ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Ένα Ομαδικό Σχήμα Μάθησης για την Υποστήριξη Δραστηριοτήτων Πρόβλεψης.

υπό

Κατσακιώρης Παναγιώτης

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ
Κολομβάτσος Κωνσταντίνος
Επίκουρος Καθηγητής

ΛΑΜΙΑ, 2024

UNIVERSITY OF THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE TELECOMMUNICATIONS

An Ensemble Learning Scheme for Supporting Forecasting Activities.

Katsakioris Panagiotis

FINAL THESIS

ADVISOR

KOLOMVATSOS KONSTANTINOS

Assistant Professor

LAMIA, 2024

Υπεύθυνη Δήλωση περί πνευματικών δικαιωμάτων

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις (1), που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1.Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί χωρίς να τα περικλείω σε εισαγωγικά και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.

2.Δέχομαι ότι η αυτολεξεί παράθεση χωρίς εισαγωγικά, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.

3.Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κ.λπ.), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια.

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής. (1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον καθηγητή μου για την ανάθεση του θέματος, για τις πολύτιμες γνώσεις και τον χρόνο που διέθεσε, ώστε να καταστεί δυνατή η διεκπεραίωση της πτυχιακής εργασίας.

Ιδιαίτερες ευχαριστίες εκφράζω και στην οικογένειά μου για την υποστήριξη τους κατά την διάρκεια των φοιτητικών μου χρόνων.

Περίληψη

Τα Ομαδικά Σχήματα Μάθησης χρησιμοποιούνται ευρέως τα τελευταία χρόνια στον κλάδο της μηχανικής μάθησης. Η χρήση και ο συνδυασμός διαφορετικών μοντέλων βελτιστοποιεί μετρικές όπως Accuracy , Precision , Recall , F1 .

Σε αυτή την πτυχιακή αρχικά θα δωθούν κάποιες περιγραφές και αναφορές σε όρους που θα χρησιμοποιοιθούν παρακάτω. Έπειτα θα γίνει μικρή αναφορά στους βασικούς αλγορίθμους μηχανικής μάθησης, έχουν επιλεγεί αλγόριθμοι με σκοπό να καλυφθεί ένα ευρύ φάσμα τρόπων λειτουργίας ώστε να έχουμε καλύτερα αποτελέσματα όταν συνδυαστούν. Ακολουθούν οι αναφορές στα ομαδικά σχήματα και στο κεφάλαιο 4 έχουμε την μεθοδολογία. Αυτή αποτελείται από αναφορά στα Dataset, στην διαδικασία επιλογής του καλύτερου αλγορίθμου και στα αποτελέσματα. Στο τέλος αναγράφεται ο κώδικας και ο επίλογος.

Abstract

Ensemble Learning Schemes are widely used in machine learning. The usage and combination of different machile learning algorithms can produce much better results in terms of Accuracy , Precision , Recall and F1 .

This thesis will initially provide some descriptions and references to terms that will be used below. Then a small reference will be made to the basic machine learning algorithms. The algorithms have been chosen in order to cover a wide range of modes of operation so that we get better results when they are combined. Next, we have the references to the ensemble learning schemes, and in Chapter 4 we have the methodology, which consists of an outline of the datasets, how the algorithms are being selected, and the results. In the end, we have the code and the epilogue.

Κατάλογος Σχημάτων

Λίστα από εξισώσεις

Περιεχόμενα

1 Εισαγωγή

1.1 **Motivation**

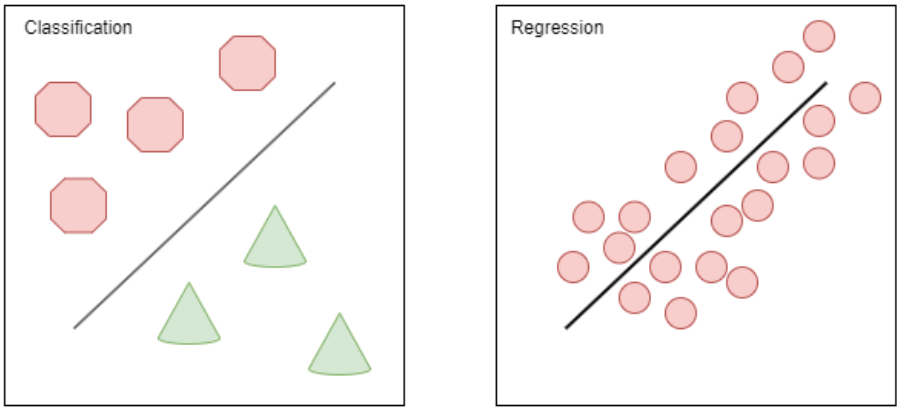
Στην παρούσα πτυχιακή γίνεται μελέτη στα Ομαδικά Σχήματα Μάθησης πάνω σε ιατρικά δεδομένα. Η επιλογή αυτή έγινε γιατί είναι ένας κλάδος που ο συνδυασμός διαφορετικών αλγορίθμων μηχανικής μάθησης μπορεί να αποφέρει σημαντική βελτίωση. Ένα από τα χαρακτηριστικά των ομαδικών σχημάτων είναι η ικανότητα να παράξουν καλά αποτελέσματα με μικρή ποσότητα δεδομένων κάτι που χρειάζεται ιδιαίτερα στα ιατρικά δεδομένα, καθώς δεν μπορούμε να έχουμε πρόσβαση σε μεγάλης ποσότητας δεδομένα διότι η διαδικασία συλλογής και ανωνυμοποίησης είναι ιδιαίτερα μεγάλου κόστους.

1.2 Περιγραφές και Αναφορές

1.2.1 **Classification vs Regression**

Classification είναι μια διαδικασία κατά την οποία αποδίδουμε ετικέτα κατηγοριοποίησης σε ένα παράδειγμα που δεν έχει.[1] Κατά την διάρκεια της εκπαίδευσής τους οι Classification αλγόριθμοι παίρνουν μια σειρά από δεδομένα που τους έχει αποδοθεί είδη ετικέτα, μαθαίνουν απο αυτά τα χαρακτηριστικά ώστε όταν εισαχθεί ένα παράδειγμα χωρίς ετικέτα να καταφέρουν να το κατηγοριοποιήσουν. Μπορούμε να χωρίσουμε το Classification σε 2 είδη, Binary το οποίο κατηγοροποιεί σε 0 ή 1 σε παραδείγματα όπως τα ιατρικά δεδομένα που θα αποφανθεί αν έχει ή όχι κάποιος μία ασθένεια. Δεύτερον, σε MultiClass στο οποίο οι αλγόριθμοι καλούνται να κατηγοροποιήσουν τα δεδομένα σε περισσότερες των 2 πιθανών ετικετών κατηγοροποίησης. Αντίθετα, Regression είναι μια διαδικασία κατά την οποία αποδίδεται μία πραγματική τιμή σε ένα παράδειγμα που δεν έχει είδη. Οι Regression αλγόριθμοι καλούνται να προβλέψουν τις συνεχείς τιμές μίας εξαρτώμενης μεταβλητής με βάση μία σειρά

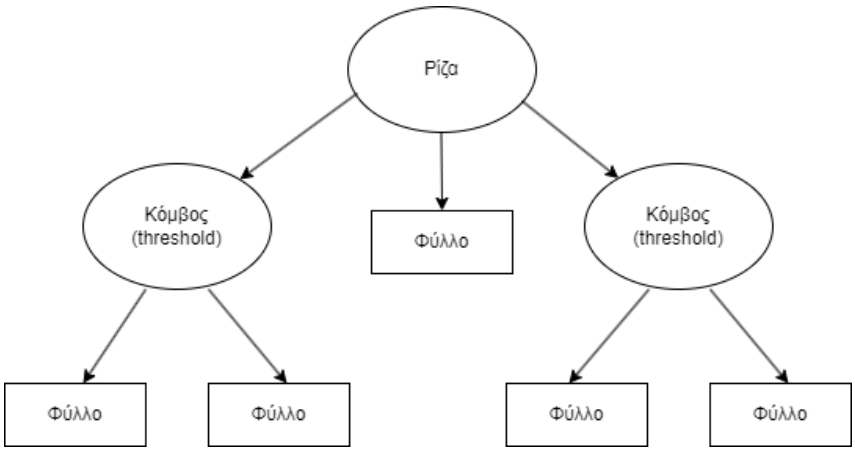
ανεξάρτητων μεταβλητών στις οποίες έχει πρόσβαση με τα δεδομένα εκπαίδευσης.



Σχήμα 1.1: *Classification vs Regression*

1.2.2 **Decision Trees**

Τα Decision Trees είναι άκυκλοι γράφοι σε δενδρική μορφή που χρησιμοποιούνται ώστε να παίρνουμε αποφάσεις.[2] Η κορυφή του δένδρου ονομάζεται ρίζα και οι άκρες που αποτελούν πρακτικά τις ετικέτες κατηγοριοποίησης, φύλλα. Για την κατασκευή του δένδρου ακολουθείται μια συνεχής διαδικασία κατά την οποία το δένδρο βρίσκει την κατάλληλη τιμή στην οποία θα χωριστεί και την αποθηκεύει στον κόμβο. Αυτή η τιμή ονομάζεται threshold και ανάλογα αν το δεδομένο που ελέγχουμε είναι μεγαλύτερο ή μικρότερο της τιμής αυτής ακολουθεί ανάλογη πορεία στην διάρκεια της κατηγοριοποίησης. Η αλγοριθμική πολυπλοκότητα των Decision Trees είναι ανάλογη του μεγέθους των δεδομένων και της ποσότητας των χαρακτηριστικών τους.



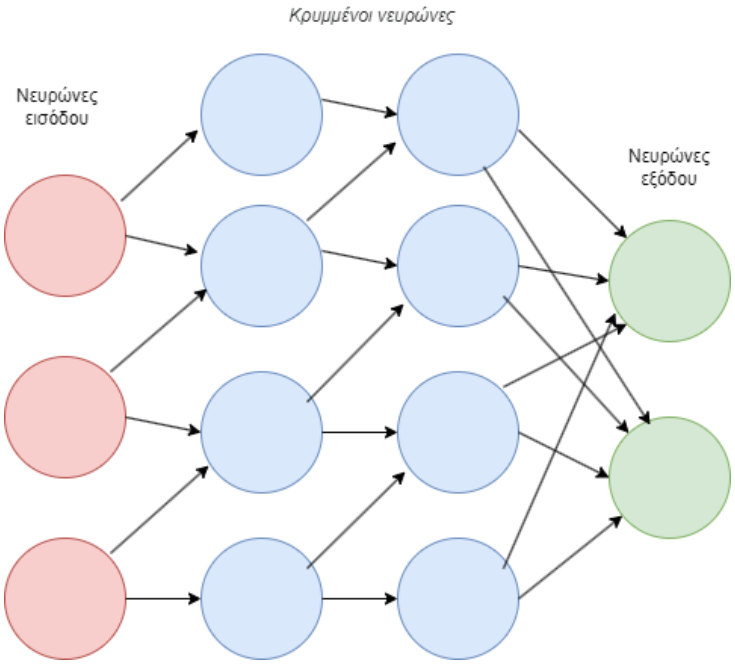
Σχήμα 1.2: *Decision Tree*

1.2.3 **Artificial Neural Network**

Ένα τεχνητό νευρωνικό δίκτυο [3] είναι ένα αλγοριθμικό κατασκεύασμα που βασίζεται μερικώς στα βιολογικά νευρωνικά δίκτυα. Αποτελούνται απο νευρώνες εισόδου, μία η περισσότερες σειρές απο κρυμμένους νευρώνες και νευρώνες εξόδου. Σκοπός των νευρώνων εισόδου είναι απλά η μεταφορά των τιμών εισόδου στους κρυμένους νευρώνες όπου αυτοί με την σειρά πολλαπλασιάζουν κάθε είσοδο με το αντίστοιχο συναπτικό βάρος και υπολογίζουν το ολικό άθροισμα των γινομένων. Το άθροισμα αυτό τροφοδοτείται ως όρισμα στη συνάρτηση ενεργοποίησης, την οποία υλοποιεί εσωτερικά κάθε κόμβος. Η τιμή που λαμβάνει η συνάρτηση για το εν λόγω όρισμα είναι και η έξοδος του νευρώνα για τις τρέχουσες εισόδους και βάρη. Εάν xk_i είναι η i-οστή είσοδος του k νερώνα , wk_i το i-οστό συναπτικό βάρος και φ η συνάρτηση ενεργοποίησης του νευρωνικού δικτύου, τότε η έξοδος y_k του k νευρώνα δίνεται από την εξίσωση :

$$\boxed{y_k = \varphi(\Sigma_{i=0}^{N} x_k i w_k i} \tag{1.1}$$

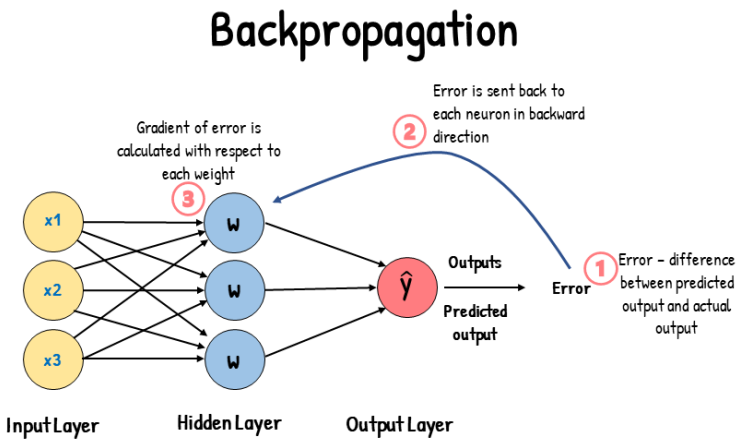
Συνάρτηση ενεργοποίησης



Σχήμα 1.3: *Artificial Neural Network*

1.2.4 **Backpropagation**

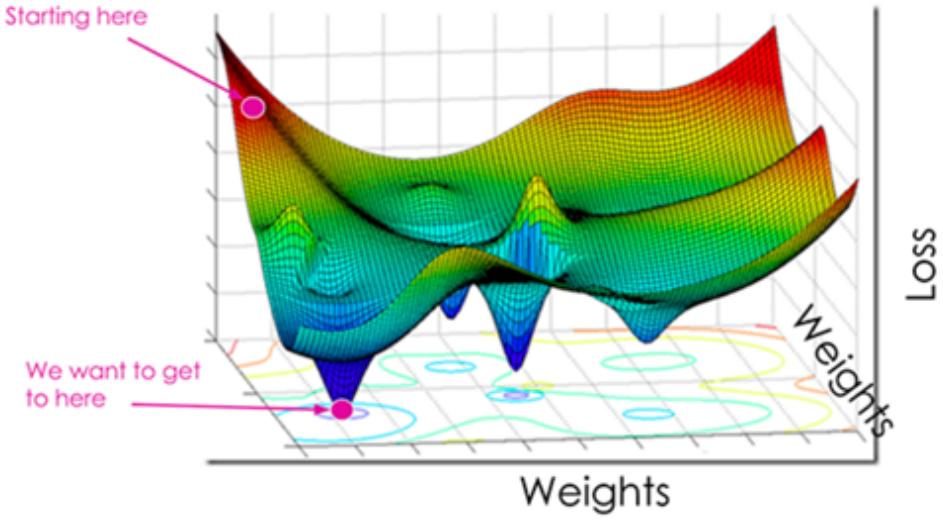
Backpropagation ή αλγόριθμος οπισθοδιάδοσης είναι ένας αλγόριθμος ο οποίος υπολογίζει τους παραγώγους της εξίσωσης κόστους σε αναλογία με τα βάρη με σκοπό τη βελτιστοποίηση του νευρωνικού δικτύου.[4] Αυτό επιτυγχάνεται υπολογίζοντας και αλλάζοντας οπισθοδρομικά τα βάρη. Για τον υπολογισμό χρησιμοποιεί τον κανόνα αλυσίδας, επιλέγει ένα βάρος που θα δεχθεί τις αλλαγές και τα υπόλοιπα θεωρούνται ως στατικές τιμές.



Σχήμα 1.4: *Backpropagation* [4]

1.2.5 **Gradient descent**

Gradient descent[4] είναι ένας αλγόριθμος βελτιστοποίησης με τον οποίο βρίσκουμε την τοπικά ελάχιστη τιμή μίας κυρτής συνάρτησης κόστους. Αρχικά θέτουμε τις παραμέτρους της συνάρτησης και ο Gradient descent χρησιμοποιεί διαφορικές εξισώσεις για να βρεθεί το ελάχιστο σημείο αυτής της συνάρτησης που είναι και η τιμή την οποία προσπαθούμε να βελτιστοποιήσουμε. Χρησιμοποιείται για να βρεθούν οι βέλτιστες τιμές τόσο σε Linear και Logistic Regression όσο και σε Support Vector Machines.



Σχήμα 1.5: *Gradient descent*
[4]

1.2.6 **Bayes theorem**

Το θεώρημα Μπέυζ [5] είναι ένας στατιστικός κανόνας ο οποίος συσχετίζει την τρέχουσα πιθανότητα με την αρχική πιθανότητα και δίνεται απο τον τύπο :

$$\boxed{P(A|B) = \frac{P(B|A) * P(A)}{|P(B)}} \tag{1.2}$$

Bayes theorem

Όπου :

• P(A) και P(B) είναι οι δύο ανεξάρτητες πιθανότητες των A και B.

• P(A|B), η υπό συνθήκη πιθανότητα, είναι η πιθανότητα του A δεδομένου του B να είναι αληθής.

• P(B|A), είναι η πιθανότητα του B δεδομένου του A να είναι αληθής.

1.2.7 **Naïve Bayes Classifiers**

Οι Naïve Bayes Classifiers είναι ομάδα «πιθανολογικών ταξινομητών» που έχουν ως βάση το θεώτημα του Bayes. Το "naive" στο όνομα τους υπάρχει καθώς αφελώς υποθέτουν υπό όρους ανεξαρτησία μεταξύ των δεδομένων. Παρά την αφέλειά τους οι Naïve Bayes Classifiers

έχουν δουλέψει αρκετά καλά σε πραγματικές συνθήκες καθώς χρειάζονται λίγα δεδομένα για να είναι αποδοτικοί και ελάχιστους πόρους απο το σύστημα. Ο στόχος των Naïve Bayes Classifiers είναι να υπολογίσουν την υπό όρους πιθανότητα για κάθε μεταβλητή k ή κλάση C_k και δίνεται από τον τύπο [6]:

$$\boxed{p(C_k) = \Pi_{i=1}^{n} p(X_i|C_k)} \tag{1.3}$$

Naïve Bayes Classifier formula

1.2.8 Ridge regression

Ridge regression [7] είναι μία μέθοδος για να υπολογιστούν οι συντελεστές από πολλαπλά μοντέλα παλινδρόμησης όταν τα δεδομένα έχουν μεγάλη συσχέτιση. Ο τύπος υπολογισμού του Ridge estimator δίνεται από τον τύπο :

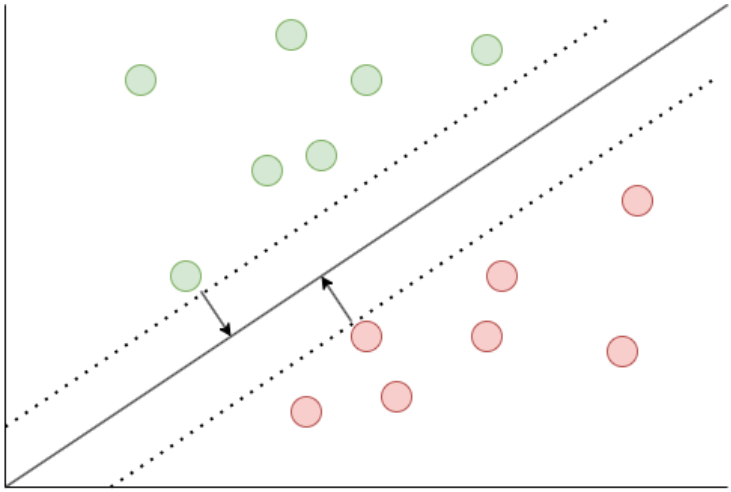
$$\boxed{\widehat{\beta}_R = (X^T X + \lambda I)^{-1} X^T y} \tag{1.4}$$

Ridge estimator

Όπου $X^T X$ ένας τετράγωνος πίνακας, y η εξαρτημένη μεταβλητή της παλινδρόμησης, I ο ταυτοτικός πίνακας και λ (λ>0) η παράμετρος Ridge ως εναλλάκτης της διαγωνίου του πίνακα X.

1.2.9 Support Vector Machine Algorithms

Τα Support Vector Machines[8] είναι μια ομάδα αλγορίθμων η οποία είναι ιδιαίτερα χρήσιμη με δεδομένα τα οποία έχουν πολλές διαστάσεις. Σκοπός τους είναι να ορίσουν ένα χώρο μέσω ενός ορίου με το οποίο θα μπορούν να ομαδοποιήσουν τα δεδομένα με βέλτιστο τρόπο. Το όριο πρέπει να έχει την μέγιστη δυνατή απόσταση από τα δεδομένα εκπαίδευσης.



Σχήμα 1.6: *Support Vector Machine*

1.2.10 **Overfitting**

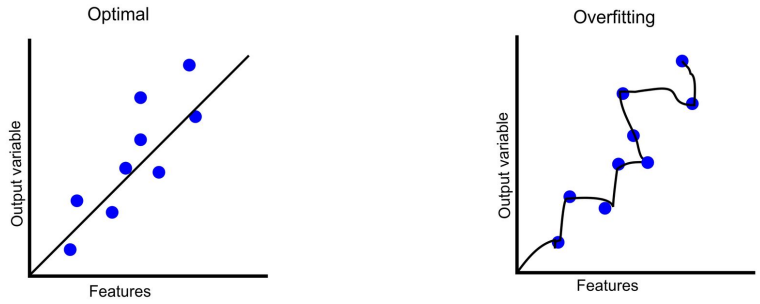
Για κάθε αλγόριθμο μηχανικής μάθησης χρησιμοποιείται ένα μεγάλο κομμάτι των δεδομένων για να εκπαιδευθεί και έπειτα να προβλέψει τις ανάλογες τιμές. Overfitting είναι όταν ο αλγόριθμος έχει προσαρμοστεί έντονα στα δεδομένα εκπαίδευσης με αποτέλεσμα να είναι αποδοτικός μόνο σε αυτά και όταν παρουσιαστούν νέα δεδομένα να έχει χαμηλότερη απόδοση. Αυτό γίνεται συνήθως για ένα από τους παρακάτω λόγους:

• Τα δεδομένα είναι πολύ λίγα και δεν αρκούν για να δημιουργήσει ο αλγόριθμος μια γενικευμένη εικόνα.

• Τα δεδομένα περιέχουν πολλά στοιχεία που δεν συσχετίζωνται με την τιμή που πρέπει να προβλεφθεί.

• Η πολυπλοκότητα στα δεδομένα είναι έντονη, έτσι ο αλγόριθμος αφομοιώνει και τιμές οι οποίες είναι πρακτικά θόρυβος.

Ο τρόπος που αντιλαμβανόμαστε αν έχουμε Overfitting είναι να απομονώνουμε ένα κομμάτι των δεδομένων πριν την εκπαίδευση και να ελέγχουμε βάση αυτών τις τελικές τιμές πρόβλεψης και απόδοσης του αλγορίθμου.



Σχήμα 1.7: *Overfitting Example*
[9]

1.2.11 **Mean squared error**

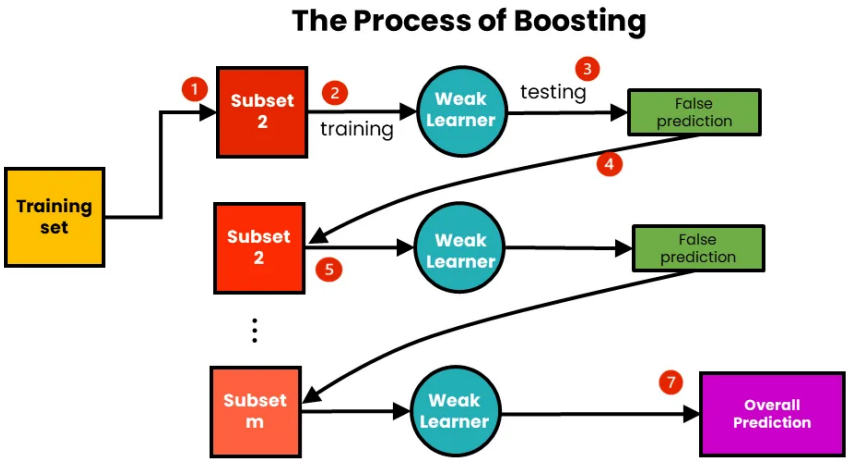
Το Mean squared error (MSE) [10] μετράει το ποσοστό των σφαλμάτων σε ένα στατιστικό μοντέλο. Είναι η τετραγωνισμένη διαφορά ανάμεσα στην τιμή που προβλέψαμε και στην πραγματική τιμή. Όταν το μοντέλο δεν έχει σφάλματα το MSE είναι μηδέν και αυξάνεται ανάλογα όταν τα σφάλματα αυξάνονται. Ο τύπος που μας δίνει το MSE είναι:

$$\boxed{MSE = \frac{1}{n}\Sigma_{i=1}^{n}(Yi - \widehat{Yi})^2} \tag{1.5}$$

MSE

1.2.12 **Boosting**

Boosting είναι μία μέθοδος η οποία χρησιμοποιεί ένα σύνολο απο φαινομενικά αδύναμους αλγορίθμους ώστε να δημιουργηθεί ένας καλύτερος. Η διαφορά με άλλους τρόπους ομαδικής μάθησης είναι ότι κατα την διάρκεια της εκπαίδευσης δίνεται βάση στα αποτελέσματα τα οποία ο αλγόριθμος έχει προβλέψει λάθος. Σκοπός είναι ο αλγόριθμος να μάθει απο τα λάθη του και να τα μειώσει στην επόμενη φάση της εκπαίδευσης.



Σχήμα 1.8: *Boosting Process*
[11]

1.2.13 **Bootstraping**

Bootstraping [12] είναι μια τεχνική αναδιοργάνωσης των δεδομένων όπου παίρνουμε κομμάτια από υπάρχοντα δεδομένα και δημιουργούμε νέα δεδομένα με σκοπό να μετρήσουμε την μετρική που μας ενδιαφέρει. Αυτό επαναλαμβάνεται αρκετές φορές και έπειτα χρησιμοποιούνται οι τιμές για να δημιουργηθεί μία κατανομή πιθανοτήτων.

1.2.14 **Dataset**

Dataset είναι μία συλλογή δεδομένων στην συγκεκριμένη περίπτωση σε μορφή πίνακα.

2 Αλγόριθμοι Μηχανικής Μάθησης

2.1 **DecisionTreeClassifier**

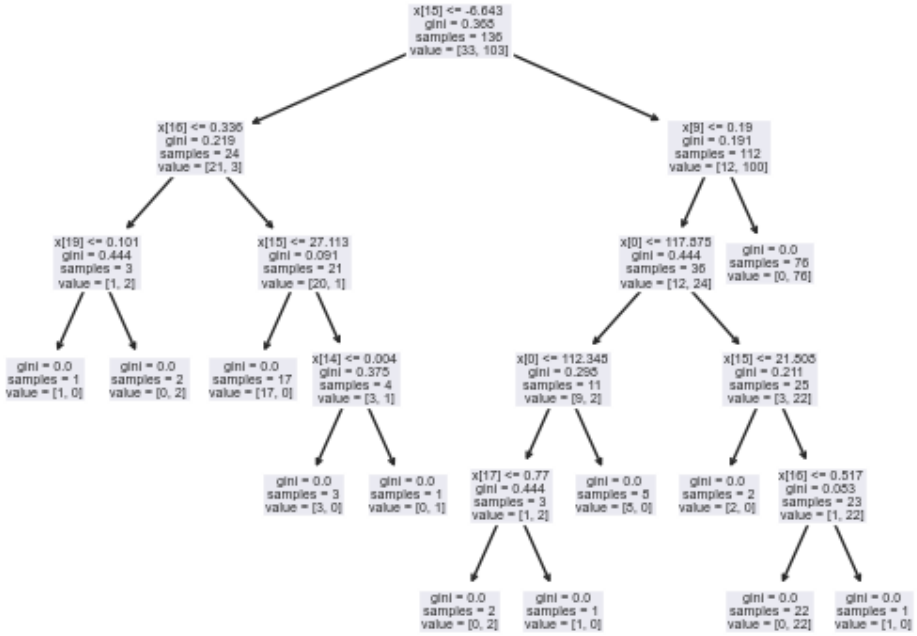
Ο DecisionTreeClassifier [13] είναι ένα απλό παράδειγμα DecisionTree το οποίο αφού εκπαιδευτεί χρησιμοποιείται για Classification. Αρχικά το δέντρο δέχεται ένα σύνολο δεδομένων και διαλέγει το καλύτερο χαρακτηριστικό ώστε να διαχωριστεί το δένδρο κάνοντας αυτό το χαρακτηριστικό κόμβο του δένδρου. Αυτή η διαδικασία συνεχίζεται μέχρι να μην υπάρχουν άλλα χαρακτηριστικά. Η επιλογή στον DecisionTreeClassifier γίνεται μέσα από μία μετρική που ονομάζεται Gini index και δίνεται από τον τύπο :

$$\boxed{Gini(D) = 1 - \Sigma_{i=1}^{m} Pi^2} \tag{2.1}$$

Gini index

Όπου Pi η πιθανότητα ένος υποσυνόλου των δεδομένων να ανήκει στην κλάση Ci. Το χαρακτηριστικό με το μικρότερο Gini index γίνεται το σημείο διαχωρισμού.

Παρακάτω ακολουθεί η δομή του δέντρου που δημιουργήθηκε για το Parkinsons Dataset.



Σχήμα 2.1: *Decision Tree in Parkinsons Dataset*

2.2 **GaussianNB**

Ο GaussianNB [14] είναι ένα παράδειγμα Naïve Bayes Classifier που υποθέτει ότι τα δεδομένα παρέχονται μέσω μίας κανονικής κατανομής Γκάους. Κατά τα άλλα δουλεύει όπως οι υπόλοιποι Naïve Bayes Classifier (1.2.8). Ο τύπος υπολογισμού της πιθανότητας να ανήκει η εκάστοτε τιμή στην κλάση δίνεται από τον τύπο:

$$\boxed{P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(\frac{(x_i - \mu_y)^2}{2\sigma_y^2})} \tag{2.2}$$

Gaussian Naive Bayes algorithm likelihood

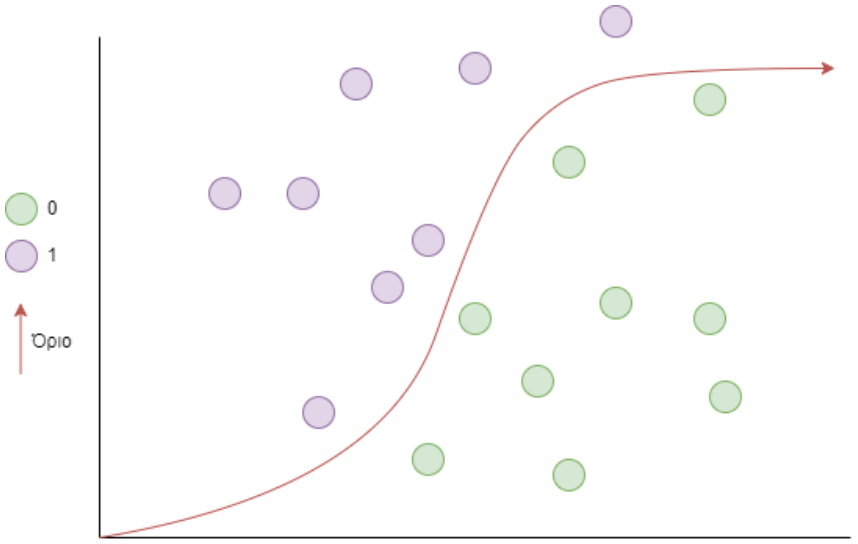
2.3 **SVC**

Ο SVC ή C-Support Vector Classification [15] είναι ένας αλγόριθμος που βασίζεται στο libsvm. Κάνει μία συνεχή βελτίωση ελαχιστοποίησης στα Support Vector Machines που

αρχικά δημιουργούνται.

2.4 LogisticRegression

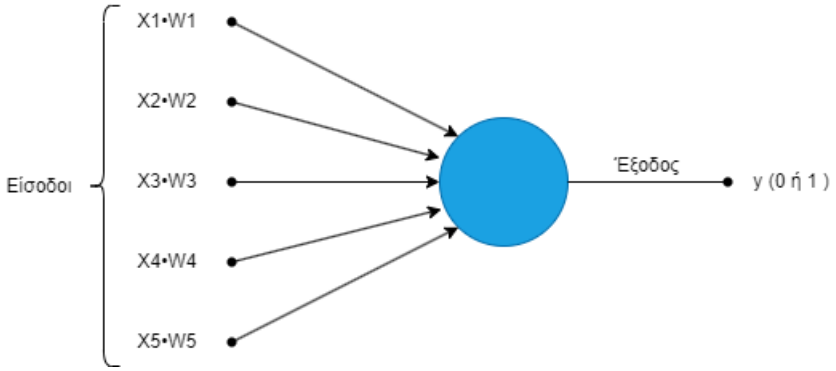
Το πρώτο που πρέπει να αναφερθεί είναι ότι ο LogisticRegression [16] δεν είναι ένας Regression αλγόριθμος αλλά είναι Classification. Πήρε το όνομά του καθώς ο τρόπος λειτουργίας του είναι παρόμοιος με τον Linear regression. Όπως και τα SVM σκοπός του LogisticRegression είναι να ορίσει ένα χώρο μέσω ενός ορίου με το οποίο θα μπορούν να ομαδοποιηθούν τα δεδομένα με βέλτιστο τρόπο. Το κριτήριο βελτιστοποίησης του LogisticRegression ονομάζεται maximum likelihood το οποίο αντί να βελτιστοποιεί το average loss όπως στη Linear regression βελτιστοποιεί την πιθανότητα του εκάστοτε παραδείγματος βάση των υπόλοιπων δεδομένων εκπαίδευσης.



