



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΕΧΝΙΟ ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

1^η ΣΕΙΡΑ ΓΡΑΠΤΩΝ ΑΣΚΗΣΕΩΝ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: Καμπουγέρης Χαράλαμπος ΑΜ: 03120098

ΜΑΘΗΜΑ: ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

ΑΚ. ΕΤΟΣ: 2023-2024 ΕΞΑΜΗΝΟ: 7^ο

Email: xarrisk2002@gmail.com / el20098@mail.ntua.gr

ΑΣΚΗΣΗ 1.1

Θεωρούμε το ακόλουθο μοντέλο παλινδρόμησης πάνω σε 11 ανεξάρτητες μεταβλητές x_i , $i=1, 2, \dots, 11$ και

$$y = \sum_{i=1}^{11} w_i x_i$$

Για την επίλυση θα χρησιμοποιηθεί η γλώσσα python και το google collab.

A)

Πρώτα θα φορτώσουμε τα δεδομένα του csv αρχείου.

Ορίζουμε το ερωτηματικό ως οριοθέτη και τη πρώτη σειρά του αρχείου ως τα ονόματα των στηλών. Επίσης μετατρέπουμε όλες τις στήλες στο Data Frame σε αριθμητικές τιμές.

```

file_path = '/content/drive/My Drive/machine_learning/ML2023-24-hwk1.csv'

# Load the data
data = pd.read_csv(file_path, delimiter=';', header=None, names=['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', 'quality'])

# Convert numeric columns to float
data = data.apply(pd.to_numeric, errors='coerce')

```

Έπειτα κανουμε κανονικοποίηση αφαιρώντας τη μέση τιμή των δεδομένων και διαιρώντας με την απόκλιση για κάθε στήλη. Έτσι διασφαλίζουμε ότι κάθε δεδομένο έχει μέσο όρο 0 και τυπική απόκλιση 1.

```

# Normalize the Data
normalized_data = (data - data.mean()) / data.std()

```

Μετά υπολογίζουμε τη συσχέτιση μεταξύ των “pH” και “sulphates” ($r_{(9,10)}$).

```

# Calculate the Normalized Correlation Coefficient (r(9,10))
correlation_9_10 =
normalized_data['pH'].corr(normalized_data['sulphates'])
print(f"Normalized Correlation Coefficient between pH and sulphates: {correlation_9_10}")

```

το αποτέλεσμα είναι:

```

Normalized Correlation Coefficient between pH and sulphates: -
0.19664759122485098

```

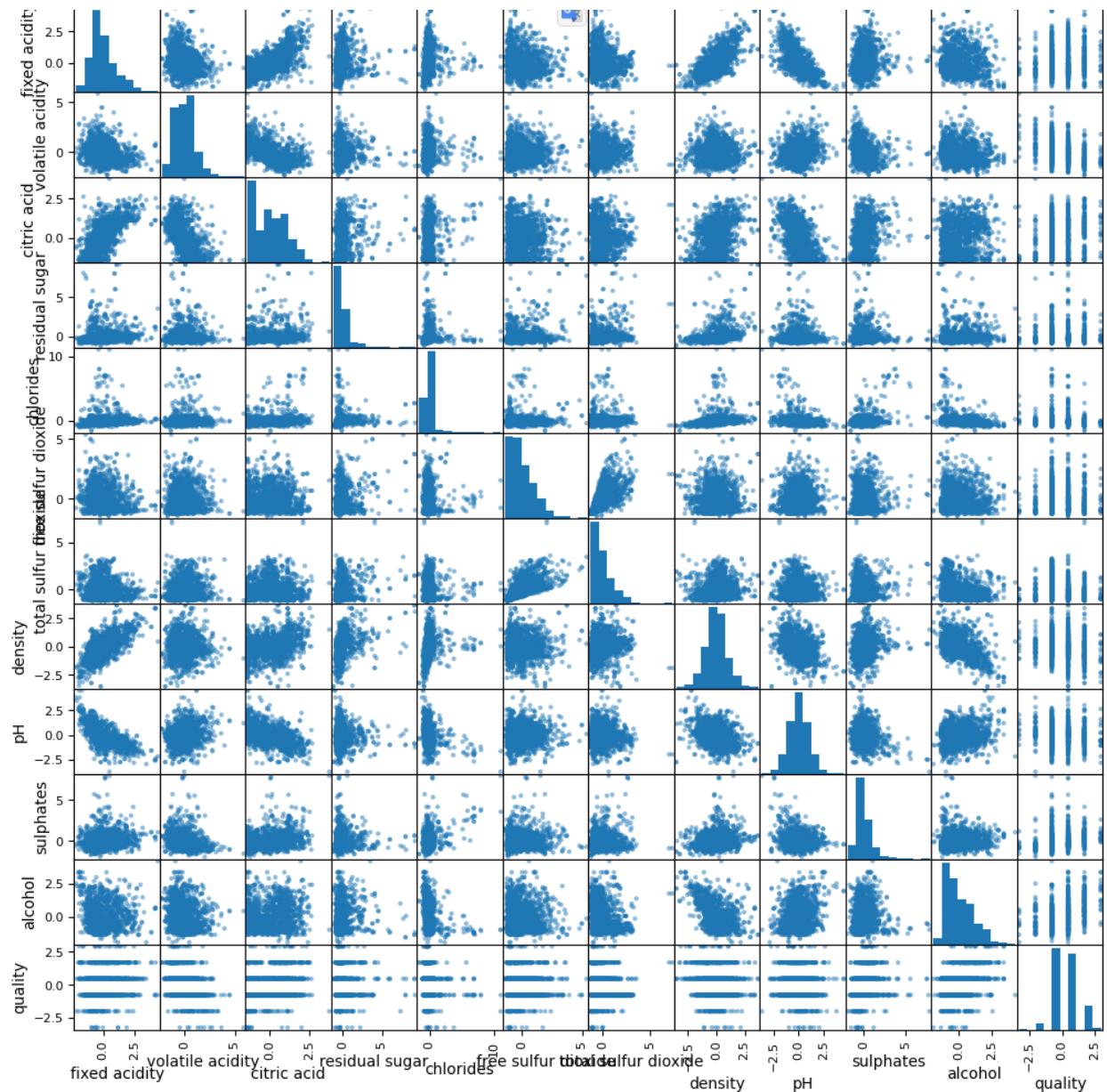
Ο συντελεστής συσχέτισης -0,197 υποδηλώνει μια ασθενή αρνητική γραμμική σχέση μεταξύ του pH και του sulphates στο dataset μας.

Τέλος, φτιάχνουμε τον πίνακα συσχέτισης με scatterplots.

```

# Create a Correlation Matrix with Scatterplots
scatter_matrix = pd.plotting.scatter_matrix(normalized_data, figsize=(12, 12), diagonal='hist')
plt.show()

```



Χωρίζουμε επίσης τα δεδομένα σε training set και test set όπως ζητήθηκε.

```
# Define the number of rows for training and test sets
num_training_rows = 100
num_test_rows = 50

# Split the data into training and test sets
training_data = normalized_data.iloc[:num_training_rows, :]
test_data = normalized_data.iloc[num_training_rows:num_training_rows + num_test_rows, :]
```

B)

Επειτα, αφαιρουμε τις σειρές με NaN τιμές, χωριζουμε τα features από τα target variables(quality).

```
import numpy as np

# Extract features (X) and target variable (y) for training data
X_train = training_data.drop('quality', axis=1) # All columns except the last one
y_train = training_data['quality'] # Last column (quality)

X_test = test_data.drop('quality', axis=1)
y_test = test_data['quality']
```

Η γραμμική παλινδρόμηση είναι μια στατιστική μέθοδος που χρησιμοποιείται για τη μοντελοποίηση της σχέσης μεταξύ μιας εξαρτημένης μεταβλητής (στόχος) και μιας ή περισσότερων ανεξάρτητων μεταβλητών (χαρακτηριστικά ή προγνωστικοί παράγοντες). Ο στόχος είναι να βρεθεί η πιο κατάλληλη γραμμική σχέση που ελαχιστοποιεί τη διαφορά μεταξύ των προβλεπόμενων και των πραγματικών τιμών της μεταβλητής στόχου.

Ο τύπος για τον υπολογισμό των βαρών στη γραμμική παλινδρόμηση μπορεί να αναπαρασταθεί ως:

$$w = (X^T X)^{-1} X^T y$$

όπου:

- W το διάνυσμα των βαρών
- X ο πίνακας τιμών των features
- y είναι το διάνυσμα των τιμών μεταβλητής στόχου.

Το αποτέλεσμα w περιέχει τα βέλτιστα βάρη για την γραμμική παλινδρόμηση.

Οπότε, εφαρμόζουμε την φόρμουλα στα στοιχεία μας με τον παρακάτω κώδικα και μας δίνει αυτό το αποτέλεσμα.

```
#Calculate the weights using the linear regression formula
weights_linear_regression = np.linalg.pinv(X_train.T @ X_train) @
X_train.T @ y_train

# Display the weights
print("Weights for Linear Regression:")
for i, weight in enumerate(weights_linear_regression):
    print(f"w{i}: {weight}")
```

```

Weights for Linear Regression:
w0: 0.20262448489665985
w1: -0.42745766043663025
w2: -0.3318268358707428
w3: -0.006668483838438988
w4: -0.008222042582929134
w5: 0.2698926329612732
w6: -0.3115069270133972
w7: -0.0863097608089447
w8: -0.19337978959083557
w9: 0.019866107031702995
w10: 0.4241105020046234

```

Γ)

Στη συνέχεια θα βρούμε τα βάρη χρησιμοποιώντας Ridge Regression στα δεδομένα μας.

