ΕΡΓΑΣΙΑ

ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

ΕΠΙΜΕΛΕΙΑ:
ΓΟΥΡΔΟΜΙΧΑΛΗΣ ΑΝΑΣΤΑΣΙΟΣ 10333 anasgour@ece.auth.gr
ΠΙΤΤΗΣ ΓΕΩΡΓΙΟΣ 10586 gkpittis@ece.auth.gr

ΜΕΡΟΣ Α

Υλοποίηση ταξινομητή Μέγιστης Πιθανοφάνειας

Στα Μέρη Α και Β εξετάζεται η αξιοπιστία ενός δείκτη-αριθμού x για τον προσδιορισμό του επιπέδου στρες που νιώθουν οι χρήστες όταν παίζουν κάποιο βιντεοπαιχνίδι. Πιο συγκεκριμένα, διακρίνουμε τους χρήστες σε 2 κλάσεις :

- Κλάση ω1 = χρήστες που δεν ένιωσαν στρες
- Κλάση ω2 = χρήστες που ένιωσαν στρες

Η κατανομή πυκνότητας πιθανότητας που ακολουθεί αυτός ο δείκτης και για τις 2 κλάσεις είναι:

$p(x|\theta) = \frac{1}{\pi} \cdot \frac{1}{1+(x-\theta)^2}$, όπου θ είναι μια άγνωστη παράμετρος.

Ακόμη, από τα 12 δείγματα του δείκτη x που διαθέτουμε:

- Τα 7 ανήκουν στη κλάση ω1 και αποτελούν το σύνολο D1 = [2.8,–0.4,–0.8,2.3,–0.3,3.6,4.1].
- Τα 5 ανήκουν στη κλάση ω2 και αποτελούν το σύνολο D2 = [–4.5,–3.4,–3.1,–3.0,–2.3].

Στη συνέχεια, θα αξιολογήσουμε την αξιοπιστία του δείκτη x, δημιουργώντας και στο Μέρος Α και στο Μέρος Β έναν ταξινομητή που θα διακρίνει τους χρήστες στις κλάσεις ω1 και ω2, βασιζόμενος στην τιμή του συγκεκριμένου δείκτη.

Υλοποίηση ταξινομητή Μέγιστης Πιθανοφάνειας

Στο Μέρος Α, κατασκευάζουμε έναν ταξινομητή Μέγιστης Πιθανοφάνειας.

Στο 1ο ερώτημα υπολογίζουμε:

- Τη παράμετρο $\widehat{\boldsymbol{\theta_1}}$ που μεγιστοποιεί τη $\boldsymbol{p(D1|\theta)}$
- Τη παράμετρο $\widehat{\boldsymbol{\theta_2}}$ που μεγιστοποιεί τη $\boldsymbol{p(D2|\theta)}$

Οι $\boldsymbol{p(D1|\theta)}$, $\boldsymbol{p(D2|\theta)}$ είναι οι πιθανοφάνειες (likelihoods) της $\boldsymbol{\theta}$ σε σχέση με τα δείγματα των συνόλων δεδομένων $D1$ και $D2$ αντίστοιχα.

Από τη θεωρία γνωρίζουμε ότι αν η παράμετρος $\widehat{\boldsymbol{\theta}}_{\boldsymbol{ML}}$ μεγιστοποιεί τη συνάρτηση πιθανοφάνειας $\boldsymbol{p(D|\theta)}$, τότε θα μεγιστοποιεί επίσης και τη λογαριθμική πιθανοφάνεια. Δηλαδή, θα μεγιστοποιεί τη $\ln(p(D|\boldsymbol{\theta}))$. Αυτό συμβαίνει διότι ο λογάριθμος είναι μια μονότονα αύξουσα συνάρτηση. Για αυτό τον λόγο, στον κώδικα επιλέξαμε να υλοποιήσουμε την $\ln(p(D|\boldsymbol{\theta}))$ με τη συνάρτηση **log_p_D_theta(self, theta, D)**.

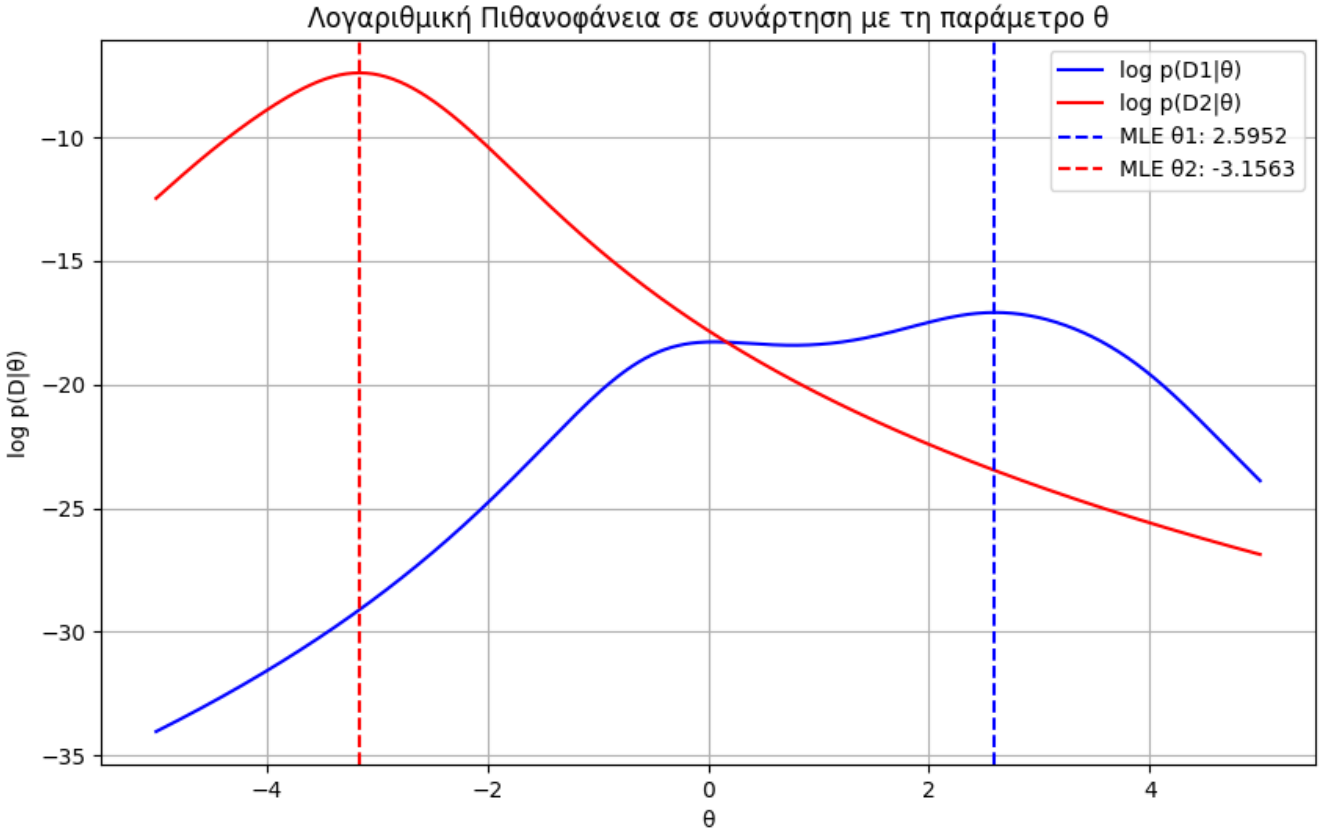
Υλοποίηση ταξινομητή Μέγιστης Πιθανοφάνειας

Μαθηματική ανάλυση :

- $\boldsymbol{p(D|\theta)} = \boldsymbol{p(x_1, x_2, \dots, x_N|\theta)} = \prod_{i=1}^{N} \boldsymbol{p(x_i|\theta)}$ όπου $\boldsymbol{x_i}$ είναι τα στοιχεία του συνόλου D. Τα $\boldsymbol{x_i}$ είναι διάφορα δείγματα του δείκτη x. Το σύνολο D έχει N σε πλήθος στοιχεία. Για το D1 έχω N=7 ενώ για το D2 έχω N=5.
- Το $\boldsymbol{p(x_i|\theta)}$ είναι η κατανομή πυκνότητας πιθανότητας που δίνεται στην εκφώνηση.
- Για διάφορες τιμές της παραμέτρου θ, υπολογίζουμε τη λογαριθμική πιθανοφάνεια: $\boldsymbol{\ln(\, p(D|\theta)\,)} = \boldsymbol{\ln(\prod_{i=1}^{N} p(x_i|\theta))} = \boldsymbol{\ln\left(\prod_{i=1}^{N}\left(\frac{1}{\pi}\cdot\frac{1}{1+(x_i-\theta)^2}\right)\right)} = \boldsymbol{\sum_{i=1}^{N}\left(\ln\left(\frac{1}{\pi}\cdot\frac{1}{1+(x_i-\theta)^2}\right)\right)}$.
- Επιπλέον, εδώ φαίνεται ότι η λογαριθμική πιθανοφάνεια υπολογίζεται ευκολότερα από την απλή πιθανοφάνεια, διότι χρησιμοποιεί άθροισμα αντί για γινόμενο.
- Στον κώδικα επιλέχθηκαν 500 υποψήφιες τιμές θ από -5 μέχρι 5 με την εντολή : **theta_candidates = np.linspace(-5, 5, 500)** .
- Με τη συνάρτηση **fit(self, D, theta_candidates)** βρίσκω τα $\widehat{\boldsymbol{\theta_1}}$, $\widehat{\boldsymbol{\theta_2}}$ που μεγιστοποιούν τα $\boldsymbol{p(D1|\theta)}$, $\boldsymbol{\ln(\, p(D1|\theta)\,)}$ και $\boldsymbol{p(D2|\theta)}$, $\boldsymbol{\ln(\, p(D2|\theta)\,)}$ αντίστοιχα.

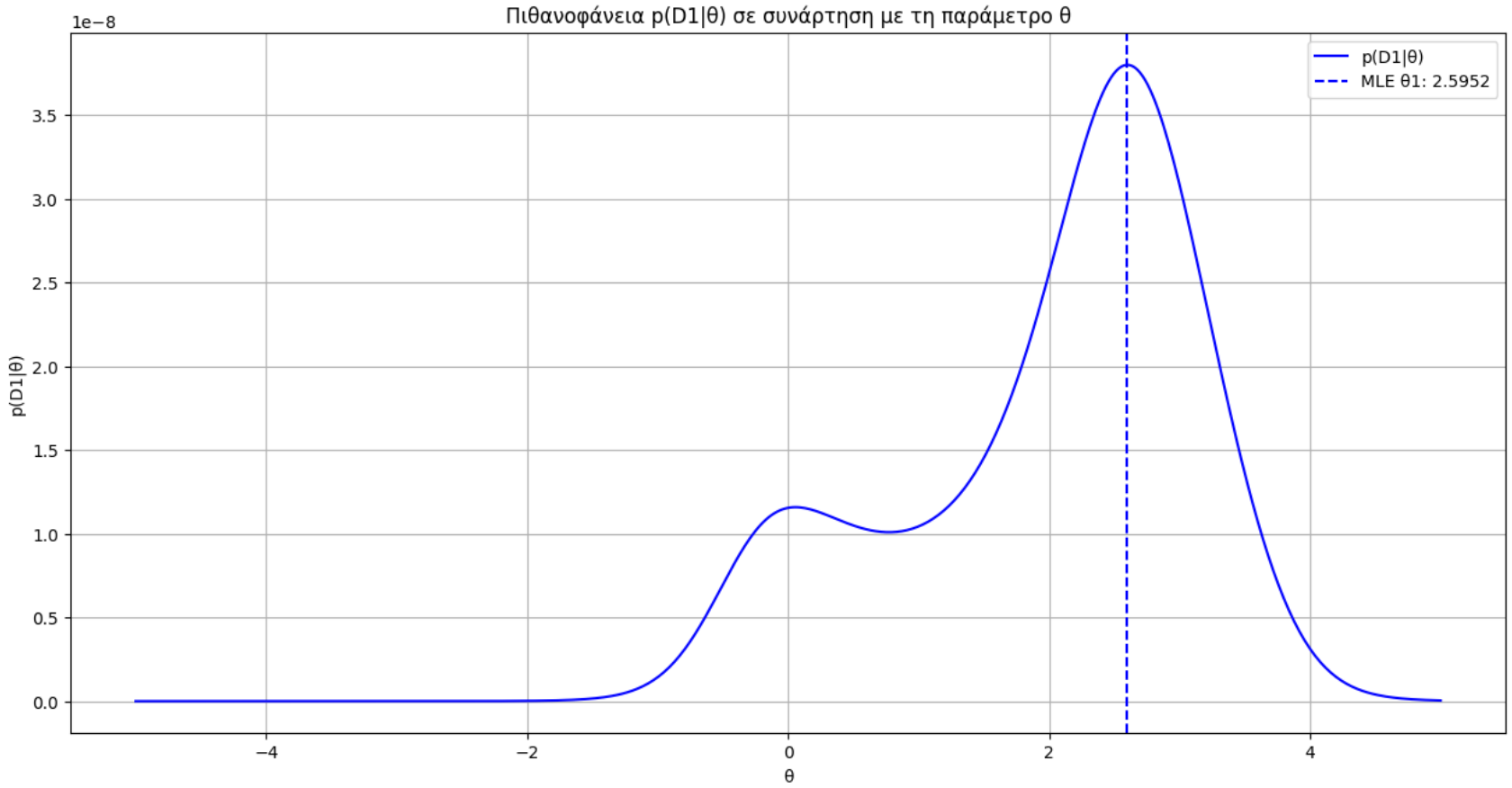
Υλοποίηση ταξινομητή Μέγιστης Πιθανοφάνειας

Στο διπλανό σχήμα απεικονίζουμε τις κυματομορφές των συναρτήσεων λογαριθμικής πιθανοφάνειας $\ln(p(D1|\theta))$ και $\ln(p(D2|\theta))$ όπως ζητείται στο 1ο ερώτημα. Επίσης, απεικονίζουμε με διακεκομμένες γραμμές τις τιμές των παραμέτρων $\widehat{\theta_1}$ και $\widehat{\theta_2}$ για τις οποίες έχουμε μέγιστη πιθανοφάνεια. Υπολογίσαμε $\widehat{\theta_1}$ = 2.5952 για τη κλάση ω1 και $\widehat{\theta_2}$ = -3.1563 για τη κλάση ω2.Παρατηρούμε σχηματικά πως για θ = $\widehat{\theta_1}$ έχουμε μέγιστο $\ln(p(D1|\theta))$ και για θ = $\widehat{\theta_2}$ έχουμε μέγιστο $\ln(p(D2|\theta))$.



