιγμή σε όλο τον κόσμο. Συγκεκριμένα επιλέγονταν κάθε φορά τα tweets που περιείχαν τα πιο δημοφιλή hashtags (αφού αυτά είναι που λογικά θα χτύπαγαν πρώτα οι spammers). Η μέθοδος αυτή χρησιμοποιήθηκε για τη συλλογή 200000 tweets. Από τη μία τα δεδομένα αυτά χρησιμοποιήθηκαν αρκετούς μήνες μετά την συλλογή τους, καθώς οι blacklists που χρησιμοποιήθηκαν για την εύρεση των spammers μέσα σε αυτά δεν ανανεώνονται συνεχώς. Από την άλλη όμως όταν ανανεώθηκαν μας έδωσαν τεράστια

πληροφορία, αφού το Twitter μπλοκάρει άμεσα αυτούς τους λογαριασμούς (σε κάθε ανανέωση των blacklists) και δεν θα είχαμε τη δυνατότητα να ανακτήσουμε τα χαρακτηριστικά τους πλέον.

- η δεύτερη χρησιμοποίησε το SNAP Twitter Dataset του Stanford University. Το dataset αυτό περιείχε 7 εκατομμύρια tweets από όλο τον κόσμο. Ωστόσο αυτά τα tweets είναι του 2009 και ίσως τα χαρακτηριστικά των spammers να έχουν αφενός αλλάξει από τότε, αφ' εταίρου οι τότε spammers να είναι τώρα ανενεργοί και η ανάκτηση έστω και των «παλιών» χαρακτηριστικών τους να είναι πια αδύνατη.

- τέλος η τρίτη χρησιμοποίησε dataset του ΙΠΣΥ με tweets προερχόμενα μόνο από την Ελλάδα και τα οποία μαζεύτηκαν μεταξύ Μάρτη και Ιούλη του 2012. Αυτά τα tweets είναι ιδιαίτερα βοηθητικά στην προσπάθεια να βελτιώσουμε την απόδοση του μοντέλου στην Ελλάδα, αλλά και να διαμορφώσουμε μία λίστα spammers, η οποία θα απευθύνεται και σε ελληνικό κοινό.

## *4.2 Κατηγοριοποίηση δεδομένων*

Οι χρήστες που δημοσίευσαν τα παραπάνω tweets που μαζεύτηκαν, πρέπει να χωριστούν σε spammers και όχι. Άρα πρέπει μέσα από τις πληροφορίες που μας παρέχει το Twitter να συμπεράνουμε ποιοι λογαριασμοί είναι «ύποπτοι». Θα μπορούσαμε να κοιτάξουμε το περιεχόμενο κάθε tweet και να δούμε κατά πόσο προσπαθεί να παρενοχλήσει τους χρήστες. Ωστόσο αυτό θα ήταν τόσο υποκειμενικό όσο και κουραστικό για περίπου 10 εκατομμύρια tweets που έχουμε μαζέψει. Οπότε η ιδέα που εφαρμόστηκε είναι η εξής:

- Σε κάθε tweet αναζητώ, εάν έχει, κάποιο url.
- Λόγω του περιορισμένου αριθμού χαρακτήρων που επιτρέπει το Twitter (140) οι περισσότεροι χρήστες χρησιμοποιούν συντομευμένες (shortened) διευθύνσεις. Άρα το επόμενο βήμα είναι η επέκταση της διεύθυνσης μέχρι αυτή να πάρει την αρχική της μορφή.
- Ψάχνω σε τρεις «μαύρες λίστες» για το domain που έχω ανακτήσει: στη SURBL, στη URIBL, στη Google Safe Browsing.

- Εάν το domain ανήκει σε κάποιες από αυτές, τότε το όνομα του χρήστη που δημοσίευσε το αντίστοιχο tweet αποθηκεύεται σε ένα αρχείο. Αυτό το αρχείο είναι η αρχική μορφή της «μαύρης» λίστας μας.
- Φυσικά επειδή χρειαζόμαστε και δεδομένα που να προέρχονται και από μη spammers αποθηκεύουμε σε άλλο αρχείο ίσα ονόματα σε αριθμό από μη spammers.

Το βήμα αυτό δεν αφορούσε το dataset με τα ελληνικά tweets, γιατί αυτό θα κατηγοριοποιηθεί από το μοντέλο που θα προκύψει από τα δύο άλλα datasets.

## *4.3 Εξαγωγή και επεξεργασία χαρακτηριστικών*

Έχουμε φτάσει σε ένα σημείο όπου έχουμε δύο αρχεία: ένα με ονόματα spammers και ένα με ονόματα μη spammers. Για να φτιάξουμε τα διανύσματα εκπαίδευσης θα πρέπει να συλλέξουμε τα χαρακτηριστικά (παράγραφος 3.1) που μας ενδιαφέρουν για κάθε έναν χρήστη από αυτούς.

Για να αποκτήσουμε αυτά τα χαρακτηριστικά χρησιμοποιήθηκε το Twitter API, το οποίο ωστόσο εισάγει πολλούς περιορισμούς. Πρώτον θα πρέπει οι χρήστες που θα ανακτήσουμε χαρακτηριστικά τους να μην είναι προστατευμένοι. Αυτή είναι μία επιλογή που μπορεί να ενεργοποιήσει ο κάθε χρήστης από τις ρυθμίσεις του λογαριασμού του. Δεύτερον θα πρέπει ο λογαριασμός του χρήστη να υπάρχει τη στιγμή που ζητάμε τα δεδομένα και να μην έχει σβηστεί στο παρελθόν. Τρίτον το Twitter εισάγει περιορισμούς ανάλογα με το request που δέχεται. Οι περιορισμοί αυτοί έχουν να κάνουν με την ποσότητα της παρεχόμενης πληροφορίας. Το Twitter επιτρέπει σε κάθε χρονικό παράθυρο των 15 λεπτών μόνο συγκεκριμένο αριθμό από κάθε τι που ζητείται. Έτσι εγώ μπορούσα να εξετάζω περίπου 120 λογαριασμούς κάθε ένα τέταρτο με βάση τα δεδομένα που είχα συνολικά ανάγκη για κάθε λογαριασμό.

Από το βήμα αυτό λοιπόν προέκυψαν τα δεδομένα εκπαίδευσης που δεν ήταν άλλα από περίπου 3,5 χιλιάδες στιγμιότυπα για την εκπαίδευση. Τα διανύσματα περιείχαν τα εξής πεδία: αναλογία friends και followers, δείκτης ομοιότητας κειμένου των tweets, αριθμός διαδικτυακών διευθύνσεων ανά tweet, αριθμός @ ανά tweet, δείκτης σταθερότητας για το

χρονικό διάστημα μεταξύ δύο συνεχόμενων tweets, πηγή από όπου συνηθίζει να δημοσιεύει ο χρήστης τα tweets, αν ο χρήστης είναι επιβεβαιωμένος χρήστης ή όχι.

## *4.4 Δοκιμή διάφορων αλγόριθμων εκπαίδευσης και αξιολόγηση*

Για την δημιουργία του μοντέλου χρειάστηκε να χρησιμοποιηθούν διάφοροι αλγόριθμοι και να αξιολογηθούν, ώστε να γίνει η επιλογή του πληρέστερου μοντέλου. Για το λόγο αυτό χρησιμοποιήθηκε η εργαλειοθήκη WEKA του University of Waikato. Για αυτήν θα κάνουμε λόγο σε επόμενη ενότητα.

Χρησιμοποιήθηκαν οι παρακάτω αλγόριθμοι μοντέλων κατηγοριοποίησης: AdaBoost, AttributeSelectedClassifier, Bagging, BayesNet, BayesSimple, decisionTable, filteredClassifier, IBK, J48, KStar, LogisticRegression, LWL, multiClass, NaiveBayes, NaiveBayesUpdatable, RandomForest, SGD, simpleLogistic, SMO, ClassificationviaRegression, VotedPerceptron.

### *4.4.1 AdaBoost*

Τα αποτελέσματα είναι τα εξής:

Time taken to build model: 0.25 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2081 | 76.4792 % |
| Incorrectly Classified Instances | 640 | 23.5208 % |
| Kappa statistic | 0.53 | |
| Mean absolute error | 0.3464 | |
| Root mean squared error | 0.4116 | |
| Relative absolute error | 69.2788 % | |
| Root relative squared error | 82.3286 % | |
| Coverage of cases (0.95 level) | 99.6325 % | |
| Mean rel. region size (0.95 level) | 97.0967 % | |

Total Number of Instances 2721

Ignored Class Unknown Instances 6

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.673 | 0.143 | 0.827 | 0.673 | 0.742 | 0.818 | no |
| | 0.857 | 0.327 | 0.722 | 0.857 | 0.784 | 0.819 | yes |
| Weighted Avg. | 0.765 | 0.234 | 0.775 | 0.765 | 0.763 | 0.819 | |

### *4.4.2 AttributeSelectedClassifier*

Time taken to build model: 1.75 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 2062 75.781 %

Incorrectly Classified Instances 659 24.219 %

Kappa statistic 0.5158

Mean absolute error 0.3255

Root mean squared error 0.4222

Relative absolute error 65.1007 %

Root relative squared error 84.4464 %

Coverage of cases (0.95 level) 98.1992 %

Mean rel. region size (0.95 level) 96.4535 %

Total Number of Instances 2721

Ignored Class Unknown Instances 6

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.712 | 0.196 | 0.786 | 0.712 | 0.747 | 0.798 | no |
| | 0.804 | 0.288 | 0.734 | 0.804 | 0.768 | 0.797 | yes |
| Weighted Avg. | 0.758 | 0.242 | 0.76 | 0.758 | 0.757 | 0.797 | |

