Αναγνώριση Προτύπων

Εργασία 2

Ονοματεπώνυμο: Μωραΐτη Παναγιώτα

ΑΜ: 58054

Πίνακας περιεχομένων

Άσκηση 1

Έχουμε τα δεδομένα δύο χαρακτηριστικών x1 και x2από 3 κλάσεις (ω1, ω2, ω3) σε ένα αρχείο. Κάθε γραμμή του αρχείου περιέχει τα δεδομένα στη μορφή: x1, x2, class_label.

Α) Να γράψετε κατάλληλο κώδικα ώστε να εκτιμήσετε τις πυκνότητες πιθανότητας με τη μέθοδο παραθύρων Parzen. Να θεωρήσετε hN=0.3 να απεικονίσετε κατάλληλα τις κατανομές σε κοινό γράφημα. Δοκιμάστε να κάνετε εκτίμηση για hN=0.7 και hN=0.1. Τι παρατηρείτε; Εάν είχατε στη διάθεσή σας μόνο το 25% των δεδομένων, τι τιμή θα έπρεπε να έχει το hN για να κάνετε εκτίμηση με παρόμοια λεπτομέρεια με την αρχική; Επιβεβαιώστε την απάντησή σας πειραματικά.

Για να υπολογίσουμε την pdf στο σημείο x, προσθέτουμε τη συνάρτηση παραθύρου για όλα τα N σημεία xi και κανονικοποιούμε το άθροισμα με το NV, οπότε προκύπτει:

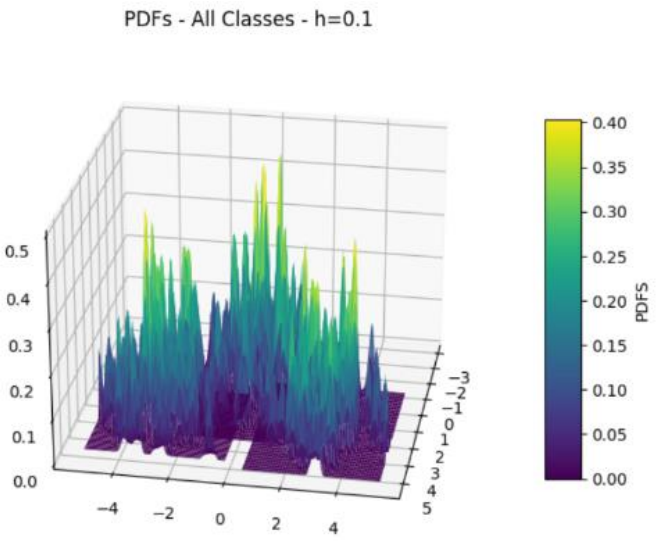
$$p_N(x) = \frac{1}{NV_N}\sum_{i=1}^{N}\varphi\left(\frac{\mathbf{x}-\mathbf{x_i}}{h_N}\right)$$

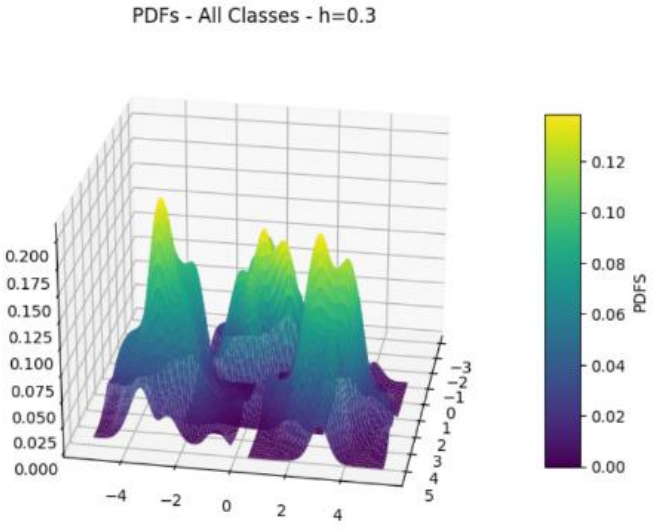
Η συνάρτηση παραθύρου είναι συνήθως η πολυδιάστατη κανονική κατανομή.

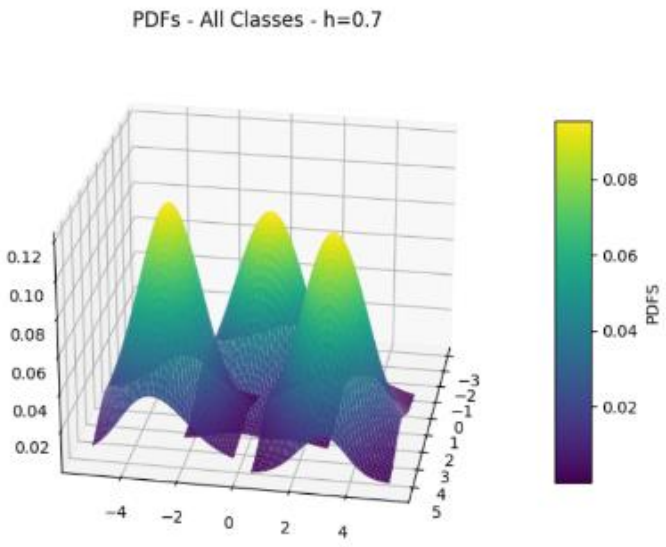
$$p_N(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi)^{\frac{d}{2}}h_N^d}\exp\left(-\frac{(\mathbf{x}-\mathbf{x_i})^T(\mathbf{x}-\mathbf{x_i})}{2h_N^2}\right)$$

Επειδή έχουμε δύο χαρακτηριστικά (2 διαστάσεις), δημιουργήσαμε ένα grid 2D και σε κάθε σημείο του grid υπολογίσαμε την pdf. Επειδή έχουμε 3 κλάσεις δημιουργήσαμε 3 διαφορετικά grid για μεγαλύτερη ακρίβεια. Στην ουσία x είναι τα σημεία του grid στα οποία θέλουμε να εκτιμήσουμε την pdf και xi είναι τα σημεία που μας δίνονται στο dataset.

Παρακάτω φαίνονται οι κατανομές σε κοινό γράφημα για διαφορετικές τιμές της παραμέτρου h.







Παρατηρούμε ότι για μεγαλύτερο h, το παράθυρο είναι πλατύτερο και η εκτίμηση της συνάρτησης πυκνότητας πιθανότητας είναι πιο ομαλή (εξομάλυνση τοπικών διακυμάνσεων πυκνότητας- μειώνεται η λεπτομέρεια), είναι υπέρθεση n πλατιών συναρτήσεων. Επίσης, οι τιμές της pdf είναι μικρότερες.

Για μικρότερο h η pdf είναι υπέρθεση n στενών συναρτήσεων (αιχμηρές συναρτήσεις τύπου δέλτα με κέντρα τα σημεία εκπαίδευσης), η εκτίμηση εμπεριέχει περισσότερο θόρυβο (μεγάλη διασπορά). Οι τιμές της pdf είναι μεγαλύτερες, οι κορυφές δηλαδή στο διάγραμμα ανεβαίνουν ψηλότερα.

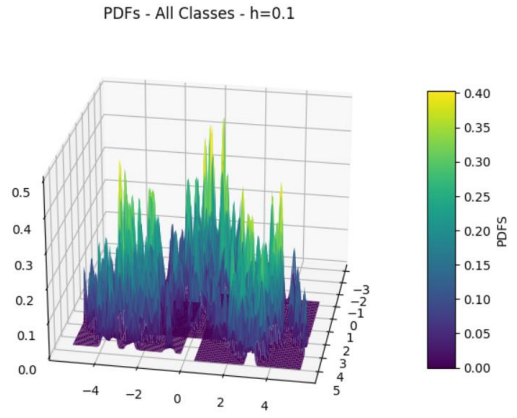
Αν αυξήσουμε το μέγεθος N του συνόλου δεδομένων, η διασπορά μειώνεται και η εκτίμηση γίνεται πιο ακριβής. Καθώς το N τείνει στο άπειρο η εκτίμηση γίνεται ακριβής ανεξάρτητα από το πλάτος του παραθύρου h.

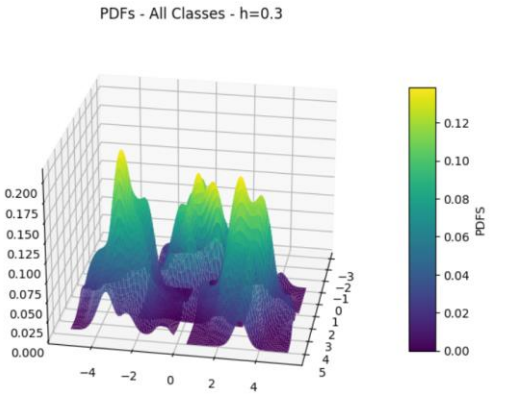
Αν κρατήσουμε μόνο το 25% των αρχικών δεδομένων παρατηρούμε ότι οι δύο εκτιμήσεις είναι αρκετά κοντά μεταξύ τους, όσον αφορά το σχήμα των καμπυλών, ειδικά για μεγαλύτερες τιμές h. Όμως η εκτίμηση δεν είναι τόσο ακριβής, οι κορυφές των pdf φαίνεται να φτάνουν υψηλότερα σε σχέση με τις εκτιμήσεις με όλα τα δεδομένα για μικρότερα h (μεγαλύτερη ανακρίβεια).

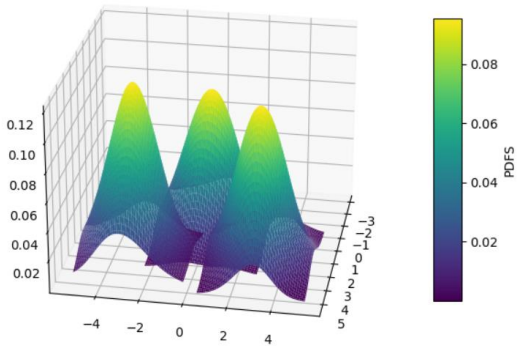
Αν μειώσουμε τα δεδομένα μας, για να κάνουμε εκτίμηση με παρόμοια λεπτομέρεια με την αρχική πρέπει να επιλέξουμε **μικρότερη τιμή για το h**, ώστε να διατηρήσουμε τη λεπτομέρεια. Βέβαια όσο μικρότερο το h, τόσο μεγαλύτερος ο κίνδυνος να οδηγηθούμε σε overfitting. Σε αντίθετη περίπτωση, μεγαλύτερες τιμές του h θα εξομαλύνουν την εκτίμηση, αλλά μπορεί να χαθεί λεπτομέρεια.

