ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΙ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΑΘΗΜΑ: ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

ΚΑΘΗΓΗΤΕΣ : ΚΩΣΤΑΣ ΔΙΑΜΑΝΤΑΡΑΣ, ΚΩΣΤΑΣ ΓΟΥΛΙΑΝΑΣ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 2

ΓΡΑΜΜΙΚΟΣ ΤΑΞΙΝΟΜΗΤΗΣ

Σκοπός της άσκησης: Η εκτίμηση της επίδοσης ενός γραμμικού ταξινομητή δύο κλάσεων σε άγνωστα δεδομένα. Θα γίνει χρήση της μεθόδου διασταύρωσης (Cross-Validation).

Θα χρησιμοποιηθούν τα εξής κριτήρια επίδοσης:

1. Ακρίβεια (accuracy)
2. Ευστοχία (precision)
3. Ανάκληση (recall)
4. F-Measure
5. Ευαισθησία (Sensitivity)
6. Προσδιοριστικότητα (Specificity)

Δείτε τη σημασία αυτών των κριτηρίων στο επισυναπτόμενο κείμενο Κριτήρια επίδοσης ταξινομητών.pdf

Βήματα υλοποίησης:

1. Χρησιμοποιήστε το σύνολο δεδομένων IRIS από το προηγούμενο εργαστήριο, καθώς και τον κώδικα από το εργαστήριο αυτό.
2. Με τη χρήση της εντολής plot δημιουργήστε τη **γραφική παράσταση** των προτύπων **των 3 κλάσεων** με **διαφορετικό σύμβολο και χρώμα για την κάθε κλάση** χρησιμοποιώντας την $1^{η}$ και $3^{η}$ στήλη του πίνακα x, ώστε να τα απεικονίσετε στο χώρο των 2 διαστάσεων και εμφανίστε τα στο ίδιο γράφημα, ώστε να πάρετε μια ιδέα για το πώς είναι η διασπορά των προτύπων στο χώρο των 4 διαστάσεων. Μη ξεχνάτε ότι η αρίθμηση ξεκινάει απ' το 0.
3. Επαυξήστε τον πίνακα των προτύπων προσθέτοντας σε κάθε πρότυπο τον αριθμό 1, δηλαδή προσθέστε μια στήλη με 1 στον πίνακα `x`. Χρησιμοποιήστε τη συνάρτηση `hstack()` του `numpy`.
4. Χρησιμοποιώντας τη συνάρτηση zeros από τη βιβλιοθήκη numpy αρχικοποιήστε τον πίνακα `t` ώστε να είναι γεμάτος μηδενικά και να έχει διάσταση `NumberOfPatterns`.
5. **Εκχωρήστε** στη μεταβλητή ans την τιμή "y'.
6. Για όσο (ans = "y')
   - **Εμφανίστε** το παρακάτω **menu** επιλογών :

     1 **Διαχωρισμός `Iris-setosa`** από `Iris-versicolor και Iris-virginica`

     2 **Διαχωρισμός `Iris-virginica`** από `Iris-setosa και Iris-versicolor`

     3 **Διαχωρισμός `Iris-versicolor`** από `Iris-setosa και Iris-virginica`

     Διαβάστε την επιλογή (1/2/3)

     Αν επιλογή = 1

Δημιουργήστε ένα dictionary `map_dict` με τα εξής ζευγάρια key/values:

- `"Iris-setosa": 1`
- `"Iris-versicolor": 0`
- `"Iris-virginica": 0`

Κατόπιν, χρησιμοποιώντας loop θέστε για κάθε pattern την τιμή στόχου `t[pattern]` ως εξής:

`t[pattern] = 1` αν η 5ο στήλη για το `pattern` είναι "`Iris-setosa`"

`t[pattern] = 0` σε διαφορετική περίπτωση

Αν επιλογή = 2

Δημιουργήστε ένα dictionary `map_dict` με τα εξής ζευγάρια key/values:

- `"Iris-setosa": 0`
- `"Iris-versicolor": 0`
- `"Iris-virginica": 1`

Κατόπιν, χρησιμοποιώντας loop θέστε για κάθε pattern την τιμή στόχου `t[pattern]` ως εξής:

`t[pattern] = 1` αν η 5ο στήλη για το `pattern` είναι "`Iris-virginica`"