### *4.4.3 Bagging*

Time taken to build model: 1.75 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2062 | 75.781 % |
| Incorrectly Classified Instances | 659 | 24.219 % |
| Kappa statistic | 0.5158 | |
| Mean absolute error | 0.3255 | |
| Root mean squared error | 0.4222 | |
| Relative absolute error | 65.1007 % | |
| Root relative squared error | 84.4464 % | |
| Coverage of cases (0.95 level) | 98.1992 % | |
| Mean rel. region size (0.95 level) | 96.4535 % | |
| Total Number of Instances | 2721 | |
| Ignored Class Unknown Instances | 6 | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.712 | 0.196 | 0.786 | 0.712 | 0.747 | 0.798 | no |
| | 0.804 | 0.288 | 0.734 | 0.804 | 0.768 | 0.797 | yes |
| Weighted Avg. | 0.758 | 0.242 | 0.76 | 0.758 | 0.757 | 0.797 | |

### *4.4.4 BayesNet*

Time taken to build model: 0.52 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2115 | 77.7288 % |
| Incorrectly Classified Instances | 606 | 22.2712 % |
| Kappa statistic | 0.555 | |
| Mean absolute error | 0.2665 | |
| Root mean squared error | 0.4108 | |

Relative absolute error 53.3094 %

Root relative squared error 82.1626 %

Total Number of Instances 2721

Ignored Class Unknown Instances 6

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.683 | 0.128 | 0.844 | 0.683 | 0.755 | 0.842 | no |
| | 0.872 | 0.317 | 0.732 | 0.872 | 0.796 | 0.841 | yes |
| Weighted Avg. | 0.777 | 0.222 | 0.788 | 0.777 | 0.775 | 0.841 | |

### *4.4.5 BayesSimple*

Time taken to build model: 0.06 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 2045 75.1562 %

Incorrectly Classified Instances 676 24.8438 %

Kappa statistic 0.5037

Mean absolute error 0.297

Root mean squared error 0.4294

Relative absolute error 59.3976 %

Root relative squared error 85.8883 %

Total Number of Instances 2721

Ignored Class Unknown Instances 6

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.636 | 0.132 | 0.829 | 0.636 | 0.72 | 0.82 | no |
| | 0.868 | 0.364 | 0.703 | 0.868 | 0.777 | 0.819 | yes |
| Weighted Avg. | 0.752 | 0.247 | 0.766 | 0.752 | 0.748 | 0.82 | |

### *4.4.6 DecisionTable*

Time taken to build model: 0.61 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2120 | 77.9125 % |
| Incorrectly Classified Instances | 601 | 22.0875 % |
| Kappa statistic | 0.5583 | |
| Mean absolute error | 0.3288 | |
| Root mean squared error | 0.4059 | |
| Relative absolute error | 65.7606 % | |
| Root relative squared error | 81.1809 % | |
| Coverage of cases (0.95 level) | 99.7427 % | |
| Mean rel. region size (0.95 level) | 99.0812 % | |
| Total Number of Instances | 2721 | |
| Ignored Class Unknown Instances | 6 | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.762 | 0.204 | 0.791 | 0.762 | 0.776 | 0.827 | no |
| | 0.796 | 0.238 | 0.768 | 0.796 | 0.782 | 0.827 | yes |
| Weighted Avg. | 0.779 | 0.221 | 0.78 | 0.779 | 0.779 | 0.827 | |

### *4.4.7 FilteredClassifier*

Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2108 | 77.4715 % |
| Incorrectly Classified Instances | 613 | 22.5285 % |
| Kappa statistic | 0.5496 | |
| Mean absolute error | 0.3215 | |
| Root mean squared error | 0.4064 | |

| | |
|---|---|
| Relative absolute error | 64.292 % |
| Root relative squared error | 81.2866 % |
| Coverage of cases (0.95 level) | 99.706 % |
| Mean rel. region size (0.95 level) | 99.1731 % |
| Total Number of Instances | 2721 |
| Ignored Class Unknown Instances | 6 |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.736 | 0.186 | 0.8 | 0.736 | 0.766 | 0.819 | no |
| | 0.814 | 0.264 | 0.753 | 0.814 | 0.782 | 0.82 | yes |
| Weighted Avg. | 0.775 | 0.225 | 0.777 | 0.775 | 0.774 | 0.82 | |

### *4.4.8 IBk*

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2032 | 74.6784 % |
| Incorrectly Classified Instances | 689 | 25.3216 % |
| Kappa statistic | 0.4936 | |
| Mean absolute error | 0.2573 | |
| Root mean squared error | 0.5035 | |
| Relative absolute error | 51.4621 % | |
| Root relative squared error | 100.7052 % | |
| Coverage of cases (0.95 level) | 74.7519 % | |
| Mean rel. region size (0.95 level) | 51.2128 % | |
| Total Number of Instances | 2721 | |
| Ignored Class Unknown Instances | 6 | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.745 | 0.251 | 0.75 | 0.745 | 0.747 | 0.755 | no |
| | 0.749 | 0.255 | 0.744 | 0.749 | 0.746 | 0.755 | yes |

| | | | | | | |
|---|---|---|---|---|---|---|
| Weighted Avg. | 0.747 | 0.253 | 0.747 | 0.747 | 0.747 | 0.755 |

### *4.4.9 J48*

Time taken to build model: 0.09 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2057 | 75.5972 % |
| Incorrectly Classified Instances | 664 | 24.4028 % |
| Kappa statistic | 0.512 | |
| Mean absolute error | 0.3174 | |
| Root mean squared error | 0.4235 | |
| Relative absolute error | 63.4829 % | |
| Root relative squared error | 84.7015 % | |
| Coverage of cases (0.95 level) | 97.6112 % | |
| Mean rel. region size (0.95 level) | 95.2958 % | |
| Total Number of Instances | 2721 | |
| Ignored Class Unknown Instances | 6 | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.735 | 0.223 | 0.769 | 0.735 | 0.752 | 0.802 | no |
| | 0.777 | 0.265 | 0.744 | 0.777 | 0.76 | 0.802 | yes |
| Weighted Avg. | 0.756 | 0.244 | 0.757 | 0.756 | 0.756 | 0.802 | |

### *4.4.10 KStar*

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2121 | 77.9493 % |
| Incorrectly Classified Instances | 600 | 22.0507 % |
| Kappa statistic | 0.559 | |

| | |
|---|---|
| Mean absolute error | 0.2762 |
| Root mean squared error | 0.3979 |
| Relative absolute error | 55.236 % |
| Root relative squared error | 79.5806 % |
| Coverage of cases (0.95 level) | 97.7582 % |
| Mean rel. region size (0.95 level) | 84.9871 % |
| Total Number of Instances | 2721 |
| Ignored Class Unknown Instances | 6 |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.765 | 0.206 | 0.789 | 0.765 | 0.777 | 0.852 | no |
| | 0.794 | 0.235 | 0.77 | 0.794 | 0.782 | 0.853 | yes |
| Weighted Avg. | 0.779 | 0.22 | 0.78 | 0.779 | 0.779 | 0.852 | |

### *4.4.11 Logistic*

Time taken to build model: 5.85 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2073 | 76.1852 % |
| Incorrectly Classified Instances | 648 | 23.8148 % |
| Kappa statistic | 0.5238 | |
| Mean absolute error | 0.3266 | |
| Root mean squared error | 0.412 | |
| Relative absolute error | 65.3197 % | |
| Root relative squared error | 82.4071 % | |
| Total Number of Instances | 2721 | |
| Ignored Class Unknown Instances | 6 | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.733 | 0.209 | 0.78 | 0.733 | 0.756 | 0.825 | no |
| | 0.791 | 0.267 | 0.746 | 0.791 | 0.768 | 0.821 | yes |
| Weighted Avg. | 0.762 | 0.238 | 0.763 | 0.762 | 0.762 | 0.823 | |

### *4.4.12 LWL*

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 1955 | 71.8486 % |
| Incorrectly Classified Instances | 766 | 28.1514 % |
| Kappa statistic | 0.438 | |
| Mean absolute error | 0.3787 | |
| Root mean squared error | 0.4324 | |
| Relative absolute error | 75.7428 % | |
| Root relative squared error | 86.4776 % | |
| Coverage of cases (0.95 level) | 100 % | |
| Mean rel. region size (0.95 level) | 100 % | |
| Total Number of Instances | 2721 | |
| Ignored Class Unknown Instances | 6 | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.528 | 0.089 | 0.856 | 0.528 | 0.653 | 0.817 | no |
| | 0.911 | 0.472 | 0.657 | 0.911 | 0.763 | 0.818 | yes |
| Weighted Avg. | 0.718 | 0.28 | 0.757 | 0.718 | 0.708 | 0.818 | |

### *4.4.13 MultiClassClassifier*

Time taken to build model: 2.37 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2073 | 76.1852 % |
| Incorrectly Classified Instances | 648 | 23.8148 % |
| Kappa statistic | 0.5238 | |
| Mean absolute error | 0.3266 | |
| Root mean squared error | 0.412 | |

| | |
|---|---|
| Relative absolute error | 65.3197 % |
| Root relative squared error | 82.4071 % |
| Coverage of cases (0.95 level) | 98.7137 % |
| Mean rel. region size (0.95 level) | 93.3848 % |
| Total Number of Instances | 2721 |
| Ignored Class Unknown Instances | 6 |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.733 | 0.209 | 0.78 | 0.733 | 0.756 | 0.825 | no |
| | 0.791 | 0.267 | 0.746 | 0.791 | 0.768 | 0.821 | yes |
| Weighted Avg. | 0.762 | 0.238 | 0.763 | 0.762 | 0.762 | 0.823 | |

### *4.4.14 NaiveBayesUpdateable*

Time taken to build model: 0.03 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2042 | 75.0459 % |
| Incorrectly Classified Instances | 679 | 24.9541 % |
| Kappa statistic | 0.5015 | |
| Mean absolute error | 0.2969 | |
| Root mean squared error | 0.4293 | |
| Relative absolute error | 59.3732 % | |
| Root relative squared error | 85.8667 % | |
| Total Number of Instances | 2721 | |
| Ignored Class Unknown Instances | 6 | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.636 | 0.134 | 0.828 | 0.636 | 0.719 | 0.82 | no |
| | 0.866 | 0.364 | 0.702 | 0.866 | 0.776 | 0.819 | yes |
| Weighted Avg. | 0.75 | 0.248 | 0.765 | 0.75 | 0.747 | 0.82 | |

