

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΩΝ ΜΕ

C++

(3^η Εργασία)



ΜΠΟΓΔΑΝΟΣ ΜΙΧΑΗΛ (3100123)
ΜΑΓΚΟΣ ΡΑΦΑΗΛ-ΓΕΩΡΓΙΟΣ(3100098)

Αρχείο classifier.h

Στο αρχείο αυτό έχουμε την δήλωση της αφηρημένης κλάσης Classifier. Η τάξη είναι αφηρημένη διότι περιλαμβάνει τις δημόσιες και καθαρά εικονικές μεθόδους :

- **virtual void train(const InstancePool& trPool) = 0** Μέθοδος εκπαίδευσης ενός ταξινομητή. Το όρισμα της είναι μια αναφορά σε μια συλλογή μηνυμάτων , ενώ δεν επιστρέφει τίποτα.
- **virtual bool classify(const Instance& inst) const = 0** Μέθοδος ταξινόμησης ενός μηνύματος. Το όρισμα της είναι μια αναφορά σε ένα μήνυμα , ενώ δεν επιτρέπεται να μεταβάλει τα μέλη του αντικειμένου classifier (const).
- **virtual ~Classifier() {}** Εικονικός καταστροφέας.

Αρχείο classifierevaluator.h

Στο αρχείο αυτό έχουμε την δήλωση της κλάσης ClassifierEvaluator :

Ιδιωτικά μέλη της κλάσης ClassifierEvaluator:

- **float truePositives** : Τα instance που σωστα κατατάχθηκαν ως ανεπιθύμητα(spam).
- **float trueNegatives** : Τα instance που σωστα κατατάχθηκαν ως επιθυμητά(ham).
- **float falsePositives**: Τα instance που ήταν επιθυμητά(ham) όμως κατατάχθηκαν ως ανεπιθύμητα(spam).
- **float falseNegatives** : Τα instance που ήταν ανεπιθύμητα(spam) όμως κατατάχθηκαν ως επιθυμητά(ham).

(Οι παραπάνω ορίστηκαν ως float για αποφυγή απώλειας ακρίβειας)

- **unsigned total** : Ο συνολικός αριθμός μηνυμάτων προς κατάταξη.

Δημόσια μέλη της κλάσης ClassifierEvaluator:

- **ClassifierEvaluator(Classifier& classifier, const InstancePool& trainingPool, const InstancePool& testPool)**: Κατασκευαστής. Πρώτα αρχικοποιούμε τις float μεταβλητές σε 0 και ακολούθως αναθέτουμε το μέγεθος του testPool στην total. Επίσης ορίζουμε δύο bool μεταβλητές c και d , που περιλαμβάνουν την απόφαση του ταξινομητή και την πραγματική κατηγορία του τρέχοντος μηνύματος αντίστοιχα. Σύμφωνα με τις τιμές που θα προκύψουν από την κλήση της classify και της getCategory στο προς εξέταση μήνυμα, αυξάνουμε την κατάλληλη μεταβλητή float.

- **float getAccuracy() const** : Επιστρέφει την τιμή του λόγου $(\text{truePositives} + \text{trueNegatives}) / \text{total}$.
- **float getPrecision() const** : Επιστρέφει την τιμή του λόγου $\text{truePositives} / (\text{truePositives} + \text{falsePositives})$ μόνο εάν ο παρονομαστής είναι διάφορος του 0. Εάν είναι 0 τότε επιστρέφεται το 0.
- **float getRecall() const** : Επιστρέφει την τιμή του λόγου $\text{truePositives} / (\text{truePositives} + \text{falseNegatives})$ μόνο εάν ο παρονομαστής είναι διάφορος του 0. Εάν είναι 0 τότε επιστρέφεται το 0.

Αρχείο `baselineclassifier.h`

Στο αρχείο αυτό έχουμε την δήλωση της κλάσης `BaselineClassifier` , μια κλάση που κληρονομεί την κλάση `Classifier`.

Ιδιωτικά μέλη της κλάσης `BaselineClassifier`:

- `bool trainCalled` : Μεταβλητή τύπου `bool` που ελέγχει εάν έχει γίνει κλήση της μεθόδου εκπαίδευσης.
- `unsigned ham,spam` : Ο συνολικός αριθμός μηνυμάτων `ham` και `spam` που βρέθηκε στα μηνύματα εκπαίδευσης.