`t[pattern] = 0` σε διαφορετική περίπτωση

Αν επιλογή = 3

Δημιουργήστε ένα dictionary `map_dict` με τα εξής ζευγάρια key/values:

- `"Iris-setosa": 0`
- `"Iris-versicolor": 1`
- `"Iris-virginica": 0`

Κατόπιν, χρησιμοποιώντας loop θέστε για κάθε pattern την τιμή στόχου `t[pattern]` ως εξής:

`t[pattern] = 1` αν η 5ο στήλη για το `pattern` είναι "`Iris-versicolor`"

`t[pattern] = 0` σε διαφορετική περίπτωση

Μπορείτε να το κάνετε αυτό χρησιμοποιώντας το `map_dict` και να αποφύγετε εντολή `if-else`.

7. Χωρισμός προτύπων σε πρότυπα εκπαίδευσης και ανάκλησης

Τεμαχίστε τα δεδομένα των πινάκων x και t σε 4 πίνακες:

- `xtrain` πίνακας με τα πρότυπα που θα χρησιμοποιηθούν στην εκπαίδευση, `τα 40 πρώτα πρότυπα της κάθε κλάσης.`
- `xtest` πίνακας με τα πρότυπα που θα χρησιμοποιηθούν στον έλεγχο, `τα 10 τελευταία πρότυπα της κάθε κλάσης.`
- `ttrain` διάνυσμα με τους στόχους που θα χρησιμοποιηθούν στην εκπαίδευση, οι 40 πρώτοι στόχοι `της κάθε κλάσης.`
- `ttest` διάνυσμα με τους στόχους που θα χρησιμοποιηθούν στον έλεγχο, οι 10 `τελευταίοι` στόχοι `της κάθε κλάσης.`

- Χρησιμοποιώντας τη συνάρτηση `plot` από τη βιβλιοθήκη `matplotlib.pyplot` σχεδιάστε

  - τα διανύσματα `xtrain[:,0]` → άξονας x, `xtrain[:,2]` → άξονας y, χρησιμοποιώντας τελείες με μπλε χρώμα και
  - τα διανύσματα `xtest[:,0]` → άξονας x, `xtest[:,2]` → άξονας y, χρησιμοποιώντας τελείες με κόκκινο χρώμα.

- Μετατρέψτε τα διανύσματα στόχων `ttrain(), ttest()` έτσι ώστε
  - Αν `ttrain(pattern) == 1` → `ttrain1(pattern) = 1`
  - Αν `ttrain(pattern) == 0` → `ttrain1(pattern) = -1`
  - Αν `ttest(pattern) == 1` → `ttest1(pattern) = 1`
  - Αν `ttest(pattern) == 0` → `ttest1(pattern) = -1`
- Βρείτε το διάνυσμα βαρών $\tilde{\mathbf{w}}$ του γραμμικού ταξινομητή

$$y = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}$$

$$\tilde{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ w_0 \end{bmatrix}, \qquad \tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

όπου $\mathbf{w}$ είναι το διάνυσμα βαρών, w_0 είναι η πόλωση, και $\mathbf{x}$ είναι το πρότυπο εισόδου.

- Υπολογίστε το βέλτιστο διάνυσμα βαρών $\tilde{\mathbf{w}}$. Σύμφωνα με τη θεωρία των γραμμικών ταξινομητών, το βέλτιστο διάνυσμα είναι

$$\tilde{\mathbf{w}}^T = \mathbf{t}_{train}^T \, \tilde{\mathbf{X}}_{train}^+$$

όπου

$\mathbf{t}_{train}^T = [t_1 \quad t_2 \quad \cdots \quad t_P]$ είναι το διάνυσμα των τροποποιημένων στόχων (-1/1),

$\tilde{\mathbf{X}}_{train} = [\tilde{\mathbf{x}}_1 \quad \tilde{\mathbf{x}}_2 \quad \cdots \quad \tilde{\mathbf{x}}_P] = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_P \\ 1 & 1 & \cdots & 1 \end{bmatrix}$ είναι ο πίνακας των επαυξημένων προτύπων του train set,

$\mathbf{X}^+$ είναι ο τελεστής του ψευδο-αντίστροφου του πίνακα $\mathbf{X}$. Στην python ο ψευδο-αντίστροφος υλοποιείται με τη συνάρτηση numpy.linalg.`pinv()`.