### *4.4.15 RandomForest*

Time taken to build model: 0.84 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2090 | 76.81 % |
| Incorrectly Classified Instances | 631 | 23.19 % |
| Kappa statistic | 0.5361 | |
| Mean absolute error | 0.2852 | |
| Root mean squared error | 0.41 | |
| Relative absolute error | 57.0401 % | |
| Root relative squared error | 81.9972 % | |
| Coverage of cases (0.95 level) | 95.9941 % | |
| Mean rel. region size (0.95 level) | 83.0761 % | |
| Total Number of Instances | 2721 | |
| Ignored Class Unknown Instances | 6 | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.788 | 0.252 | 0.76 | 0.788 | 0.773 | 0.835 | no |
| | 0.748 | 0.212 | 0.777 | 0.748 | 0.763 | 0.834 | yes |
| Weighted Avg. | 0.768 | 0.232 | 0.768 | 0.768 | 0.768 | 0.835 | |

### *4.4.16 NaiveBayes*

Time taken to build model: 0.39 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2042 | 75.0459 % |
| Incorrectly Classified Instances | 679 | 24.9541 % |
| Kappa statistic | 0.5015 | |
| Mean absolute error | 0.2969 | |
| Root mean squared error | 0.4293 | |

Relative absolute error 59.3732 %

Root relative squared error 85.8667 %

Total Number of Instances 2721

Ignored Class Unknown Instances 6

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.636 | 0.134 | 0.828 | 0.636 | 0.719 | 0.82 | no |
| | 0.866 | 0.364 | 0.702 | 0.866 | 0.776 | 0.819 | yes |
| Weighted Avg. | 0.75 | 0.248 | 0.765 | 0.75 | 0.747 | 0.82 | |

### *4.4.17 SGD*

Time taken to build model: 6.06 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 2076 76.2955 %

Incorrectly Classified Instances 645 23.7045 %

Kappa statistic 0.5261

Mean absolute error 0.237

Root mean squared error 0.4869

Relative absolute error 47.4101 %

Root relative squared error 97.3756 %

Coverage of cases (0.95 level) 76.2955 %

Mean rel. region size (0.95 level) 50 %

Total Number of Instances 2721

Ignored Class Unknown Instances 6

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.711 | 0.185 | 0.795 | 0.711 | 0.751 | 0.763 | no |
| | 0.815 | 0.289 | 0.736 | 0.815 | 0.774 | 0.762 | yes |
| Weighted Avg. | 0.763 | 0.237 | 0.766 | 0.763 | 0.762 | 0.763 | |

### *4.4.18 SimpleLogistic*

Time taken to build model: 26.9 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2044 | 75.1194 % |
| Incorrectly Classified Instances | 677 | 24.8806 % |
| Kappa statistic | 0.5026 | |
| Mean absolute error | 0.3526 | |
| Root mean squared error | 0.416 | |
| Relative absolute error | 70.5247 % | |
| Root relative squared error | 83.1956 % | |
| Coverage of cases (0.95 level) | 99.9265 % | |
| Mean rel. region size (0.95 level) | 98.0522 % | |
| Total Number of Instances | 2721 | |
| Ignored Class Unknown Instances | 6 | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.706 | 0.203 | 0.778 | 0.706 | 0.74 | 0.814 | no |
| | 0.797 | 0.294 | 0.729 | 0.797 | 0.761 | 0.815 | yes |
| Weighted Avg. | 0.751 | 0.248 | 0.754 | 0.751 | 0.751 | 0.815 | |

### *4.4.19 SMO*

Time taken to build model: 6.45 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2070 | 76.075 % |
| Incorrectly Classified Instances | 651 | 23.925 % |
| Kappa statistic | 0.5217 | |
| Mean absolute error | 0.2393 | |
| Root mean squared error | 0.4891 | |

| | |
|---|---|
| Relative absolute error | 47.8511 % |
| Root relative squared error | 97.8274 % |
| Coverage of cases (0.95 level) | 76.075 % |
| Mean rel. region size (0.95 level) | 50 % |
| Total Number of Instances | 2721 |
| Ignored Class Unknown Instances | 6 |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.715 | 0.193 | 0.789 | 0.715 | 0.75 | 0.761 | no |
| | 0.807 | 0.285 | 0.737 | 0.807 | 0.771 | 0.76 | yes |
| Weighted Avg. | 0.761 | 0.239 | 0.763 | 0.761 | 0.76 | 0.76 | |

### *4.4.20 ClassificationViaRegression*

Time taken to build model: 20.53 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 2114 | 77.692 % |
| Incorrectly Classified Instances | 607 | 22.308 % |
| Kappa statistic | 0.554 | |
| Mean absolute error | 0.3105 | |
| Root mean squared error | 0.4005 | |
| Relative absolute error | 62.1034 % | |
| Root relative squared error | 80.1059 % | |
| Coverage of cases (0.95 level) | 98.7137 % | |
| Mean rel. region size (0.95 level) | 91.7126 % | |
| Total Number of Instances | 2721 | |
| Ignored Class Unknown Instances | 6 | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.734 | 0.179 | 0.805 | 0.734 | 0.768 | 0.838 | no |
| | 0.821 | 0.266 | 0.753 | 0.821 | 0.785 | 0.837 | yes |

| | | | | | | |
|---|---|---|---|---|---|---|
| Weighted Avg. | 0.777 | 0.223 | 0.779 | 0.777 | 0.777 | 0.838 |

### *4.4.21 VotedPerceptron*

Time taken to build model: 2.95 seconds

=== Stratified cross-validation ===

=== Summary ===

| | | |
|---|---|---|
| Correctly Classified Instances | 1602 | 58.8754 % |
| Incorrectly Classified Instances | 1119 | 41.1246 % |
| Kappa statistic | 0.1805 | |
| Mean absolute error | 0.4114 | |
| Root mean squared error | 0.6413 | |
| Relative absolute error | 82.2732 % | |
| Root relative squared error | 128.261 % | |
| Coverage of cases (0.95 level) | 58.8754 % | |
| Mean rel. region size (0.95 level) | 50.0184 % | |
| Total Number of Instances | 2721 | |
| Ignored Class Unknown Instances | 6 | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|---|
| | 0.203 | 0.022 | 0.903 | 0.203 | 0.332 | 0.591 | no |
| | 0.978 | 0.797 | 0.549 | 0.978 | 0.703 | 0.592 | yes |
| Weighted Avg. | 0.589 | 0.408 | 0.726 | 0.589 | 0.517 | 0.592 | |

## *4.5 Επιλογή καταλληλότερου μοντέλου*

### *4.5.1 Αξιολόγηση με Cross Validation*

Για την αξιολόγηση των μοντέλων χρησιμοποιήθηκε η μέθοδος cross validation. Αυτή είναι μια τεχνική η οποία ελέγχει τον τρόπο που θα χρησιμοποιηθούν τα δεδομένα σε μια στατιστική ανάλυση. Χρησιμοποιείται σε περιπτώσεις όπου στόχος είναι η πρόβλεψη, και επιχειρείται να βρεθεί πόσο ακριβή είναι τα αποτελέσματα. Έτσι ένα σύνολο δεδομένων χωρίζεται σε δύο ή περισσότερα υποσύνολα, χρησιμοποιώντας το ένα υποσύνολο για την εκπαίδευση του αλγορίθμου (training dataset) και το άλλο για αξιολόγηση του (testing dataset). Για πιο ασφαλή αποτελέσματα, γίνονται πολλές επαναλήψεις του cross-validation χρησιμοποιώντας διαφορετικά υποσύνολα κάθε φορά, και βγαίνει ο μέσος όρος από όλες τις επαναλήψεις. Στην εργασία επιλέχθηκε το είδος k-fold cross-validation με k=10. Σε αυτή τη μέθοδο το αρχικό σύνολο δεδομένων χωρίζεται σε k υποσύνολα. Από τα k υποσύνολα ένα χρησιμοποιείται για επαλήθευση (test dataset) και τα υπόλοιπα (k-1) υποσύνολα χρησιμοποιούνται για την εκπαίδευση (training data). Η διαδικασία αυτή επαναλαμβάνεται k φορές (όσες και τα folds), όπου κάθε ένα από τα k υποσύνολα χρησιμοποιείται μία φορά σαν δεδομένα επαλήθευσης. Το μεγαλύτερο πλεονέκτημα αυτής της μεθόδου είναι ότι όλα τα αντικείμενα του συνόλου δεδομένων χρησιμοποιούνται και για εκπαίδευση αλλά και για επαλήθευση.

Από τα στοιχεία που επιστρέφει η αξιολόγηση (4.4) το πιο αξιόπιστο ως δείκτης μέτρησης της ακρίβειας είναι η καμπύλη ROC ή αλλιώς η περιοχή κάτω από αυτήν. Είναι μια γραφική παράσταση που αναπαριστά την απόδοση ενός μοντέλου κατηγοριοποίησης συναρτήσει της μεταβολής του κατωφλίου διαχωρισμού. Αυτό το πετυχαίνει με τον λόγο true positive rate (sensitivity ή recall) προς false positive rate (fall-out) για διάφορες τιμές κατωφλίου. Οι τιμές που μπορεί να πάρει κυμαίνονται από 0 έως 1. Συγκεκριμένα:

- 0.9-1 = άριστο
- 0.8-0.9 = καλό
- 0.7-0.8 = ικανοποιητικό
- 0.6-0.7 = φτωχό
- 0.5-0.6 = αποτυχία

Ένα παράδειγμα του πώς πραγματοποιείται το cross validation με τη βιβλιοθήκη του weka είναι το εξής (για 10-fold cross validation):

Έστω ότι έχουμε 100 στιγμιότυπα εκπαίδευσης:

1. Το weka χωρίζει τα στιγμιότυπα σε 10 ίσου μεγέθους μέρη
2. Χρησιμοποιεί τα 9 τελευταία μέρη για την εκπαίδευση και τον εφαρμόζει στο πρώτο μέρος για τον έλεγχο
3. Κάνει το ίδιο για το σύνολο 2 έως το 10 και φτιάχνει 9 ακόμα μοντέλα κατηγοριοποίησης
4. Υπολογίζει το μέσο όρο της απόδοσης των 10 κατηγοριοποιητών

### *4.5.2 Τελική επιλογή αλγορίθμου*

Παρατηρούμε από την ενότητα 4.4 ότι τα επικρατέστερα μοντέλα από άποψη ακρίβειας είναι σε φθίνουσα σειρά τα εξής: KStar (ROC: 0.852), BayesNet (ROC: 0.841), ClassificationViaRegression (ROC: 0.838), RandomForest (ROC: 0.835), MultiClassClassifier (ROC: 0.823), Logistic (ROC: 0.823), DecisionTable (ROC: 0.827), NaiveBayesUpdatable (ROC: 0.82), NaiveBayes (ROC: 0.82), FilteredClassifier (ROC:0.82), BayesSimple (ROC: 0.82), LWL (ROC:0.818), SimpleLogistic (ROC: 0.815).

