ΤΕΧΝΗΤΗ ΝΟΥΜΟΣΥΝΗ

ΑΝΑΦΟΡΑ ΕΡΓΑΣΙΑΣ 2

**ΧΡΙΣΤΙΝΑ ΑΙΜΙΛΙΑ ΦΛΑΣΚΗ 3180195**

**ΓΕΩΡΓΙΑ ΨΥΧΑ 3190225**

Μέθοδοι-> ...

Μεταβλητές-> …

**NAIVE BAYES**

Για την επίλυση του προβλήματος αυτού ξεκινήσαμε καλώντας την μέθοδο naiveBayes(n, m) με n=1 και m=4000. (μεταβλητες που χρησιμοποησαμε με βαση το εργαστηριο που μας ειπε ότι οι 4000 λεξεις είναι καλες ώστε να παρουμε ένα καλο ογκο δεδομενων.

* **naiveBayes:** Η naiveBayes καλεί πρώτα την train(n,m) (με ορίσματα τα ίδια n και m που είχε και η naiveBayes), στέλνει τα αποτελέσματα της train στην naiveBayes\_Test και από αυτή παίρνει τα αποτελέσματα των TP, TN, FP, FN με τα οποία υπολογίζει τις τιμές (για θετικά και αρνητικά) Accuracy,Precision, Recall, F1 τα οποία και εκτυπώνει.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, απόδειξη

Περιγραφή που δημιουργήθηκε αυτόματαΟ υπολογισμός των TP, TN, FP, FN έγινε με την εφαρμογή των τύπων:

* **train:** Η train διαβάζει τα δεδομένα από το αρχείο labeledBow.feat που βρίσκεται στον φάκελο data, με την εντολή read\_csv().

Δημιουργεί vectors για τα δεδομένα που πήρε, τα οποία βρίσκονται στην reviews,για την δημιουργία των vectors καλεί την create\_vectors(dataset, i, n, m) (όπου σαν dataset στέλνει τα δεδομένα που πήρε από το αρχείο) εκχωρώντας μέσα στην μεταβλητή vectors αυτά που επιστρέφει η create\_vectors.

Χρησιμοποιεί το 70% των vectors που έχει καταχωρήσει τα δεδομένα και με αυτά υπολογίζει τις πιθανότητες (P(X|C)) και (P(C)) καθώς επιστρέφει τις μεταβλητές που αντιστοιχούν στις πιθανότητες αυτές, pcx1, pc1 για τα θετικά και pcx0, pc0 για τα αρνητικά αντίστοιχα. Για τον υπολογισμό των (P(X|C)) καλείται η voc\_propability(vectors).

* Εικόνα που περιέχει κείμενο

  Περιγραφή που δημιουργήθηκε αυτόματα**voc\_propability**: Η voc\_propability δέχεται ως όρισμα τους vectors, σκοπός της είναι να υπολογίσει την πιθανότητα (P(X|C)) για Χ=1(θετικό) και για Χ=0 (αρνητικό). Δημιουργεί και επιστρέφει 2 λίστες, μια για τα θετικά και μια για τα αρνητικά στις οποίες προσθέτει τα αποτελέσματα του υπολογισμού της πιθανότητας.

(από την διαφάνεια 16)

* **create\_vectors:** H create\_vectors είναι αυτή που δημιουργεί το διάνυσμα 0,1 για κάθε review ανάλογα αν είναι αρνητικό ή θετικό. Χρησιμοποιεί την εντολή split() για να διαχωρίσει κάθε review σε λέξεις.

Ελέγχει αν η βαθμολογία είναι πάνω από 5, αν ναι τότε προστίθεται στα θετικά αλλιώς στα αρνητικά (το πρώτο token κάθε review είναι η βαθμολογία του, για αυτό int(tokens[0]) > 5). Μέχρι την m-ιοστή λέξη ελέγχει αν ισχύει “λέξη”:, αν ισχύει τότε προσθέτει την λέξη στα θετικά, αλλιώς στα αρνητικά. Στο τέλος κάθε review προσθέτει το binary\_vector στο x\_binary, το οποίο και επιστρέφει(αυτό αφορά ολόκληρο το dataset)

Η μορφη των δεδομενων είναι αυτή :Εικόνα που περιέχει κείμενο, υπαίθριος, πληκτρολόγιο

Περιγραφή που δημιουργήθηκε αυτόματα

Οπου αρχικα είναι το νουμερο της κριτικης και ακολουθειτε από την αξιολογηση πχ 9>5 αρα θετικο που μας προσιοριζει τον τυπο της στην συνεχεια εχει την μορφη των λεξεων “λέξη”:, που βρισκει αν η λεξη βρισκκεται στα δεδομενα η όχι .