- Υπολογίστε την έξοδο του ταξινομητή για όλα τα πρότυπα του $train$ set:

$$\mathbf{y}_{train}^T = \tilde{\mathbf{w}}^T \tilde{\mathbf{X}}_{train}$$

- Υπολογίστε την εκτίμηση που κάνει ο ταξινομητής για την κλάση στην οποία ανήκουν τα πρότυπα του $train$ set:

$$predict_{train}\,(i) = \begin{cases} 0, & y_{train}\,(i) < 0 \\ 1, & y_{train}\,(i) \geq 0 \end{cases}$$

- Τυπώστε το εξής γράφημα:
  - δείξτε με μπλε τελείες τους πραγματικούς στόχους $t_{train}\,[i]$ για όλα τα πρότυπα του $train$ set
  - δείξτε με κόκκινους κύκλους τους εκτιμώμενους στόχους $predict_{train}\,[i]$ για όλα τα πρότυπα του $train$ set.

- Υπολογίστε την έξοδο του ταξινομητή για όλα τα πρότυπα του test set:

$$\mathbf{y}_{test}^T = \tilde{\mathbf{w}}^T \tilde{\mathbf{X}}_{test}$$

- Υπολογίστε την εκτίμηση που κάνει ο ταξινομητής για την κλάση στην οποία ανήκουν τα πρότυπα του test set:

$$predict_{test}\,(i) = \begin{cases} 0, & y_{test}\,(i) < 0 \\ 1, & y_{test}\,(i) \geq 0 \end{cases}$$

- Τυπώστε το εξής γράφημα:

- δείξτε με μπλε τελείες τους πραγματικούς στόχους $t_{test}\,[i]$ για όλα τα πρότυπα του test set
- δείξτε με κόκκινους κύκλους τους εκτιμώμενους στόχους $predict_{test}\,[i]$ για όλα τα πρότυπα του test set

8. Θα εφαρμοστεί η μέθοδος `train_test_split(...)` για K=9 folds. Στο αντίστοιχο loop θα πρέπει να κάνετε τα εξής:

Για κάθε *fold:*

- Έχετε ήδη δημιουργήσει τους πίνακες `xtrain, xtest` καθώς και τα διανύσματα στόχων `ttrain, ttest.` Φροντίστε να είναι `numpy arrays` τύπου `float.`
- Βρείτε το πλήθος των προτύπων στο train set (P_{train}) και στο test set (P_{test}) για παράδειγμα χρησιμοποιώντας τη συνάρτηση `len()`.
- Μετατρέψτε τα διανύσματα στόχων `ttrain(), ttest()` έτσι ώστε
  - Αν `ttrain(pattern) == 1` → `ttrain1(pattern) = 1`
  - Αν `ttrain(pattern) == 0` → `ttrain1(pattern) = -1`
  - Αν `ttest(pattern) == 1` → `ttest1(pattern) = 1`
  - Αν `ttest(pattern) == 0` → `ttest1(pattern) = -1`
- Βρείτε το διάνυσμα βαρών $\tilde{\mathbf{w}}$ του γραμμικού ταξινομητή

$$y = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}$$

$$\tilde{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ w_0 \end{bmatrix}, \qquad \tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

όπου $\mathbf{w}$ είναι το διάνυσμα βαρών, w_0 είναι η πόλωση, και $\mathbf{x}$ είναι το πρότυπο εισόδου.

- Υπολογίστε το βέλτιστο διάνυσμα βαρών $\tilde{\mathbf{w}}$. Σύμφωνα με τη θεωρία των γραμμικών ταξινομητών, το βέλτιστο διάνυσμα είναι

$$\tilde{\mathbf{w}}^T = \mathbf{t}_{train}^T \, \tilde{\mathbf{X}}_{train}^+$$

όπου

$\mathbf{t}_{train}^T = [t_1 \quad t_2 \quad \cdots \quad t_P]$ είναι το διάνυσμα των τροποποιημένων στόχων (-1/1),

$\tilde{\mathbf{X}}_{train} = [\tilde{\mathbf{x}}_1 \quad \tilde{\mathbf{x}}_2 \quad \cdots \quad \tilde{\mathbf{x}}_P] = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_P \\ 1 & 1 & \cdots & 1 \end{bmatrix}$ είναι ο πίνακας των επαυξημένων προτύπων του train set,

$\mathbf{X}^+$ είναι ο τελεστής του ψευδο-αντίστροφου του πίνακα $\mathbf{X}$. Στην python ο ψευδο-αντίστροφος υλοποιείται με τη συνάρτηση numpy.linalg.`pinv()`.