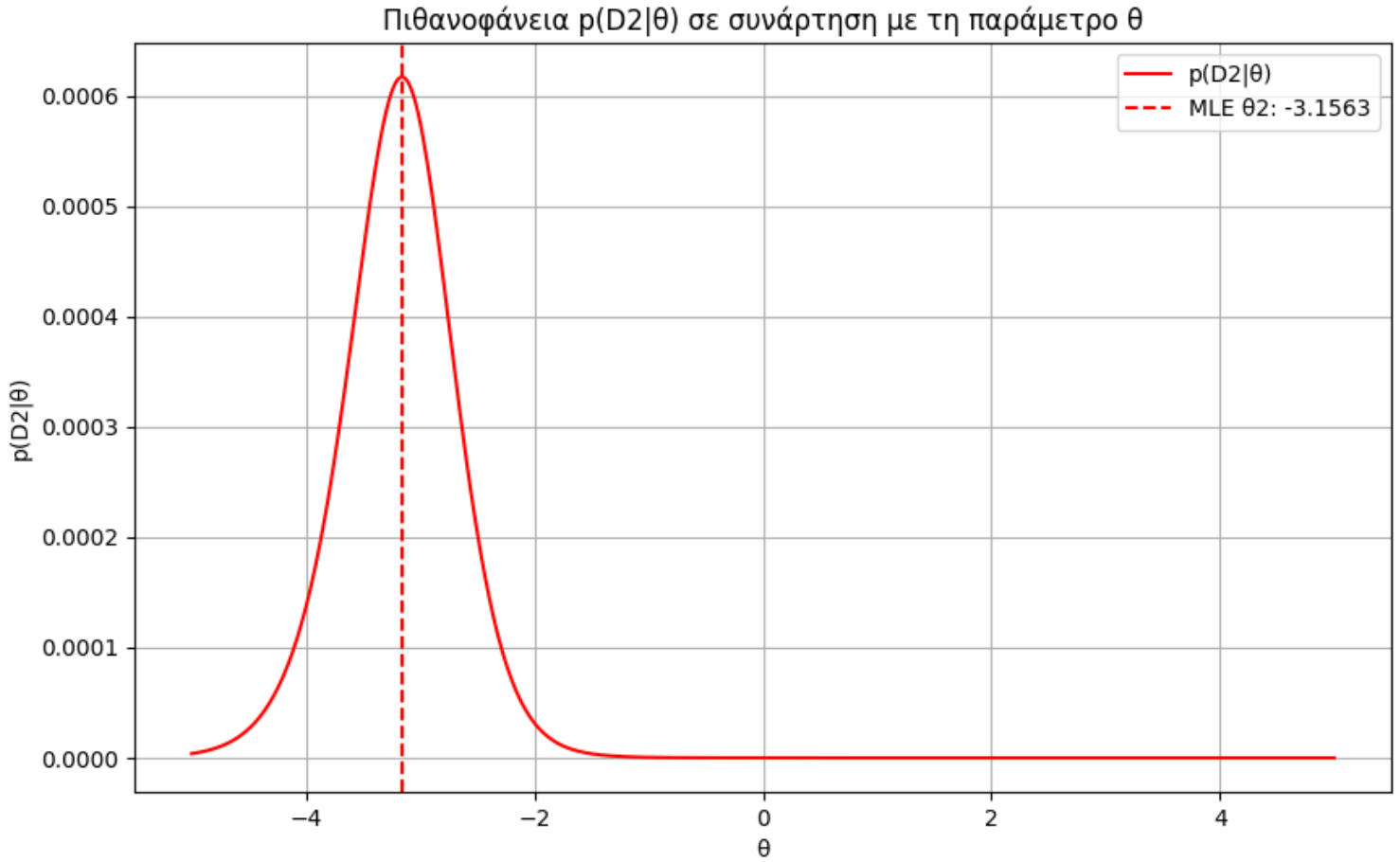
Υλοποίηση ταξινομητή Μέγιστης Πιθανοφάνειας

- Η κυματομορφή της πιθανοφάνειας $\boldsymbol{p(D1|\theta)}$ για τη κλάση ω1 μαζί με το $\widehat{\boldsymbol{\theta_1}}$. Παρατηρούμε σχηματικά πως για θ = $\widehat{\boldsymbol{\theta_1}}$ έχουμε μέγιστο $\boldsymbol{p(D1|\theta)}$.



Υλοποίηση ταξινομητή Μέγιστης Πιθανοφάνειας

- Η κυματομορφή της πιθανοφάνειας $\boldsymbol{p(D2|\theta)}$ για τη κλάση ω2 μαζί με το $\widehat{\boldsymbol{\theta_2}}$. Παρατηρούμε σχηματικά πως για θ = $\widehat{\boldsymbol{\theta_2}}$ έχουμε μέγιστο $\boldsymbol{p(D2|\theta)}$.



Υλοποίηση ταξινομητή Μέγιστης Πιθανοφάνειας

Στο 2ο ερώτημα χρησιμοποιούμε τη συνάρτηση διάκρισης g(x) για να ταξινομήσουμε τα δείγματα $\boldsymbol{x_i}$ των συνόλων D1 και D2 στις κλάσεις ω1 και ω2 αντίστοιχα.

- Μαθηματική ανάλυση :

$$\boldsymbol{g(x) = \ln\left(P\left(x\middle|\widehat{\theta_1}\right)\right) - \ln\left(P\left(x\middle|\widehat{\theta_2}\right)\right) + \ln(P(\omega_1)) - \ln(P(\omega_2)) = \ln\left(\frac{P\left(x\middle|\widehat{\theta_1}\right)}{P\left(x\middle|\widehat{\theta_2}\right)}\right) + \ln\left(\frac{P(\omega_1)}{P(\omega_2)}\right)}$$

Προκύπτει ότι έχω $\boldsymbol{g(x) = g_1(x) - g_2(x)}$, με $\boldsymbol{g_1(x) = \ln\left(P\left(x\middle|\widehat{\theta_1}\right)\right) + \ln(P(\omega_1))}$ και $\boldsymbol{g_2(x) = \ln\left(P\left(x\middle|\widehat{\theta_2}\right)\right) + \ln(P(\omega_2))}$.

- Ο **Κανόνας Απόφασης** της g(x) είναι ο εξής: Αποφασίζει να ταξινομήσει το δείγμα $\boldsymbol{x_i}$ στη κλάση ω1 αν $\boldsymbol{g(x) > 0 \Longrightarrow g_1(x) > g_2(x)}$. Διαφορετικά, αποφασίζει να ταξινομήσει το δείγμα $\boldsymbol{x_i}$ στη κλάση ω2.

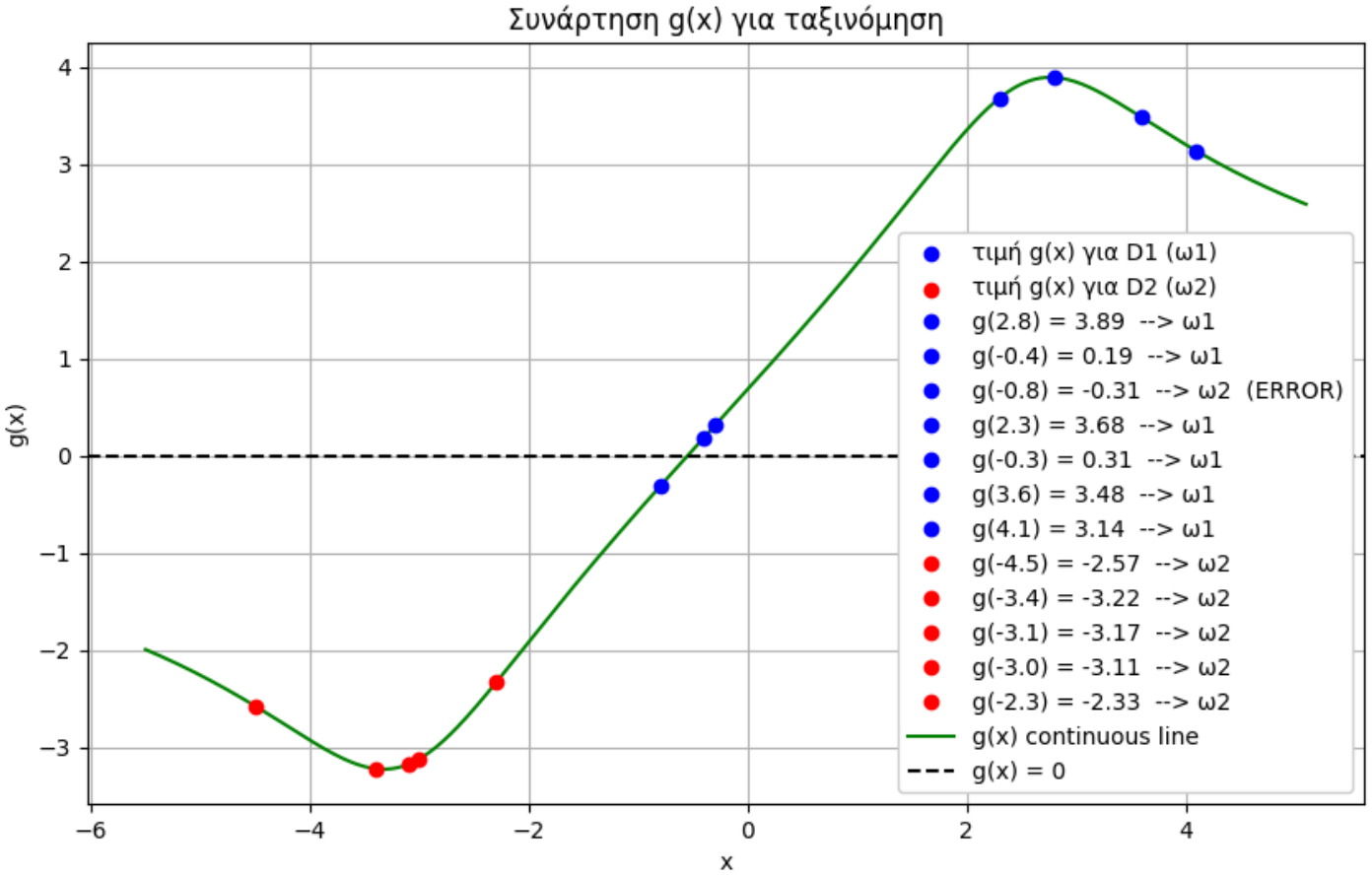
- **Όριο απόφασης** : $\boldsymbol{g(x) = 0}$ (το σημείο στο οποίο διαχωρίζονται οι 2 κλάσεις).

- Ο υπολογισμός των τιμών της g(x) γίνεται με τη συνάρτηση **predict(self, D, prior1, prior2)**

με **prior1** = $\boldsymbol{P(\omega_1)}$ και **prior2** = $\boldsymbol{P(\omega_2)}$.