Ορίζουμε τις τιμές της υπερπαραμετρου λ σε 10, 100 και 200.

Η Ridge Regression είναι μια επέκταση της γραμμικής παλινδρόμησης που εισάγει έναν όρο τακτοποίησης στη συνάρτηση κόστους. Αυτή η τεχνική είναι ιδιαίτερα χρήσιμη όταν δύο ή περισσότερες ανεξάρτητες μεταβλητές συσχετίζονται σε μεγάλο βαθμό.

Ο τύπος για τον υπολογισμό των βαρών δίνεται από:

$$w = (X^T X + \lambda I)^{-1} X^T y$$

όπου:

- W το διάνυσμα των βαρών
- X ο πίνακας τιμών των features
- y είναι το διάνυσμα των τιμών μεταβλητής στόχου.
- I είναι το identity matrix

```

#Define the regularization parameters
lambda_values = [10, 100, 200]

#Initialize an empty dictionary to store weights for each lambda
weights_ridge_regression = []
ridge_weights = {}

#Iterate over lambda values
for lambda_val in lambda_values:
    #Ridge Regression formula
    w_ridge = np.linalg.inv(X_train.T @ X_train + lambda_val *
np.identity(X_train.shape[1])) @ (X_train.T @ y_train)
    weights_ridge_regression.append(w_ridge)

#Store the weights in the dictionary

```

```

ridge_weights[lambda_val] = w_ridge

#Display the weights
for lambda_val, weights in ridge_weights.items():
    print(f"Weights for λ = {lambda_val}:")
    for i, weight in enumerate(weights):
        print(f"w{i}: {weight}")
    print()

```

Weights for $\lambda = 10$:

```

w0: 0.18100236572219375
w1: -0.32868156787091013
w2: -0.21136125974714082
w3: 0.003151488951650794
w4: -0.0201447706106405
w5: 0.2002102939657162
w6: -0.2654465581847494
w7: -0.07115676847634034
w8: -0.16396430991367222
w9: 0.014855298800767164
w10: 0.35758755850870483

```

Weights for $\lambda = 100$:

```

w0: 0.10664283271953415
w1: -0.13515933651423595
w2: -0.012598286218773732
w3: 0.009675205793405153
w4: -0.0310310669960041
w5: 0.04474869803682916
w6: -0.12276156908053337
w7: -0.017653716800999995
w8: -0.07544580310819109
w9: -0.00011754689701604093
w10: 0.16824376006969455

```

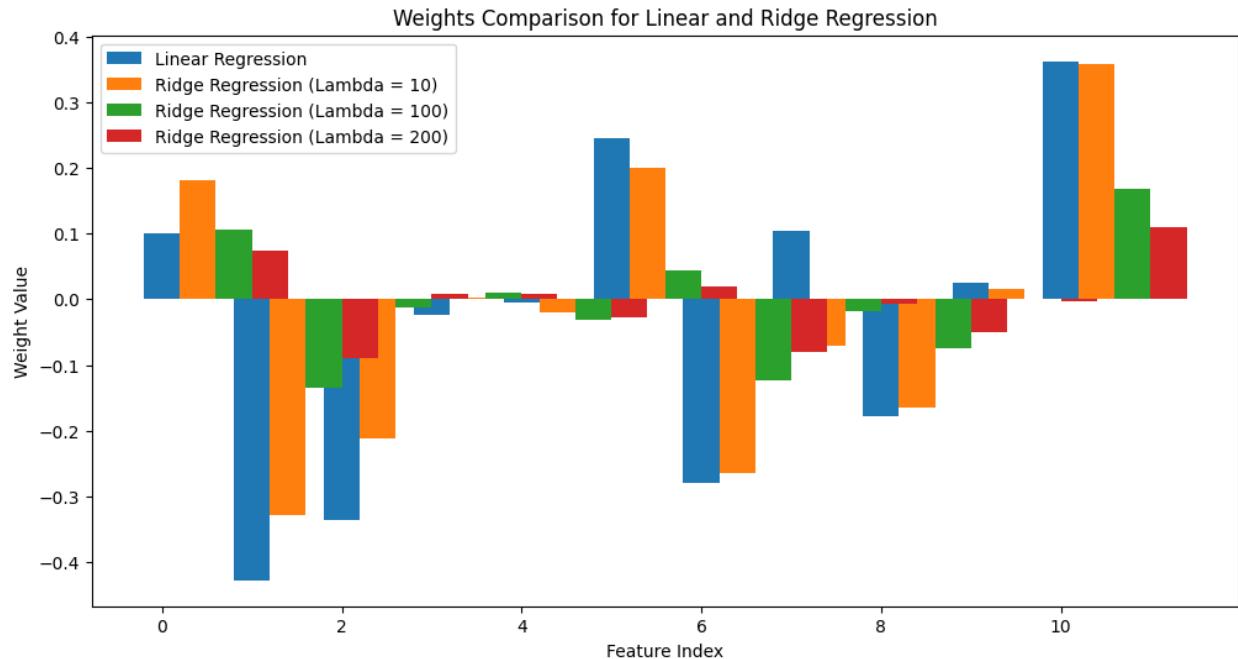
Weights for $\lambda = 200$:

```

w0: 0.07466473696029755
w1: -0.08964547279936119
w2: 0.008288733594097175
w3: 0.007402421147498225
w4: -0.02662987012203488
w5: 0.019048169412965672
w6: -0.08078192560860473
w7: -0.007586363543530622
w8: -0.04922597370338143
w9: -0.0027821410863150837
w10: 0.10948081147742252 $\Delta$ )

```

Στη συνέχεια, το κοινό γράφημα που παράγεται από τα προηγούμενα δεδομένα είναι:



E)

Root Mean Squared Error (RMSE) είναι μια τιμή που χρησιμοποιείται συνήθως για τη μέτρηση του μέσου μεγέθους των σφαλμάτων μεταξύ προβλεπόμενων και πραγματικών τιμών σε ένα πρόβλημα παλινδρόμησης. Παρέχει έναν τρόπο ποσοτικοποίησης της ακρίβειας ενός μοντέλου.

Για να το υπολογίσουμε θα κανουμε τα επόμενα βήματα:

1. Υπολογισμός της υπόλοιπης τιμής (error) :
υπόλοιπο = actual – predicted
2. Τετραγωνισμός υπολοίπου
3. Μέση τιμή τετραγωνισμένου υπολοίπου
$$\text{μέση τιμή} = \frac{1}{N} \sum_{i=1}^v (\text{τετραγωνισμένο υπολοίπο})$$
4. $RMSE = \sqrt{\text{μέση τιμη τετραγωνισμενου υπολοιπου}}$

Αυτό θα το κανουμε και για τα training και για τα validation sets.

```
rom sklearn.metrics import mean_squared_error
from math import sqrt

# Linear Regression
# Calculate predictions for the training set
y_train_pred_linear = X_train @ weights_linear_regression

# Calculate predictions for the test set
y_test_pred_linear = X_test @ weights_linear_regression
```

```

# Calculate RMSE for training set
rmse_train_linear = sqrt(mean_squared_error(y_train, y_train_pred_linear))
print(f"RMSE for Linear Regression (Training): {rmse_train_linear}")

# Calculate RMSE for test set
rmse_test_linear = sqrt(mean_squared_error(y_test, y_test_pred_linear))
print(f"RMSE for Linear Regression (Test): {rmse_test_linear}")
print()

# Ridge Regression
for lambda_val, weights in ridge_weights.items():
    # Calculate predictions for the training set
    y_train_pred_ridge = X_train @ weights

    # Calculate predictions for the test set
    y_test_pred_ridge = X_test @ weights

    # Calculate RMSE for training set
    rmse_train_ridge = sqrt(mean_squared_error(y_train,
y_train_pred_ridge))
    print(f"RMSE for Ridge Regression (Training) with λ = {lambda_val}:
{rmse_train_ridge}")

    # Calculate RMSE for test set
    rmse_test_ridge = sqrt(mean_squared_error(y_test, y_test_pred_ridge))
    print(f"RMSE for Ridge Regression (Test) with λ = {lambda_val}:
{rmse_test_ridge}")
    print()

```

και το αποτέλεσμα που παράγεται είναι:

RMSE for Linear Regression (Training): 0.7156631541673862

RMSE for Linear Regression (Test): 0.6819783710800131

RMSE for Ridge Regression (Training) with $\lambda = 10$: 0.7168834205261595
RMSE for Ridge Regression (Test) with $\lambda = 10$: 0.608263983248362

RMSE for Ridge Regression (Training) with $\lambda = 100$: 0.7978857359579214
RMSE for Ridge Regression (Test) with $\lambda = 100$: 0.5455839303648995

RMSE for Ridge Regression (Training) with $\lambda = 200$: 0.8363143974201055
RMSE for Ridge Regression (Test) with $\lambda = 200$: 0.584786817771439

Η επιλογή της κατάλληλης τιμής του λ χρειάζεται έναν συμβιβασμό μεταξύ της καλής προσαρμογής των δεδομένων εκπαίδευσης και της αποτροπής της υπερπροσαρμογής.