- Υπολογίστε την έξοδο του ταξινομητή για όλα τα πρότυπα του test set:

$$\mathbf{y}_{test}^T = \tilde{\mathbf{w}}^T \tilde{\mathbf{X}}_{test}$$

- Υπολογίστε την εκτίμηση που κάνει ο ταξινομητής για την κλάση στην οποία ανήκουν τα πρότυπα του test set:

$$predict_{test}\,(i) = \begin{cases} 0, & y_{test}\,(i) < 0 \\ 1, & y_{test}\,(i) \geq 0 \end{cases}$$

- Υλοποιήστε τη συνάρτηση `evaluate()` με τρεις εισόδους και μια έξοδο ως εξής:

```
def evaluate( t, predict, criterion ):
"""
Είσοδος t : διάνυσμα με τους πραγματικούς στόχους (0/1)
Είσοδος predict : διάνυσμα με τους εκτιμώμενους στόχους (0/1)
Είσοδος criterion : text-string με τις εξής πιθανές τιμές:
  'accuracy'
  'precision'
  'recall'
  'fmeasure'
  'sensitivity'
  'specificity'
Έξοδος value : η τιμή του κριτηρίου που επιλέξαμε.
"""
```

- Πρώτα υπολογίστε τα true-negatives, false-negatives, true-positives, false positives, τα οποία ορίζονται ως εξής:

  (α) **true negatives** (πραγματικά αρνητικά) οι περιπτώσεις όπου:
  το πρότυπο βγήκε *αρνητικό*, δηλ. $y_i = 0$
  και όντως *ανήκει στην κλάση 0*, δηλ. $t_i = 0$

  (β) **false negatives** (εσφαλμένα αρνητικά) οι περιπτώσεις όπου:
  το πρότυπο βγήκε *αρνητικό*, δηλ. $y_i = 0$
  αλλά *ανήκει στην κλάση 1*, δηλ. $t_i = 1$

  (γ) **true positives** (πραγματικά θετικά) οι περιπτώσεις όπου:
  το πρότυπο βγήκε *θετικό*, δηλ. $y_i = 1$
  και όντως *ανήκει στην κλάση 1*, δηλ. $t_i = 1$

  (δ) **false positives** (εσφαλμένα θετικά) οι περιπτώσεις όπου:
  το πρότυπο βγήκε *θετικό*, δηλ. $y_i = 1$
  αλλά *ανήκει στην κλάση 0*, δηλ. $t_i = 0$

Σημαντικό! Φροντίστε τα tp , tn , fp , fn να είναι τύπου `float`. Χρησιμοποιήστε την συνάρτηση `float()`

Κατόπιν

- Αν επιλεγεί `criterion = 'accuracy'` τότε η συνάρτηση επιστρέφει

$$value = Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

- Αν επιλεγεί `criterion = 'precision'` τότε η συνάρτηση επιστρέφει

$$value = Precision = \frac{tp}{tp + fp}$$

- Αν επιλεγεί `criterion = 'recall'` τότε η συνάρτηση επιστρέφει

$$value = Recall = \frac{tp}{tp + fn}$$

- Αν επιλεγεί `criterion = 'fmeasure'` τότε η συνάρτηση επιστρέφει

$$value = Fmeasure = \frac{Precision \cdot Recall}{(Precision + Recall)/2}$$

- Καλέστε τη συνάρτηση `evaluate()` όσες φορές χρειάζεται έτσι ώστε για το συγκεκριμένο fold να υπολογίσετε το Accuracy, Precision, Recall, F-measure, Sensitivity και Specificity.
- Χρησιμοποιώντας κατάλληλο subplot σε grid 3x3 στο figure(1) τυπώστε το εξής γράφημα:
  - δείξτε με μπλε τελείες τους πραγματικούς στόχους $t_{test}\,[i]$ για όλα τα πρότυπα του test set
  - δείξτε με κόκκινους κύκλους τους εκτιμώμενους στόχους $predict_{test}\,[i]$ για όλα τα πρότυπα του test set

```
# end for
```

Μετά το τέλος του loop υπολογίστε και τυπώστε στην οθόνη τα εξής:

1. τη μέση τιμή του Accuracy για όλα τα folds
2. τη μέση τιμή του Precision για όλα τα folds
3. τη μέση τιμή του Recall για όλα τα folds
4. τη μέση τιμή του F-Measure για όλα τα folds
5. τη μέση τιμή του Sensitivity για όλα τα folds
6. τη μέση τιμή του Specificity για όλα τα folds

Διαβάστε την απάντηση ans του χρήστη, αν θέλετε να συνεχίσετε.

