Εργασία-Αναγνώριση Προτύπων και Μηχανική Μάθηση

ΟΜΑΔΑ 11

ΠΑΝΑΓΙΩΤΗΣ ΜΠΕΛΑΝΤΑΚΗΣ(10305)

ΑΛΕΞΑΝΔΡΟΣ ΦΩΤΙΑΔΗΣ(10392)

Ακαδημαϊκό Έτος: 2024-2025

Περιεχόμενα

Μέρος Α: Ταξινομητής Μέγιστης Πιθανοφάνειας(MLE)

1.1.Περιγραφή Προβλήματος

Πρόβλημα δυαδικής ταξινόμησης για το αν χρήστες βιντεοπαιχνιδιών αισθάνονται στρες ή όχι με τα εξής δεδομένα:

- Κλάσεις ω1: χωρίς στρες και ω2: με στρες
- Δείκτης x για την αξιολόγηση του στρες με pdf:

$$p(x|\theta) = \frac{1}{\pi}\frac{1}{1+(\mathrm{x}-\theta)^2}$$

- Prior πιθανότητες: p(ω1)= 7/12 και p(ω2) = 5/12
- Datasets D1 = [2.8,−0.4,−0.8,2.3,−0.3,3.6,4.1] και D2 = [−4.5,−3.4,−3.1,−3.0,−2.3] που αντιστοιχούν στις κλάσεις ω1 και ω2 αντίστοιχα
- Συνάρτηση διάκρισης: $g(x) = logP\left(x\middle|\hat{\theta}_1\right) - logP\left(x\middle|\hat{\theta}_2\right) + logP(\omega 1) - logP(\omega 2)$

1.2.Επίλυση του προβλήματος

- Ζητούμενο του προβλήματος είναι να εκτιμήσουμε τις παραμέτρους θ1 και θ2, οι οποίες μεγιστοποιούν τις συναρτήσεις πυκνότητας πιθανότητας p(D1|θ) και p(D2|θ) αντίστοιχα.
- Υπολογίζουμε, αρχικά τις pdf, θεωρώντας ότι τα δείγματα είναι i.i.d:

$$p(D|\theta) = p(x_1, x_2, \dots, x_{\mathrm{N}}|\theta) = \prod_{1}^{n} p(x_n|\theta)$$

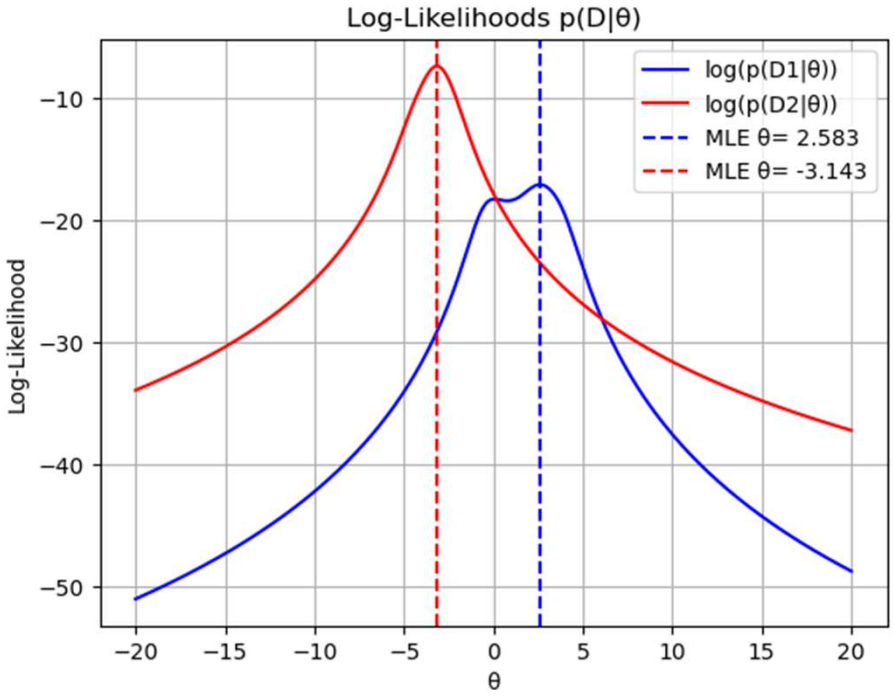
- Βρίσκουμε τη log-likelihood:

$$l(\theta) = \ln \prod_{1}^{\mathrm{N}} p(x_n|\theta) = \sum_{1}^{\mathrm{N}} \ln p(x_n|\theta)$$

1.2.Επίλυση του προβλήματος(συν.)

- Λύνοντας την εξίσωση $\nabla_\theta l(\theta) = 0$, υπολογίζουμε τους εκτιμητές $\hat{\theta}_1$και $\hat{\theta}_2$.
- Από την ανάλυση σε python, παίρνουμε:

$$\hat{\theta}_1 = 2.582582582582582$$
$$\hat{\theta}_2 = -3.143143143143142$$



1.2.Επίλυση του προβλήματος(συν.)

- Στη συνέχεια, χρησιμοποιούμε τους δύο εκτιμητές για τον ορισμό της συνάρτησης διάκρισης

$$g(x) = logP\left(x\middle|\hat{\theta}_1\right) - logP\left(x\middle|\hat{\theta}_2\right) + logP(\omega 1) - logP(\omega 2) = g_1(x) - g_2(x)$$

αξιοποιώντας τον κανόνα ταξινόμησης GBR:

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}}\frac{p(\omega_2)}{p(\omega_1)} \text{ όπου } \lambda_{12} = \lambda_{21} = 1 \; και \; \lambda_{11} = \lambda_{22} = 0$$

- Επιπλέον από το Θεώρημα Bayes $p(\omega_i|x) = \frac{p(\omega_i)p(x|\omega_i)}{p(x)}$
- Έτσι, ο κανόνας γίνεται $p(\omega_1|x) > p(\omega_2|x)$

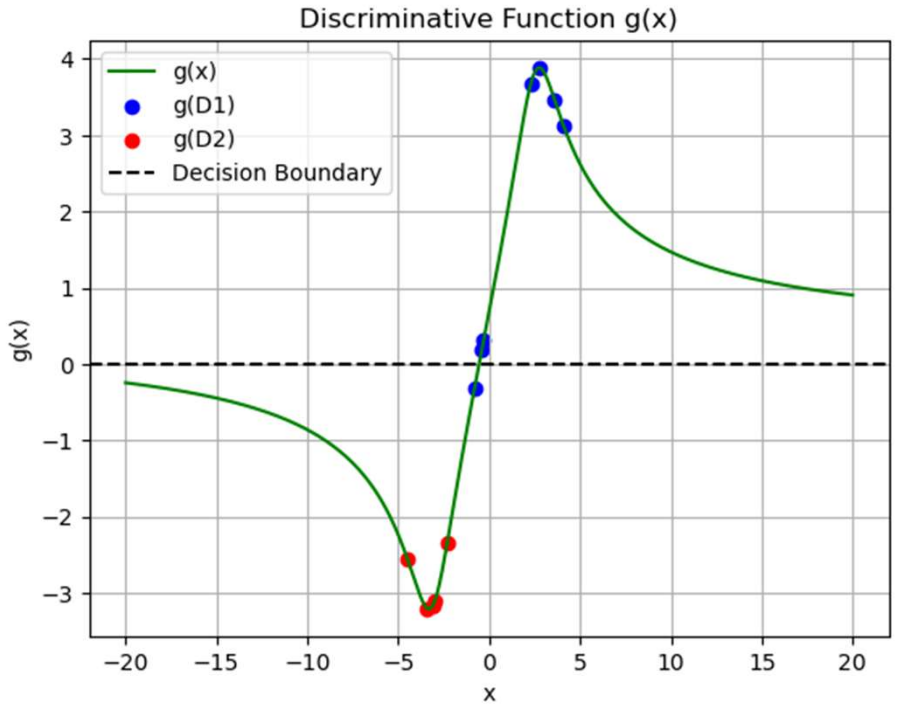
1.2.Επίλυση του προβλήματος(συν.)

- Τελικά και με αντικατάσταση του $p(x|\omega_i)$ $\mu\varepsilon$ τo $p(x|\hat{\theta}_i)$ παίρνουμε τον τελικό κανόνα απόφασης:

Αν $p(\omega_1|x) > p(\omega_2|x) \Leftrightarrow g_1(x) > g_2(x)$

$$\Leftrightarrow \boldsymbol{g(x) > 0}$$

τότε αποφασίζουμε ότι το x ανήκει στην κλάση ω1, αλλιώς ανήκει στην κλάση ω2.



1.2.Επίλυση του προβλήματος(συν.)

Παρακάτω παρατίθεται ένα dataframe το οποίο περιέχει τους δείκτες για καθένα dataset με την κλάση στην οποία θα έπρεπε να ανήκουν καθώς και την κλάση στην οποία τους αντιστοίχισε ο ταξινομητής:

D1

x	g(x)	Expected Class	Class
2.8	3.882679861313016	ω1	ω1
-0.4	0.18734551340020866	ω1	ω1
-0.8	-0.31428294962355885	ω1	ω2
2.3	3.6815557245112016	ω1	ω1
-0.3	0.31191573270416706	ω1	ω1
3.6	3.4647150087161034	ω1	ω1
4.1	3.120767756844003	ω1	ω1

D2

x	g(x)	Expected Class	Class
-4.5	-2.554366501084412	ω2	ω2
-3.4	-3.2048987235041513	ω2	ω2
-3.1	-3.1669776252937933	ω2	ω2
-3.0	-3.1141309013796823	ω2	ω2
-2.3	-2.338953514286934	ω2	ω2

1.2.Επίλυση του προβλήματος(συν.)

Από την απεικόνιση της g καθώς και των σημείων g(D) για καθένα από τα δύο datasets έχουμε τις εξής παρατηρήσεις:

- Το όριο απόφασης είναι το 0. Αυτό σημαίνει ότι οι θετικές τιμές του x ταξινομούνται στην κλάση ω1 ενώ οι αρνητικές στην ω2, όπως προέκυψε και από τη θεωρητική ανάλυση.
- 11/12 τιμές του συνολικού dataset ταξινομούνται σωστά εκτός του x = -0.8, για το οποίο προκύπτει ότι g(-0.8) < 0 οπότε ταξινομείται στην κλάση ω2, λανθασμένα αφού ανήκει στο dataset D1.

Μέρος Β: Μπεϋζιανός Ταξινομητής

2.1.Περιγραφή Προβλήματος

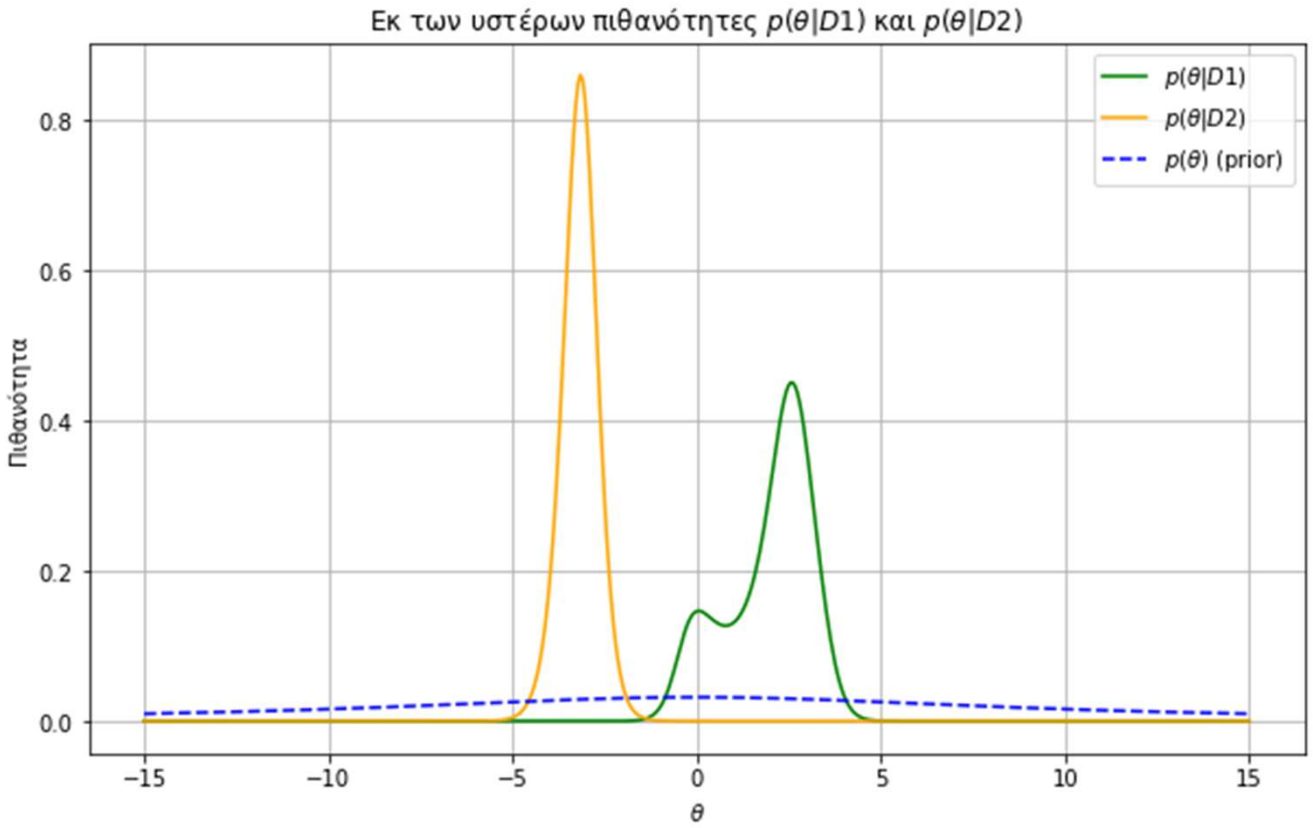
Πρόβλημα δυαδικής ταξινόμησης για το αν χρήστες βιντεοπαιχνιδιών αισθάνονται στρες ή όχι με τα εξής δεδομένα:

- Κλάσεις ω_1: χωρίς στρες και ω_2: με στρες
- Prior pdf της παραμέτρου θ:
  - $p(\theta) = \frac{1}{10\pi} \frac{1}{1+\left(\frac{\theta}{10}\right)^2}$
- Prior πιθανότητες $p(\omega_1)$, $p(\omega_2)$ και Datasets D_1 , D_2 όμοια με το μέρος Α
- Συνάρτηση διάκρισης: $h(x) = logP(x|D_1) - logP(x|D_2) + logP(\omega_1) - logP(\omega_2)$

2.2 Επίλυση του Προβλήματος

Το πρώτο ζητούμενο του προβλήματος είναι η απεικόνιση των posterior pdf των δεδομένων μας $p(\theta|D_1), p(\theta|D_2)$.

- Εφόσον σε αυτό το μέρος μας δίνεται η Prior pdf των δεδομένων μας και επίσης από το μέρος Α γνωρίζουμε και την $p(x|\theta)$ μπορούμε να προχωρήσουμε στα παρακάτω βήματα:
  - Υπολογισμός της συνάρτησης πιθανοφάνειας με τον ίδιο τύπο που αναφέρθηκε στο μέρος Α
  - Υπολογισμός της posterior pdf σύμφωνα με τον τύπο $p(\theta|\mathrm{D}) = \frac{p(D|\theta)\mathrm{p}(\theta)}{\int p(D|\theta)\mathrm{p}(\theta)d\theta}$
  - Γράφοντας τον κατάλληλο κώδικα παράγουμε την prior pdf και τις 2 posterior στο γράφημα της επόμενης διαφάνειας:



Παρατηρήσεις

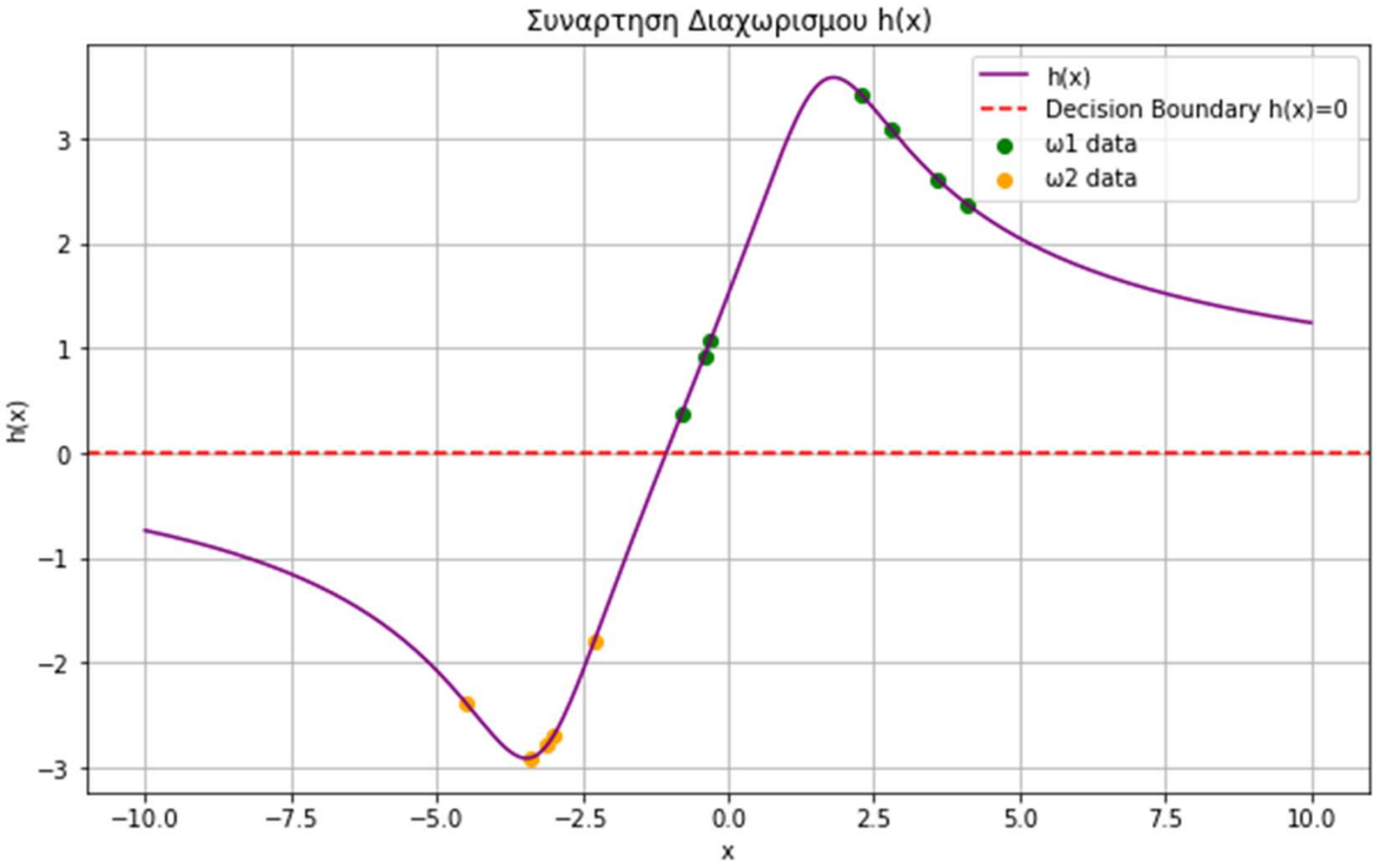
- Οι 2 Posterior συναρτήσεις πυκνότητας πιθανότητας είναι πιο συγκεντρωμένες γύρω από τις τιμές της παραμέτρου θ που εξηγούν τα εκάστοτε δεδομένα
- Η prior κατανομή της παραμέτρου θ έχει πολύ μεγαλύτερη διασπορά, είναι πιο «απλωμένη» καθώς δεν γνωρίζει τίποτα για τα δεδομένα που θέλουμε να εκτιμηθούν.

2.2 Επίλυση του Προβλήματος (συν.)

Το ακόλουθο ζητούμενο του Μέρους Β ήταν η υλοποίηση μιας συνάρτησης πρόβλεψης η οποία θα ταξινομεί τα δεδομένα χρησιμοποιώντας ως συνάρτηση διάκρισης την h(x) που παρουσιάστηκε στην διαφάνεια 9.

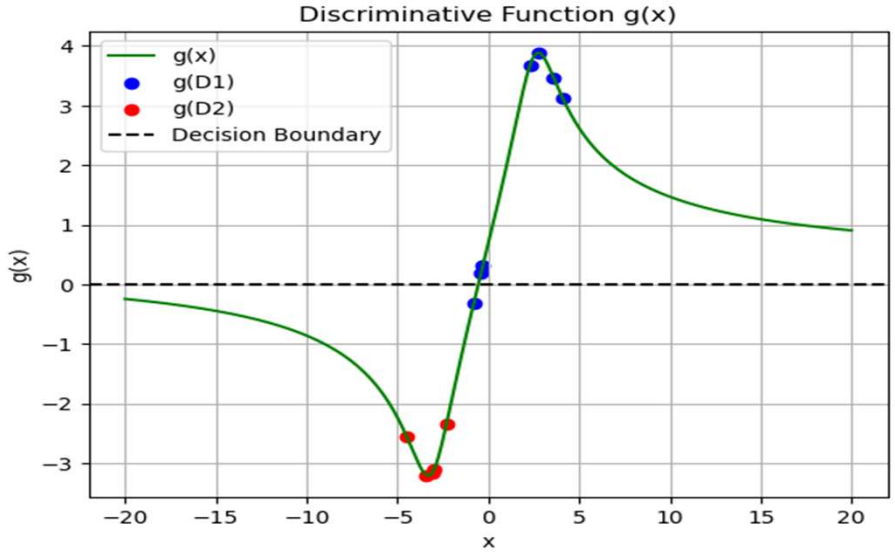
Για κάθε δείγμα x που θέλουμε να ταξινομήσουμε ακολουθούμε την εξής διαδικασία:

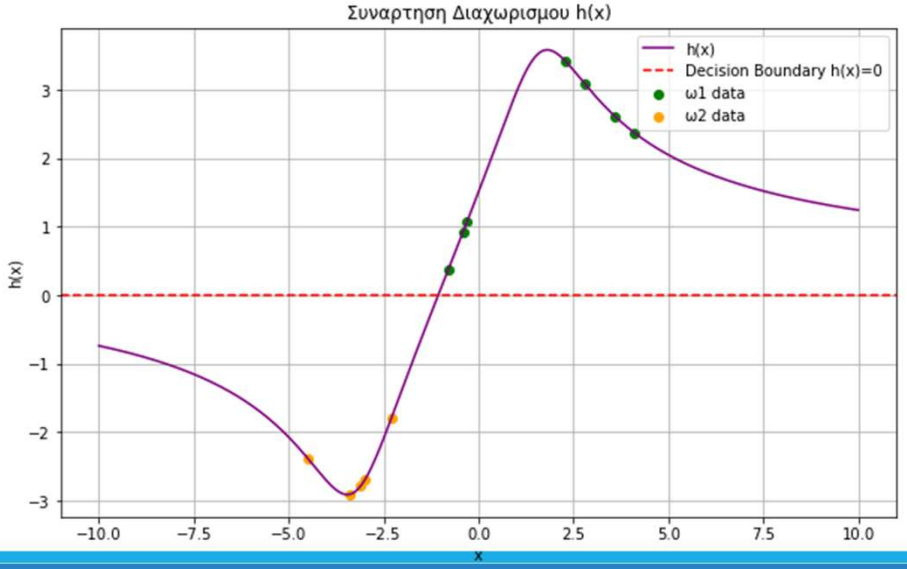
- Υπολογίζουμε την $p(x|D) = \int p(x|\theta)p(\theta|D)d\theta$
- Υπολογίζουμε την τιμή της Discriminant function h(x) για το συγκεκριμένο x
- Σύμφωνα με τον γενικό κανόνα Bayes GBR ισχύει αυτή η σχέση:
  - $logp(x|\omega_1) - logp(x|\omega_2) + LogP(\omega_1) - LogP(\omega_2) > 0$
- Στη συγκεκριμένη περίπτωση δεν γνωρίζουμε την p(x|ω) ωστόσο αντικαθιστώντας όπου ω το D παίρνουμε μια πολύ καλή εκτίμησή της και καταλήγουμε τροποποιώντας την παραπάνω σχέση στο ακόλουθο:
  - h(x)>0 και αυτό είναι το κατώφλι που καθορίζει την απόφαση του ταξινομητή. Πιο συγκεκριμένα
    - $h(x) > 0 \rightarrow x\epsilon\, \omega_1$
    - $h(x) > 0 \rightarrow x\epsilon\, \omega_2$



Υλοποίηση μοντέλου

Αφού υλοποιήσαμε όσα αναφέρθηκαν στην προηγούμενη διαφάνεια, χρησιμοποιήσαμε τον ταξινομητή για να προβλέψουμε σε ποια κλάση ανήκουν τα 12 δείγματά μας. Όπως γίνεται αντιληπτό από την διπλανή εικόνα ο ταξινομητής έκανε σωστά τη δουλειά του διαχωρίζοντας με 100% επιτυχία τα 2 datasets χωρίς εξαιρέσεις.





Σύγκριση MLE – Bayesian Estimator

- Η μέθοδος κατά Bayes είναι σημαντικά πιο περίπλοκη αλγοριθμικά από την μέθοδο μέγιστης πιθανοφάνειας. Εδώ δεν γίνεται αντιληπτό αλλά σε μεγαλύτερα προβλήματα ταξινόμησης (όσον αφορά το κομμάτι το διαστάσεων) η διαφορά θα γινόταν εύκολα αντιληπτή.
- Η γνώση της Prior pdf στην μέθοδο Bayes αποτελεί διευκόλυνση στην δημιουργία ταξινομητή. Ωστόσο στα Real-Life προβλήματα ταξινόμησης σχεδόν ποτέ δεν γνωρίζουμε την prior pdf παρά μόνον την εκτιμούμε.
- Παρά το μικρό πλήθος των δεδομένων η διαφορά στην ακρίβεια των 2 αλγορίθμων γίνεται και εδώ αντιληπτή καθώς στη μέθοδο μέγιστης πιθανοφάνειας ένα δείγμα γίνεται misclassified, πράγμα που δεν συμβαίνει στον Bayesian Estimator.
- Βέβαια δεν ενδείκνυται να ελέγχουμε την απόδοση ενός ταξινομητή χρησιμοποιώντας το training set ως test set όμως στο συγκεκριμένο παράδειγμα δεν υπάρχει κίνδυνος Overfitting και ακόμα εκτιμούμε παραμέτρους για τις οποίες υπάρχουν (ενδεχομένως) αρκετά δείγματα

Μέρος Γ: Ταξινόμηση με δέντρα απόφασης και τη μέθοδο Random Forest

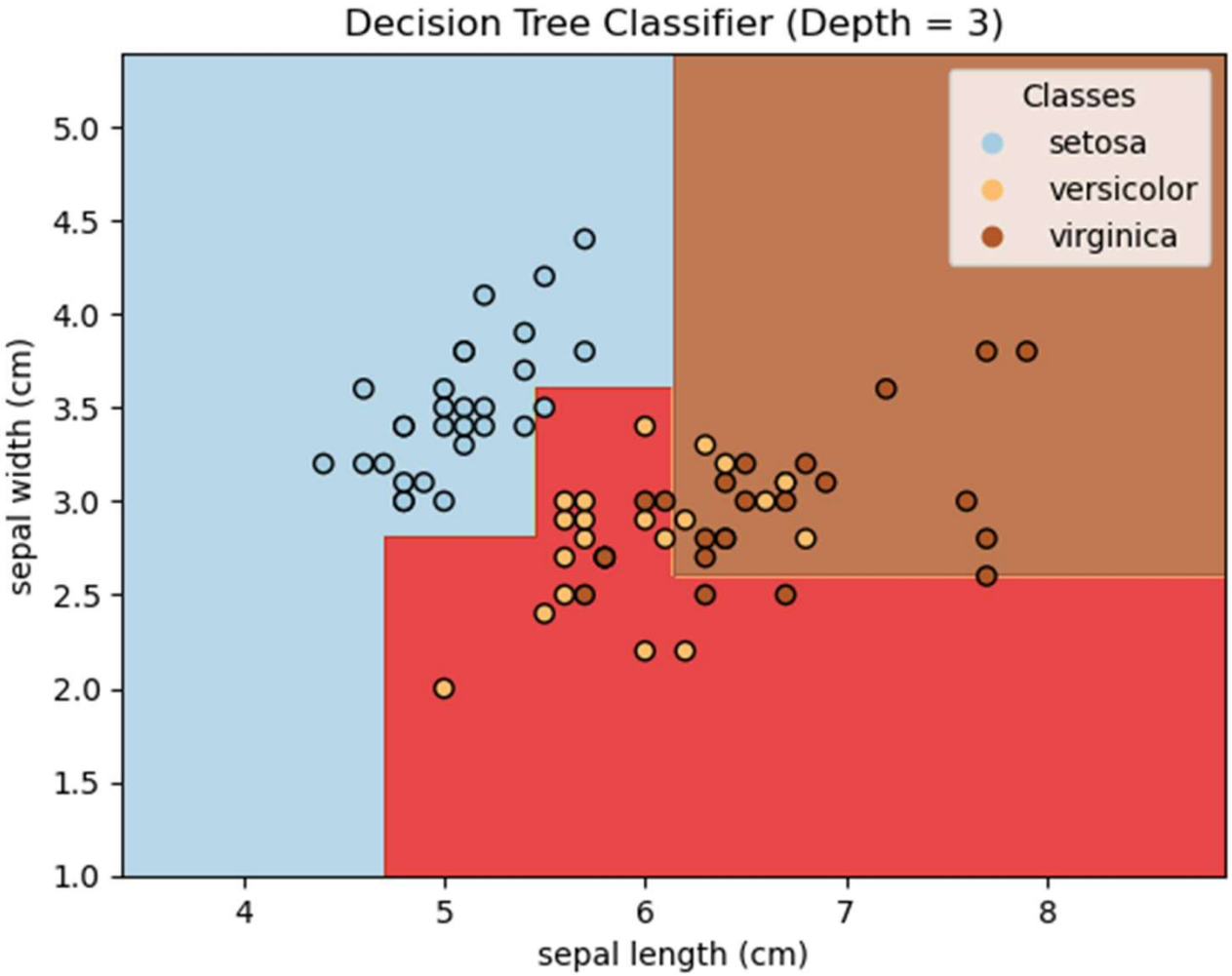
3.1.Περιγραφή Προβλήματος (1η ενότητα)

- Στα πλαίσια μιας έρευνας καλούμαστε να αναγνωρίσουμε τρία διαφορετικά είδη του φυτού Ίριδα. Τα είδη αυτά ξεχωρίζουν μεταξύ τους μέσω κάποιων μετρικών χαρακτηριστικών τους όπως μήκος και πλάτος σεπάλων και πετάλων. Χρησιμοποιώντας την sklearn βιβλιοθήκη κατεβάζουμε ένα dataset με 150 δείγματα Ίριδας (50 από κάθε είδος) στο οποίο περιέχονται στοιχεία για τις προαναφερθείσες μετρικές.
- Στη συνέχεια κάνουμε split το dataset και χρησιμοποιούμε το 50% για την εκπαίδευση του μοντέλου και το άλλο 50% για τον έλεγχο καλής λειτουργίας του μοντέλου. Η διάκριση των ειδών του φυτού, βάσει εκφώνησης γίνεται με βάση το μήκος και το πλάτος των σεπάλων.
- Στην 1η ενότητα καλούμαστε να υλοποιήσουμε ένα δέντρο απόφασης για την ταξινόμηση των ειδών Ίριδας που το dataset περιλαμβάνει.

3.2.Επίλυση Προβλήματος

- Στην πρώτη ενότητα του Μέρους Γ καλούμαστε να υλοποιήσουμε ένα Δέντρο Απόφασης για να ταξινομήσουμε τα δείγματα που έχουμε στο set μας. Χρησιμοποιώντας τον έτοιμο αλγόριθμο DecisionTreeClassifier για ταξινόμηση αρχίζουμε τις δοκιμές για να βρούμε το βέλτιστο βάθος δέντρου και την υψηλότερη δυνατή ακρίβεια ταξινόμησης.

- Αυτό το επιτυγχάνουμε μέσα από βρόχους επανάληψης όπου δοκιμάζουμε μοντέλα με βάθος από 1 έως και 11 και στη συνέχεια ζητάμε να τυπωθεί στην οθόνη το μέγιστο accuracy score και το βέλτιστο βάθους δέντρου που το συνοδεύει. Εκτελώντας τον κώδικα παίρνουμε ως οπτικοποίηση του ταξινομητή το διάγραμμα της επόμενης διαφάνειας στην οποία διατυπώνονται και οι παρατηρήσεις μας.



3.2 Επίλυση Προβλήματος (συν.)

Αφού εκτελεστεί ο κώδικας στην οθόνη τυπώνεται το διπλανό γράφημα και η πρόταση: Καλύτερο βάθος δέντρου 3 με ακρίβεια 0.79 (ακρίβεια 2 δεκαδικών ψηφίων).

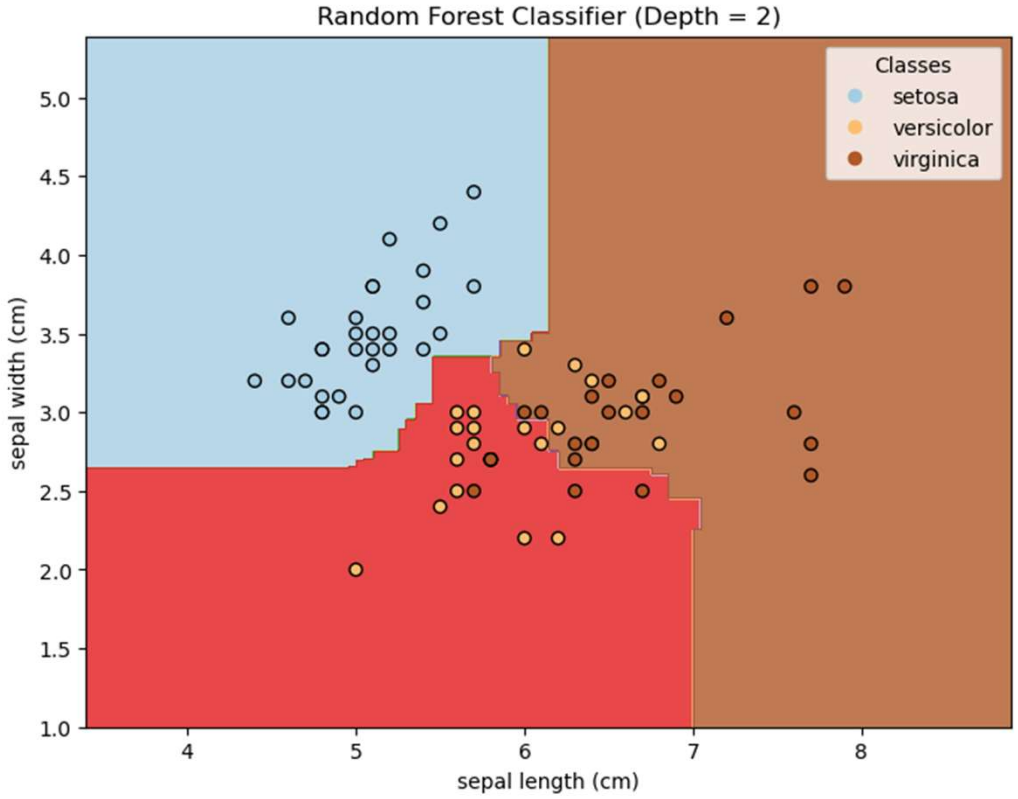
Παρατηρήσεις:

- Καλή προσπάθεια για διαχωρισμό με γραμμικά όρια απόφασης ωστόσο το αποτέλεσμα δεν είναι αρκετά ικανοποιητικό.
- Από το γράφημα γίνεται αντιληπτό πως ο διαχωρισμός με βάση μόνο το μήκος και το πλάτος των σεπάλων είναι δύσκολος ως διαδικασία καθώς στα δείγματά μας υπάρχει επικάλυψη.
- Σε μοντέλα υψηλότερων διαστάσεων όπου η ταξινόμηση θα γινόταν με βάση όλα τα features του dataset μπορεί η επικάλυψη να είχε αποφευχθεί.

3.3.Περιγραφή Προβλήματος (2η ενότητα)

- Στην 2η ενότητα του Μέρους Γ χρησιμοποιούμε ταξινομητή που εφαρμόζει την μέθοδος του Τυχαίου Δάσους.
- Με χρήση της μεθόδου Bootstrap στο training set που χρησιμοποιήσαμε και στην 1η ενότητα (δηλ. στο 50% του Iris Dataset) παράγουμε 100 νέα σύνολα εκπαίδευσης με μέγεθος ίσο με το μισό του συνόλου δεδομένων μας. Για να αξιολογήσουμε την απόδοση της μεθόδου αυτής πάλι βασιζόμαστε στο grid ταξινόμησης που προκύπτει και το accuracy score. Όλα τα δέντρα που απαρτίζουν το δάσος μας έχουν το ίδιο μέγιστο βάθος.
- Για την επίλυση χρησιμοποιήθηκε η μέθοδος RandomForestClassifier της βιβλιοθήκης sklearn Τα αποτελέσματα και οι παρατηρήσεις παρουσιάζονται στην επόμενη διαφάνεια.

3.4 Επίλυση Προβλήματος (2η ενότητα)



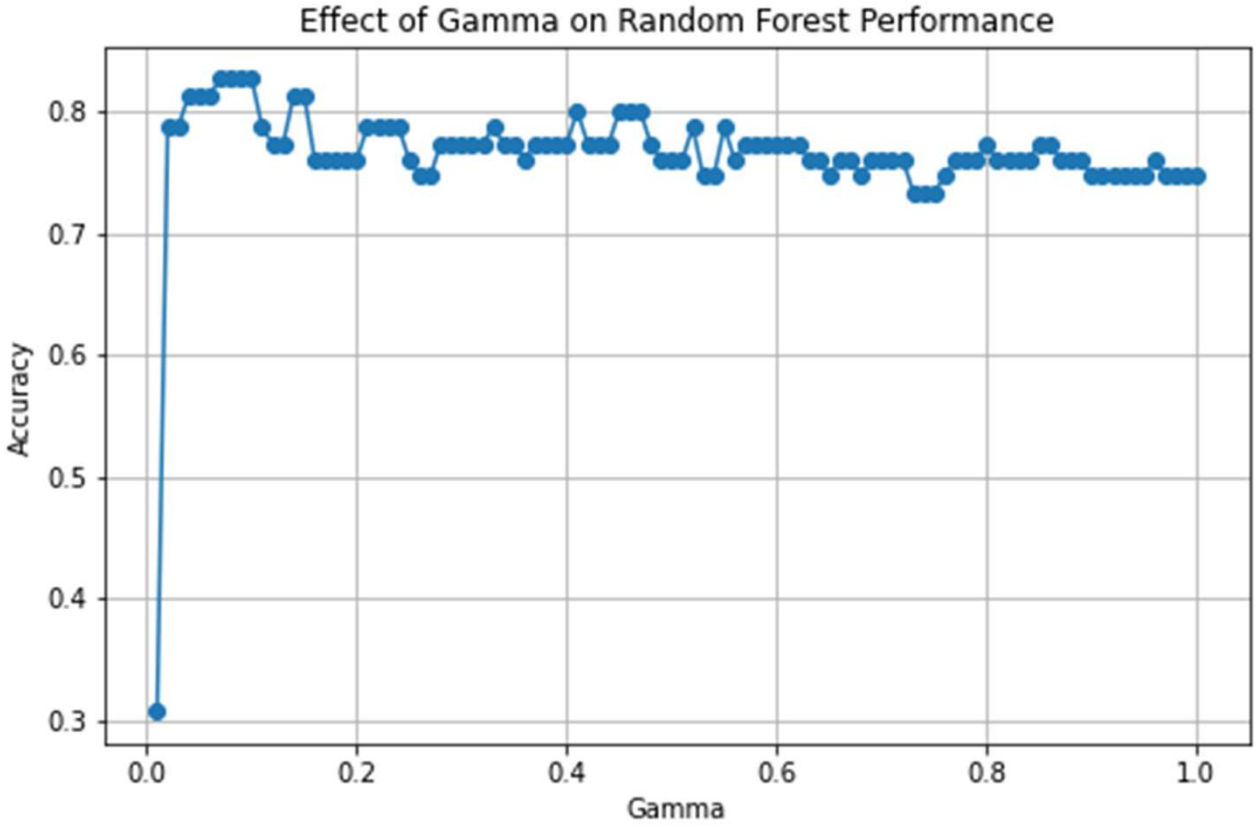
Αφού εκτελεστεί ο κώδικας στην οθόνη τυπώνεται το διπλανό γράφημα και η πρόταση: Καλύτερο βάθος δέντρου 2 με ακρίβεια 0.83 (ακρίβεια 2 δεκαδικών ψηφίων).

Παρατηρήσεις:

- Στην περίπτωση αυτή έχουμε πιο προσαρμοσμένα όρια απόφασης γεγονός που εξηγεί και το ελαφρώς βελτιωμένο accuracy score που λάβαμε.
- Τα πιο εξειδικευμένα όρια απόφασης που παρουσιάζονται εδώ ίσως να ελλοχεύουν και μια τάση για overfitting στα δεδομένα μας.
- Το αποτέλεσμα αυτό δεν μπορεί να ερμηνευθεί ως καθολικά αποδεκτό καθώς η μέθοδος Bootstrap βασίζεται καθαρά στην τυχαιότητα και για αυτό αν δεν βάλουμε ένα σταθερό random state παράγοντα (στη δική μας περίπτωση random state=42), το πρόγραμμά μας δεν θα έχει reproducibility που σημαίνει ότι σε κάθε εκτέλεση το αποτέλεσμα θα διαφέρει. Διαφορετική τιμή στο random state μπορεί να επιφέρει και διαφορετικό βέλτιστο βάθος και ακρίβεια στον αλγόριθμό μας.

3.4 Επίλυση Προβλήματος (2η ενότητα συν.)

Στο τελευταίο ερώτημα της 2ης ενότητας κληθήκαμε να πειραματιστούμε δίνοντας διάφορες τιμές στον συντελεστή γ ο οποίος κυμαίνεται πάντα στο διάστημα (0,1). Ο συντελεστής αυτός καθορίζει για το σύνολο εκπαίδευσης Α (δηλ. το μισό του αρχικού μας dataset) ποιο ποσοστό του θα χρησιμοποιηθεί για τη δημιουργία των καινούργιων συνόλων εκπαίδευσης για τη μέθοδο του τυχαίου δάσους. Στα προηγούμενα ερωτήματα το γ το παίρναμε σταθερά ίσο με 0.5 δηλαδή το 50% του συνόλου Α χρησιμοποιείται για την δημιουργία και εκπαίδευση των 100 νέων δέντρων απόφασης. Στον κώδικα μας, πειραματιστήκαμε δίνοντας τιμές στο γ σχεδόν σε όλο το εύρος του διαστήματος 0 έως 1 και δημιουργήσαμε ένα διάγραμμα που δείχνει τη μεταβολή του accuracy score συναρτήσει της τιμής του γ. Ακολουθούν τα αποτελέσματα και οι παρατηρήσεις.



Παρατηρήσεις:

- Για πολύ μικρές τιμές του γ της τάξεως γ<0.05 το ποσοστό σωστής ταξινόμησης πέφτει σε τιμές που καθιστούν το μοντέλο μας απαγορευτικό προς χρήση (ποσοστά ευστοχίας κοντά στο 30%).
- Όσο περισσότερο αυξάνεται η τιμή του γ τόσο καλύτερα αποτελέσματα έχουμε μέχρι περίπου την τιμή γ=0.15.
- Από την τιμή 0.15 και έπειτα όσο αυξάνεται η τιμή του συντελεστή παρατηρούνται ταλαντώσεις στο ποσοστό σωστής ταξινόμησης ενώ από γ=0.8 και έπειτα υπάρχει μια σχετικά σταθερή πορεία στο διάγραμμα.
- Φυσικά, για υπερβολικά μεγάλες τιμές του γ, όσο περισσότερο προσεγγίζουμε την τιμή της μονάδας τόσο αυξάνεται ο κίνδυνος του overfitting.

Μέρος Δ: Δημιουργία αλγορίθμου ταξινόμησης πραγματικών δεδομένων

4.1. Περιγραφή προβλήματος

- Το παρόν πρόβλημα αφορά την ταξινόμηση πραγματικών δεδομένων σε 5 κλάσεις.
- Δίνονται:
  - Το training set που αποτελείται από 8743 δείγματα και 224 χαρακτηριστικά(features).Επίσης κάθε δείγμα αντιστοιχεί σε μία ετικέτα(label) που αποτελεί την κλάση του(1 έως 5)
  - Το test set που αποτελείται από 6955 δείγματα χωρίς αυτά να έχουν ετικέτα
- Ζητούμενο είναι η δημιουργία ενός ταξινομητή που θα εκπαιδευτεί στο training set και η παραγωγή ενός διανύσματος με ετικέτες όσες και το test set
- Οι ετικέτες αυτές αποτελούν προβλέψεις του ταξινομητή
- Ο βέλτιστος ταξινομητής είναι εκείνος που θα πετύχει το ελάχιστο σφάλμα ταξινόμησης στο test set.
- Ακολουθεί η διαδικασία επίλυσης του προβλήματος και επιλογής του ταξινομητή.

4.2. Προεπεξεργασία των δεδομένων

- Πρώτο βήμα είναι η προεπεξεργασία των δεδομένων του training set.
- Για αρχή, το μόνο που θα εφαρμόσουμε είναι ο διαχωρισμός των δεδομένων σε δύο subsets: training set και validation set.
- Όπως θα δούμε στη συνέχεια, μπορούμε να πειραματιστούμε με τα δεδομένα και να εφαρμόσουμε διάφορες τεχνικές, όπως μείωση διάστασης με PCA, κανονικοποίηση κλπ.
- Στα πλαίσια του πρώτου βήματος, εφαρμόστηκε κανονικοποίηση με την StandardScaler(), ωστόσο έδωσε χειρότερα αποτελέσματα γι' αυτό και αφαιρέθηκε.

4.3.1. Ταξινομητές

Για το παρόν πρόβλημα εφαρμόστηκαν οι παρακάτω ταξινομητές:

- Naïve Bayes(default επιλογή)
- Νευρωνικά Δίκτυα (MLP:2 hidden layers με 128 νευρώνες το πρώτο και 64 το δεύτερο)
- Logistic Regression
- K-NN(5 nearest neighbors)
- SVM (Με kernel function την πολυωνυμική)
- Random Forest

Σημείωση: Γενικά, οι παράμετροι που χρησιμοποιήθηκαν, επιλέχθηκαν τυχαία, ώστε να έχουμε μία πρώτη εικόνα για την απόδοση των ταξινομητών. Εξαίρεση το SVM που επιλέξαμε επί σκοπού ως kernel την πολυωνυμική(αντί της γραμμικής π.χ) λόγω της πολυπλοκότητας των δεδομένων.

Naïve Bayes

- Απλή αλλά ισχυρή μέθοδος ταξινόμησης.
- Βασίζεται στο Θεώρημα Bayes.
- Λειτουργεί κάτω από τις εξής υποθέσεις.
  - Όλα τα χαρακτηριστικά είναι ανεξάρτητα μεταξύ τους (Naive Assumption)
  - Η επίδραση κάθε χαρακτηριστικού στο σετ είναι ισοδύναμη.
- **Πλεονεκτήματα:** Γρήγορη εκπαίδευση και πρόβλεψη, ανθεκτικότητα στον θόρυβο, αποτελεσματική σε μεγάλα δεδομένα.
- **Μειονεκτήματα:** Συχνά η υπόθεση της ανεξαρτησίας δεν ισχύει στην πράξη, δεν υπολογίζει συσχετίσεις μεταξύ features, οπότε στα δοθέντα δεδομένα ίσως να μην είναι τόσο αποδοτικός, καθώς πιθανότατα δεν είναι ανεξάρτητα.

Νευρωνικά Δίκτυα (MLP)

- Τύπος τεχνητού νευρωνικού δικτύου. Αποτελείται από πολλαπλά επίπεδα νευρώνων, ικανό να μαθαίνει πολύπλοκες μη γραμμικές συναρτήσεις.
- Αποτελείται από την είσοδο, τα κρυφά επίπεδα και την έξοδο.
- Χρησιμοποιεί backpropagation και ρυθμίζει τα βάρη βάσει του σφάλματος εξόδου και εφαρμόζει μεθόδους βελτιστοποίησης.
- **Πλεονεκτήματα:** Μπορεί να μάθει πολύπλοκες σχέσεις μεταξύ δεδομένων και να υποστηρίξει μη γραμμικούς διαχωρισμούς.
- **Μειονεκτήματα:** Απαιτεί: μεγάλο όγκο δεδομένων για αποδοτική εκπαίδευση, υψηλές υπολογιστικές απαιτήσεις. Ευάλωτος στο overfitting.

Logistic Regression

- Μοντέλο ταξινόμησης που προβλέπει την πιθανότητα να ανήκει ένα γνώρισμα σε μια κατηγορία χρησιμοποιώντας την σιγμοειδή συνάρτηση. Εκπαιδεύεται ελαχιστοποιώντας την συνάρτηση απώλειας log-loss μέσω της μεθόδου βελτιστοποίησης Gradient-Descent.
- **Πλεονεκτήματα:** Απλή , ερμηνεύσιμη, γρήγορη και αποδοτική για binary classification problems.
- **Μειονεκτήματα:** Ακατάλληλη για μη γραμμικούς διαχωρισμούς χωρίς μετασχηματισμό χαρακτηριστικών.

K-NN

- Μη παραμετρική μέθοδος ταξινόμησης που βασίζεται στο Κ-κοντινότερους γείτονες ενός δείγματος στον χώρο χαρακτηριστικών.
- Υπολογίζει την απόσταση μεταξύ του δείγματος και των άλλων δεδομένων και το ταξινομεί με βάση την κατηγορία στην οποία ανήκει η πλειοψηφία των Κ κοντινότερων γειτόνων
- **Πλεονεκτήματα:** Απλή και εύκολη στην κατανόηση μέθοδος, δεν απαιτείται εκπαίδευση του μοντέλου.
- **Μειονεκτήματα:** Υψηλό κόστος πρόβλεψης για μεγάλο όγκο δεδομένων. Ευαίσθητη μέθοδος όσων αφορά την επιλογή του Κ και την κλίμακα των χαρακτηριστικών.
- Η απόδοση της μεθόδου βελτιώνεται με την επιλογή κατάλληλης απόστασης και με κανονικοποίηση στην προεπεξεργασία των δεδομένων

SVM

- Μέθοδος supervised learning για ταξινόμηση και παλινδρόμηση
- Χρησιμοποιεί hyperplanes για να δημιουργήσει τα όρια απόφασης.
- Επειδή τα πραγματικά δεδομένα τις περισσότερες φορές δεν είναι γραμμικά διαχωρίσιμα, χρησιμοποιεί kernel συναρτήσεις(πολυωνυμική,radius basis κλπ) για να προβάλλει τα δεδομένα σε τέτοιο χώρο ώστε να είναι γραμμικά διαχωρίσιμα.
- Support Vectors ονομάζονται τα vectors που βρίσκονται πάνω στα margins
- Ζητουμενο του ταξινομητή είναι να βρει την κατάλληλη ισορροπία μεταξύ training error και margin(Μεγάλο margin συνεπάγεται περισσότερα misclassified σημεία ενώ μικρό margin σημαίνει μικρότερο error αλλά κίνδυνος overfitting)
- Αρκετά ικανοποιητικός για δεδομένα μεγάλων διαστάσεων

Random Forest

- Συγκεντρωτική (Ensemble) μέθοδος που συνδυάζει πολλαπλά δέντρα απόφασης για ταξινόμηση χρησιμοποιώντας τυχαία υποσύνολα δέντρων και χαρακτηριστικών. Δημιουργεί πολλά δέντρα απόφασης μέσω bootstrap sampling και η τελική πρόβλεψη προκύπτει μέσω πλειοψηφίας ψήφων. Χρησιμοποιεί την τυχαιότητα για αύξηση της γενίκευσης και μείωση της συσχέτισης μεταξύ δέντρων.
- **Πλεονεκτήματα:** Ανθεκτική στο overfitting, αποτελεσματική σε δεδομένα υψηλών διαστάσεων.
- **Μειονεκτήματα:** Υψηλές υπολογιστικές απαιτήσεις, δυσκολία ερμηνείας των αποτελεσμάτων.

4.3.2. Εφαρμογή των ταξινομητών(συν.)

Υπολογίζουμε το accuracy score πρώτα στο validation set που προέκυψε από τον αρχικό διαχωρισμό των δεδομένων και μετά εφαρμόζουμε 3-fold cross validation:

ΧΩΡΙΣ CROSS VALIDATION(%)

- Naïve Bayes: 69.93
- Νευρωνικά Δίκτυα: 83.65
- Logistic Regression: 78.44
- 5-NN: 84.05
- SVM: 86.96
- Random Forest: 81.36

ΜΕ CROSS VALIDATION(%)

- Naïve Bayes: 70.34
- Νευρωνικά Δίκτυα: 80.82
- Logistic Regression: 75.71
- 5-NN: 80.24
- SVM: 82.43
- Random Forest: 80.19

4.3.2. Εφαρμογή των ταξινομητών(συν.)