Υλοποίηση ταξινομητή Μέγιστης Πιθανοφάνειας

Στο διπλανό σχήμα με μπλε κουκκίδες αναπαριστούμε τα στοιχεία του συνόλου D1, τα οποία πρέπει να ταξινομηθούν στην κλάση ω1. Επίσης, με κόκκινες κουκκίδες αναπαριστούμε τα στοιχεία του συνόλου D2 τα οποία πρέπει να ταξινομηθούν στη κλάση ω2. Επιπλέον, αναπαριστούμε τη κυματομορφή της g(x) με μια πράσινη συνεχή γραμμή. Το όριο απόφασης g(x) = 0 απεικονίζεται με μια μαύρη διακεκομμένη γραμμή. Στο σχήμα φαίνονται οι τιμές της g(x) για όλα τα στοιχεία των D1 και D2. Παρατηρούμε, ότι το δείγμα x = -0,8 του D1 ταξινομείται **λανθασμένα** στη κλάση ω2.



Υλοποίηση ταξινομητή Μέγιστης Πιθανοφάνειας

- Παρατηρήσεις για τη ταξινόμηση :

Όλα τα δείγματα $\boldsymbol{x_i}$ του D1 έχουν θετικές τιμές για τη συνάρτηση g(x) , με εξαίρεση το δείγμα x = -0,8 για το οποίο έχω g(-0,8) = -0,31 < 0. Όμως, η τιμή αυτή είναι αρκετά κοντά στο όριο απόφασης g(x) = 0. Δηλαδή, μπορούμε να υποθέσουμε ότι αν η συνάρτηση g(x) έπαιρνε **λίγο μεγαλύτερες θετικές τιμές** για τα δείγματα $\boldsymbol{x_i}$ του D1, τότε θα κατάφερνε να ταξινομήσει σωστά το x = -0,8.

Όλα τα δείγματα $\boldsymbol{x_i}$ του D2 έχουν αρνητικές τιμές για τη συνάρτηση g(x).

Επομένως, ο ταξινομητής Μέγιστης Πιθανοφάνειας που υλοποιήσαμε, **αποτυχαίνει** να ταξινομήσει στις σωστές κλάσεις όλα τα δείγματα $\boldsymbol{x_i}$.

ΜΕΡΟΣ Β

Υλοποίηση Μπεϋζιανού ταξινομητή

- Στο Μέρος Β, υλοποιούμε έναν νέο ταξινομητή εκτιμώντας την άγνωστη παράμετρο θ με τη μέθοδο εκτίμησης κατά Bayes.
- Τα σύνολα δεδομένων D1 και D2 που θα μελετηθούν είναι ίδια με εκείνα του Μέρους Α.
- Ο μπεϋζιανός ταξινομητής που πρόκειται να υλοποιήσουμε , λαμβάνει υπόψη του την **εκ των προτέρων συνάρτηση πυκνότητας πιθανότητας** της παραμέτρου θ:

$$\boldsymbol{p(\theta) = \frac{1}{10\pi} \cdot \frac{1}{1+\left(\frac{\theta}{10}\right)^2}}$$

- Η εκ των υστέρων κατανομή πυκνότητας πιθανότητας $\boldsymbol{p(x|\theta)}$ ταυτίζεται με εκείνη του Μέρους Α.
- <u>Σύντομη εξήγηση των συναρτήσεων που υλοποιήθηκαν στον κώδικα</u> :

Η συνάρτηση **p_x_given_theta(self, x, theta)** υπολογίζει τη $\boldsymbol{p(x|\theta) = \frac{1}{\pi} \cdot \frac{1}{1+(x-\theta)^2}}$.

Η συνάρτηση **p_D_given_theta(self, D, theta)** υπολογίζει τη πιθανοφάνεια $\boldsymbol{p(D|\theta)}$:

$\boldsymbol{p(D|\theta) = \prod_{n=1}^{N} p(x_n|\theta) = p(x_1|\theta) \cdot p(x_2|\theta) \cdot p(x_3|\theta) \cdots p(x_N|\theta)}$, *όπου* $\boldsymbol{D = [x_1, x_2, x_3, \dots, x_N]}$.

Υλοποίηση Μπεϋζιανού ταξινομητή

- Σύντομη εξήγηση των συναρτήσεων που υλοποιήθηκαν στον κώδικα (συνέχεια):

Η συνάρτηση **prior(self, theta)** υπολογίζει την εκ των προτέρων πιθανότητα

$$p(\theta) = \frac{1}{10\pi} \cdot \frac{1}{1+\left(\frac{\theta}{10}\right)^2} .$$

Η συνάρτηση **integral_calc(self, f, x)** υλοποιεί τον κανόνα του τραπεζίου.

Η συνάρτηση **p_theta_given_D(self, D, theta)** υπολογίζει την εκ των υστέρων πιθανότητα $p(\theta|D)$:

$$p(\theta|D) = \frac{p(D|\theta) \cdot p(\theta)}{\int p(D|\theta) \cdot p(\theta) d\theta}$$

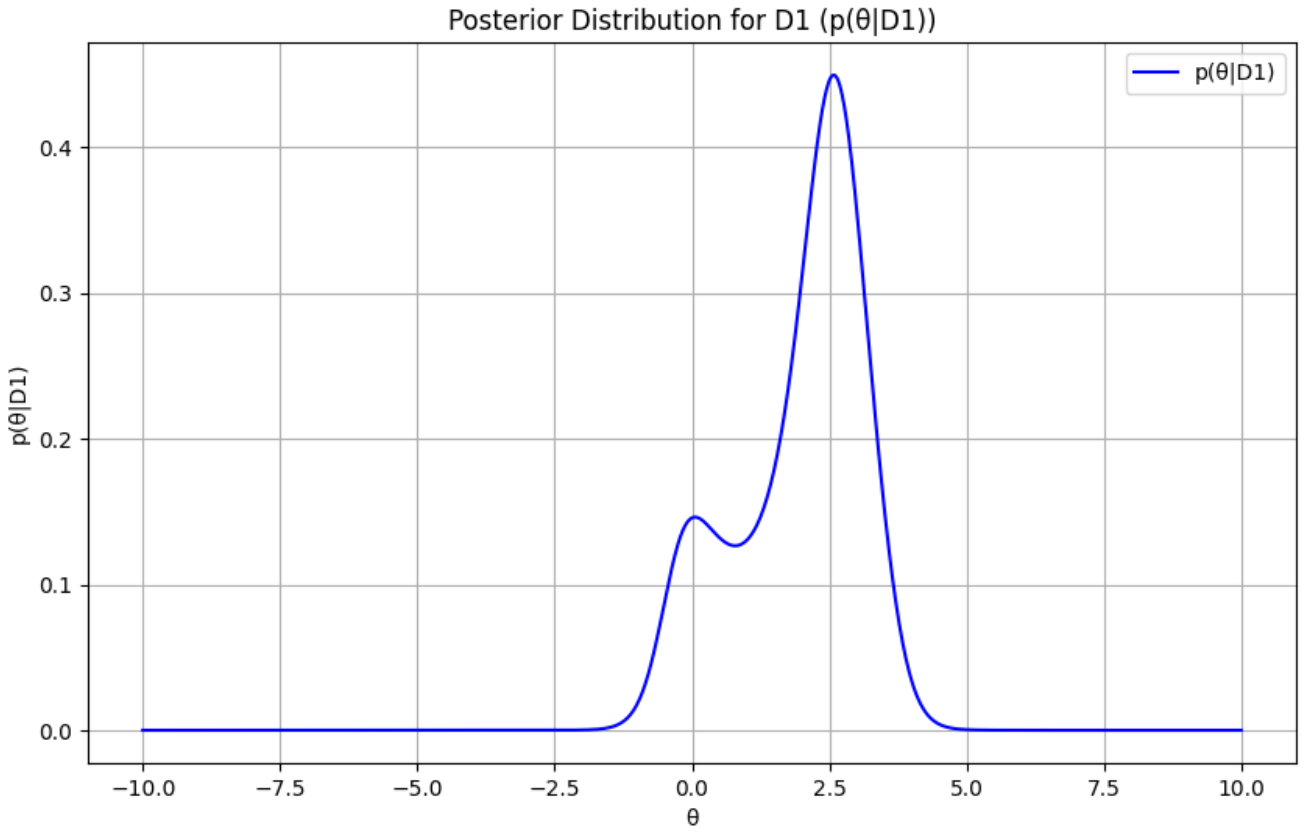
Η συνάρτηση **p_x_given_D(self, D, x, theta)** υπολογίζει την εκ των υστέρων πυκνότητα πιθανότητας $p(x|D) = \int p(x|\theta) \cdot p(\theta|D) \cdot d\theta$

Η συνάρτηση **predict(self, D1, D2, x_values, theta, P_omega1, P_omega2)** υπολογίζει τις τιμές της συνάρτησης διάκρισης h(x), που χρησιμοποιούμε στο 2ο ερώτημα για να ταξινομήσουμε τα δείγματα x_i των συνόλων D1 και D2 στις κλάσεις ω1 και ω2 αντίστοιχα.

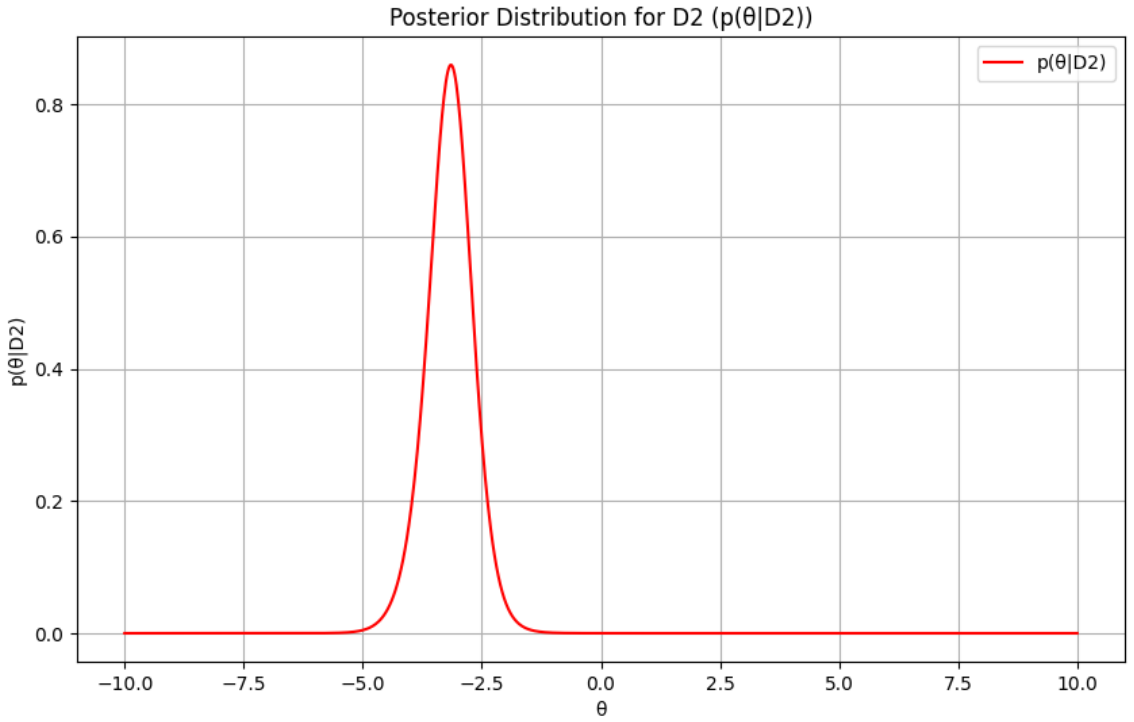
Υλοποίηση Μπεϋζιανού ταξινομητή

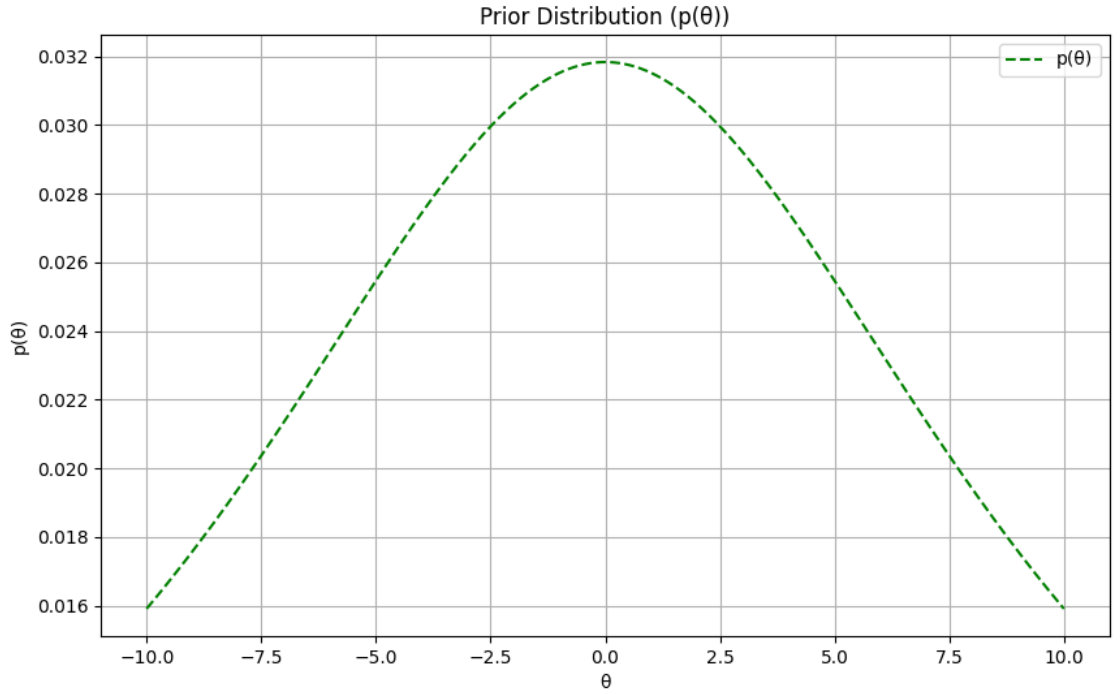
- Στο 1ο ερώτημα, υπολογίσαμε και απεικονίσαμε τις εκ των υστέρων πυκνότητες πιθανότητας $\boldsymbol{p(\theta|D1)}$ και $\boldsymbol{p(\theta|D2)}$. Επίσης, απεικονίσαμε και την εκ των προτέρων πιθανότητα $\boldsymbol{p(\theta)}$.

- Στο διπλανό σχήμα, φαίνεται η κυματομορφή της $\boldsymbol{p(\theta|D1)}$ για διάφορες τιμές του θ.



Υλοποίηση Μπεϋζιανού ταξινομητή

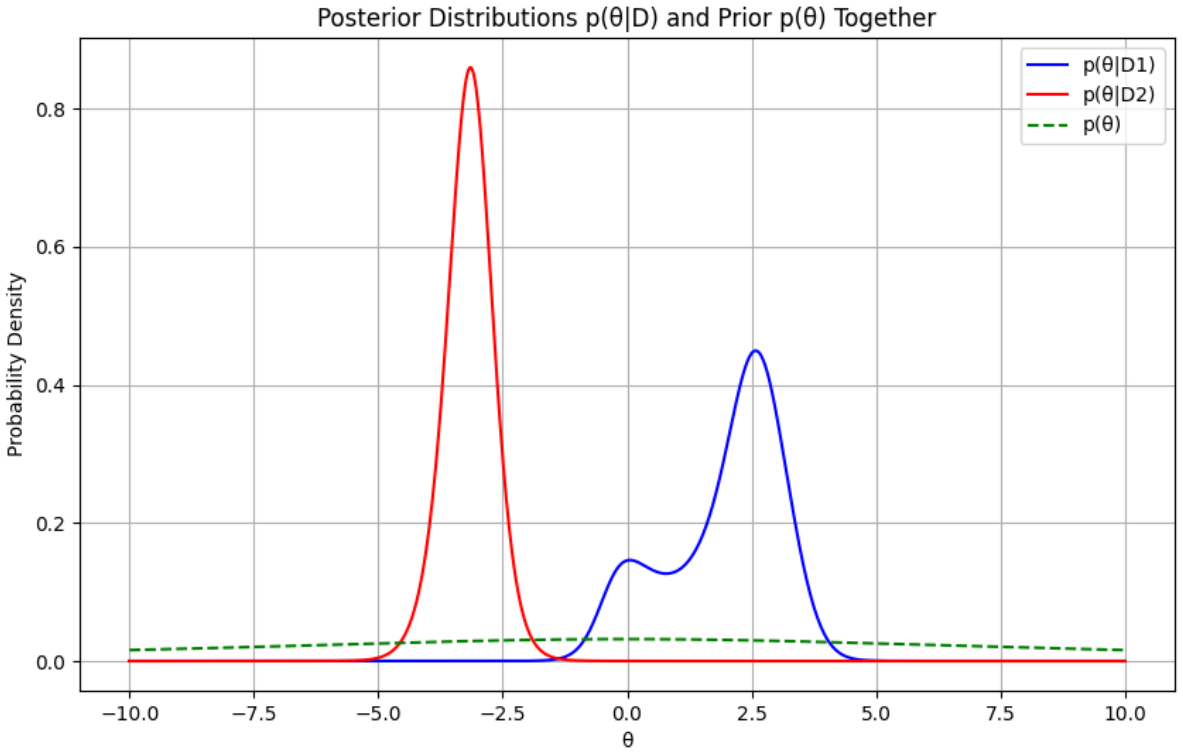




Στο αριστερό σχήμα φαίνεται η κυματομορφή της $\boldsymbol{p(\theta|D2)}$ για διάφορες τιμές του θ. Στο δεξιό σχήμα φαίνεται η κυματομορφή της $\boldsymbol{p(\theta)}$ για διάφορες τιμές του θ. Στον κώδικα επιλέχθηκαν 1000 υποψήφιες τιμές θ από -10 μέχρι 10 με την εντολή **theta = np.linspace(-10, 10, 1000).**

Υλοποίηση Μπεϋζιανού ταξινομητή

Στο διπλανό σχήμα, παρουσιάζονται οι κυματομορφές των $\boldsymbol{p(\theta \mid D1)}$, $\boldsymbol{p(\theta \mid D2)}$ και $\boldsymbol{p(\theta)}$ σε κοινό διάγραμμα. Οι κατανομές $\boldsymbol{p(\theta|D1)}$ και $\boldsymbol{p(\theta|D2)}$ συγκεντρώνονται σε ένα στενότερο εύρος τιμών σε σχέση με την $\boldsymbol{p(\theta)}$. Αυτό συμβαίνει επειδή γνωρίζουν τα διαθέσιμα δεδομένα (D1 και D2) και με αυτόν τον τρόπο καταφέρνουν να εστιάσουν στις πιο πιθανές περιοχές τιμών του θ με βάση τα δεδομένα αυτά. Δηλαδή, δίνουν μεγαλύτερη έμφαση σε συγκεκριμένα διαστήματα θ. Επίσης, οι $\boldsymbol{p(\theta \mid D1)}$ και $\boldsymbol{p(\theta \mid D2)}$ έχουν υψηλότερες κορυφές από την $\boldsymbol{p(\theta)}$, διότι η εκ των προτέρων γνώση των δεδομένων παρέχει πληροφορίες που αυξάνουν τη πιθανότητα εμφάνισης συγκεκριμένων τιμών της παραμέτρου θ. Αντίθετα, η $\boldsymbol{p(\theta)}$ απλώνεται σε όλες τις τιμές του θ, κατανέμοντας έτσι την πιθανότητα ακόμη και σε τιμές θ πρακτικά απίθανες με βάση τα δεδομένα των συνόλων D1 και D2.



Υλοποίηση Μπεϋζιανού ταξινομητή

- Στο 2ο ερώτημα, υλοποιώ τη συνάρτηση predict που υπολογίζει τις τιμές της συνάρτησης διάκρισης h(x). Η συνάρτηση h(x) χρησιμοποιείται για τη ταξινόμηση των δειγμάτων x_i των συνόλων D1 και D2 στις κλάσεις ω1 και ω2 αντίστοιχα.
- Μαθηματική ανάλυση : Ακολουθούμε την ίδια λογική με Μέρος Α.

Τα $P(x|D1)$ και $P(x|D2)$ υπολογίζονται από τη συνάρτηση **p_x_given_D(self, D, x, theta) .**

$$h(x) = \ln P(x|D1) - \ln P(x|D2) + \ln P(\omega 1) - \ln P(\omega 2) \Leftrightarrow h(x) = \ln\left(\frac{P(x|D1)}{P(x|D2)}\right) + \ln\left(\frac{P(\omega 1)}{P(\omega 2)}\right)$$

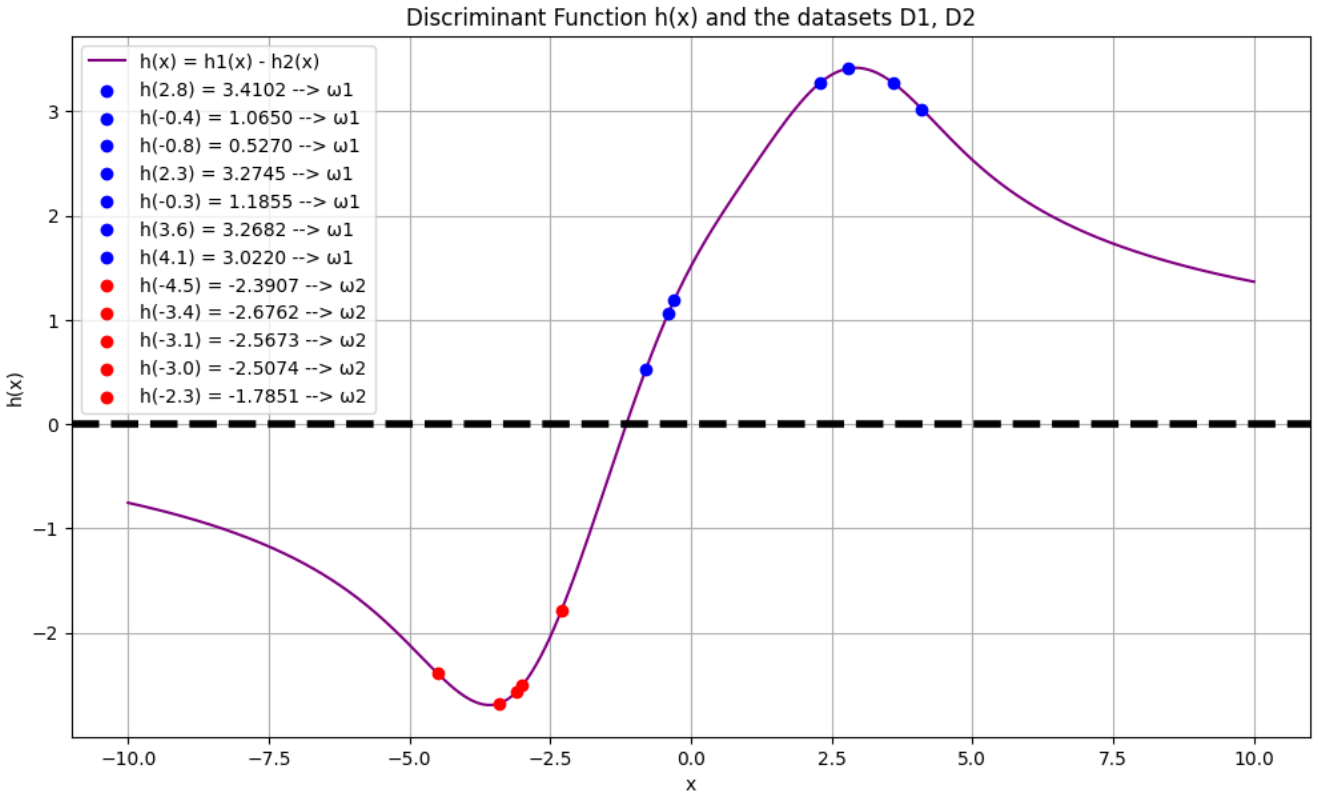
Προκύπτει ότι έχω $h(x) = h_1(x) - h_2(x)$, όπου $h_1(x) = \ln P(x|D1) + \ln P(\omega 1)$ και

$h_2(x) = \ln P(x|D2) + \ln P(\omega 2)$.

- Ο **Κανόνας Απόφασης** της h(x) είναι ο εξής: Αποφασίζει να ταξινομήσει το δείγμα x_i στη κλάση ω1 αν $h(x) > 0 \Longrightarrow h_1(x) > h_2(x)$. Διαφορετικά, αποφασίζει να ταξινομήσει το δείγμα x_i στη κλάση ω2.
- **Όριο απόφασης** : $h(x) = 0$ (το σημείο στο οποίο διαχωρίζονται οι 2 κλάσεις).

Υλοποίηση Μπεϋζιανού ταξινομητή

Στο διπλανό σχήμα με μπλε κουκκίδες αναπαριστούμε τα στοιχεία του συνόλου D1, τα οποία πρέπει να ταξινομηθούν στην κλάση ω1. Επίσης, με κόκκινες κουκκίδες αναπαριστούμε τα στοιχεία του συνόλου D2 τα οποία πρέπει να ταξινομηθούν στη κλάση ω2. Επιπλέον, αναπαριστούμε τη κυματομορφή της h(x) με μια μωβ συνεχή γραμμή. Το όριο απόφασης h(x) = 0 απεικονίζεται με μια μαύρη διακεκομμένη γραμμή. Στο σχήμα φαίνονται οι τιμές της h(x) για όλα τα στοιχεία των D1 και D2. Παρατηρούμε ότι όλα τα δείγματα x_i των συνόλων D1 και D2 ταξινομούνται στις σωστές κλάσεις. Γίνεται **σωστή** ταξινόμηση.



Υλοποίηση Μπεϋζιανού ταξινομητή

- Παρατηρήσεις για τη ταξινόμηση :

Όλα τα δείγματα $\boldsymbol{x_i}$ του D1 έχουν θετικές τιμές για τη συνάρτηση h(x).

Όλα τα δείγματα $\boldsymbol{x_i}$ του D2 έχουν αρνητικές τιμές για τη συνάρτηση h(x).

Τα δείγματα x = -0.4 και x = -0.3 του συνόλου D1, για τα οποία η συνάρτηση g(x) ήταν οριακά θετική, παρουσιάζουν στη συνάρτηση h(x) σημαντικά αυξημένες θετικές τιμές μακριά από το 0. Το ίδιο ισχύει και για το δείγμα x = −0.8 , για το οποίο η συνάρτηση g(x) ήταν οριακά αρνητική και επομένως γινόταν λανθασμένα η ταξινόμηση του στη κλάση ω2. Αυτό καθιστά πιο ξεκάθαρη την ταξινόμησή των δειγμάτων -0.4 και -0.3 στην κλάση ω1 και συγχρόνως διορθώνει την λανθασμένη ταξινόμηση του δείγματος -0.8 τοποθετώντας το ορθά στη κλάση ω1. Με άλλα λόγια, ο μπεϋζιανός ταξινομητής καταφέρνει να ταξινομήσει περισσότερα αρνητικά δείγματα $\boldsymbol{x_i}$ στην κλάση ω1 με μεγαλύτερη ακρίβεια.

Σε αντίθεση με τον ταξινομητή Μέγιστης Πιθανοφάνειας του Μέρους Α, ο μπεϋζιανός ταξινομητής πετυχαίνει σωστή ταξινόμηση.

Μπεϋζιανός ταξινομητής VS ταξινομητής Μέγιστης Πιθανοφάνειας

Ο Μπεϋζιανός ταξινομητής παρουσιάζει μεγαλύτερη πολυπλοκότητα από τον ταξινομητή Μέγιστης Πιθανοφάνειας. Ωστόσο, κάτι τέτοιο δεν έχει ιδιαίτερη σημασία στο συγκεκριμένο πρόβλημα διότι το πλήθος των δειγμάτων που ταξινομήθηκαν ήταν πολύ μικρό.

Η **βασική διαφορά** μεταξύ των δύο ταξινομητών έγκειται στο γεγονός ότι ο Μπεϋζιανός ταξινομητής αξιοποιεί την εκ των προτέρων γνώση που περιγράφεται από την κατανομή $\boldsymbol{p(\theta)}$. Επιπλέον, χρησιμοποιεί τις εκ των υστέρων κατανομές $\boldsymbol{p(\theta|D)}$ και $\boldsymbol{p(x|D)}$, αξιοποιώντας με αυτόν τον τρόπο την πληροφορία που παρέχουν τα δεδομένα εκπαίδευσης. Αυτή η προσέγγιση καθιστά τον Μπεϋζιανό ταξινομητή πιο ικανό να ταξινομήσει σωστά όλα τα δείγματα $\boldsymbol{x_i}$.

ΜΕΡΟΣ Γ

Ενότητα 1 - Δέντρο Απόφασης

Στην ενότητα 1 του Μέρος Γ χρησιμοποιώ τη βάση Iris από το sklearn, η οποία περιέχει 150 δείγματα (50 ανά είδος) για τα μήκη και πλάτη σεπάλων και πετάλων τριών ειδών του φυτού Ίριδα : Iris setosa , Iris versicolor , Iris virginica.

Επιλέγουμε μόνο τα δύο πρώτα χαρακτηριστικά της βάσης και εκπαιδεύουμε τον έτοιμο αλγόριθμο DecisionTreeClassifier με το 50% των δεδομένων, ταξινομώντας στη συνέχεια το υπόλοιπο 50%.

Με την εντολή **X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42)** και ειδικότερα με τη παράμετρο test_size=0.5, καθορίζεται ότι το 50% των δεδομένων θα χρησιμοποιηθεί ως σύνολο εκπαίδευσης (training set) και το υπόλοιπο 50% ως σύνολο δοκιμής (test set).

Αξίζει να σημειώσουμε ότι στον κώδικα μας, χρησιμοποιούμε τη παράμετρο random_state=42 για να εξασφαλίσουμε την αναπαραγωγή των ίδιων αποτελεσμάτων. Χωρίς τον ορισμό του συγκεκριμένου seed, κάθε εκτέλεση του κώδικα μπορεί να παράγει ελαφρώς διαφορετικά αποτελέσματα, καθώς η τυχαιότητα στον διαχωρισμό των δεδομένων ή στην επιλογή των υποδειγμάτων για το bootstrap στην ενότητα 2 μπορεί να οδηγήσει σε διαφορετικά σύνολα εκπαίδευσης και δοκιμής.

Ενότητα 1 - Δέντρο Απόφασης

Στο 1ο ερώτημα :

Ο ταξινομητής DecisionTreeClassifier εκπαιδεύεται για διαφορετικά βάθη, από 1 έως 10. Σε κάθε επανάληψη του βρόχου **for depth in range(1, 11)** :

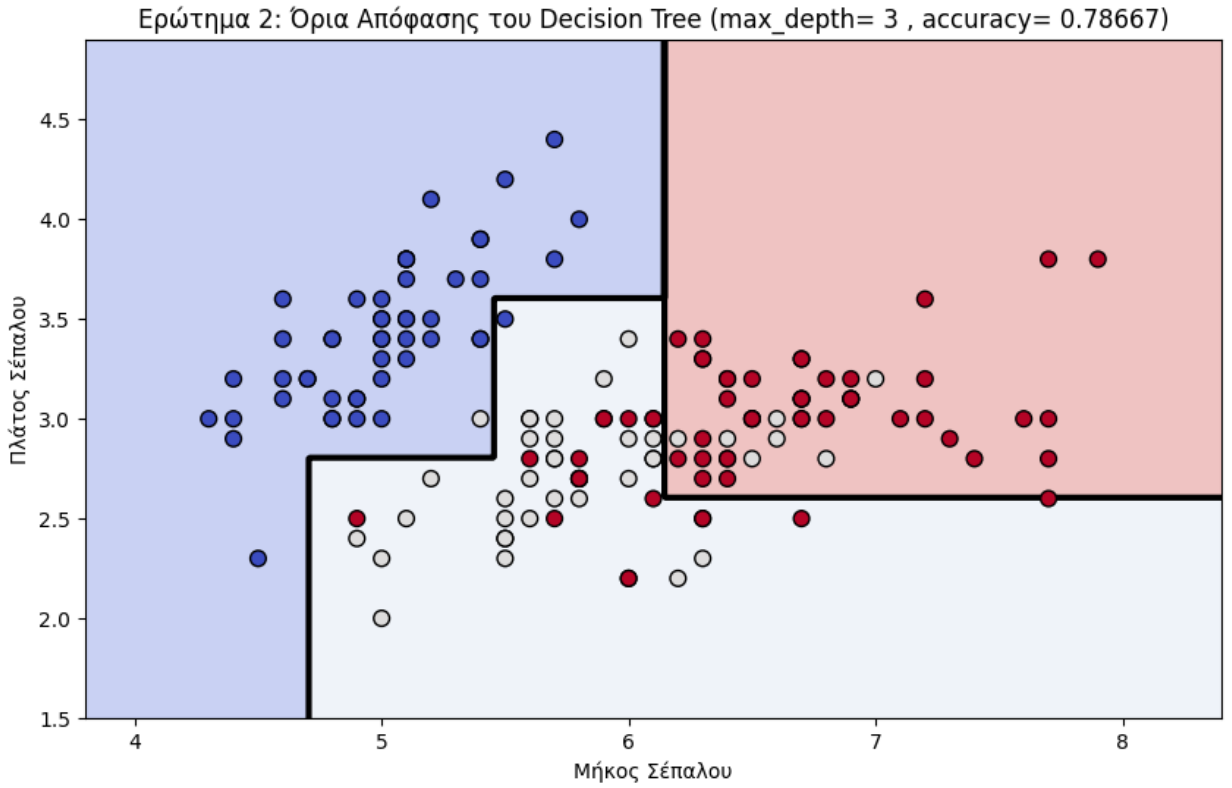
- Δημιουργείται ένας ταξινομητής DecisionTreeClassifier με το αντίστοιχο βάθος.
- Το μοντέλο εκπαιδεύεται χρησιμοποιώντας το σύνολο εκπαίδευσης (training set).
- Το μοντέλο πραγματοποιεί προβλέψεις στο σύνολο δοκιμής (test set).
- Υπολογίζεται το ποσοστό σωστής ταξινόμησης (accuracy) για την αξιολόγηση της απόδοσης του μοντέλου.
- Το ποσοστό αυτό συγκρίνεται με το καλύτερο ποσοστό που έχει επιτευχθεί έως εκείνη τη στιγμή. Αν το νέο ποσοστό είναι υψηλότερο, ενημερώνεται το καλύτερο ποσοστό ακρίβειας (best accuracy) και το αντίστοιχο βάθος (depth) του δέντρου που το πέτυχε.

Με αυτόν τον τρόπο, ο αλγόριθμος εντοπίζει το βέλτιστο βάθος του δέντρου, το οποίο επιτυγχάνει τη μέγιστη απόδοση στο συγκεκριμένο test set. Το καλύτερο ποσοστό σωστής ταξινόμησης που πετύχαμε είναι 78.67% (0.78667).Το βέλτιστο μοντέλο που πετυχαίνει αυτό το ποσοστό είναι το **DecisionTreeClassifier(max_depth=3, random_state=42).** Δηλαδή, το βάθος του δέντρου που δίνει το καλύτερο ποσοστό είναι 3 (max_depth=3).

Ενότητα 1 - Δέντρο Απόφασης

Στο 2ο ερώτημα απεικονίσαμε τα όρια απόφασης του ταξινομητή για το καλύτερο αποτέλεσμα.

Στο διπλανό σχήμα φαίνονται τα όρια απόφασης του ταξινομητή για το καλύτερο αποτέλεσμα. Τα όρια απόφασης επισημαίνονται με μαύρη συνεχή γραμμή για μεγαλύτερη ευκρίνεια. Ο ταξινομητής επιχειρεί να διαχωρίσει τα δείγματα των 3 κατηγοριών(Iris setosa, Iris versicolor, Iris virginica) του φυτού Ίριδα βασιζόμενος σε γραμμικά όρια απόφασης. Παρατηρείται επικάλυψη μεταξύ των δειγμάτων των 3 κατηγοριών. Αυτή η επικάλυψη δυσχεραίνει τον σωστό διαχωρισμό των δειγμάτων, γεγονός που εξηγεί τη μέτρια ακρίβεια ταξινόμησης που επιτυγχάνει ο ταξινομητής (78.67%).



Ενότητα 2 - Τυχαίο Δάσος

Στην ενότητα 2, δημιουργούμε έναν Random Forest ταξινομητή με 100 δέντρα ίδιου μέγιστου βάθους, χρησιμοποιώντας την τεχνική Bootstrap. Στόχος μας είναι να βελτιώσουμε την ακρίβεια ταξινόμησης σε σύγκριση με το αποτέλεσμα της ενότητας 1, όπου χρησιμοποιήσαμε έναν απλό Decision Tree ταξινομητή. Ειδικότερα, το 50% των συνολικών δειγμάτων που χρησιμοποιήσαμε για την εκπαίδευση του Δέντρου Απόφασης στην προηγούμενη ενότητα (σύνολο Α), το χρησιμοποιούμε τώρα για να δημιουργήσουμε 100 νέα υποσύνολα εκπαίδευσης. Κάθε δέντρο του Random Forest εκπαιδεύεται σε ένα ξεχωριστό υποσύνολο, το οποίο αποτελείται από το 50% των δεδομένων του συνόλου Α. Δηλαδή, κάθε υποσύνολο εκπαίδευσης είναι το μισό του Α. Το σύνολο που ταξινομήσαμε (test set) στο προηγούμενο μέρος, το χρησιμοποιούμε και εδώ για αξιολόγηση του αλγορίθμου.

Στον Random Forest ταξινομητή χρησιμοποιούμε τις εξής παραμέτρους :

- **n_estimators=100**: Ορίζουμε ότι το τυχαίο δάσος θα αποτελείται από 100 δέντρα.
- **bootstrap=True**: Χρησιμοποιούμε τη μέθοδο bootstrap.
- **max_samples=0.5**: το 50% του συνόλου Α χρησιμοποιείται για την εκπαίδευση κάθε δέντρου του Random Forest.
- **random_state=42**: Ορίζουμε το seed για την αναπαραγωγή των ίδιων αποτελεσμάτων σε επόμενες εκτελέσεις του κώδικα.

Ενότητα 2 – Τυχαίο Δάσος

Στο 1ο ερώτημα : Ακολουθήσαμε την ίδια λογική με το 1ο ερώτημα της ενότητας 1.

