Τεχνητή Νοημοσύνη

3η εργασία

Κώστας Κοπανίδης p3130098

Χρύσα Μαυράκη p3130128

Λευτέρης Χατζηαράπης p3130255

Αλγόριθμος ID3:

Αρχικά, για τον αλγόριθμο id3 επεξεργάζομαι τα δεδομένα ώστε να τα μετατρέψω σε μορφή που βολεύει τον αλγόριθμο Id3.Διαβάζω τα δεδομένα και τα αποθηκεύω σε μια λίστα απο λίστες, δηλαδή μια λίστα απο όλες τις γραμμές του αρχείου. Έτσι επιλέγω ανεξαρτήτως των δεδομένων να τα μετατρέψω σε διακριτούς φυσικούς αριθμούς ξεκινώντας απο το 0 (ανεξάρτητους σε κάθε στήλη, δηλαδή κάθε διαφορετική κατηγορία και η απάντηση). [Μ'αυτόν τον τρόπο θα μπορέσω να επεξεργαστώ και dataset με φυσικούς αριθμούς προσαρμόζοντας τους σε φυσικούς αριθμούς.]

- Στη συνάρτηση read_data απλώς διαβάζω το αρχείο,ανακατεύω τα περιεχόμενα του ώστε όταν τα χωρίσω σε train και test να μην χάσω χρήσιμα παραδείγματα. Και τέλος, αποθηκεύω τα δεδομένα του ως λίστα απο λίστες.
- Στη συνάρτηση split_train_test δέχομαι το ποσοστό των δεδομένων που θέλω να κρατήσω για εκπάιδευση και την λίστα των δεδομένων και επιστρέφω δύο μικρότερες λίστες μια για εκπαίδευση και μια για δοκιμές.
- Στη συνάρτηση datahandler κάνω αυτή τη δουλειά, παίρνω τα δεδομένα (ήδη ως λίστα απο λίστες) και επιστρέφω
    1. Το index_list, το οποίο έχει ακριβώς την μορφή που είχαν και τα δεδομένα, αντικατεστημένα απο φυσικούς αριθμούς πλεον
    2. Το index_mapping το οποίο παίρνει σαν όρισμα τον αριθμό της στήλης και την τιμή των δεδομένων εκεί και μας δείνει τον αριθμο στον οποίο μετατρέπεται το δεδομένο.
    3. Το value_to_index το οποίο δέχεται τον αριθμό της στήλης και την αριθμητική τιμή και επιστρέφει το δεδομένο όπως ήταν αρχικά. (Ακριβώς το ανάποδο απο το index_mapping δηλαδή).

4. Το Possible_column_values στο οποίο αποθηκεύουμε (χωρίς διπλότυπα) τις πιθανές τιμές που μπορεί να έχει κάθε στήλη (πχ 0,1,2) (λίστα απο σετ)

Πολλές φορές όμως, ιδιαίτερα αν στο train_set δεν υπάρχουν κάποιοι συνδιασμοί τιμών που ενδέχεται να υπάρχουν στο test_set, ο ασφαλέστερος τρόπος για να κάνουμε το Mapping είναι απο το αρχείο .names. Για αυτόν τον λόγο έχω φτιάξει και μια συνάρτηση data_mapping, Η οποία επιστρέφει τα, index_mapping, value_to_index και possible_column_values. Αν δώσουμε αυτά ως ορίσματα στην datahandler θα μας επιστρέψει μόνο το index_list.

Για την υλοποίηση του αλγορίθμου id3 χρειάζομαι επίσης την εντροπία και το information gain τα οποία απλώς υλοποιώ προγραμματιστικά ακολουθώντας τη θεωρεία.