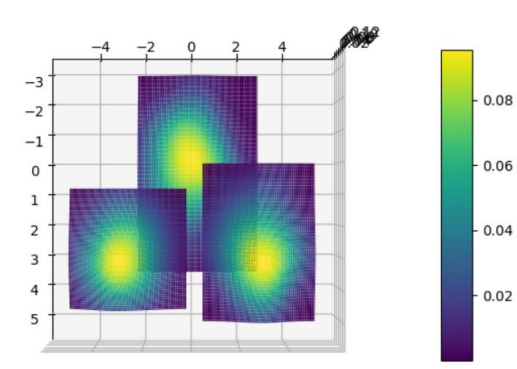
Αυτό το διαπιστώνουμε κάνοντας δοκιμές και παρατηρώντας που έχουμε την μεγαλύτερη λεπτομέρεια στην εκτίμηση σε σύγκριση με την αρχική. Παρακάτω μπορούμε να συγκρίνουμε τα διαγράμματα.

Όλα τα δεδομένα

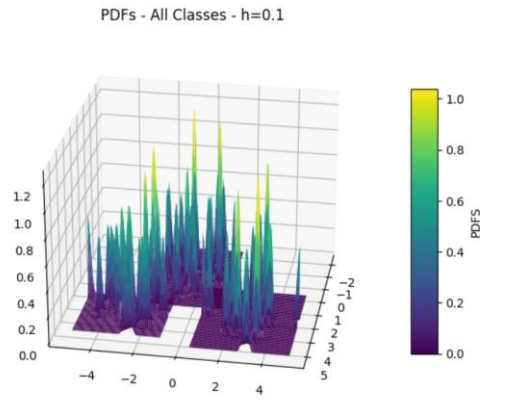


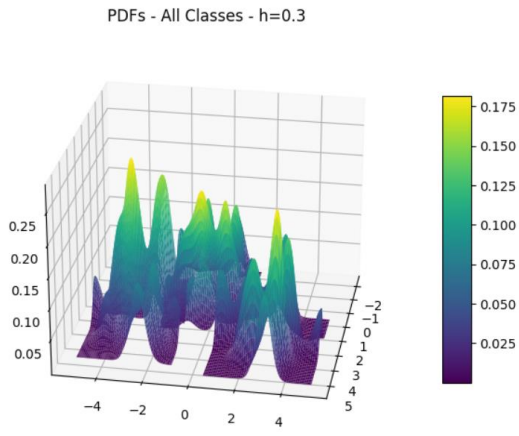


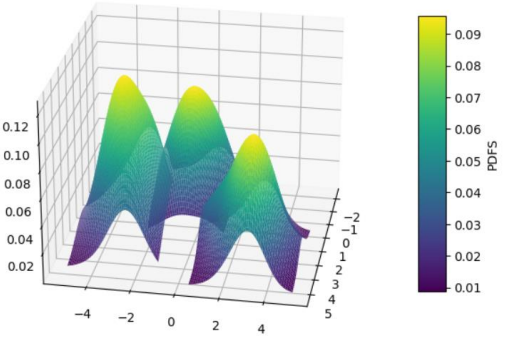


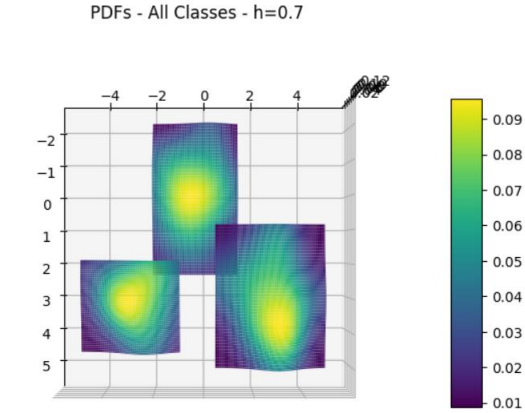


Μόνο το 25% των αρχικών δεδομένων

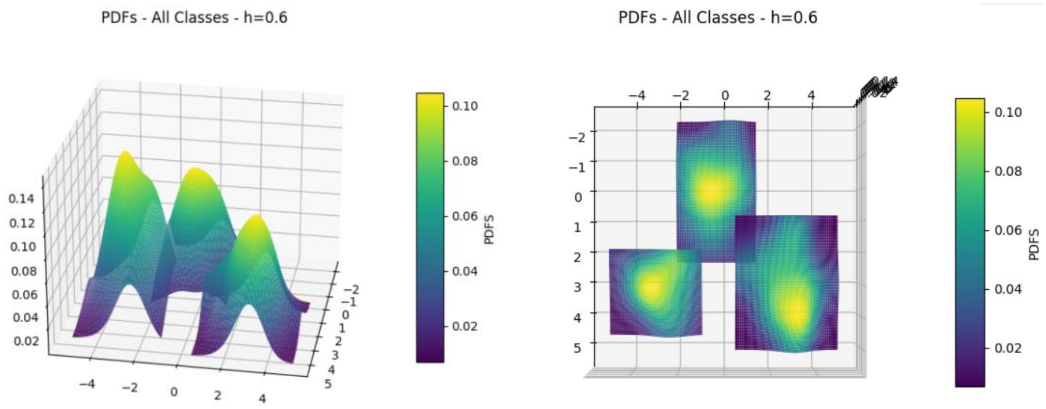








Παρατηρούμε ότι η καλύτερη τιμή για το h, ώστε να αποφύγουμε το overfitting (όχι πολύ μικρή τιμή), αλλά να διατηρήσουμε τη λεπτομέρεια (όχι πολύ μεγάλη τιμή) είναι περίπου **0.6**.

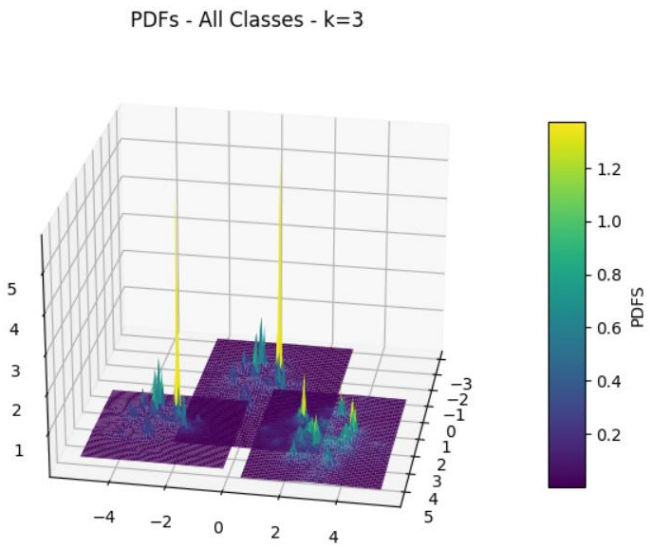


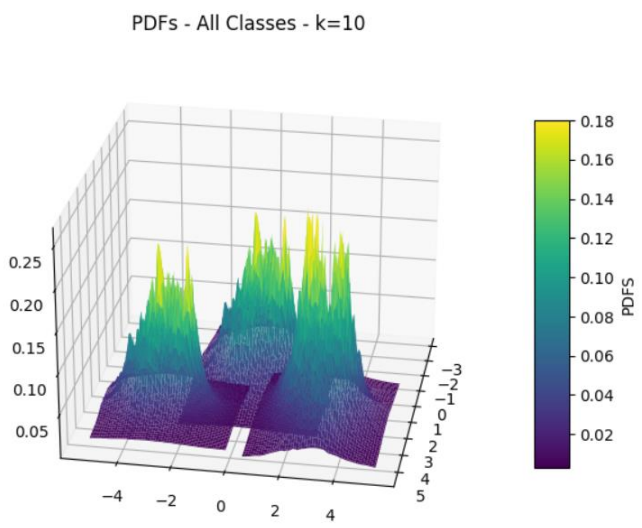
Β) Να γράψετε κατάλληλο κώδικα ώστε να εκτιμήσετε τις πυκνότητες πιθανότητας με τη μέθοδο k-NN, και για k=10 να απεικονίσετε κατάλληλα τις κατανομές σε κοινό γράφημα. Δοκιμάστε να κάνετε εκτίμηση για k=3 και k=30. Τι παρατηρείτε;

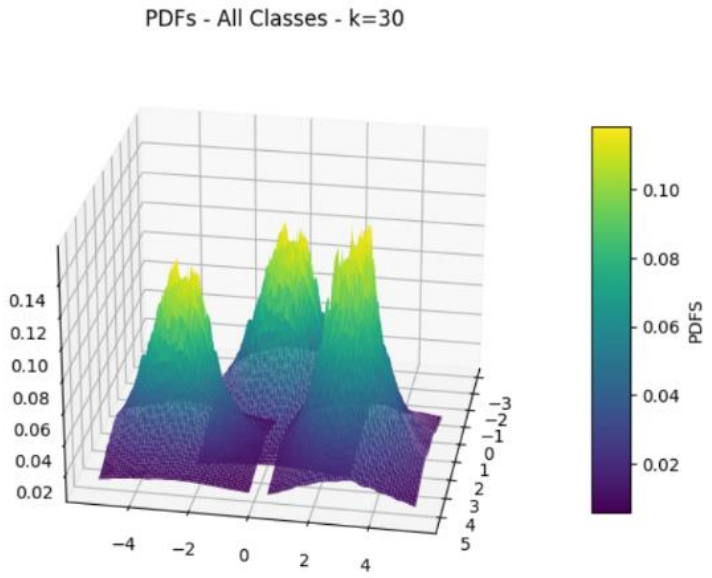
Για να εκτιμήσουμε τις pdf με τη μέθοδο k-NN στο σημείο x (σημείο του grid), πρέπει να βρούμε την απόσταση του x από όλα τα δεδομένα (dataset). Επιλέξαμε να χρησιμοποιήσουμε την Ευκλείδεια απόσταση. Στη συνέχεια, βρίσκουμε τα κ πλησιέστερα δεδομένα στο x, δηλαδή τους κ κοντινότερους γείτονες. Υπολογίζουμε τον όγκο V που περικλείει τα κ σημεία. Όταν χρησιμοποιούμε την Ευκλείδεια απόσταση έχουμε υπερσφαίρες (για απόσταση Mahalanobis υπερελλείψεις). Αυτό σημαίνει ότι για τις δύο διαστάσεις θα έχουμε έναν κύκλο με κέντρο το x που θα περικλείει κ σημεία. Οπότε, αρκεί να βρω την απόσταση από τον κ γείτονα για να υπολογίσω τον όγκο και η pdf δίνεται από τη σχέση: $p(\mathrm{x}) = \frac{k_n}{n * V_n}$.