Πάρα πολύ μεγάλη σημασία εκτός από την ακρίβεια παίζει και η ταχύτητα εκπαίδευσης. Αυτό συμβαίνει, διότι θέλουμε να υπάρχει η δυνατότητα επανεκπαίδευσης του μοντέλου μας online και ως εκ τούτου ο χρόνος εκπαίδευσης να είναι όσο το δυνατόν μικρότερος.

Οι χρόνοι εκπαίδευσης των παραπάνω μοντέλων σε αύξουσα σειρά είναι: KStar (0 seconds), LWL (0 seconds), NaiveBayesUpdatable (0.03 seconds), FilteredClassifier (0.04 seconds), FilteredClassifier (0.04 seconds), BayesSimple (0.06 seconds), NaiveBayes (0.39 seconds), BayesNet (0.52 seconds), DecisionTable (0.61 seconds), RandomForest (0.84 seconds), MultiClassClassifier (2.37 seconds), Logistic (5.85 seconds), ClassificationViaRegression (20.53 seconds), SimpleLogistic (26.9 seconds).

Μία επιλογή που συνδυάζει την καλή ακρίβεια με τη μέγιστη ταχύτητα είναι ο αλγόριθμος NaiveBayesUpdatable, ο οποίος είναι και ανανεώσιμος, δηλαδή μπορεί να επανεκπαιδεύσει ένα μοντέλο με ένα καινούριο dataset από instances χωρίς να χρειάζεται και το dataset της αρχικής εκπαίδευσης.

## *4.6 Εφαρμογή μοντέλου στα ελληνικά tweets*

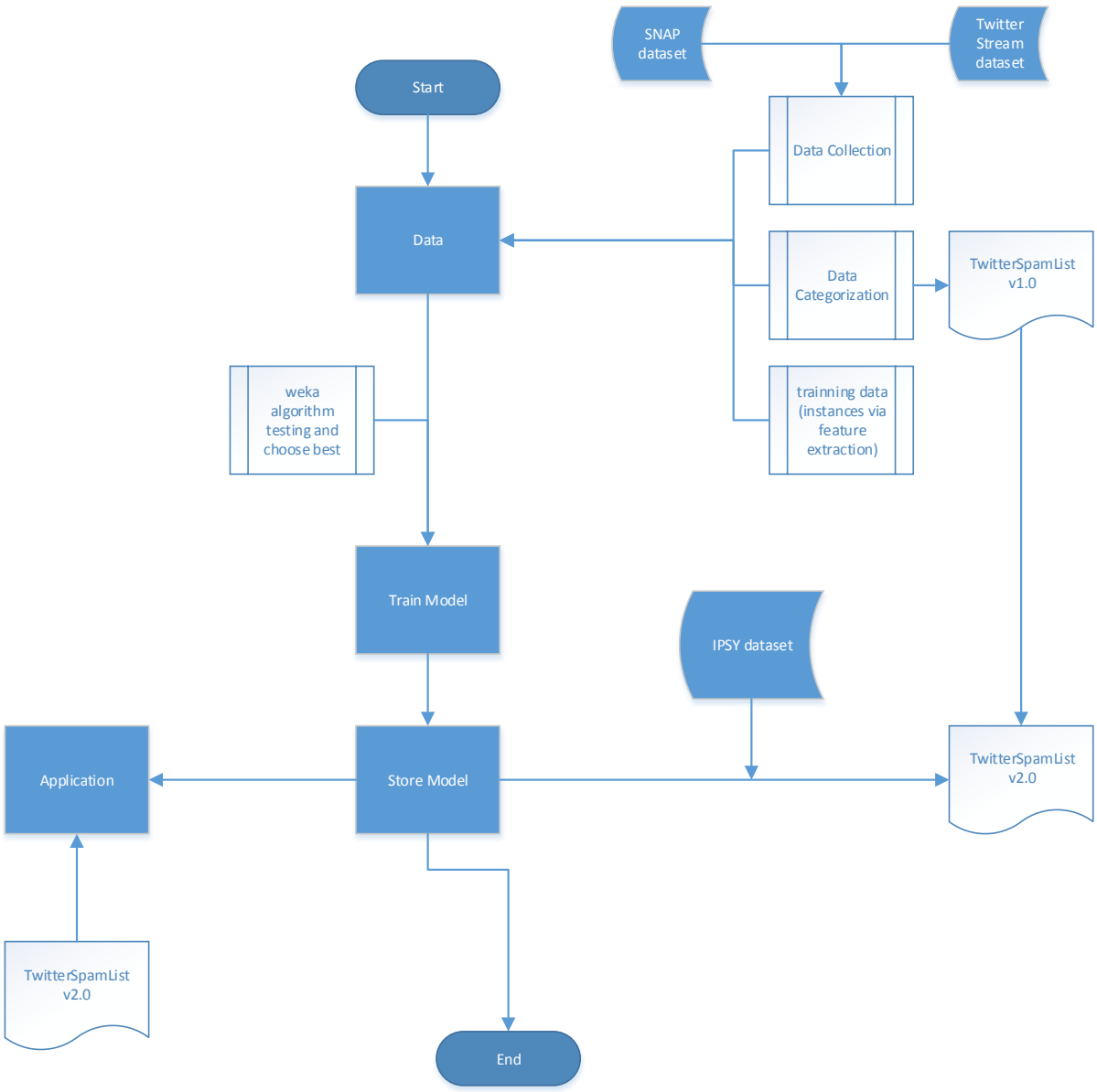
Τέλος, το μοντέλο που εκπαιδεύτηκε με βάση τα tweets από το Twitter Stream και το SNAP Twitter Dataset του Stanford University και με αλγόριθμο εκπαίδευσης τον NaiveBayesUpdatable, εφαρμόζεται στα ελληνικά tweets που έχει συλλέξει το ΙΠΣΥ. Έτσι προκύπτει μία «μαύρη» λίστα από ελληνικούς spammers, η οποία μαζί με αυτή που προέκυψε από τα άλλα datasets διαμορφώνουν την TwitterSpamList, η οποία αποδεικνύεται πολύ χρήσιμη για την ορθή λειτουργία της web εφαρμογής.

# 5

# *Σχεδίαση Συστήματος*

## *5.1 Αρχιτεκτονική*

Η αρχιτεκτονική του συστήματος είναι αρκετά απλή, ενώ δεν είχε διαμορφωθεί εξ αρχής στη μορφή που κατέληξε. Ένα δείγμα της αρχιτεκτονικής αυτής φαίνεται στο παρακάτω διάγραμμα. Το διάγραμμα περιλαμβάνει και την ανάπτυξη της web εφαρμογής παρόλο που αυτή αναλύεται σε επόμενη ενότητα.




## 5.2 *Περιγραφή Κλάσεων*

### *5.2.1 FindSpammers*

Η κλάση αυτή χρησιμοποιήθηκε για την κατηγοριοποίηση των χρηστών με βάση τα δεδομένα από το Twitter Stream και το SNAP Twitter Dataset σε δύο κατηγορίες: spammers και μη. Ουσιαστικά ψάχνει στο περιεχόμενο των tweets για συνδέσμους σε διευθύνσεις, οι οποίες ανήκουν σε μία από τις τρεις «μαύρες λίστες»: στη SURBL, στη URIBL, στη Google Safe Browsing. Αναλυτικότερα υλοποιεί τις εξής μεθόδους:

1. HasLinks: η μέθοδος αυτή λαμβάνει ως όρισμα το κείμενο από ένα tweet και επιστρέφει true εάν το tweet περιέχει τουλάχιστον ένα σύνδεσμο σε κάποια σελίδα.

Στην ουσία αναζητεί μέσα στο tweet τις συμβολοσειρές: «http», «https», «ftp», «www.», «bit.ly/».