Για τον Αλγόριθμο ID3 έχω κάνει μια αναδρομική υλοποιησή του, που δέχεται ως ορίσματα τα δεδομένα εκπαίδευσης, τις υπολοιπόμενες κατηγορίες, τις πιθανές τιμές των κατηγοριών, μια προκαθορισμένη κατηγορία και ένα όριο για το πόσο θέλουμε να κατηγοριοποιήσουμε τα παραδείγματα. Αρχικοποιούμε το δεντρο ως ένα dict, το ans_count Κ αυτό ως dict το οποίο θα μετρά πόσες φορές εμφανίζεται η κάθε απάντηση. Ορίζω το answers ως την τελευταία στήλη των δεδομένων και τα max_ig,max_index = 0. Άν έχουμε εξαντλήσει τις διαθέσιμες κατηγορίες ή δεν έχουμε καθόλου παραδείγματα,επιστρέφουμε την προεπιλεγμένη κατηγορία. Ύστερα αρχικοποιούμε το preselected_count = -1 Και το Preselected το ίδιο. Αυτό το κάνουμε για να βρούμε την πιο συχνή κατηγορία αυτή τη στιγμή για να περάσω στα υποδέντρα μας. Μετρώ το πλήθος της κάθε κατηγορίας απάντησης. Υπολογίζω την αρχική εντροπία με μια βοηθητική συνάρτηση. Μετά ελέγχω αν το πλήθος μιας κατηγορίας είναι περισότερο απο ένα threshold των συνολικών αποτελεσμάτων τότε επιστρέφω την κατηγορία αυτή. Αλλιως αν το πλήθος είναι μεγαλύτερο απο της προεπιλεγμένης κατηγορίας θέτω αυτή τη κατηγορία ως προεπιλεγμένη, έτσι θα βρω την πιο συχνή κατηγορία αποτελεσμάτων. Ύστερα με την βοήθεια του IG βρίσκω την επόμενη κατηγορία στην οποία θα ¨σπάσω¨, θα είναι αυτή με το μεγαλύτερο IG, την αφαιρώ απο τις υπολοιπόμενες κατηγορίες ορίζω ως key στο δέντρο αυτή τη κατηγορία, ως value ορίζω ένα νεο dict.
Έπειτα, χωρίζουμε τα δεδομένα σε υποκατηγορίες ανάλογα με τις τιμές κάθε κατηγορίας και για κάθε μια απ'αυτές καλούμε αναδρομικά τον αλγόριθμο ώστε να συμπληρώσουμε το δέντρο.
Τέλος, επιστρέφουμε το δέντρο.

Predict: Το Predict είναι μια αναδρομική συνάρτηση, η οποία ανατρέχει το δέντρο και καταγράφει που θα κατατάξουμε τα δεδομένα αξιολόγησης βάσει του δέντρου που έχουμε φτιάξει. Καταγράφει αυτά τα αποτελέσματα σε δυο λίστες μια με τα Indexes και μια με την τιμή τους σε string όπως ήταν αρχικά.

Οι συναρτήσεις για τα αποτελέσματα:

- Print_accuracy: Στην οποία απλά υπολογίζουμε πόσα δεδομένα κατατάξαμε σωστά και τα διαιρούμε με το σύνολο ώστε να βρούμε τη μέση ακρίβεια και την τυπώνουμε.
- Print_precission_recall: Στην οποία υπολογίζουμε τα precission – recall για κάθε κλάση αποτελεσμάτων ξεχωριστά αλλά και το μέσο όρο τους. Τυπώνουμε αυτά τα αποτελέσματα.

Αποτελέσματα:

- Για αρχείο το οποίο έχουμε σπάσει σε train_data(90%) Και test_data(10%) Με threshold κατάταξης 80%

  Precision for each class
  {0: 0.9834710743801653, 1: 0.8837209302325582, 2: 1.0, 3: 0.42857142857142855}

  Mean precision
  0.823940858296038

  Recall for each class
  {0: 0.9596774193548387, 1: 0.95, 2: 0.3333333333333333, 3: 1.0}

  Mean recall
  0.810752688172043

  Mean accuracy
  0.9364161849710982

  Τα αποτελέσματα του weka για το ίδιο dataset

```
Correctly Classified Instances         124               71.6763 %
Incorrectly Classified Instances        49               28.3237 %
Kappa statistic                          0
Mean absolute error                      0.2235
Root mean squared error                  0.3288
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances              173

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         1          0.717     1         0.835      0.5      unacc
                 0         0          0         0         0          0.5      acc
                 0         0          0         0         0          0.5      good
                 0         0          0         0         0          0.5      vgood
Weighted Avg.    0.717     0.717      0.514     0.717     0.599      0.5

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 124   0   0   0 |   a = unacc
  40   0   0   0 |   b = acc
   3   0   0   0 |   c = good
   6   0   0   0 |   d = vgood
```

- Για αρχείο το οποίο έχουμε σπάσει σε train_data(80%) Και test_data(20%)
  Με threshold κατάταξης 80%

  Precision for each class
  {0: 0.9615384615384616, 1: 0.8928571428571429, 2: 0.6, 3: 0.8461538461538461}

  Mean precision
  0.8251373626373627

  Recall for each class
  {0: 0.9615384615384616, 1: 0.872093023255814, 2: 0.9, 3: 0.6875}

  Mean recall
  0.8552828711985688

  Mean accuracy
  0.9248554913294798

  Τα αποτελέσματα του weka για το ίδιο dataset

```
Correctly Classified Instances         306               88.4393 %
Incorrectly Classified Instances        15                4.3353 %
Kappa statistic                          0.8932
Mean absolute error                      0.0234
Root mean squared error                  0.1529
Relative absolute error                 11.457  %
Root relative squared error             48.6373 %
UnClassified Instances                  25                7.2254 %
Total Number of Instances              346

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.961     0.023      0.991     0.961     0.976      0.97     unacc
                 0.957     0.032      0.892     0.957     0.923      0.868    acc
                 0.75      0.003      0.9       0.75      0.818      0.78     good
                 1         0.013      0.636     1         0.778      0.844    vgood
Weighted Avg.    0.953     0.024      0.959     0.953     0.954      0.938

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 224   8   1   0 |   a = unacc
   2  66   0   1 |   b = acc
   0   0   9   3 |   c = good
   0   0   0   7 |   d = vgood
```