- Linear Regression ($\lambda = N/A$):

Training RMSE: 0.7156631541673862 Validation RMSE: 0.6819783710800131

Το μοντέλο γραμμικής παλινδρόμησης χωρίς κανονικοποίηση ($\lambda = N/A$) επιτυγχάνει ένα σχετικά χαμηλό RMSE τόσο στα σύνολα εκπαίδευσης όσο και στα σύνολα validation. Αυτό υποδηλώνει ότι το μοντέλο μπορεί να μην ταιριάζει υπερβολικά με τα δεδομένα εκπαίδευσης.

- Ridge Regression:

$\lambda = 10$: Training RMSE: 0.7168834205261595 Validation RMSE: 0.608263983248362

$\lambda = 100$: Training RMSE: 0.7978857359579214 Validation RMSE: 0.545583930364899

$\lambda = 200$: Training RMSE: 0.8363143974201055 Validation RMSE: 0.584786817771439

Το $\lambda = 10$ παρέχει κανονικοποίηση διατηρώντας τις σχετικά χαμηλές τιμές RMSE.

Επιτυγχάνει ισορροπία μεταξύ της προσαρμογής των δεδομένων προπόνησης και της αποτροπής της υπερπροσαρμογής.

ΑΣΚΗΣΗ 1.2

Ακολουθεί χειρόγραφη απάντηση στα ερωτήματα:

1. Ασύρμον 1.2

a) • Κυριότερη όψη ο ρυθμός που δίνει συνάρτησην πυκνότητας μεταβολής για την κανονική λαζαρούτη πεδίο λεπτήτες, δίνει την ανάτομη σχέση:

$$p(x_1, x_2) =$$

$$= \frac{1}{2\pi |\Sigma|} \exp \left(-\frac{1}{2|\Sigma|} [x_1 - t_1, x_2 - t_2] \begin{bmatrix} \sigma_2^2 & -\sigma_{12} \\ -\sigma_{12} & \sigma_1^2 \end{bmatrix}^{-1} \begin{bmatrix} x_1 - t_1 \\ x_2 - t_2 \end{bmatrix} \right)$$

$$= \frac{1}{2\pi \sqrt{|\Sigma|}} \exp \left(-\frac{1}{2|\Sigma|} [x_1 - t_1, x_2 - t_2] \begin{bmatrix} \sigma_2^2(x_1 - t_1) - \sigma_{12}(x_2 - t_2) \\ -\sigma_{12}(x_1 - t_1) + \sigma_1^2(x_2 - t_2) \end{bmatrix} \right)$$

$$= \frac{1}{2\pi \sqrt{|\Sigma|}} \exp \left(-\frac{1}{2|\Sigma|} (\sigma_2^2(x_1 - t_1)^2 - 2\sigma_{12}(x_1 - t_1)(x_2 - t_2) + \sigma_1^2(x_2 - t_2)^2) \right)$$

• Η συνάρτηση πυκνότητας μεταβολής πάνω στην αριθμητική σχέση:

$$p_{x_2}(x_2) = \int_{-\infty}^{+\infty} p(x_1, x_2) dx_1 = \int_{-\infty}^{+\infty} \frac{1}{2\pi \sqrt{|\Sigma|}} \exp \left(-\frac{1}{2|\Sigma|} \sigma_2^2(x_2 - t_2)^2 \right) \cdot$$

$$\exp \left(-\frac{1}{2|\Sigma|} (\sigma_2^2(x_2 - t_2)^2 - 2\sigma_{12}(x_1 - t_1)(x_2 - t_2)) \right) dx_1 =$$

$$= \frac{1}{2\pi \sqrt{|\Sigma|}} \exp \left(-\frac{1}{2|\Sigma|} \sigma_2^2(x_2 - t_2)^2 \right) \int_{-\infty}^{+\infty} \exp \left(-\frac{1}{2|\Sigma|} (\sigma_2^2(x_2 - t_2)^2 -$$

$$-2\sigma_{12}(x_1-t_1)(x_2-t_2) dx = C \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2|\varepsilon|}(\sigma_2^2(x_1-t_1)^2 -$$

$$-\frac{2\sigma_{12}}{\delta_2}(x_1-t_1)(x_2-t_2) + \frac{\sigma_{12}}{\delta_2}(x_2-t_2)^2 - \frac{\sigma_{12}}{\delta_2}(x_2-t_2)\right) dx$$

νέσα ανδ οδοκηρίου προειπει:

$$F_{X_2}(x_2) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2|\varepsilon|}(x_2-t_2)^2(\sigma_2^2 - \left(\frac{\sigma_{12}}{\delta_2}\right)^2)\right)$$

$$\text{οδοζε } p(X_1/x_2=a) = \frac{p(X_1, x_2=a)}{p_{X_2}(x_2=a)} = \\ = \frac{1}{\sqrt{2\pi} \sqrt{1|\varepsilon|}} \exp\left(-\frac{\delta_2^2}{2|\varepsilon|} \left(x_1-t_1 + (x_2-t_2) \frac{\sigma_{12}}{\delta_2^2}\right)^2\right)$$

που είναι της καρωνίδης / Gaussian καρωνής ή

$$\mu = t_1 - \frac{(a-t_2)\sigma_{12}}{\delta_2^2} \quad \text{και} \quad \sigma^2 = \frac{1|\varepsilon|}{\delta_2^2} = \frac{\sigma_1^2 \sigma_2^2 - \sigma_{12}^2}{\delta_2^2}$$

B) Οι χρονολογίες των διατηρίου που συνοւν αυτά τα δεδομένα

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \quad \Sigma_{11} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 2 \end{bmatrix}$$

$$\Sigma_{22} = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$

$$\Sigma_{21} = \begin{bmatrix} 0.5 & 1 \end{bmatrix}$$

$$\Sigma_{33} = [3]$$

$$\text{Kai } \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$$

$$\rightarrow \text{Geod}\varepsilon: \hat{\mu} = \mu + \varepsilon_{12} \varepsilon_{22}^{-1} (\mu - \mu_2) = \\ = \begin{bmatrix} -\frac{11}{6} \\ -\frac{1}{3} \end{bmatrix}$$

$$\Rightarrow \hat{\varepsilon} = \varepsilon_{11} - \varepsilon_{12} \cdot \varepsilon_{22}^{-1} \varepsilon_{21} = \\ = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 2 \end{bmatrix} - \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ \frac{1}{3} \end{bmatrix} \begin{bmatrix} 0.5 & 1 \end{bmatrix} = \\ = \dots = \begin{bmatrix} \frac{13}{12} & \frac{29}{30} \\ \frac{13}{30} & \frac{5}{3} \end{bmatrix}$$

kan jwpijouze in even kanonini kuzaveti
 $N(\mu, \varepsilon)$.

$$d) \mu = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, \quad \varepsilon = \begin{bmatrix} 1 & 0.5 & 0.8 \\ 0.5 & 3 & 1 \\ 0.8 & 1 & 2 \end{bmatrix}$$

$$\hat{\varepsilon}_{\text{Koutz}}: \hat{\mu} = \begin{bmatrix} -2 \\ 2 \end{bmatrix} - \begin{bmatrix} 0.8 \\ 1 \end{bmatrix} \frac{1}{2} (1-0) = \begin{bmatrix} -1.6 \\ 2.5 \end{bmatrix}$$

$$\text{Kai } \hat{\varepsilon} = \varepsilon_{11} - \varepsilon_{12} \cdot \varepsilon_{22}^{-1} \varepsilon_{21} =$$

$$= \begin{bmatrix} 1 & 0,5 \\ 0,5 & 3 \end{bmatrix} - \begin{bmatrix} 0,8 \\ 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 0,8 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 0,65 & 0,1 \\ 0,1 & 2,5 \end{bmatrix}, \text{ oforus aronfci}$$

Kazavoch Gauss te NCF, ε^1).

ΑΣΚΗΣΗ 1.3

(α)

Για τα σημεία δύο κλάσεων που ακολοθούν κανονική κατανομή με κοινούς πίνακες συμμεταβλητότητας για τις δύο κλάσεις ισχύει σύμφωνα με τις σημειώσεις της αντίστοιχης ενότητας :

Επειδή $P(\omega_1) = P(\omega_2) = 0.5$ προκύπτει τελικά,

$$x_0 = (\mu_1 + \mu_2)/2 ,$$

$$\theta = \Sigma^{-1} (\mu_1 - \mu_2),$$

$$g(x) = \theta^T (x - x_0) = 0.$$

Επίσης $\Sigma = I$.

Με βάση τα παραπάνω και τα δεδομένα της άσκησης ακολουθεί ο παρακάτω κώδικας για υπολογισμό του Decision Boundary.

```
import numpy as np
```

```
# Given parameters
```

```
mu1 = np.array([-2, 0])
```

```
mu2 = np.array([2, 1])
```

```
cov_matrix = np.eye(2) # Common covariance matrix I
```

```
P_1 = 0.5
```

```
P_2 = 0.5
```

```
# Calculate the weight vector (w) and bias term (b)
```

```
cov_matrix_inv = np.linalg.inv(cov_matrix)
```

```
bT = 2*np.dot((mu2 - mu1).T,cov_matrix_inv)
```

```
c = np.dot(mu1.T, np.dot(cov_matrix_inv, mu1)) - np.dot(mu2.T, np.dot(cov_matrix_inv, mu2))
- np.log(P_1/P_2)
```

```
# Print the decision boundary formula
```

```
print(f"Decision Boundary Formula: {bT[0]} * x1 + {bT[1]} * x2 + ({c}) = 0")
```

Decision Boundary Formula: $8.0 * x1 + 2.0 * x2 + -1.0 = 0$

$\Delta \eta \lambda \alpha \delta \dot{\gamma}, 8x1 + 2x2 - 1 = 0.$

(β)

```
import numpy as np
```

```
# Given parameters
```

```
mu1 = np.array([-2, 0])
```

```
mu2 = np.array([2, 1])
```

```
cov_matrix = np.array([[1, -0.6], [-0.6, 1]]) # Common covariance matrix
```

```
P_1 = 0.5
```

```
P_2 = 0.5
```

```
# Calculate the weight vector (w) and bias term (b)
```

```
cov_matrix_inv = np.linalg.inv(cov_matrix)
```

```
bT = 2*np.dot((mu2 - mu1).T, cov_matrix_inv)
```

```
c = np.dot(mu1.T, np.dot(cov_matrix_inv, mu1)) - np.dot(mu2.T, np.dot(cov_matrix_inv, mu2))
- np.log(P_1 / P_2)
```

```
# Print the decision boundary formula
```

```
print(f"Decision Boundary Formula: {bT[0]} * x1 + {bT[1]} * x2 + ({c}) = 0")
```

Decision Boundary Formula: $14.375 * x1 + 10.625 * x2 + (-5.3125) = 0$

Δηλαδή, $14.375 x1 + 10.625 x2 - 5.3125 = 0$

(γ)

Λόγω των συντελεστών βαρύτητας των σφαλμάτων ταξινόμησης ισχύει για ταξινόμηση στην ω_1 κλάση χωρίς βλάβη της γενικότητας:

$\lambda_{12} p(\omega_1) p(x | \omega_1) > \lambda_{21} p(\omega_2) p(x | \omega_2) \Rightarrow$

$\Rightarrow 2 p(x | \omega_1) - p(x | \omega_2) > 0$

Οπότε η εξίσωση $g(x) = p(\omega_1 | x) - p(\omega_2 | x) = 0$ για την υπερεπιφάνεια απόφασης θα γίνει

$g(x) = \ln[p(x|\omega_1)/p(x|\omega_2)] - \ln[p(\omega_1)/p(\omega_2)] + \ln 2 = 0$

Τελικά, προκύπτει :

```
import numpy as np
```

```
# Given parameters
```

```
mu1 = np.array([-2, 0])
```

```
mu2 = np.array([2, 1])
```

```
cov_matrix = np.array([[1, -0.6], [-0.6, 1]]) # Common covariance matrix
```

```
P_1 = 0.5
```

```
P_2 = 0.5
```

```
# Calculate the weight vector (w) and bias term (b)
```

```
cov_matrix_inv = np.linalg.inv(cov_matrix)
```

```

bT = 2*np.dot((mu2 - mu1).T,cov_matrix_inv)

c = np.dot(mu1.T, np.dot(cov_matrix_inv, mu1)) - np.dot(mu2.T, np.dot(cov_matrix_inv, mu2))
- np.log(P_1 / P_2)+ np.log(2)

# Print the decision boundary formula
print(f"Decision Boundary Formula: {bT[0]} * x1 + {bT[1]} * x2 + {c} = 0")

Output :
Decision Boundary Formula: 14.375 * x1 + 10.625 * x2 + (-4.619352819440055) = 0
Δηλαδή, 14.375 x1 + 10.625 x2 – 4.619352819440055 = 0

(δ)
Ο κώδικας για τα ζητούμενα είναι ο ακόλουθος :

# Generate 200 points for each class
def generate_points(mu, sigma, num_points):
    return np.random.multivariate_normal(mu, sigma, num_points)

mu1 = np.array([-2, 0])
mu2 = np.array([2, 1])

#case 1
cov_matrix = np.eye(2) # Common covariance matrix I
P_1 = 0.5
P_2 = 0.5

# Calculate the weight vector (w) and bias term (b)
cov_matrix_inv = np.linalg.inv(cov_matrix)

bT_a = 2*np.dot((mu2 - mu1).T,cov_matrix_inv)

```

```
c_a = np.dot(mu1.T, np.dot(cov_matrix_inv, mu1)) - np.dot(mu2.T, np.dot(cov_matrix_inv, mu2)) - np.log(P_1 / P_2)
```

```
x_class1_a = generate_points(mu1, cov_matrix, 200)
```

```
x_class2_a = generate_points(mu2, cov_matrix, 200)
```

```
#case 2
```

```
cov_matrix = np.array([[1, -0.6],[-0.6, 1]]) # Common covariance matrix
```

```
P_1 = 0.5
```

```
P_2 = 0.5
```

```
cov_matrix_inv = np.linalg.inv(cov_matrix)
```

```
bT_b = 2*np.dot((mu2 - mu1).T,cov_matrix_inv)
```

```
c_b = np.dot(mu1.T, np.dot(cov_matrix_inv, mu1)) - np.dot(mu2.T, np.dot(cov_matrix_inv, mu2)) - np.log(P_1 / P_2)
```

```
x_class1_b = generate_points(mu1, cov_matrix, 200)
```

```
x_class2_b = generate_points(mu2, cov_matrix, 200)
```

```
#case 3
```

```
cov_matrix = np.array([[1, -0.6],[-0.6, 1]]) # Common covariance matrix
```

```
P_1 = 0.5
```

```
P_2 = 0.5
```

```
cov_matrix_inv = np.linalg.inv(cov_matrix)
```

```

bT_c = 2*np.dot((mu2 - mu1).T,cov_matrix_inv)

c_c = np.dot(mu1.T, np.dot(cov_matrix_inv, mu1)) - np.dot(mu2.T, np.dot(cov_matrix_inv,
mu2)) - np.log(P_1 / P_2) + np.log(2)

x_class1_c = generate_points(mu1, cov_matrix, 200)
x_class2_c = generate_points(mu2, cov_matrix, 200)
import matplotlib.pyplot as plt

# Function to plot the points, class centers, and decision boundaries
def plot_points_decision_boundary(x_class1, x_class2, mu1, mu2, bT, c):
    # Plot the points
    plt.scatter(x_class1[:, 0], x_class1[:, 1], color='blue', label='Class 1')
    plt.scatter(x_class2[:, 0], x_class2[:, 1], color='red', label='Class 2')

    # Plot the class centers
    plt.scatter(mu1[0], mu1[1], color='green', marker='x', label='Class 1 Center')
    plt.scatter(mu2[0], mu2[1], color='orange', marker='x', label='Class 2 Center')

    # Plot the decision boundary
    x = np.linspace(-6, 6, 100)
    y = np.dot(bT,x) + c
    y = (-bT[0] * x - c) / bT[1]
    plt.plot(x, y, color='black', label='Decision Boundary')

    # Set plot labels and legend
    plt.xlabel('x1')
    plt.ylabel('x2')

```

```
plt.legend()

# Show the plot
plt.show()

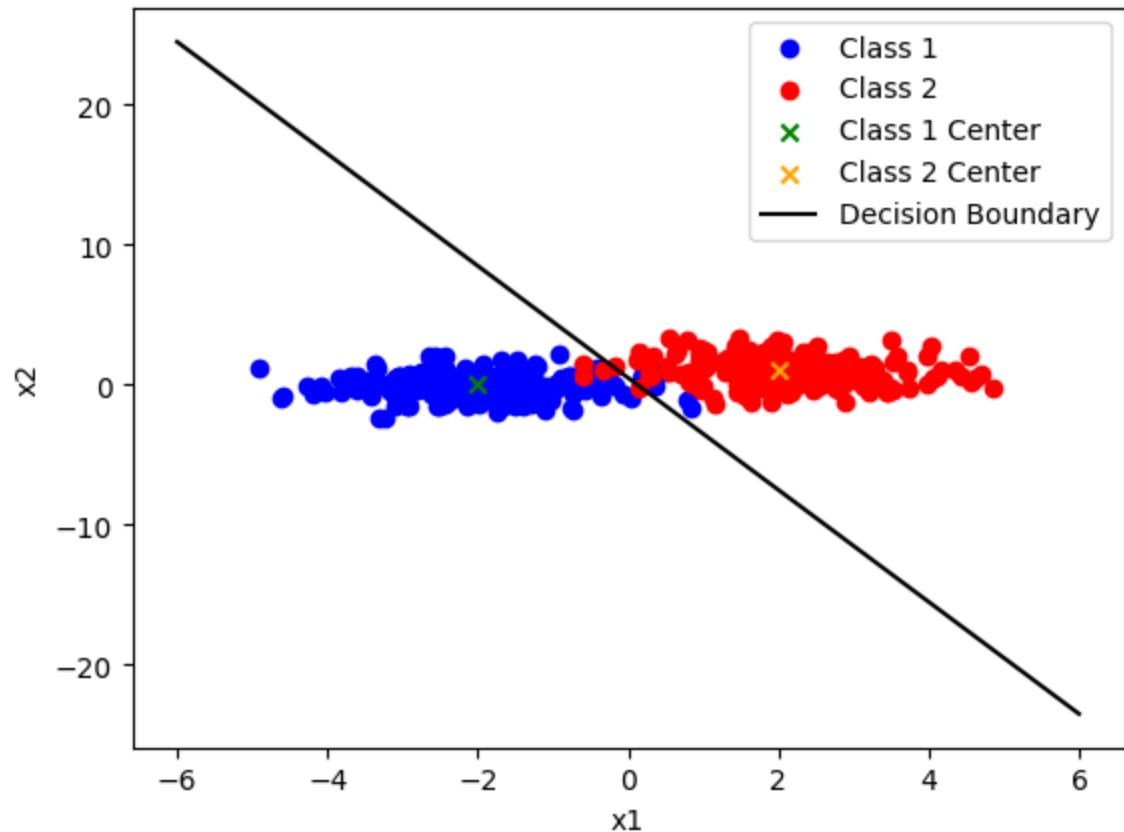
# Plot for case 1
plot_points_decision_boundary(x_class1_a, x_class2_a, mu1, mu2, bT_a, c_a)

# Plot for case 2
plot_points_decision_boundary(x_class1_b, x_class2_b, mu1, mu2, bT_b, c_b)

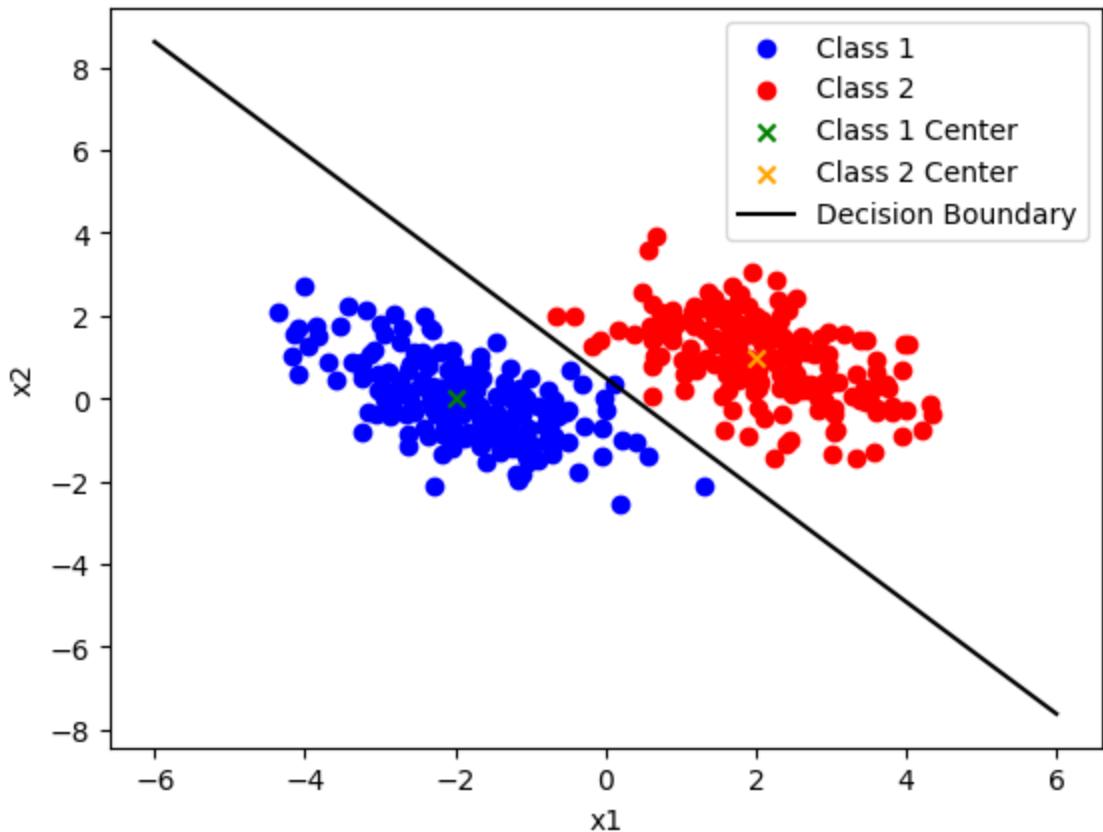
# Plot for case 3
plot_points_decision_boundary(x_class1_b, x_class2_b, mu1, mu2, bT_c, c_c)
```

Τα διαγράμματα που ακολουθούν είναι τα εξής :

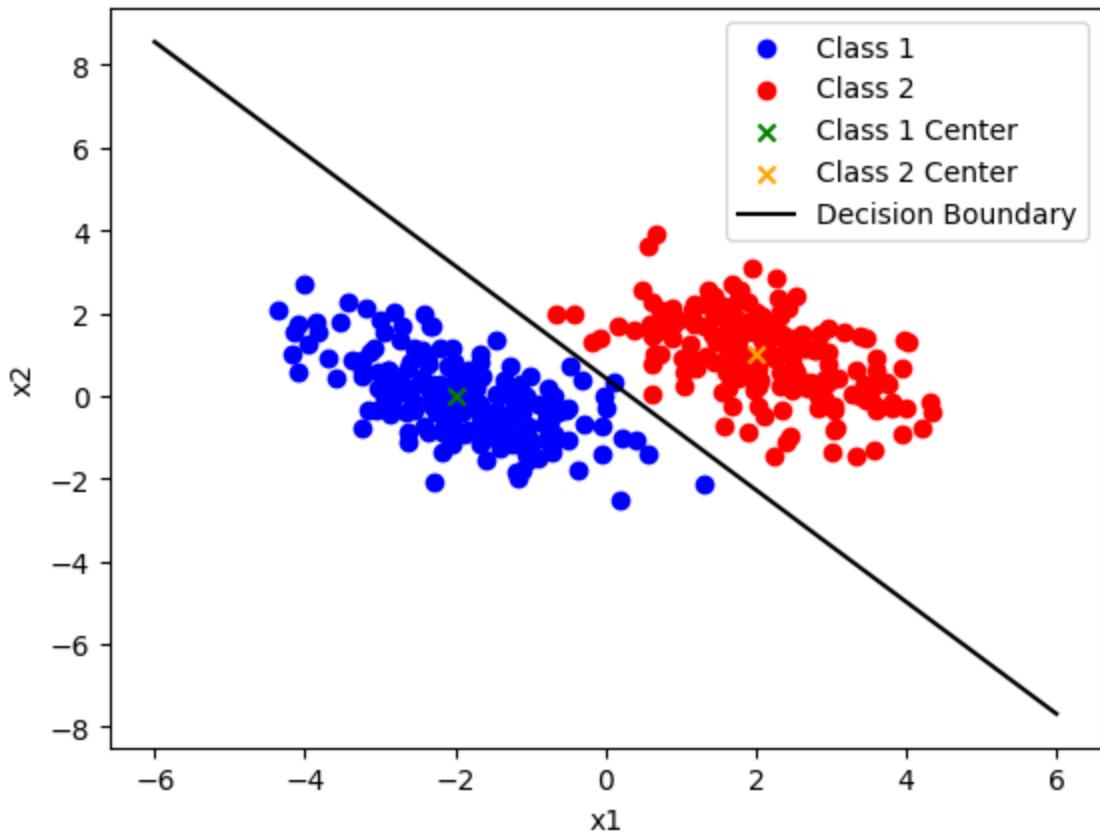
A)



B)



Γ)



ΑΣΚΗΣΗ 1.4

A)

Θα χρησιμοποιήσουμε τον αλγόριθμο perceptron με $(\omega_1, \omega_2, \omega_3, \omega_4) = (1, 0, 0, 0)^T = T^T$

$$\Delta \text{είγμα} : \chi^{(1)} = (1, 4, 3, 6)^T$$

Έξοδος του perceptron: $z = T^T \cdot \chi^{(1)} = (1, 0, 0, 0)^T \cdot (1, 4, 3, 6)^T = 1$, $\text{step}(2)=1$ για $z \geq 0$, 0 για $z < 0$

$$y = \text{step}(1) = 1 \text{ ára } y = 1$$

Άρα από τα αποτελέσματα της κλάσης έχουμε false positive (FP)

Διάνυσμα μεταβολής βαρών: error = 0 - y = 0 - 1 = -1

$$\Delta T = n \cdot e \cdot x^{(1)} = (-1, -4, -3, -6)^T$$

Και το ανανεωμένο διάνυσμα βαρών είναι $T = T + \Delta T = (1, 0, 0, 0)^T + (-1, -4, -3, -6)^T = (0, -4, -3, -6)^T$

Δείγμα: $x^{(2)} = (1, 2, -2, 3)^T$

Έξοδος του perceptron: $z = T^T \cdot \chi^{(2)} = (0, -4, -3, -6)^T \cdot (1, 2, -2, 3)^T = -20$

Άρα $y = \text{step}(-20) = 0$ άρα $y = 0$

Το δείγμα είναι false negative (FN)

Διάνυσμα μεταβολής βαρών: error = $1 - y = 1 - 0 = 1$

$\Delta T = n \cdot e \cdot x^{(2)} = (1, 2, -2, 3)^T$

Και το ανανεωμένο διάνυσμα βαρών είναι $T = T + \Delta T = (0, -4, -3, -6)^T + (1, 2, -2, 3)^T = (1, -2, -5, -3)^T$

Δείγμα: $x^{(3)} = (1, 1, 0, -3)^T$

Έξοδος του perceptron: $z = T^T \cdot \chi^{(3)} = (1, -2, -5, -3)^T \cdot (1, 1, 0, -3)^T = 8$

Άρα $y = \text{step}(8) = 1$ άρα $y = 1$

Το δείγμα είναι true positive (TP)