Παρατηρήσεις

- Με βάση τα παραπάνω, συμπεραίνουμε ότι την καλύτερη επίδοση έχει ο SVM και ακολούθως ο MLP classifier.
- Σημειώνεται ότι και ο 5-NN classifier παρουσιάζει καλή επίδοση ωστόσο στο cross validation η ακρίβειά του πέφτει σε τιμή μικρότερη από την αντίστοιχη του MLP
- Λαμβάνοντας υπόψιν ότι έχουμε ένα σύνθετο πρόβλημα με μεγάλο αριθμό features και samples και το accuracy που πήραμε στο cross validation, οι βέλτιστοι αλγόριθμοι φαίνεται να είναι οι παρακάτω:
  - SVM
  - MLP(Multiple Layers Perceptron)
  - Random Forest

4.4.1.Hyperparameter Tuning στον SVM

Προχωράμε στην επεξεργασία του SVM ταξινομητή. Οι υπερπαράμετροι που θα προσαρμοστούν για τη βελτιστοποίηση του ταξινομητή είναι οι εξής:

- **'C'**: ελέγχει το tradeoff μεταξύ error και margin. Μεγαλύτερο C σημαίνει μικρότερο error αλλά μεγαλύτερο margin, ενώ για μικρότερο C ισχύει το αντίστροφο. Εδώ ελέγχουμε τις τιμές: [0.1, 1, 10, 100].

- **'gamma'**: αφορά το πόσο ομαλά θα είναι τα όρια απόφασης. Μικρό γ οδηγεί σε ομαλά όρια απόφασης που όμως μειώνει την ακρίβεια(underfitting), ενώ μεγάλο γ σημαίνει πιο πολύπλοκα όρια απόφασης(overfitting). Για βέλτιστο αποτέλεσμα, ορίζουμε gamma = 'scale'. Με αυτό τον τρόπο, λαμβάνεται υπόψιν η διασπορά των datapoints και προσαρμόζεται στα δεδομένα του προβλήματος.

- **'kernel'**: προβάλλει τα δεδομένα σε κάποιο feature space , έτσι ώστε να είναι γραμμικά διαχωρίσιμα τα δεδομένα. Εδώ, δοκιμάζουμε τις συναρτήσεις 'poly'(πολυωνυμική) και 'rbf'(radius basis function).

4.4.1.Hyperparameter Tuning στον SVM(συν.)

- Εφαρμόζοντας όλους τους συνδυασμούς των παραπάνω υπερπαραμέτρων με τη συνάρτηση GridSearchCV που χρησιμοποιείται για hyperparameter tuning, η βέλτιστη επιλογή είναι:

C = 10, gamma = ‘scale’, ‘kernel’ = ‘rbf’

- Για αυτές τις υπερπαραμέτρους, η ακρίβεια βελτιώνεται σε: 86.74%

- Σημείωση για τη συνάρτηση kernel: Βάζοντας ως επιλογές και την πολυωνιμική και την radius basis function, ως βέλτιστη ακρίβεια προκύπτει για

C = 10, gamma = ‘scale’, ‘kernel’ = ‘poly’

Accuracy = 86.56% < 86.74%

- Καθώς οι υπόλοιπες υπερπαράμετροι είναι κοινές, πιθανώς τυπώνεται αυτό το αποτέλεσμα λόγω του variability που εισάγει το cross validation και του random state = 42 που εισάγει reproducibility. Για το λόγο αυτό , τρέχουμε τον κώδικα μόνο για kernel = ‘rbf’ ώστε να πάρουμε τη βέλτιστη ακρίβεια.

4.4.2Μείωση Διάστασης(PCA)

- Επόμενο βήμα η μείωση διάστασης όπως αναφέρθηκε και στην αρχή της ενότητας στα πλαίσια της προεπεξεργασίας των δεδομένων.
- Για το πρόβλημα αυτό θα εφαρμόσουμε τη μέθοδο PCA
- PCA: Μέθοδος μείωσης διάστασης η οποία αναγνωρίζει τις κατευθύνσεις(principal components) προς τις οποίες μεγιστοποιείται η διασπορά των δεδομένων.
- Τα δεδομένα προβάλλονται σε νέο σύστημα συντεταγμένων οι οποίες είναι ασυσχέτιστες.
- Έτσι, μειώνεται ο αριθμός των features, αφού κρατάμε εκείνα που συγκρατούν το μεγαλύτερο μέρος της διασποράς.
- Υπάρχει έτοιμη συνάρτηση PCA από τη βιβλιοθήκη sklearn.
- Μετά από δοκιμές για την παράμετρο n_components που αποτελεί τον αρθμό των principal components, πετυχαίνουμε την μεγαλύτερη ακρίβεια για n_components = 50, την τιμή 88.56%.

4.4.2.Μείωση Διάστασης(Kernel PCA)

- Για την υλοποίησή της χρησιμοποιούμε την έτοιμη συνάρτηση kernelPCA από τη βιβλιοθήκη sklearn.
- Διαφορά ως προς την κλασική PCA: Χρησιμοποιεί μία kernel συνάρτηση για να προβάλει τα δεδομένα σε ένα μεγαλύτερης διάστασης χώρο όπου θα είναι γραμμικά διαχωρίσιμα.
- Στη συνέχεια, εφαρμόζει PCA στο νέο χώρο υπολογίζοντας ιδιοτιμές και ιδιοδιανύσματα ως του πίνακα Kernel.
- Μικρή βελτίωση σε σχέση με το αρχικό μοντέλο και επίδοση πολύ κοντά στην κλασική PCA(87.99).
- Συνεπώς, επιλέγουμε να χρησιμοποιήσουμε την κλασική η οποία απαιτεί λιγότερους υπολογισμούς. Προφανώς, είναι και πιο γρήγορη.

4.4.3. Bagging

- Στη συνέχεια, εφαρμόζουμε τη μέθοδο bagging στο παρόν μοντέλο.
- Bagging: Χρησιμοποιεί Bootstrap δειγματοληψία για να δημιουργήσει με επανάθεση νέα dataset από το αρχικό.
- Εκπαιδεύονται διαφορετικά μοντέλα, ένα για κάθε δείγμα bootstrap.
- Συνδυάζει τις προβλέψεις των μοντέλων(π.χ μέσος όρος)
- Συχνότερα χρησιμοποιείται στα δέντρα απόφασης, οπότε ενδέχεται να μην είναι ιδανική μέθοδος για το SVM μοντέλο μας.
- Ωστόσο, βοηθάει στη γενίκευση του μοντέλου και στην αντιμετώπιση του overfitting.
- Χρησιμοποιώντας την έτοιμη συνάρτηση της sklearn BaggingClassifier, παίρνουμε accuracy 88.05%, αρκετά ικανοποιητικό, αλλά μικρότερο από τον SVM με μείωση διάστασης.

4.4.3. Bagging & PCA

- Έχοντας την καλή απόδοση των δύο προηγούμενων αλγορίθμων ατομικά, προχωράμε στο συνδυασμό τους.
- Εφαρμόζουμε πρώτα μείωση διάστασης PCA στα δεδομένα και μετά τον BaggingClassifier για το SVM μοντέλο μας.
- Τελικά, λαμβάνουμε ακρίβεια ίση με 89.02% τη μεγαλύτερη έως τώρα τιμή.
- Προβληματίζει, ωστόσο το γεγονός ότι αν ξανατρέξουμε τον κώδικα εμφανίζει ελαφρώς διαφορετικές τιμές.

4.5. Τελικό Μοντέλο

Με βάση τα παραπάνω και λαμβάνοντας υπόψιν τους εξής παράγοντες:

- Ακρίβεια,
- Δυνατότητα γενίκευσης,

καταλήγουμε στο SVM μοντέλο με μείωση διάστασης PCA(pca components = 50) και παραμέτρους: C = 10, gamma = ‘scale’, ‘kernel’ = ‘rbf’.

- Ολοκληρώνοντας, παίρνουμε με τη συνάρτηση svm_pca_classifier.predict τα labels που αντιστοιχούν στο test set και αποτελούν τις προβλέψεις για την κλάση στην οποία αντιστοιχεί κάθε sample.