- Για αρχείο το οποίο έχουμε σπάσει σε train_data(75%) Και test_data(25%)
  Με threshold κατάταξης 80%

Precision for each class

{0: 0.9901315789473685, 1: 0.8064516129032258, 2: 0.9473684210526315, 3: 0.8125}

Mean precision

0.8891129032258064

Recall for each class

{0: 0.9585987261146497, 1: 0.9615384615384616, 2: 0.8181818181818182, 3: 0.7222222222222222}

Mean recall

0.865135307014288

Mean accuracy

0.9421296296296297

Τα αποτελέσματα του weka για το ίδιο dataset

```
Correctly Classified Instances         396               91.6667 %
Incorrectly Classified Instances        13                3.0093 %
Kappa statistic                          0.9216
Mean absolute error                      0.0159
Root mean squared error                  0.1261
Relative absolute error                  7.744  %
Root relative squared error             40.886  %
UnClassified Instances                  23                5.3241 %
Total Number of Instances              432

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.971     0          1         0.971     0.985      0.979    unacc
                 1         0.033      0.866     1         0.928      0.94     acc
                 0.833     0.005      0.833     0.833     0.833      0.775    good
                 0.875     0          1         0.875     0.933      0.818    vgood
Weighted Avg.    0.968     0.006      0.972     0.968     0.969      0.96

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 301   9   0   0 |   a = unacc
   0  71   0   0 |   b = acc
   0   2  10   0 |   c = good
   0   0   2  14 |   d = vgood
```

- Για αρχείο το οποίο έχουμε σπάσει σε train_data(70%) Και test_data(30%)
  Με threshold κατάταξης 80%

Precision for each class

{0: 0.9584487534626038, 1: 0.8083333333333333, 2: 0.8, 3: 0.5882352941176471}

Mean precision

0.7887543452283962

Recall for each class

{0: 0.9637883008356546, 1: 0.8584070796460177, 2: 0.7272727272727273, 3: 0.4166666666666667}