2. PlainUrl: η μέθοδος αυτή λαμβάνει ως όρισμα το κείμενο από ένα tweet και επιστρέφει ένα String που είναι μόνο ο σύνδεσμος προς άλλη σελίδα. Ουσιαστικά χρησιμοποιείται για να απομονώσει τους συνδέσμους, με τη χρήση patterns.
3. expandShortURL: η μέθοδος αυτή λαμβάνει ως όρισμα έναν σύνδεσμο από την PlainUrl και επιστρέφει την επεκταμένη μορφή του συνδέσμου. Με άλλα λόγια, λόγω της αυξημένης πιθανότητας να έχουμε συντομευμένες διευθύνσεις μέσα στα tweets και αυτό να κρύβει και το ενδεχομλενως blacklisted domain τους, η μέθοδος επεκτείνει τις διευθύνσεις όσο γίνεται περισσότερο. Αυτό το πετυχαίνει με επαναλαμβανόμενη χρήση της μεθόδου getHeaderField της κλάσης HttpURLConnection.
4. getDomainName: η μέθοδος αυτή λαμβάνει ως όρισμα τον σύνδεσμο από την expandShortURL και εξάγει από αυτόν το domain name. Επιστρέφει δηλαδή ένα String που είναι το domain name της διεύθυνσης. Αυτό γίνεται διότι η αναζήτηση που γίνεται αργότερα στις blacklists γίνεται με βάση αυτό και όχι με βάση όλη τη διεύθυνση. Αυτό το πετυχαίνει με τη χρήση της μεθόδου getHost της κλάσης URI.
5. isSpam: : η μέθοδος αυτή λαμβάνει ως όρισμα τον σύνδεσμο από την expandShortURL και εξάγει από αυτόν το domain name με τη χρήση της μεθόδου getDomainName. Στη συνέχεια αναζητεί το domain name αυτό μέσα σε τρεις blacklists, τις: SURBL, URIBL, Google Safe Browsing. Αυτό το πετυχαίνει με τη χρήση των API τους. Αξίζει να σημειωθεί ότι από την αναζήτηση αυτή των domain names εξαιρέσαμε όλους τους συνδέσμους σε διευθύνσεις με domain τα facebook.com, twitter.com, instagram.com, διότι οι blacklists είναι υπερευαίσθητες σε αυτά και τα εμφανίζουν όλα ως spam. Η μέθοδος στο τέλος επιστρέφει true, εάν το domain name έχει εντοπιστεί έστω και σε μία από τις blacklists, ή false αν δεν εντοπίστηκε σε καμία.

Τέλος εάν για κάποιο tweet η isSpam επιστρέψει true, τότε η FindRobots ανακτά το όνομα του χρήστη που το δημοσίευσε και το γράφει σε ένα αρχείο, που λέγεται SpamNames. Όταν ολοκληρωθεί η διαδικασία της κατηγοριοποίησης αυτής το αρχείο αυτό περιέχει όλους τους spammers εκτός από αυτούς που βρίσκονται στα ελληνικά tweets.

Επίσης εάν η isSpam για κάποιο tweet επιστρέψει false, τότε η FindRobots ανακτά το όνομα του χρήστη που το δημοσίευσε και το γράφει σε ένα αρχείο, που λέγεται HumanNames. Έτσι συγκεντρώνουμε και έναν αριθμό από μη spammers για την εκπαίδευση του μοντέλου μας.

### *5.2.2 DataRetriever*

Η κλάση αυτή χρησιμοποιήθηκε για την ανάκτηση των χαρακτηριστικών που μας ενδιαφέρουν (3.1) από spammers και μη, με σκοπό τη διαμόρφωση των στιγμιότυπων εκπαίδευσης του μοντέλου μας. Ουσιαστικά διαβάζει τα δύο αρχεία που έχουμε φτιάξει το SpamNames και το HumanNames και για κάθε ένα όνομα καλεί μια σειρά μεθόδους της για την εξαγωγή καθενός χαρακτηριστικού που θα εισαχθεί σε στιγμιότυπο εκπαίδευσης. Για τη χρήση του Twitter API αξιοποίηθηκε η βιβλιοθήκη Twitter4j. Αναλυτικότερα υλοποιεί τις εξής μεθόδους:

1. Ratio: Αυτή η μέθοδος παίρνει ως όρισμα των χρήστη του Twitter που εξετάζουμε τη δεδομένη στιγμή και επιστρέφει την αναλογία friend και followers (3.1.1). Με χρήση του Twitter API ανακτά των αριθμό των friends και των followers και φτιάχνει το λόγο που είδαμε στο 3.1.1. Πιο συγκεκριμένα χρησιμοποιεί τις μεθόδους getFollowersCount και getFriendsCount της κλάσης User.
2. LevDist: Αυτή η μέθοδος παίρνει ως όρισμα των χρήστη του Twitter που εξετάζουμε τη δεδομένη στιγμή και επιστρέφει έναν δείκτη ομοιότητας μεταξύ των tweets που έχει δημοσιεύσει και είναι διαθέσιμα σε εμάς. Αναλυτικότερα χρησιμοποιεί τη μέθοδο getUserTimeline της κλάσης User για να ανακτήσει τα πιο πρόσφατα tweets του χρήστη. Στη συνέχεια υπολογίζει τη Levenshtein απόσταση μεταξύ κάθε δύο tweets (εξετάζονται όλα τα tweets σε συνδυασμό με όλα). Στο τέλος εξάγει τον μέσο όρο των αποστάσεων αυτών. Για την απόσταση Levenshtein θα κάνουμε λόγο πιο κάτω, ωστόσο στο σημείο αυτό ότι μετράει με πόσες αλλαγές του ενός χαρακτήρα μετατρέπεται μία συμβολοσειρά σε κάποια άλλη.
3. NumHttp: Αυτή η μέθοδος παίρνει ως όρισμα των χρήστη του Twitter που εξετάζουμε τη δεδομένη στιγμή και τελευταία tweets του (όπως παραπάνω) και επιστρέφει πόσες διευθύνσεις σε σελίδες περιλαμβάνουν τα tweets του χρήστη (3.1.1) κατά μέσο όρο. Ο αριθμός που επιστρέφει είναι αυτός ο μέσος όρος.

4. AtReply: Αυτή η μέθοδος παίρνει ως όρισμα των χρήστη του Twitter που εξετάζουμε τη δεδομένη στιγμή και τελευταία tweets του (όπως παραπάνω) και επιστρέφει πόσες αναφορές-@ περιλαμβάνουν τα tweets του χρήστη (3.1.1) κατά μέσο όρο. Ο αριθμός που επιστρέφει είναι αυτός ο μέσος όρος.
5. InterTweetDelay: Αυτή η μέθοδος παίρνει ως όρισμα των χρήστη του Twitter που εξετάζουμε τη δεδομένη στιγμή και τελευταία tweets του (όπως παραπάνω) και επιστρέφει έναν δείκτη συχνότητας των tweets που έχει δημοσιεύσει και είναι διαθέσιμα σε εμάς. Αναλυτικότερα χρησιμοποιεί τη μέθοδο getCreatedAt της κλάσης Status για να ανακτήσει τη χρονική στιγμή που αναρτήθηκε το κάθε tweet του χρήστη. Στη συνέχεια υπολογίζει το πιο συχνά χρησιμοποιούμενο χρονικό διάστημα σε λεπτά μεταξύ δύο διαδοχικών tweets και με τη βοήθεια της μεθόδου frequency της κλάσης Collections. Στο τέλος μετράει το μέγιστο πλήθος των ίδιων χρονικών διαστημάτων και ο μέσος όρος αυτών προς όλα τα χρονικά διαστήματα επιστρέφεται από τη μέθοδο.
6. Source: Αυτή η μέθοδος παίρνει ως όρισμα των χρήστη του Twitter που εξετάζουμε τη δεδομένη στιγμή και τελευταία tweets του (όπως παραπάνω) και βρίσκει από πού στέλνει συνήθως ο χρήστης τα tweets του. Για το σκοπό αυτό χρησιμοποιεί τη μέθοδο getSource της κλάσης Status για κάθε tweet. Τέλος, βρίσκει την πιο συχνή πηγή των tweets και την επιστρέφει ως String.
7. Verified: Αυτή η μέθοδος παίρνει ως όρισμα των χρήστη του Twitter που εξετάζουμε και βρίσκει αν είναι επιβεβαιωμένος χρήστης (verified). Για το σκοπό αυτό χρησιμοποιεί τη μέθοδο isVerified της κλάσης User και επιστρέφει true αν είναι και false αν δεν είναι.

Τέλος η DataRetriever γράφει σε μορφή διανύσματος τα χαρακτηριστικά αυτά για κάθε χρήστη και προσθέτει ένα πεδίο ακόμα το spam, το οποίο εάν ο χρήστης προέρχεται από την SpamList θα παίρνει την τιμή yes, ενώ αν προέρχεται από την HumanList, θα παίρνει την τιμή no.

### *5.2.3 CSVMaker*

Η κλάση αυτή μετατρέπει τα διανύσματα (στιγμιότυπα εκπαίδευσης) που παράγει η κλάση DataRetriever σε ένα CSV αρχείο έτοιμο να εισαχθεί στον αλγόριθμο εκπαίδευσης για να εκπαιδευτεί το μοντέλο.

### *5.2.4 BuildClassifier*

Η κλάση αυτή έχει σκοπό την κατασκευή και εκπαίδευση του μοντέλου κατηγοριοποίησης. Σε προηγούμενη ενότητα (4.5) αποφασίσαμε ότι μία επιλογή που συνδυάζει την καλή ακρίβεια με τη μέγιστη ταχύτητα είναι ο αλγόριθμος NaiveBayesUpdatable. Σε αυτή την κλάση χρησιμοποιούμε τη μέθοδο getDataSet της κλάσης DataSource η οποία παίρνει ως όρισμα το αρχείο CSV που κατασκευάσαμε με την κλάση DataRetriever. Το αρχείο αυτό λοιπόν με αυτή τη μέθοδο μετατρέπεται σε σύνολο στιγμιοτύπων όπως το χρειάζεται ο αλγόριθμος εκπαίδευσης του μοντέλου. Στη συνέχεια κατασκευάζουμε ένα μοντέλο NaiveBayeUpdatable με τον ομώνυμο κατασκευαστή της βιβλιοθήκης weka. Μετά με τη μέθοδο buildClassifier τα βιβλιοθήκης του weka και με είσοδο το σύνολο των στιγμιοτύπων, εκπαιδεύεται το μοντέλο με βάση τα δεδομένα εκαπίδευσης που έχουμε συλλέξει και αποθηκεύεται.

## *5.3 Κωδικοποίηση αρχείων*

### *5.3.1 ARRF αρχεία*

Ένα ARFF (Attribute-Relation File Format) αρχείο είναι ένα ASCII αρχείο κειμένου που περιγράφει μία λίστα από στιγμιότυπα με κοινά πεδία (χαρακτηριστικά). Τα αρχεία ARFF αναπτύχθηκαν από το University of Waikato για να χρησιμοποιηθούν από τα μοντέλα μηχανικής μάθησης του weka.