Δημόσια μέλη της κλάσης `BaselineClassifier`:

- **`BaselineClassifier()`** : Κατασκευαστής όπου αρχικοποιούμε την `bool trainCalled` σε `false` και τις `unsigned` μεταβλητές σε 0.
- **`void train(const InstancePool& training)`**: Μέθοδος που ουσιαστικά μετράει πόσα επιθυμητά και πόσα ανεπιθυμητά μηνύματα περιέχονται στο σύνολο μηνυμάτων εκπαίδευσης(όρισμα `training`) και τοποθετούμε αυτή την πληροφορία στην αντίστοιχη `unsigned` μεταβλητή `spam` ή `ham`.
- **`bool classify(const Instance& inst) const`** : Ο ταξινομητής τοποθετεί το όρισμα `inst` στην κατηγορία που πλειοψηφεί στο σύνολο των μηνυμάτων εκπαίδευσης(πληροφορία που βρήκαμε στην `train`).Εάν δεν υπήρξε κλήση της `train` τότε εμφανίζεται προειδοποίηση και το `inst` κατατάσσεται ως επιθυμητό(τιμή `false`).

Αρχείο `knnclassifier.h`

Στο αρχείο αυτό έχουμε την δήλωση της κλάσης `KNNClassifier` , μια κλάση που κληρονομεί την κλάση `Classifier`.

Ιδιωτικά μέλη της κλάσης `KNNClassifier`:

- **`InstancePool memory`**: Μεταβλητή τύπου `InstancePool` που χρησιμοποιείται για τον υπολογισμό των ευκλίδειων αποστάσεων μεταξύ του προς κατατάταξη `Instance` και όλων των μηνυμάτων του `memory`.
- `bool trainCalled` : Μεταβλητή τύπου `bool` που ελέγχει εάν έχει γίνει κλήση της μεθόδου εκπαίδευσης.
- `unsigned _kIn` : Μεταβλητή `unsigned` που θα καθορίζει πόσοι θα πρέπει να θεωρούνται οι κοντινότεροι γείτονες.

➤ **void sort(float *num, unsigned size, bool *inst) const:** Μέθοδος που ταξινομεί δύο πίνακες ίσου μεγέθους size. Χρησιμοποιείται ταξινόμηση φυσαλίδας στον πίνακα num. Αλλαγές στον float αλλάζουν και τις αντίστοιχες θέσεις του inst.

Δημόσια μέλη της κλάσης KNNClassifier:

➤ **KNNClassifier(unsigned short kln=5) :** Κατασκευαστής όπου δίνουμε την τιμή kln στην μεταβλητή _kln και την τιμή false στην μεταβλητή trainCalled. Εάν δεν δοθεί τιμή ως όρισμα τότε χρησιμοποιείται η προκαθορισμένη τιμή 5.

➤ **static float distance(const Instance& inst1, const Instance& inst2):** Μέθοδος υπολογισμού της Ευκλίδειας απόστασης των αντικειμένων inst1 και inst2. Χρησιμοποιούμε ένα map όπου με χρήση ενός βρόγχου αποθηκεύουμε για κάθε keyword του inst1 ως κλειδί το keyword ID του και ως τιμή την συχνότητα εμφάνισης του. Με χρήση ενός βρόγχου for στο αντικείμενο inst2 ανακτάμε με το keyword ID το frequency που βρέθηκε πριν και εκτελούμε την πράξη της αφαίρεσης (προσέχοντας να αφαιρούμε το μεγαλύτερο από το μικρότερο γιατί χειριζόμαστε unsigned). Σε περίπτωση όμως που το map δεν περιείχε πριν αυτό το keyword ID τότε προφανώς έχει frequency 0.

➤ **void train(const InstancePool& training):** Μέθοδος όπου με χρήση του τελεστή = (που έχει υπερφορτωθεί στην εργασία 2) κάνουμε το memory ίσο με το όρισμα training. Επίσης θέτουμε την μεταβλητή trainCalled σε true εφόσον κλήθηκε η μέθοδος εκπαίδευσης.

➤ **bool classify(const Instance& inst) const :** Αρχικά γίνεται έλεγχος για προηγούμενη κλήση της train. Εάν δεν έχει γίνει κλήση τότε εμφανίζεται προειδοποίηση και το inst κατατάσσεται ως επιθυμητό (επιστρέφεται false). Εάν έχει γίνει εκπαίδευση, τότε δεσμεύουμε χώρο δυναμικά για δύο πίνακες float και bool μεγέθους όσα και τα μηνύματα στο memory. Ακολουθώντας με ένα βρόγχο for , υπολογίζουμε την Ευκλίδεια απόσταση μεταξύ του i-οστού μηνύματος του memory και του ορίσματος inst. Επίσης αποθηκεύουμε στην i-οστή θέση του bool πίνακα την κατηγορία του τρέχοντος μηνύματος του memory. Με κλήση της sort, ταξινομούμε τον πίνακα float με τις αποστάσεις (κάνοντας και τις κατάλληλες μεταθέσεις στον bool) και βρίσκουμε τα _kln πρώτα στοιχεία του πίνακα float (δηλαδή τα _kln κοντινότερα) και βρίσκουμε πόσα από αυτά είναι spam (με μια μεταβλητή μετρητή). Εάν τα spam πλειοψηφούν στα _kln κοντινότερα τότε επιστρέφουμε true (απόφαση ότι είναι spam το inst), αλλιώς false.