Θα χρησιμοποιήσετε τις παρακάτω εντολές ή συναρτήσεις:

- Υλοποιήστε τη λειτουργία `switch` με dictionary.

- Χρησιμοποιήστε τη συνάρτηση `pinv()` του `numpy.linalg` για τον υπολογισμό ψευδο-αντίστροφου ενός πίνακα

```
# Παράδειγμα:
>>> import numpy as np
>>> A = np.array([[0.3,    0.1,    0.2],
                  [0.5,    1,    0.2]])
>>> A1 = np.linalg.pinv(A)
>>> A1
array([[ 2.50777202, -0.1761658 ],
       [-1.66839378,  1.15025907],
       [ 2.07253886, -0.31088083]])
```

- Πολλαπλασιασμός πίνακα επί διάνυσμα

```
>>> w = np.array([3, -2, 0.5])
>>> A.dot(w)
array([ 0.8, -0.4])
>>> np.matmul(A, w)
array([ 0.8, -0.4])     # Ακριβώς ίδιο με το .dot()
```

- Λογικοί τελεστές με διανύσματα:

```
# Παράδειγμα:
>>> z = np.array([0.1,   -0.5,   0,    0.5,   -0.1,    0.3,    0.2])
>>> y1 = (z>0)
>>> y1
array([ True, False, False,  True, False,  True,  True], dtype=bool)
>>> y2 = (z==0)
>>> y2
array([False, False,  True, False, False, False, False], dtype=bool)
>>> y3
array([False,  True, False, False,  True, False, False], dtype=bool)
```

Στην Python υπάρχουν οι Boolean σταθερές *True* και *False*. Συνεπώς μπορούμε να κάνουμε λογικές πράξεις μεταξύ binary διανυσμάτων ως εξής:

```
# Παράδειγμα λογικού OR:
>>> y1|y2
array([ True, False,  True,  True, False,  True,  True], dtype=bool)
# Παράδειγμα λογικού NOT:
>>> y1
array([ True, False, False,  True, False,  True,  True], dtype=bool)
>>> ~y1
array([False,  True,  True, False,  True, False, False], dtype=bool)
```

- Η συνάρτηση `sum()`: αν το όρισμά της είναι διάνυσμα, επιστρέφει ένα αριθμό που είναι το άθροισμα των στοιχείων του διανύσματος. Αν το όρισμα είναι πίνακας υπολογίζει το άθροισμα των στοιχείων κάθε στήλης και επιστρέφει διάνυσμα.

```
# Παράδειγμα:
>>> z = np.array([0.1,   -0.5,   0,    0.5,   -0.1,    0.3,    0.2])
>>> y
array([ True, False, False,  True, False,  True,  True], dtype=bool)
>>> sum(y)
4
```

Οδηγίες κατάθεσης ασκήσεων

1. Συνδεθείτε στο URL: http://aetos.it.teithe.gr/s
1. Επιλέξτε το μάθημα “Μηχανική Μάθηση – Εργαστήριο Χ” (Όπου Χ ο αριθμός του εργαστηρίου του οποίου τις ασκήσεις πρόκειται να καταθέσετε) και πατήστε επόμενο.
2. Συμπληρώστε τα στοιχεία σας. Πληκτρολογείστε USERNAME 00003 και PASSWORD 30000 (Επώνυμο και Όνομα με ΛΑΤΙΝΙΚΟΥΣ ΧΑΡΑΚΤΗΡΕΣ).
3. Αν θέλετε να καταθέσετε μόνο ένα αρχείο μη το βάζετε σε zip file. Αντίθετα, αν θέλετε να καταθέσετε περισσότερα από ένα αρχεία, τοποθετήστε τα σε ένα zip ή rar file.
4. Επιλέξτε το αρχείο που θέλετε να στείλετε επιλέγοντας “choose file” στο πεδίο FILE1 και πατήστε “Παράδοση”