Σχήμα 2.2: *LogisticRegression*

2.5 Perceptron

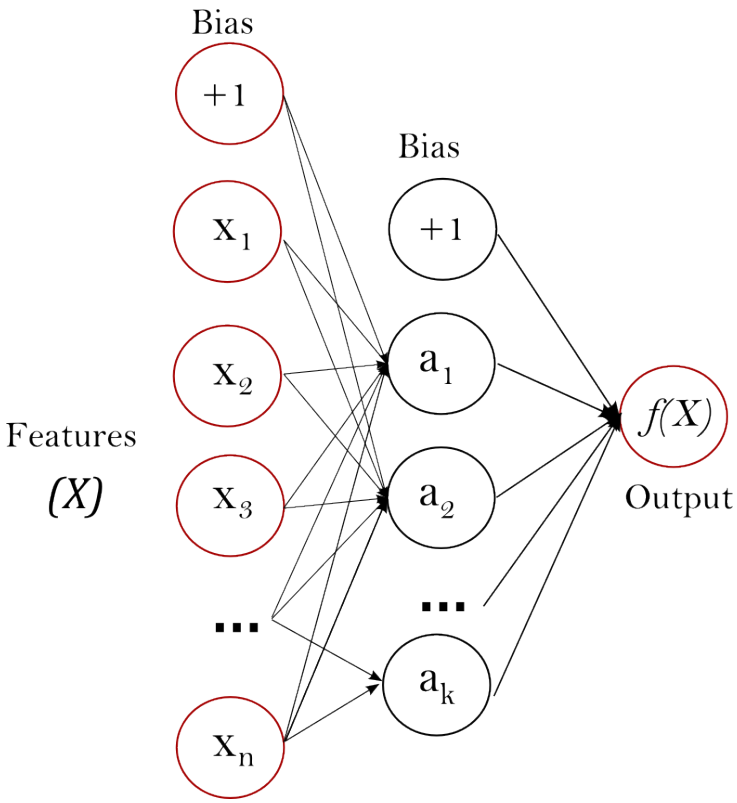
Ο Perceptron [17] είναι ένα νευρωνικό δίκτυο πολλαπλών εισόδων αλλά μίας εξόδου και χρησιμοποείται για κατηγοριοποίηση. Όπως σε όλα τα νευρωνικά οι είσοδοι πολλαπλασιάζονται με τα βάρη και έπειτα προστίθενται και εισέρχονται στην Activation Function ώστε να μετατραπούν σε τιμές ανάμεσα στο 0 και 1 και να κατηγοριοποιηθούν.



Σχήμα 2.3: *Perceptron*

2.6 **MLPClassifier**

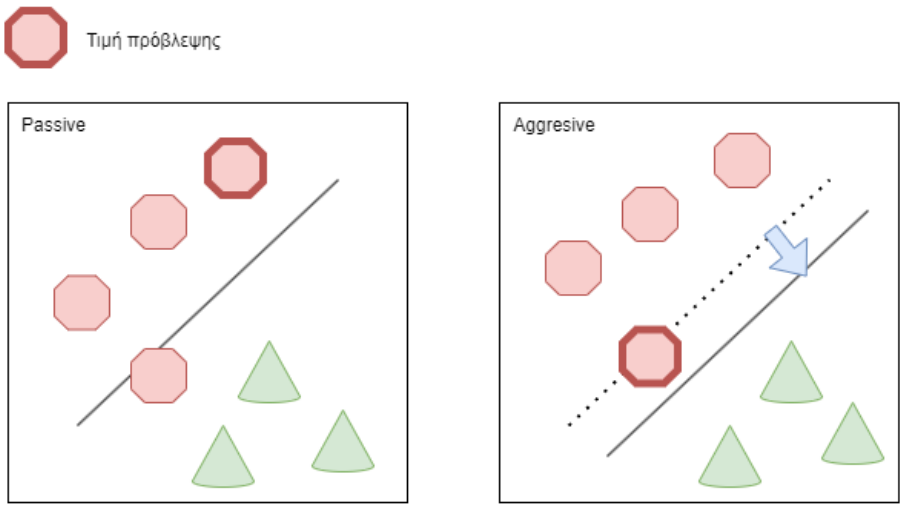
Ο MLPClassifier ή αλλιώς Multi-layer Perceptron classifier [18] δημιουργεί ένα MLP το οποίο είναι είναι ένα feedforward νευρωνικό δίκτυο δηλαδή ένα νευρωνικό δίκτυο με φορά προς μία κατεύθυνση από τους νευρώνες εισόδου προς τους νευρώνες εξόδου. Κατά τη διάρκεια της εκπαίδευσης χρησιμοποιεί backpropagation και 2 πίνακες. Ο πίνακας μεγέθους χ ο οποίος διατηρεί τα δείγματα εκπαίδευσης σαν διανύσματα χαρακτηριστικών κινητής υποδιαστολής και ο πίνακας μεγέθους y ο οποίος διατηρεί τις ετικέτες κλάσεων για τα δείγματα εκπαίδευσης.



Σχήμα 2.4: *One hidden layer MLP*
[18]

2.7 **PassiveAggressiveClassifier**

Ο PassiveAggressiveClassifier [19] σε κάθε επανάληψη ελέγχει τις νέες μετρικές και αλλάζει τα βάρη. Αν η πρόβλεψη είναι σωστή δεν έχουμε καμία αλλαγή σε αντίθετη περίπτωση έχουμε μια σειρά αλλαγών βάση μίας μετρικής c η οποία είναι υπεύθυνη για το πόσο "Aggressive" θα είναι ο αλγόριθμος καθώς μπορεί να δημιουργηθούν προβλήματα όπως overfitting.



Σχήμα 2.5: *Passive Aggressive Classifier example*

2.8 RidgeClassifier

Ο RidgeClassifier [20] είναι ένας αλγόριθμος μηχανικής μάθησης που χρησιμοποιεί τεχνικές ταξινόμησης και την παλινδρόμηση Ridge ώστε να κατηγοριοποιήσει τα δεδομένα. Ο συγκεκριμένος αλγόριθμος μειώνει το overfitting προσθέτοντας έναν όρο ποινής και μία παράμετρο που το διαχειρίζεται. Επιπρόσθετα χρησιμοποιείται μια συνάρτηση απώλειας που υπολογίζει το mean squared loss την οποία διαχειρίζεται η προαναφερθείσα παράμετρος που ονομάζεται Alpha parameter. Όπως και η Ridge Regression ο RidgeClassifier χρησιμοποιεί την κανονικοποίηση L2 όμως το κύριο χαρακτηριστικό του Ridge Classifier είναι ότι μετατρέπει όλες τις τιμές των δεδομένων σε ένα φάσμα απο το -1 έως το 1 ώστε να διαχειριστεί ένα πρόβλημα classification σαν regression. Παρακάτω βρίσκεται η τιμή κόστους της κανονικοποίησης L2:

$$\boxed{\Sigma_{i=1}^{n}(yi - \Sigma_{j=1}^{p} xij\beta j)^2 + \lambda\Sigma_{j=1}^{p} b^2 j} \tag{2.3}$$

L2 Cost function

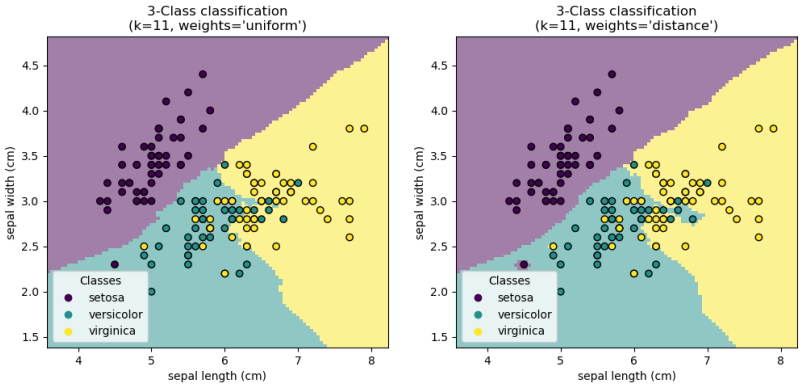
2.9 KneighborsClassifier

Ο KneighborsClassifier [21] είναι ένας μη παραμετρικός αλγόριθμος που σημαίνει ότι σε αντίθεση με άλλους αλγόριθμους που όταν τελειώνει η εκπαίδευση διαγράφουν τα δεδομένα

και κρατούν μόνο το μοντέλο αυτός κρατά τα δεδομένα εκπαίδευσης. Έτσι όταν εισαχθεί μία νέα τιμή ο αλγόριθμος βρίσκει τα 'κ' κοντινότερα σε αυτό παραδείγματα και ανάλογα την πλειοψηφία αυτών αποδίδει την ετικέτα κατηγοριοποίησης στο νέο δεδομένο. Σε κάθε γείτονα αποδίδεται ένα βάρος το οποίο μπορεί να είναι ίδιο για όλους ή να έχει άμεση σχέση με την τιμή της απόστασης από την τιμή. Σε αυτή την πτυχιακή χρησιμοποιήθηκαν ίδια βάρη σε όλους τους γείτονες. Η τιμή απόστασης για τα εκάστοτε στοιχεία βρίσκεται απο τον τύπο της ευκλείδειας απόστασης .

$$\boxed{d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}} \tag{2.4}$$

Euclidean distance



Σχήμα 2.6: *Kneighbors Classifier example* [22]

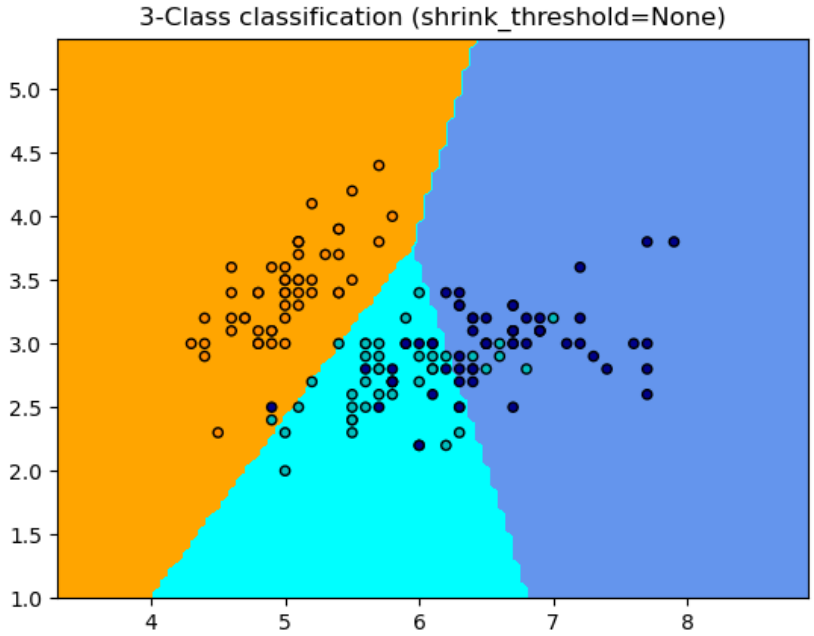
2.10 **NearestCentroid**

Ο NearestCentroid [23] αρχικά χωρίζει γεωμετρικά τα δεδομένα εκπαίδευσης σε χώρους που ονομάζονται centroids και με βάση αυτά κάνει τις απαραίτητες προβλέψεις. Τα centroids είναι γεωμετρικά κέντρα απο το σύνολο των γειτονικών τους δεδομένων. Ο αλγόριθμος δημιουργεί τόσα centroids όσα είναι και οι κλάσεις τις οποίες πρέπει να προβλέψουμε, αν είναι 2 όπως στην παρούσα πτυχιακή τότε δημιουργούνται 2 centroids. Έπειτα όταν έχουμε καινούρια δεδομένα υπολογίζουμε την απόσταση απο τα centroids και κατατάσουμε τα νέα

δεδομένα στο πιο κοντινό. Τόσο για τη δημιουργία των centroids όσο και για την μέτρηση της απόστασης των νέων δεδομένων χρησιμοποιείται ευκλείδια απόσταση.

$$\boxed{d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}} \tag{2.5}$$

Euclidean distance

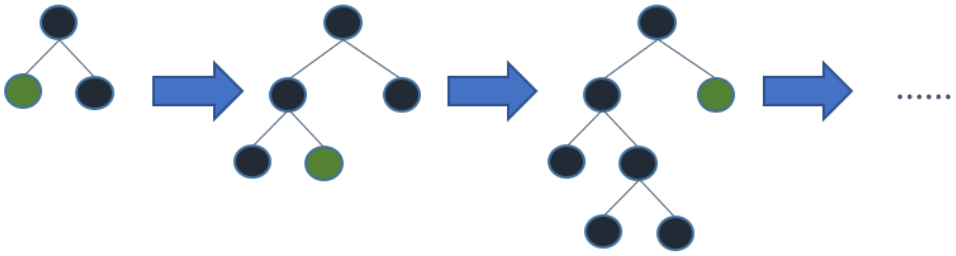


Σχήμα 2.7: *NearestCentroid example*
[22]

3 Ensemble Learning

3.1 LGBMClassifier

Ο LGBMClassifier [24] είναι ένας σχετικά καινούριος gradient boosting αλγόριθμος μηχανικής μάθησης ο οποίος σχεδιάστηκε για να είναι γρήγορος και να καταναλώνει χαμηλούς πόρους. Η διαφορά με τους υπόλοιοπους gradient boosting αλγόριθμους είναι ότι χρησιμοποιεί δέντρα αλλά αντί τα δέντρα να μεγαλώνουν κάθετα μεγαλώνουν οριζόντια. Σε περιπτώσεις που παράγεται όλο το δέντρο έχουμε το ίδιο αποτέλεσμα όμως τις περισσότερες φορές αυτό δεν γίνεται καθώς παράμετροι όπως max depth και περιορισμοί στους κύκλους εκπαίδευσης δεν το επιτρέπουν. Αυτό έχει σαν αποτέλεσμα να δημιουργούνται δέντρα με λιγότερα errors σε λιγότερο χρόνο απο ότι στους υπόλοιπους gradient boosting αλγόριθμους.

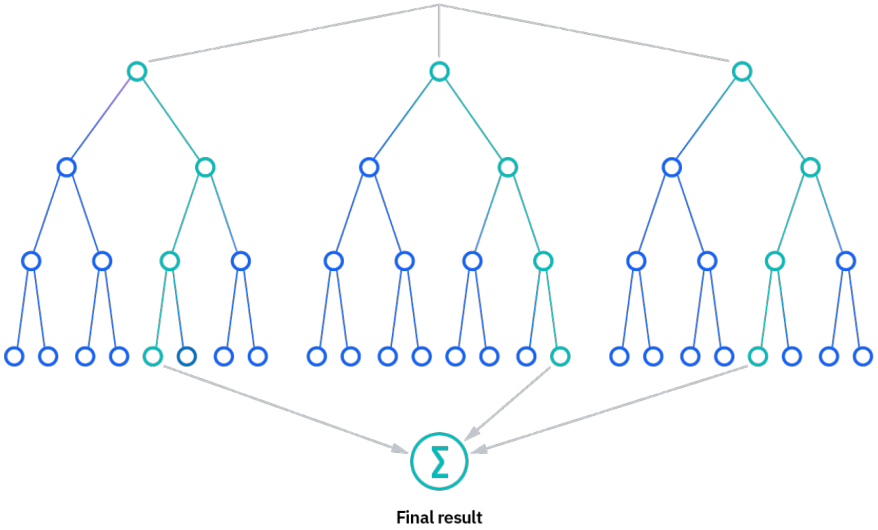


Σχήμα 3.1: *Leaf wise tree growth*
[25]

3.2 RandomForestClassifier

Ο RandomForest [26] είναι ένας αλγόριθμος μηχανικής μάθησης που δημιουργεί δέντρα απο διάφορα κομμάτια των δεδομένων χρησιμοποιώντας bootstraping. Κατά τη διάρκεια δημιουργίας των δέντρων ο αλγόριθμος βρίσκει το καλύτερο σημείο διακλάδωσης χρησιμοποιώντας εξαντλητική αναζήτηση είτε σε όλα τα στοιχεία των δεδομένων, είτε σε ένα ένα τυχαίο κομ-

μάτι αυτών. Παρότι το κάθε δέντρο ξεχωριστα μπορεί να έχει προβλήματα όπως πολλά errors και overfitting παίρνοντας τον μέσο όρο όλων αυτών τα προβλήματα μερικώς λύνονται και είναι αρκετό για να έχουμε αρκετά καλά αποτελέσματα.