Τα αρχεία ARFF αποτελούνται από δύο τμήματα. Το πρώτο, Header information,ακολουθείται από το Data information.

Το Header του ARFF περιέχει το όνομα της σχέσης, μία λίστα χαρακτηριστικών και τους τύπους τους. Ένα παράδειγμα από τα δεδομένα εκπαίδευσης που εξετάζονται σε αυτή την εργασία:

```
@relation classify

@attribute ratio numeric
@attribute levTweet numeric
@attribute HttpCnt numeric
@attribute AtReplyCnt numeric
@attribute inter numeric
@attribute source {web,API}
@attribute verified numeric
@attribute robot {yes}

@data
0.315961,76.815,0.25,0.15,0.1,web,1,yes
0.54,33,0.245,0.22,0.2,API,1,yes
```

# 6

# *Ανάπτυξη web εφαρμογής*

Καθοριστικής σημασίας βήμα για την ολοκλήρωση αυτής της εργασίας ήταν η δημιουργία μιας web εφαρμογής, η οποία θα δίνει στο χρήστη τη δυνατότητα να ελέγχει έναν οποιοδήποτε λογαριασμό για το εάν είναι spammer ή όχι. Επιπλέον θα του δίνει τη δυνατότητα να ελέγχει ποιοι από τους friends και followers του είναι spammers, ενώ θα μπορεί για να αναφέρει ως spammer ή ως μη spammer οποιονδήποτε λογαριασμό. Με αυτόν τον τρόπο η TwitterSpammerList θα εμπλουτίζεται και τυχόν λάθη της θα διορθώνονται. Επίσης το μοντέλο κατηγοριοποίησης θα επανεκπαιδεύεται με βάση τις αναφορές των χρηστών του.

## *6.1 Δομή του web application*

Σκοπός είναι να κατασκευαστεί μία εφαρμογή που θα φέρει σε πέρας τους στόχους μας, αλλά ταυτόχρονα θα είναι εύχρηστη και αποδοτική για τον χρήστη. Έτσι αποτελείται ουσιαστικά από τρεις σελίδες: index.jsp, result.jsp, update.jsp.

### *6.1.1 index.jsp*

Αυτή είναι και η πρώτη σελίδα που βλέπει ο χρήστης. Εδώ εμφανίζεται το όνομα της εφαρμογής, μία σύντομη περιγραφή και η δυνατότητα να επιλέξει μία από τις τρεις λειτουργίες της:

- Check single account: εδώ ο χρήστης μπορεί να ελέγξει εάν ένας λογαριασμός είναι spammer ή όχι. Στην ουσία ο χρήστης εισάγει το όνομά του λογαριασμού και συνεχίζει.
- Check friends of: εδώ ο χρήστης μπορεί να ελέγξει έναν έναν τους φίλους ενός λογαριασμού για το αν είναι spammers ή όχι. Στην ουσία ο χρήστης εισάγει το όνομά του λογαριασμού και συνεχίζει.
- Check followers of: εδώ ο χρήστης μπορεί να ελέγξει έναν ένα τους followers ενός λογαριασμού για το αν είναι spammers ή όχι. Στην ουσία ο χρήστης εισάγει το όνομά του λογαριασμού και συνεχίζει.

### *6.1.2 result.jsp*

Αυτή είναι η δεύτερη σελίδα που θα δει ο χρήστης. Ανάλογα την επιλογή που έκανε ο χρήστης στην index.jsp, αλλάζει και το περιεχόμενο της result.jsp.

- Αν ο χρήστης είχε επιλέξει τη λειτουργία «Check single account», τότε στη σελίδα result εμφανίζεται η φωτογραφία του προφίλ του χρήστη μαζί με μία σύντομη περιγραφή του και τα τελευταία του tweets. Σε εμφανές σημείο φαίνεται το αποτέλεσμα της αναζήτησης. Αρχικά το σύστημα αναζητά το όνομα αυτό στην TwitterSpamList. Εάν το βρει τοτε εμφανίζεται το μήνυμα που ενημερώνει το χρήστη και η επιλογή αν θέλει να τον αναφέρει ως μη spammer. Εάν δεν το βρει τότε καλείται η κλάση ClassifyAccount η οποία χρησιμοποιεί το μοντέλο κατηγοριοποίησης και αναλύεται παρακάτω. Επίσης ανάλογα με το αποτέλεσμα εμφανίζεται κατάλληλο μήνυμα. Τέλος υπάρχει η δυνατότητα να αναφερθεί ο χρήστης σαν spammer ή όχι spammer.
- Αν ο χρήστης είχε επιλέξει τη λειτουργία «Check friends of», τότε στη σελίδα result.jsp εμφανίζονται σε μορφή λίστας οι φωτογραφίες των friends και δίπλα το μήνυμα αν είναι ή όχι spammers. Εδώ γίνεται για κάθε έναν αναζήτηση στην

TwitterSpamList και όχι με το μοντέλο καθότι θα υπήρχε τεράστια καθυστέρηση για τον έλεγχο όλων των φίλων. Αυτό οφείλεται στους περιορισμούς που επιβάλει το Twitter. Και εδώ υπάρχει η δυνατότητα να αναφερθεί ο κάθε φίλος σαν spammer ή όχι spammer.

- Αν ο χρήστης είχε επιλέξει τη λειτουργία «Check followers of», τότε στη σελίδα result.jsp εμφανίζονται σε μορφή λίστας οι φωτογραφίες των followers και δίπλα το μήνυμα αν είναι ή όχι spammers. Εδώ γίνεται για κάθε έναν αναζήτηση στην TwitterSpamList και όχι με το μοντέλο καθότι θα υπήρχε τεράστια καθυστέρηση για τον έλεγχο όλων των followers. Αυτό οφείλεται στους περιορισμούς που επιβάλει το Twitter. Και εδώ υπάρχει η δυνατότητα να αναφερθεί ο κάθε follower σαν spammer ή όχι spammer.

### *6.1.3 update.jsp*

Αυτή η σελίδα εμφανίζεται σε περίπτωση που ο χρήστης επέλεξε να αναφέρει κάποιον λογαριασμό ως spammer ή όχι spammer. Το πρόγραμμα εδώ καλεί την μέθοδο Update της κλάσης ClassifyAccount η οποία επανεκπαιδεύει το μοντέλο με βάση την επιλογή του χρήστη. Επίσης τροποποιεί την TwitterSpamList κατάλληλα προσθέτοντας ή αφαιρώντας το όνομα του λογαριασμού από αυτήν. Τέλος εμφανίζει μήνυμα στον χρήστη για την επιτυχία ή την αποτυχία της επανεκπαίδευσης.

## *6.2 Περιγραφή κλάσεων*

### *6.2.1 ClassifyAccount*

Ουσιαστικά πρόκειται για την κλάση που περιλαμβάνει όλες τις λειτουργίες που χρειάζεται να γίνουν για την εξυπηρέτηση του χρήστη. Από την χρήση του μοντέλου για κατηγοριοποίηση μέχρι και την επανεκπαίδευσή του. . Αναλυτικότερα υλοποιεί τις εξής μεθόδους:

#### *6.2.1.1 AddSpammer*

Η μέθοδος αυτή παίρνει ως όρισμα το όνομα του λογαριασμού που ο χρήστης ανέφερε ως spammer και το προσθέτει στην TwitterSpammerList.

#### *6.2.1.2 RemoveSpammer*

Η μέθοδος αυτή παίρνει ως όρισμα το όνομα του λογαριασμού που ο χρήστης ανέφερε ως μη spammer και το αφαιρεί από την TwitterSpammerList.

#### *6.2.1.3 CsvToArrf*

Η μέθοδος αυτή παίρνει ως όρισμα το στιγμιότυπο που έχει φτιαχτεί για την επανεκπαίδευση του μοντέλου και τροποποιεί τη μορφή του από csv σε arrf που χρειάζεται το μοντέλο για να κατηγοριοποιήσει και επανεκπαιδευτεί.

#### *6.2.1.4 Detect*

Η μέθοδος αυτή παίρνει ως όρισμα το arrf αρχείο που περιέχει το στιγμιότυπο που πρέπει να κατηγοριοποιηθεί από το μοντέλο. Επιστρέφει την συμβολοσειρά «human», εάν το μοντέλο αποφανθεί ότι είναι μη spam. Αλλιώς επιστρέφει «robot», εάν το μοντέλο αποφανθεί ότι είναι spam. Το μοντέλο κατηγοριοποιεί με χρήση της μεθόδου distributionForInstance της κλάσης Classifier της βιβλιοθήκης weka.

#### *6.2.1.5 RetrieveDataOnce*

Η μέθοδος αυτή χρησιμοποιείται για την εξαγωγή χαρακτηριστικών για την κατηγοριοποίηση ενός χρήστη. Παίρνει ως όρισμα των χρήστη του Twitter που εξετάζουμε τη δεδομένη στιγμή και τελευταία tweets του και δημιουργεί ένα csv αρχείο που περιέχει το στιγμιότυπο προς κατηγοριοποίηση.

#### *6.2.1.6 Update*

Η μέθοδος αυτή παίρνει ως όρισμα το arrf αρχείο που περιέχει το στιγμιότυπο που πρέπει να χρησιμοποιηθεί για την επανεκπαίδευση του μοντέλου και το επανεκπαιδεύει. Για το σκοπό αυτό χρησιμοποιείται η μέθοδος updateClassifier της κλάσης Classifier της βιβλιοθήκης weka.