Εφόσον βρισκόμαστε στις δύο διαστάσεις πρέπει να υπολογίσουμε τον όγκο ενός κύκλου (πr^2) με ακτίνα r ίση με την απόσταση από τον κ γείτονα.

Όπως στο ερώτημα α δημιουργούμε 3 διαφορετικά grid και εκτιμούμε την pdf σε κάθε σημείο τους.







Παρατηρούμε ότι όσο το κ αυξάνεται, τόσο αυξάνεται και η ακρίβεια της εκτίμησης. Για πολύ μικρά κ η εκτίμηση δεν πλησιάζει ούτε στο ελάχιστο την πραγματική.

Γ) Θεωρώντας ότι όλες οι κλάσεις έχουν ίδιες apriori πιθανότητες, να απεικονίσετε τα δεδομένα και τις περιοχές απόφασης για ταξινόμηση σύμφωνα με τον κανόνα του Bayes και κατανομές που εκτιμώνται με τη μέθοδο των παραθύρων parzen για hN=0.1, hN=0.3, hN=1.5. Πως επηρεάζονται οι περιοχές απόφασης από την παράμετρο παραθύρου; Σχολιάστε.

Σύμφωνα με τον κανόνα του Bayes, για να κατηγοριοποιήσω ένα δείγμα σε μια κλάση χρησιμοποιώ το MAP (maximum aposteriori estimation) πρέπει δηλαδή να ισχύει:

$$P(\omega_1 \mid \mathbf{x}) \geq P(\omega_2 \mid \mathbf{x})$$

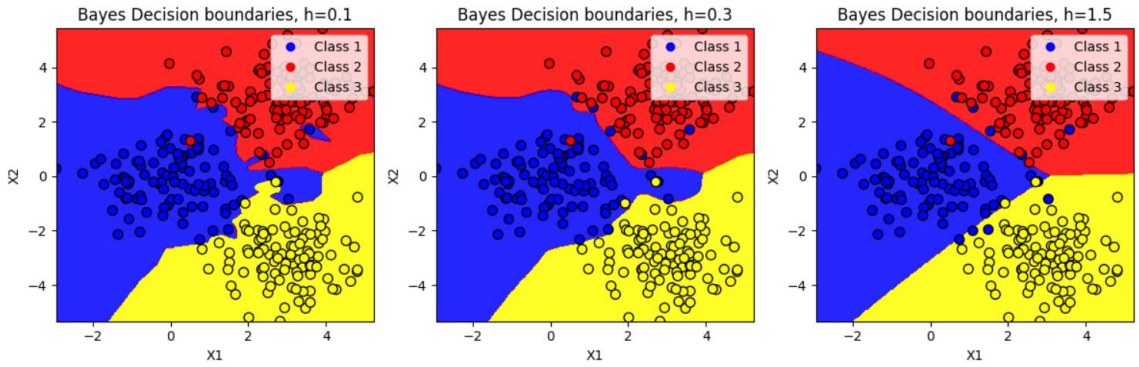
Αντικαθιστώντας στον τύπο προκύπτει:

$$\frac{p(x \mid \omega_1)P(\omega_1)}{p(x)} \geq \frac{p(x \mid \omega_2)P(\omega_2)}{p(x)}$$

Εφόσον οι apriori πιθανότητες είναι ίσες για όλες τις κλάσεις, το δείγμα θα κατηγοριοποιηθεί στην κλάση με το μεγαλύτερο likelihood (πιθανοφάνεια), δηλαδή στην κλάση με την μεγαλύτερη τιμή pdf.

Για να γίνει αυτό επιλέξαμε να δημιουργήσουμε ένα grid που θα εμπεριέχει τις περιοχές και των 3 κλάσεων. Αυτό έγινε για να μην έχουμε κενές περιοχές τις οποίες δεν θα μπορούμε να αποδώσουμε σε κάποια κλάση, κατά την απεικόνιση των περιοχών απόφασης. Στην ουσία εκτιμούμε την pdf σε κάθε σημείο x του grid με τη μέθοδο των παραθύρων Parzen για κάθε κλάση και στη συνέχεια βλέπουμε για ποια κλάση η pdf έχει μεγαλύτερη τιμή και σε αυτήν αποδίδουμε το σημείο x.

Οι περιοχές απόφασης φαίνονται στην παρακάτω εικόνα, τα σημεία είναι τα δεδομένα που μας δόθηκαν.

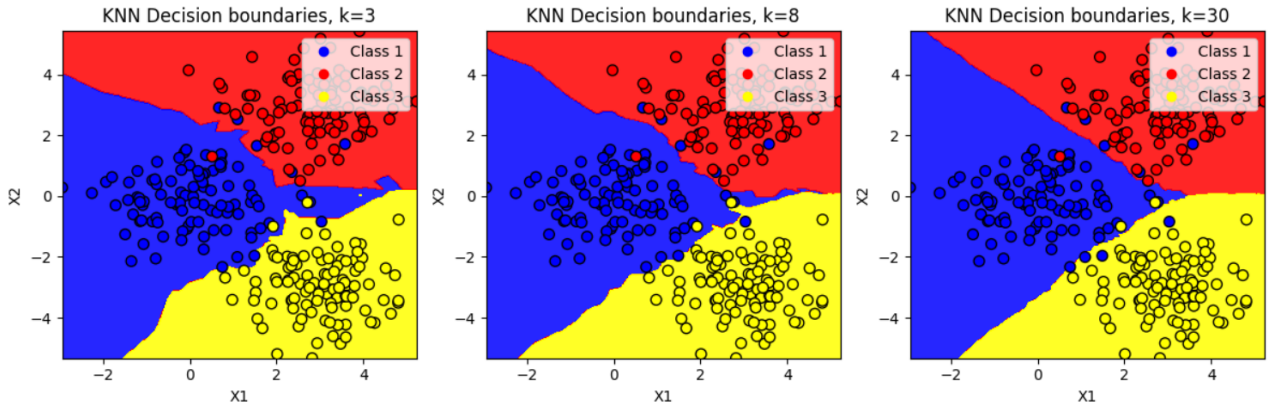


Παρατηρούμε ότι για πολύ μικρό h έχουμε πολύ μικρότερη και λεπτομερής διαμέριση του χώρου. Το σφάλμα εκπαίδευσης μπορεί να γίνει πολύ μικρό, γεγονός όμως που δεν είναι επιθυμητό γιατί μπορεί να οδηγήσει σε overfitting (υπερβολική προσκόλληση στο σύνολο εκπαίδευσης-δεν επιτυγχάνεται η γενίκευση). Δηλαδή, το σφάλμα θα είναι πολύ μεγάλο όταν θα χρειαστεί να ταξινομηθούν νέα δεδομένα, εκτός του συνόλου εκπαίδευσης. Μεγαλύτερο h οδηγεί σε περισσότερα λάθη κατά την εκπαίδευση, η επιφάνεια απόφασης είναι πιο ομαλή. Αυτό οδηγεί σε καλύτερη γενίκευση σε νέα δεδομένα, κάτι που είναι επιθυμητό.

Δ) Να απεικονίστε τα δεδομένα και τις περιοχές απόφασης για ταξινόμηση σύμφωνα με τον κανόνα k-NN για k=3, k=8 και k=30. Πως επηρεάζονται οι περιοχές απόφασης από την παράμετρο k; Σχολιάστε.

Όπως και στο ερώτημα γ έχουμε ένα grid. Για κάθε σημείο x του grid υπολογίζουμε τις Ευκλείδειες αποστάσεις από όλα τα σημεία των δεδομένων (dataset) μας. Στη συνέχεια, κρατάμε τα κ σημεία του dataset με τις κ μικρότερες αποστάσεις και κοιτάμε σε ποιες κλάσεις ανήκουν. Τέλος, αποδίδουμε το σημείο x στην κλάση στην οποία ανήκουν οι περισσότεροι από τους κοντινότερους γείτονες.

Οι περιοχές απόφασης φαίνονται στην παρακάτω εικόνα, τα σημεία είναι τα δεδομένα που μας δόθηκαν.



Παρατηρούμε ότι για μικρότερο κ, η επιφάνεια απόφασης έχει πιο περίπλοκο σχήμα. Αυτό μπορεί να οδηγήσει σε overfitting. Για μεγαλύτερα κ η επιφάνεια απόφασης είναι πιο ομαλή, αυτό σημαίνει ότι μπορεί να επιτευχθεί η γενίκευση σε νέα δεδομένα.

Ε) Ποια τα κυριότερα πλεονεκτήματα και μειονεκτήματα των δύο τεχνικών τόσο στη μεταξύ τους σύγκριση, όσο και σε σύγκριση με γεωμετρικές τεχνικές ταξινόμησης (π.χ. linear discriminants, SVMs κλπ.);

Οι μη παραμετρικές τεχνικές μπορούν να έχουν πολύ μεγάλη απόδοση σε προβλήματα ταξινόμησης, καθώς δεν κάνουν υποθέσεις για τη δομή των δεδομένων ή την κατανομή τους, επιτρέποντας τους να προσαρμόζονται σε πιο πολύπλοκες σχέσεις. Επίσης, Μπορούν να λειτουργήσουν καλά ακόμη και όταν οι κλάσεις έχουν διαφορετικό αριθμό δειγμάτων. Σε μικρά σύνολα δεδομένων, όπου η πολυπλοκότητα των παραμετρικών τεχνικών μπορεί να οδηγήσει σε overfitting, οι μη παραμετρικές τεχνικές μπορεί να έχουν καλή απόδοση.