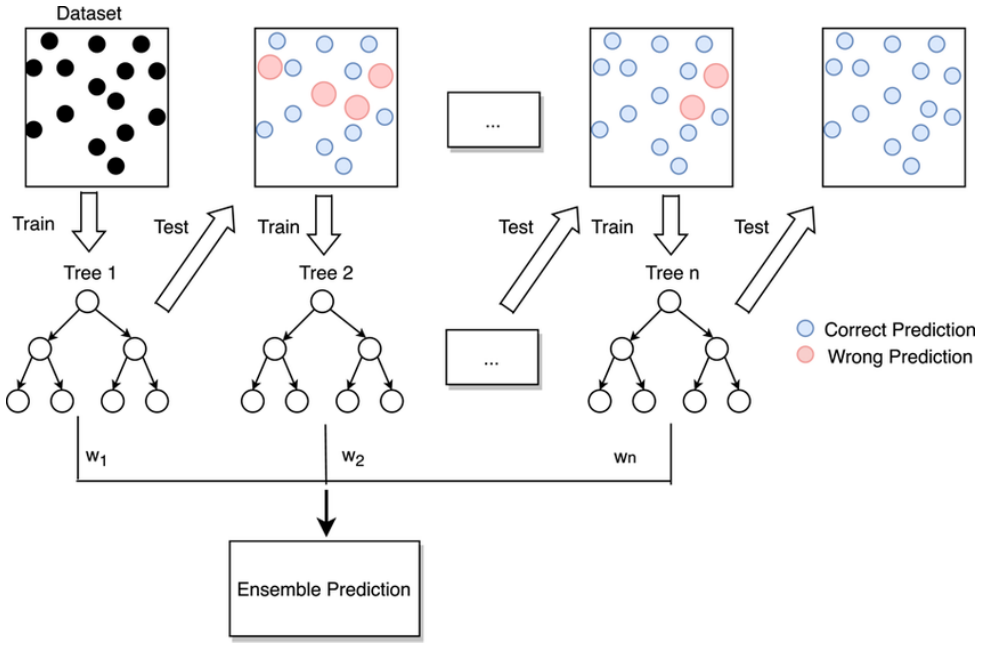
Σχήμα 3.2: *Random Forest tree.*
[26]

3.3 ExtraTreesClassifier

Ο ExtraTreesClassifier [27] είναι ένας αλγόριθμος μηχανικής μάθησης που δημιουργεί δέντρα από τυχαία κομμάτια των δεδομένων και έπειτα χρησιμοποιεί τους μέσους όρους αυτών για να βγάλει τις τελικές τιμές. Η διαφορά με τον RandomForest είναι ότι δεν χρησιμοποιεί bootstraping για την επιλογή δεδομένων και επιπρόσθετα τα σημεία διακλάδωσης στα δέντρα γίνονται τελείως τυχαία χωρίς να υπολογίζονται τα βέλτιστα χρησιμοποιώντας εξαντλητική αναζήτηση.

3.4 GradientBoostingClassifier

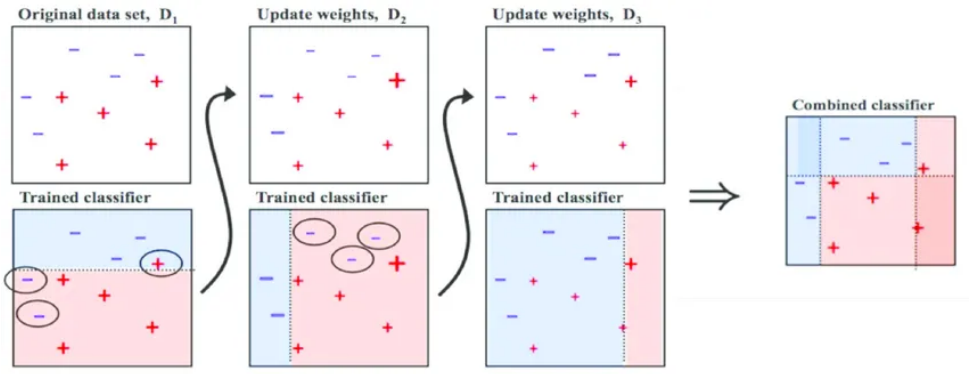
Ο GradientBoostingClassifier [28] είναι ένας αλγόριθμος ο οποίος δημιουργεί ένα μοντέλο με παράλληλα δέντρα βελτιστοποιώντας το σε κάθε επανάληψη χρησιμοποιώντας τις εκάστοτε διαφοροποιήσιμες απώλειες (loss functions). Σε αντίθεση με τον AdaBoost που θα δούμε παρακάτω τα βάρη δεν δέχονται αλλαγές αλλά τα λάθη της μίας εκπαίδευσης χρησιμοποιούνται σαν είσοδος στην επόμενη με σκοπό να αποφευχθούν.



Σχήμα 3.3: *GradientBoosting*
[29]

3.5 **AdaBoostClassifier**

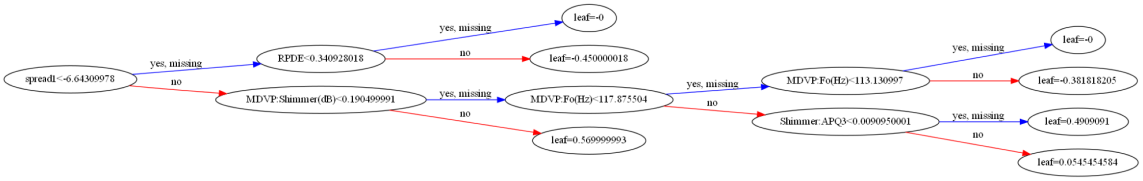
Ο AdaBoostClassifier [30] δημιουργεί παράληλους αλγόριθμους πάνω στα δεδομένα και έπειτα τα βάρη τους ενός δέχονται επεξεργασία ανάλογα με τις τιμές τις οποίες έχει προβλέψει λάθος ο προηγούμενος. Η ιδέα πίσω απο τον AdaBoost είναι να διορθώνονται τα λάθη του επόμενου αλγόριθμου βάση τον προηγούμενο. Στο τέλος γίνεται συμψηφισμός όμως το ποσοστό της ψήφου είναι διαφορετικό συμφωνα με το πόσο καλά αποτελέσματα είχαν καθόλη την διάρκεια της εκπαίδευσης.



Σχήμα 3.4: *AdaBoost*
[31]

3.6 XGBClassifier

Ο XGBoost Classifier [32] είναι ένας extreme gradient boosting αλγόριθμος ο οποίος χρησιμοποιεί παράλληλα δέντρα τα οποία βελτιστοποιεί για να έχει την καλύτερη πιθανή πρόβλεψη. Αρχικά δημιουργείται το πρώτο δέντρο το οποίο συνήθως δεν έχει ιδιαίτερα καλή απόδοση. Έπειτα δημιουργείται ένα δεύτερο δέντρο που σκοπό έχει να προβλέψει ότι το πρώτο δέντρο δεν μπορούσε. Αυτή η διαδικασία συνεχίζεται μέχρι να πληρούνται τα κριτήρια που έχουμε θέσει.



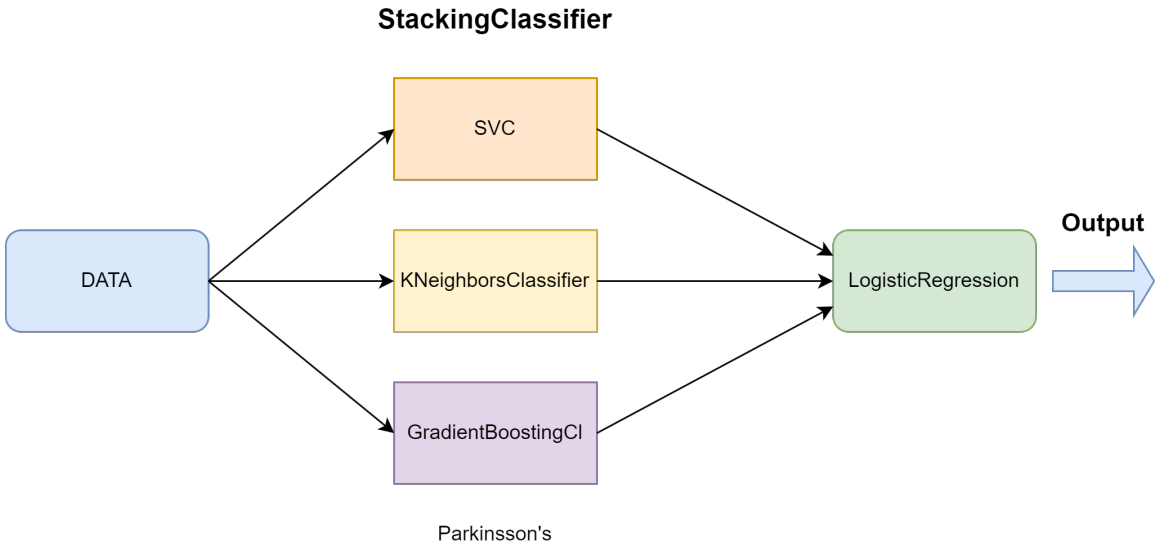
Σχήμα 3.5: *XGBtree in parkinsons dataset.*

3.7 StackingClassifier

Ο Stacking Classifier [33] χρησιμοποιεί τα αποτελέσματα αλγορίθμων και τα συνδυάζει χρησιμοποιώντας τα σαν είσοδο σε έναν άλλο. Η επιλογή των αλγορίθμων είναι ιδιαίτερα σημαντική. Ωστόσο η επιλογή δεν είναι τόσο εύκολη, καθώς δεν βασίζεται στα καλύτερα αποτελέσματα των αλγορίθμων. Ο σκοπός είναι να βελτιστοποιηθούν τα λάθη και οι αλγόριθμοι που έχουν παρόμοιο τρόπο λειτουργίας τείνουν να έχουν μεγάλο μερος των λαθών

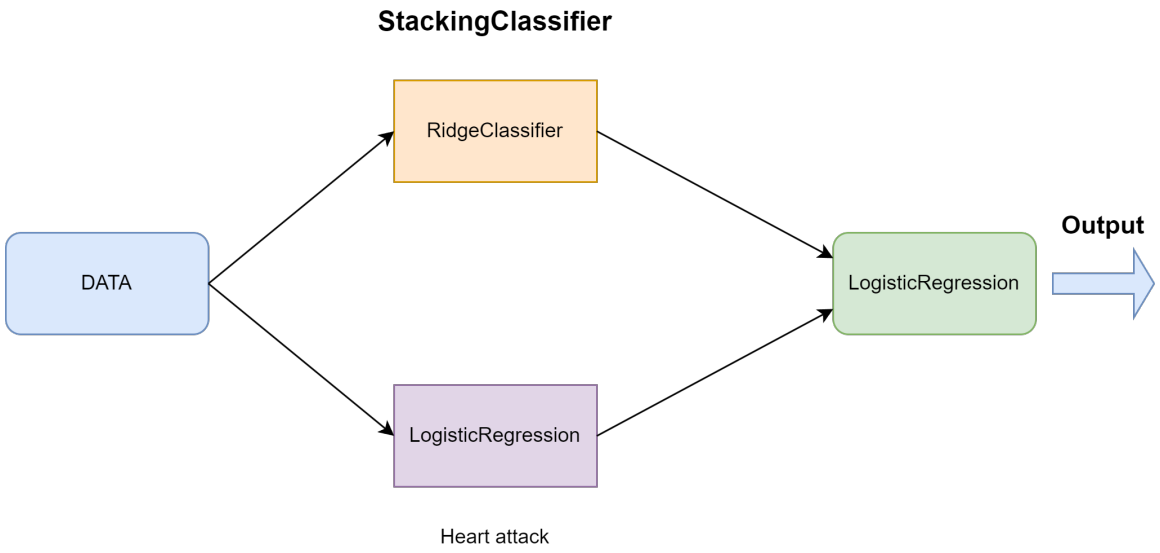
τους κοινό. Η επιλογή για παράδειγμα 2 ή 3 Boosting αλγόριθμων δεν θα επιφέρει καλύτερα αποτελέσματα. Η λύση λοιπόν είναι να επιλεγούν αλγόριθμοι που δεν έχουν τόσο κοινό τρόπο λειτουργίας με αποτέλεσμα να έχουν διαφοροποιήσεις στα λάθη και όταν συνδυαστούν να υπάρξουν καλύτερα αποτελέσματα. Ολα τα αποτελέσματα εισάγωνται σε ένα LogisticRegression αλγόριθμο ο οποίος δίνει το τελικό μοντέλο. Οι εκάστοτε επιλογές αλγορίθμων για κάθε Dataset είναι οι παρακάτω:

Parkinson's Dataset : SVC , KneighborsClassifier , GradientBoostingClassifier.



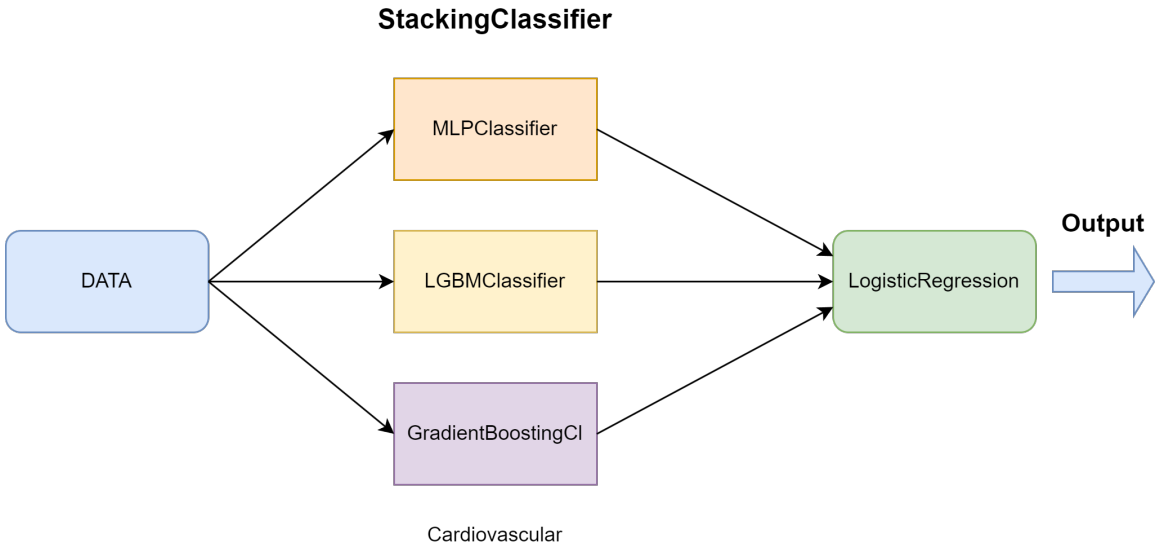
Σχήμα 3.6: *StackingClassifier Parkinson's Dataset.*

Heart attack Dataset : RidgeClassifier , LogisticRegression.



Σχήμα 3.7: *StackingClassifier Heart attack Dataset.*

Cardiovasculaar Dataset : MLPClassifier , LGBMClassifier , GradientBoostingClassifier.



Σχήμα 3.8: *StackingClassifier Cardiovasculaar Dataset.*

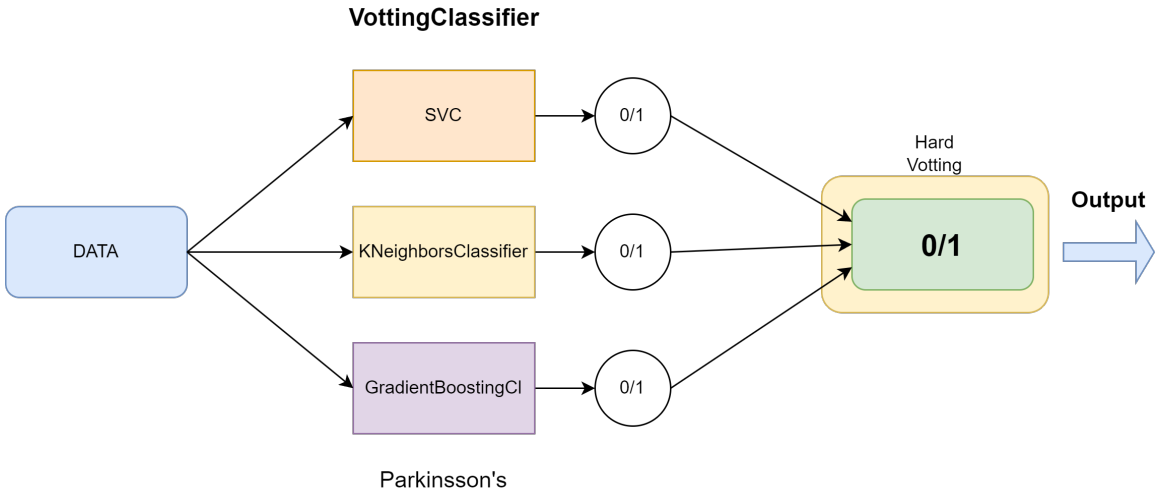
3.8 VotingClassifier

Ο Voting Classifier χρησιμοποιεί τα αποτελέσματα αλγορίθμων και μέσω ψηφοφορίας ανάλογα το τι έχει αποφανθεί κάθε αλγόριθμός δίνεται η τελική απάντηση. Η ψηφοφορία μπορεί να γίνει με 2 τρόπους : 1) Hard Voting : η απλή πλειοψηφία αποφασίζει την τελική πρόβλεψη,

βάση της πιο συχνής πρόβλεψης από τα μεμονωμένα μοντέλα. 2) Soft Voting : υπολογίζει τον μέσο όρο από τα μεμονωμένα μοντέλα και παράγει την τελική πρόβλεψη. Στην παρούσα πτυχιακή θα χρησιμοποιηθεί Hard Voting.