# 7

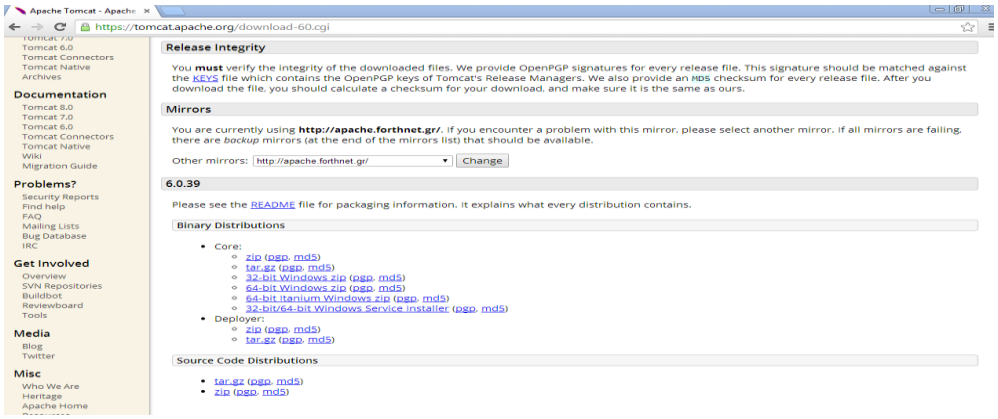
# *Οδηγός χρήσης web εφαρμογής*

## *7.1 Εγκατάσταση*

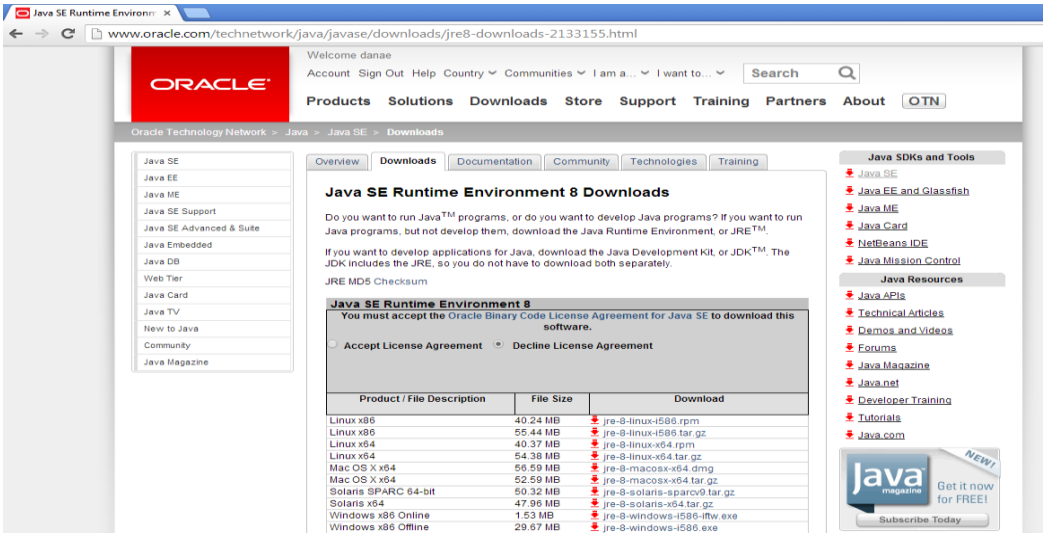
Για την εγκατάσταση του web app θα χρειαστεί η σωστή εγκατάσταση του Apache Tomcat 6 καθώς και το απαραίτητου Oracle JRE 8. Επιπλέον υπάρχει cd που περιέχει το TwitterSpamDetector.war αρχείο.

### *7.1.1 Εγκατάσταση Apache Tomcat 6*

- Κατεβάστε τον Apache Tomcat 6: https://tomcat.apache.org/download-60.cgi, ανάλογα με το λειτουργικό σας σύστημα και τη διανομή του.



- Αποσυμπιέστε το αρχείο σε όποιο directory επιθυμείται. Το directory που περιέχει το φάκελο bin του Apache Tomcat 6 θα το ονομάσουμε tomcatpath. (Θα είναι της μορφής C:\Users\danae\Desktop\app\apache-tomcat-6.0.39-windows-x86\apache-tomcat-6.0.39)
- Κατεβάστε το Oracle JRE 8: http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html, ανάλογα με το λειτουργικό σας σύστημα και τη διανομή του.




- Το directory που περιέχει το φάκελο bin του Oracle JRE 8 θα το ονομάσουμε jrepath. (Θα είναι της μορφής C:\PROGRA~1\Java\jre8)
- Κατασκευάστε μία μεταβλητή συστήματος με όνομα CATALINA_HOME και τιμή το tomcatpath.
- Κατασκευάστε μία μεταβλητή συστήματος με όνομα JRE_HOME και τιμή το jrepath.
- Αντιγράψτε το TwitterSpamDetector.war που περιέχεται στο cd στο φάκελο webapps του Apache Tomcat 6.

## *7.2 Χρήση*

### *7.2.1 Εκκίνηση Tomcat*

Για windows:

- Επιλέξτε run από το μενού Έναρξη
- Εισάγετε την εντολή %CATALINA_HOME%\bin\startup.bat

Για linux:

- Ανοίξτε ένα νέο Terminal
- Εισάγετε την εντολή $CATALINA_HOME/bin/startup.sh

Για να επιβεβαιώσετε ότι ο Tomcat λειτουργεί κανονικά ανοίγετε τον web browser σας και γράγετε την εξής διεύθυνση: http://localhost:8080/.

Εάν εμφανίζεται η σελίδα του Tomcat, τότε όλα λειτουργούν σωστά.

### *7.2.2 Web App*

Στη συνέχεια γράφουμε τη διεύθυνση http://localhost:8080/TwitterSpamDetector και το application λειτουργεί!! Μην ξεχνάτε το Application για να λειτουργήσει χρειάζεται σύνδεση internet.

Μπορείτε να επιλέξετε μία από τις τρεις λειτουργίες:

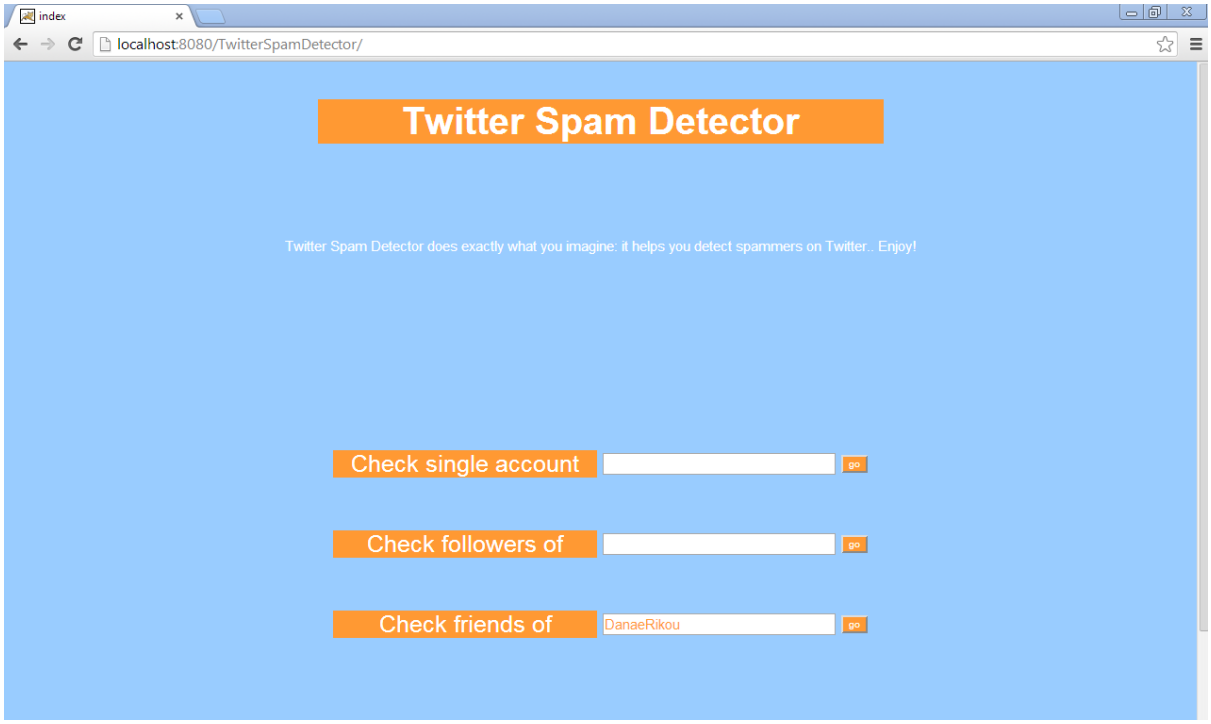
- Check single account: εδώ μπορείτε να ελέγξετε εάν ένας λογαριασμός είναι spammer ή όχι. Στην ουσία εισάγετε το όνομά του λογαριασμού και συνεχίζετε. Στην επόμενη σελίδα εμφανίζεται το όνομα του λογαριασμού, μία φωτογραφία του και μία σύντομη περιγραφή του. Εδώ εμφανίζεται και το αποτέλεσμα της κατηγοριοποίησης. Τέλος μπορείτε να αναφέρετε τον λογαριασμό ως spammer ή ως μη spammer και να τον

προσθέσετε ή αφαιρέσετε στη TwitterSpamList και με βάση αυτό να επανεκπαιδεύσετε το μοντέλο.

- Check friends of: εδώ μπορείτε να ελέγξετε εάν ένας λογαριασμός έχει φίλους που είναι spammer. Στην ουσία εισάγετε το όνομά του λογαριασμού και συνεχίζετε. Στην επόμενη σελίδα εμφανίζονται τα ονόματα των φίλων του λογαριασμού που ανήκουν στην TwitterSpamList, μία φωτογραφία τους και μία σύντομη περιγραφή τους. Τέλος για κάθε έναν από αυτούς μπορείτε να αναφέρετε τον λογαριασμό ως spammer ή ως μη spammer και να τον προσθέσετε ή αφαιρέσετε στη TwitterSpamList και με βάση αυτό να επανεκπαιδεύσετε το μοντέλο.
- Check followers of: εδώ μπορείτε να ελέγξετε εάν ένας λογαριασμός έχει followers που είναι spammer. Στην ουσία εισάγετε το όνομά του λογαριασμού και συνεχίζετε. Στην επόμενη σελίδα εμφανίζονται τα ονόματα των followers του λογαριασμού που ανήκουν στην TwitterSpamList, μία φωτογραφία τους και μία σύντομη περιγραφή τους. Τέλος για κάθε έναν από αυτούς μπορείτε να αναφέρετε τον λογαριασμό ως spammer ή ως μη spammer και να τον προσθέσετε ή αφαιρέσετε στη TwitterSpamList και με βάση αυτό να επανεκπαιδεύσετε το μοντέλο.