Το μειονέκτημά τους είναι ότι καθώς η διάσταση του προβλήματος αυξάνει, υπάρχει πολύ μεγάλη απαίτηση σε δεδομένα (curse of dimensionality). Η έλλειψη αρκετών δεδομένων εκπαίδευσης μπορεί να επηρεάσει την απόδοσή τους. Άρα σε χώρους υψηλών διαστάσεων, οι μη παραμετρικές τεχνικές μπορεί να μην είναι τόσο αποτελεσματικές. Επίσης, οι τεχνικές αυτές είναι ευαίσθητες στο θόρυβο και τις ανεπιθύμητες διακυμάνσεις στα δεδομένα, καθώς δεν κάνουν υποθέσεις για την κατανομή των δεδομένων.

Όσον αφορά την ταξινόμηση με τη μέθοδο των παραθύρων Parzen, το πλεονέκτημα της είναι ότι δεν προϋποθέτει καμία γνώση για το πρόβλημα εκτός από την ύπαρξη του συνόλου δειγμάτων εκπαίδευσης. Όμως, όταν έχουμε μεγάλες αλλαγές πυκνότητας θα θέλαμε παράθυρα μικρού πλάτους σε περιοχές υψηλής πυκνότητας και παράθυρα μεγαλύτερου πλάτους σε περιοχές χαμηλής πυκνότητας, κάτι τέτοιο δεν μπορεί να επιτευχθεί με τα παράθυρα Parzen που θεωρούν το παράθυρο σταθερό, ανεξαρτήτως της πυκνότητας των δεδομένων. Επίσης, απαιτεί τον υπολογισμό της pdf σε κάθε σημείο, κάτι που είναι απαιτητικό υπολογιστικά. Τέλος, ένα άλλο μειονέκτημα είναι ότι απαιτείται η επιλογή της παραμέτρου h και υπάρχει πολύ μεγάλη ευαισθησία στη μεταβολή αυτής της παραμέτρου.

Η ταξινόμηση με τη χρήση της μεθόδου κ-πλησιέστερων γειτόνων έχει το πλεονέκτημα ότι το παράθυρο δεν έχει σταθερό πλάτος, αλλά αλλάζει ανάλογα με την πυκνότητα των

δεδομένων σε κάθε σημείο. Το παράθυρο επεκτείνεται μέχρι να συμπεριλάβει κ σημεία, δηλαδή κ γείτονες. Άρα σε δεδομένα με μεγάλες αλλαγές στην πυκνότητα, μπορεί να κάνει πολύ καλή εκτίμηση. Το αρνητικό είναι ότι πρέπει να επιλέξουμε την παράμετρο κ. Επίσης, ο KNN απαιτεί τεράστιο υπολογιστικό κόστος (πολυπλοκότητα O(κN)^2) για μεγάλο αριθμό δεδομένων εκπαίδευσης, αυτό όμως το πρόβλημα μπορεί να περιοριστεί με διάφορες τεχνικές.

Άσκηση 2

Για να παράξουμε τυχαία δείγματα, τα οποία αποτελούνται από περισσότερα από ένα χαρακτηριστικά και ακολουθούν Γκαουσιανές κατανομές με γνωστές μέσες τιμές και πίνακες συνδιασποράς, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση **numpy.random.multivariate_normal**. Η συνάρτηση αυτή είναι μια γενίκευση της μονοδιάστατης κανονικής κατανομής (numpy.random.normal) σε υψηλότερες διαστάσεις.

Α) Υλοποιείστε τον αλγόριθμό του Batch Perceptron και με αυτόν υπολογίστε έναν γραμμικό ταξινομητή για τις κλάσεις αυτές. Απεικονίστε την επιφάνεια απόφασης που προέκυψε επάνω στο προηγούμενο γράφημα.

Για τον διαχωρισμό δύο γραμμικώς διαχωρίσιμων κλάσεων αρκεί να υπολογιστεί κάποιο υπερεπίπεδο που να τις διαχωρίζει. Το υπερεπίπεδο, αν τα δεδομένα μας είναι διάστασης d, θα είναι διάστασης d-1 και περιγράφεται από την ακόλουθη σχέση:

$$g(\mathrm{x}) = \mathrm{w}^T\, x + w_0$$

$$w^T = [w_1, w_2, \ldots, w_d]$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_d \end{bmatrix}$$

Τα w ονομάζονται βάρη και τα x είναι τα χαρακτηριστικά (features) που έχουν τα δεδομένα μας.

Μπορούμε να θεωρήσουμε ότι:

$$g(\mathrm{x}) = \mathrm{w}\, x$$

$$w = [w_0, w_1, w_2, \ldots, w_d]$$

$$x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \ldots \\ x_d \end{bmatrix}$$

- Αν wx>0, το δείγμα ανήκει στην κλάση ω1.
- Αν wx<0, το δείγμα ανήκει στην κλάση ω2.

Μπορούμε να ενώσουμε τις δύο ανισώσεις σε μία με τον ακόλουθο τρόπο. Αν ορίσουμε τις ετικέτες (labels) να είναι:

$$y = \begin{cases} 1, \text{ το δεύγμα ανήκει στην } \omega 1 \\ -1, \text{ το δεύγμα ανήκει στην } \omega 2 \end{cases}$$

Οπότε έχουμε:

Αν **wxy>0**, τότε το δείγμα είναι ταξινομημένο σωστά.

Το κόστος για τον αλγόριθμο batch perceptron ορίζεται με την ακόλουθη σχέση:

$J(\mathrm{w}) = \sum_{x \in X} (-\mathrm{wx})$, όπου X είναι το σύνολο των δειγμάτων που δεν έχουν ταξινομηθεί σωστά.

Η συνάρτηση κόστους μηδενίζεται όταν το w είναι διάνυσμα λύσης (καμία λάθος ταξινόμηση).

Στο πρόβλημά μας έχουμε να διαχωρίσουμε δύο κλάσεις που αποτελούνται από δύο χαρακτηριστικά η κάθε μία. Οι κλάσεις είναι γραμμικώς διαχωρίσιμες, αν δημιουργήσουμε ένα διάγραμμα μπορούμε να δούμε ότι μπορούν να διαχωριστούν με μια ευθεία γραμμή.

Για να βρούμε μια ευθεία γραμμή που διαχωρίζει τις δύο κλάσεις, πρέπει να ελαχιστοποιήσουμε το κόστος, να βρούμε δηλαδή τα κατάλληλα βάρη w που θα κάνουν το κόστος ελάχιστο (θέλουμε να γίνει 0 αν αυτό είναι δυνατό). Η ελαχιστοποίηση της συνάρτησης μπορεί να γίνει με τον αλγόριθμο Gradient Descent, στον οποίο η ανανέωση των βαρών γίνεται με τον ακόλουθο τρόπο: $w_i := w_i - \eta \frac{\partial \mathrm{J(w)}}{\partial w_i}$.

Η υπερπαράμετρος η ονομάζεται ρυθμός μάθησης και ρυθμίζει το πόσο θα αλλάζουν τα βάρη σε κάθε ανανέωση.

Για τον αλγόριθμο για τον αλγόριθμο batch perceptron ισχύει: $\nabla \mathrm{J(w)} = \frac{\partial \mathrm{J(w)}}{\partial w_i} = \sum_{x \in X} (-\mathrm{x})$

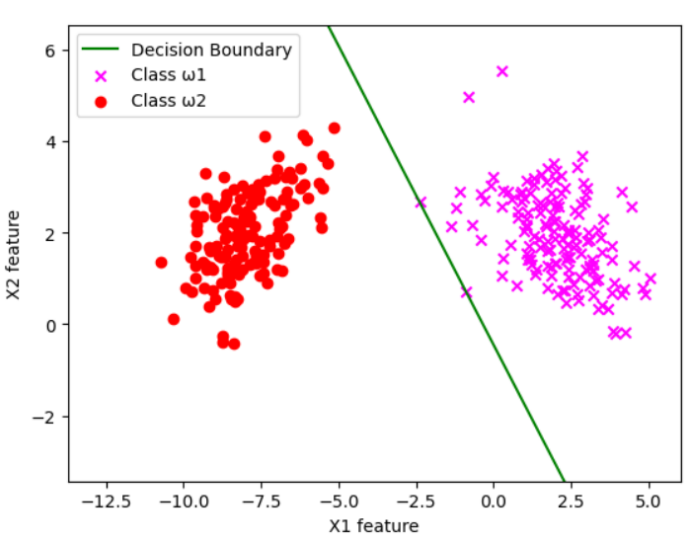
Όλα τα παραπάνω υλοποιήθηκαν με κώδικα. Ο πίνακας x περιέχει τα δείγματα σε κάθε γραμμή του (έχουμε 300 δείγματα άρα 300 γραμμές). Στην 1η στήλη έχει άσσους και στην 2η και 3η περιέχει τα χαρακτηριστικά. Οπότε θα έχουμε 3 βάρη, w0, w1, w2. Ο πίνακας y, περιέχει τις ετικέτες 1 και -1 για κάθε δείγμα, ανάλογα με την κλάση στην οποία ανήκει.

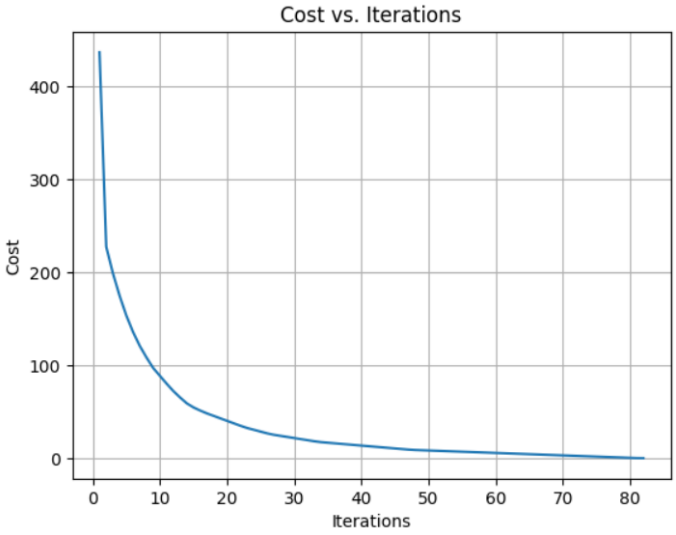
Υλοποιούμε την επαναληπτική διαδικασία. Τα κόστη τα κρατάμε σε μια λίστα για να κάνουμε το διάγραμμα αν θέλουμε να δούμε πως εξελίσσεται το κόστος. Για το κόστος αθροίζονται μόνο οι προβλέψεις που είναι λανθασμένες (wxy<0).