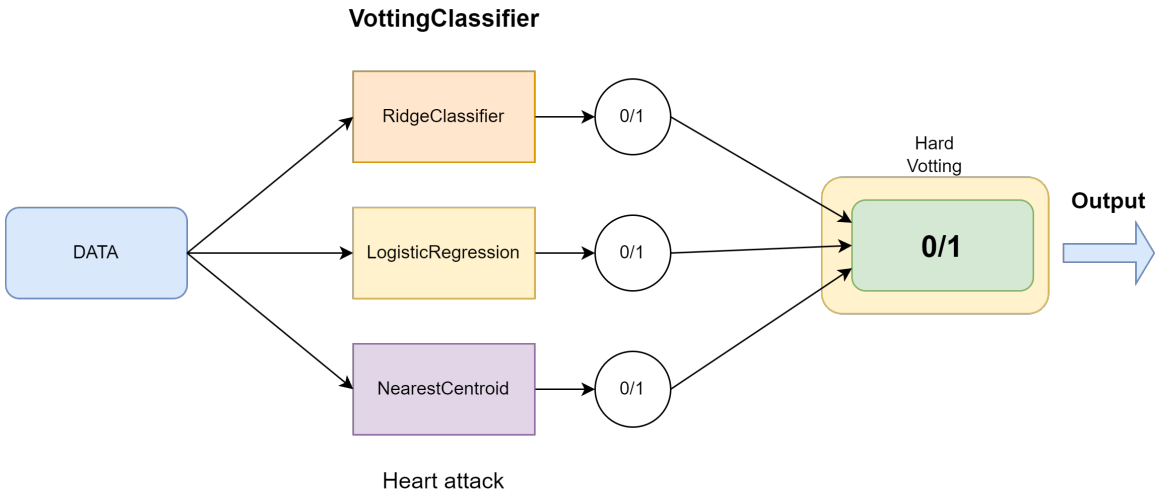
Παρακάτω έχουμε αναλυτικά τους αλγορίθμους που συνδυάστηκαν σε κάθε Dataset :

Parkinson's Dataset : SVC , KneighborsClassifier , GradientBoostingClassifier.



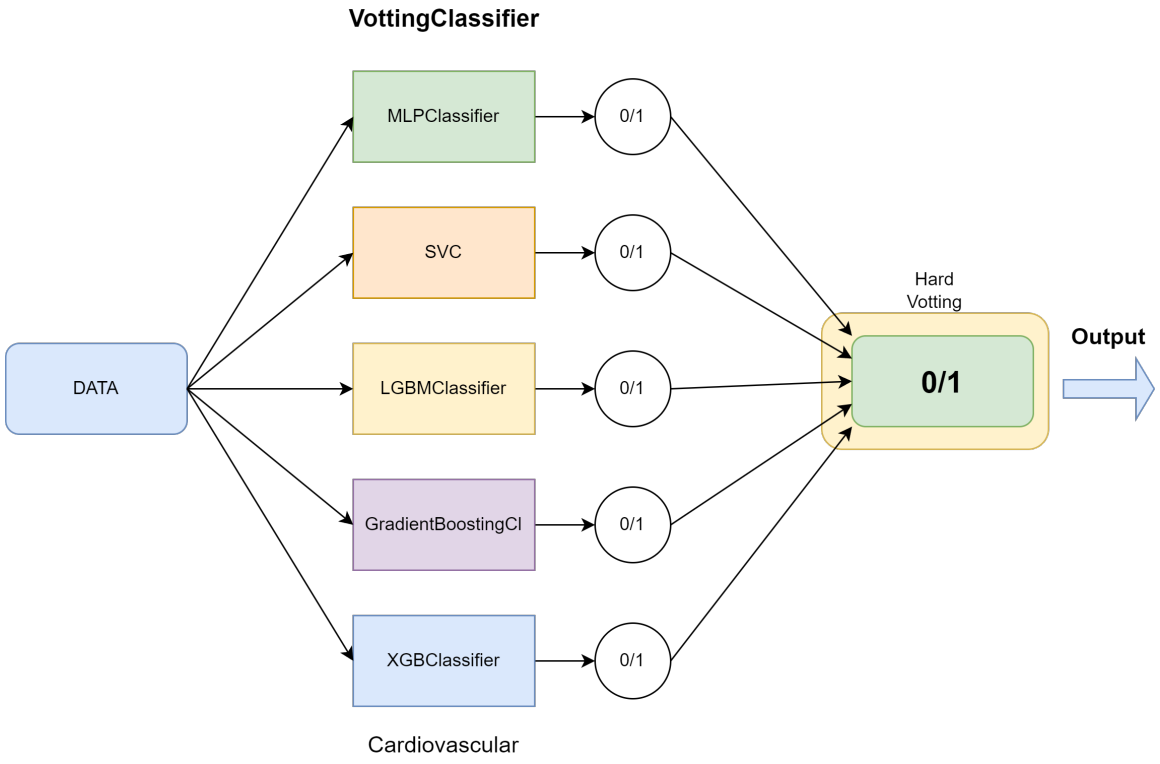
Σχήμα 3.9: *VotingClassifier Parkinson's Dataset.*

Heart attack Dataset : RidgeClassifier , LogisticRegression , NearestCentroid .



Σχήμα 3.10: *VotingClassifier Heart attack Dataset.*

Cardiovascular Dataset : MLPClassifier , SVC , LGBMClassifier , GradientBoostingClassifier , XGBClassifier.



Σχήμα 3.11: *VotingClassifier Cardiovascular Dataset.*

3.9 Dynamic Ensemble Selection (DES)

Η Dynamic Ensemble Selection (DES) [34] είναι μία τεχνική η οποία επιλέγει αυτόματα ποιοι αλγόριθμοι μηχανικής μάθησης από ένα σύνολο θα απαρτίζουν την ομάδα που θα συνδυαστεί για να υπάρξει βελτιστοποίηση στις μετρικές μας. Η επιλογή γίνεται βάση των χαρακτηριστικών από τα παραδείγματα που έχουν τεθεί για έλεγχο και οι αλγόριθμοι που θα έχουν τα καλύτερα αποτελέσματα όταν συνδυαστούν επιλέγονται. Αυτός ο τρόπος επιλογής έχει καλύτερα αποτελέσματα απο μία τυχαία επιλογή ή από ένα μέσο όρο των αποτελεσμάτων όλων των αλγορίθμων.[35]

3.9.1 KNORA-Union (KNORA-U)

Ο KNORA-U [36] επιλέγει αλγορίθμους οι οποίοι έχουν τουλάχιστον μία σωστή πρόβλεψη στην περιοχή όπου αναζητούμε. Στον κάθε αλγόριθμο αποδίδονται ψήφοι ανάλογα με τις σωστές προβλέψεις που έχει κάνει. Στο τέλος οι ψήφοι συνδυάζονται και παίρνουμε το τελικό αποτέλεσμα.

3.9.2 KNORA-Eliminate (KNORA-E)

Ο KNORA-E [37] ψάχνει για local Oracles οι οποίες είναι περιοχές όπου ο βασικός αλγόριθμος έχει το απόλυτο κατά την διάρκεια της κατηγοριοποίησης. Αν αυτό επιτευχθεί τότε ο αλγόριθμος εισάγεται στην ομάδα που θα συνδυαστεί. Άν κανένας αλγόριθμος δεν έχει 100% ποσοστό επιτυχίας η περιοχή μειώνεται αφαιρώντας στοιχεία από έξω προς τα μέσα μέχρι αυτό να επιτευχθεί. Αν δεν επιλεχθεί κάποιος αλγόριθμος τότε συνδυάζονται όλοι μεταξύ τους. Ο συνδυασμός γίνεται με majority voting.

3.9.3 Dynamic Ensemble Selection performance (DES-P)

Ο DES-P [38] επιλέγει όλους τους αλγορίθμους από ένα σύνολο οι οποίοι έχουν καλύτερη απόδοση από έναν προκαθορισμένο αλγόριθμο που ονομάζεται random classifier (RC). Η τιμή απόδοσης του random classifier δίνεται απο τον τύπο RC = 1/L όπου L ο αριθμός των κλάσεων στις οποίες τα δεδομένα θα κατηγοριοποιηθούν. Αν κανένας αλγόριθμος δεν επιλεχθεί τότε συνδυάζονται όλοι μεταξύ τους.

Ο γενικός τύπος για έλεγχο της απόδοσης του εκάστοτε αλγορίθμου είναι :

$$\boxed{\delta i, j = \widehat{P}(c_i|\theta_j) - 1/L} \tag{3.1}$$

DES-P competence estimation

Όπου c_i ο εκάστοτε αλγόριθμος.

4 Μεθοδολογία

Σε αυτό το κεφάλαιο παρουσιάζεται η μεθοδολογία που εφαρμόστηκε για την υλοποίηση της πτυχιακής εργασίας. Αρχικά γίνεται αναφορά στο περιβάλλον υλοποίησης, τη γλώσσα προγραμματισμού και στις βιβλιοθήκες που χρησιμοποιήθηκαν. Έπειτα παρουσιάζονται τα σύνολα δεδομένων τα οποία χρησιμοποιήθηκαν και θα αναφερθούμε επίσης στην προεπεξεργασία των δεδομένων δηλαδή στη διαδικασία που ακολουθήθηκε ώστε τα δεδομένα να έρθουν στην κατάληλη μορφή ώστε να χρησιμοποιηθούν από τους αλγόριθμους. Ακολούθως θα αναφερθούμε στην διαδικασία και τις μετρικές με τις οποίες θα επιλέξουμε και θα κρίνουμε τον καλύτερο αλγόριθμο. Στο τέλος ακολουθούν τα αποτελέσματα και το σύνολο του κώδικα.

4.1 Περιβάλλον υλοποίησης

Το περιβάλλον υλοποίησης που χρησιμοποιήθηκε είναι το Anaconda Navigator (version : 2023.03) και Jupyter Notebook (version : 6.5.2) με χρήση της γλώσσας προγραμματισμού Python 3.9. Οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι οι ακόλουθες :

• Pandas : Ανοιχτή βιβλιοθήκη που βοηθάει στην ανάλυση και μοντελοποίησή των δεδομένων.

• Numpy :Ανοιχτή βιβλιοθήκη που περιέχει μαθηματικές συναρτήσεις για επεξεργασία των δεδομένων και πολυδιάστατων πινάκων.

• Scipy :Συλλογή μαθηματικών λειτουργιών βασισμένη στο Numpy

• Scikit-Learn :Βιβλιοθήκη που περιέχει αλγορίθμους μηχανικής μάθησης βασισμένη στο Scipy

• Matplotlib :Βιβλιοθήκη γραφημάτων.

• Seaborn :Βιβλιοθήκη γραφημάτων.

• Plotly :Βιβλιοθήκη γραφημάτων.

• Lazypredict : Βιβλιοθήκη για το γρήγορο έλεγχο πολλαπλών αλγορίθμων.

• Warnings : Βιβλιοθήκη για δημιουργία και έλεγχο warnings.
• re : Βιβλιοθήκη της Python για Regular expressions.
• collections : Βιβλιοθήκη που περιέχει εναλακτικούς τρόπους αποθήκευσης των δεδομένων.
• xgboost : Βιβλιοθήκη για χρήση του extreme gradient boosting αλγορίθμου.
• lightgbm :Βιβλιοθήκη για χρήση του LGBMClassifier αλγορίθμου.
• deslib.des :Βιβλιοθήκη που περιέχει Dynamic Ensemble Selection αλγορίθμους.
• imblearn :Βιβλιοθήκη για over sampling ελέγχθηκε δεν χρησιμοποιήθηκε στα τελικά αποτέλεσματα.

4.2 Επεξήγηση δεδομένων

Στην παρούσα πτυχιακή χρησιμοποιήθηκαν 3 binary classification dataset ιατρικών δεδομένων, δηλαδή 3 σύνολα δεδομένων όπου περιέχουν μετρικές από διάφορους ασθενείς και κατηγοριοποίηση 0 ή 1 αν πάσχουν ή όχι απο μία ασθένεια. Παρακάτω υπάρχουν λίγες πληροφορίες και αναλυτικά το τι περιέχει το κάθε dataset.

4.2.1 **Dataset 1**

Αυτό το Dataset αποτελείται από 13 στήλες μετρικών και 1 στήλη κατηγοριοποίησης. Ο κύριος στόχος των δεδομένων είναι η διάκριση των υγιών ατόμων από τα άτομα που έχουν μεγάλη πιθανότητα για καρδιακή προσβολή σύμφωνα με τη στήλη Target. Το Dataset έχει 303 σειρές από διαφορετικά άτομα.

Heart Attack Analysis and Prediction Dataset

• Age: Ηλικία σε χρόνια
• Sex : Φύλο.
• Cp : Πόνος στο θώρακα
• Trestbps : Συστολική αρτηριακή πίεση.
• Chol : Χοληστερίνη.
• Fbs : Ζάχαρο.
• Restecg : Ηλεκτροκαρδιογράφημα.
• Thalach : Μέγιστοι παλμοί.

• Exang : Στηθάγχη λόγω άθλησης.

• Oldpeak : ST-segment depression.

• Slope : Slope of the peak exercise ST segment.

• Ca : Αριθμός κύριων αρτηριών.

• Thal : Κατηγοριοποίηση παλμών.

• Target : Υπάρχει ή όχι πιθανότητα για καρδιακή προσβολή

4.2.2 **Dataset 2**

Αυτό το Dataset αποτελείται από μια σειρά βιοϊατρικών φωνητικών μετρήσεων από 31 άτομα, 23 με νόσο του Πάρκινσον. Κάθε στήλη στον πίνακα είναι μία φωνητική μέτρηση και κάθε σειρά αντιστοιχεί σε μία από τις 195 ηχογραφήσεις φωνής από αυτά τα άτομα. Ο κύριος στόχος των δεδομένων είναι η διάκριση των υγιών ατόμων από τα άτομα με πάρκινσον σύμφωνα με τη στήλη Status.[39] Το Dataset περιέχει 22 στήλες μετρικών 1 στήλη ονόματος και 1 στήλη κατηγοριοποίησης. Το Dataset έχει 195 σειρές.

Oxford Parkinson Disease Detection Dataset

• Name: Όνομα.

• MDVP Fo(Hz) : Μέση συχνότητα φωνής.

• MDVP Fhi(Hz) : Μέγιστη συχνότητα φωνής.

• MDVP Flo(Hz) : Ελάχιστη συχνότητα φωνής.

• MDVP Jitter(%),MDVP Jitter(Abs),MDVP RAP,MDVP PPQ,Jitter DDP : Διάφορες μετρικές στην διαφοροποίηση της συχνότητας .

• MDVP Shimmer,MDVP Shimmer(dB),Shimmer APQ3,Shimmer APQ5,MDVP APQ,Shimmer DDA : Διάφορες μετρικές στο εύρος της συχνότητας.

• NHR , HNR : 2 Μετρικές για το ποσοστό θορύβου στα δομικά χαρακτηριστικά της φωνής.

• Status : Πάσχει ή όχι απο Parkinsson's.

• RPDE , D2 : 2 μη γραμμικές μετρικές πολυπλοκότητας.

• DFA : Μετρική μέτρησης δυναμικού σήματος.

• Spread1 , Spread2 , PPE : 3 μη γραμμικές μετρικές διαφοροποίησης συχνότητας.

4.2.3 **Dataset 3**

Αυτό το Dataset αποτελείται από 11 στήλες μετρικών 1 στήλη ονόματος και 1 στήλη κατηγοριοποίησης. Ο κύριος στόχος των δεδομένων είναι η διάκριση των υγιών ατόμων από τα άτομα που έχουν καρδιαγγειακό πρόβλημα σύμφωνα με τη στήλη Cardio. Το Dataset έχει 70.000 σειρές απο διαφορετικά άτομα που αποτελεί το μεγαλύτερο με διαφορά απο τα 3 που χρησιμοποιήθηκαν.

Cardiovascular Disease dataset

- Age: Ηλικία σε χρόνια.
- Gender : Φύλο.
- Height : Ύψος σε εκατοστά.
- Weight : Βάρος σε κιλά.
- ApHi : Συστολική αρτηριακή πίεση.
- ApLow : Διαστολική αρτηριακή πίεση.
- Cholesterol : Χοληστερίνη.
- Gluc : Γλυκόζη.
- Smoke : Καπνιστής
- Alco : Καταναλώνει αλκοόλ.
- Active : Αθλείται.
- Cardio : Υπάρχει ή όχι καρδιαγγειακό πρόβλημα.

4.3 Αρχική επεξεργασία δεδομένων

Αρχικά έγινε έλεγχος και αφαιρέθηκαν οι μηδενικές τιμές από τα δεδομένα. Πολλές φορές κατά τη διάρκεια συλλογής ή επεξεργασίας των δεδομένων γίνονται λάθη και τιμές αποτυπώνονται ως μηδέν ή NULL αυτές αφαιρέθηκαν καθώς επηρεάζουν την εκπάιδευση των αλγορίθμων αρνητικά. Επίσης αφαιρέθηκαν οι διπλότυπες τιμές από τα δεδομένα για τον ίδιο λόγο. Στη συνέχεια στο Cardiovascular Disease dataset έγινε προσθήκη νέου δοδομένου

(BMI) το οποίο δημιουργήθηκε από άλλα ήδη υπάρχοντα δεδομένα (kg/height). Επίσης το (BMI) και η ηλικία χωρίστηκαν σε κατηγορίες για καλύτερα αποτελέσματα. Στα άλλα δύο dataset δεν έγινε ανάλογη προσθήκη δεδομένων. Τώρα όσον αναφορά τους Outliers που πολλές φορές αφαιρούνται για καλύτερα αποτελέσματα στα δεδομένα που χρησιμοποιήθηκα δεν αφαιρέθηκαν. Τα ιατρικά δεδομένα έχουν την ιδιαιτερότητα ότι φαινομενικά λάθος δεδομένα μπορεί να είναι στην πράξη πολύ ισχυροί δείκτες (π.χ αρτηριακή πίεση πάνω από 20). Επιπρόσθετα ελέγχονται καλύτερα οι διαφορές ανάμεσα στους πιθανούς αλγορίθμους καθώς βλέπουμε πώς αντιδρούν στους πιθανούς outliers. Κάποιοι από τους αλγορίθμους χρειάζονται σαν είσοδο κλιμακωτά δεδομένα οπότε ένα αντίγραφο των δεδομένων δέχθηκε την κατάληλη επεξεργασία χρησιμποποιώντας τον StandardScaler του sklearn.

