Εργασία Αναγνώρησης Προτύπων και Μηχανικής Μάθησης

Βασίλης Αϊτσίδης (10330)
Φίλιππος Ρωσσίδης (10379)
Ομάδα 30

ΑΠΘ

24 Δεκεμβρίου 2024

Πίνακας Περιεχομένων

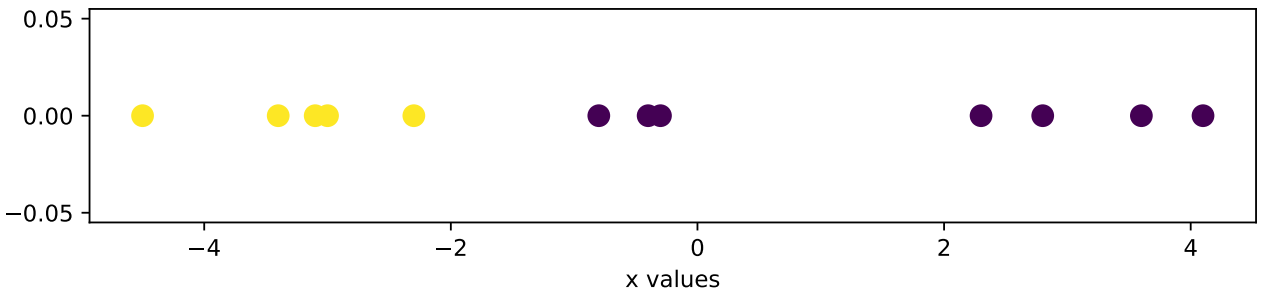
1. ΜΕΡΟΣ Α - Εκτίμηση με Μέγιστη Πιθανοφάνεια
2. ΜΕΡΟΣ Β - Μπεϋζιανή Εκτίμηση
3. ΜΕΡΟΣ Γ - Ίριδα (Δέντρο Απόφασης/ Τυχαίο Δάσος)
4. ΜΕΡΟΣ Δ

ΜΕΡΟΣ Α - Εκτίμηση με Μέγιστη Πιθανοφάνεια

Υποθέτουμε ότι έχουμε δείγματα μιας τιμής x για τα οποία θέλουμε να αναλύσουμε κατά πόσο είναι αξιόπιστοι δείκτες του στρες για παίκτες βιντεοπαιχνιδιών.
Γνωρίζουμε ότι η πυκνότητα πιθανότητας του x είναι:
$p(x|\theta) = \frac{1}{\pi}\frac{1}{1+(x-\theta)^2}$, όπου το θ είναι άγνωστο.
Γνωρίζουμε επίσης ότι από σύνολο 12 δειγμάτων: για 7 παίκτες που δεν ένιωσαν στρες (κλάση ω_1) οι δείκτες x ήταν
$D_1 = [2.8, -0.4, -0.8, 2.3, -0.3, 3.6, 4.1]$, ενώ για τους 5 παίκτες που ένιωσαν στρες (κλάση ω_2) οι δείκτες x ήταν
$D_2 = [-4.5, -3.4, -3.1, -3.0, -2.3]$.
Για να βρούμε εάν είναι αξιόπιστος ο δείκτης x θα προσπαθήσουμε να βρούμε τρόπο να ταξινομούμε κάποιον παίκτη σε κλάσεις: στρες, όχι στρες, με τη χρήση του δείκτη x.
Στο πρώτο μέρος θα χρησιμοποιήσουμε τη μέθοδο Μέγιστης Πιθανοφάνειας.

Εκτίμηση με Μέγιστη Πιθανοφάνεια

Αρχικά μπορούμε να παρατηρήσουμε (και με μια οπτικοποίηση στο σχήμα 1): ότι οι δύο κλάσεις (για τα δείγματα που έχουμε) είναι γραμμικά διαχωρίσημες, άρα ήδη μπορούμε να συμπεράνουμε ότι το x θα έχει σχετικά αξιόπιστα αποτελέσματα.



Σχήμα 1: Οι τιμές των δειγμάτων. Κίτρινο=στρες, μωβ=οχι στρες

Εκτίμηση με Μέγιστη Πιθανοφάνεια

Εκτιμάμε τις παραμέτρους $\hat{\theta}_1, \hat{\theta}_2$ των ΣΠΠ και για τις δύο κλάσεις, δηλαδή τις τιμές που μεγιστοποιούν τις (συναρτήσεις πιθανοφάνειας) $p(D_1|\theta)$ και $p(D_2|\theta)$, αντίστοιχα.
Για διάφορες τιμές του θ υπολογίζουμε τις συναρτήσεις πιθανοφάνειας με τον τύπο:

$$p(D|\theta) = p(x_1, x_2, \ldots, x_N|\theta) = \prod_{n=1}^{N} p(x_n|\theta) \tag{1}$$

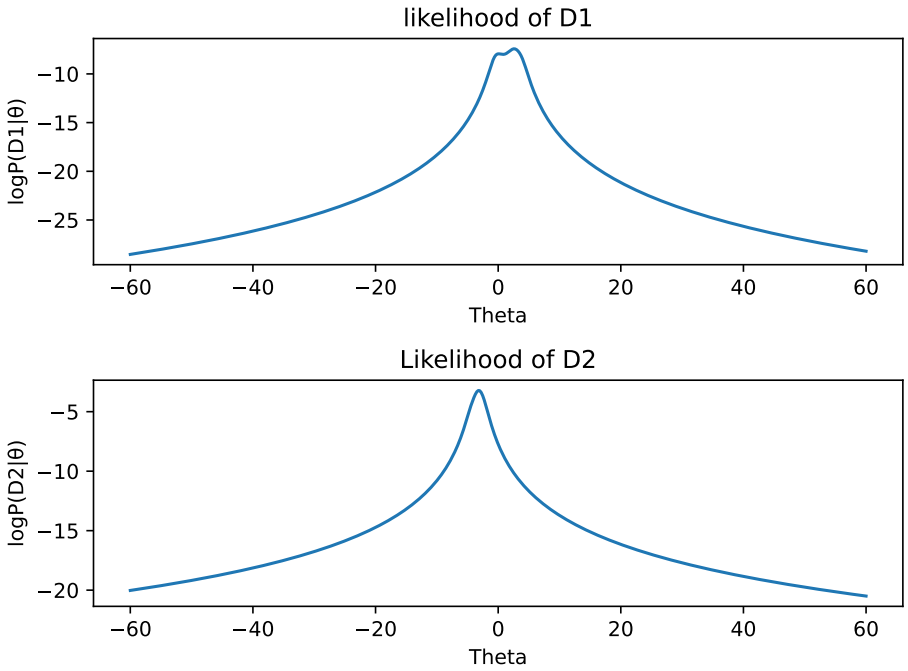
όπου:

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

και $x_i \in D$. Βρίσκουμε την τιμή $\hat{\theta}$ που μεγιστοποιεί το $p(D|\theta)$.
Στην υλοποίηση επιλέχθηκαν 500 τιμές για το θ ομοιόμορφα στο διάστημα $[-60, 60]$.

Εκτίμηση με Μέγιστη Πιθανοφάνεια

Δεξιά παρατίθονται οι συναρτήσεις πιθανοφάνειας $p(D|\theta)$ για τις δύο κλάσεις. Οι τιμές του θ που τις μεγιστοποιούν είναι η εκτίμηση του αλγορίθμου. Συγκεκριμένα για τα δεδομένα που έχουμε, οι τιμές είναι: $\hat{\theta_1} = 2.525, \hat{\theta_2} = -3.246$.
Επίσης, να σημειωθεί ότι όσο περισσότερα είναι τα δείγματα, τόσο πιο στενή θα είναι η καμπύλη $p(D|\theta)$.



Σχήμα 2: Συναρτήσεις πιθανοφάνειας για τις δύο κλάσεις.