Διάνυσμα μεταβολής βαρών: error = $1 - y = 1 - 1 = 0$

$\Delta T = n \cdot e \cdot x^{(3)} = (0, 0, 0, 0)^T$

Και το ανανεωμένο διάνυσμα βαρών είναι $T = T + \Delta T = (1, -2, -5, -3)^T + (0, 0, 0, 0)^T = (1, -2, -5, -3)^T$

Δείγμα: $x^{(4)} = (1, 4, 2, 3)^T$

Έξοδος του perceptron: $z = T^T \cdot \chi^{(4)} = (1, -2, -5, -3)^T \cdot (1, 4, 2, 3)^T = -26$

Άρα $y = \text{step}(-26) = 0$ άρα $y = 0$

Το δείγμα είναι true negative (TN)

Διάνυσμα μεταβολής βαρών: error = $0 - y = 0 - 0 = 0$

$\Delta T = n \cdot e \cdot x^{(4)} = (0, 0, 0, 0)^T$

Και το ανανεωμένο διάνυσμα βαρών είναι $T = T + \Delta T = (1, -2, -5, -3)^T + (0, 0, 0, 0)^T = (1, -2, -5, -3)^T$

Εποχή	Δείγμα	Ετικέτα	$\sum w_i x_i$	Έξοδος γ	Χαρακτηρισμός	ΔT	T
1	(1, 4, 3, 6)	0	1	1	FP	(-1, -4, -3, -6)	(0, -4, -3, -6)
1	(1, 2, -2, 3)	1	-20	0	FN	(1, 2, -2, 3)	(1, -2, -5, -3)

1	(1,1,0,-3)	1	8	1	TP	(0,0,0,0)	(1,-2,-5,-3)
1	(1,4,2,3)	0	-26	0	TN	(0,0,0,0)	(1,-2,-5,-3)
2	(1,4,3,6)	0	-40	0	TN	(0,0,0,0)	(1,-2,-5,-3)
2	(1,2,-2,3)	1	-2	0	FN	(1,2,-2,3)	(2,0,-7,0)
2	(1,1,0,-3)	1	2	1	TP	(0,0,0,0)	(2,0,-7,0)
2	(1,4,2,3)	0	-12	0	TN	(0,0,0,0)	(2,0,-7,0)
3	(1,4,3,6)	0	-19	0	TN	(0,0,0,0)	(2,0,-7,0)
3	(1,2,-2,3)	1	16	1	TP	(0,0,0,0)	(2,0,-7,0)
3	(1,1,0,-3)	1	2	1	TP	(0,0,0,0)	(2,0,-7,0)
3	(1,4,2,3)	0	-12	0	TN	(0,0,0,0)	(2,0,-7,0)

B)

Η περιοχή $x_1 \geq 2$ θα ταξινομηθεί από τον πρώτο νευρώνα, με την οριακή εξίσωση του νευρώνα 1 να είναι $x_1 - 2 = 0$. Γενικά, ισχύει η εξίσωση $w_{11}x_1 + w_{12}x_2 + b_1 = x_1 - 2$.

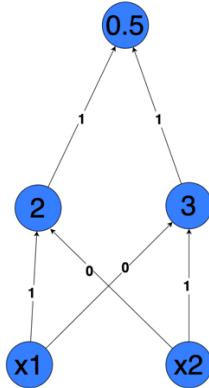
Για την επιλογή των βαρών, θα θέσουμε $w_{11} = 1$ και $w_{12} = 0$, καθώς το x_1 συμβάλει στη λήψη απόφασης, σε αντίθεση με το x_2 το οποίο δεν συμβάλει. Επομένως, η εξίσωση μειώνεται σε $w_{11}x_1 + b_1 = x_1 + b_1$. Δεδομένου ότι $w_{11}x_1 + b_1 = x_1 - 2$, προκύπτει ότι $b_1 = -2$.

Η περιοχή $x_2 \geq 3$ θα ταξινομηθεί από τον δεύτερο νευρώνα, με την οριακή εξίσωση του νευρώνα 2 να είναι $x_2 - 3 = 0$. Γενικά, ισχύει η εξίσωση $w_{21}x_1 + w_{22}x_2 + b_2 = x_2 - 3$.

Για την επιλογή των βαρών, θα θέσουμε $w_{21} = 0$ και $w_{22} = 1$, καθώς το x_1 δεν συμβάλει στη λήψη απόφασης, σε αντίθεση με το x_2 το οποίο συμβάλει. Έτσι, η εξίσωση μειώνεται σε $w_{21}x_1 + w_{22}x_2 + b_2 = x_2 + b_2$. Δεδομένου ότι $w_{21}x_1 + w_{22}x_2 + b_2 = x_2 - 3$, προκύπτει ότι $b_2 = -3$.

Ο τρίτος νευρώνας θα συνδυάσει τους προηγούμενους δύο για να εξάγει το τελικό αποτέλεσμα. Έστω $'a_1'$ το αποτέλεσμα του πρώτου νευρώνα και $'a_2'$ το αποτέλεσμα του δεύτερου νευρώνα. Τα αποτέλεσματα είναι 0 ή 1, με 0 αν ταξινομείται στην αρνητική κλάση και 1 αν ταξινομείται στη θετική.

Άρα, $w_{1a_1} + w_{2a_2} + b_3 = 0$. Θέλω $w_1 = w_2 = 1$, αφού εξίσου τα $'a_1'$ και $'a_2'$ συμβάλλουν στη λήψη απόφασης, και $b_3 = -0.5$, αφού έστω και ένα γινόμενο (w_{1a_1}, w_{2a_2}) να είναι '1', ταξινομείται στη θετική κλάση. Άρα συνολικά:



ΑΣΚΗΣΗ 1.5

Γραμμικό SVM:

- Χρησιμοποιείται για γραμμικά διαχωρίσιμα δεδομένα, όπου μπορούν να ταξινομηθούν σε δύο κλάσεις με μια ευθεία γραμμή.
- Παρατηρείται στα (3) και (4).

Μη Γραμμικό SVM:

- Χρησιμοποιείται για μη γραμμικά διαχωρίσιμα δεδομένα.
- Εφαρμόζεται στα (1), (2), (5) και (6).

Παράμετρος C:

- Είναι παράμετρος τακτοποίησης που επηρεάζει την αντιστάθμιση μεταξύ του χαμηλού σφάλματος εκπαίδευσης και του χαμηλού σφάλματος δοκιμής.
- Όσο μικρότερο είναι το C, τόσο ενθαρρύνει ένα ευρύτερο περιθώριο αλλά αποδέχεται περισσότερα λάθη εκπαίδευσης.
- Όσο μεγαλύτερο είναι το C, οδηγεί σε μικρότερο περιθώριο αλλά λιγότερα λάθη εκπαίδευσης.
- Άρα, αν `a -> 4` και `b -> 3`.

Μη Γραμμικοί Πυρήνες SVM:

- Οι μη γραμμικοί πυρήνες SVM (γ , δ , ε) διαφέρουν στις μαθηματικές τους μορφές και εφαρμόζονται μεταξύ τους ανάλογα με την πολυπλοκότητα του προβλήματος.

Radial-Basis Function (RBF):

- Η συνάρτηση RBF ορίζεται ως: $\exp(1/(2\sigma^2) | |x-x_1| |^2)$
- Αντιστοιχεί στα (δ) και (ε),
- Αντιστοιχεί στα σχήματα (1) και (6).

Παράμετρος σ:

- Ελέγχει το πλάτος της καμπύλης σε ένα σχήμα καμπάνας Gauss. Επηρεάζει το όριο απόφασης.
- Όταν το σ είναι μικρό, η καμπύλη Gauss στενεύει, και το όριο απόφασης τείνει να ακολουθεί τις μικρές λεπτομέρειες των δεδομένων εκπαίδευσης, δημιουργώντας ένα πιο περίπλοκο όριο απόφασης.
- Αντιθέτως, όσο μεγαλύτερο είναι σ, καθιστά τον πυρήνα πιο σφαιρικό, καθώς το όριο είναι πιο ομαλό. Άρα $\delta \rightarrow 1$ $\varepsilon \rightarrow 6$
- Όταν το σ αυξάνεται, το $1/2\sigma^2$ μειώνεται.

Μη Γραμμικό SVM:

- Ο πυρήνας $k(u,v) = u^T v + (u^T v)^2$ καταγράφει τετραγωνικές σχέσεις.
- Το όριο απόφασης αντιτροσωπεύει μια καμπύλη διαχωρισμού στον χώρο εισόδου, άρα $\gamma > 2$.