Ο ταξινομητής RandomForestClassifier εκπαιδεύεται για διαφορετικά βάθη, από 1 έως 10. Σε κάθε επανάληψη του βρόχου **for depth in range(1, 11)**:

- Ορίζουμε το μέγιστο βάθος των δέντρων του Random Forest ταξινομητή, **RFC.set_params(max_depth=depth) .**
- Το μοντέλο εκπαιδεύεται χρησιμοποιώντας το σύνολο εκπαίδευσης (training set).
- Το μοντέλο πραγματοποιεί προβλέψεις στο σύνολο δοκιμής (test set).
- Υπολογίζεται η ακρίβεια σωστής ταξινόμησης (accuracy) για την αξιολόγηση της απόδοσης του μοντέλου.
- Αν η τρέχουσα ακρίβεια είναι υψηλότερη από την προηγούμενη, τότε αποθηκεύουμε το νέο καλύτερο ποσοστό ακρίβειας στη μεταβλητή **best_accuracy** και το αντίστοιχο βάθος στη μεταβλητή **best_depth**.
- Στο τέλος, το μοντέλο εκπαιδεύεται ξανά χρησιμοποιώντας το βέλτιστο βάθος **RFC.set_params(max_depth=best_depth)** που προέκυψε από τη διαδικασία, εξασφαλίζοντας έτσι τη βέλτιστη απόδοση του μοντέλου.

Ενότητα 2 - Τυχαίο Δάσος

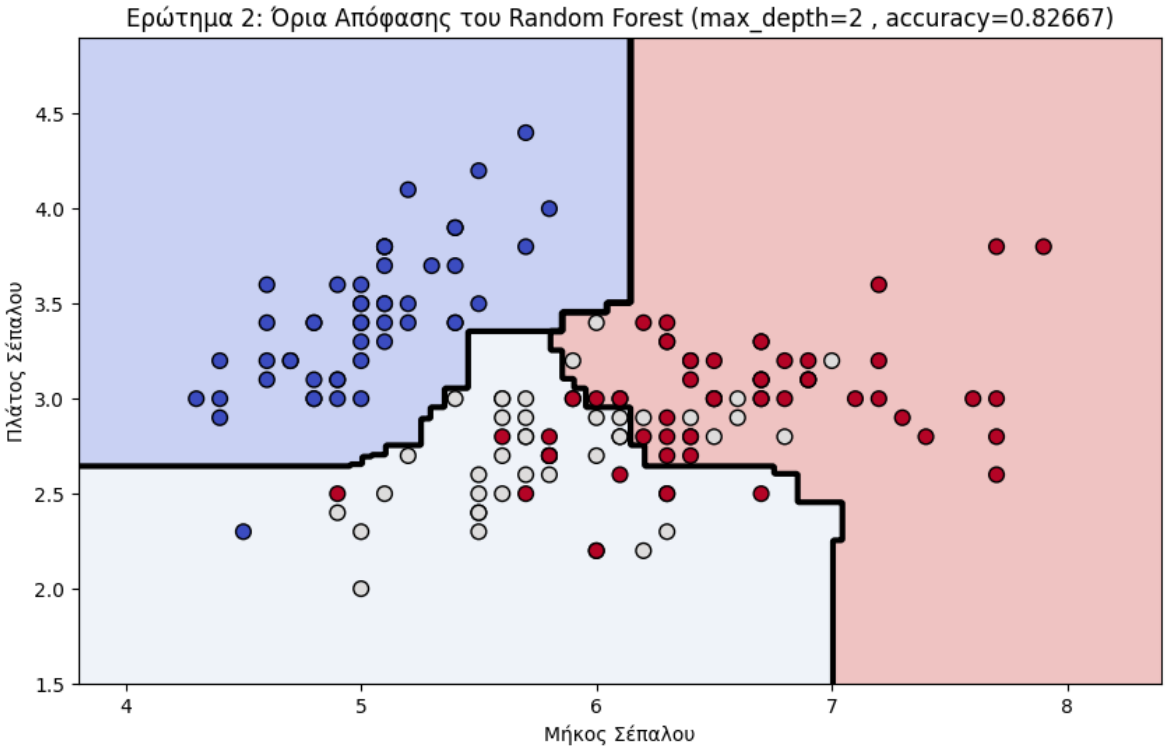
Στο 1ο ερώτημα πετύχαμε καλύτερο ποσοστό σωστής ταξινόμησης 82,67% (0.82667) για βάθος 2. Το καλύτερο μοντέλο είναι το **RandomForestClassifier(max_depth=2, max_samples=0.5, random_state=42) .**

Παρατηρούμε λοιπόν ότι το Τυχαίο Δάσος έδωσε καλύτερα αποτελέσματα σωστής ταξινόμησης από το Δέντρο Απόφασης της ενότητας 1.

Ενότητα 2 - Τυχαίο Δάσος

Στο 2ο ερώτημα απεικονίσαμε τα όρια απόφασης του ταξινομητή Random Forest για το καλύτερο αποτέλεσμα.

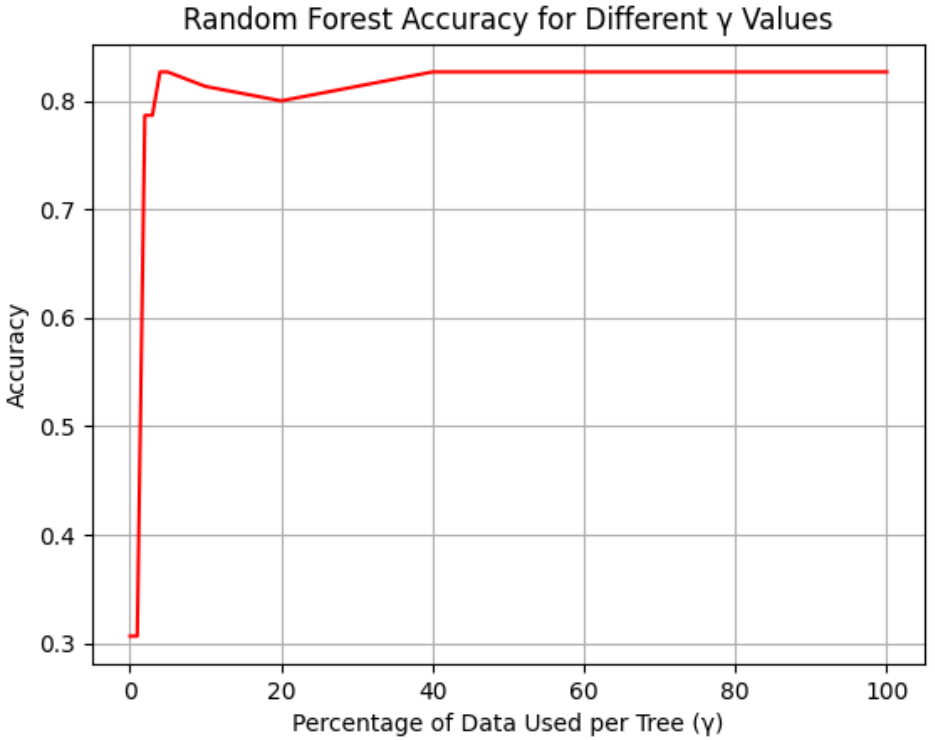
Στο διπλανό σχήμα φαίνονται τα όρια απόφασης του ταξινομητή Random Forest για το καλύτερο αποτέλεσμα. Τα όρια απόφασης επισημαίνονται με μαύρη συνεχή γραμμή για μεγαλύτερη ευκρίνεια. Σε αντίθεση με τον απλό ταξινομητή της ενότητας 1, ο Random Forest επιχειρεί να διαχωρίσει τα δείγματα των 3 κατηγοριών του φυτού Ίριδα βασιζόμενος σε **μη γραμμικά όρια απόφασης**. Με τον τρόπο αυτό δημιουργούνται μη γραμμικές περιοχές απόφασης , οι οποίες προσαρμόζονται πιο αποτελεσματικά πάνω στα ίδια δεδομένα. Έτσι, επιτυγχάνεται καλύτερος διαχωρισμός δεδομένων και αυξάνεται το ποσοστό σωστής ταξινόμησης. Ακόμη, ο Random Forest παρουσιάζει μικρότερη πολυπλοκότητα, καθώς όλα τα δέντρα του έχουν το ίδιο μέγιστο βάθος = 2, το οποίο καθορίστηκε από το **RFC.set_params(max_depth=best_depth).**



Ενότητα 2 - Τυχαίο Δάσος

Στο 3ο ερώτημα μελετήσαμε την απόδοση του αλγορίθμου Random Forest για διάφορα ποσοστά γ. Το γ είναι το ποσοστό του συνόλου Α που χρησιμοποιείται κάθε φορά για την εκπαίδευση κάθε δέντρου του Τυχαίου Δάσους. Χρησιμοποιήσαμε ενδεικτικά 17 διαφορετικές τιμές γ που κυμαίνονται από το 0.05% έως και το 100% των δεδομένων εκπαίδευσης. Για κάθε τιμή γ, εκπαιδεύουμε και αξιολογούμε τον αλγόριθμο Random Forest, δοκιμάζοντας διαφορετικά μέγιστα βάθη (max_depth) ώστε να εντοπίσουμε το βέλτιστο. Με αυτόν τον τρόπο βρίσκουμε το καλύτερο μοντέλο, το οποίο στη συνέχεια το εκπαιδεύουμε και κάνουμε προβλέψεις με αυτό.

Στο διπλανό σχήμα, παραθέτουμε ένα διάγραμμα που απεικονίζει την μεταβολή της ακρίβειας ταξινόμησης συναρτήσει του ποσοστού γ. Παρατηρούμε ότι η ακρίβεια του μοντέλου αυξάνεται καθώς αυξάνεται το ποσοστό γ, φτάνοντας σε μέγιστη τιμή 82,67%. Επίσης, παρατηρούμε μια σταθεροποίηση της ακρίβειας για τιμές του γ μεγαλύτερες από 0,4. Αντίθετα, η ακρίβεια μειώνεται αισθητά για γ < 0,01 γεγονός που υποδηλώνει ανεπαρκή εκπαίδευση λόγω περιορισμένου όγκου δεδομένων ανά δέντρο. Γενικά, συμπεραίνουμε ότι όσο αυξάνεται το ποσοστό γ, τα δέντρα του Random Forest χρησιμοποιούν περισσότερα δεδομένα, γεγονός που συμβάλλει στην καλύτερη εκπαίδευση τους και στον καλύτερο διαχωρισμό των δειγμάτων.



ΜΕΡΟΣ Δ

Εισαγωγή

Η λογική που ακολουθήσαμε στο θέμα Δ είναι εξής :

Αρχικά, χωρίσαμε το συνολικό training set των 8743 δειγμάτων σε 2 υποσύνολα:

- στο σύνολο εκπαίδευσης [X_train, y_train] που είναι το 80% του συνολικού training set
- στο validation set [X_val, y_val] που ταυτίζεται με το 20% του συνολικού training set.

Τα X_train, X_val περιέχουν τα 224 χαρακτηριστικά(features) των δειγμάτων. Τα y_train, y_val περιέχουν τις ετικέτες(labels) των δειγμάτων. Στη συνέχεια, κάνουμε manual hyperparameter tuning για κάθε αλγόριθμο ταξινόμησης που χρησιμοποιήσαμε στα πλαίσια του μέρους Δ της εργασίας. Ουσιαστικά, κατά την διαδικασία του manual hyperparameter tuning, εκπαιδεύουμε διάφορα μοντέλα από κάθε αλγόριθμο ταξινόμησης στο [X_train, y_train] και τα τεστάρουμε στο X_val. Με βάση την ακρίβεια(accuracy) στο validation set, διαλέγουμε το καλύτερο μοντέλο για κάθε αλγόριθμο ταξινόμησης. Έπειτα, διαλέγουμε το συνολικό καλύτερο μοντέλο. Για τη βελτίωση της απόδοσης του συνολικού καλύτερου μοντέλου, χρησιμοποιήσαμε τη μέθοδο PCA (Principal Component Analysis) προκειμένου να μειώσουμε τη διάσταση του dataset, διατηρώντας 139 χαρακτηριστικά (principal components). Ο αριθμός αυτός επιλέχθηκε μετά από διαδικασία tuning στο εύρος [1, 224]. Το εύρος αυτό επιλέχθηκε ώστε να εξεταστεί το σύνολο των 224 χαρακτηριστικών και να προσδιοριστεί το πλήθος των χαρακτηριστικών που μεγιστοποιεί την ακρίβεια του μοντέλου.

Πιο συγκεκριμένα, χρησιμοποιούμε το 80% του συνολικού training set για να εφαρμόσουμε τον αλγόριθμο PCA. Στη συνέχεια, εκπαιδεύουμε ξανά το συνολικό καλύτερο μοντέλο και αξιολογούμε την νέα απόδοσή του στο validation set. Παρατηρούμε ότι παίρνουμε καλύτερο ποσοστό ταξινόμησης από πριν.

Για τον λόγο αυτό, εκπαιδεύουμε ξανά το συνολικό καλύτερο μοντέλο, εφαρμόζοντας πλέον τον αλγόριθμο PCA στο **συνολικό** training set. Τέλος, χρησιμοποιούμε το εκπαιδευμένο μοντέλο για να κάνουμε προβλέψεις στο test set. Έτσι, παράγουμε το αρχείο labels25.npy.