Mean recall

0.7415336936052664

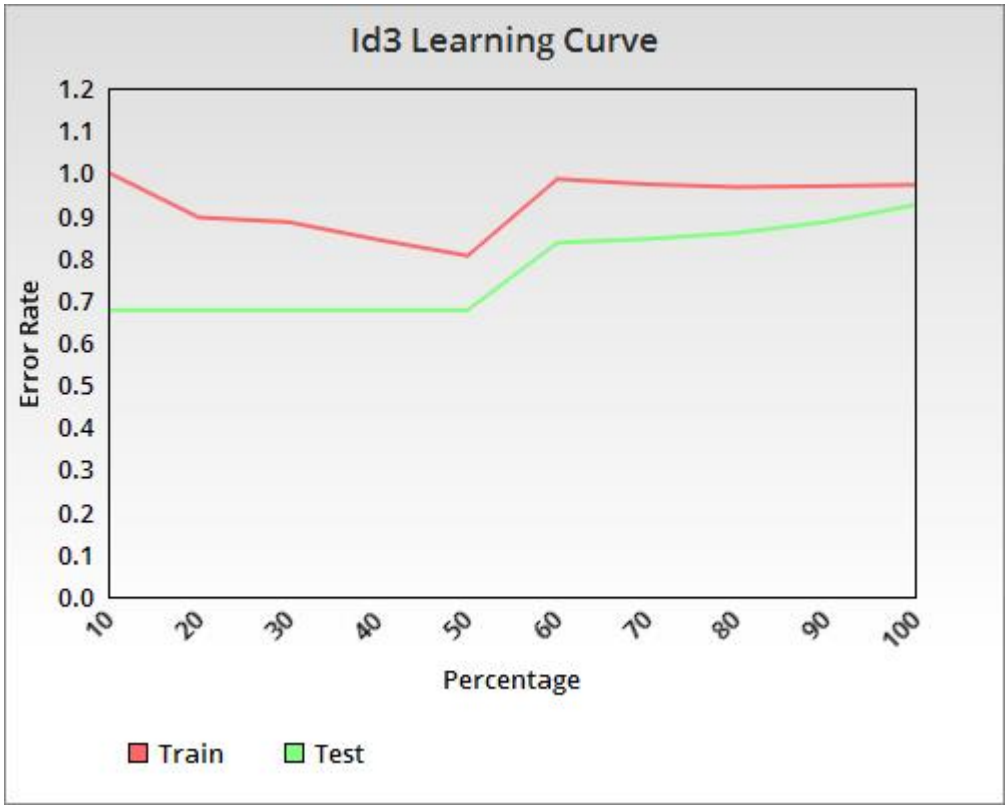
Mean accuracy

0.9054054054054054

Τα αποτελέσματα του weka για το ίδιο dataset

```
Correctly Classified Instances         452               87.2587 %
Incorrectly Classified Instances        34                6.5637 %
Kappa statistic                          0.8338
Mean absolute error                      0.035
Root mean squared error                  0.187
Relative absolute error                 17.0125 %
Root relative squared error             59.4409 %
UnClassified Instances                  32                6.1776 %
Total Number of Instances              518

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.969     0.077      0.972     0.969      0.97       0.949    unacc
                 0.888     0.036      0.861     0.888      0.874      0.868    acc
                 0.5       0.013      0.571     0.5        0.533      0.661    good
                 0.75      0.009      0.75      0.75       0.75       0.769    vgood
Weighted Avg.    0.93      0.064      0.929     0.93       0.929      0.917

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 345  10   1   0 |   a = unacc
  10  87   1   0 |   b = acc
   0   4   8   4 |   c = good
   0   0   4  12 |   d = vgood
```

Η καμπύλη μάθησης για το dataset με train data 80% και test 20% του car.data με threshold 80%



Ada Boost Algorithm

Εισαγωγή

Αρχικά το πρόγραμμα ζητά από τον χρήστη την εισαγωγή ενός dataset προς εκπαίδευση. Δίνεται έπειτα η επιλογή να διαχωριστεί το συγκεκριμένο dataset σε training set και test set. Αν ο χρήστης το επιλέξει τότε το σετ θα χωριστεί με το test set να ειναι το 30% του αρχικού data set. Αν ο χρήστης θέλει μεγαλύτερο έλεγχο ως προς το ποσοστό τότε μπορεί να χρησιμοποιήσει το βοηθητικό split_script. Η υλοποίηση του αλγόριθμου είναι βασισμένη στους αλγόριθμους του βιβλίου τόσο για τον κύριο αλγόριθμο της ada boost όσο και για τον αλγόριθμο του αδύναμου ταξινομητή.

DataSet

Η κλάση DataSet παρέχει μεθόδους ανάγνωσης, αποθήκευσης και επεξεργασίας των datasets. Δέχεται datasets της μορφής attribute1,attribute1,....,attributeN,result. Μετά την εισαγωγή, διασπά τα δεδομένα σε μορφή εύκολη προς χρήση, και ανακαλύπτει όλες τις πιθανές τιμές για κάθε attribute καθώς και όλα τα πιθανά αποτελέσματα Παρέχει διάφορες βοηθητικές μεθόδους για πολλές πιθανές χρήσεις του dataset και επεξεργασίας αυτού μετά την αρχική εισαγωγή. Παράδειγμα η μέθοδος repopulate που αλλάζει τις “εμφανίσεις” του κάθε παραδείγματος ανάλογα με το βάρος του.

AdaBoost

Ο βασικός αλγόριθμος. Η υλοποίησή του είναι όμοια με τον ψευδοκώδικα του βιβλίου οπότε δεν χρειάζεται ιδιαίτερη ανάλυση. Οι διαφορές όμως είναι η μεταβλητή ep για περιορισμό overflow ή undeflow του σφάλματος.