Για την παράγωγο, πρέπει να πάμε στα δείγματα (X), στα οποία η πρόβλεψη είναι λάθος και να αθροίσουμε στον κατακόρυφο άξονα, δηλαδή να αθροίσουμε κατά μήκος των 3 χαρακτηριστικών (προκύπτουν 2 τιμές). Τέλος, ανανεώνουμε τα βάρη.

Η συνθήκη τερματισμού εδώ επιλέξαμε να είναι η σωστή ταξινόμηση όλων των δειγμάτων, λόγω της απλότητας του προβλήματος. Συνήθως επιλέγουμε να σταματάμε τη διαδικασία όταν το κόστος γίνει πολύ μικρό, που σημαίνει ότι έχουμε φτάσει σε μια καλή λύση.

Παρακάτω φαίνεται η επιφάνεια απόφασής και το διάγραμμα κόστους.





Στο πρόβλημα με μόνο 2 κατηγορίες, οι οποίες είναι γραμμικά διαχωρίσιμες μπορούμε να έχουμε πολλούς γραμμικούς ταξινομητές (πολλές διαφορετικές γραμμές για δύο χαρακτηριστικά) που να διαχωρίζουν τα δύο σύνολα.

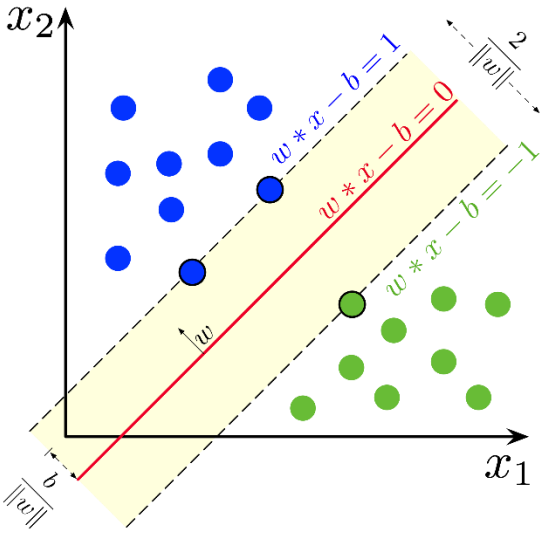
Η ανανέωση των βαρών στο batch perceptron βασίζεται στα λάθος ταξινομημένα δείγματα. Οπότε, αν όλα τα δείγματα έχουν ταξινομηθεί σωστά, δεν μπορούν να ανανεωθούν περεταίρω τα βάρη και να βρεθεί κάποια καλύτερη επιφάνεια απόφασης. Άρα, ο αλγόριθμος αυτός δε βρίσκει βέλτιστη λύση. Για αυτό το λόγο, όπως βλέπουμε στο διάγραμμα η γραμμή είναι πολύ κοντά στην μία από τις δύο κλάσεις. Όταν η γραμμή είναι πολύ κοντά στη ροζ κλάση, υπάρχει μεγαλύτερη πιθανότητα περισσότερα δείγματα να ταξινομηθούν στην κόκκινη κλάση λανθασμένα, ενώ βρίσκονται κοντά στη ροζ στην οποία και ανήκουν.

Το σύνορο που διαχωρίζει τις κατηγορίες, πρέπει να είναι όσο το δυνατόν πιο μακριά από τα δεδομένα, ώστε αν έρθει κάποιο νέο δείγμα, να έχουμε περισσότερες πιθανότητες να το ταξινομήσουμε σωστά. Κάτι τέτοιο δεν μπορούμε να το επιτύχουμε με το batch perceptron.

Β) Χρησιμοποιείστε γραμμικό SVM (από κατάλληλη βιβλιοθήκη της επιλογής σας) για να υπολογίσετε έναν νέο γραμμικό ταξινομητή για τα ίδια δεδομένα. Απεικονίστε τα δεδομένα, τα support vectors και το επίπεδο απόφασης σε νέο διάγραμμα.

Για να βρούμε τη βέλτιστη επιφάνεια απόφασης που διαχωρίζει δύο κατηγορίες, μπορούμε να χρησιμοποιήσουμε τα support vector machines (svm).

Στόχος του svm είναι, εάν έχουμε δύο κατηγορίες που είναι γραμμικά διαχωρίσιμες, να βρει την επιφάνεια απόφασης, η οποία μεγιστοποιεί το περιθώριο (margin) και από τις δύο κατηγορίες.



Το περιθώριο (margin), δίνεται από τη σχέση:

$$m = \frac{1}{\|\mathbf{w}\|} + \frac{1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

Επίσης, πρέπει να ισχύει:

$$\mathbf{w}^{\mathbf{T}}\mathbf{x} + w_0 \geq 1 \quad \forall \mathbf{x} \in \omega_1$$
$$\mathbf{w}^{\mathbf{T}}\mathbf{x} + w_0 \leq -1 \quad \forall \mathbf{x} \in \omega_2$$

Μπορούμε να κάνουμε τις δύο ανισότητες μία με το κόλπο που αναφέραμε και στο batch percetron.

Οπότε, αν ισχύει η παρακάτω σχέση τα δείγματα έχουν ταξινομηθεί σωστά.

$$y_i(\underline{w}^T \underline{x}_i + w_0) \geq 1, \quad i = 1,2,\ldots,N$$

$$y_i = 1, \text{ for } \underline{x}_i \in \omega_i,$$
$$y_i = -1, \text{ for } \underline{x}_i \in \omega_2$$

Τέλος, μπορούμε να μετατρέψουμε το πρόβλημα, από πρόβλημα μεγιστοποίησης σε πρόβλημα ελαχιστοποίησης, το οποίο έχει και έναν περιορισμό (constrained optimization).

$$\text{maximize} \frac{2}{\|w\|} \Rightarrow \text{minimize} \quad J(w) = \frac{1}{2}\|w\|^2 \quad \text{s.t. } y_i(w \cdot x_i + w_0) - 1 \geq 0, \quad i = 1,2,..,N$$

Το υπερεπίπεδο του ταξινομητή που βρίσκουμε με τη χρήση του svm είναι μοναδικό.

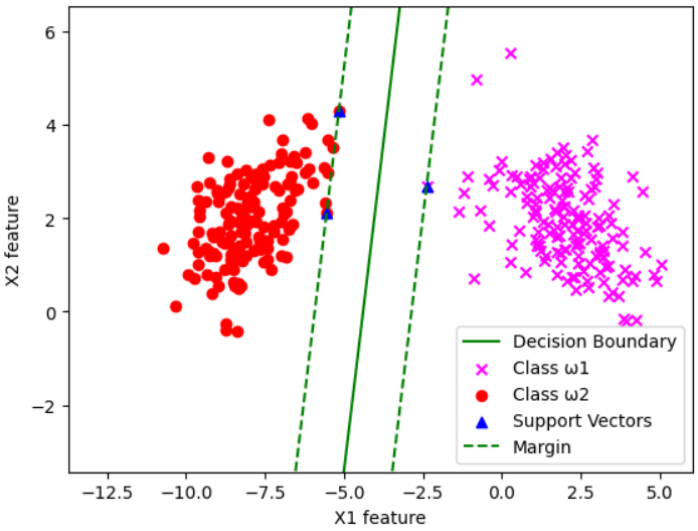
Τα σημεία που βρίσκονται μακριά από το υπερεπίπεδο δεν συνεισφέρουν στη διαμόρφωση της επιφάνειας απόφασης. Τα σημεία που βρίσκονται πιο κοντά στην επιφάνεια απόφασης (ορίζουν το περιθώριο), ονομάζονται διανύσματα υποστήριξης (support vectors).

Για να ορίσουμε έναν γραμμικό svm ταξινομητή χρησιμοποιήσαμε τη βιβλιοθήκη sklearn (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html), η οποία βασίζεται στη βιβλιοθήκη libsvm.

Προσοχή πρέπει να δώσουμε στο γεγονός ότι πλέον η 1η στήλη στον πίνακα x που έχουμε δεν έχει πλέον άσσους. Ο πίνακας x περιέχει μόνο τις δύο στήλες με τα χαρακτηριστικά.

Με τη συνάρτηση fit εκπαιδεύεται ο ταξινομητής στα δεδομένα μας. Στη συνέχεια, από κατάλληλες συναρτήσεις μπορούμε να πάρουμε τα διανύσματα υποστήριξης και τα βάρη w0, w1, w2.

Στο παρακάτω διάγραμμα φαίνονται οι κλάσεις, η επιφάνεια απόφασης, τα διανύσματα υποστήριξης και το περιθώριο. Πλέον αυτή η γραμμή που διαχωρίζει τα δεδομένα είναι η βέλτιστη.

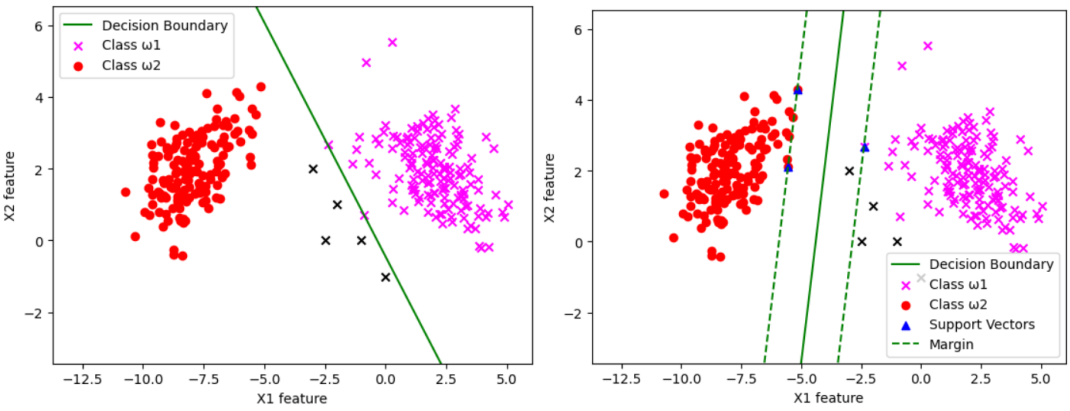


Γ) Σχολιάστε το αποτέλεσμα των δύο τεχνικών. Ποιες οι διαφορές και που οφείλονται;

Το batch perceptron βρίσκει κάποια επιφάνεια απόφασης που εξαρτάται από τις τιμές στις οποίες αρχικοποιούνται τα βάρη. Αυτή η επιφάνεια δεν είναι η βέλτιστη, καθώς ο αλγόριθμος αυτός μόλις βρει κάποια λύση για τα δεδομένα μας, σταματάει να ανανεώνει τα βάρη, ανεξάρτητα από το αν η λύση αυτή είναι καλή.