Αλγόριθμοι ταξινόμησης

Οι αλγόριθμοι ταξινόμησης που χρησιμοποιήθηκαν στο Μέρος Δ είναι οι εξής :

- k-Nearest Neighbors (KNN)
- Gaussian Naive Bayes
- Multinomial Logistic Regression (Λογιστική Παλινδρόμηση για Πολλαπλές Κλάσεις)
- Decision Trees (Δέντρα Απόφασης)
- Random Forest (Τυχαία Δάση)
- MLP (Multilayer Perceptron)
- SVM (Support Vector Machine)

Οι συναρτήσεις των παραπάνω αλγορίθμων είναι υλοποιημένες στην βιβλιοθήκη sklearn.

k-Nearest Neighbors (KNN)

Ο k-Nearest Neighbors(k-NN) είναι ένας αλγόριθμος που βασίζεται στην ιδέα της ομοιότητας (εγγύτητας) μεταξύ των δεδομένων. Χρησιμοποιείται ευρέως λόγω της ικανότητάς του να διαχωρίζει δεδομένα με μη γραμμικά όρια απόφασης. Ο αλγόριθμος K-NN δεν εκπαιδεύει κάποιο μοντέλο με την παραδοσιακή έννοια. Αντί να προχωρά σε διαδικασία εκπαίδευσης, αποθηκεύει απλώς τα δεδομένα εκπαίδευσης κατά την κλήση της εντολής fit(). Όταν απαιτείται να κάνει πρόβλεψη για κάποιο νέο δείγμα, ο αλγόριθμος αναζητά τα k πιο κοντινά δείγματα χρησιμοποιώντας την εντολή predict(). Δηλαδή, η "μάθηση" συμβαίνει μόνο κατά την κλήση της predict().

Ο k-NN λειτουργεί ως εξής:

- Για ένα νέο δείγμα προς ταξινόμηση, υπολογίζεται η απόστασή του από όλα τα δείγματα του συνόλου εκπαίδευσης, χρησιμοποιώντας ένα κατάλληλο μέτρο απόστασης (π.χ. Ευκλείδεια απόσταση).
- Εντοπίζονται οι k πλησιέστεροι γείτονες του δείγματος, ανεξάρτητα από την κλάση στην οποία ανήκουν.
- Από αυτούς τους k γείτονες, υπολογίζεται ο αριθμός των δειγμάτων $\boldsymbol{k_i}$ που ανήκουν σε κάθε κλάση $\boldsymbol{\omega_i}$.
- Το νέο δείγμα ταξινομείται στην κλάση που εμφανίζεται πιο συχνά μεταξύ των k γειτόνων (πλειοψηφική ψήφος).

Πλεονεκτήματα:

- Απλή και εύκολη στην κατανόηση μέθοδος.
- Κατάλληλη μέθοδος για μη γραμμικά διαχωρίσιμα δεδομένα.

Μειονεκτήματα:

- Αργή διαδικασία πρόβλεψης ειδικά σε μεγάλα σύνολα δεδομένων
- Χρειάζεται κατάλληλη επιλογή του k για καλή απόδοση.

k-Nearest Neighbors (KNN)

- Συνάρτηση αλγόριθμου ταξινόμησης : KNeighborsClassifier
- Οι συνολικές υπερπαράμετροι που δοκιμάστηκαν στο tuning είναι οι εξής :
  - 'n_neighbors': [3, 5, 7, 10]
  - 'weights': ['uniform', 'distance']
  - 'metric': ['euclidean', 'manhattan']
- Οι υπερπαράμετροι του καλύτερου μοντέλου που βρέθηκαν με το tuning είναι :
  - 'n_neighbors': [5]
  - 'weights': ['distance']
  - 'metric': ['euclidean']
- Η ακρίβεια του καλύτερου μοντέλου στο validation set είναι : **84.96%**.

Gaussian Naive Bayes

Ο Gaussian Naive Bayes είναι ένας αλγόριθμος ταξινόμησης που βασίζεται στο Θεώρημα του Bayes και κάνει την υπόθεση ότι τα χαρακτηριστικά (features) x_i του δείγματος $x = [x_1, x_2, \dots, x_N]$ με i = 1,..,N είναι ανεξάρτητα μεταξύ τους. Παρά την υπόθεση αυτή, ο αλγόριθμος μπορεί να είναι ανθεκτικός στην παραβίαση της ανεξαρτησίας των χαρακτηριστικών και να αποδίδει καλά σε πολλές εφαρμογές στην πράξη.

Επιπλέον, ο Gaussian Naive Bayes υποθέτει ότι τα χαρακτηριστικά των δειγμάτων ακολουθούν κανονική κατανομή (Gaussian distribution). Οι υπολογισμοί του αλγορίθμου βασίζονται στην εκτίμηση της μέσης τιμής και της διακύμανσης των χαρακτηριστικών.

Πλεονεκτήματα:

- Πολύ γρήγορο και αποτελεσματικό για μεγάλα σύνολα δεδομένων.
- Γρήγορη Εκπαίδευση και Πρόβλεψη: Ο αλγόριθμος υπολογίζει απλά μεγέθη όπως τη μέση τιμή και τη διακύμανση για κάθε χαρακτηριστικό, σε αντίθεση με άλλους αλγόριθμους που απαιτούν πιο πολύπλοκους υπολογισμούς (π.χ. νευρωνικά δίκτυα).
- Είναι ανθεκτικός αλγόριθμος στο πρόβλημα των υψηλών διαστάσεων (curse of dimensionality). Δηλαδή, μπορεί να αποδώσει καλά ακόμα και όταν ο αριθμός των χαρακτηριστικών των δειγμάτων είναι πολύ μεγάλος.

Μειονεκτήματα:

- Χαμηλή ακρίβεια σε πολύπλοκα προβλήματα: Ο αλγόριθμος υποθέτει ότι τα δεδομένα ακολουθούν κανονική κατανομή, κάτι που μπορεί να μην ισχύει. Αυτό μπορεί να οδηγήσει σε χαμηλότερη ακρίβεια.

Gaussian Naive Bayes

- Συνάρτηση αλγόριθμου ταξινόμησης : GaussianNB
- Οι συνολικές υπερπαράμετροι που δοκιμάστηκαν στο tuning είναι οι εξής :
  - 'var_smoothing': [1e-9, 1e-8, 1e-7]
- Οι υπερπαράμετροι του καλύτερου μοντέλου που βρέθηκαν με το tuning είναι :
  - 'var_smoothing': [1e-9]
- Η ακρίβεια του καλύτερου μοντέλου στο validation set είναι : **69.87%.**

Multinomial Logistic Regression

Η Λογιστική Παλινδρόμηση για Πολλαπλές Κλάσεις επεκτείνει τη δυαδική λογιστική παλινδρόμηση σε προβλήματα με πολλές κλάσεις. Στην περίπτωση αυτή, οι πιθανότητες ενός δείγματος να ανήκει σε μία από τις Κ κλάσεις του προβλήματος ταξινόμησης υπολογίζονται μέσω του μετασχηματισμού softmax. Αυτός ο μετασχηματισμός μετατρέπει τις παραμέτρους ενεργοποίησης $\boldsymbol{a_k}$ για κάθε κλάση $\boldsymbol{C_k}$ σε πιθανότητες, όπως εκφράζεται από την εξίσωση:

$$\boldsymbol{p(C_k|\varphi) = \frac{exp(a_k)}{\sum_j exp(a_j)}}$$

Οι παράμετροι ενεργοποίησης $\boldsymbol{a_k}$ ορίζονται ως το εσωτερικό γινόμενο του διανύσματος χαρακτηριστικών $\boldsymbol{\varphi}$ με το διάνυσμα βαρών $\boldsymbol{w_k}$ της αντίστοιχης κλάσης : $\boldsymbol{a_k = w_k^T \cdot \varphi}$.

Η εκπαίδευση του μοντέλου γίνεται με βάση τη συνάρτηση σφάλματος της διεντροπίας (cross-entropy). Η συνάρτηση cross-entropy μετρά την απόκλιση μεταξύ της σωστής κλάσης του δείγματος και του λογαρίθμου της πιθανότητας που υπολόγισε το μοντέλο για τη σωστή κλάση του δείγματος. Όσο μεγαλύτερη είναι η πιθανότητα που υπολόγισε για τη σωστή κλάση, τόσο μικρότερη είναι η τιμή της συνάρτησης cross-entropy. Στην ιδανική περίπτωση, αν η πιθανότητα για τη σωστή κλάση είναι 1, τότε η cross-entropy είναι 0, πράγμα που σημαίνει ότι το μοντέλο έκανε τέλεια πρόβλεψη. Αν η πιθανότητα για τη σωστή κλάση είναι χαμηλή και κοντά στο 0, τότε η cross-entropy θα έχει μεγάλη τιμή και αυτό υποδεικνύει ότι το μοντέλο έκανε λάθος πρόβλεψη. Ο στόχος είναι η cross-entropy να είναι όσο το δυνατόν πιο κοντά στο 0.

Multinomial Logistic Regression

Πλεονεκτήματα:

- Η λογιστική παλινδρόμηση πολλαπλών κλάσεων είναι εύκολη στην υλοποίηση.
- Καλή Απόδοση για Γραμμικά Δεδομένα: Όταν τα δεδομένα είναι γραμμικά διαχωρίσιμα ή σχεδόν γραμμικά, η λογιστική παλινδρόμηση είναι ένα ισχυρό εργαλείο.

Μειονεκτήματα:

- Η λογιστική παλινδρόμηση υποθέτει ότι η σχέση μεταξύ των χαρακτηριστικών του δείγματος είναι γραμμική. Αυτό μπορεί να είναι περιοριστικό όταν τα δεδομένα είναι μη γραμμικά.
- Η απόδοση της λογιστικής παλινδρόμησης μπορεί να μειωθεί όταν τα δεδομένα δεν είναι σωστά προεπεξεργασμένα.
- Περιορισμένη ικανότητα με πολύπλοκα δεδομένα: Όταν η σχέση μεταξύ των δειγμάτων και των κλάσεων είναι πολύπλοκη ή μη γραμμική, ο αλγόριθμος δεν αποδίδει καλά.

Multinomial Logistic Regression

- Συνάρτηση αλγόριθμου ταξινόμησης : LogisticRegression
- Οι συνολικές υπερπαράμετροι που δοκιμάστηκαν στο tuning είναι οι εξής :
  - 'C': [0.01, 0.1, 1, 10, 100]
  - 'solver': ['lbfgs', 'liblinear', 'saga', 'newton-cg']
  - 'max_iter': [1000, 2000]
  - 'class_weight': [None, 'balanced']
  - 'tol': [1e-4, 1e-3, 1e-2]
- Οι υπερπαράμετροι του καλύτερου μοντέλου που βρέθηκαν με το tuning είναι :
  - 'C': [0.1]
  - 'solver': ['lbfgs']
  - 'max_iter': [1000]
  - 'class_weight': [None]
  - 'tol': [1e-2]
- Η ακρίβεια του καλύτερου μοντέλου στο validation set είναι : **78.67%**.

Σημείωση: Από την έκδοση 1.5 του scikit-learn και μετά, η παράμετρος multi_class (για την ταξινόμηση σε πολλές κλάσεις) στον αλγόριθμο Logistic Regression έχει αποσυρθεί. Ο αλγόριθμος χρησιμοποιεί αυτόματα την τιμή 'multinomial', και επομένως χρησιμοποιείται by default για ταξινόμηση πολλών κλάσεων.

Decision Trees

Τα Δέντρα Απόφασης είναι μοντέλα μηχανικής μάθησης που βασίζονται σε μια σειρά ερωτήσεων για τη λήψη αποφάσεων. Κάθε κόμβος του δέντρου αντιπροσωπεύει μια ερώτηση, ενώ τα φύλλα περιέχουν την τελική πρόβλεψη/απόφαση του Δέντρου. Τα δεδομένα διαιρούνται σε μικρότερα υποσύνολα μέσω ερωτήσεων που αφορούν τα χαρακτηριστικά τους. Ο μηχανισμός του δέντρου διαχωρίζει τον χώρο χαρακτηριστικών(feature space) σε διάφορες περιοχές, οι οποίες συχνά είναι ορθογώνιες. Τα Δέντρα Απόφασης, συνήθως είναι αδύναμοι ταξινομητές όταν χρησιμοποιούνται ως αυτόνομα μοντέλα. Ωστόσο, η συνδυαστική χρήση πολλών Δέντρων Απόφασης, όπως στον Random Forest ταξινομητή, μπορεί να βελτιώσει σημαντικά την ακρίβεια των προβλέψεων.

Πλεονεκτήματα:

- Τα Δέντρα Απόφασης είναι διαισθητικά και εύκολα κατανοητά, καθώς βασίζονται σε μια σειρά απλών ερωτήσεων που οδηγούν σε αποφάσεις. Αυτό επιτρέπει την εύκολη ερμηνεία της διαδικασίας λήψης αποφάσεων.
- Η εκπαίδευση του μοντέλου είναι σχετικά γρήγορη.