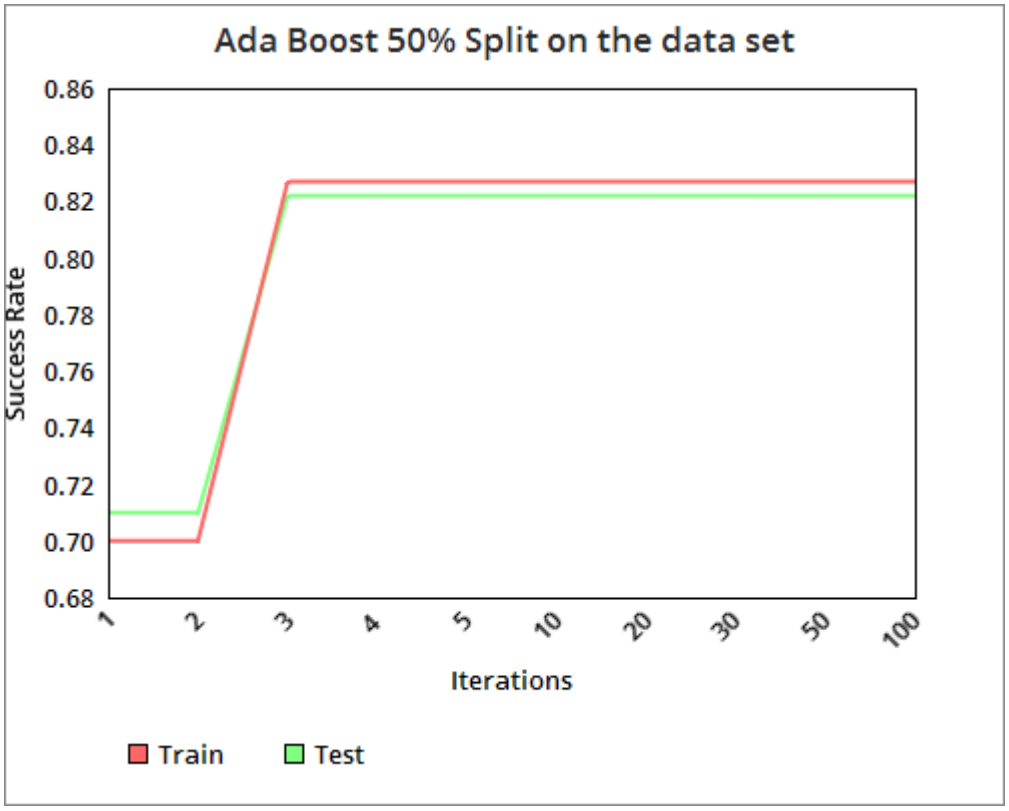
DecisionStump

Ο αδύναμος ταξινομητής, που είναι ένα δέντρο απόφασης βάθους 1. Για να σιγουρέψουμε την μια εκβάθυνση του δέντρου μετά την πρώτη κλήση του train, καλούμε την μέθοδο αναδρομικά με attributes=0, ώστε να είναι σίγουρο ότι το αλγόριθμος θα επιστρέψει το πολύ στην 3η συνθήκη.

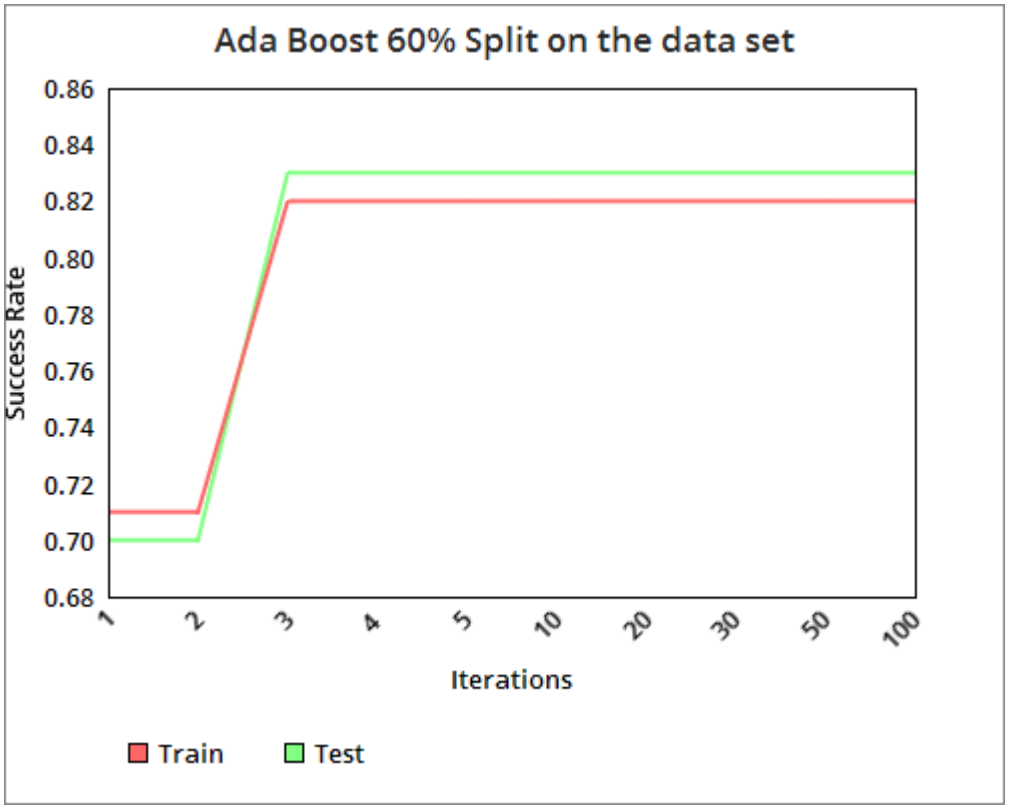
Η συγκεκριμένη υλοποίηση υποστηρίζει και γνωρίσματα αριθμητικά και όχι μόνο κατηγορηματικά Τα αριθμητικά γνωρίσματα υποστηρίζονται μέσω της μεθόδου enhanced split, που χωρίζει τις τιμές που έχει ανακαλύψει το dataset, εφόσον ελέγξει αν αυτές είναι αριθμητικές, σε n κομμάτια όσα είναι τα δυνατά αποτελέσματα Στην πράξη αυτό το κόψιμο δεν είναι ακριβές και έχει αποκλίσεις όμως έχει αρκετή ακρίβεια για τους σκοπούς του αλγορίθμου. Κάθε συνθήκη προστίθεται σαν φύλλο στο δέντρο υπό την μορφή αντικειμένου operation που εξομοιώνει την μορφή μιας φράσης τύπου x=y ή z < x < y.

Για σκοπούς ελέγχου εγκυρότητας του αλγορίθμου χρησιμοποιήθηκαν πολλά και διαφορετικά dataset. Στα πλαίσια αυτής της αναφοράς θα παρουσιάσουμε τα αποτελέσματα για ένα dataset το hospital.data το οποίο περιλαμβάνεται εντός του φακέλου που παραδώσαμε. Το συγκεκριμένο dataset έχει παραδείγματα με nominal values και είναι της μορφής που περιγράψαμε παραπάνω

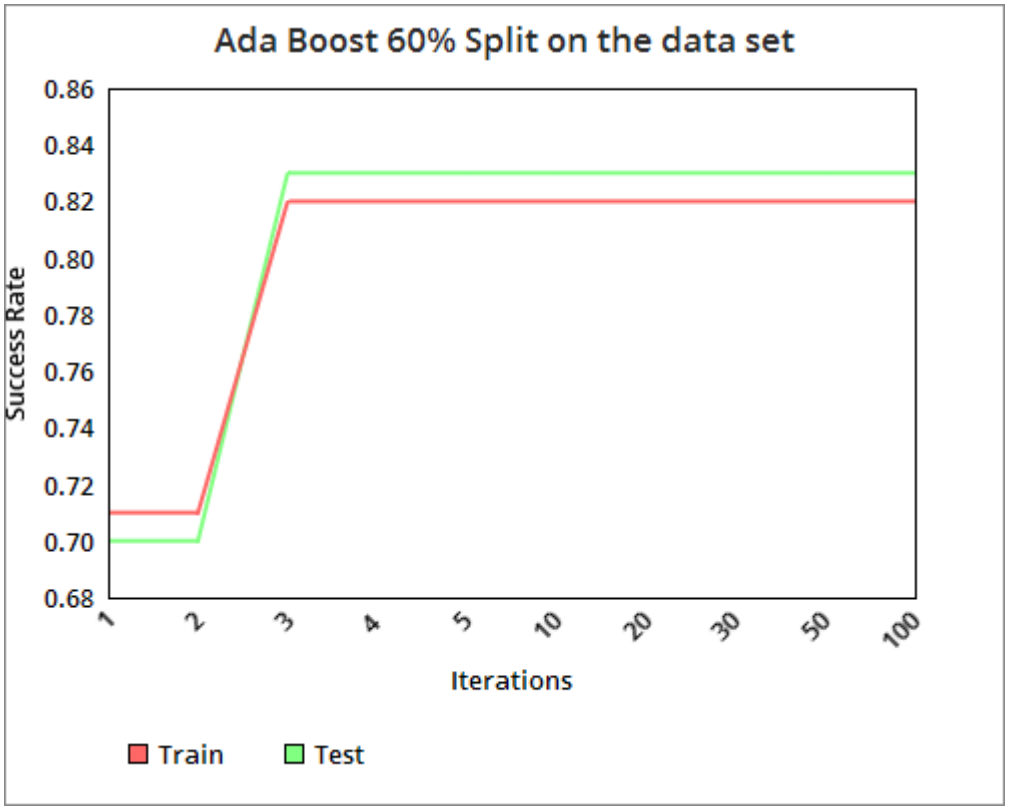
Εκπαίδευση στο 50% των δεδομένων, έλεγχος στο 50%



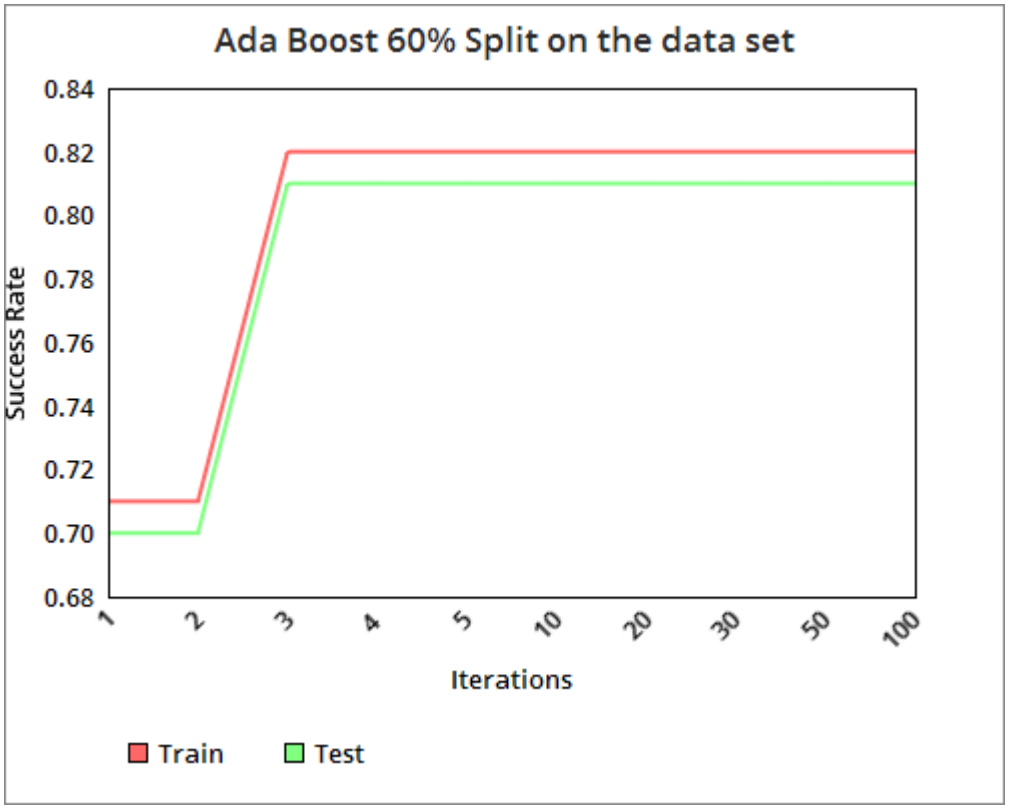
Εκπαίδευση στο 60% των δεδομένων, έλεγχος στο 40%