Ο svm βρίσκει πάντα τη βέλτιστη επιφάνεια απόφασης (αν οι κλάσεις είναι γραμμικώς διαχωρίσιμες). Η επιφάνεια απόφασης που βρίσκει μεγιστοποιεί το περιθώριο, δηλαδή βρίσκεται όσο το δυνατόν μακρύτερα από τις δύο κλάσεις, εξασφαλίζοντας έτσι ότι για νέα δεδομένα που μπορεί να βρεθούν κοντά στο περιθώριο θα ταξινομηθούν σωστά.

Για παράδειγμα τα 5 νέα δεδομένα της ροζ κλάσης (παριστάνονται με μαύρο x) που φαίνονται στα παρακάτω διαγράμματα, με το batch perceptron θα είχαν ταξινομηθεί λανθασμένα, παρόλο που τα 4 από αυτά δε βρίσκονται καν μέσα στο περιθώριο. Ενώ με τον svm ταξινομούνται σωστά στη ροζ κλάση.



Άσκηση 3

Στην άσκηση αυτή θα επεξεργαστούμε το wine dataset. Τα δεδομένα αυτά αποτελούν τα αποτελέσματα μιας χημικής ανάλυσης κρασιών από τρεις διαφορετικές καλλιεργητικές ποικιλίες {c1,c2,c3}, και περιλαμβάνουν τιμές για 13 χημικά συστατικά που μετρήθηκαν σε κάθε κρασί. Η πρώτη στήλη του αρχείου των δεδομένων περιλαμβάνει την ετικέτα της ποικιλίας του κάθε κρασιού και οι επόμενες στήλες τις τιμές των συστατικών που μετρήθηκαν. Στόχος μας είναι να διερευνήσουμε το πρόβλημα της πρόβλεψης της ποικιλίας από τα αποτελέσματα της χημική ανάλυσης.

Α) Θεωρείστε το υποσύνολο των δεδομένων που περιέχει τιμές μόνο για τα 5 πρώτα συστατικά και για τα κρασιά από τις ποικιλίες c2 και c3. Να χωρίσετε το παραπάνω σε σύνολο εκπαίδευσης, επικύρωσης και δοκιμής (training, validation και test sets) με αναλογία 50%, 25% και 25% αντίστοιχα, με τυχαία επιλογή δεδομένων και ίδια αναλογία μεταξύ των κλάσεων σε κάθε σύνολο.

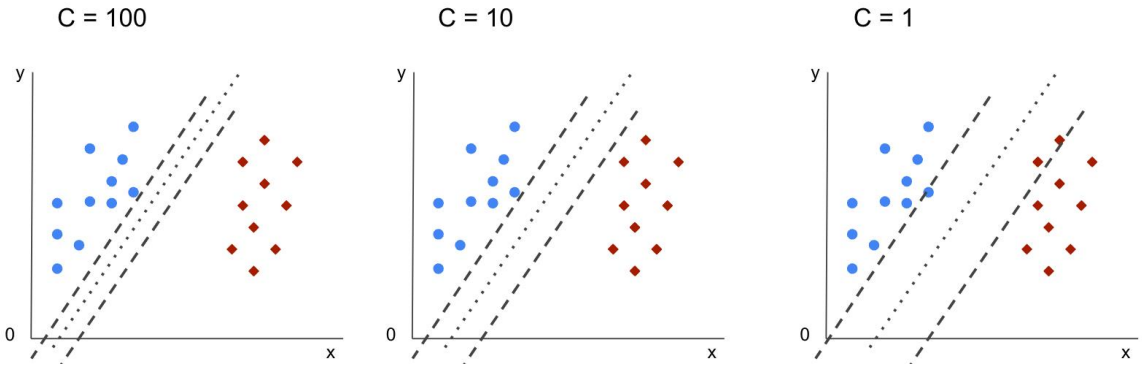
Τα δεδομένα χωρίστηκαν στα τρία υποσύνολα, με τη βοήθεια της συνάρτησης **train_test_split**, της βιβλιοθήκης sklearn. Προσοχή πρέπει να δοθεί στην παράμετρο stratify, με την οποία μπορούμε να διατηρήσουμε την αναλογία των κλάσεων σε κάθε σύνολο. Επίσης, με την παράμετρο random_state, μπορούμε κάθε φορά που τρέχουμε το πρόγραμμα να αναπαράγουμε τα ίδια τυχαία χωρισμένα σύνολα δεδομένων.

Β) Χρησιμοποιήστε γραμμικό SVM για να εκπαιδεύσετε ταξινομητή που να διαχωρίζει την κλάση c2 από τη c3. Χρησιμοποιήστε το validation set για να ρυθμίσετε την παράμετρο C (box constraint) κατάλληλα, και για την καλύτερη τιμή εφαρμόστε τον ταξινομητή που εκπαιδεύσατε στο test set. Τι σφάλμα ταξινόμησης πετύχατε;

Η υπερπαράμετρος C είναι παράμετρος κανονικοποίησης (regularization parameter) και καθορίζει το βάρος που δίνουμε στην ύπαρξη λαθών. Ρυθμίζοντας το C μπορούμε να ελέγξουμε το αν θα υπάρχουν κάποια σημεία τα οποία θα είναι λάθος ταξινομημένα.

Μεγαλύτερο C σημαίνει ότι δεν επιτρέπονται οι λάθος ταξινομήσεις, δίνεται άρα μεγαλύτερη βαρύτητα στην εξάλειψη των λαθών παρά στη μεγιστοποίηση του περιθωρίου. Για αυτό για μεγάλα C το περιθώριο είναι μικρό.

Για πολύ μικρά C δίνεται μεγαλύτερη βαρύτητα στη μεγιστοποίηση του περιθωρίου και ήμαστε πιο ανεκτικοί στα λάθη. Οπότε, έχουμε μεγαλύτερο περιθώριο, αλλά και κάποια λάθος ταξινομημένα σημεία.



Πλέον με την προσθήκη αυτής της σταθεράς η συνάρτηση κόστους του svm γίνεται:

$$J(\underline{w}, w_0, \underline{\xi}) = \frac{1}{2}\|\underline{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

Θέλουμε να μεγιστοποιήσουμε το περιθώριο και να ελαχιστοποιήσουμε τα δείγματα με $\xi_i > 0$.

Τα δείγματα με $\xi_i > 0$ είναι δείγματα που βρίσκονται μέσα στο περιθώριο ή έχουν ταξινομηθεί λανθασμένα.

Αρχικά, εκπαιδεύουμε το μοντέλο χρησιμοποιώντας το σύνολο εκπαίδευσης (training set). Στη συνέχεια, υπολογίζουμε το ποσοστό των προβλέψεων που έχουμε βρει σωστά, δηλαδή την ακρίβεια του ταξινομητή μας, χρησιμοποιώντας το σύνολο επικύρωσης (validation set). Αυτή η διαδικασία επαναλαμβάνεται πολλές φορές για διαφορετικές τιμές της υπερπαραμέτρου C. Τέλος, αποθηκεύουμε την τιμή C για την οποία υπολογίσαμε την υψηλότερη ακρίβεια και εκπαιδεύουμε τον ταξινομητή με αυτήν την τιμή. Η αξιολόγηση πλέον γίνεται στο σύνολο δοκιμής (test set).

Δεν πρέπει ποτέ να χρησιμοποιούμε το σύνολο δοκιμής (test set) για τη ρύθμιση των υπερπαραμέτρων, το χρησιμοποιούμε μόνο για την αξιολόγηση του τελικού μας ταξινομητή. Το σύνολο επικύρωσης (validation set), χρησιμοποιείται ώστε να μπορούμε να παρακολουθούμε την πορεία της εκπαίδευσης και να παρεμβαίνουμε σε αυτήν αν χρειάζεται, δε χρησιμοποιείται για την εκπαίδευση του ταξινομητή (ανανέωση παραμέτρων).

Την καλύτερη ακρίβεια στον σύνολο επικύρωσης μας την έδωσε το C=0.5. Η ακρίβεια με αυτό το C στο σύνολο δοκιμής είναι 93.33%. Οπότε, το **σφάλμα ταξινόμησης** είναι **0.0667**.

Γ) Επαναλάβετε το προηγούμενο για 5 νέους τυχαιοποιημένους διαμερισμούς των δεδομένων, και υπολογίστε τη μέση τιμή και την τυπική απόκλιση του σφάλματος ταξινόμησης στο test set.

Για 5 νέους τυχαιοποιημένους διαμερισμούς των δεδομένων (όχι εντελώς τυχαίοι γιατί ρυθμίζουμε το seed κάθε φορά) έχουμε:

```
Best C value: 0.5, Best accuracy in validation set: 90.0%
Training accuracy: 81.36%
Test accuracy: 93.33%

Best C value: 0.1, Best accuracy in validation set: 86.67%
Training accuracy: 91.53%
Test accuracy: 73.33%

Best C value: 5, Best accuracy in validation set: 86.67%
Training accuracy: 86.44%
Test accuracy: 86.67%

Best C value: 0.01, Best accuracy in validation set: 86.67%
Training accuracy: 67.80%
Test accuracy: 60.00%

Best C value: 0.05, Best accuracy in validation set: 86.67%
Training accuracy: 86.44%
Test accuracy: 86.67%
```

Παρατηρούμε ότι κάθε φορά που τρέχουμε τον κώδικα, οι βέλτιστες τιμές για το C είναι διαφορετικές. Αυτό είναι λογικό, καθώς τα σύνολα εκπαίδευσης, επικύρωσης και δοκιμής είναι διαφορετικά κάθε φορά. Επίσης και η ακρίβεια του ταξινομητή στο σύνολο δοκιμής αλλάζει κάθε φορά.

Οπότε, το πόσο καλός θα είναι ο τελικός ταξινομητής εξαρτάται σε μεγάλο βαθμό από το πώς θα διαχωριστεί το dataset στα επιμέρους σύνολα. Αυτό μπορεί να οφείλεται στο γεγονός ότι δεν έχουμε πάρα πολλά δεδομένα, το dataset είναι μικρό.

Για τα σφάλματα έχουμε: [0.0667, 0.2667, 0.1333, 0.4, 0.1333]

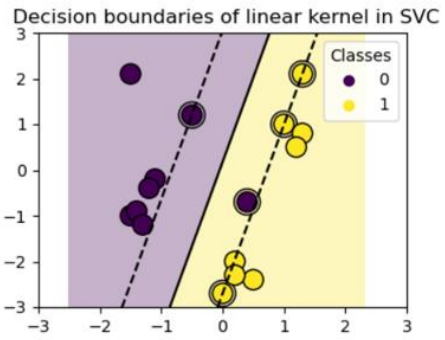
Μέση τιμή σφάλματος: **0.2**, Τυπική απόκλιση σφάλματος: **0.1333**.

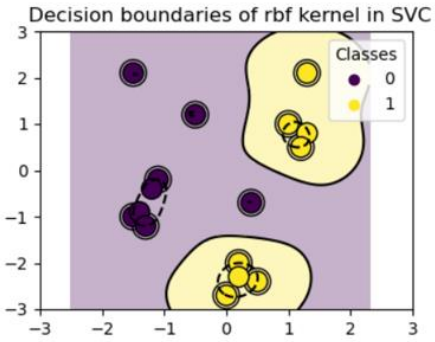
Άρα, για τυχαίους διαμερισμούς του dataset, το σφάλμα θα κυμαίνεται στο εύρος 0.2+0.1333 ή 0.2-0.1333.

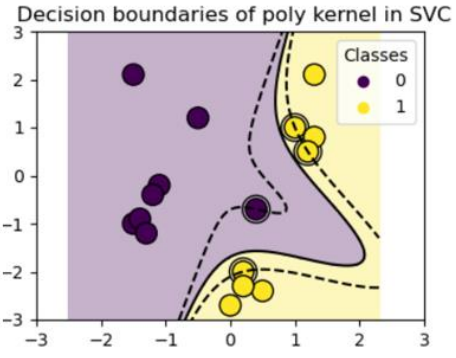
Δ) Επαναλάβετε το Γ για μη-γραμμικό SVM δοκιμάζοντας διάφορες συναρτήσεις πυρήνα (RBF, polynomial κλπ). Τι σφάλμα πετύχατε; Ποιος είναι ο καλύτερος ταξινομητής για το πρόβλημα; Σχολιάστε.

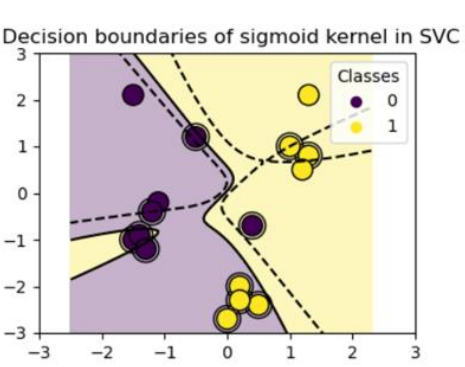
Οι συναρτήσεις πυρήνα που υπάρχουν είναι οι linear, poly, rbf και sigmoid.

Για να καταλάβουμε λίγο πως λειτουργούν χωρίς να αναφέρουμε πολλές λεπτομέρειες μπορούμε να δούμε τις παρακάτω εικόνες.









Τα αποτελέσματα για τις τέσσερεις συναρτήσεις πυρήνα είναι τα ακόλουθα:

```
Kernel linear
Best C value: 0.5, Best accuracy in validation set: 90.0%
Training accuracy: 81.36%
Test accuracy: 93.33%

Best C value: 0.1, Best accuracy in validation set: 86.67%
Training accuracy: 91.53%
Test accuracy: 73.33%

Best C value: 5, Best accuracy in validation set: 86.67%
Training accuracy: 86.44%
Test accuracy: 86.67%

Best C value: 0.01, Best accuracy in validation set: 86.67%
Training accuracy: 67.80%
Test accuracy: 60.00%

Best C value: 0.05, Best accuracy in validation set: 86.67%
Training accuracy: 86.44%
Test accuracy: 86.67%
```

```
Kernel poly
Best C value: 20, Best accuracy in validation set: 93.33%
Training accuracy: 74.58%
Test accuracy: 90.00%

Best C value: 60, Best accuracy in validation set: 90.0%
Training accuracy: 89.83%
Test accuracy: 83.33%

Best C value: 30, Best accuracy in validation set: 83.33%
Training accuracy: 86.44%
Test accuracy: 70.00%

Best C value: 15, Best accuracy in validation set: 86.67%
Training accuracy: 74.58%
Test accuracy: 83.33%

Best C value: 500, Best accuracy in validation set: 93.33%
Training accuracy: 88.14%
Test accuracy: 76.67%
```

```
Kernel rbf
Best C value: 70, Best accuracy in validation set: 86.67%
Training accuracy: 76.27%
Test accuracy: 86.67%

Best C value: 1200, Best accuracy in validation set: 90.0%
Training accuracy: 89.83%
Test accuracy: 83.33%

Best C value: 200, Best accuracy in validation set: 83.33%
Training accuracy: 88.14%
Test accuracy: 76.67%

Best C value: 500, Best accuracy in validation set: 86.67%
Training accuracy: 84.75%
Test accuracy: 96.67%

Best C value: 300, Best accuracy in validation set: 93.33%
Training accuracy: 86.44%
Test accuracy: 83.33%
```

```
Kernel sigmoid
Best C value: 0.001, Best accuracy in validation set: 60.0%
Training accuracy: 59.32%
Test accuracy: 60.00%

Best C value: 20, Best accuracy in validation set: 70.0%
Training accuracy: 66.10%
Test accuracy: 60.00%

Best C value: 0.001, Best accuracy in validation set: 60.0%
Training accuracy: 59.32%
Test accuracy: 60.00%

Best C value: 0.001, Best accuracy in validation set: 60.0%
Training accuracy: 59.32%
Test accuracy: 60.00%

Best C value: 0.001, Best accuracy in validation set: 60.0%
Training accuracy: 59.32%
Test accuracy: 60.00%
```

Για τα σφάλματα έχουμε:

```
For kernel linear
Classification errors: [0.0667, 0.2667, 0.1333, 0.4, 0.1333]
Mean: 0.2
Standard Deviation: 0.1333

For kernel poly
Classification errors: [0.1, 0.1667, 0.3, 0.1667, 0.2333]
Mean: 0.1933
Standard Deviation: 0.076

For kernel rbf
Classification errors: [0.1333, 0.1667, 0.2333, 0.0333, 0.1667]
Mean: 0.1467
Standard Deviation: 0.073

For kernel sigmoid
Classification errors: [0.4, 0.4, 0.4, 0.4, 0.4]
Mean: 0.4
Standard Deviation: 0.0
```

Οπότε, η καλύτερη συνάρτηση πυρήνα επιλέγεται με βάση την ελάχιστη μέση τιμή σφάλματος σε συνδυασμό με μικρή τυπική απόκλιση. Οπότε, ο καλύτερος ταξινομητής είναι ο **rbf**, με μέση τιμή σφάλματος **0.1467** και τυπική απόκλιση **0.073**.

Για C=500 ο **rbf** επιτυγχάνει μέχρι και **96.67%** ακρίβεια σε κάποιους διαμερισμούς του dataset.

Κατά βάση παρατηρούμε ότι η χρήση μη γραμμικών συναρτήσεων πυρήνα (rbf, polynomial) οδηγεί σε μικρότερο σφάλμα. Εξαίρεση αποτελεί το kernel sigmoid, το οποίο φαίνεται ότι

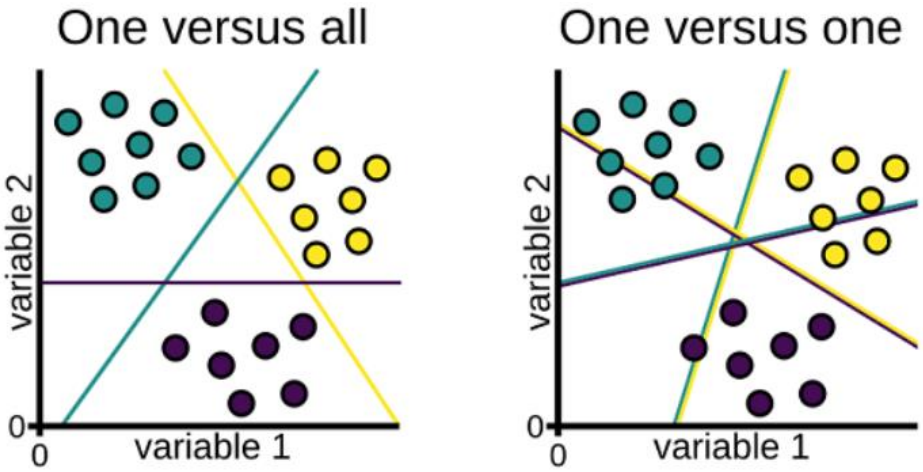
δεν είναι κατάλληλο για την επίλυση του συγκεκριμένου προβλήματος. Η επιλογή του kernel εξαρτάται συχνά από τη φύση του προβλήματος που προσπαθούμε να επιλύσουμε.

Ο rbf kernel δουλεύει καλύτερα όταν τα δεδομένα δεν είναι γραμμικώς διαχωρίσιμα στον χώρο των χαρακτηριστικών. Επιτρέπει στον SVM να δημιουργήσει μια μη γραμμική επιφάνεια απόφασης που μπορεί να διαχωρίσει τα δεδομένα. Ένα τέτοιο πρόβλημα δεν θα μπορούσε να επιλύσει ο γραμμικός kernel. Όταν τα δεδομένα είναι γραμμικώς διαχωρίσιμα προτιμάμε τον γραμμικό kernel, διαφορετικά αν χρησιμοποιήσουμε κάποιον μη γραμμικό kernel, αν και μπορεί να μας επιλύσει το πρόβλημα κινδυνεύουμε να οδηγηθούμε σε overfitting.

Στο συγκεκριμένο πρόβλημα, εφόσον ο rbf kernel τα πηγαίνει αρκετά καλύτερα από τον linear kernel, συμπεραίνουμε ότι τα δεδομένα μας είναι σε κάποιο βαθμό μη γραμμικώς διαχωρίσιμα.