Αρχείο nbclassifier.h

Στο αρχείο αυτό έχουμε την δήλωση της κλάσης NaiveBayesClassifier , μια κλάση που κληρονομεί την κλάση Classifier.

Ιδιωτικά μέλη της κλάσης NaiveBayesClassifier:

- **float Pspam,Pham:** Μεταβλητές τύπου float που δείχνουν την πιθανότητα ένα μήνυμα στην συλλογή μηνυμάτων να είναι spam ή ham αντίστοιχα.
- **bool trainCalled :** Μεταβλητή τύπου bool που ελέγχει εάν έχει γίνει κλήση της μεθόδου εκπαίδευσης.
- **unsigned Mham,Mspam :** Μεταβλητές unsigned που δείχνουν το πλήθος των spam και των ham αντίστοιχα, στην συλλογή μηνυμάτων.
- **unsigned map<unsigned,double> PTspam,PTham :** Map που δείχνουν την πιθανότητα να υπάρχει η λέξη κλειδί με ID t στα μηνύματα spam ή στα μηνύματα ham ,της συλλογής μηνυμάτων εκπαίδευσης, αντίστοιχα.
- **set <unsigned> keywords:** Set που θα περιέχει τα keyword IDs όλων των keywords που εμφανίζονται στα μηνύματα εκπαίδευσης.

Δημόσια μέλη της κλάσης NaiveBayesClassifier:

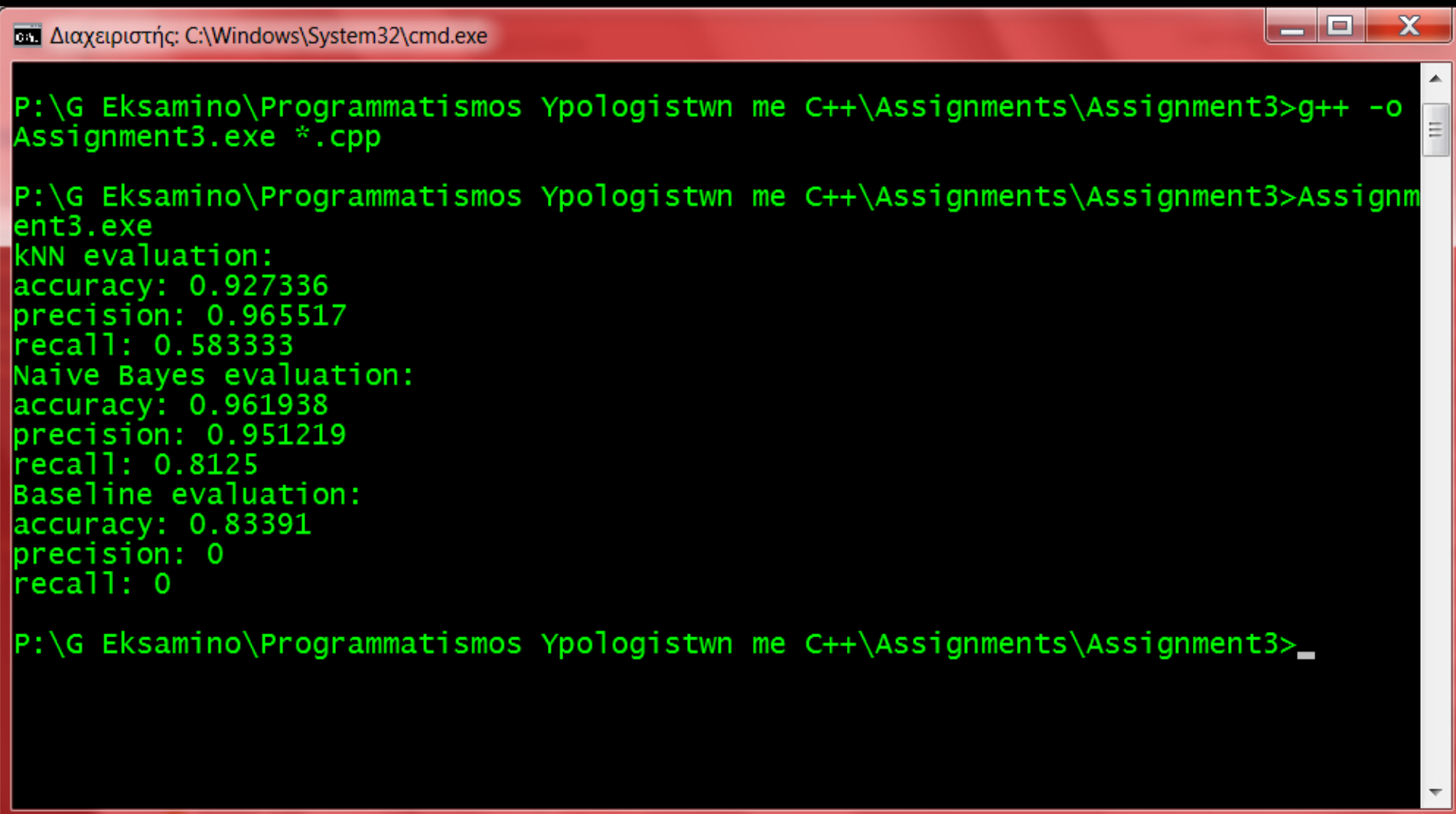
➤ **NaiveBayesClassifier()** : Κατασκευαστής όπου αρχικοποιούμε την trainCalled σε false και τις unsigned και float μεταβλητές σε 0.