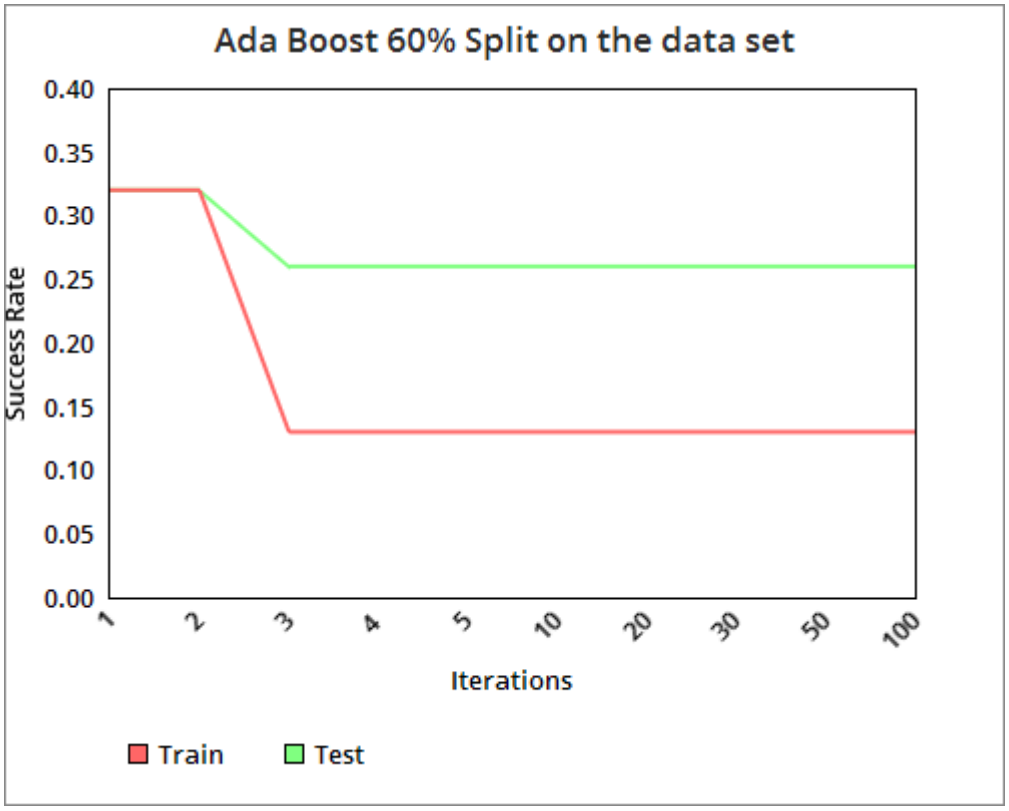
Εκπαίδευση στο 70% των δεδομένων, έλεγχος στο 30%



Εκπαίδευση στο 80% των δεδομένων, έλεγχος στο 20%



Εκπαίδευση στο 90% των δεδομένων, έλεγχος στο 10%



Για τους ελέγχους στο 70/30:

Ποσοστό επιτυχίας με 1 iteration:**70%** αντίστοιχο στο weka: **66,6%**

Ποσοστό επιτυχίας μετά από 3 iterations : **80%** αντίστοιχο στο weka: **66,6%**

Τα ποσοστά παραμένουν ίδια για τα υπόλοιπα iterations.

Ενδεικτικά αποτελέσματα σε 70% train σε άλλα dataset:

Iris:**95%** stable Weka **95%** stable

Car: **70%** stable Weka: **70%** stable

Σε αντίθεση με τον αντίστοιχο αλγόριθμο στο weka ο συγκεκριμένος δείχνει να έχει πολύ καλύτερα αποτελέσματα στο hospital dataset ενώ έχει ίδια αποδοτικότητα στα υπόλοιπα.