Μειονεκτήματα:

- Είναι επιρρεπή σε υπερπροσαρμογή (Overfitting), ιδιαίτερα όταν τα δέντρα είναι πολύ βαθιά.
- Όταν χρησιμοποιούνται μεμονωμένα, έχουν περιορισμένη ικανότητα γενίκευσης σε νέα δεδομένα.
- Τα Δέντρα Απόφασης ενδέχεται να μην προσφέρουν την καλύτερη ακρίβεια σε μεγάλα ή πολύπλοκα σύνολα δεδομένων, σε σύγκριση με άλλες τεχνικές.

Decision Trees

- Συνάρτηση αλγόριθμου ταξινόμησης : DecisionTreeClassifier
- Οι συνολικές υπερπαράμετροι που δοκιμάστηκαν στο tuning είναι οι εξής :
  - 'max_depth': range(1, 50)
- Οι υπερπαράμετροι του καλύτερου μοντέλου που βρέθηκαν με το tuning είναι :
  - 'max_depth': [11]
- Η ακρίβεια του καλύτερου μοντέλου στο validation set είναι : **65.24%.**

Random Forest

Τα Τυχαία Δάση είναι ένα σύνολο από Δέντρα Απόφασης (DT). Κάθε δέντρο του Τυχαίου Δάσους εκπαιδεύεται χρησιμοποιώντας την τεχνική bootstrapping. Η τεχνική αυτή επιλέγει τυχαία υποσύνολα δεδομένων από το σύνολο εκπαίδευσης. Έτσι, κάθε δέντρο εκπαιδεύεται σε διαφορετικό υποσύνολο των δεδομένων εκπαίδευσης , πράγμα που σημαίνει ότι «βλέπει» διαφορετικά δεδομένα από τα υπόλοιπα δέντρα του δάσους. Επομένως, αυτή η διαδικασία επιτρέπει στο τυχαίο δάσος να μάθει από ποικιλία δεδομένων, μειώνοντας έτσι την πιθανότητα υπερπροσαρμογής (overfitting). Ακόμη, σε κάθε κόμβο των Δέντρων Απόφασης του Τυχαίου Δάσους, δεν εξετάζονται όλα τα διαθέσιμα χαρακτηριστικά. Αντίθετα, επιλέγεται τυχαία ένα υποσύνολο χαρακτηριστικών (feature bagging). Μόνο τα χαρακτηριστικά αυτού του υποσυνόλου, χρησιμοποιούνται για τον προσδιορισμό του καλύτερου χαρακτηριστικού για τον διαχωρισμό των δεδομένων. Αυτή η διαδικασία μειώνει τη συσχέτιση μεταξύ των δέντρων και κατά συνέπεια τον κίνδυνο του overfitting. Η τελική πρόβλεψη γίνεται μέσω της πλειοψηφίας των αποφάσεων όλων των δέντρων του δάσους.

Πλεονεκτήματα:

- Πολύ ακριβές μοντέλο λόγω του συνδυασμού πολλών δέντρων.
- Αντιμετωπίζει αποτελεσματικά την υπερπροσαρμογή (overfitting).

Μειονεκτήματα:

- Δύσκολη ερμηνεία των τελικών αποτελεσμάτων : Τα Τυχαία Δάση αποτελούνται από έναν μεγάλο αριθμό Δέντρων Απόφασης, τα οποία συνδυάζουν τα αποτελέσματά τους για να παράγουν μια τελική πρόβλεψη. Συνεπώς, είναι δύσκολο να κατανοήσουμε πώς το μοντέλο έφτασε σε μια συγκεκριμένη πρόβλεψη.
- Απαιτεί μεγάλη υπολογιστική ισχύ και μνήμη για μεγάλα δεδομένα.
- Μπορεί να γίνει αργό όταν ο αριθμός των δέντρων είναι μεγάλος.

Random Forest

- Συνάρτηση αλγόριθμου ταξινόμησης : RandomForestClassifier
- Οι συνολικές υπερπαράμετροι που δοκιμάστηκαν στο tuning είναι οι εξής :
  - 'n_estimators': [200, 300, 400]
  - 'max_depth': [10, 20, 30, 40, 50]
  - 'max_samples': [0.7, 1.0]
- Οι υπερπαράμετροι του καλύτερου μοντέλου που βρέθηκαν με το tuning είναι :
  - 'n_estimators': [300]
  - 'max_depth': [40]
  - 'max_samples': [1.0]
- Η ακρίβεια του καλύτερου μοντέλου στο validation set είναι : **81.93%.**

MLP

Το MLP αποτελείται από αρκετά επίπεδα, καθένα από τα οποία περιλαμβάνει νευρώνες οι οποίοι συνδέονται πλήρως με τους νευρώνες του επόμενου επιπέδου. Το MLP έχει τα παρακάτω βασικά επίπεδα:

Επίπεδο Εισόδου (Input Layer) : Δέχεται τα χαρακτηριστικά εισόδου (features).

Κρυφά Επίπεδα (Hidden Layers) : Περιέχουν νευρώνες με μη γραμμικές συναρτήσεις ενεργοποίησης. Έτσι, αυξάνουν την ικανότητα του δικτύου να αναπαριστά σύνθετες σχέσεις στα δεδομένα.

Επίπεδο εξόδου (Output Layer) : Παράγει την τελική πρόβλεψη/ταξινόμηση του μοντέλου.

Μηχανισμός Λειτουργίας

- **Forward Propagation**: Τα δεδομένα περνούν από το επίπεδο εισόδου στα κρυφά επίπεδα. Κάθε νευρώνας των κρυφών επιπέδων , υπολογίζει ένα σταθμισμένο άθροισμα των εισόδων του, προσθέτει bias και εφαρμόζει μια μη γραμμική συνάρτηση ενεργοποίησης. Το αποτέλεσμα από κάθε κρυφό επίπεδο προχωρά στο επόμενο επίπεδο μέχρι να φτάσει στο επίπεδο εξόδου.
- **Υπολογισμός Σφάλματος πρόβλεψης**: Εκτιμούμε την ποιότητα της πρόβλεψης του μοντέλου για τη σωστή κλάση του δείγματος χρησιμοποιώντας την cross-entropy. Όπως και στη Multinomial Logistic Regression , όταν η cross-entropy είναι μικρή τότε έχει γίνει καλή πρόβλεψη. Όταν η cross-entropy είναι μεγάλη τότε έχει γίνει λανθασμένη πρόβλεψη.
- **Backpropagation**: Η εκπαίδευση ενός MLP περιλαμβάνει τη ρύθμιση των βαρών του δικτύου με σκοπό την ελαχιστοποίηση της cross-entropy.

MLP

Πλεονεκτήματα :

- Το MLP αναγνωρίζει και ταξινομεί πολύπλοκα μοτίβα.
- Αν έχει αρκετά δεδομένα εκπαίδευσης, μπορεί να γενικεύσει αποτελεσματικά.

Μειονεκτήματα :

- Η εκπαίδευση ενός μεγάλου δικτύου απαιτεί πολλούς υπολογιστικούς πόρους.
- Κίνδυνος Υπερπροσαρμογής αν δεν υπάρχουν πολλά δεδομένα εκπαίδευσης.

- Συνάρτηση αλγόριθμου ταξινόμησης : MLPClassifier
- Οι συνολικές υπερπαράμετροι που δοκιμάστηκαν στο tuning είναι οι εξής :
  - 'hidden_layer_sizes': [(50,), (100,), (200,), (300,), (400,), (500,), (600,), (700,),
    (100, 100), (200, 200), (300, 300), (400, 400), (500, 500), (600, 600),
    (700, 700), (800, 800), (900, 900),
    (1000, 1000), (100, 100, 50), (100, 100, 100), (200, 200, 200), (300, 300, 200),
    (300, 300, 300), (400, 400, 400), (500, 500, 500), (600, 600, 600),
    (700, 700, 700), (800, 800, 800), (900, 900, 900), (1000, 1000, 1000)]

MLP

- Οι συνολικές υπερπαράμετροι που δοκιμάστηκαν στο tuning είναι οι εξής (Συνέχεια):
  - 'activation': ['relu', 'logistic', 'tanh’]
  - 'solver': ['adam', 'lbfgs', 'sgd’]
  - 'learning_rate': ['constant', 'adaptive’]
  - 'max_iter': [2000]
  - 'alpha': [0.00001, 0.0001, 0.001]
  - 'epsilon': [1e-6, 1e-7, 1e-8, 1e-9, 1e-10]
- Οι υπερπαράμετροι του καλύτερου μοντέλου που βρέθηκαν με το tuning είναι :
  - 'hidden_layer_sizes': [(300,)]
  - 'activation': ['relu’]
  - 'solver': ['adam’]
  - 'learning_rate': ['constant’]
  - 'max_iter': [2000]
  - 'alpha': [0.0001]
  - 'epsilon': [1e-7]
- Η ακρίβεια του καλύτερου μοντέλου στο validation set είναι : **86.34%.**

SVM

Βασική λειτουργία του SVM:

- Χρησιμοποιεί ένα υπερεπίπεδο (hyperplane) για να διαχωρίσει τα δεδομένα σε διαφορετικές κλάσεις.
- Επιλέγει το υπερεπίπεδο με το μέγιστο περιθώριο (maximum margin) μεταξύ των κλάσεων. Το μέγιστο περιθώριο είναι η μεγαλύτερη απόσταση μεταξύ του υπερεπιπέδου και των πλησιέστερων σημείων από κάθε κλάση(τα υποστηρικτικά διανύσματα).
- Μπορεί να διαχειριστεί μη γραμμικά διαχωρίσιμα δεδομένα μέσω των πυρήνων (kernels). Οι πυρήνες επιτρέπουν την μετατροπή των δεδομένων σε υψηλότερες διαστάσεις, όπου τα δεδομένα μπορεί να γίνουν γραμμικά διαχωρίσιμα. Οι πιο συνηθισμένοι πυρήνες είναι ο γραμμικός, ο πολυωνυμικός και ο RBF.

Πλεονεκτήματα:

- Καλή απόδοση για δεδομένα υψηλών διαστάσεων. Το SVM είναι πολύ αποτελεσματικό όταν τα δεδομένα περιέχουν πολλές διαστάσεις (δηλαδή πολλά χαρακτηριστικά).
- Είναι κατάλληλος αλγόριθμος για μικρά και μεσαία σύνολα δεδομένων.

Μειονεκτήματα:

- Υπολογιστικά απαιτητικό σε μεγάλα σύνολα δεδομένων.
- Απαιτεί βελτιστοποίηση των υπερπαραμέτρων του για καλύτερη απόδοση.
- Δύσκολη ερμηνεία του μοντέλου : το SVM μπορεί να λειτουργεί σε υψηλές διαστάσεις και μπορεί να χρησιμοποιεί πυρήνες για μη γραμμικό διαχωρισμό των δεδομένων. Επομένως, η ερμηνεία του τελικού μοντέλου μπορεί να είναι δύσκολη.

SVM

- Συνάρτηση αλγόριθμου ταξινόμησης : SVC (από το Support Vector Classifier)
- Οι συνολικές υπερπαράμετροι που δοκιμάστηκαν στο tuning είναι οι εξής :
  - 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]
  - 'gamma': ['scale', 'auto']
  - 'kernel': ['linear', 'rbf', 'poly']
  - 'tol': [1e-9, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2]
  - 'coef0': [0.001, 0.01, 0.1, 0.2, 0.5, 1.0]
  - 'class_weight': [None, 'balanced']
- Οι υπερπαράμετροι του καλύτερου μοντέλου που βρέθηκαν με το tuning είναι :
  - 'C': [1]
  - 'gamma': ['scale']
  - 'kernel': ['poly']
  - 'tol': [1e-3]
  - 'coef0': [0.2]
  - 'class_weight': ['balanced']
- Η ακρίβεια του καλύτερου μοντέλου στο validation set είναι : **87.36%.**

Συμπεράσματα

- Από τα παραπάνω, συμπεραίνουμε ότι το συνολικό καλύτερο μοντέλο είναι το **SVM** με υπερπαραμέτρους **{ C = 1, gamma = 'scale' , kernel = 'poly', tol = 0.001, coef0 = 0.2, class_weight = 'balanced' }**. Το μοντέλο αυτό πέτυχε την καλύτερη ακρίβεια **(87.36%)** στο validation set**.**
- Στη συνέχεια, εφαρμόζουμε τη μέθοδο PCA στο 80% του training set και εκπαιδεύουμε ξανά το μοντέλο. Ως αποτέλεσμα, η ακρίβεια στο validation set αυξήθηκε στο **88.56%**, σημειώνοντας βελτίωση 1.2%. Για να εκμεταλλευτούμε τη βελτίωση αυτή, αποφασίσαμε να επανεκπαιδεύσουμε το μοντέλο χρησιμοποιώντας το συνολικό training set των 8743 δειγμάτων και την τεχνική PCA, αντί να προχωρήσουμε σε τυπική εκπαίδευση με τη συνάρτηση fit(). Το τελικό εκπαιδευμένο SVM μοντέλο χρησιμοποιήθηκε για να πραγματοποιήσει προβλέψεις στο test set, παράγοντας το αρχείο labels25.npy.