Οπότε ισχύει

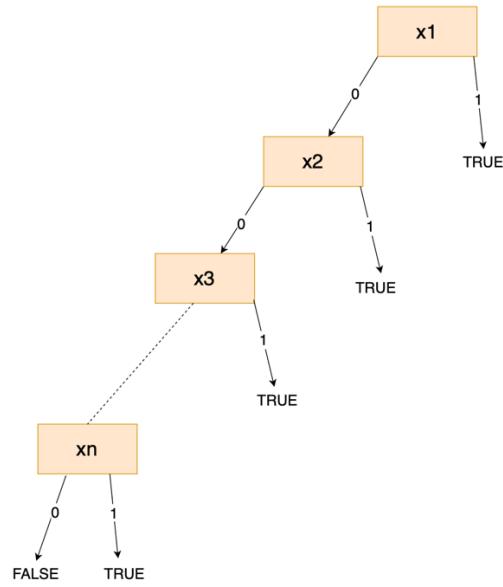
- $\alpha > 4$
- $\beta > 3$
- $\gamma > 2$
- $\delta > 1$
- $\varepsilon > 6$
- $\sigma \rightarrow 5$

ΑΣΚΗΣΗ 1.6

Μέρος 1:

Χωρίς βλάβη της γενικότητας θεωρώ: $f_n = (x_1 \vee x_2) \vee (x_3 \vee x_4) \vee \dots \vee (x_{n-1} \vee x_n)$

Για το δέντρο απόφασης έχουμε:

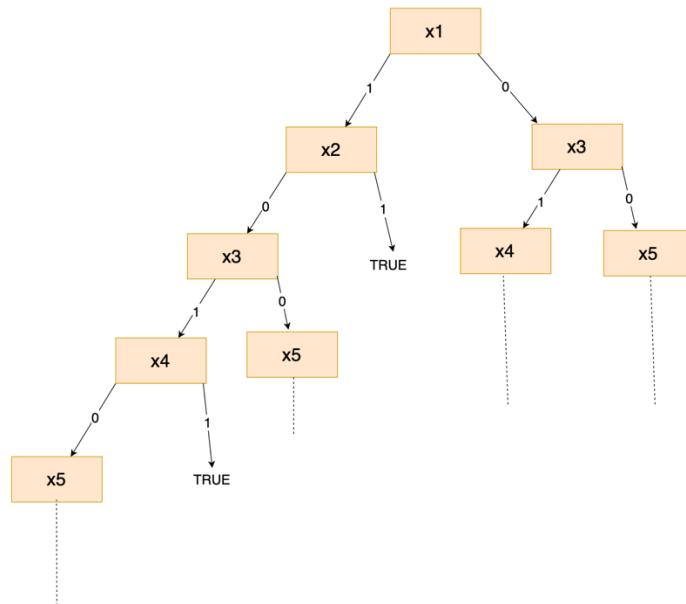


Παρατηρούμε ότι το δέντρο απόφασης έχει μέγιστο μήκος n και προσεγγίζει τον f_n με ακρίβεια 1.

Μέρος 2:

Χωρίς βλάβη της γενικότητας θεωρώ: $f_n = (x_1 \wedge x_2) \vee (x_3 \wedge x_4) \vee \dots \vee (x_{n-1} \wedge x_n)$

Για το δέντρο απόφασης έχουμε:



B)

1.

Υπολογισμός με gini:

$$gini(\text{Root}) = 1 - (5/10)^2 - (5/10)^2 = 0.5$$

Για το x1:

$$gini(x1 = 1) = 1 - (3/5)^2 - (2/5)^2 = 12/25$$

$$gini(x1 = 0) = 1 - (2/5)^2 - (3/5)^2 = 12/25$$

$$gini(x1) = (5/10) gini(x1 = 1) + (5/10) gini(x1 = 0) = 12/25$$

$$ig(x1) = gini(\text{Root}) - gini(x1) = 0.02$$

Για το x2:

$$gini(x2 = 1) = 1 - (2/3)^2 - (1/3)^2 = 4/9$$

$$gini(x2 = 0) = 1 - (3/7)^2 - (4/7)^2 = 24/49$$

$$gini(x2) = (3/10) gini(x2 = 1) + (7/10) gini(x2 = 0) = 10/21$$

$$ig(x2) = gini(\text{Root}) - gini(x2) = 0.0238$$

Για το x3:

$$gini(x3 = 1) = 1 - (3/6)^2 - (3/6)^2 = 0.5$$

$$gini(x3 = 0) = 1 - (2/4)^2 - (2/4)^2 = 0.5$$

$$gini(x3) = (6/10) gini(x3 = 1) + (4/10) gini(x3 = 0) = ig(x3) = gini(\text{Root}) - gini(x3) = 0$$

Για το x4:

$$gini(x4 = 1) = 1 - (2/3)^2 - (1/3)^2 = 4/9$$

$$gini(x4 = 0) = 1 - (3/7)^2 - (4/7)^2 = 24/49$$

$$gini(x4) = (3/10) gini(x4 = 1) + (7/10) gini(x4 = 0) = 10/21$$

$$ig(x4) = gini(\text{Root}) - gini(x4) = 0.0238$$

$$ig(x2) = ig(x4) > ig(x1) > ig(x3)$$

Το x2 έχει το υψηλότερο ig οπότε το χρησιμοποιούμε ως root.

Συνεχίζουμε την ίδια διαδικασία για το αριστερό παιδί της ρίζας (για $x_2=0$):

$$\text{gini}(\text{Root}) = 1 - (3/7)^2 - (4/7)^2 = 0.489$$

Για το x_1 :

$$\text{gini}(x_1 = 1) = 1 - (3/5)^2 - (2/5)^2 = 12/25$$

$$\text{gini}(x_1 = 0) = 1 - 0 - (2/2)^2 = 0$$

$$\text{gini}(x_1) = (5/7) \text{ gini}(x_1 = 1) + (2/7) \text{ gini}(x_1 = 0) = (12/25) * (5/7)$$

$$\text{ig}(x_1) = \text{gini}(\text{Root}) - \text{gini}(x_1) = 0.1461$$

Για το x_3 :

$$\text{gini}(x_3 = 1) = 1 - (3/5)^2 - (2/5)^2 = 0.48$$

$$\text{gini}(x_3 = 0) = 1 - (2/2)^2 - (0/2)^2 = 0$$

$$\text{gini}(x_3) = (5/7) \text{ gini}(x_3 = 1) + (2/7) \text{ gini}(x_3 = 0) = (12/25) * (5/7)$$

$$\text{ig}(x_3) = \text{gini}(\text{Root}) - \text{gini}(x_3) = 0.1461$$

Για το x_4 :

$$\text{gini}(x_4 = 1) = 1 - (1/2)^2 - (1/2)^2 = 0.5$$

$$\text{gini}(x_4 = 0) = 1 - (2/5)^2 - (3/5)^2 = 0.48$$

$$\text{gini}(x_4) = (2/7) \text{ gini}(x_4 = 1) + (5/7) \text{ gini}(x_4 = 0) = 0.4857$$

$$\text{ig}(x_4) = \text{gini}(\text{Root}) - \text{gini}(x_4) = 0.003$$

$$\text{ig}(x_1) = \text{ig}(x_3) > \text{ig}(x_4)$$

Το x_1 έχει το υψηλότερο ig οπότε το χρησιμοποιούμε ως root (κάτω από τον x_2 για $x_2=0$).

Συνεχίζουμε την ίδια διαδικασία για το αριστερό παιδί της ρίζας (για $x_2=1$):

$$\text{gini}(\text{Root}) = 1 - (2/3)^2 - (1/3)^2 = 0.444$$

Για το x1:

$$\text{gini}(x1 = 1) = 1 - 0 = 1$$

$$\text{gini}(x1 = 0) = 1 - (2/3)^2 - (1/3)^2 = 0.444$$

$$\text{gini}(x1) = 0 * \text{gini}(x1 = 1) + (3/3) \text{gini}(x1 = 0) = 0.444$$

$$\text{ig}(x1) = \text{gini}(\text{Root}) - \text{gini}(x1) = 0$$

Για το x3:

$$\text{gini}(x3 = 1) = 1 - (1/1)^2 = 0$$

$$\text{gini}(x3 = 0) = 1 - (2/2)^2 = 0$$

$$\text{gini}(x3) = (1/3) \text{gini}(x3 = 1) + (2/3) \text{gini}(x3 = 0) = 0$$

$$\text{ig}(x3) = \text{gini}(\text{Root}) - \text{gini}(x3) = \text{gini}(\text{Root})$$

Δεν χρειάζεται να προχωρήσουμε άλλο καθώς $\text{ig}(x3) = \text{gini}(\text{Root})$.

Συνεχίζουμε την ίδια διαδικασία για το αριστερό παιδί του (για $x1=1$):

$$\text{gini}(\text{Root}) = 1 - (3/5)^2 - (2/5)^2 = 0.48$$

Για το x3:

$$\text{gini}(x3 = 1) = 1 - (3/4)^2 - (1/4)^2 = 0.375$$

$$\text{gini}(x3 = 0) = 1 - (1/1)^2 = 0$$

$$\text{gini}(x3) = (4/5) \text{gini}(x3 = 1) + (1/5) \text{gini}(x3 = 0) = 0.3$$

$$\text{ig}(x3) = \text{gini}(\text{Root}) - \text{gini}(x3) = 0.16$$

Για το x4:

$$\text{gini}(x4 = 1) = 1 - (1/2)^2 - (1/2)^2 = 0.5$$

$$\text{gini}(x4 = 0) = 1 - (2/3)^2 - (1/3)^2 = 0.4444$$

$$\text{gini}(x4) = (2/5) \text{gini}(x4 = 1) + (3/5) \text{gini}(x4 = 0) = 0.4664$$

$$\text{ig}(x4) = \text{gini}(\text{Root}) - \text{gini}(x4) = 0.0136$$

$$\text{ig}(x3) > \text{ig}(x4)$$

Το x_3 έχει το υψηλότερο ig οπότε το χρησιμοποιούμε ως root (κάτω από τον x_1 για $x_2=1$).

Συνεχίζουμε την ίδια διαδικασία για το δεξί παιδί του x_1 (για $x_1=0$):

$$\text{gini}(\text{Root}) = 1 - (2/2)^2 = 0$$

Αφού $\text{gini}(\text{root}) = 0$ δε χρειάζεται να συνεχίσουμε τους υπολογισμούς. Για $x_1=0$ η απάντηση είναι -1.