Ακόμη παρατηρούμε οτι το decision stump αν και είναι ένας weak learner έχει πολύ καλύτερη επιτυχία απ'ότι περιμέναμε. Παρακάτω ακολουθούν τα σφάλματα 5 επαναλήψεων όπως αυτά επιστράφηκαν απο τον αλγόριθμο adaboost:

```
0.2902336860670214
```

0.30658703812285626

0.410401083645663

0.5252803574996121

0.5000000000000064

Είναι αντιληπτό ότι ο weak learner είναι πολύ καλύτερος από το “κάτι καλύτερο από τυχαία επιλογή” που είναι η υπόθεση των weak learners. Δεν γνωρίζουμε τι το προκαλεί αυτό αλλά είναι και ο λόγος που η συμπεριφορά του αλγορίθμου φαίνεται να μην είναι η προβλεπόμενη, αν και φυσικά δουλεύει πολύ καλύτερα

Αντίστοιχα για εκπαίδευση στο 80%

0.2869405864197488

0.30780674883236614

0.4155085172754358

0.5248557887216169

0.49999999999990224

και 90%

0.6663237311385531

0.49999999999993816

0.5000000000001809

0.49999999999991673

0.4999999999999164

Στο 90% τα ποσοστά είναι εμφανώς χειρότερα που φυσικά είναι λογικό λόγω του μειωμένου μεγέθους των test περιπτώσεων και την κακή κατανομή αυτών που γίνεται τυχαία.

Αλγόριθμος Naive Bayes:

Αρχικά,για τον αλγόριθμο naive bayes μετατρέπουμε τα δεδομέν με ένα τρόπο τέτοιο ώστε που να μας βολεύει στην επεξεργασία και στη καλύτερη εξαγωγή

συμπερασμάτων απο τον αλγόριθμο naive bayes.Διαβάζω τα δεδομένα και τα αποθηκεύω σε μία λίστα η οποία αποτελείτε και αυτή απο λίστες όπου η καθε λίστα προσδιορίζει το δεδομένο είτε είναι αριθμός ειτε ειναι η κλάση.(Ουσιαστικά δουλεύω πολύ πιο εύκολα με αυτόν το τρόπο).

Το πρόγραμμα αποτελέιτε απο τις εξής κλάσεις:

- Parser:Είναι υπεύθυνος για την ανάγνωση και μορφοποιηση των δεδομένων.Περιέχει τη μέθοδο read η οποία διαβάζει ενα αρχείο δεδομένων και το επιστρέφει σε μορφή λιστας.

- Data:Η κλάση αυτή περιέχει χρήσιμες μεθόδους που βοηθούν στην εξαγωγή συμπερασμάτων απο τα δεδομένα που διαβάστηκαν απο τον parser.Πρώτα πρωτα κρατάει δυο μεταβλητές:numOfClasses,frequencies όπου στην ουσια προσδιορίζουν τον αριθμό όλων των κλάσεων που εμφανίστηκαν στο αρχείο και το πόσες εμφανίζεται μια τιμή για μια συγκεκριμένη κλάση αντίστοιχα.Ολες οι μεθοδοι της κλάσης βοηθούν στην αναγνώριση και διάκριση των στοιχείων μεταξύ τους.

- MachineLearner:Εδω αυτη η κλάση ειναι χρήσιμη στο να εκπαιδευεται απο τα δεδομένα.Ουσιαστικά διαβάζει τα δεδομένα και βλέπει πόσες κλάσεις υπάρχουν και πόσες μεταβλητές για κάθε κλάσση υπάρχουν.

- Classifier:Είναι η καρδιά του προγράμματος.Εδω γίνονται όλοι οι υπολογισμοί που χρειάζονται.Παρέχει όλες τις μεθόδους για τον υπολογισμό των πιθανοτήτων.Επίσης η μέθοδος που πρέπει να δώσουμε μεγαλύτερη προσοχή είναι η classify.Σε αυτη τη μέθοδο δίνονται τα δεδομένα που θέλουμε να δοκιμάσουμε και υπολογίζει την πιθανότητα με βάση τα δεδομένα που έχουμε ήδη εκπαιδευτεί.

- split_script:Ειναι υπεύθυνο για την ανάγνωση του αρχείου με αποτέλεσμα τη διάσπαση του σε δυο επιμέρους κομμάτια με βάση ενός ποσοστού.Για παράδειγμα διαχωρίζει ενα αρχείο σε αρχείο με δεδομένα για εκπαίδευση και σε δεδομένα για εξέταση.

- MainFile:Εδω γίνεται η επίδειξη του προγράμματος.Εδώ γίνονται ολες οι αναγνώσεις και η εξαγωγή συμπερασμάτων.