Ε) Χρησιμοποιήστε γραμμικό SVM για να εκπαιδεύσετε ταξινομητές για το πλήρες πρόβλημα των 3 κλάσεων, με την προσέγγιση ένας-εναντίον-ενός (one-vs-one) και καταμέτρηση ψήφων. Μπορείτε να αξιοποιήσετε τις σχετικές αυτοματοποιημένες λειτουργίες που έχουν κάποιες βιβλιοθήκες SVM. Θέτοντας το C=1 και ακολουθώντας πρωτόκολλο 5-fold cross validation, να υπολογίστε τη μέση τιμή του σφάλματος ταξινόμησης χρησιμοποιώντας α) τα 5 πρώτα χαρακτηριστικά όπως και παραπάνω, και β) όλα τα διαθέσιμα χαρακτηριστικά. Σχολιάστε τα αποτελέσματα. Υπολογίστε για κάθε περίπτωση τον πίνακα σύγχυσης (confusion matrix) της ταξινόμησης και σχολιάστε ποιες κλάσεις ομοιάζουν περισσότερο.

Ένας ταξινομητής svm μπορεί να διαχωρίσει δύο κλάσεις. Για το διαχωρισμό περισσότερων από δύο κλάσεων μπορούμε να εφαρμόσουμε πολλούς svm ταξινομητές χρησιμοποιώντας τις προσεγγίσεις one versus all ή one versus one που φαίνονται στην παρακάτω εικόνα.

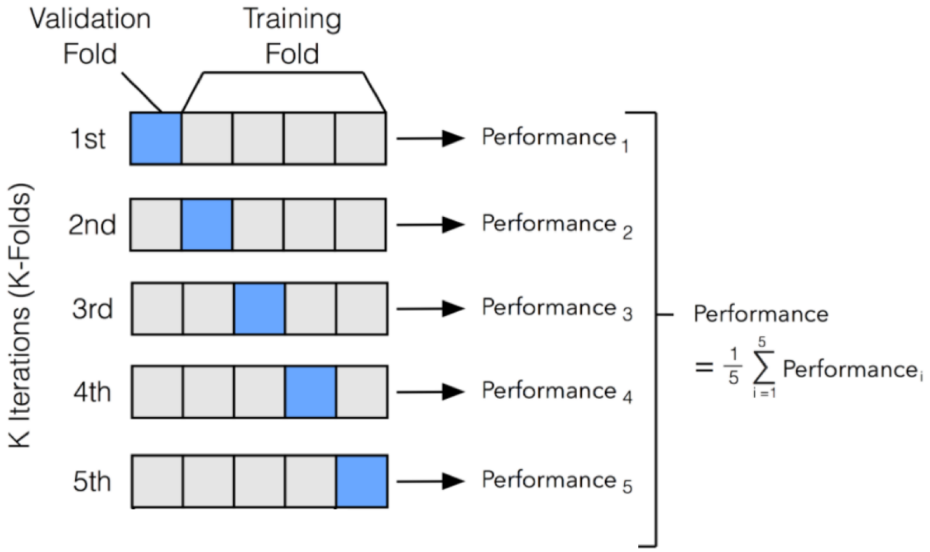


Στην προσέγγιση one-vs-all, έχουμε τόσους svm ταξινομητές όσες και οι κλάσεις και καθένας διαχωρίζει την κάθε κλάση από όλες τις υπόλοιπες.

Στην προσέγγιση one-vs-one, έχουμε $\frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$ ταξινομητές. Αυτό συμβαίνει επειδή κάθε ζεύγος κλάσεων σχηματίζει ένα πρόβλημα δυαδικής ταξινόμησης και ο αριθμός των τρόπων επιλογής 2 κλάσεων από n κλάσεις δίνεται από την παραπάνω σχέση. Στη συνέχεια, για να αποφασιστεί σε ποια κλάση θα ταξινομηθούν τα σημεία

Κάθε δυνατό ζεύγος κλάσεων έχει έναν δικό του ταξινομητή και κατά την πρόβλεψη, καθένας από αυτούς ψηφίζει για την κλάση που πιστεύει ότι ανήκει το δείγμα. Έπειτα, η τελική κλάση επιλέγεται με βάση την κλάση που έχει λάβει τις περισσότερες ψήφους από όλους τους δυαδικούς ταξινομητές. Αν υπάρχει ισοπαλία, μπορεί να χρησιμοποιηθούν πρόσθετοι κανόνες, όπως η ανάθεση της κλάσης που έχει το υψηλότερο σκορ από τους ταξινομητές.

Το K-fold cross-validation είναι μια τεχνική εκτίμησης της απόδοσης ενός ταξινομητή. Τα δεδομένα διασπώνται σε K ομάδες και ο ταξινομητής εκπαιδεύεται K φορές. Σε κάθε βήμα, η μία ομάδα χρησιμοποιείται ως σύνολο επικύρωσης (validation set), ενώ οι υπόλοιπες χρησιμοποιούνται ως σύνολο εκπαίδευσης (training set). Ο ταξινομητής αξιολογείται σε κάθε ομάδα ελέγχου, καταγράφονται οι μετρικές απόδοσης και υπολογίζεται ο μέσος όρος. Αυτή η διαδικασία βοηθά στην αξιολόγηση της γενίκευσης του ταξινομητή, χωρίς να εξαρτάται από ένα συγκεκριμένο διαχωρισμό των δεδομένων. Η διαδικασία που περιγράφηκε φαίνεται στην παρακάτω εικόνα.



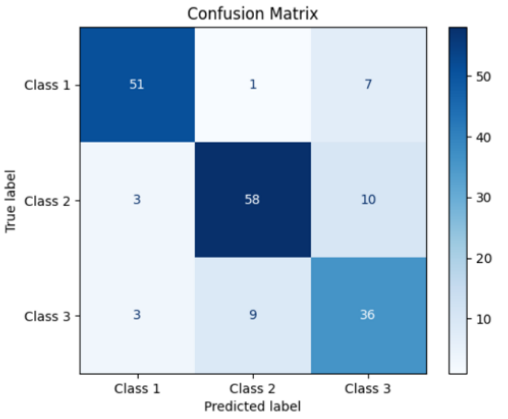
Ο πίνακας σύγχυσης (confusion matrix) είναι ένα εργαλείο που χρησιμοποιείται για την αξιολόγηση της απόδοσης ενός ταξινομητή. Περιγράφει τον αριθμό των σωστών και των λανθασμένων προβλέψεων για κάθε κλάση. Στην κύρια διαγώνιο του πίνακα υπάρχουν τα στοιχεία που έχουν ταξινομηθεί σωστά, οπότε ιδανικά θα θέλαμε ο πίνακας σύγχυσης να περιέχει μη μηδενικές τιμές μόνο στην κύρια διαγώνιο.

Παρακάτω φαίνονται τα αποτελέσματα.

A) Χρήση μόνο 5 πρώτων χαρακτηριστικών

Mean classification error is 0.1852

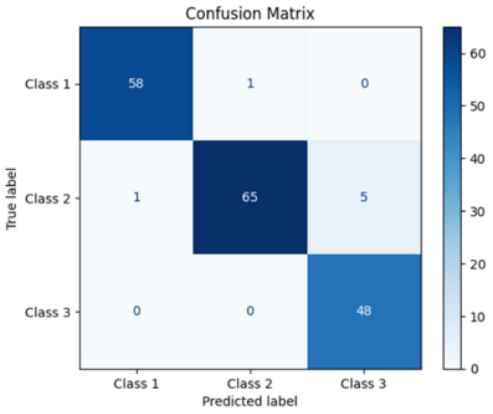
Mean classification accuracy is 81.48%



B) Χρήση όλων των χαρακτηριστικών

Mean classification error is 0.0389

Mean classification accuracy is 96.11%



Παρατηρούμε ότι με τη χρήση όλων το χαρακτηριστικών η μέση τιμή του σφάλματος ταξινόμησης μειώθηκε αρκετά. Από **18.52%** μειώθηκε στο **3.89%**. Οπότε, η χρήση περισσότερων χαρακτηριστικών συνέβαλε καθοριστικά στην μείωση των λανθασμένων ταξινομήσεων. Από 33 λανθασμένες ταξινομήσεις, πλέον έχουμε μόνο 7 λανθασμένες.

Οι λάθος προβλέψεις μπορεί να οφείλονται στο γεγονός ότι κάποιες κλάσεις μοιάζουν αρκετά μεταξύ τους, με αποτέλεσμα ο ταξινομητής να ταξινομεί κάποια δείγματα λανθασμένα.

Τα περισσότερα δείγματα ταξινομούνται λανθασμένα μεταξύ των κλάσεων 2 και 3. Στον 2^{ο} πίνακα 5 δείγματα που ανήκουν στην κλάση 2, ταξινομούνται λανθασμένα στην κλάση 3. Στον 1^{ο} πίνακα συνολικά από τις 33 λανθασμένες ταξινομήσεις οι 19 (πάνω από τις μισές) είναι μεταξύ των κλάσεων 2 και 3. Άρα οι κλάσεις 2 και 3 μοιάζουν πολύ μεταξύ τους. Όταν προστέθηκαν τα επιπλέον χαρακτηριστικά, η ομοιότητα μειώθηκε λίγο (λιγότερα λάθη), όμως ακόμα υπάρχει σε κάποιο βαθμό.

Τα υπόλοιπα λάθη που υπάρχουν στον 1^{ο} πίνακα οφείλονται κυρίως στο γεγονός ότι, με τα 5 χαρακτηριστικά φαίνεται και κάποιες άλλες κλάσεις να μοιάζουν πολύ. Για παράδειγμα, πολλά δείγματα της κλάσης 1 ταξινομούνται στην κλάση 3 (συνολικά 10 λάθη για τις κλάσεις 1 και 3). Στον 2^{ο} πίνακα δεν υπάρχουν δείγματα που να ταξινομούνται λάθος μεταξύ των κλάσεων 1 και 3, οπότε αυτό σημαίνει ότι τα επιπλέον χαρακτηριστικά συνέβαλαν καθοριστικά στο διαχωρισμό των δύο αυτών κλάσεων.

Τέλος, τα 2 δείγματα που ταξινομούνται λανθασμένα μεταξύ των κλάσεων 1 και 2 στον 2^{ο} πίνακα μπορεί να οφείλονται στο γεγονός ότι αυτά τα δείγματα ξεφεύγουν λίγο (είναι λίγο διαφορετικά) από την κλάση στην οποία ανήκουν, τέτοια δείγματα ονομάζονται outliers και δεν είναι αντιπροσωπευτικά της κλάσης όπου ανήκουν.