Συνεχίζουμε την ίδια διαδικασία για το δεξί παιδί του x_3 (για $x_3=0$):

$$\text{gini}(\text{Root}) = 1 - (1/1)^2 = 0$$

Αφού $\text{gini}(\text{root}) = 0$ δε χρειάζεται να συνεχίσουμε τους υπολογισμούς. Για $x_3=0$ η απάντηση είναι -1.

Συνεχίζουμε την ίδια διαδικασία για το αριστερό παιδί του x_3 (για $x_3=1$):

$$\text{gini}(\text{Root}) = 1 - (3/4)^2 - (1/4)^2 = 0.375$$

Για το x_4 :

$$\text{gini}(x_4 = 1) = 1 - (1/2)^2 - (1/2)^2 = 0.5$$

$$\text{gini}(x_4 = 0) = 1 - (2/2)^2 = 0$$

$$\text{gini}(x_4) = (2/4) \text{ gini}(x_4 = 1) + (2/4) \text{ gini}(x_4 = 0) = 0.25$$

$$\text{ig}(x_4) = \text{gini}(\text{Root}) - \text{gini}(x_4) = 0.125$$

Θεωρούμε σαν root το x_4 .

Συνεχίζουμε την ίδια διαδικασία για το δεξί παιδί του x_4 (για $x_4=0$):

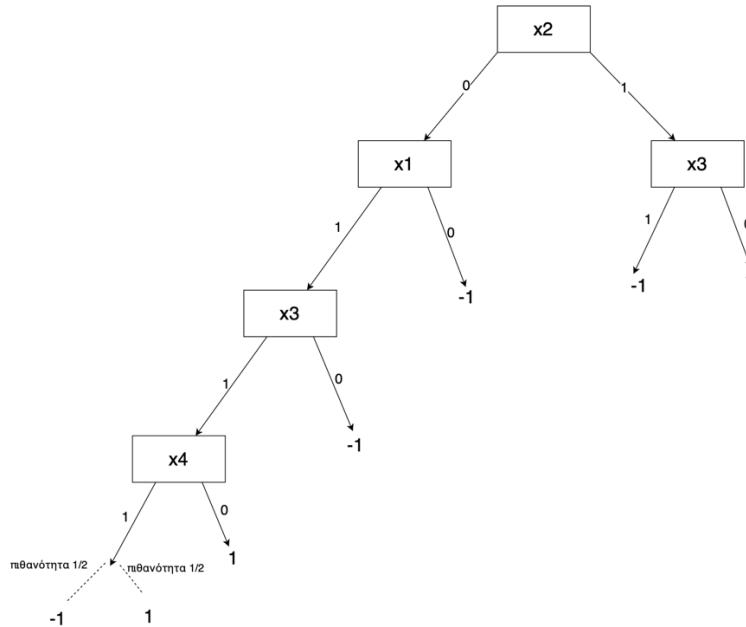
$$\text{gini}(\text{Root}) = 1 - (1/1)^2 = 0$$

Αφού $\text{gini}(\text{root}) = 0$ δε χρειάζεται να συνεχίσουμε τους υπολογισμούς. Για $x_4=0$ η απάντηση είναι 1.

Συνεχίζουμε την ίδια διαδικασία για το δεξί παιδί του x_4 (για $x_4=1$):

Σε αυτή την περίπτωση, δηλαδή για $x_1=1$, $x_2=0$, $x_3=1$, $x_4=1$ έχουμε δύο πιθανά output(είτε 1 είτε -1) με ίδια πιθανότητα.

Οπότε έχουμε τελικό σχήμα:



2.

Υπολογισμός με εντροπία:

$$E(\text{Root}) = -(5/10) \log_2(5/10) - (5/10) \log_2(5/10) = 1$$

x_1

$$E(x_1 = 1) = -(3/5) \log_2(3/5) - (2/5) \log_2(2/5) = 0.971$$

$$E(x_1 = 0) = -(2/5) \log_2(2/5) - (3/5) \log_2(3/5) = 0.971$$

$$E(x_1) = (5/10) E(x_1 = 1) + (5/10) E(x_1 = 0) = 0.971$$

$$\text{ig}(x_1) = E(\text{Root}) - E(x_1) = 0.029$$

x_2

$$E(x_2 = 1) = -(2/3) \log_2(2/3) - (1/3) \log_2(1/3) = 0.918$$

$$E(x_2 = 0) = -(3/7) \log_2(3/7) - (4/7) \log_2(4/7) = 0.985$$

$$E(x_2) = (3/10) E(x_1 = 1) + (7/10) E(x_1 = 0) = 0.9649$$

$$ig(x_2) = E(\text{Root}) - E(x_2) = 0.0351$$

x3

$$E(x_3 = 1) = -(3/6) \log_2(3/6) - (3/6) \log_2(3/6) = 1$$

$$E(x_3 = 0) = -(2/4) \log_2(2/4) - (2/4) \log_2(2/4) = 1$$

$$E(x_3) = (6/10) E(x_3 = 1) + (4/10) E(x_3 = 0) = 1$$

$$ig(x_3) = E(\text{Root}) - E(x_3) = 0$$

x4

$$E(x_4 = 1) = -(2/3) \log_2(2/3) - (1/3) \log_2(1/3) = 0.918$$

$$E(x_4 = 0) = -(3/7) \log_2(3/7) - (4/7) \log_2(4/7) = 0.985$$

$$E(x_4) = (3/10) E(x_4 = 1) + (7/10) E(x_4 = 0) = 0.9649$$

$$ig(x_4) = E(\text{Root}) - E(x_4) = 0.0351$$

x2 = 0

$$E(\text{Root}) = -(3/7) \log_2(3/7) - (4/7) \log_2(4/7) = 0.985$$

x1

$$E(x_1 = 1) = -(3/5) \log_2(3/5) - (2/5) \log_2(2/5) = 0.971$$

$$E(x_1 = 0) = -(2/2) \log_2(2/2) = 0$$

$$E(x_1) = (5/7) E(x_1 = 1) + (2/7) E(x_1 = 0) = 0.694$$

$$ig(x_1) = E(\text{Root}) - E(x_1) = 0.277$$

x3

$$E(x_3 = 1) = -(3/5) \log_2(3/5) - (2/5) \log_2(2/5) = 0.971$$

$$E(x_3 = 0) = -(2/2) \log_2(2/2) = 0$$

$$E(x_3) = (5/7) E(x_3 = 1) + (2/7) E(x_3 = 0) = 0.694$$

$$ig(x_3) = E(\text{Root}) - E(x_3) = 0.277$$

x4

$$E(x_4 = 1) = -(1/2) \log_2(1/2) - (1/2) \log_2(1/2) = 1$$

$$E(x_4 = 0) = -(2/5) \log_2(2/5) - (3/5) \log_2(3/5) = 0.971$$

$$E(x_4) = (2/7) E(x_4 = 1) + (5/7) E(x_4 = 0) = 0.979$$

$$ig(x_4) = E(\text{Root}) - E(x_4) = 0.02$$

$$x_2 = 0 \quad x_1 = 0$$

$$E(\text{Root}) = -(2/2) \log_2(2/2) = 0$$

Άρα η απάντηση είναι γνωστή πλεόν και ίση με -1.

$$x_2 = 0 \quad x_1 = 1$$

$$E(\text{Root}) = -(3/5) \log_2(3/5) - (2/5) \log_2(2/5) = 0.971$$

$$x_3$$

$$E(x_3 = 1) = -(3/4) \log_2(3/4) - (1/4) \log_2(1/4) = 0.811$$

$$E(x_3 = 0) = 0$$

$$E(x_3) = (4/5) \quad E(x_3 = 1) = 0.6488$$

$$ig(x_3) = E(\text{Root}) - E(x_3) = 0.162$$

$$x_4$$

$$E(x_4 = 1) = -(1/2) \log_2(1/2) - (1/2) \log_2(1/2) = 1$$

$$E(x_4 = 0) = -(2/3) \log_2(2/3) - (1/3) \log_2(1/3) = 0.918$$

$$E(x_4) = (2/5) \quad E(x_4 = 1) + (3/5) \quad E(x_4 = 0) = 0.22$$

$$ig(x_4) = E(\text{Root}) - E(x_4) = 0.751$$

Άρα επιλέγω x_4 και τερματίζουμε αφού μετά απέμεινε το x_3

$$x_2 = 1$$

$$E(\text{Root}) = -(2/3) \log_2(2/3) - (1/3) \log_2(1/3) = 0.918$$

$$x_1$$

$$E(x_1 = 1) = 0$$

$$E(x_1 = 0) = -(2/3) \log_2(2/3) - (1/3) \log_2(1/3) = 0.918$$

$$E(x_1) = (3/3) \quad E(x_1 = 0) = 0.918$$

$$ig(x_1) = E(\text{Root}) - E(x_1) = 0$$

$$x_3$$

$$E(x_3 = 1) = -(1/1) \log_2(1/1) = 0$$

$$E(x_3 = 0) = -(2/2) \log_2(2/2) = 0$$

$$E(x_3) = 0$$

$$ig(x_3) = E(\text{Root}) - E(x_3) = E(\text{Root})$$

Άρα επόμενο χαρακτηριστικό το x_3 το οποίο μας οδηγεί σε απάντηση και σταματάμε.

Οπότε έχουμε τελικό σχήμα:

