Naive Bayes Classifier

Οι τάξεις που αφορούν το συγκεκριμένο ερώτημα βρίσκονται στο φάκελο Naive Bayes. Με την τάξη FindAttributes υπολογίζουμε το Information Gain κάθε λέξης και με την τάξη NaiveBayes βρίσκουμε σε ποια κατηγορία βρίσκεται ένα κείμενο. Η τάξη Separate χωρίζει ένα Data Set στα εξείς τμήματα : a) το 60% αποτελεί το training set, b) το 20% το evaluation set, c) το υπόλοιπο 20% το validation. Επίσης χωρίζει το training test σε 10%,20%,....,100% .

ΠΡΟΣΟΧΗ : Ο αλγόριθμος NaiveBayes έχει υλοποιηθεί έτσι ώστε να κατατάσσει ένα κείμενο σε δύο κατηγορίες. Κατά την υλοποίηση χρησιμοποιούμε τις λέξεις ham και spam αλλά μπορεί να λειτουργήσει για οποιαδήποτε δυάδα κατηγοριών. Στην τάξη FindAttributes δίνουμε στις μεταβλητές "cat1" και "cat0" τα ονόματα των κατηγοριών. Επίσης σε κάθε data set είτε είναι αξιολόγησης είτε είναι εκπαίδευσης, η πρώτη λέξη πρέπει να είναι η κατηγορία του κειμένου και κάθε κείμενο πρέπει να καταλαμβάνει ακριβώς μια σειρά στο αρχείο. Αυτό γίνεται για να ξέρουμε κατά την εκπαίδευση την κατηγορία του κειμένου και κατά την αξιολόγηση το αν η απόκριση του αλγορίθμου είναι σωστή.

Λεπτομέριες Υλοποίησης :

Στην τάξη FindAttributes, δίνουμε στη μεταβλητή "path" το path του training set το οποίο θα χρησιμοποιήσουμε κατά την εκπαίδευση. Η μεταβλητή " maxAttributes " περιέχει το πλήθος των ιδιοτήτων που θα χρησιμοποιήσουμε. Αν π.χ. έχει την τιμή 100 τότε θα χρησιμοποιήσουμε τις 100 ιδιότητες με το μεγαλύτερο information gain. Στη συνέχεια, ρυθμίζουμε τη μεταβλητή path της τάξης NaiveBayes, δίνοντας ως τιμή το path των data των οποίων θέλουμε να μάθουμε την κατηγορία τους (validation set/evaluation set).

Σημαντικοί μέθοδοι :

Τάξη FindAttributes :
- public void search() : Επεξεργάζεται το training set βρίσκοντας τα πόσα κείμενα ανήκουν σε κάθε μια από τις δύο κατηγορίες και τις πιθανότητες P(C=1) και P(C=0).

- Public void IG(): υπολογίζει το information gain κάθε λέξης.

- public void sort(ArrayList <String> array, int n) : Ταξινομεί τις ιδιότητες με βάση το information gain της καθεμίας.

- public void changeAttr() : Αποθηκεύει στο ArrayList "attributes" τις k ιδιότητες με το μεγαλύτερο information gain.

Τάξη NaiveBayes :

- public void newMessage : Επεξεργάζεται ένα κείμενο προς αξιολόγηση.

- public double spamProb : Επιστρέφει την πιθανότητα του να είναι στην κατηγορία spam ένα κείμενο.

- Public double hamProb : Επιστρέφει την πιθανότητα του να είναι στην κατηγορία ham ένα κείμενο.

- Public boolean classify() : καλεί τις δύο παραπάνω συναρτήσεις και επιστρέφει false αν η πιθανότητα να είναι spam είναι μεγαλυτερη από τη πιθανότητα να είναι ham, αλλιώς επιστρέφει ham.

Περαιτέρω λεπτομέριες υπάρχουν στα σχόλια.

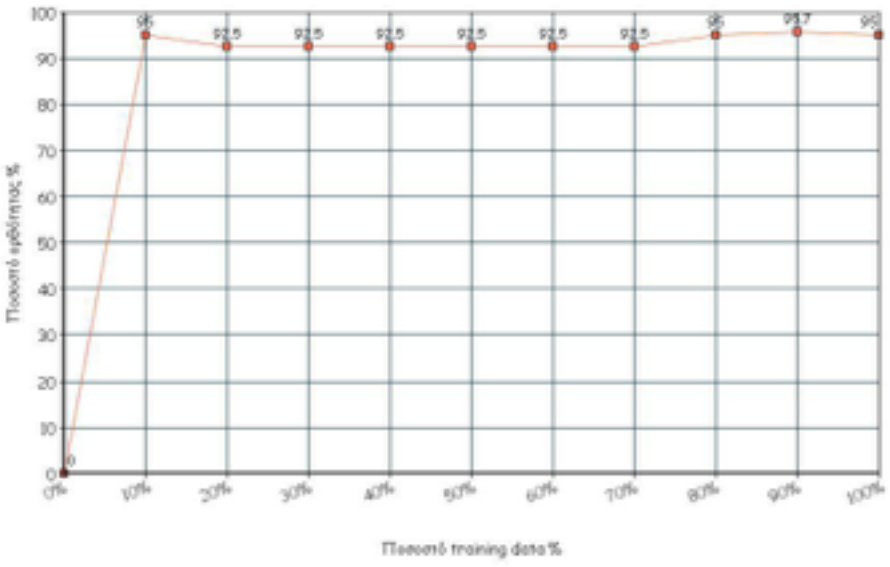
ΠΕΙΡΑΜΑΤΑ

Αρχικά δοκιμάζουμε τον αλγόριθμο με δεδομένα επικύρωσης ώστε να βρούμε ποιο είναι το καλύτερο πλήθος ιδιοτήτων. Όπως φένεται στον παρακάτω πίνακα, υπολογίζουμε τα σφάλματα που γίνονται έχοντας διαφορετικό πλήθος ιδιοτήτων. Στην αριστερή στήλη είναι το ποσοστό του Training Set. Ξεκιναμε απο 10% και φτάνουμε μέχρι 100% που είναι όλο το Training Set. Κάθε στήλη έχει τα σφάλματα που γίνανε με πληθος ιδιοτήτων 50 κτλ.. Το Training Set που χρησιμοποιήσαμε έχει 400 παραδείγματα ενώ τα δεδομένα επικύρωσης περιλαμβάνουν 132 παραδείγματα.

	50	80	120	160	200	250	300
10%	17	17	17	17	17	17	17
20%	17	17	17	17	17	17	17
30%	9	12	14	16	14	14	14
40%	8	6	11	13	13	14	14
50%	11	10	10	10	12	14	16
60%	11	11	10	10	13	14	14
70%	9	10	10	10	10	11	13
80%	9	10	10	9	10	11	12
90%	10	11	10	7	8	9	12
100%	6	9	11	8	9	10	12

Μπορούμε να συμπεράνουμε λοιπόν πως τα λιγότερα σφάλματα τα έχουμε με 50 ιδιότητες.

Αφού επιλέξουμε τον αριθμό των ιδιοτήτων που θα χρησιμοποιούμε, θα δοκιμάσουμε τον αλγόριθμο με με τα δεδομένα αξιολόγησης τα οποία είναι διαφορετικά από τα δεδομένα εκπαίδευσης και επικύρωσης. Παρακάτω βρίσκεται το διάγραμμα ορθότητας. Ο άξονας Y δείχνει το ποσοστό ορθότητας και ο άξονας X το ποσοστό των Training Data.



Τώρα θα δοκιμάσουμε τον αλγόριθμο αξιολογώντας τον με τα δεδομένα εκπαίδευσης. Στο γράφημα που ακολουθεί βλέπουμε τα ποσοστά ορθότητας.