➤ **void train(const InstancePool& training):** Μέθοδος εκπαίδευσης που με χρήση ενός βρόγχου for , ανακτά τα μηνύματα στο training , ελέγχει την κατηγορία του μηνύματος , αυξάνει τον κατάλληλο μετρητή(Mham για ham και Mspam για spam) και τοποθετεί στο κατάλληλο map (PTspam/ham) το keyword ID για να βρούμε αρχικά το MTHam/Spam(αρχικά το map περιλαμβάνει αυτά).Ακολούθως υπολογίζονται οι Pspam(πλήθος spam Mham/συνολικά μηνύματα) και Pham(1-Pspam).Με χρήση επαναλήπτη διασχίζουμε κάθε ένα από τα δύο map και τοποθετούμε πλέον τα Ptham/spam με χρήση των τύπων.

➤ **bool classify(const Instance& inst) const :** Ο ταξινομητής τοποθετεί το όρισμα inst στην κατηγορία που είναι πιο πιθανό να ανήκει με βάση τα keywords του.Εάν δεν υπήρξε κλήση της train τότε εμφανίζεται προειδοποίηση και το inst κατατάσσεται ως επιθυμητό(τιμή false). Ορίζουμε τα δύο μέλη της ανισότητας ως left και right και υπολογίζουμε και αναθέτουμε την $\log(P_{spam})$ και την $\log(P_{ham})$ αντίστοιχα.Το σκεπτικό είναι ότι για όλα τα ID του set ψάχνουμε εάν υπάρχουν τα αντίστοιχα keyword στο προς εξέταση μήνυμα.Εάν υπάρχει τότε προσθέτουμε το $\log(P_{tspam})$ στο left και το $\log(P_{Tham})$ στο right.Εάν δεν υπάρχει τότε προσθέτουμε το $\log(1-P_{tspam})$ στο left και το $\log(1-P_{Tham})$ στο right.Σε κάθε περίπτωση δεν ξεχνάμε πώς μια λέξη κλειδί μπορεί να εμφανίζεται είτε μόνο σε spam ,είτε μόνο σε ham ,είτε και στα 2.Γι'αυτό

Αναζητάμε το ID στο map με τα PTsram. Εάν βρεθεί τότε υπολογίζουμε όπως αναλύθηκε παραπάνω. Εάν δεν βρεθεί τότε θεωρούμε ότι το πλήθος εμφάνισης στα sram είναι 0 άρα $MTsram = (1+0)/(2+Msram)$. Με όμοιο σκεπτικό υπολογίζουμε και την περίπτωση που δεν υπάρχει το ID στο Ptham map.

Παράδειγμα Χρήσης



```
Διαχειριστής: C:\Windows\System32\cmd.exe

P:\G Eksamino\Programmatismos Υπολογιστων με C++\Assignments\Assignment3>g++ -o
Assignment3.exe *.cpp

P:\G Eksamino\Programmatismos Υπολογιστων με C++\Assignments\Assignment3>Assignm
ent3.exe
kNN evaluation:
accuracy: 0.927336
precision: 0.965517
recall: 0.583333
Naive Bayes evaluation:
accuracy: 0.961938
precision: 0.951219
recall: 0.8125
Baseline evaluation:
accuracy: 0.83391
precision: 0
recall: 0

P:\G Eksamino\Programmatismos Υπολογιστων με C++\Assignments\Assignment3>_
```