Εκτίμηση με Μέγιστη Πιθανοφάνεια

Προκειμένου να ταξινομήσουμε τα x σε κλάσεις χρησιμοποιούμε την συνάρτηση διάκρισης:

$$g(x) = \log P(x|\hat{\theta}_1) - \log P(x|\hat{\theta}_2) + \log P(\omega_1) - \log P(\omega_2)$$

Η συνάρτηση αυτή προκύπτει από τον γενικό κανόνα του Bayes *GBR*:

$$\frac{p(\mathrm{x} \mid \omega_1)}{p(\mathrm{x} \mid \omega_2)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \frac{P(\omega_2)}{P(\omega_1)}$$

με θεωρήσεις:

- $\lambda_{11} = \lambda_{22} = 0$, δηλαδή η ποινή σωστής ταξινόμησης είναι 0.
- $\lambda_{12} = \lambda_{21} = 1$, δηλαδή η ποινή λανθασμένης ταξινόμησης είναι 1.

αν λογαριθμίσουμε και τις δύο πλευρές.

- $g(x) > 0 \Rightarrow P(\omega_1|x) > P(\omega_2|x)$, αρά κατατάσουμε το x στην ω_1.
- $g(x) < 0 \Rightarrow P(\omega_1|x) < P(\omega_2|x)$, αρά κατατάσουμε το x στην ω_2.

Εκτίμηση με Μέγιστη Πιθανοφάνεια

Τα παραπάνω ισχύουν έχοντας υπόψη τις εξής ερμηνείες:

- $P(x|\theta)$: Είναι η συνάρτηση πυκνότητας πιθανότητας των δεδομένων x, δεδομένου ότι η κατανομή τους περιγράφεται από την παραμέτρο θ. Εξαρτάται από την κατανομή που υποθέτουμε ότι ακολουθούν τα δεδομένα x. Στην περίπτωση μας αυτή της έκφρασης 2.
- $P(x|\omega)$: Είναι η πιθανότητα των δεδομένων x, δεδομένου ότι ανήκουν στην κλάση ω. Μπορεί να υπολογιστεί ως:

$$P(x|\omega) = \int P(x|\theta)P(\theta|\omega)d\theta$$

  Όταν η θ είναι σταθερή για κάθε κλάση (εκτιμάται δηλαδή ως $\hat{\theta}$), τότε η $P(x|\omega)$ γίνεται $P(x|\hat{\theta})$.

Εκτιμούμε τις πιθανότητες $P(\omega_1)$, $P(\omega_2)$ ως τον λόγο τον δειγμάτων που ανήκουν στην εκάστοτε κλάση ως προς τα συνολικά δείγματα.

Εκτίμηση με Μέγιστη Πιθανοφάνεια

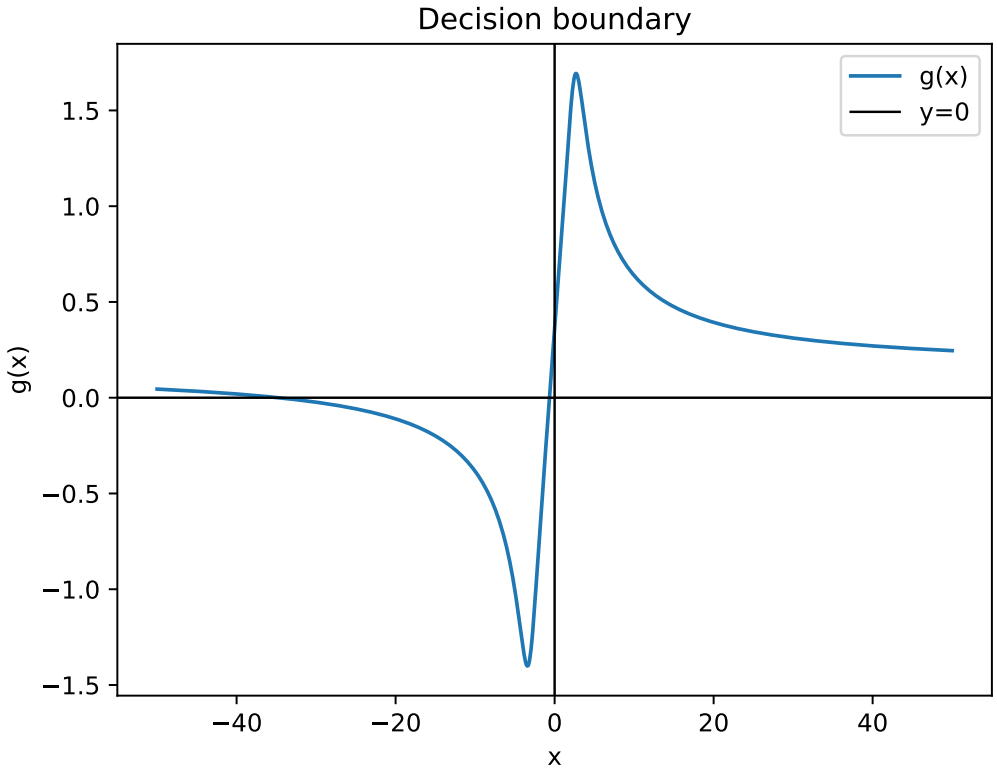
Μετά την υλοποίηση του παραπάνω μοντέλου, το χρησιμοποιήσαμε για να προβλέψουμε σε ποια κλάση θα ανήκουν τα 12 δείγματα μας, δεδομένου του x:

Παρατηρήσεις

- Οι τιμές της $g(x)$ καθορίζουν την κλάση.
- Θετικές τιμές της $g(x)$ αντιστοιχούν στην κλάση ω_1.
- Αρνητικές τιμές της $g(x)$αντιστοιχούν στην κλάση ω_2.
- Με μία εξαίρεση την τιμή $x = -0.8$ όπου η $g(x)$ είναι αρνητική, αλλά το δείγμα ανήκει στην κλάση ω_1.

x	$g(x)$	Σωστή Κλάση
2.8	1.689205	ω_1
-0.4	0.125017	ω_1
-0.8	-0.090887	ω_1
2.3	1.626600	ω_1
-0.3	0.178765	ω_1
3.6	1.492680	ω_1
4.1	1.344624	ω_1
-4.5	-1.145734	ω_2
-3.4	-1.401339	ω_2
-3.1	-1.358416	ω_2
-3.0	-1.326927	ω_2
-2.3	-0.961337	ω_2

Εκτίμηση με Μέγιστη Πιθανοφάνεια



Σχήμα 3: $g(x)$ ως προς x. Φαίνονται οι περιοχές απόφασης του αλγορίθμου. Αν η $g(x)$ είναι θετική αποφασίζουμε ω_1, αλλιώς ω_2. Όπως προαναφέρθηκε, για $x = -0.8$ η απόφαση είναι λανθασμένη.

ΜΕΡΟΣ Β - Μπεϋζιανή Εκτίμηση

Σε αυτό το μέρος χρησιμοποιούμε το ίδιο σύνολο δεδομένων με το πρώτο μέρος, αλλά αυτή τη φορά υλοποιούμε έναν νέο ταξινομητή με τη μέθοδο εκτίμησης κατά Bayes. Η παράμετρος θ παραμένει άγνωστη, όμως τώρα γνωρίζουμε την prior συνάρτηση πυκνότητας πιθανότητας (το οποίο αποτελεί και μια από τις βασικές διαφορές των δύο εκτιμητών):

$$p(\theta) = \frac{1}{10\pi} \frac{1}{1 + (\frac{\theta}{10})^2} \tag{3}$$

Μπεϋζιανή Εκτίμηση

Γνωρίζοντας, λοιπόν, την prior συνάρτηση πυκνότητας πιθανότητας αλλά και την $p(x|\theta)$ από πριν, μπορούμε σύμφωνα με τη θεωρία να εκτιμήσουμε τα εξής:

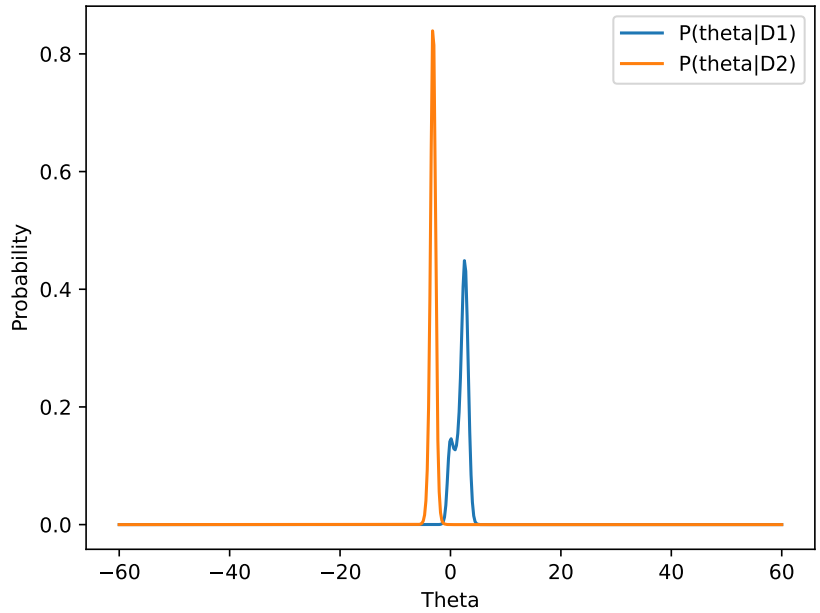
- Υπολογισμός συνάρτησης πιθανοφάνειας $p(D|\theta)$ σύμφωνα πάλι με τον τύπο 1.
- Υπολογισμός a posteriori πιθανότητας σύμφωνα με τον τύπο

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int p(D|\theta)p(\theta)d\theta} \tag{4}$$

όπου τα $p(\theta)$ υπολογίζονται από τον τύπο 3.

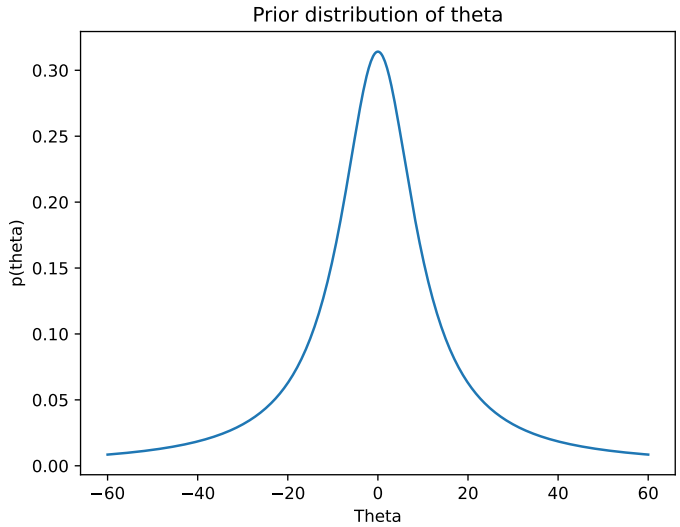
Μπεϋζιανή Εκτίμηση

Δεξιά παρατίθενται οι καμπύλες των εκ των υστέρων πιθανοτήτων $p(\theta|D_1)$ και $p(\theta|D_2)$ για διάφορες τιμές του θ. Όπως και πριν, επιλέχθηκαν 500 τιμές ομοιόμορφα στο διάστημα $[-60, 60]$.

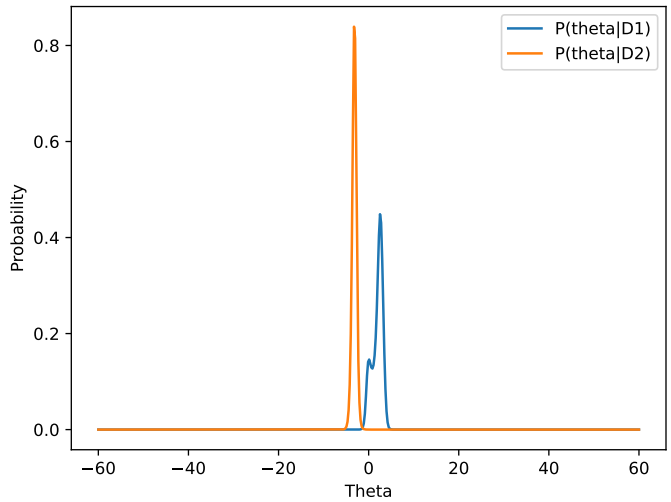


Σχήμα 4: Απεικόνιση των *εκ των υστέρων* πυκνοτήτων πιθανοτήτων για τα δύο σύνολα

Μπεϋζιανή Εκτίμηση



Σχήμα 5: Γράφημα της $p(\theta)$



Σχήμα 6: Posterior κατανομές

Συμπεράσματα:

- Οι $p(\theta|D_1), p(\theta|D_2)$ είναι πιο συγκεντρωμένες γύρω από τις πιθανές τιμές θ που εξηγούν τα δεδομένα.
- Η $p(\theta)$ είναι πιο διάχυτη καθώς δεν έχει πληροφόρηση από τα δεδομένα.

Μπεϋζιανή Εκτίμηση

Τώρα είμαστε σε θέση να υλοποιήσουμε την ταξινόμηση χρησιμοποιόντας τη συνάρτηση διάκρησης:

$$h(x) = \log P(x|D_1) - \log P(x|D_2) + \log P(\omega_1) - logP(\omega_2) \tag{5}$$

Για κάθε δείγμα x το οποίο θέλουμε να ταξινομήσουμε:

- Υπολογίζουμε την $p(x|D)$ ως:

$$p(x|D) = \int p(x|\theta)p(\theta|D)d\theta \tag{6}$$

- Υπολογίζουμε την συνάρτηση διάκρησης $h(x)$ για το x.

Μπεϋζιανή Εκτίμηση

Όπως και στη μέθοδο Μέγιστης Πιθανοφάνειας, έτσι κι εδώ, δίνουμε κάποιες απαραίτητες ερμηνείες:

- $P(x|\omega)$: Αυτή είναι η πιθανότητα να παρατηρηθεί το x, δεδομένου ότι ανήκει στην κλάση ω. Αντιπροσωπεύει την πραγματική (θεωρητική) κατανομή του x υπό την υπόθεση ότι το x παράγεται από την κλάση ω. Χρησιμοποιείται ως συνάρτηση πιθανοφάνειας στο πλαίσιο της Μπεϋζιανής Εκτίμησης.
- $P(x|D)$: Αυτή είναι η πιθανότητα να παρατηρηθεί το x, δεδομένου ότι παρατηρήθηκε το σύνολο δεδομένων D, το οποίο αποτελείται από δείγματα που ανήκουν στην ω. Επειδή το D είναι πεπερασμένο, το $P(x|D)$ αποτελεί προσέγγιση του $P(x|\omega)$. Η ακρίβεια αυτής της προσέγγισης εξαρτάται από το πόσο καλά το D αναπαριστά την πραγματική κατανομή του x για την ω.

Μπεϋζιανή Εκτίμηση

Ο γενικός κανόνας του Bayes *GBR* με τις ίδιες θεωρήσεις όπως και στο πρώτο ερώτημα, μπορεί να γραφτεί:

$$\log p(x|\omega_1) - \log p(x|\omega_2) + \log P(\omega_1) - \log P(\omega_2) > 0 \tag{7}$$

Όμως εφόσον δεν γνωρίζουμε την $p(x|\omega)$, η $p(x|D)$ είναι μια καλή εκτίμηση της. Έτσι η παραπάνω γράφεται:

$$\log p(x|D_1) - \log p(x|D_2) + \log P(\omega_1) - \log P(\omega_2) > 0 \Leftrightarrow \tag{8}$$

$$h(x) > 0$$

Συγκεκριμένα, ισχύει:

- $h(x) > 0 \Rightarrow P(\omega_1|x) > P(\omega_2|x)$, αρά κατατάσουμε το x στην ω_1.
- $h(x) < 0 \Rightarrow P(\omega_1|x) < P(\omega_2|x)$, αρά κατατάσουμε το x στην ω_2.

Μπεϋζιανή Εκτίμηση

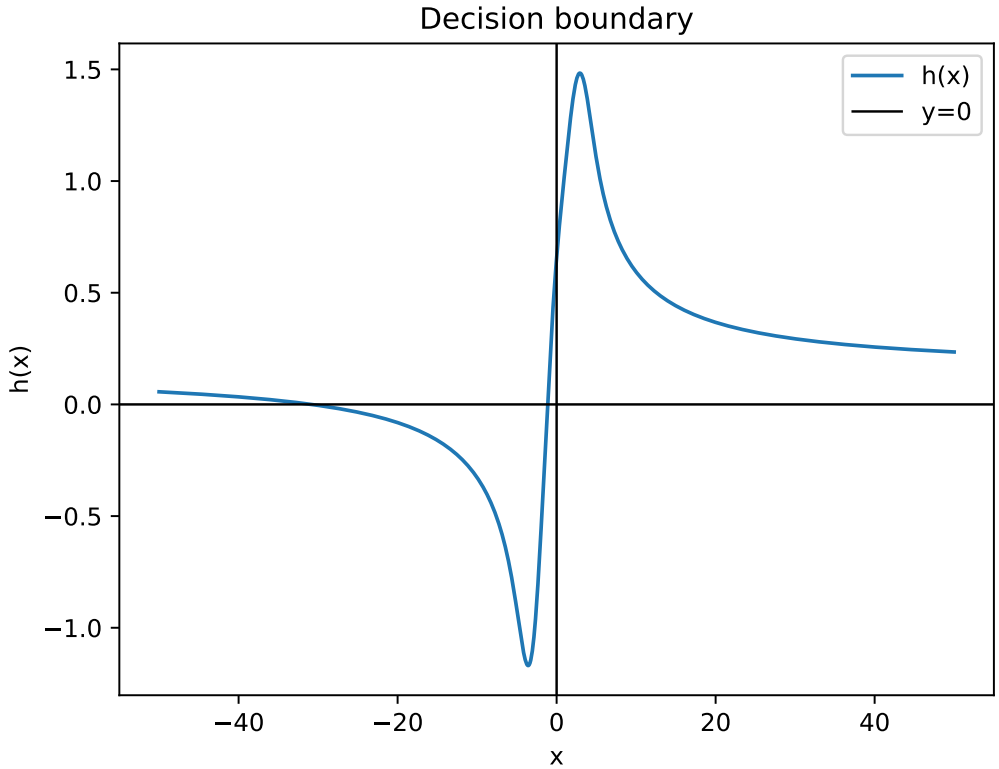
Μετά την υλοποίηση του παραπάνω μοντέλου, το χρησιμοποιήσαμε για να προβλέψουμε σε ποια κλάση θα ανήκουν τα 12 δείγματα μας, δεδομένου του x:

Παρατηρήσεις.

- Οι τιμές της $h(x)$ είναι θετικές για τα δείγματα που ανήκουν στην κλάση ω_1 και αρνητικές σε αυτά που ανήκουν στην ω_2.
- Δεν υπάρχουν εξεραίσεις.

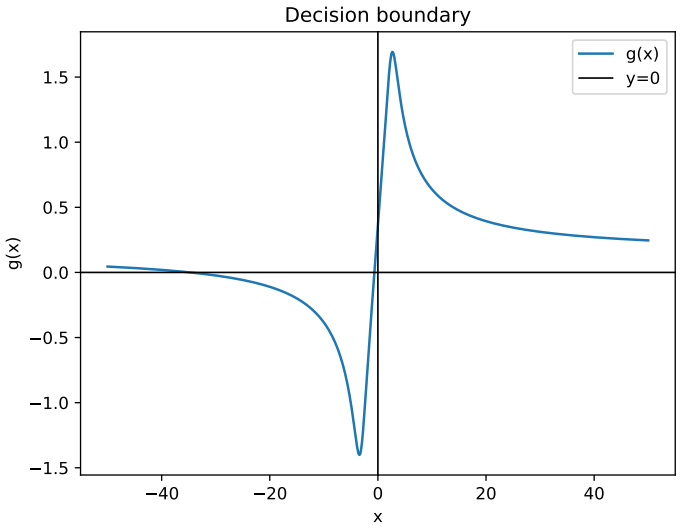
x	$h(x)$	Κλάση
2.8	1.481023	ω_1
-0.4	0.462545	ω_1
-0.8	0.228873	ω_1
2.3	1.422087	ω_1
-0.3	0.514863	ω_1
3.6	1.419364	ω_1
4.1	1.312459	ω_1
-4.5	-1.038287	ω_2
-3.4	-1.162248	ω_2
-3.1	-1.114946	ω_2
-3.0	-1.088929	ω_2
-2.3	-0.775261	ω_2

Μπεϋζιανή Εκτίμηση

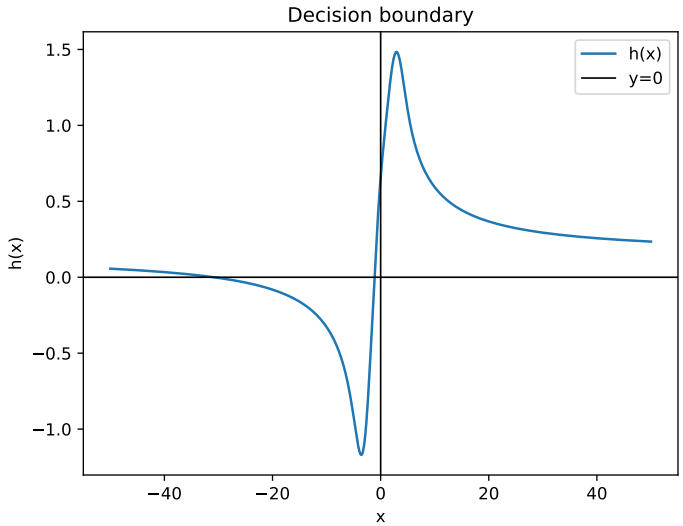


Σχήμα 7: $h(x)$ ως προς x. Φαίνονται οι περιοχές απόφασης του αλγορίθμου. Αν η $h(x)$ είναι θετική αποφασίζουμε ω_1, αλλιώς ω_2.

Σύγκριση Μέγιστης Πιθανοφάνειας και Μπεϋζιανής Εκτίμησης



Σχήμα 8: $g(x)$ ως προς x.



Σχήμα 9: $h(x)$ ως προς x. Φαίνονται οι περιοχές απόφασης του αλγορίθμου. Αν η $h(x)$ είναι θετική αποφασίζουμε ω_1, αλλιώς ω_2.

Οι περιοχές αποφάσεων της μεθόδου μέγιστης πιθανοφάνειας και μπεϋζιανής εκτίμησης είναι σχεδόν ίδιες. Όμως παρατηρούμε μια διαφορά στην περιοχή κοντά του 0. Συγκεκριμένα η μπεϋζιανή εκτίμηση ταξινομεί στην κλάση ω_1 κάποιες πιο αρνητικές τιμές, και έτσι 'πιάνει' και το $x = -0.8$, ενώ η μέγιστης πιθανοφάνειας το ταξινομεί λανθασμένα στην ω_2.