4.4 **Split Data,stratify**

Στα δεδομένα έγινε ένα split 70/30 (80/20 στο cardio dataset καθώς είχε μεγάλο αριθμό δεδομένων). Αυτό σημαίνει ότι πριν κάνουμε train τους αλγορίθμους μηχανικής μάθησης χωρίζουμε τα δεδομένα σε δυο κομμάτια. Το 70% (80%) χρησιμοποιείται για το train και το υπόλοιπο 30% (20%) για τον έλεγχο των αποτελεσμάτων. Με αυτόν τον τρόπο μπορούμε εύκολα να ελέγξουμε αν ο αλγόριθμος έχει προσαρμοστεί ιδιαίτερα έντονα στα δεδομένα (Overfitting) . Επιπρόσθετα χρησιμοποιήθηκε η μετρική statify y, η συγκεκριμένη μετρική θα διατηρήσει την αναλογία των δεδομένων στα 2 νέα κομμάτια ίδια με τα αρχικά δεδομένα.

4.5 Δημιουργία συναρτήσεων

Καθώς θα χρησιμοποιηθούν αρκετοί αλγόριθμοι δημιουργήθηκαν 2 συναρτήσεις, μία για τους αλγορίθμους που χρειάζονται κλιμακωτά δεδομένα και μία για αυτούς που δεν χρειάζονται. Σκοπός αυτών των συναρτήσεων είναι να τις καλούμε με το όνομα του αλγορίθμου να παίρνουμε κατευθείαν αποτελέσματα και να τα προσθέτουμε σε λίστες. Τα αποτελέσματα που παίρνουμε είναι Accuracy (testing , training data) , Precision , Recall , F1 , F2 , Confusion Matrix από αυτά στις λίστες αποθηκεύονται τα Accuracy (testing , training data) , F1 , F2 τα οποία παρουσιάζονται πλήρως στα αποτελέσματα.

4.6 Διαδικασία επιλογής καλύτερου αλγορίθμου

Η διαδικασία με την οποία αντιλαμβανόμστε την απόδοση του κάθε αλγορίθμου περιέχει αρκετές τιμές και συνδυασμό αυτών. Στη παρούσα πτυχιακή ελέγχθηκαν αρκετές ευρέως διαδεδομένες μετρικές αλλά αυτή που χρησιμοποιήθηκε σαν κύρια είναι μια λιγότερο γνωστή με όνομα F2 score μια παραλλαγή της F1 score. Παρακάτω ακολουθούν περιγραφές των τιμών και μετρικών που χρησιμοποιήθηκαν.

4.6.1 **TP/TN/FP/FN**

• True positives (TP) : Είναι τα δείγματα τα οποία το μοντέλο έχει προβλέψει σαν θετικά και είναι πράγματι θετικά.

• True negatives (TN) : Είναι τα δείγματα τα οποία το μοντέλο έχει προβλέψει σαν αρνητικά και είναι πράγματι αρνητικά.

• False positives (FP) : Είναι τα δείγματα τα οποία το μοντέλο έχει προβλέψει σαν θετικά όμως δεν είναι.

• False negatives (FN) : Είναι τα δείγματα τα οποία το μοντέλο έχει προβλέψει σαν αρνητικά όμως δεν είναι.

4.6.2 **Accuracy on Trainining data**

Είναι το ποσοστό των δειγμάτων τα οποία το μοντέλο έχει προβλέψει σωστά στα δεδομένα που χρησιμοποιήθηκαν για την εκπαίδευση.

$$\boxed{Accurasy = \frac{TP + TN}{TP + TN + FP + FN}} \tag{4.1}$$

Accuracy

4.6.3 **Accuracy on Testing data**

Είναι το ποσοστό των δειγμάτων τα οποία το μοντέλο έχει προβλέψει σωστά στα δεδομένα που χρησιμοποιήθηκαν για τον έλεγχο.

$$\boxed{Accurasy = \frac{TP + TN}{TP + TN + FP + FN}} \tag{4.2}$$

Accuracy

4.6.4 **Precision**

Είναι το ποσοστό των δειγμάτων τα οποία το μοντέλο έχει προβλέψει σαν θετικά και είναι πράγματι θετικά. Αυτή η μετρική χρησιμοποιείται κυρίως όταν τα FP έχουν μεγάλο κόστος

$$\boxed{Precision = \frac{TP}{TP + FP}} \tag{4.3}$$

Precision

4.6.5 **Recall**

Είναι το ποσοστό των θετικών δειγμάτων τα οποία το μοντέλο έχει προβλέψει σαν θετικά και είναι πράγματι θετικά. Η διαφορά με την Precision είναι ότι αυτή η μετρική χρησιμοποιείται κυρίως όταν τα FN έχουν μεγάλο κόστος.

$$\boxed{Recall = \frac{TP}{TP + FN}} \tag{4.4}$$

Recall

4.6.6 **F-Measure , Fbeta-Measure**

Σε γενικές γραμμές το F-Measure είναι μια μέτρηση που συνδυάζει το Precision και το Recall ώστε να υπάρξει μία πιο ολοκληρωμένη εικόνα καθώς στα περισσότερα προβλήματα μόνο το Precision ή μόνο το Recall δεν είναι αρκετά. Βέβαια ανάλογα το πρόβλημα μπορεί να χρειαστεί να δωθεί περισσότερη σημασία στην μία ή στην άλλη μετρική όπως για παράδειγμα στα Ιατρικά δεδομένα όπου τα false negatives έχουν μεγαλύτερη σημασία χωρίς όμως να σημαίνει ότι δεν πρέπει να ληφθούν υπόψιν τα false positives. Αυτό το πρόβλημα λύνει η Fbeta-Measure μία παραλλαγή της F-Measure στην οποία η ισορροπία ανάμεσα σε Precision και Recall ελέγχεται από μία μετρική που ονομάζεται Beta.

$$\boxed{F - Measure = \frac{2 * Precision * Recall}{Precision + Recall}} \tag{4.5}$$

F-Measure

$$\boxed{Fbeta - Measure = \frac{(1 + beta^2) * Precision * Recall}{beta^2 * Precision + Recall}} \tag{4.6}$$

Fbeta-Measure

4.6.7 **F1 score**

Το F1 score είναι η Fbeta-Measure με τιμή Beta 1 που πρακτικά είναι ο ορισμός της F-Measure.

$$\boxed{F1 = \frac{2 * Precision * Recall}{Precision + Recall}} \tag{4.7}$$

F1

4.6.8 **F2 score**

Το F2 score είναι η Fbeta-Measure με τιμή Beta 2, αυτό επιφέρει την μείωση της σημαντικότητας του Precision και συνάμα την αύξηση της σημαντικότητας του Recall. Αυτό έχει σαν αποτέλεσμα το F2 score να δίνει περισσότερη σημασία στην μείωση των false negatives σε σχέση με τα false positives. Το F2 score θα είναι η κύρια μετρική που θα χρησιμοποιηθεί για να ελεγχθεί ποιος είναι ο αλγόριθμος με την καλύτερη απόδοση.

$$\boxed{F2 = \frac{5 * Precision * Recall}{4 * Precision + Recall}} \tag{4.8}$$

F2

5 Αποτελέσματα

Τα αποτελέσματα για κάθε Dataset και κάθε αλγόριθμο όπως αυτά αποτυπώθηκαν απο τις συναρτήσεις που δημιουργήσαμε ακολουθούν παρακάτω. Οι τιμές είναι (Taining score , Testing score , F1 score , F2 score) με φθίνουσα σειρά του F2 score. Τα αποτελέσματα είναι πολλών δεκαδικών ψηφίων όμως για λόγους ευκολίας στην ανάγνωση αποτυπώθηκαν τα πρώτα δυο, παρόλα αυτά τόσο η σειρά στον πίνακα όσο και τα γραφήματα είναι σύμφωνα με την πλήρης τιμή. Αυτό σημαίνει ότι παρότι 2 τιμές F2 score είναι ίδιες στον πίνακα η από κάτω είναι ελάχιστα μικρότερη.

5.0.1 **Parkinsons Dataset**

Algorithms	Training Score	Testing Score	F1 Score	F2 Score
Knorau	1.00	0.93	0.96	0.98
Desp	1.00	0.93	0.96	0.98
StackingClassifier	1.00	0.92	0.95	0.98
Knorae	1.00	0.92	0.95	0.96
VotingClassifier	0.90	0.85	0.91	0.96
SVC	0.89	0.90	0.93	0.96
Ada Boost	1.00	0.90	0.93	0.95
Gradient Boost	1.00	0.88	0.92	0.94
MLPClassifier	0.96	0.88	0.92	0.91
KNN	0.93	0.88	0.92	0.91
ExtraTreesClassifier	1.00	0.86	0.91	0.91
Decision Tree	1.00	0.88	0.92	0.90
Random Forest	1.00	0.83	0.88	0.87
RidgeClassifier	0.89	0.83	0.88	0.87
XGBoost	0.90	0.76	0.84	0.83
Logistic Regression	0.88	0.78	0.84	0.81
PassiveAggressiveCl	0.82	0.71	0.77	0.70
Perceptron	0.88	0.69	0.76	0.70
NearestCentroid	0.76	0.66	0.73	0.66
Gausian	0.75	0.64	0.71	0.63

Parkinson's (Ranked F2)

Σχήμα 5.1: *Parkinsons Dataset results.*

5.0.2 **Heart Attack Analysis**

Algorithms	Training Score	Testing Score	F1 Score	F2 Score
VotingClassifier	0.84	0.90	0.91	0.94
StackingClassifier	0.82	0.90	0.91	0.94
RidgeClassifier	0.82	0.89	0.90	0.94
Logistic Regression	0.82	0.89	0.90	0.92
Desp	0.93	0.87	0.88	0.92
PassiveAggressiveCl	0.79	0.86	0.88	0.91
NearestCentroid	0.82	0.87	0.88	0.90
LGBMClassifier	1.00	0.84	0.85	0.88
Knorau	0.94	0.84	0.85	0.88
Gausian	0.82	0.82	0.84	0.86
MLPClassifier	0.91	0.86	0.87	0.86
KNN	0.87	0.84	0.85	0.85
Perceptron	0.82	0.85	0.85	0.84
Ada Boost	0.94	0.80	0.82	0.82
SVC	0.90	0.82	0.83	0.81
ExtraTreesClassifier	1.00	0.81	0.82	0.81
Gradient Boost	1.00	0.79	0.80	0.80
XGBoost	1.00	0.79	0.80	0.80
Knorae	1.00	0.79	0.80	0.80
Random Forest	1.00	0.81	0.82	0.79
Decision Tree	1.00	0.75	0.77	0.77

Heart Attack (Ranked F2)

Σχήμα 5.2: *Heart Attack Analysis Dataset results.*

5.0.3 **Cardiovascular Disease dataset**

Algorithms	Training Score	Testing Score	F1 Score	F2 Score
VotingClassifier	0.75	0.74	0.73	0.71
Knorau	0.79	0.73	0.73	0.71
StackingClassifier	0.74	0.74	0.73	0.71
LGBMClassifier	0.75	0.74	0.73	0.71
Gradient Boost	0.74	0.74	0.73	0.71
MLPClassifier	0.74	0.73	0.72	0.70
SVC	0.73	0.73	0.72	0.70
XGBoost	0.77	0.73	0.72	0.70
Random Forest	1.00	0.71	0.71	0.70
Desp	0.81	0.73	0.72	0.70
Knorae	1.00	0.71	0.71	0.70
ExtraTreesClassifier	1.00	0.70	0.70	0.70
Logistic Regression	0.72	0.72	0.71	0.69
PassiveAggressiveCl	0.65	0.66	0.67	0.69
Ada Boost	0.73	0.73	0.71	0.68
Perceptron	0.46	0.46	0.57	0.66
Decision Tree	1.00	0.64	0.64	0.63
KNN	0.76	0.65	0.64	0.63
NearestCentroid	0.64	0.64	0.63	0.62
RidgeClassifier	0.65	0.65	0.63	0.62
Gausian	0.62	0.62	0.51	0.44

Cardiovascular (Ranked F2)

Σχήμα 5.3: *Cardiovascular Disease Dataset results.*

5.0.4 Επεξήγηση Αποτελεσμάτων

Μελετώντας τα αποτελέσματα μπορούμε να βγάλουμε αρκετά συμπεράσματα για τους αλγορίθμους μηχανικής μάθησης που χρησιμοποιήθηκαν.

5.1 Καλύτερα Αποτελέσματα

Parkinsons	Heart Attack	Cardiovascular
Knorau	VotingClassifier	VotingClassifier
Desp	StackingClassifier	Knorau
StackingClassifier	RidgeClassifier	StackingClassifier
Knorae	Logistic Regresion	LGBMClassifier
VotingClassifier	Desp	Gradient Boost

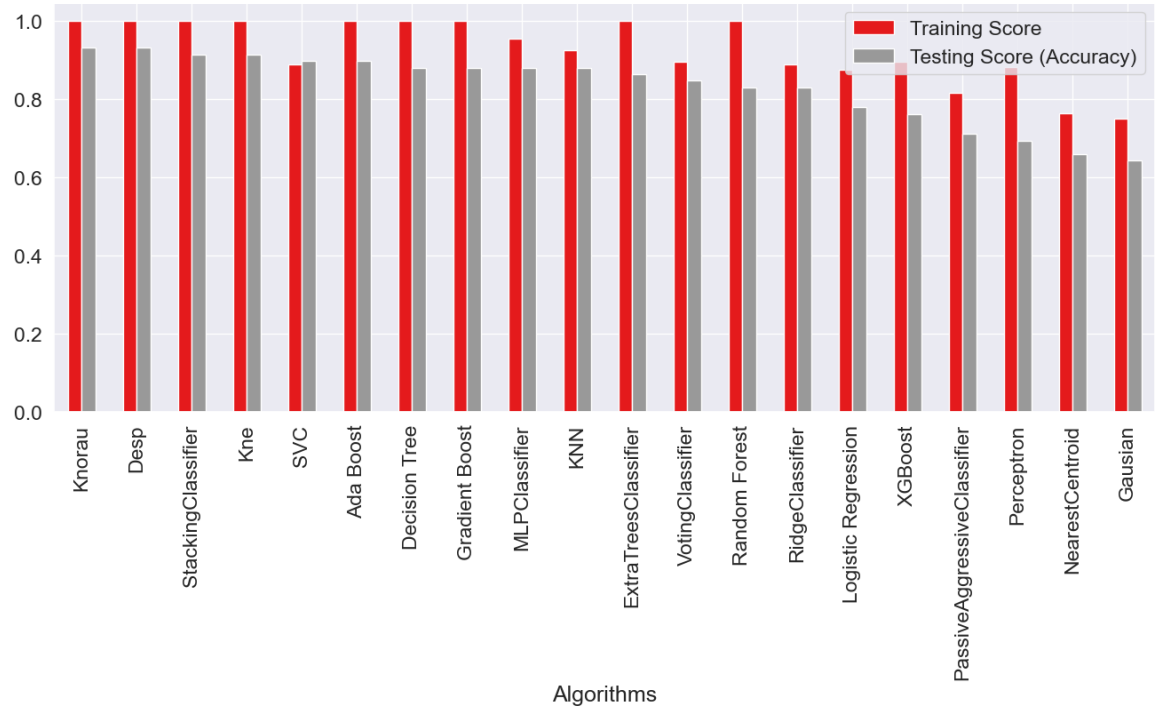
Σχήμα 5.4: 6 αλγόριθμοι με καλύτερα αποτελέσματα *F2 score*.

Μπορούμε να βρούμε κάποιους αλγόριθμους στην κορυφή της λίστας με την καλύτερη απόδοση και στα 3 Dataset. Συγκεκριμένα ο StackingClasiffier βρίσκεται μια φορά στην 2η θέση και δύο στην 3η οπότε αντιλαμβανόμαστε ότι μπορεί να έχει καλή απόδοση τόσο σε μικρά Dataset (Parkinsosns , Heart attack) όσο και σε μεγαλύτερα όπως το Cardiovascular Disease Dataset. Επίσης ο VotingClasiffier είχε εξαιρετικά αποτελέσματα καθώς όχι μόνο βρισκόταν στην πρώτη θέση σε 2 απο τα 3 Dataset αλλά και στο τρίτο είχε πολύ μικρή διαφορά απο την κορυφή. Επιπρόσθετα ο VotingClasiffier είχε απο τα μικρότερα προβλήματα Overfitting. Αξιοι αναφοράς είναι και 2 απο τους DES αλγόριθμους Knorau, Desp που ήταν ψηλά στην λίστα σε 2 απο τα 3 Dataset αλλά έχουν έντονα σημάδια Overfitting.

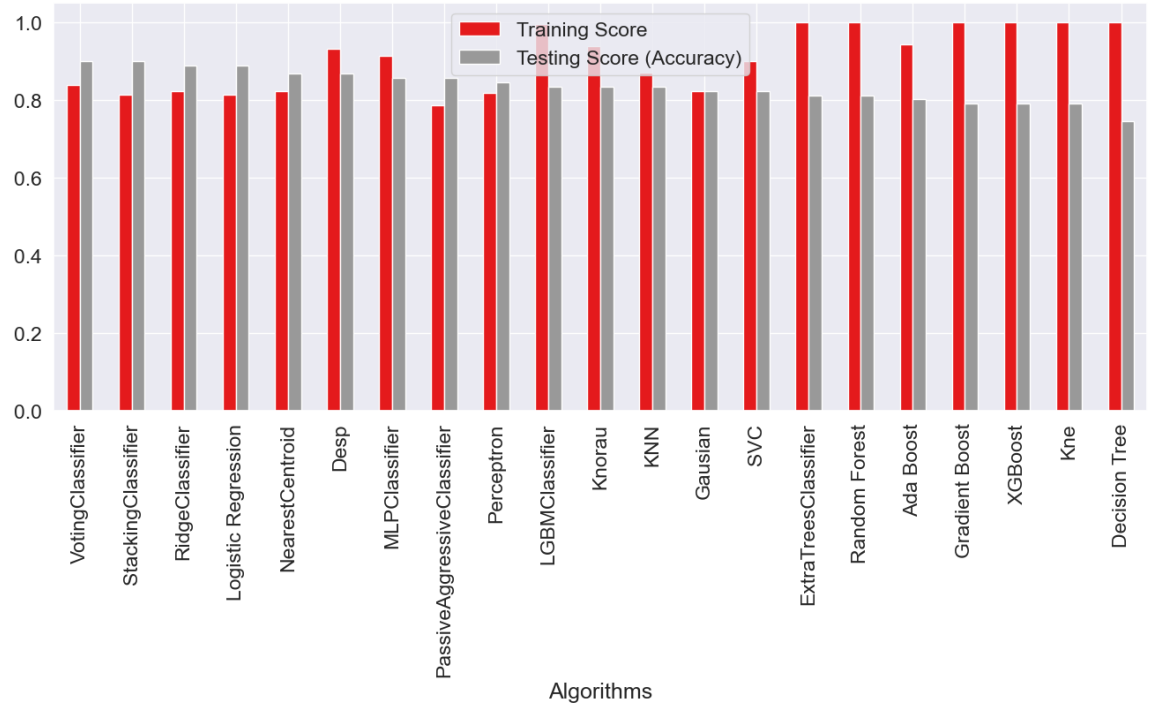
5.2 **Overfitting**

Για να αντιληφούμε αν ο αλγόριθμος έχι προβλήματα Overfitting αρκεί να κοιτάξουμε την διαφορά ανάμεσα στο Training και Testing score. Αυτό μπορεί να γίνει εύκολα με κάποια

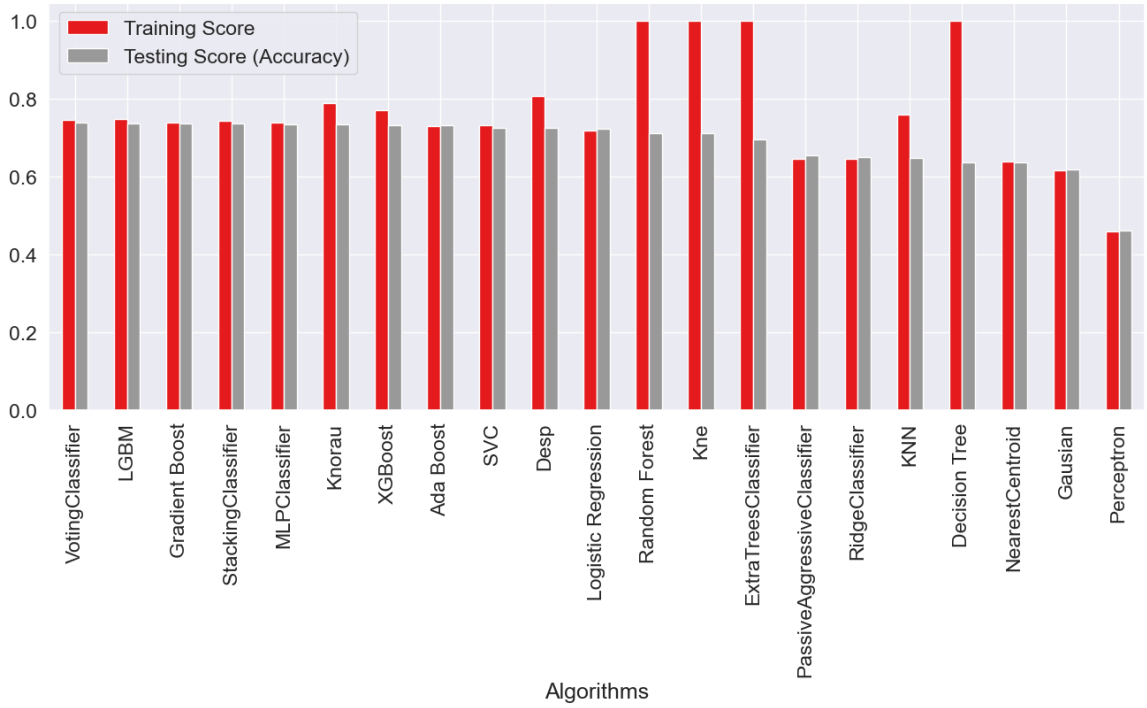
γραφήματα που τα συγκρίνουν.



Σχήμα 5.5: *Train , Test score Parkinsons*



Σχήμα 5.6: *Train , Test score Heart Attack Analysis*

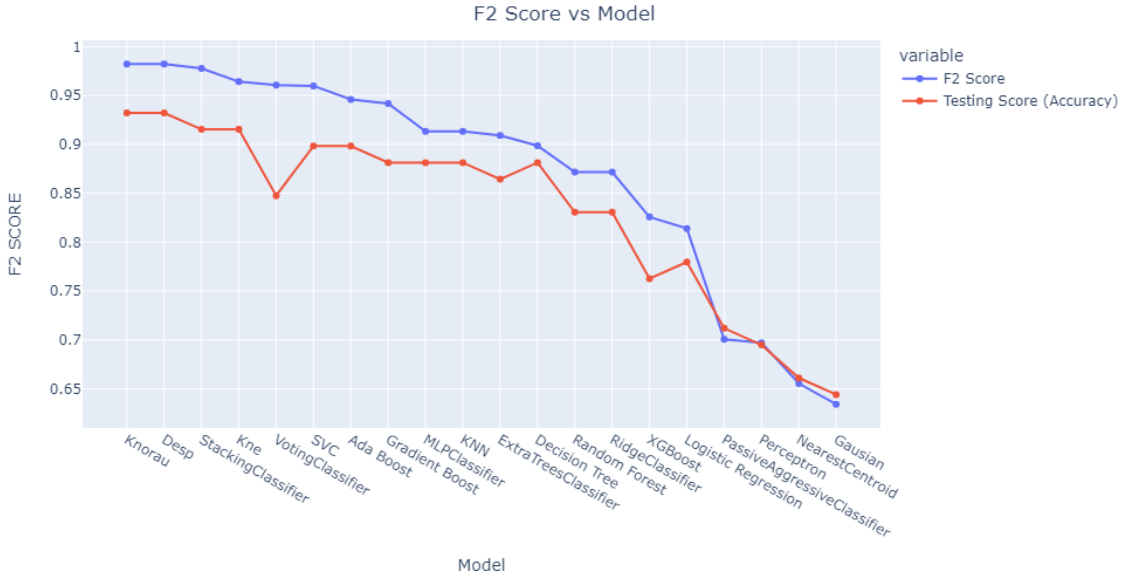


Σχήμα 5.7: *Train , Test score Cardiovascular Disease dataset*

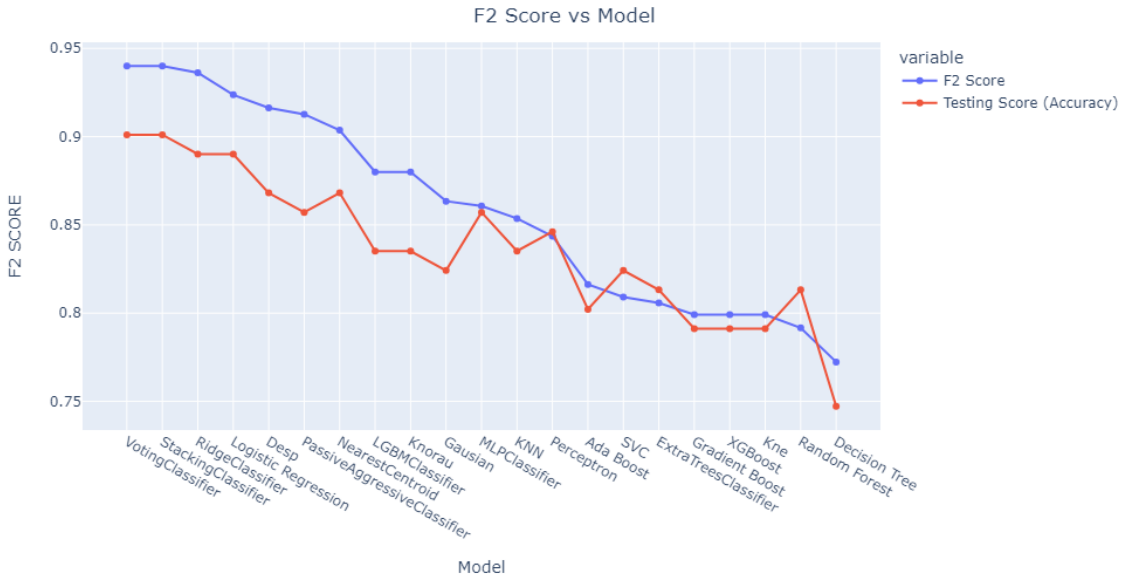
Απο τα γραφήματα καταλαβαίνουμε ότι οι αλγόριθμοι Random Forest και Extra trees έχουν ιδιαίτερα έντονο πρόβλημα με Overfitting αλλά παρόλα αυτά τα αποτελέσμτατά τους δεν είναι απαγορευτικά και μπορούν να χρησιμοποιηθούν.

5.3 **Accurasy vs F2 score**

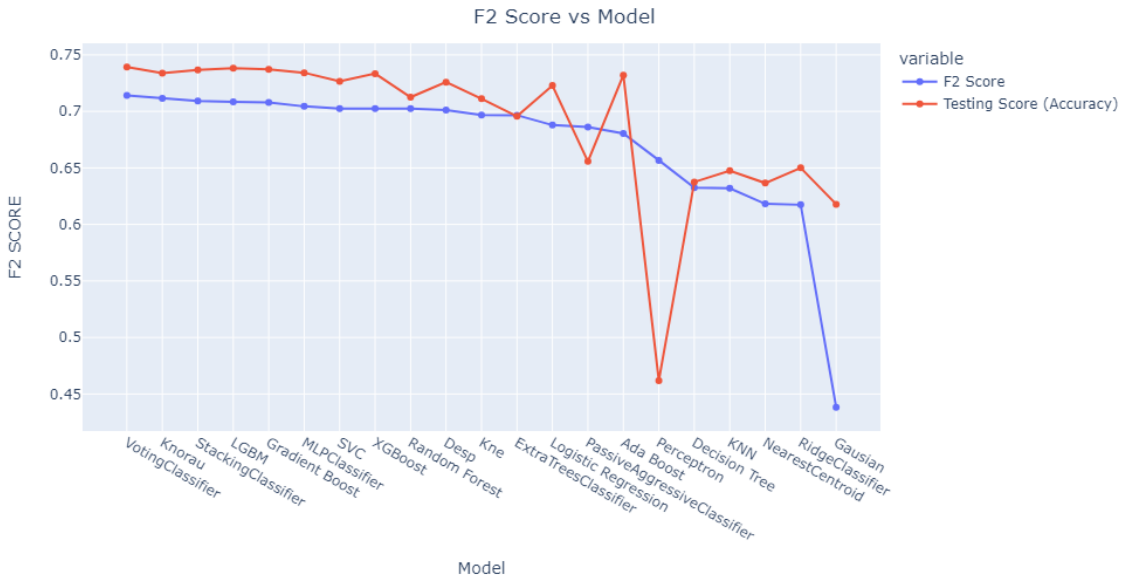
Η σύγκριση ανάμεσα σε μοντέλα μηχανικής μάθησης γίνεται συνήθως με τη χρήση του Accurasy Score δηλαδή τα αποτελέσματα στα Testing data. Όπως αναφέρθηκε στο κεφάλαιο για την διαδικασία επιλογής αλγορίθμου επιλέχθηκε η μετρική F2 score, καλό θα ήταν να δούμε μία σύγκριση τον δύο μετρικών στα 3 dataset.



Σχήμα 5.8: *Accurasy vs F2 score Parkinsons*



Σχήμα 5.9: *Accurasy vs F2 score Heart Attack Analysis*



Σχήμα 5.10: *Accurasy vs F2 score Cardiovascular Disease dataset*

Παρότι ακολουθούν μία σχετικά παράλληλη πορεία, όσο πλησιάζουμε στους λιγότερα αποδοτικούς αλγορίθμους βλέπουμε πολλές διακυμάνσεις. Έτσι αντιλαμβανόμαστε ότι η επιλογή της σωστής μετρικής είναι ένα από τα σημαντικότερα βήματα καθώς μπορεί να επιφέρει τελείως διαφορετικά αποτελέσματα χωρίς να ξέρουμε γιατί γίνεται αυτό.

6 Κώδικας

Παρακάτω ακολουθεί αναλυτικά ο κώδικας που δημιουργήθηκε για την επεξεργασία των δεδομένων και τον έλεγχο της απόδοσης τόσο των βασικών αλγορίθμων όσο και των Ομαδικών Σχημάτων μάθησης. Ο κώδικας είναι συγκεκριμένα για το Cardiovascular Disease dataset αλλά η διαδικασία που ακολουθήθηκε είναι ίδια και για τα άλλα δύο dataset.

```
import numpy as np

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
sns.set(style="darkgrid",font_scale=1.5)
pd.set_option("display.max.rows",None)
pd.set_option("display.max.columns",None)
from lazypredict.Supervised import LazyClassifier
import lazypredict
import plotly.express as px
import re
from collections import Counter

from sklearn.svm import SVC,LinearSVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression,Perceptron,
    PassiveAggressiveClassifier,RidgeClassifier
from sklearn.semi_supervised import LabelPropagation
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier,NearestCentroid
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,
    GradientBoostingClassifier,AdaBoostClassifier,
    StackingClassifier,VotingClassifier,BaggingRegressor,
    ExtraTreesClassifier
from xgboost import XGBClassifier

from lightgbm import LGBMClassifier

from deslib.des import DESP
from deslib.des import KNORAE
from deslib.des import KNORAU
from deslib.des import KNOP

from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split, GridSearchCV,
     RandomizedSearchCV
from sklearn.metrics import accuracy_score, f1_score, recall_score,
     precision_score, confusion_matrix , fbeta_score
from sklearn.preprocessing import LabelEncoder, OneHotEncoder,
    StandardScaler, MinMaxScaler

from imblearn.over_sampling import SMOTE
```

Listing 1: Imports

```
train_df = pd.read_csv(r"C:\..", delimiter = ';')
train_df.head()
```

Listing 2: Read Dataset

```
train_df.info()
```

Listing 3: Dataset Info.

```
train_df["age"]=(train_df["age"])/365
```

Listing 4: Age from years to days.

```
train_df['bmi'] = train_df['weight']/((train_df['height']/100)**2)
```

Listing 5: New Feature Bmi.

```
train_df.loc[(train_df.bmi <= 18.4), 'bmicat'] = 1
train_df.loc[(train_df.bmi > 18.4) & (train_df.bmi < 25), 'bmicat']
     = 2
train_df.loc[(train_df.bmi >= 25) & (train_df.bmi < 30), 'bmicat']
    = 3
train_df.loc[(train_df.bmi >= 30) & (train_df.bmi < 40), 'bmicat']
    = 4
train_df.loc[(train_df.bmi >= 40), 'bmicat'] = 5
```

Listing 6: Feature Bmi categories.

```
train_df.loc[(train_df.age <= 40), 'agecat'] = 1
train_df.loc[(train_df.age > 40)&(train_df.age<=45), 'agecat'] = 2
train_df.loc[(train_df.age > 40)&(train_df.age<=45), 'agecat'] = 3
train_df.loc[(train_df.age > 45)&(train_df.age<=50), 'agecat'] = 4
train_df.loc[(train_df.age > 50)&(train_df.age<=55), 'agecat'] = 5
train_df.loc[(train_df.age > 55)&(train_df.age<=60), 'agecat'] = 6
train_df.loc[(train_df.age > 60)&(train_df.age<=65), 'agecat'] = 7
train_df.loc[(train_df.age > 65)&(train_df.age<=70), 'agecat'] = 8
train_df.loc[(train_df.age >= 70), 'agecat'] = 9
```

Listing 7: Feature Age categories.

```
train_df.isnull().values.any()
```

Listing 8: Null check.

```
train_df.drop("id",axis=1,inplace=True)
train_df.drop_duplicates(inplace=True)
```

Listing 9: Drop id and duplicates.

```
X = train_df.drop(columns=["cardio"])
y = train_df[["cardio"]]
```

Listing 10: X drop cardio.

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Listing 11: Scaled copy of data.

```
x_train,x_test,y_train,y_test = train_test_split(X,y,stratify=y,
    test_size=0.2,random_state=0)
x_train1, x_test1, y_train1, y_test1 = train_test_split(X_scaled,y,
    stratify=y,test_size=0.2,random_state=0)
```

Listing 12: Split into train and test on both scaled and unscaled copies.

```
training_score = []
testing_score = []
ff1_score = []
ffbeta_score = []
```

Listing 13: Helping lists.

```
def model_prediction(model):
    model.fit(x_train1,y_train1)
    x_train_pred1 = model.predict(x_train1)
    x_test_pred1 = model.predict(x_test1)
    a = accuracy_score(y_train1,x_train_pred1)
    b = accuracy_score(y_test1,x_test_pred1)
    c = f1_score(y_test1,x_test_pred1)
    d = fbeta_score(y_test1,x_test_pred1,beta=2.0)
    training_score.append(a)
    testing_score.append(b)
    ff1_score.append(c)
    ffbeta_score.append(d)

    print(f"Accuracy_Score of {model} model on Training Data is:",a
    )
    print(f"Accuracy_Score of {model} model on Testing Data is:",b)
    print("\n
    ----------------------------------------------------------------"
    )
    print(f"Precision Score of {model} model is:",precision_score(
    y_test1,x_test_pred1))
    print(f"Recall Score of {model} model is:",recall_score(y_test1
    ,x_test_pred1))
    print(f"F1 Score of {model} model is:",f1_score(y_test1,
    x_test_pred1))
    print("\n
    ----------------------------------------------------------------"
    )
    print(f"Confusion Matrix of {model} model is:")
    cm = confusion_matrix(y_test1,x_test_pred1)
    plt.figure(figsize=(8,4))
    sns.heatmap(cm,annot=True,fmt="g",cmap="Blues")
    plt.show()
```

Listing 14: General model construction for scaled data.

```
In [23]: model_prediction(LogisticRegression())

Accuracy_Score of LogisticRegression() model on Training Data is: 0.7198642372275813
Accuracy_Score of LogisticRegression() model on Testing Data is: 0.7229208345241498

----------------------------------------------------------------------
Precision Score of LogisticRegression() model is: 0.7465590887517798
Recall Score of LogisticRegression() model is: 0.6746247319513938
F1 Score of LogisticRegression() model is: 0.7087714028236708

----------------------------------------------------------------------
Confusion Matrix of LogisticRegression() model is:
```

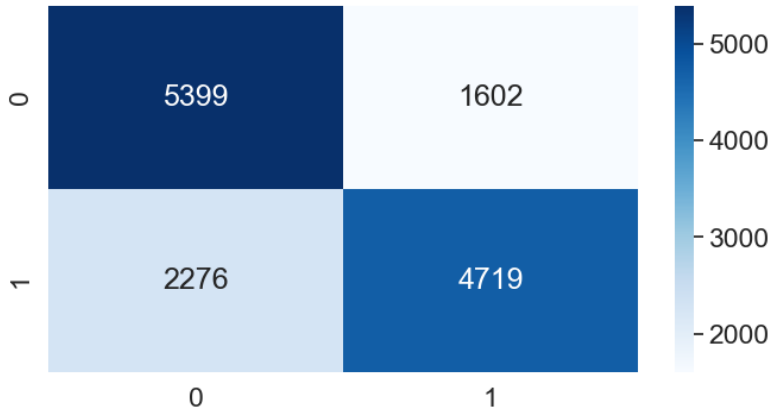


Figure 1: Example of how a model call is shown.

```
model_prediction(LogisticRegression())
model_prediction(KNeighborsClassifier())
model_prediction(SVC())
model_prediction(MLPClassifier())
model_prediction(Perceptron())
model_prediction(PassiveAggressiveClassifier())
model_prediction(RidgeClassifier())
model_prediction(ExtraTreesClassifier())
model_prediction(NearestCentroid())
model_prediction(GaussianNB())
```

Listing 15: Call models which need scaled data.(In different cells)

```
def model_prediction(model):
    model.fit(x_train,y_train)
    x_train_pred = model.predict(x_train)
    x_test_pred = model.predict(x_test)
    a = accuracy_score(y_train,x_train_pred)
    b = accuracy_score(y_test,x_test_pred)
    c = f1_score(y_test,x_test_pred)
    d = fbeta_score(y_test,x_test_pred,beta=2.0)
    training_score.append(a)
    testing_score.append(b)
    ff1_score.append(c)
    ffbeta_score.append(d)

```

```
    print(f"Accuracy_Score of {model} model on Training Data is:",a
    )
    print(f"Accuracy_Score of {model} model on Testing Data is:",b)
    print("\n
    ----------------------------------------------------------"
    )
    print(f"Precision Score of {model} model is:",precision_score(
    y_test,x_test_pred))
    print(f"Recall Score of {model} model is:",recall_score(y_test,
    x_test_pred))
    print(f"F1 Score of {model} model is:",f1_score(y_test,
    x_test_pred))
    print("\n
    ----------------------------------------------------------"
    )
    print(f"Confusion Matrix of {model} model is:")
    cm = confusion_matrix(y_test,x_test_pred)
    plt.figure(figsize=(8,4))
    sns.heatmap(cm,annot=True,fmt="g",cmap="Blues")
    plt.show()
```

Listing 16: General model construction for unscaled data.

```
model_prediction(DecisionTreeClassifier())
model_prediction(RandomForestClassifier())
model_prediction(AdaBoostClassifier())
model_prediction(GradientBoostingClassifier())
model_prediction(LGBMClassifier())
model_prediction(XGBClassifier())
```

Listing 17: Call models which don't need scaled data.(In different cells)

```
model1 = MLPClassifier()
model1.fit(x_train,y_train)
model2 = Perceptron()
model2.fit(x_train,y_train)
model3 = GradientBoostingClassifier()
model3.fit(x_train,y_train)
model4 = SVC()
model4.fit(x_train,y_train)
model5 = LGBMClassifier()
model5.fit(x_train,y_train)
model6 = XGBClassifier()
model6.fit(x_train,y_train)
model7 = RandomForestClassifier()
model7.fit(x_train,y_train)
model8 = LogisticRegression()
model8.fit(x_train,y_train)
```

Listing 18: Making a pool of learners for next models.(In different cells)

```
pool_classifiers = [model1, model2, model3,model4, model5,
model6,model7,model8]
```

Listing 19: Pool of classifiers for DESP KNORAU KNORAE.

```
desp = DESP(pool_classifiers)
knorau = KNORAU(pool_classifiers)
kne = KNORAE(pool_classifiers)
```

Listing 20: Models

```
Desp = model_prediction(desp)
Knorau = model_prediction(knorau)
Kne = model_prediction(kne)
```

Listing 21: Call models .(In different cells)

```
stacking_model2 = StackingClassifier(estimators=
[('MLPClassifier', model1),
('GradientBoostingClassifier', model3),
('LGBMClassifier', model5)])
```

Listing 22: Stacking model .

```
stacking_model2.fit(x_train, y_train)
x_train_pred5 = stacking_model2.predict(x_train)
x_test_pred5 = stacking_model2.predict(x_test)
```

Listing 23: Stacking model (In different cells).

```
cm = confusion_matrix(y_test,x_test_pred5)
plt.figure(figsize=(8,4))
sns.heatmap(cm,annot=True,fmt="g",cmap="Blues")
plt.show()
```

Listing 24: Stacking model confusion matrix.

```
 a = accuracy_score(y_train,x_train_pred5)
    b = accuracy_score(y_test,x_test_pred5)
    c = f1_score(y_test,x_test_pred5)
    d = fbeta_score(y_test,x_test_pred5,beta=2.0)
    training_score.append(a)
    testing_score.append(b)
    ff1_score.append(c)
    ffbeta_score.append(d)
```

Listing 25: Put on list.

```
stacking_model3 = VotingClassifier(estimators=[('MLPClassifier',
    model1),
('GradientBoostingClassifier', model3),
('SVC', model4),
('LGBMClassifier', model5),
('XGBClassifier', model6)],voting='hard')
```

Listing 26: Voting model .

```
stacking_model3.fit(x_train, y_train)
x_train_pred6 = stacking_model3.predict(x_train)
x_test_pred6 = stacking_model3.predict(x_test)
```

Listing 27: Voting model (In different cells).

```
cm = confusion_matrix(y_test,x_test_pred6)
plt.figure(figsize=(8,4))
sns.heatmap(cm,annot=True,fmt="g",cmap="Blues")
plt.show()
```

Listing 28: Votting model confusion matrix.

```
    a = accuracy_score(y_train,x_train_pred6)
    b = accuracy_score(y_test,x_test_pred6)
    c = f1_score(y_test,x_test_pred6)
    d = fbeta_score(y_test,x_test_pred6,beta=2.0)
    training_score.append(a)
    testing_score.append(b)
    ff1_score.append(c)
    ffbeta_score.append(d)
```

Listing 29: Put on list.

```
models2 = ["Logistic Regression","KNN","SVC","MLPClassifier","
    Perceptron","PassiveAggressiveClassifier","RidgeClassifier","
    ExtraTreesClassifier","NearestCentroid","Gausian","Decision
    Tree","Random Forest","Ada Boost",
"Gradient Boost","LGBM","XGBoost","Desp","Knorau","Kne","
    StackingClassifier","VotingClassifier"]
```

Listing 30: Making final list.

```
df = pd.DataFrame({"Algorithms":models2,
                   "Training Score":training_score,
                   "Testing Score (Accuracy)":testing_score,
                   "F1 Score":ff1_score,
                   "F2 Score":ffbeta_score,
                    })
```

Listing 31: Making final list.

```
df = df.sort_values(by='F2 Score', ascending=False)
```

Listing 32: Making final list.

```
df.plot(x="Algorithms",y=["F2 Score"], figsize=(16,6),kind="bar",
        title="Performance Visualization of Different Models (
    Ranked by accuracy)",colormap="Set1")
plt.show()
```

Listing 33: Plot bars ranked f2.

```
line = px.line(data_frame= df , x="Algorithms", y =["F2 Score","
    Testing Score (Accuracy)" ] , markers = True)
line.update_xaxes(title="Model",
                  rangeslider_visible = False)
line.update_yaxes(title = "F2 SCORE")
line.update_layout(showlegend = True,
    title = {
        'text': 'F2 Score vs Model',
        'y':0.94,
        'x':0.5,
```

```
        'xanchor': 'center',
        'yanchor': 'top'})

line.show()
```

Listing 34: Plot line f2 and testing score.

```
df = df.sort_values(by='Testing Score (Accuracy)', ascending=False)
```

Listing 35: Sort by accuracy.

```
df.plot(x="Algorithms",y=["Training Score","Testing Score (Accuracy
    )"], figsize=(16,6),kind="bar",
title="Performance Visualization of Different Models (Ranked by
    accuracy)",colormap="Set1")
plt.show()
```

Listing 36: Plot bars Train an Test score.

```
df.plot(x="Algorithms",y=["Testing Score (Accuracy)"], figsize
    =(16,6),kind="bar",
title="Performance Visualization of Different Models (Ranked by
    accuracy)",colormap="Set1")
plt.show()
```

Listing 37: Plot bars Test score.

7 Επίλογος

Μελετώντας τα αποτελέσματα αντιλαμβανόμαστε ότι τα ομαδικά σχήματα μάθησης είναι ιδιαίτερα χρήσιμα και έχουν αρκετά καλή απόδοση στα ιατρικά δεδομένα. Ιδιαίτερα ο Voting Classifier και ο Stacking Classifier είναι στην κορυφή της απόδοσης σε όλες τις μετρικές τόσο σε μικρά όσο και σε μεγάλα Dataset. Επίσης αντιλαμβανόμαστε ότι η επιλογή μετρικής για να συγκρίνουμε τους αλγορίθμους παρότι φαινομενικά μικρής σημασίας είναι ίσως απο τα πιο σημαντικά κομμάτια καθώς μπορεί να επιφέρει τελείως διαφορετικά αποτελέσματα.

Βιβλιογραφία

[1] A. Burkov. “The Hundred-Page Machine Learning Book”. In: (2019), p. 20. URL: https://themlbook.com/.

[2] IBM. “What is a Decision Tree?” In: (2024). URL: https://www.ibm.com/topics/decision-trees.

[3] Wikipedia. “Artificial neural network”. In: (2024). URL: https://el.wikipedia.org/wiki/%CE%9D%CE%B5%CF%85%CF%81%CF%89%CE%BD%CE%B9%CE%BA%CF%8C_%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF.

[4] M. Kalirane. “Gradient Descent vs. Backpropagation: What’s the Difference?” In: (2024). URL: https://www.analyticsvidhya.com/blog/2023/01/gradient-descent-vs-backpropagation-whats-the-difference/.

[5] Wikipedia. “Bayes’ theorem”. In: (2024). URL: https://en.wikipedia.org/wiki/Bayes%27_theorem.

[6] Yang. “An Introduction to Naïve Bayes Classifier”. In: (2024). URL: https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf.

[7] Wikipedia. “Ridge regression”. In: (2024). URL: https://en.wikipedia.org/wiki/Ridge_regression.

[8] Sklearn. “Support Vector Machines”. In: (2024). URL: https://scikit-learn.org/stable/modules/svm.html.

[9] M.-E. OGBEMI. “What is Overfitting in Machine Learning?” In: (2024). URL: https://www.freecodecamp.org/news/what-is-overfitting-machine-learning/.

[10] Wikipedia. “Mean squared error”. In: (2024). URL: https://en.wikipedia.org/wiki/Mean_squared_error.

[11] B. Soni. “Understanding Boosting in Machine Learning: A Comprehensive Guide”. In: (2024). URL: https://medium.com/@brijesh_soni/understanding-boosting-in-machine-learning-a-comprehensive-guide-bdeaa1167a6.

[12] masters in datascience. “What Is Bootstrapping?” In: (2024). URL: https://www.mastersindatascience.org/learning/machine-learning-algorithms/bootstrapping/.

[13] Sklearn. “DecisionTreeClassifier”. In: (2024). URL: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[14] Sklearn. “GaussianNB”. In: (2024). URL: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html.

[15] Sklearn. “SVC”. In: (2024). URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[16] Sklearn. “Logistic”. In: (2024). URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

[17] Sklearn. “Perceptron”. In: (2024). URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html.

[18] Sklearn. “MLPClassifier”. In: (2024). URL: https://scikit-learn.org/stable/modules/neural_networks_supervised.html.

[19] Sklearn. “Passive”. In: (2024). URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.PassiveAggressiveClassifier.html.

[20] Sklearn. “RidgeClassifier”. In: (2024). URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html.

[21] Sklearn. “KneighborsClassifier”. In: (2024). URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[22] Sklearn. “Nearest Neighbors”. In: (2024). URL: https://scikit-learn.org/stable/modules/neighbors.html.

[23] Sklearn. “NearestCentroid”. In: (2024). URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestCentroid.html.

[24] lightgbm. “LGBMClassifier”. In: (2024). URL: https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html.

[25] P. BANERJEE. “LightGBM Classifier in Python”. In: (2024). URL: https://www.kaggle.com/code/prashant111/lightgbm-classifier-in-python.

[26] IBM. “What is random forest?” In: (2024). URL: https://www.ibm.com/topics/random-forest.

[27] Sklearn. “ExtraTreesClassifier”. In: (2024). URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html.

[28] A. Wiki. “Gradient Boosting”. In: (2024). URL: https://machine-learning.paperspace.com/wiki/gradient-boosting.

[29] T. Zhang et al. “Improving Convection Trigger Functions in Deep Convective Parameterization Schemes Using Machine Learning”. In: *Journal of Advances in Modeling Earth Systems* 13 (May 2021). DOI: 10.1029/2020MS002365.

[30] Sklearn. “AdaBoostClassifier”. In: (2024). URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html.

[31] V. Alto. “Understanding AdaBoost for Decision Tree”. In: (2024). URL: https://towardsdatascience.com/understanding-adaboost-for-decision-tree-ff8f07d2851.

[32] dmlc XGBoost. “XGBoost”. In: (2024). URL: https://xgboost.readthedocs.io/en/latest/index.html.

[33] Sklearn. “StackingClassifier”. In: (2024). URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html.

[34] deslib. “Dynamic Ensemble Selection (DES)”. In: (2024). URL: https://deslib.readthedocs.io/en/latest/api.html#dynamic-ensemble-selection-des.

[35] R. M. O. Cruz et al. “DESlib: A Dynamic ensemble selection library in Python”. In: *Journal of Machine Learning Research* 21.8 (2020), pp. 1–5. URL: http://jmlr.org/papers/v21/18-144.html.

[36] deslib. “k-Nearest Oracle Union (KNORA-U)”. In: (2024). URL: https://deslib.readthedocs.io/en/latest/modules/des/knora_u.html.

[37] deslib. “k-Nearest Oracle-Eliminate (KNORA-E)”. In: (2024). URL: https://deslib.readthedocs.io/en/latest/modules/des/knora_e.html.

[38] deslib. “Dynamic Ensemble Selection performance (DES-P)”. In: (2024). URL: https://deslib.readthedocs.io/en/latest/modules/des/des_p.html.

[39] M. Little. *Parkinsons*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C59C74. 2008.