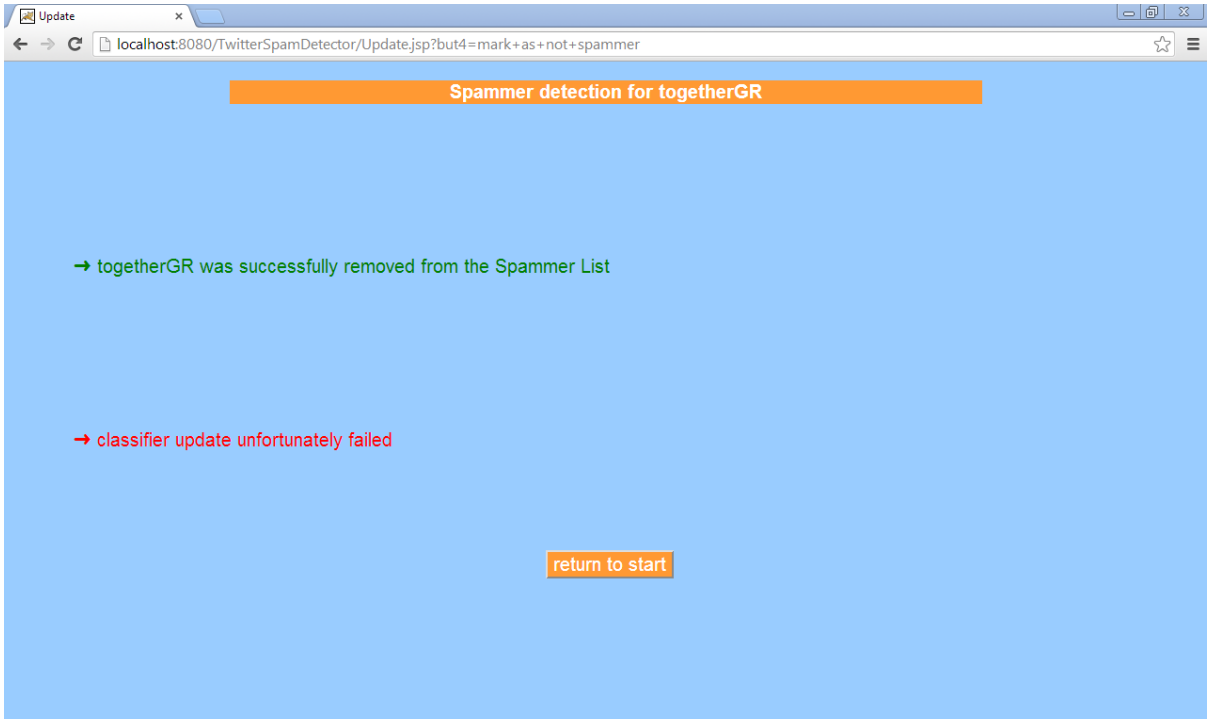
#### *7.2.2.1 Παράδειγμα χρήσης*

- Επιλέγω την λειτουργία «Check friends of:» και πληκτρολογό το όνομα του λογαριασμού μου:



- Εμφανίζεται η λίστα με τους φίλους μου που είναι spammers:


- Επιλέγω να αναφέρω ως μη spammer τον φίλο «bones7456» και εμφανίζεται μήνυμα επιτυχίας ή αποτυχίας:



### *7.2.3 Απενεργοποίηση Tomcat*

Για windows:

- Επιλέξτε run από το μενού Έναρξη
- Εισάγετε την εντολή %CATALINA_HOME%\bin\shutdown.bat

Για linux:

- Ανοίξτε ένα νέο Terminal
- Εισάγετε την εντολή $CATALINA_HOME/bin/shutdown.sh

# 8

# *Επίλογος*

## *8.1 Σύνοψη και συμπεράσματα*

Συμπερασματικά η διπλωματική πέτυχε σε γενικές γραμμές τους στόχους της. Πιο συγκεκριμένα σχεδιάστηκε και υλοποιήθηκε σύστημα που:

- Αναλύει το περιεχόμενο των tweets και τα χαρακτηριστικά των χρηστών
- Δοκιμάζει και συγκρίνει διάφορα μοντέλα κατηγοριοποίησης
- Εκπαιδεύει το βέλτιστο μοντέλο κατηγοριοποίησης των χρηστών
- Χρησιμοποιεί το μοντέλο αυτό για την κατηγοριοποίηση λογαριασμών στο Twitter
- Καθορίστηκε ένα σύνολο λογαριασμών που είναι spam για να διαμορφωθεί μία black list.

Επιπλέον αναπτύχθηκε web εφαρμογή που επιτρέπει:

- Ο χρήστης κατηγοριοποιεί λογαριασμούς Twitter, ανιχνεύει spammers σε friends και followers
- Ο χρήστης να χαρακτηρίζει λογαριασμούς, ώστε τα δεδομένα αυτά να χρησιμοποιούνται για την περεταίρω εκπαίδευση του μοντέλου.

## *8.2 Μελλοντικές επεκτάσεις*

Το σύστημα αυτό θα μπορούσε σε μελλοντική του επέκταση να βελτιωθεί σημαντικά τόσο ως προς την απόδοση όσο και ως προς την ποικιλία των υπηρεσιών που παρέχει.

Θα μπορούσε να χρησιμοποιηθεί πιο σύγχρονο και μεγαλύτερο σύνολο δεδομένων για την εκπαίδευση του μοντέλου με βάση το Twitter Stream. Ωστόσο αυτό θα προϋπέθεται πολύ περισσότερο χρόνο για δύο λόγους:

- Για τη συλλογή τους από το API
- Για την αναμονή μέχρι την ανανέωση των Blacklists (SURBL,URIBL, Google Safe Browsing)

Ένα άλλο ενδιαφέρον πεδίο επέκτασης θα ήταν ο εμπλουτισμός της ανάλυσης των χαρακτηριστικών των spammers με επιπλέον χαρακτηριστικά που θα εξάγονται από τις σχέσεις μεταξύ τους

- Spammers που έχουν φίλους ή followers επίσης spammers
- Οι φωτογραφίες που χρησιμοποιούν οι spammers και κατά πόσο μοιάζουν μεταξύ τους
- Αν οι ώρες που δημοσιεύουν spam tweets είναι ασυνήθιστες για τη δεδομένη χώρα
- Αν οι spammers προτιμούν ή όχι τα trending hashtags
- Αν προτιμούν να έχουν σχέσεις με συγκεκριμένες ομάδες ανθρώπων (μαθητές, νέους, ηλικιωμένους, φοιτητές, κλπ)

Επιπλέον θα μπορούσαν να χρησιμοποιηθούν άλλες πλατφόρμες κατηγοριοποίησης όταν θα αποκτήσουν μεγαλύτερη ποικιλία και μάλιστα σε ανανεώσιμους αλγόριθμους. Πχ το Apache Mahout.

Τέλος πολύ σημαντικό είναι να γίνει διασταύρωση της απόδοσης του μοντέλου που αναπτύχθηκε με δεδομένα που ήδη υπάρχουν όπως πχ τα δεδομένα του ΙΠΣΥ με χειροκίνητο τρόπο και όχι με πρόγραμμα.

# 9

# *Βιβλιογραφία*


[1] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, San Jose, CA, USA, 2007.

[2] Balachander Krishnamurthy, Phillipa Gill, and Martin Arlitt. A few chirps about twitter. In *Proceedings of the First Workshop on Online Social Networks*, Seattle, WA, USA, 2008.

[3] Sarita Yardi, Daniel Romero, Grant Schoenebeck, and Danah Boyd. Detecting spam in a twitter network. *First Monday*, 15(1), January 2010.

[4] Nigel Williams, Sebastian Zander, Grenville Armitage. A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. Centre for Advanced Internet Architectures (CAIA) Swinburne University of Technology Melbourne, Australia

[5] Balachander Krishnamurthy, Phillipa Gill, and Martin Arlitt. A few chirps about twitter. In *Proceedings of the First Workshop on Online Social Networks*, Seattle, WA, USA, 2008.

[6] Sarita Yardi, Daniel Romero, Grant Schoenebeck, and Danah Boyd. Detecting spam in a twitter network. *First Monday*, 15(1), January 2010.

[7] Zi Chu, Steven Gianvecchio and Haining Wang. Who is Tweeting on Twitter: Human, Bot, or Cyborg? *ACSAC* '10, , Austin, Texas USA, Dec. 6-10, 2010

[8] Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A bayesian approach to filtering junk e-mail. In: AAAI Workshop on Learning for Text Categorization (1998).

[9] S. B. Kotsiantis.Supervised Machine Learning: A Review of Classification Techniques. Department of Computer Science and Technology University of Peloponnese, Greece.

[10] Shamanth Kumar,Fred Morstatter,Huan Liu. Twitter Data Analytics. (2013)

[11] Aaditya Desai, Dr. Sunil Rai. Analysis of Machine Learning Algorithms using WEKA. NMIMS University.

[12] Chris Grier, Kurt Thomas, Vern Paxson, Michael Zhang. @spam: The Underground on 140 Characters or Less. University of California, Berkeley, University of Illinois, Champaign-Urbana.

[13] Alex Hai Wang. Detecting Spam Bots in Online Social Networking Sites: A Machine Learning Approach. College of Information Sciences and Technology The Pennsylvania State University

[14] Zi Chu, Steven Gianvecchio, Haining Wang, Sushil Jajodia. Who is Tweeting on Twitter: Human, Bot, or Cyborg? Department of Computer Science The College of William and Mary, Center for Secure Information Systems George Mason University.