Σύγκριση Μέγιστης Πιθανοφάνειας και Μπεϋζιανής Εκτίμησης

Ο αλγόριθμος της μπεϋζιανής εκτίμησης είναι σημαντικά πιο περίπλοκος, και ενώ στο συγκεκριμένο παράδειγμα δεν υπήρξε τέτοιο ζήτημα, σε παραδείγματα με περισσότερες διαστάσεις και περισσότερα δείγματα η διαφορά θα ήταν ορατή.
Όμως η γνώση της prior συνάρτησης πυκνότητας πιθανότητας $p(\theta)$ δίνει μεγαλύτερη ακρίβεια και καλύτερα αποτελέσματα.
Παρ΄ότι τα δεδομένα του συγκεκριμένου προβλήματος ήταν λίγα, υπήρξε διαφορά στις δύο μεθόδους, όπου η μέθοδος μέγιστης πιθανοφάνειας ταξινόμησε λάθος το δείγμα $x = -0.8$, ενώ η μέθοδος μπεϋζιανής ταξινόμησης τα ταξινόμησε σωστά.
Βέβαια δεν είναι ιδανικό να ελέγχουμε την απόδοση ενός αλγορίθμου στα δείγματα στο οποία εκπαιδεύτηκε, όμως στην προκειμένη περίπτωση, εκτός από το ότι δεν διαθέτουμε επιπλέον, δεν υπάρχει σοβαρός κίνδυνος overfitting μιας και γνωρίζουμε ήδη τη δομή της πυκνότητας πιθανότητας και εκτιμούμε απλώς τις παραμέτρους της, για τις οποίες υπάρχουν (ενδεχομένως) αρκετά δείγματα.

ΜΕΡΟΣ Γ - Ίριδα (Δέντρο Απόφασης/ Τυχαίο Δάσος)

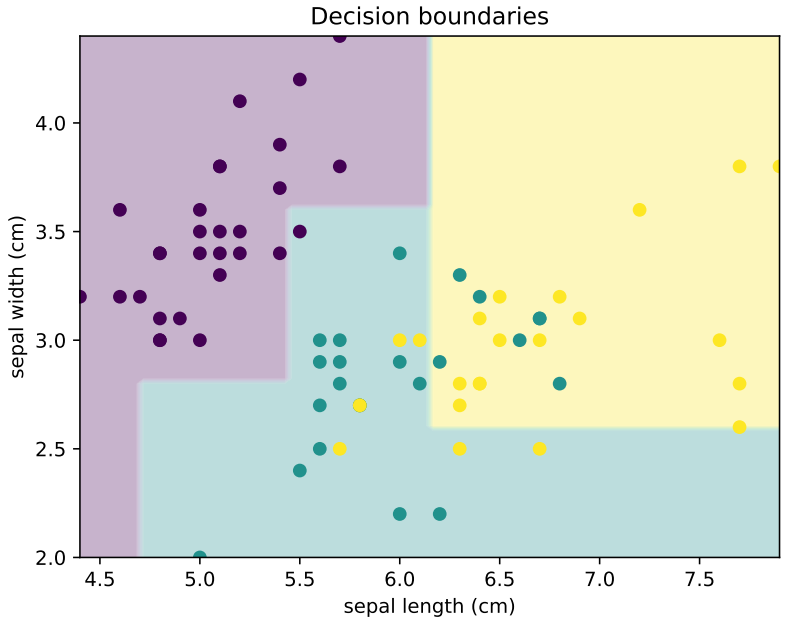
Στα πλαίσια μιας έρευνας, ασχολούμαστε με την αναγνώριση των τριών διαφορετικών ειδών ενός συγκεκριμένου φυτού, της Ίριδας. Κάθε ένα από αυτά τα είδη μπορεί να διαχωριστεί εξετάζοντας τις διάφορες τιμές μήκους και πλάτους των σεπάλων και των πετάλων του άνθους τους. Φορτώνουμε, λοιπόν, μια βάση με 150 δείγματα του φυτού της Ίριδας και συγκεκριμένα 50 για κάθε είδος, από την βιβλιοθήκη sklearn, όπου κάθε δείγμα περιέχει τιμές για τα 4 διαφορετικά χαρακτηριστικά που προαναφέρθηκαν. Ακολουθώντας τη συνήθη τακτική στη μηχανική μάθηση, χωρίζουμε το σύνολο των 150 δειγμάτων στα δύο, με το ένα υποσύνολο να χρησιμοποιείται για την εκπαίδευση του αλγορίθμου και το άλλο (υπόλοιπο μισό) για την αξιολόγηση του ήδη εκπαιδευμένου μοντέλου. Ωστόσο, δεν χρησιμοποιούνται όλα τα χαρακτηρηστικά των δειγμάτων αλλά μόνο τα πρώτα δύο (μήκος και πλάτος σεπάλου).

Ίριδα (Δέντρο Απόφασης/ Τυχαίο Δάσος)

Σε αυτήν την ενότητα χρησιμοποιούμε τον έτοιμο αλγόριθμο ταξινόμησης DecisionTreeClassifier. Προκειμένου να πάρουμε το καλύτερο ποσοστό σωστής ταξινόμησης, θέτουμε ως υπερπαράμετρο το βάθος του δέντρου και αφού δώσουμε διάφορες τιμές (στη συνάρτηση που εκτελεί τη βελτιστοποίηση) στη συνέχεια εφαρμόζουμε τεχνικές hyperparameter tuning μέχρι να βρούμε τη βέλτιστη. Η συνάρτηση επιστρέφει υψηλότερη ακρίβεια ταξινόμησης ίση με 0.7866 για βάθος δένδρου ίσο με 3. Να σημειωθεί ότι στο στάδιο αυτό χρησιμοποιείται η στρατηγική 3-fold cross validation.

Ίριδα (Δέντρο Απόφασης/ Τυχαίο Δάσος)

Όπως παρατηρούμε και στο διπλανό γράφημα, ο ταξινομητής κάνει μια καλή προσπάθεια διαχωρισμού των δειγμάτων, χρησιμοποιώντας γραμμικές περιοχές απόφασης, ωστόσο το αποτέλεσμα δεν είναι αρκετά ικανοποιητικό. Είναι προφανές και από τη διασπορά των δειγμάτων ότι υπάρχει επικάλυψη, γεγονός που οδηγεί σε μέτρια αποτελέσματα. Ωστόσο, να σημειωθεί ότι στην εκπαίδευση του αλγορίθμου λαμβάνονται υπόψη μόνο τα 2 από τα 4 χαρακτηριστικά των δειγμάτων. Ενδεχομένως η επικάλυψη να μπορεί να αποφευχθεί σε υψηλότερες διαστάσεις (χρήση περισσότερων features).



Σχήμα 10: Περιοχές απόφασης του ταξινομητή και σημεία διασποράς που απεικονίζουν τα δείγματα χρωματισμένα σύμφωνα με την κλάση που ανήκουν πραγματικά

Ίριδα (Δέντρο Απόφασης/ Τυχαίο Δάσος)

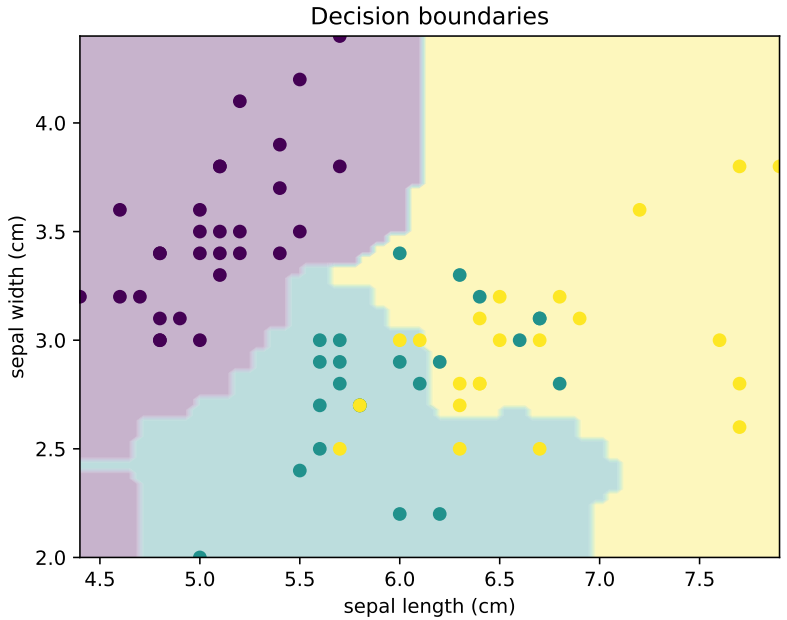
Στη 2η Ενότητα χρησιμοποιούμε ταξινομητή RandomForest.Συγκεκριμένα, εφαρμόζουμε τεχνική Bootstrap στο σύνολο εκπαίδευσης Α που χρησιμοποιήσαμε και στην προηγούμενη ενότητα (που είναι το μισό του αρχικού με τα 150 δείγματα) με αποτέλεσμα να καταλήγουμε με 100 νέα σύνολα που είναι σε μέγεθος μισά του Α. Για την αξιολόγηση του αλγορίθμου χρησιμοποιούμε πάλι το ίδιο με την 1η ενότητα. Τα 100 δέντρα που προκύπτουν έχουν όλα το ίδιο μέγιστο βάθος.

Ίριδα (Δέντρο Απόφασης/ Τυχαίο Δάσος)

Χωρίς βελτιστοποίηση υπερπαραμέτρων, λαμβάνουμε ποσοστό σωστής ταξινόμησης 0.76, τιμή που δεν είναι ιδιαίτερα ικανοποιητική. Βελτιστοποιώντας, ωστόσο, ως προς το μέγιστο βάθος (κάθε δένδρου στο τυχαίο δάσος), πετυχαίνουμε ακρίβεια ίση με 0.8 για βάθος 3. Παρατηρούμε, συνεπώς, ότι το Τυχαίο Δάσος μας έδωσε καλύτερα αποτελέσματα από το Δένδρο Απόφασης, για το ίδιο φυσικά σύνολο δεδομένων.

Ίριδα (Δέντρο Απόφασης/ Τυχαίο Δάσος)

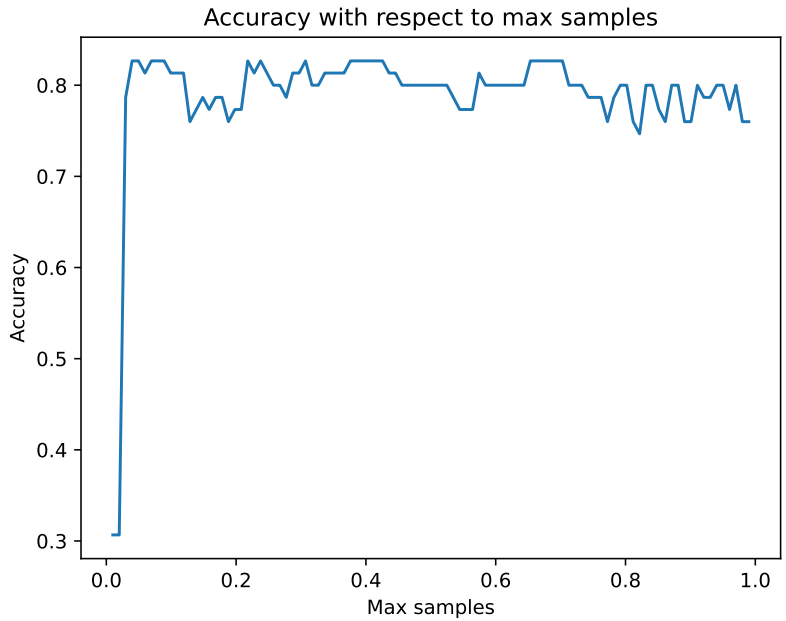
Από το διπλανό γράφημα συμπεραίνουμε ότι στην περίπτωση του Τυχαίου Δάσους, τα όρια απόφασης προσαρμόζουν καλύτερα στα δεδομένα και οι γραμμές είναι πιο "refined", γεγονός που εξηγεί και το υψηλότερο ποσοστό σωστής ταξινόμησης που λάβαμε συγκριτικά με αυτό του Δένδρου Απόφασης. Ωστόσο, ενδεχομένως να υπάρχει και μεγαλύτερος κίνδυνος για overfitting.



Σχήμα 11: Περιοχές απόφασης του ταξινομητή και σημεία διασποράς που απεικονίζουν τα δείγματα χρωματισμένα σύμφωνα με την κλάση που ανήκουν πραγματικά

Ίριδα (Δέντρο Απόφασης/ Τυχαίο Δάσος)

Στο τελευταίο ερώτημα της ενότητας, εξετάζουμε πως μεταβάλλεται η απόδοση του αλγορίθμου συναρτήσει του γ, όπου γ είναι το ποσοστό του συνόλου A που χρησιμοποιείται κάθε φορά για κάθε δέντρο. Τρέξαμε τον αλγόριθμο για 100 τιμές $\gamma \in (0, 1)$. Παραθέτουμε ένα γράφημα της μεταβολής του ποσοστού της ακρίβειας ταξινόμησης συναρτήσει του γ. Παρατηρούμε ότι για πολύ μικρά γ (< 0.1) η ακρίβεια μειώνεται σημαντικά. Όμως για τις υπόλοιπες τιμές, η ακρίβεια αυξομειώνεται χωρίς αναγνωρίσιμο μοτίβο.



Σχήμα 12: Γράφημα ακρίβειας συναρτήσει της τιμής του γ

ΜΕΡΟΣ Δ - Εισαγωγή

Σε αυτό το κομμάτι της εργασίας μας δίνεται ένα training dataset με 8743 δείγματα, 224 features ανά δείγμα και οι ετικέτες για κάθε δείγμα/διάνυσμα χαρακτηριστικών (*featurevector*), συνολικά, ένας πίνακας διαστάσεων 8743 επί 225 με την τελευταία στήλη να είναι οι ετικέτες. Οι τιμές των ετικετών κυμαίνονται από 1,..,5, δηλαδή οι κλάσεις στις οποίες ταξινομούνται τα δείγματα είναι συνολικά 5. Προσπαθούμε να αναπτύξουμε έναν αλγόριθμο ταξινόμησης με όποια μέθοδο κρίνουμε ότι αποδίδει τα καλύτερα αποτελέσματα στον εν λόγω dataset. Προκειμένου, να καταλήξουμε στο βέλτιστο μοντέλο, δοκιμάζουμε διάφορες διαδεδομένες τεχνικές εκπαίδευσης και ταξινόμησης, όπου προφανώς κάποιες εφαρμόζουν πολύ καλύτερα από κάποιες άλλες, δεδομένων των αριθμών των δειγμάτων μας και των αριθμών των διαστάσεων των feature μας. Αφού έχουμε εκπαιδεύσει και επαληθεύσει το μοντέλο μας, στη συνέχεια το εφαρμόζουμε σε ένα νέο test dataset που αποτελείται από 6955 δείγματα και για τα οποία δεν γνωρίζουμε εκ των προτέρων την ετικέτα τους. Τέλος, εξάγουμε το διάνυσμα με τις ετικέτες που προέβλεψε το εκπαιδευμένο μοντέλο μας πάνω στα δείγματα του test dataset.

Εισαγωγή

Η υλοποίηση έγινε με έτοιμες συναρτήσεις της βιβλιοθήκης sklearn. Για κάθε μοντέλο που δοκιμάστηκε, η επιλογή των υπερπαραμέτρων έγινε με τη χρήση της GridSearchCV. Τα μοντέλα εκπαιδεύονται στο 80% των δεδομένων με 3-fold cross validation, και υπολογίζεται η απόδοση τους στο υπόλοιπο 20%. Γίνεται εκπαίδευση για κάθε υπερπαράμετρο στο param_grid που ορίζουμε εμείς, και επιστρέφεται το καλύτερο μοντέλο.
Επειδή οι δοκιμές δεν έγιναν όλες ταυτόχρωνα (διότι αυτό θα επιθυμούσε πολύ χρόνο), αλλά δοκιμάζονταν διάφορες παράμετροι σε διαφορετικές κλήσεις του κώδικα, ο τελικός κώδικας python περιέχει μόνο τις τελικές υπερπαραμέτρους στις οποίες καταλήξαμε.

Εισαγωγή

Τα μοντέλα τα οποία δοκιμάστηκαν είναι τα εξίς:

- Random Forest Classifier
- Gaussian Naive Bayes Classifier
- K-NN Classifier
- Suport Vector Classifier
- MLP Classifier

Random Forest Classifier

- **Εκπαίδευση**: Κάθε δέντρο εκπαιδεύεται σε ένα διαφορετικό υποσύνολο των δεδομένων εκπαίδευσης, που δημιουργείται με τη μέθοδο bootstrapping. Για κάθε κόμβο, η επιλογή του καλύτερου χαρακτηριστικού γίνεται με τυχαία υποσύνολα χαρακτηριστικών.
- **Πρόβλεψη**: Για ένα νέο δείγμα, κάθε δέντρο κάνει μια πρόβλεψη. Ο Random Forest συνδυάζει τις προβλέψεις αυτές μέσω ψηφοφορίας για την ταξινόμηση, όπου και μας ενδιαφέρει.
- **Αντιμετώπιση υπεράρμωσης**: Η τυχαιότητα που εισάγεται τόσο στα δεδομένα όσο και στα χαρακτηριστικά μειώνει την πιθανότητα υπεράρμωσης (overfitting).
- **Πλεονεκτήματα**: Ο Random Forest είναι ιδιαίτερα αποτελεσματικός σε προβλήματα με πολλές μεταβλητές, αλληλεπιδράσεις ή δεδομένα με ελλείψεις.

Ο Random Forest είναι ένα μοντέλο που προσφέρει καλή απόδοση χωρίς να απαιτεί σημαντική παραμετροποίηση και είναι κατάλληλο για πολλές εφαρμογές.

Random Forest Classifier Αποτελέσματα

- Συνάρτηση: *RandomForestClassifier*()
- Υπερπαράμετροι που δοκιμάστηκαν:
  1. $n\_estimators = \{100, 200, 300, 400, 500\}(default = 100)$
  2. $criterion = \{gini, entropy, log\_loss\}(default = gini)$
  3. $max\_depth = \{5, 10, 20, 30, None\}(default = None)$
  4. $max\_features = \{sqrt, log2, None\}(default = sqrt)$
- Accuracy: 0.8147512864493996
- Τελικές υπερπαράμετροι:
  - $n\_estimators = 300$
  - defaults

Gaussian Naive Bayes Classifier

Χρησιμοποιήθηκε η συνάρτηση *GaussianNB*().
Πρόκειται για έτοιμη υλοποίηση του μοντέλου Μπεϋζιανής ταξινόμησης.
Συγκεκριμένα, επειδή στη μέθοδο αυτή πρέπει να γνωρίζουμε εκ των προτέρων την συνάρτηση πυκνότητας πιθανότητας, η ρουτίνα αυτή υποθέτει γκαουσιανή κατανομή.
Περιμένουμε να μην έχει πολύ καλά αποτελέσματα διότι δεν έχουμε κανένα λόγο να πιστεύουμε ότι τα συγκεκριμένα δεδομένα ακολουθούν γκαουσιανή κατανομή. Όμως, εφόσον δεν γνωρίζουμε τίποτα για τη φύση των δεδομένων (τι υποδηλώνουν, από που δειγματοληπτήθηκαν, τη πυκνότητα πιθανότητας τους) δεν μπορούμε να κάνουμε κάποια καλύτερη υπόθεση.

Gaussian Naive Bayes Classifier Αποτελέσματα

- Συνάρτηση: *GaussianNB*()
- Υπερπαράμετροι που δοκιμάστηκαν: Καμία
- Accuracy: 0.6986849628359062

K-NN Classifier

Ο K-Nearest Neighbors (K-NN)είναι ένας απλός αλλά αποτελεσματικός ταξινομητής που βασίζεται στη γειτνίαση των δεδομένων. Η βασική ιδέα του K-NN είναι ότι παρόμοια δείγματα τείνουν να ανήκουν στην ίδια κλάση.

- **Εκπαίδευση**: Ο K-NN δεν απαιτεί εκπαίδευση με την παραδοσιακή έννοια. Αντίθετα, αποθηκεύει ολόκληρο το σύνολο δεδομένων εκπαίδευσης και το χρησιμοποιεί για πρόβλεψη.
- **Πρόβλεψη**: Για ένα νέο δείγμα:
  1. Υπολογίζει την απόσταση του νέου δείγματος από κάθε σημείο του συνόλου εκπαίδευσης (συνήθως χρησιμοποιείται η Ευκλείδεια απόσταση).
  2. Επιλέγει τους k πιο κοντινούς γείτονες.
  3. Η κλάση της πρόβλεψης καθορίζεται από την πλειοψηφία των γειτόνων για την ταξινόμηση.

K-NN Classifier

- **Πλεονεκτήματα**: Δεν απαιτεί εκπαίδευση, και είναι κατάλληλος για δεδομένα με μη γραμμικούς διαχωρισμούς.
- **Μειονεκτήματα**: Η πρόβλεψη μπορεί να είναι αργή για μεγάλα σύνολα δεδομένων, ενώ είναι ευαίσθητος στην κλιμάκωση των χαρακτηριστικών.

Ο K-NN είναι ιδανικός για μικρά και μεσαία σύνολα δεδομένων, ειδικά όταν οι σχέσεις μεταξύ των δειγμάτων είναι σαφείς και μπορούν να αναπαρασταθούν γεωμετρικά.

K-NN Classifier Αποτελέσματα

- Συνάρτηση: $KNeighborsClassifier()$
- Υπερπαράμετροι που δοκιμάστηκαν:
  1. $n\_neighbors = \{5, 10, 14, 16\}(default = 5)$
  2. $weights = \{uniform, distance\}(default = uniform)$
  3. $algorithm = \{auto, ball\_tree, kd\_tree, brute\}(default = auto)$
  4. $p = \{1, 2\}(default = 2)$
- Accuracy: 0.8416237850200115
- Τελικές υπερπαράμετροι:
  - $n\_neighbors = 14$
  - $p = 1$
  - $weights = distance$
  - defaults

Suport Vector Classifier

Ο Support Vector Classifier (SVC) είναι μια μέθοδος μηχανικής μάθησης που ανήκει στην οικογένεια των Support Vector Machines (SVMs)και χρησιμοποιείται για προβλήματα ταξινόμησης. Η βασική ιδέα του SVC είναι να βρει έναν υπερεπίπεδο που διαχωρίζει τις κλάσεις με τον καλύτερο δυνατό τρόπο.

- **Βασική Αρχή:**
  - Ο SVC αναζητά το υπερεπίπεδο που μεγιστοποιεί το περιθώριο (margin), δηλαδή την ελάχιστη απόσταση μεταξύ του υπερεπιπέδου και των πιο κοντινών σημείων κάθε κλάσης (support vectors).
  - Στόχος είναι η εύρεση ενός γραμμικού ή μη γραμμικού διαχωρισμού ανάμεσα στις κλάσεις.
- **Επεκτάσεις για μη γραμμικά προβλήματα:**
  - Χρησιμοποιείται η τεχνική του *kernel trick*, που επιτρέπει τη μετατροπή των δεδομένων σε υψηλότερες διαστάσεις για να γίνουν γραμμικά διαχωρίσιμα.
  - Συχνά χρησιμοποιούμενοι πυρήνες (*kernels*):
    - Γραμμικός (*linear kernel*)
    - Πολυωνυμικός (*polynomial kernel*)
    - Radial Basis Function (RBF)

Support Vector Classifier

- **Πλεονεκτήματα**:
  - Κατάλληλος για μικρά και μεσαία σύνολα δεδομένων.
  - Αποτελεσματικός ακόμα και με υψηλής διάστασης δεδομένα.
- **Μειονεκτήματα**:
  - Μπορεί να είναι αργός για πολύ μεγάλα σύνολα δεδομένων.
  - Η επιλογή κατάλληλων παραμέτρων (C, g) και πυρήνα είναι κρίσιμη.

Ο SVC είναι ιδανικός για προβλήματα ταξινόμησης όπου απαιτείται υψηλή ακρίβεια και οι κλάσεις δεν είναι εύκολα διαχωρίσιμες γραμμικά.

Suport Vector Classifier Αποτελέσματα

- Συνάρτηση: $SVC()$
- Υπερπαράμετροι που δοκιμάστηκαν:
  1. $C = \{10, 100, 1000\}$
  2. $\gamma = \{0.1, 0.01, 0.001\}$
  3. $kernel = \{linear, poly, rbf, sigmoid\}$
  4. $class\_weight = \{dict, balanced, None\}$
- Accuracy: 0.8650657518582047
- Τελικές υπερπαράμετροι:
  - $C = 10$
  - $class\_weight = balanced$
  - $\gamma = 0.01$
  - $kernel = poly$
  - defaults

Multi-Layer-Perceptron Classifier

Χρησιμοποιήθηκε η συνάρτηση *MLPClassifier*().
Πρόκειται για έτοιμη υλοποίηση ενός νευρωνικού δικτύου που εκπαιδεύεται με τη μέθοδο του backpropagation.
Είναι ίσως το πιο γενικό μοντέλο που δοκιμάστηκε, μιας και μπορει με κατάλληλες υπερπαραμέτρους να προσεγγίσει οποιαδήποτε συνάρτηση.
Επομένως περιμένουμε πολύ καλά αποτελέσματα.

- **Βασική Δομή:**
  - Αποτελείται από τρία βασικά επίπεδα:
    - **Είσοδος (Input Layer):** Παίρνει τις εισόδους (χαρακτηριστικά των δεδομένων).
    - **Κρυφά Επίπεδα (Hidden Layers):** Περιέχουν νευρώνες που μαθαίνουν μη γραμμικές σχέσεις.
    - **Έξοδος (Output Layer):** Παράγει τις τελικές προβλέψεις (π.χ. πιθανότητες για κάθε κλάση).
  - Οι νευρώνες κάθε επιπέδου είναι πλήρως συνδεδεμένοι με τους νευρώνες του επόμενου επιπέδου.

Multi-Layer-Perceptron Classifier

- **Μηχανισμός Μάθησης**:
  - Χρησιμοποιείται η μέθοδος της **προώθησης (forward propagation)** για τον υπολογισμό της εξόδου.
  - Ο υπολογισμός των σφαλμάτων γίνεται με τη χρήση της συνάρτησης κόστους (π.χ. cross-entropy loss για ταξινόμηση).
  - Η **οπισθοδιάδοση (backpropagation)** ενημερώνει τα βάρη με βάση το σφάλμα μέσω του αλγορίθμου gradient descent.
- **Πλεονεκτήματα**:
  - Μπορεί να μάθει πολύπλοκα μοτίβα στα δεδομένα.
  - Προσαρμόζεται καλά σε μεγάλα σύνολα δεδομένων.
- **Μειονεκτήματα**:
  - Απαιτεί σημαντικούς υπολογιστικούς πόρους.
  - Υπάρχει κίνδυνος υπεράρμωσης (overfitting) αν δεν γίνει καλή ρύθμιση.

Ο MLP είναι ένα ευέλικτο εργαλείο για προβλήματα ταξινόμησης, αλλά απαιτεί σωστή παραμετροποίηση για βέλτιστα αποτελέσματα.

Multi-Layer-Perceptron Classifier Αποτελέσματα

- Συνάρτηση: *MLPClassifier*()
- Υπερπαράμετροι που δοκιμάστηκαν:
  - *hidden_layer_sizes* = $\{(10), (20), (30), (40), (50), (60), (70), (80), (90), (100), (500), (10000), (30, 30), (40, 40), (50, 50), (100, 100), (100, 10), (200, 20), (400, 36), (500, 50), (10, 100, 10), (20, 100, 20), (50, 500, 50)\}$, *default* = (100)
  - *activation* = $\{'logistic', 'tanh', 'relu'\}$, *default* = $'relu'$
  - *solver* = $\{'lbfgs', 'sgd', 'adam'\}$, (*default* = $'adam'$)
  - alpha = linear space between (0.000001, 100), *default* = 0.0001
  - *learning_rate* = $\{'comstant', 'adaptive'\}$, *default* = $'constant'$
- Accuracy: 0.8502001143510578
- Τελικές υπερπαράμετροι:
  - *hidden_layer_sizes* = (400, 36)
  - defaults

Ίσως με κατάλληλες υπερπαραμέτρους να μπορεί να βελτιωθεί περαιτέρω.

Σύγκριση Μοντέλων

Μοντέλο	Accuracy
Random Forest Classifier	0.8147512864493996
Gaussian Naive Bayes	0.6986849628359062
K-NN Classifier	0.8416237850200115
Support Vector Classifier	0.865065751858204 7
MLP Classifier	0.8502001143510578

Όπως φαίνεται στον πίνακα, τα καλύτερα αποτελέσματα πάρθηκαν μέσω του μοντέλου support vector classifier. Κάθε ένα από τα μοντέλα ενδέχεται να είχε καλύτερα αποτελέσματα με κατάλληλη επιλογή υπερπαραμέτρων. Ειδικότερα το νευρωνικό δίκτυο θα έπρεπε να έχει εξίσου καλά αποτελέσματα με το support vector όμως δεν βρέθηκαν οι κατάλληλες υπερπαράμετροι.
Η διαφορά στο accuracy μπορεί επίσης να οφείλεται σε ιδιαιτερότητες του test set, και δεν είναι απαραίτητο ότι το support vector θα έχει τα καλύτερα αποτελέσματα στο dataset που μας δόθηκε να κάνουμε την πρόβλεψη.

Κατάληξη

Η ταξινόμηση των τελικών αποτελεσμάτων έγινε με το support vector μοντέλο, με υπερπαραμέτρους:

- C = 10
- gamma = 0.01
- kernel = 'poly'
- class_weight = 'balanced'

Ό,τι προσπάθεια και να κάναμε, δεν καταφέραμε να υπερβούμε το κατόφλι των $\sim 85\%$. Υποθέτουμε, έτσι, ότι υπήρχε κάποια επικάλυψη των κλάσεων, ώστε ήταν να είναι αδύνατη η ακριβέστερη ταξινόμηση χωρίς επιπλέον features.
Ελπίζουμε να μην έπεσε θύμα της υπεράρμωσης, αφού χρησιμοποιήσαμε 3-fold cross valitation στην εκπαίδευση του και η αξιολόγησή του έγινε σε ξεχωριστό σετ δεδομένων.
Τα αποτελέσματα αποθηκεύτηκαν στο αρχείο labels30.npy. Μπορείτε να βρείτε την δουλειά μας στο github εδώ.