* **naiveBayes\_Test**: Η naiveBayes\_Test διαβάζει τα δεδομένα από το αρχείο TlabeledBow.feat που βρίσκεται στον φάκελο data, με την εντολή read\_csv().

Δημιουργεί vectors για τα δεδομένα που πήρε, με την create\_vectors(dataset, i, n, m) (όπου σαν dataset στέλνει τα δεδομένα που πήρε από το αρχείο). Υπολογίζει τα TP (true positive), TN (true negative), FP (false positive), FN (false negative) και τα επιστρέφει.

ID3

Για την επίλυση του προβλήματος αυτού ξεκινήσαμε καλώντας την μέθοδο ID3(n, m) με n=1 και m=4000. (μεταβλητες που χρησιμοποησαμε με βαση το εργαστηριο που μας ειπε ότι οι 4000 λεξεις είναι καλες ώστε να παρουμε ένα καλο ογκο δεδομενων.

* informationGain(vectors, features): υπολογίζει το informationgain για τα χαρακτηριστικα αυτά . Για την μεθοδο αυτή στηριχτηκαμε σην information gain του εργαστηριου 7 οπου αρχικα βρισκει το πληθος των αρνητικων(0) και θετικων(1) αξιολογησεων αναλογα με την τιμη της μεταβλητης “C” αναλογα αν είναι 0 η 1 αντιστοιχα και στην συνεχεια με τους καταλληλους τυπους υπολογιζει τα :
  + - # P(X=1) --> prob. of X=1
    - # P(X=0) --> 1 - P(X=1)
    - # P(C=1|X=1)
    - # P(C=0|X=1) = 1 - P(C=1|X=1)
    - # P(C=1|X=0)
    - # P(C=0|X=0) = 1 - P(C=1|X=0)
    - # H(C=1|X=1)
    - # H(C=1|X=0)

Χρησιμοποιώντας για τους υπολογισμούς αυτους την συναρτηση calc και δημιορυγει ένα dictionairy IG{} που περιεχει το informationgain κάθε χαρακτηριστικου.

* def twoCEntropy(cProb): def twoCEntropy(cProb): είναι υπευθυνα για τον υπολογισμο της εντροπιας που δειχνει ποσο αβεβαιοι ειμαστε για την τιμη της C .εδω εχουμε δυο χαρακτηριστικα αρα ο τυπος είναι :Εικόνα που περιέχει κείμενο

  Περιγραφή που δημιουργήθηκε αυτόματα
* def create\_tree(vectors, features, percentage\_class, depth): δημιουργει το decision tree βασιστήκαμε στον ψευδοκωδικα των διαφανειών και σε πήγες στο ιντερνετ κοινως :

αρχικα κανουμε τους απαραίτητους ελέγχους σε περίπτωση που το length των χαρακτηριστικων είναι 0 και επιστρέφουμε την προεπιλεγμένη κατηγορία στην συνέχεια αν μια κατηγορία είναι 100% δηλ υπάρχει μονο αυτή πράγμα που το ελέγχουμε μέσα από defcounter (examples):που απλα υπολογιζει το αθροισμα τα κάθε κατηγοριας τοτε επιστρεφουμε την κατηγορια αυτή ΔΙΑΦΟΡΕΤΙΚΑ δημιουργουμε κενο δεντρο και βρισκουμε το καλυτερο χαρακτηριστικο αναλογα με το informationgain και το προσθετουμε στην ριζα του δεντρου μετα για κάθε δυνατη τιμη της καλυτερης δημιουγουμε υποδεντρο και το προσθετουμε στο κανονικο δεντρο .

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα