ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΡΑΙΩΣ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Πρόγραμμα Μεταπτυχιακών Σπουδών

“Πληροφοριακά Συστήματα & Υπηρεσίες”

ΕΙΔΙΚΕΥΣΗ : Μεγάλα Δεδομένα & Αναλυτική

Ανάλυση κλινικών δεδομένων με χρήση αλγορίθμων εξόρυξης δεδομένων για την πρόβλεψη του διαβήτη

Βασιλική Σιδερή ΜΕ2228

Επιβλέπων Καθηγητής : Μιχαήλ Φιλιππάκης

Φεβρουάριος 2024

Περίληψη

Σύμφωνα με στατιστικές, μελέτες καθώς και την ανάπτυξη της ιατρικής με νέα μηχανήματα και τεχνολογικά μέσα τα οποία καθημερινά συμβάλλουν στην καλύτερη ποιότητα ζωής και άμεση ίαση των ασθενών αποδεικνύεται πως η μηχανική μάθηση έχει επιφέρει σημαντικές εξελίξεις στον τομέα της υγείας και της ιατρικής. Έχει εφαρμοστεί σε πολλούς τομείς για τη βελτίωση της διάγνωσης, της πρόληψης, της θεραπείας και της παροχής υγειονομικής περίθαλψης. Ανάμεσα στις εφαρμογές της μηχανικής μάθησης στην υγεία συμπεριλαμβάνονται η εξατομικευμένη θεραπεία δηλαδή η ανάλυση γενετικών δεδομένων για τη δημιουργία εξατομικευμένων θεραπειών και φαρμάκων για ειδικές περιπτώσεις ασθενών, η διαχείριση υγείας που μπορεί να οδηγήσει στη βέλτιστη διαχείριση των νοσοκομειακών πόρων καθώς και στην ψηφιακή υγεία και στους φορείς υγείας όπως για παράδειγμα η διαχείριση των ιατρικών αρχείων. Αυτό τη κατατάσσει σε ένα πάρα πολύ σημαντικό και καίριο τομέα που ολοένα και θέλουμε σαν κοινωνία να αναπτύσσεται και να εξελίσσεται.

Η συγκεκριμένη διπλωματική εργασία αφορά τη συγκριτική μελέτη αλγορίθμων μηχανικής μάθησης, καθώς και την αξιολόγηση της επίδοσης αυτών για τη διάγνωση του διαβήτη μέσω του συνόλου των κλινικών δεδομένων Diabetes prediction dataset.

Για κάθε ένα από τα μοντέλα μηχανικής μάθησης αναλύσαμε διάφορες μετρικές ώστε να καταλήξουμε στο συμπέρασμα ποιο από αυτά μας προσέφερε το βέλτιστο αποτέλεσμα για το σύνολο των δεδομένων μας.

Summary

According to statistics, studies and the development of medicine with new machines and technological tools that daily contribute to a better quality of life and immediate healing of patients, it is proven that machine learning has brought significant developments in the field of health and medicine. It has been applied in many areas to improve diagnosis, prevention, treatment and healthcare delivery. Among the applications of machine learning in healthcare include personalised therapy i.e. the analysis of genetic data to create personalised treatments and medicines for specific patient cases, health management which can lead to optimal management of hospital resources as well as digital health and healthcare providers such as medical records management. This places it in a very, very important and key area that we increasingly want as a society to develop and evolve.

This thesis is about the comparative study of machine learning algorithms and the evaluation of their performance for diabetes diagnosis using the Diabetes prediction dataset clinical dataset.

For each of the machine learning models, we analyzed several metrics to conclude which one provided the best result for our dataset.

Ευχαριστίες

Για την υλοποίηση της διπλωματικής μου εργασίας θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου κο. Μιχαήλ Φιλιππάκη καθηγητή του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς, τόσο για την επίβλεψη και την καθοδήγηση του σε όλο το διάστημα εκπόνησης της διπλωματικής εργασίας, όσο και για τη πολύτιμη βοήθεια του ως προς την ενθάρρυνση προς το πρόσωπο μου.

Επίσης θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για την αμέριστη συμπαράσταση, κατανόηση και στήριξη τους και εύχομαι να συνεχίσουν να το κάνουν και στις επόμενες επιλογές μου όποιες κι αν πρόκειται να είναι αυτές.

Πίνακας Περιεχομένων

Περίληψη 2
Summary 3
Ευχαριστίες 4
Πίνακας Περιεχομένων 5
Λίστα γραφημάτων & εικόνων 7
Λίστα πινάκων 8
1. Εισαγωγή & Ορισμός του προβλήματος 9
1.1 Εισαγωγή 9
1.2 Δομή της διπλωματικής εργασίας 9
1.3 Συνεισφορά της μεταπτυχιακής διπλωματικής εργασίας 10
1.4 Ορισμός προβλήματος - Σακχαρώδης Διαβήτης 10
1.5 Τύποι σακχαρώδη διαβήτη 10
1.5 Διάγνωση σακχαρώδη διαβήτη 11
1.6 Τρόποι αντιμετώπισης 12
2. Περιγραφή Μηχανικής μάθησης & αλγορίθμων εξόρυξης δεδομένων 13
2.1 Η ανάγκη ύπαρξης της μηχανικής μάθησης & των αλγορίθμων εξόρυξης δεδομένων 13
2.2 Αλγόριθμος Μηχανών Διανυσματικής υποστήριξης (Support Vector Machines) 13
2.3 Αλγόριθμος Κ-πλησιέστερων γειτόνων (K-Nearest Neighbors) 14
2.4 Αλγόριθμος Τυχαία δάση (Random Forest) 17
2.5 Αλγόριθμος Λογιστικής Παλινδρόμησης (Logistic Regression) 19
2.6 Αλγόριθμος Δέντρου Απόφασης (Decision Tree) 20
2.7 Μέτρα ακρίβειας & απόδοσης 22
2.7.1 Ακρίβεια/Πρόβλεψη 22
2.7.2 Ευαισθησία 22
2.7.3 Πίνακας πιθανοτήτων (Confusion matrix) 23
2.7.4 F1-Score 24
2.7.5 Receiver Operating Characteristic (ROC) 24
2.7.6 Area under the curve (AUC) 25
3. Αποτελέσματα 27
3.1 Δεδομένα 27
3.2 Αποτελέσματα των αλγορίθμων 28
4. Συμπεράσματα 33
4.1 Συμπεράσματα ανάλυσης αλγορίθμων 33

4.2 Μελλοντικές βελτιώσεις 33
Βιβλιογραφία 35
Παράρτημα - Κώδικας 36
Εφαρμογή Random Forest 36
Εφαρμογή KNN 40
Εφαρμογή SVM 44
Εφαρμογή Logistic Regression 48
Εφαρμογή Decision Tree 52
Συγκρίσεις μοντέλων ως προς διάφορες μετρικές 56

Λίστα γραφημάτων & εικόνων

Γράφημα 2.1 α, β 14
Γράφημα 2.2 Ταξινόμηση δεδομένου με βάση τον KNN 15
Γράφημα 2.3 Ευκλείδεια απόσταση δύο σημείων 16
Γράφημα 2.4 Ταξινόμηση νέου στοιχείου 16
Γράφημα 2.5 Ανάλυση λειτουργίας αλγορίθμου τυχαίων δασών 17
Γράφημα 2.6 Παράδειγμα ταξινόμησης φρούτων σε κλάσεις με τη μέθοδο τυχαίων δασών 18
Γράφημα 2.7 Παράδειγμα ταξινόμησης με τη μέθοδο της λογιστικής παλινδρόμησης 19
Γράφημα 2.8 Διάγραμμα ταξινόμησης με τη μέθοδο δέντρου απόφασης 21
Γράφημα 2.9 Παράδειγμα ταξινόμησης με τη μέθοδο δέντρου απόφασης 21
Γράφημα 2.10 Confusion matrix 23
Γράφημα 2.11 Απεικόνιση του Confusion Matrix 23
Γράφημα 2.12 Μέτρα ανάλυσης Confusion Matrix 24
Γράφημα 2.13 Γράφημα ROC 25
Γράφημα 2.14 Υπολογισμός μετρικής AUC 26
Γράφημα 3.1 Ακρίβεια KNN 29
Γράφημα 3.2 Ακρίβεια Random Forest 29
Γράφημα 3.3 Ακρίβεια SVM 29
Γράφημα 3.4 Ακρίβεια Decision Tree 29
Γράφημα 3.5 Ακρίβεια Logistic Regression 29
Γράφημα 3.6 Σύγκριση μεθόδων ως προς την ακρίβεια 31
Γράφημα 3.7 Σύγκριση μεθόδων ως προς F1 Score 31
Γράφημα 3.8 Σύγκριση μεθόδων ως προς χρόνο εκτέλεσης 31
Γράφημα 3.9 Σύγκριση μεθόδων ως προς MAE και RMSE .....

Λίστα πινάκων

Πίνακας 3.1 Ορισμός των μεταβλητών diabetes_prediction dataset 27
Πίνακας 3.2 Αποτελέσματα αλγορίθμων 28
Πίνακας 3.3 Τιμές ταξινομητών 30

1. Εισαγωγή & Ορισμός του προβλήματος

1.1 Εισαγωγή

Η εφαρμογή της μηχανικής μάθησης στον τομέα του διαβήτη αλλά και άλλων τομέων της ιατρικής και ασθενειών αντιπροσωπεύει ένα σημαντικό συνδυασμό της προηγμένης τεχνολογίας με την υγειονομική πράξη. Με το διαβήτη να αυξάνει την επίπτωση του σε παγκόσμιο επίπεδο, η ανάγκη για προηγμένες και αποτελεσματικές προσεγγίσεις στον τομέα της διάγνωσης, της πρόληψης και της διαχείρισης είναι πιο επείγουσα από ποτέ. Η μηχανική μάθηση, σε συνδυασμό με την ανάλυση των υγειονομικών δεδομένων, αναδεικνύει μια πολλά υποσχόμενη προσέγγιση που επιτρέπει την εξατομίκευση της περίθαλψης, την πρόβλεψη επιπλοκών και τη βελτίωση της ποιότητας ζωής των ατόμων με διαβήτη.

Στο επίκεντρο αυτής της συνεισφοράς βρίσκεται η ικανότητα των μοντέλων μηχανικής μάθησης να αντλούν σημαντική πληροφορία από πολύπλοκα δεδομένα, όπως τα υγειονομικά ιστορικά, τα επίπεδα γλυκόζης, τη διατροφή και την άσκηση. Η δυνατότητα πρόβλεψης πιθανών επιπλοκών, η αυτόματη προσαρμογή των θεραπευτικών προσεγγίσεων καθώς και η παροχή εξατομικευμένων συμβουλών για τον τρόπο ζωής αποτελούν κρίσιμα κομμάτια που συνεισφέρουν στην αντιμετώπιση του διαβήτη.

Συνολικά η συμβολή της μηχανικής μάθησης στον τομέα του διαβήτη ανοίγει νέους ορίζοντες για την προοπτική της προσωποποιημένης, αποτελεσματικής και προληπτικής υγειονομικής περίθαλψης.

1.2 Δομή της διπλωματικής εργασίας

Η διπλωματική εργασία αποτελείται από τέσσερα κεφάλαια και τις επιμέρους ενότητες που περιέχονται σε κάθε ένα από αυτά.

Στην εισαγωγή αναλύεται η αξία της μηχανικής μάθησης γενικότερα στον τομέα της υγείας και ειδικότερα στη συνεισφορά της αντιμετώπισης του διαβήτη. Στη συνέχεια γίνεται αναφορά σχετικά με το διαβήτη , τα είδη του, η σημαντικότητα της διάγνωσης του και τρόποι αντιμετώπισης αυτού.

Στο δεύτερο κεφάλαιο γίνεται λεπτομερής ανάλυση των τριών αλγορίθμων μηχανικής μάθησης που χρησιμοποιούνται στην υλοποίηση της εργασίας. Οι αλγόριθμοι αυτοί είναι ο Κ πλησιέστεροι γείτονες (KNN), οι μηχανές διανυσμάτων υποστήριξης (SVM) και τα τυχαία δάση αποφάσεων Random Forest (RF).

Στο τρίτο κεφάλαιο παρουσιάζονται τα αποτελέσματα που προέκυψαν από τους τρεις αλγορίθμους ως προς την ακρίβεια καθώς και άλλες μετρικές συνδυαστικά ώστε να

πιστοποιηθεί χωρίς αμφιβολία ότι ο αλγόριθμος που επιλέξαμε ως βέλτιστος είναι ο πιο αποτελεσματικός για το σύνολο δεδομένων μας.

Στο τέταρτο κεφάλαιο παρουσιάζονται τα συμπεράσματα που προέκυψαν από τα αποτελέσματα των αλγορίθμων καθώς αναφέρονται και μελλοντικές βελτιώσεις όσων αφορά το σύνολο των δεδομένων και την εφαρμογή των αλγορίθμων.

1.3 Συνεισφορά της μεταπτυχιακής διπλωματικής εργασίας

Παρόλο την ύπαρξη ανάλογων ερευνών, η συγκεκριμένη διπλωματική περιέχει την εφαρμογή και ανάλυση αλγορίθμων σε κλινικά δεδομένα και συμβάλει στον εντοπισμό του καλύτερου αλγορίθμου από αυτούς που αναλύονται για την πρόβλεψη του διαβήτη.

Με αυτόν τον τρόπο, η διπλωματική εργασία σας προσφέρει νέα φώτα στην εξέλιξη των διαγνωστικών πρακτικών και στη βελτίωση της προσωποποιημένης περίθαλψης για ασθενείς με διαβήτη.

1.4 Ορισμός προβλήματος - Σακχαρώδης Διαβήτης

Ο Σακχαρώδης διαβήτης αποτελεί ένα παγκόσμιο πρόβλημα υγείας με σοβαρές επιπτώσεις στον ανθρώπινο οργανισμό. Ο σακχαρώδης διαβήτης είναι μία μεταβολική ασθένεια η οποία αυξάνει τη συγκέντρωση του σακχάρου στο αίμα και διαταράσσει τον μεταβολισμό της γλυκόζης με αποτέλεσμα είτε την ελαττωμένη έκκριση της ινσουλίνης είτε την ελάττωση της ευαισθησίας των κυττάρων του σώματος στην ινσουλίνη.

Τον 20ο αιώνα ο διαβήτης κυριαρχούσε στις πιο περιορισμένες οικονομικά περιοχές, όμως τις τελευταίες δεκαετίες παρατηρείται αύξηση των περιστατικών διαβήτη σε παγκόσμιο επίπεδο. Σήμερα εκατομμύρια άνθρωποι ζουν με διαβήτη ενώ πολλοί ακόμη διαγιγνώσκονται κάθε χρόνο. Σύμφωνα με προβλέψεις του Διεθνούς Διαβητολογικού Συνδέσμου περίπου 700 εκατομμύρια άνθρωποι αναμένεται να ζουν με διαβήτη έως το 2045. Η αύξηση αυτή οφείλεται σε παράγοντες όπως η γήρανση του πληθυσμού, η αύξηση του αριθμού των αστικών πληθυσμών και οι αλλαγές στον τρόπο ζωής όπως η κακή διατροφή και η έλλειψη της φυσικής δραστηριότητας, η κληρονομικότητα, η παχυσαρκία, η αύξηση του στρες και η έλλειψη ύπνου.

Ο διαβήτης συνεπώς αποτελεί μία παγκόσμια πρόκληση για τη δημόσια υγεία και η έγκαιρη διάγνωση και πρόληψη του είναι καίριο παγκόσμιο ζήτημα.

1.5 Τύποι σακχαρώδη διαβήτη

Ο σακχαρώδης διαβήτης διακρίνεται στο σακχαρώδη διαβήτη τύπου Ι , στο σακχαρώδη διαβήτη τύπου ΙΙ, στο διαβήτη κυήσεως και σε άλλους ειδικούς και σπάνιους τύπους. Οι κύριοι και πιο συνηθισμένοι τύποι είναι ο τύπος Ι και ΙΙ.

Ο σακχαρώδης διαβήτης τύπου Ι προκαλείται από την καταστροφή των Β κυττάρων του παγκρέατος τα οποία υπό φυσιολογικές συνθήκες παράγουν ινσουλίνη. Έτσι σε αυτόν τον τύπο παρατηρείται πλήρης έλλειψη ινσουλίνης και για τη θεραπεία αυτού είναι απαραίτητη η εξωγενής χορήγηση της ινσουλίνης. Αφορά το 5-10% των περιπτώσεων του διαβήτη και εκδηλώνεται με έντονα συμπτώματα όπως πολυουρία, πολυδιψία, πολυφαγία και απώλεια βάρους.

Ο σακχαρώδης διαβήτης τύπου ΙΙ προκαλείται από συνδυασμό διαταραχής της έκκρισης αλλά και της δράσης της ινσουλίνης. Δηλαδή ο οργανισμός δεν χρησιμοποιεί την ινσουλίνη αποτελεσματικά ή δεν παράγει αρκετή. Αφορά το 90% των περιπτώσεων, συνήθως εμφανίζεται σε ενήλικες και δεν υπάρχουν συμπτώματα για μεγάλο χρονικό διάστημα αλλά στο διάστημα αυτό που ονομάζεται προδιαβήτης ξεκινούν οι επιπλοκές της νόσου και τα συμπτώματα που εμφανίζονται είναι πολυουρία, πολυδιψία, μυκητιασικές λοιμώξεις και άλλες δερματοπάθειες.

Ο διαβήτης κυήσεως αναπτύσσεται κατά τη διάρκεια της εγκυμοσύνης και αφορά γυναίκες που παρουσιάζουν υψηλά επίπεδα γλυκόζης στο αίμα κατά τη διάρκεια της κύησης. Ορισμένες γυναίκες αντιμετωπίζουν δυσκολίες στην παραγωγή της απαραίτητης ποσότητας ινσουλίνης για να αντιμετωπίσουν την αυξημένη ανάγκη του σώματος κατά την κύηση. Η ασφαλής διαχείριση του είναι ο διατροφικός περιορισμός, η συχνή παρακολούθηση των επιπέδων της ινσουλίνης στο αίμα και σε κάποιες περιπτώσεις και η χορήγηση γλυκόζης.

Τέλος οι άλλοι ειδικοί τύποι διαβήτη που υπάρχουν είναι ο διαβήτης τύπου 1.5 πρόκειται για ένα σπάνιο τύπο διαβήτη που εμφανίζεται σε νεαρή ηλικία και είναι συνήθως από κληρονομικότητα. Ο διαβήτης LADA είναι μία μορφή διαβήτη που μοιάζει με τον τύπο Ι, το ανοσοποιητικό σύστημα επιτίθεται στα κύτταρα παραγωγής της ινσουλίνης αλλά η εξέλιξη της νόσου είναι πιο αργή συγκριτικά με τον τύπο Ι. Ο διαβήτης νεογνών είναι επίσης μία σπάνια μορφή διαβήτη που εμφανίζεται στα βρέφη και οφείλεται σε γενετικές αλλαγές και τέλος υπάρχει και ο διαβήτης τύπου ΙΙΙ τον συναντάμε κατά τη διάρκεια της κύησης και σχετίζεται με την παραγωγή ή την αντίσταση του ορμονικού αργινίνης βασοπρεσσίνης μιας ορμόνης που αυξάνει την πίεση του αίματος.

1.5 Διάγνωση σακχαρώδη διαβήτη

Η διάγνωση του σακχαρώδη διαβήτη και μάλιστα η έγκαιρη διάγνωση του αποτελεί καίριο ζήτημα διότι προφυλάσει τους ασθενείς από τη πρόκληση σοβαρότερων νοσημάτων που μπορούν να προκληθούν από τη καθυστέρηση διάγνωσης του. Η διάγνωση γίνεται με τη μέτρηση του πρωινού σακχάρου αίματος ή των τιμών των σακχάρων κατά τη διάρκεια της καμπύλης γλυκόζης. Συγκεκριμένα η τιμή του σακχάρου νηστείας όταν ξεπερνά τα 126 mg/dl ή σε μία τυχαία μέτρηση ξεπερνά τα 200 mg/dl και συνυπάρχουν συμπτώματα τότε λέμε ότι κάποιος πάσχει από σακχαρώδη διαβήτη.

Τα τελευταία χρόνια η Αμερικανική Διαβητολογική Εταιρεία προτείνει ως μέθοδο διάγνωσης την μέτρηση της γλυκοζυλιωμένης αιμοσφαιρίνης (HbA1c). Η τιμή της HbA1c εάν είναι μεγαλύτερη από 6.5 υποδηλώνει την διάγνωση του διαβήτη. Φυσικά εάν η τιμή της HbA1c είναι μικρότερη από 6.5 δεν αποκλείει ότι ο ασθενής πάσχει από διαβήτη, συνυπολογίζεται και από άλλες μετρήσεις σε κλινικά δεδομένα του ασθενούς.

Τα άτομα τα οποία πρέπει να ελέγχονται είναι τα ασυμπτωματικά άτομα, τα μη παχύσαρκα και χωρίς παράγοντες κινδύνου οφείλουν να ελεγχθούν για σακχαρώδη διαβήτη στην ηλικία των 45 ετών. Νωρίτερα οφείλουν να ελέγχονται τα άτομα τα οποία είναι υπέρβαρα ή παχύσαρκα και έχουν κάποιο επιπλέον παράγοντα κινδύνου όπως η έλλειψη άσκησης , ιστορικό οικογενείας με διαγνωσμένο άτομο με σακχαρώδη διαβήτη, άτομα με υπέρταση και με καρδιαγγειακή νόσο. Επίσης πρέπει να ελέγχονται τακτικά οι υπέρβαρες/παχύσαρκες μητέρες που είχαν διαγνωστεί με διαβήτη κυήσεως ή έχουν γεννήσει παχύσαρκα μωρά.

1.6 Τρόποι αντιμετώπισης

Η αντιμετώπιση του διαβήτη συνήθως περιλαμβάνει συνδυασμό διατροφικών, φυσικών, φαρμακευτικών και εκπαιδευτικών προσεγγίσεων. Ανάλογα με τον τύπο του διαβήτη οι προσεγγίσεις μπορεί να διαφέρουν. Τρόποι παρέμβασης στον διαβήτη είναι οι εξής:

1. Υγιεινός τρόπος διατροφής και διατροφικοί περιορισμοί

- Μείωση του σωματικού βάρους κατά τουλάχιστον 5%, εφόσον είναι αυξημένο
- Μείωση του ολικού λίπους μικρότερο του 30% της ημερήσιας θερμιδικής πρόσληψης
- Μείωση του κορεσμένου λίπους μικρότερη από 10% της ενεργειακής πρόσληψης
- Αύξηση της πρόσληψης φυτικών ινών (τουλάχιστον 25-30g ημερησίως)

2. Γυμναστική

- Αερόβια άσκηση τουλάχιστον 30 λεπτά ημερησίως, το λιγότερο 5 φορές εβδομαδιαίως
- Περπάτημα, κολύμπι, τρέξιμο, ποδήλατο

Έχει αποδειχθεί βάση μελετών ότι η απώλεια βάρους σε υπέρβαρους και παχύσαρκους 5-10% μπορεί να ελαττώσει την πιθανότητα εμφάνισης διαβήτη ή να τον αναστρέψει. Παράλληλα η ένταξη της άσκησης τουλάχιστον 30 λεπτά τη μέρα στην καθημερινότητα μπορεί να συμβάλλει στη μείωση κατά 58% του κινδύνου εμφάνισης του διαβήτη.

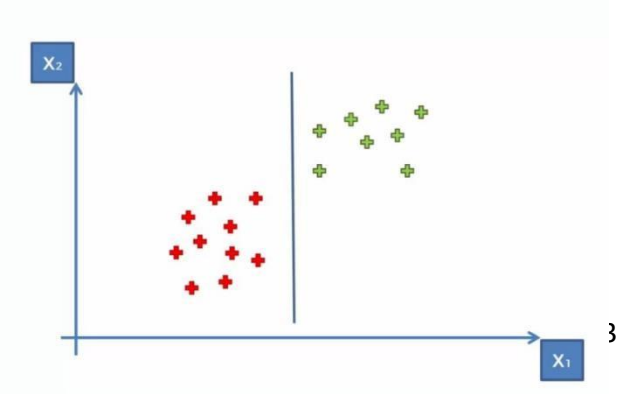
2. Περιγραφή Μηχανικής μάθησης & αλγορίθμων εξόρυξης δεδομένων

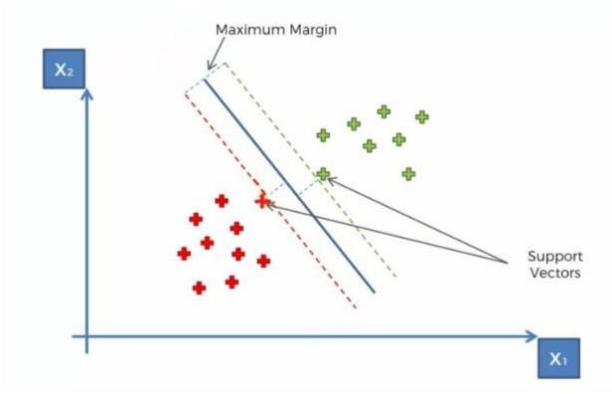
2.1 Η ανάγκη ύπαρξης της μηχανικής μάθησης & των αλγορίθμων εξόρυξης δεδομένων

Η Μηχανική μάθηση επικεντρώνεται στη διδασκαλία των υπολογιστών να μαθαίνουν από τα δεδομένα και να βελτιώνονται με την εμπειρία αντί να είναι ρητά προγραμματισμένοι να το κάνουν . Οι αλγόριθμοι εκπαιδεύονται σε μοτίβα και συσχετίσεις σε μεγάλα σύνολα δεδομένων και λαμβάνουν τις καλύτερες αποφάσεις και προβλέψεις με βάση αυτή την ανάλυση. Η εξόρυξη δεδομένων (Data mining) είναι ένα πεδίο έρευνας το οποίο συνδυάζει διάφορες τεχνικές από τη μηχανική μάθηση , τη στατιστική, τις βάσεις δεδομένων, την οπτικοποίηση και την ανάπτυξη αλγορίθμων. Η ολοένα και μεγαλύτερη αύξηση των δεδομένων σε όλους τους τομείς , η ανάγκη αποθήκευσης και επεξεργασίας αυτών καθώς και της διαθεσιμότητας και αύξησης της πρόσβασης στα δεδομένα λόγω του παγκόσμιου ιστού και των μεγάλων δικτύων ενεργοποιεί την ανάγκη για ανάπτυξη στον τομέα του της εξόρυξης δεδομένων και της μηχανικής μάθησης.

2.2 Αλγόριθμος Μηχανών Διανυσματικής υποστήριξης (Support Vector Machines)

Ο αλγόριθμος Μηχανών διανυσματικής υποστήριξης χρησιμοποιείται για προβλήματα ταξινόμησης κατά κύριο λόγο μηχανικής μάθησης και παλινδρόμησης. Στόχος του είναι να δημιουργήσει ένα υπερεπίπεδο που διαχωρίζει τα δεδομένα σε δύο κλάσεις με τον μεγαλύτερο δυνατό τρόπο. Η εύρεση του υπερεπιπέδου που μεγιστοποιεί το περιθώριο μεταξύ αυτών των κλάσεων. Το περιθώριο αναφέρεται στην απόσταση μεταξύ του υπερεπιπέδου και των πλησιέστερων δειγμάτων κάθε κλάσης. Η μεγιστοποίηση αυτού του περιθωρίου οδηγεί σε μεγαλύτερη και καλύτερη ικανότητα γενίκευσης σε νέα δεδομένα. Ένα σημαντικό πλεονέκτημα είναι η ανεξαρτησία τους από τις διαστάσεις των δεδομένων, καθώς και η ικανότητα τους να αντιμετωπίζουν προβλήματα υψηλών διαστάσεων χωρίς να προκαλούν προβλήματα υπερεκπαίδευσης. Επίσης ο αλγόριθμος SVM έχει πολλές εφαρμογές σε πολλούς τομείς όπως η αναγνώριση εικόνων, η ανάλυση κειμένου και άλλους πολυποίκιλους τομείς. Λόγω της αποδοτικότητας του στην ταξινόμηση δεδομένων ιδίως όταν αυτά είναι λίγα σε σχέση με τις διαστάσεις ή αντίστοιχα όταν αντιμετωπίζουμε προβλήματα μεγάλης διάστασης. Παρακάτω παρατίθενται παράδειγμα της ταξινόμησης του αλγορίθμου αυτού.





(α) **(β)**

Γράφημα 2.1 α, β Διαχωρισμός δεδομένων σε κλάσεις.
Πηγή: https://www.mltut.com/

Όπως παρατηρούμε στο γράφημα 2.1α, β φαίνονται δύο διαφορετικοί τρόποι διαχωρισμού των δεδομένων μας. Στο γράφημα 2.1α διαχωρίζονται με μία κάθετη γραμμή ενώ στο γράφημα 2.1β με μία διαγώνια. Ο τρόπος με τον οποίο θα διαχωρίσουμε τα δεδομένα μας είναι σημαντικός διότι όταν στο μέλλον θα προσθέσουμε κι άλλα δεδομένα- σημεία στο γράφημα μας και θέλουμε να τα ταξινομήσουμε τότε θα ξέρουμε που ανήκουν. Το κύριο λοιπόν καθήκον είναι να βρούμε τη βέλτιστη γραμμή ή το καλύτερο σημείο απόφασης. Αυτό ακριβώς πραγματοποιεί ο αλγόριθμος μηχανών διανυσματικής υποστήριξης ο οποίος βρίσκει το καλύτερο όριο απόφασης μέσω του μέγιστου περιθωρίου που σημαίνει ότι έχει τη μέγιστη και ίση απόσταση από τις δύο κλάσεις ή χώρους. Το άθροισμα αυτών των δύο κλάσεων πρέπει να μεγιστοποιηθεί για να γίνει η γραμμή το μέγιστο περιθώριο.

Τα πλεονεκτήματα του αλγορίθμου αυτού λοιπόν με βάση όσα αναφέρθηκαν παραπάνω είναι η εξαιρετική απόδοση σε χώρους υψηλών διαστάσεων διότι καθίστανται κατάλληλα για σύνολα δεδομένων με πολλά χαρακτηριστικά. Επίσης η δυνατότητα να ελαχιστοποιούν τον κίνδυνο υπερεκπαίδευσης τον καθιστά κατάλληλο για νέα δεδομένα. Τέλος σε προβλήματα κατηγοριοποίησης με δύο διαστάσεις ο αλγόριθμος μπορεί να δημιουργήσει ένα υπερεπίπεδο που αποτελεί αποτελεσματικό για τη διαχώριση των κλάσεων όταν αυτές είναι γραμμικά διαχωρίσιμες.

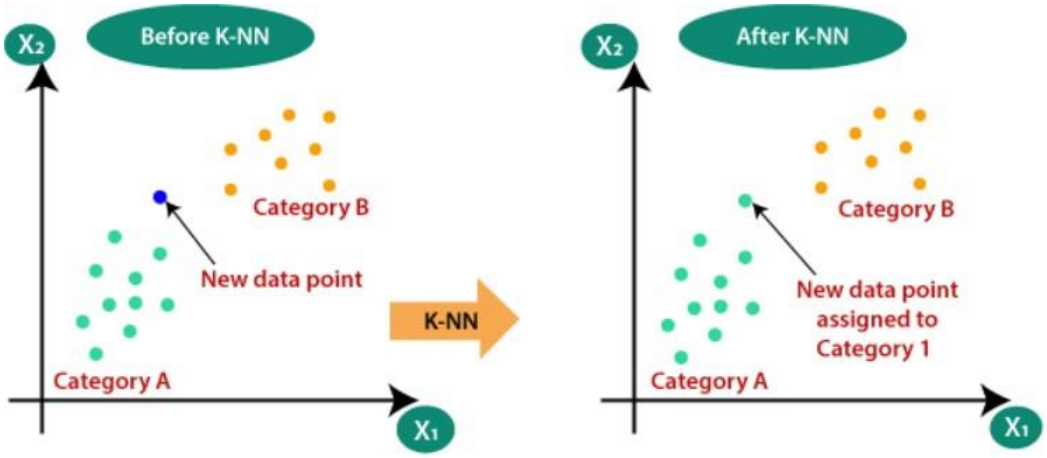
Τα μειονεκτήματα που εμφανίζει ο αλγόριθμος είναι αρχικά ο χρόνος εκπαίδευσης των μηχανών διανυσματικής υποστήριξης ο οποίος μπορεί να αυξηθεί σημαντικά με την αύξηση του μεγέθους του συνόλου δεδομένων . Επίσης τα υπερεπίπεδα που δημιουργούνται είναι δύσκολα ως προς την ερμήνευση τους από τον άνθρωπο. Τέλος όταν το μέγεθος του συνόλου δεδομένων αυξάνεται αυτό κάνει τον αλγόριθμο λιγότερο κλιμακούμενο σε μεγάλες εφαρμογές.

2.3 Αλγόριθμος Κ-πλησιέστερων γειτόνων (K-Nearest Neighbors)

Ο αλγόριθμος Κ-πλησιέστερων γειτόνων αποτελεί έναν από τους απλούστερους αλγορίθμους μηχανικής μάθησης που βασίζεται στη τεχνική της επιβλεπόμενης μάθησης. Ο KNN υποθέτει την ομοιότητα μεταξύ της νέας περίπτωσης δεδομένων και των διαθέσιμων περιπτώσεων και τοποθετεί τη νέα περίπτωση στην κατηγορία που μοιάζει περισσότερο με τις διαθέσιμες κατηγορίες. Επίσης αποθηκεύει όλα τα διαθέσιμα δεδομένα και ταξινομεί ένα νέο σημείο δεδομένων με βάση την ομοιότητα, δηλαδή όταν εμφανίζεται ένα νέο σημείο δεδομένων τότε μπορεί να ταξινομηθεί σε μία καλά προσαρμοσμένη κατηγορία. Είναι ένας μη παραμετρικός αλγόριθμος που σημαίνει ότι δεν κάνει καμία υπόθεση για τα

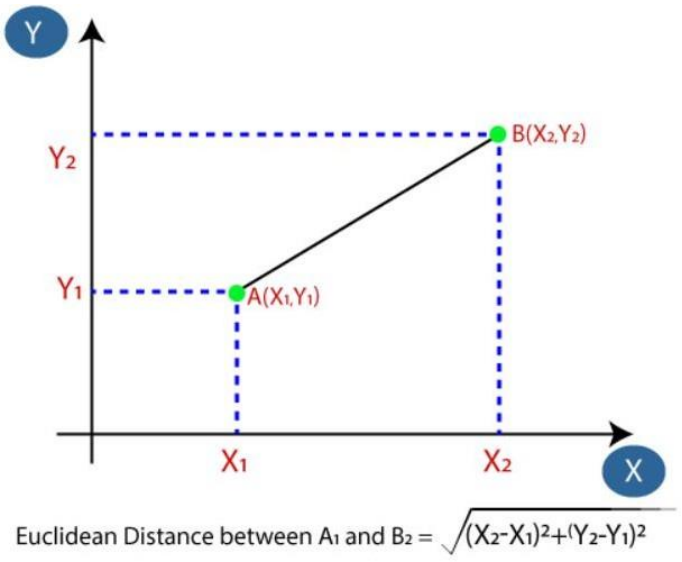
υποκείμενα δεδομένα. Τέλος ονομάζεται και “τεμπέλικος” αλγόριθμος διότι δεν μαθαίνει

από το σύνολο εκπαίδευσης αμέσως αλλά αποθηκεύει το σύνολο δεδομένων και τη στιγμή της ταξινόμησης εκτελεί μία ενέργεια σε αυτό. Παρακάτω βλέπουμε τον τρόπο με τον οποίον ταξινομείτε ένα δεδομένο με βάση τον αλγόριθμο.



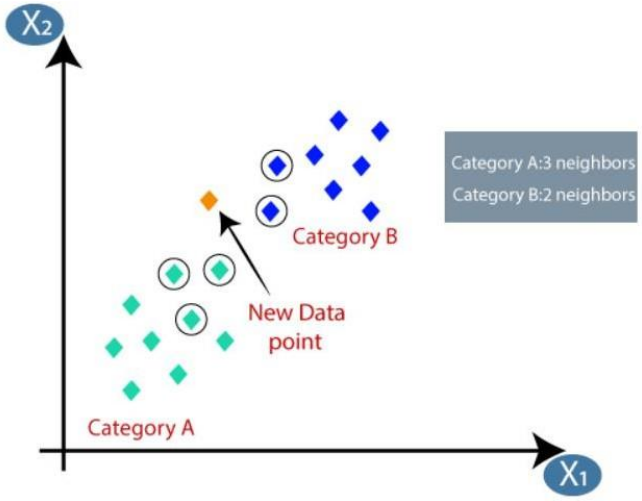
Γράφημα 2.2 Ταξινόμηση δεδομένου με βάση τον KNN
Πηγή: https://www.javatpoint.com/

Για να πραγματοποιηθεί η ταξινόμηση όπως βλέπουμε στο γράφημα 2.2 παραπάνω , πρώτον πρέπει να επιλέξουμε τον αριθμό των γειτόνων . Στη συνέχεια θα υπολογίσουμε την ευκλείδεια απόσταση μεταξύ των σημείων δεδομένων (Γράφημα 2.3).



Γράφημα 2.3 Ευκλείδεια απόσταση δύο σημείων
Πηγή: https://www.javatpoint.com/

Με τον υπολογισμό της ευκλείδειας απόστασης πήραμε τους τρεις πλησιέστερους γείτονες στην κατηγορία Α και δύο γείτονες στη κατηγορία Β όπως φαίνεται στο γράφημα 2.4 παρακάτω. Οι τρεις πιο κοντινοί γείτονες ανήκουν στη κατηγορία Α και οι δύο στη κατηγορία Β. Έτσι καταλήγουμε ότι το νέο δεδομένο ανήκει στη κατηγορία Α όπως φαίνεται στο γράφημα 2.2.



Γράφημα 2.4 Ταξινόμηση νέου στοιχείου
Πηγή: https://www.javatpoint.com/

Στον αλγόριθμο KNN δεν υπάρχει κανόνας για το πόσα κέντρα επιλέγουμε, χρειάζεται να κάνουμε δοκιμές για να καταλήξουμε στο καλύτερο αποτέλεσμα διότι τα dataset που χρησιμοποιούμε είναι διαφορετικά και τα σημεία στο χώρο, άρα κάθε φορά η επιλογή του κπρέπει να δοκιμάζετε. Όταν η επιλογή του κ αντιστοιχεί σε λίγα κέντρα τότε αυτό σημαίνει ότι μπορεί να έχουμε επιδράσεις από τιμές που αποτελούν outliers . Αντίθετα η επιλογή πολλών κέντρων είναι καλή αλλά εμφανίζει δυσκολίες όπως υπολογιστική πολυπλοκότητα διότι θα χρειαστεί να υπολογίσεις την απόσταση κάθε κέντρου από κάθε σημείο και απαιτείται χρόνος αλλά θα χρειαστεί και η απαιτούμενη μνήμη ιδιαίτερα σε μεγάλα σύνολα δεδομένων.

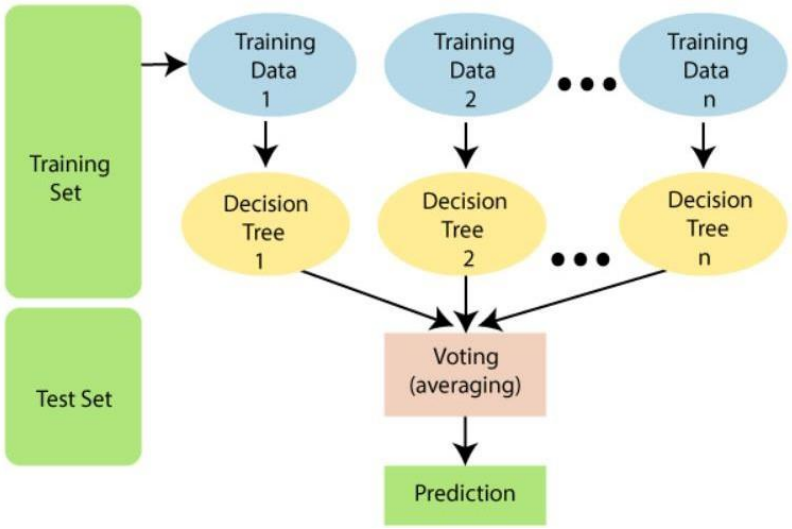
Τα πλεονεκτήματα του αλγορίθμου αυτού είναι αρχικά ότι είναι απλός στην εφαρμογή. Επίσης είναι ανθεκτικός στα θορυβώδη δεδομένα της εκπαίδευσης. Τέλος εάν τα δεδομένα εκπαίδευσης είναι μεγάλα μπορεί να είναι πιο αποτελεσματικός.

Αντίστοιχα όμως υπάρχουν και μειονεκτήματα. Ένα από αυτά είναι ότι χρειάζεται πάντοτε να προσδιορίζεται η τιμή του κ, η οποία μπορεί να γίνει πολύπλοκη. Επίσης το κόστος υπολογισμού είναι υψηλό λόγω του υπολογισμού της απόστασης μεταξύ των σημείων

δεδομένων για όλα τα δείγματα εκπαίδευσης. Τέλος μπορεί να αντιμετωπίσει δυσκολίες όταν τα δεδομένα είναι αραιά.

2.4 Αλγόριθμος Τυχαία δάση (Random Forest)

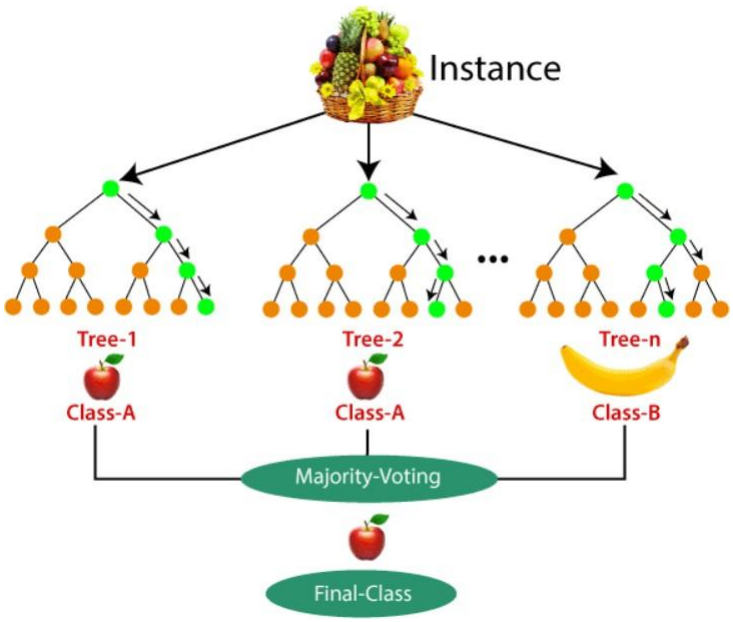
Ο αλγόριθμος Τυχαία δάση όπως αναφέρεται και το όνομα του είναι ένας ταξινομητής που περιέχει έναν αριθμό δέντρων απόφασης σε διάφορα υποσύνολα του δεδομένου συνόλου δεδομένων και λαμβάνει το μέσο όρο για να βελτιώσει την ακρίβεια πρόβλεψης του συνόλου αυτού. Το τυχαίο δάσος λαμβάνει τη πρόβλεψη από κάθε δέντρο και με βάση τη πλειοψηφία των προβλέψεων , προβλέπει τη τελική έξοδο. Όσο μεγαλύτερος είναι ο αριθμός των δέντρων τόσο σε υψηλότερη ακρίβεια θα φτάσουμε. Με βάση το διάγραμμα που εμφανίζεται παρακάτω μπορούμε να παρατηρήσουμε ότι παρόλο που υπάρχουν πολλά δέντρα πρόβλεψης της κλάσης ,κάποια από αυτά μπορεί να καταλήξουν σε λαθεμένη πρόβλεψη. Συνολικά όμως όλα μαζί καταλήγουν στη σωστή πρόβλεψη.



Γράφημα 2.5 Ανάλυση λειτουργίας αλγορίθμου τυχαίων δασών
Πηγή: https://www.javatpoint.com/

Τα πλεονεκτήματα αυτού του αλγορίθμου είναι ότι συγκριτικά με άλλους αλγορίθμους προβλέπει με υψηλή ακρίβεια τη κλάση ακόμη και εάν το σύνολο δεδομένων που αναλύουμε είναι μεγάλο σε όγκο. Επίσης ακόμα και να μην υπάρχει μεγάλο ποσοστό δεδομένων και πάλι μπορεί να διατηρήσει την ακρίβεια του. Η δομή του αλγορίθμου αυτού αρχικά ξεκινάει με την επιλογή n δεδομένων από το σύνολο εκπαίδευσης, ύστερα δημιουργεί δέντρα απόφασης που σχετίζονται με τα δεδομένα που έχουμε επιλέξει και στη συνέχεια επιλέγει x τυχαία δέντρα απόφασης που θέλουμε εμείς να δημιουργήσουμε. Η

διαδικασία επιλογής δεδομένων και δέντρων απόφασης επαναλαμβάνεται. Μόλις γίνει η προσθήκη νέων δεδομένων, αυτά ανατίθενται με βάση τα δέντρα απόφασης στην κλάση εκείνη που κερδίζει αυτή που έχει τα περισσότερα κοινά με το νέο δεδομένο μας. Ας δούμε το παρακάτω παράδειγμα για να καταλάβουμε πιο πρακτικά τη λειτουργία του.



Γράφημα 2.6 Παράδειγμα ταξινόμησης φρούτων σε κλάσεις με τη μέθοδο τυχαίων δασών
Πηγή: https://www.javatpoint.com/

Στο διάγραμμα παραπάνω βλέπουμε ότι τα δεδομένα μας είναι εικόνες φρούτων και στόχος μας είναι να διαχωρίσουμε τα φρούτα σωστά στις κλάσεις τους με βάση που ανήκουν. Χρησιμοποιώντας λοιπόν τον αλγόριθμο των τυχαίων δασών αρχικά το σύνολο των εικόνων μας χωρίζεται σε υποσύνολα και αντιστοιχίζονται σε δέντρα απόφασης. Κάθε δέντρο απόφασης κατά τη διάρκεια τη εκπαίδευσης των δεδομένων μας παράγει ένα αποτέλεσμα πρόβλεψης και έτσι όταν εμφανίζεται ένα νέο σημείο δεδομένων, μία νέα εικόνα στη συγκεκριμένη περίπτωση τότε με βάση την πλειοψηφία των αποτελεσμάτων ο αλγόριθμος των τυχαίων δασών θα καταλήξει στη τελική του απόφαση δηλαδή στη πρόβλεψη της κλάσης όπου ανήκει η εικόνα.

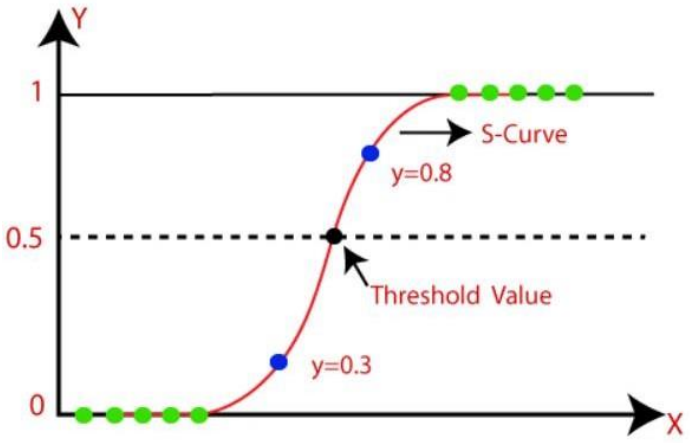
Φυσικά όμως ο αλγόριθμος παρουσιάζει και μειονεκτήματα. Αρχικά ενώ παρέχουν πολύ καλή απόδοση η ερμηνεία των αποτελεσμάτων μπορεί να είναι πολύπλοκη καθώς είναι δύσκολο να εξηγηθεί πως λαμβάνονται οι αποφάσεις σε κάθε δέντρο και σε κάθε στάδιο. Επίσης η εκπαίδευση μπορεί να είναι χρονοβόρα, ειδικά σε πολύ μεγάλα σύνολα δεδομένων. Τέλος κάθε δέντρο στο τυχαίο δάσος χρειάζεται να αποθηκευτεί στη μνήμη κατά τη διάρκεια της εκπαίδευσης και της πρόβλεψης κάτι που μπορεί να απαιτεί αρκετή μνήμη για μεγάλα δάση.

2.5 Αλγόριθμος Λογιστικής Παλινδρόμησης (Logistic Regression)

Ο αλγόριθμος της λογιστικής παλινδρόμησης αποτελεί ένα δημοφιλή αλγόριθμο μηχανικής μάθησης , ο οποίος χρησιμοποιείται στην επίλυση προβλημάτων ταξινόμησης. Η λογιστική παλινδρόμηση προβλέπει την πιθανότητα για το πού ανήκει ένα παρατηρούμενο αντικείμενο σε μία από τις 2 κατηγορίες του συνόλου δεδομένων που εξετάζουμε.

Στη λογιστική παλινδρόμηση προσαρμόζουμε μία λογιστική συνάρτηση σχήματος S , η οποία προβλέπει τις πιθανότητες κατηγοριοποίησης των δεδομένων. Αυτό επιτρέπει στο μοντέλο να παράγει μία πρόβλεψη με βάση την πιθανότητα αντί για μία απόλυτη τιμή. Οι καμπύλες από τη λογιστική συνάρτηση αντιπροσωπεύουν αυτές τις πιθανότητες και μπορούν να χρησιμοποιηθούν για να κατανοήσουμε τη σχέση μεταξύ των ανεξάρτητων μεταβλητών και τις πιθανότητες να ανήκει σε μία συγκεκριμένη κατηγορία.

Συνολικά η λογιστική παλινδρόμηση προσφέρει τη δυνατότητα να εξετάσουμε τις πιθανότητες των κατηγοριών και να κατανοήσουμε την επίδραση των μεταβλητών στην κατηγοριοποίηση των παρατηρήσεων. Είναι ένα χρήσιμο εργαλείο για προβλήματα ταξινόμησης με διακριτά αποτελέσματα και προσφέρει ευελιξία στον τρόπο που μπορεί να προσαρμοστεί σε διάφορα είδη δεδομένων. Στο γράφημα 2.7 μπορούμε να δούμε πως λειτουργεί και πως διαχωρίζει τα δεδομένα ο αλγόριθμος αυτός.



Γράφημα 2.7 Παράδειγμα ταξινόμησης με τη μέθοδο της λογιστικής παλινδρόμησης
Πηγή: https://www.javatpoint.com/

Η σιγμοειδής συνάρτηση που αναφέραμε παραπάνω και παρατηρούμε και στο διάγραμμα είναι ένα μαθηματικό εργαλείο που χρησιμοποιείται στη λογιστική παλινδρόμηση. Η βασική ιδέα είναι να μετατρέπει πραγματικές τιμές σε πιθανότητες καθορίζοντας ένα εύρος τιμών μεταξύ του 0 και 1.

Η συνάρτηση χαρτογραφεί οποιαδήποτε πραγματική είσοδο σε μία τιμή μεταξύ 0 και 1, προσφέροντας ένα τρόπο να εκφραστεί η πιθανότητα ενός γεγονότος. Η μορφή S της

καμπύλης ονομάζεται σιγμοειδής καμπύλη και χρησιμεύει ως μέσο για να εξασφαλίσει ότι οι προβλέψεις της λογιστικής παλινδρόμησης βρίσκονται μεταξύ των ορίων 0 και 1.

Ουσιαστικά χρησιμοποιούμε ένα καθορισμένο ‘’κατώφλι’’ για να αποφανθούμε αν η πρόβλεψη πρέπει να είναι 0 ή 1. Όταν η εκτίμηση της πιθανότητας είναι πάνω από το κατώφλι , τότε το μοντέλο τείνει να κατατάξει το αντικείμενο στην κατηγορία 1, ενώ όταν

είναι κάτω από το κατώφλι το μοντέλο τείνει προς τη κατηγορία 0. Αυτό παρέχει ένα δυναμικό τρόπο απόφασης επιτρέποντας την αντιστοίχιση προβλεπόμενων τιμών σε διακριτές κατηγορίες.

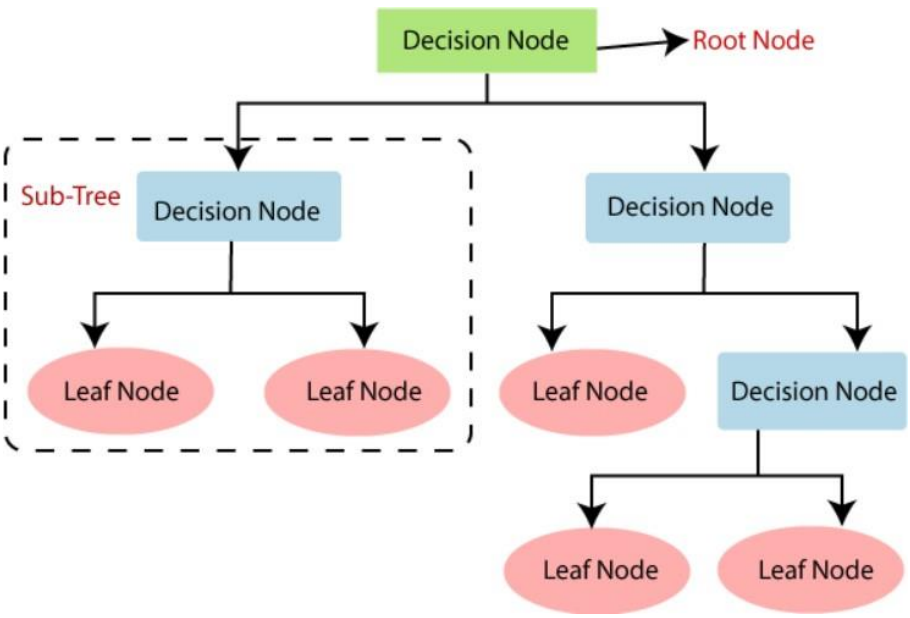
Η λογιστική παλινδρόμηση έχει αρκετά πλεονεκτήματα που την καθιστούν δημοφιλή και χρήσιμη σε πολλά προβλήματα ταξινόμησης όπως :

- Προσαρμοστικότητα σε διάφορα είδη δεδομένων
- Εύκολη ερμηνεία λόγω της συγμοειδούς συνάρτησης
- Αντιμετώπιση συνεπικαλύψεων στις κατηγορίες

2.6 Αλγόριθμος Δέντρου Απόφασης (Decision Tree)

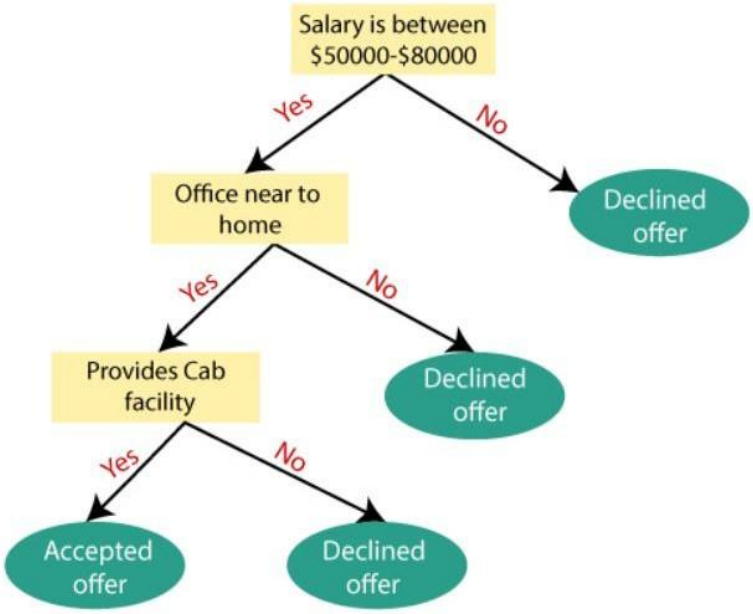
Ο Αλγόριθμος του δέντρου απόφασης όπως αναφέρει και το όνομα του βασίζεται σε μία δομή δέντρου όπου οι εσωτερικοί κόμβοι αντιπροσωπεύουν χαρακτηριστικά των δεδομένων, οι κλάδοι αντιπροσωπεύουν τους κανόνες απόφασης και οι κόμβοι των φύλλων αναπαριστούν τα τελικά αποτελέσματα.

Στο δέντρο απόφασης υπάρχουν δύο είδη κόμβων, ο κόμβος απόφασης όπου εκεί λαμβάνονται οι αποφάσεις και ο κόμβος φύλλο που παρέχει το τελικό αποτέλεσμα. Οι αποφάσεις λαμβάνονται βάσει των χαρακτηριστικών του συνόλου δεδομένων. Ξεκινώντας από ένα κόμβο ρίζας καταλήγουμε σε κλάδους που αντιπροσωπεύουν πιθανές λύσεις. Συνοπτικά το δέντρο απόφασης απλά θέτει ερωτήσεις βασισμένες στα χαρακτηριστικά των δεδομένων και διακλαδώνει το δέντρο ανάλογα τις απαντήσεις, δημιουργώντας έτσι μια γραφική αναπαράσταση για τις πιθανές λύσεις ενός προβλήματος όπως φαίνεται παρακάτω στο γράφημα 2.8.



Γράφημα 2.8 Διάγραμμα ταξινόμησης με τη μέθοδο δέντρου απόφασης
Πηγή: https://www.javatpoint.com/

Παρακάτω μπορούμε να δούμε ένα παράδειγμα επεξήγησης της δόμησης του αλγορίθμου. Έστω ότι υπάρχει υποψήφιος που έχει τη παρακάτω προσφορά και θέλει να αποφασίσει εάν θα τη δεχτεί ή όχι. Για την επίλυση του προβλήματος το δέντρο αποφάσεων ξεκινά με τον κόμβο ρίζα που είναι ο μισθός που μπορεί να δοθεί. Ο κόμβος ρίζα χωρίζεται στον επόμενο κόμβο απόφασης έως ότου φτάσουμε στο τελικό κόμβο απόφασης όπου αυτός χωρίζεται στους δύο κόμβους φύλλων όπου εκεί καταλήγουμε και στη τελική απόφαση δηλαδή εάν ο υποψήφιος θα δεχτεί ή θα απορρίψει την απόφαση.



Γράφημα 2.9 Παράδειγμα ταξινόμησης με τη μέθοδο δέντρου απόφασης
Πηγή: https://www.javatpoint.com/

Το πρώτο λοιπόν πλεονέκτημα το οποίο παρατηρούμε και στο γράφημα 2.9 είναι πως τα δέντρα αποφάσεων είναι εύκολα και κατανοητά διότι μοιάζουν με τον τρόπο της ανθρώπινης σκέψης κατά τη λήψη μιας απόφασης. Ο διαχωρισμός των δεδομένων σε υποομάδες με βάση τα χαρακτηριστικά τους αποτελεί ένα δεύτερο πλεονέκτημα το οποίο βοηθά στην αποτελεσματική ταξινόμηση και πρόβλεψη. Τέλος η ελάχιστη προ επεξεργασία αποτελεί ένα ακόμη πλεονέκτημα. Δεν απαιτείται κλιμάκωση των χαρακτηριστικών ή κανονικοποίηση.

Φυσικά όμως υπάρχουν και μειονεκτήματα. Ένα από αυτά είναι ότι τα δέντρα αποφάσεων μπορούν να περιέχουν πολλά επίπεδα γεγονός που το καθιστά πολύπλοκο. Επίσης τα δέντρα απόφασης έχουν την τάση να προσαρμόζονται υπερβολικά στα δεδομένα εκπαίδευσης με αποτέλεσμα να ενδέχεται να μην γενικεύουν καλά σε νέα και απρόβλεπτα δεδομένα. Τέλος ενώ τα δέντρα απόφασης είναι καλά για προβλήματα ταξινόμησης μπορεί να μην είναι πάντα η καλύτερη επιλογή για προβλήματα παλινδρόμησης όπου η πρόβλεψη της συνεχούς τιμής είναι πιο σύνθετη.

2.7 Μέτρα ακρίβειας & απόδοσης

Τα μέτρα ακρίβειας αναφέρονται στο πόσο κοντά είναι ένα μέτρο ή μία μέτρηση στην πραγματική της τιμή. Η απόδοση αφορά τη συνολική ικανότητα ενός συστήματος ή μεθόδου να παράγει ακριβείς αποτελέσματα.

2.7.1 Ακρίβεια/Πρόβλεψη

Η πρόβλεψη (precision) είναι ένας μετρικός δείκτης ο οποίος αναφέρεται στο ποσοστό των προβλέψεων που είναι πραγματικά σωστές αναφορικά με το σύνολο των προβλέψεων που έγιναν για μια συγκεκριμένη κλάση. Ένα μοντέλο με υψηλή ακρίβεια θα κάνει λιγότερα λάθη και θα έχει μικρότερο ποσοστό από προβλέψεις που ήταν λανθασμένες σε σχέση με τον συνολικό αριθμό προβλέψεων. Για παράδειγμα στη δική μας ανάλυση η ακρίβεια θα μας δείξει πόσο καλά το μοντέλο μας καταφέρνει να προβλέψει εάν ο ασθενής έχει διαβήτη βάση των κλινικών του δεδομένων ή όχι.

Σε έναν ιδανικό κόσμο το βέλτιστο αποτέλεσμα θα ήταν να είχαμε 100% ακρίβεια στη πρόβλεψη του μοντέλου και να μπορούσε να κάνουμε διάγνωση με την απόλυτη επιτυχία. Υπάρχουν όμως κι άλλοι πολλοί παράγοντες συνδυαστικά με την ακρίβεια που μας επιτρέπουν να θεωρούμε ότι η ακρίβεια αυτή που έχουμε υπολογίσει είναι και αξιόπιστη.

2.7.2 Ευαισθησία

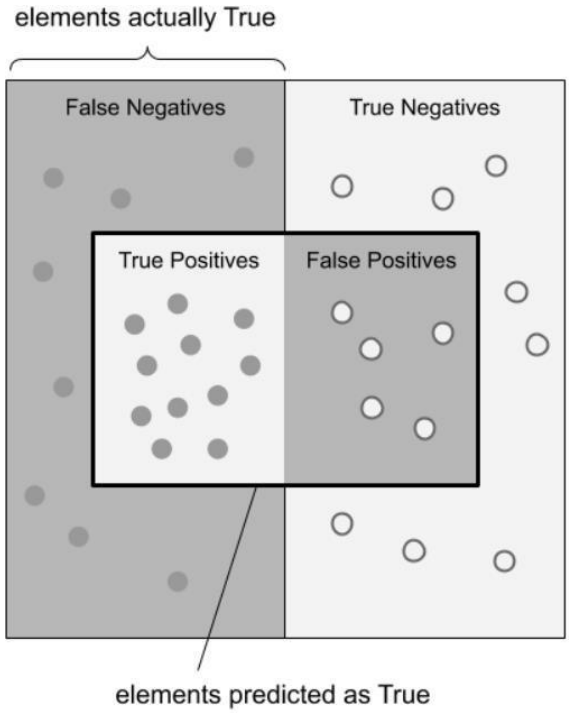
Η ευαισθησία είναι ένα μέτρο που αναφέρεται στη δυνατότητα ενός μοντέλου να αναγνωρίσει και να ανιχνεύσει τα πραγματικά θετικά παραδείγματα. Υπολογίζει δηλαδή το ποσοστό των σωστών θετικών αποτελεσμάτων του test στο σύνολο των πραγματικών θετικών αποτελεσμάτων. Η υψηλή ευαισθησία σημαίνει ότι το μοντέλο είναι σε θέση να ανιχνεύσει τις περισσότερες πραγματικές θετικές περιπτώσεις.

2.7.3 Πίνακας πιθανοτήτων (Confusion matrix)

Ο πίνακας πιθανοτήτων είναι ένα εργαλείο που χρησιμοποιείται για την αξιολόγηση της απόδοσης ενός μοντέλου μηχανικής μάθησης σε προβλήματα κατηγοριοποίησης. Απεικονίζει τον τρόπο με τον οποίο τα δεδομένα ελέγχου διασπώνται σε διάφορες κατηγορίες ανάλογα με το τι προβλέπει το μοντέλο και τι είναι η πραγματική τους κατηγορία. Παρακάτω παρουσιάζονται τα 4 βασικά στοιχεία ελέγχου του πίνακα :

		Actual Class	
		Positive (**P**)	Negative (**N**)
Predicted Class	Positive (**P**)	True Positive (**TP**)	False Positive (**FP**)
	Negative (**N**)	False Negative (**FN**)	True Negative (**TN**)

Γράφημα 2.10 Confusion matrix
Πηγή: https://www.ml-science.com/



Γράφημα 2.11 Απεικόνιση του Confusion Matrix
Πηγή: https://www.ml-science.com/

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

$$TPR = \frac{TP}{TP + FN}$$

$$TNR = \frac{TN}{TN + FP}$$

$$FPR = \frac{FP}{TN + FP}$$

Γράφημα **2.12** Μέτρα ανάλυσης Confusion Matrix
Πηγή: https://www.ml-science.com/

Όπου ACC είναι η ακρίβεια , TP και TN ο αριθμός των πραγματικών θετικών και αρνητικών προβλέψεων αντίστοιχα, FP και FN είναι ο αριθμός των λανθασμένων θετικών και αρνητικών προβλέψεων αντίστοιχα. Τέλος το TPR και FPR είναι το ποσοστό των πραγματικών θετικών και αρνητικών προβλέψεων αντίστοιχα.

2.7.4 F1-Score

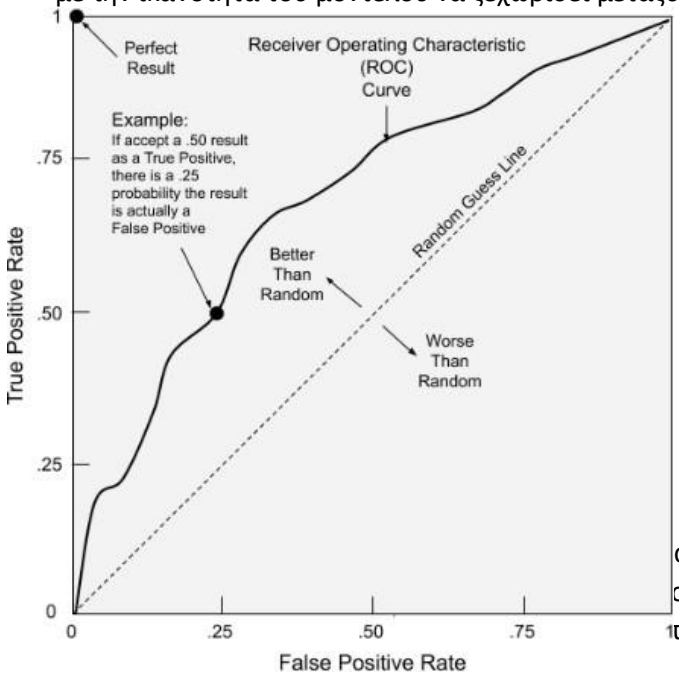
Το F1-Score είναι ένας τρόπος μέτρησης της απόδοσης ενός μοντέλου μηχανικής μάθησης σε ένα σύνολο δεδομένων. Συνήθως χρησιμοποιείται σε προβλήματα κατηγοριοποίησης όπως η ταξινόμηση.

Το F1-Score συνδυάζει δύο μετρικές που αναφέραμε και αναλύσαμε παραπάνω την ακρίβεια (precision) και την ανάκληση(recall). Το F1-Score εκφράζει τον αρμονικό μέσο όρο της ακρίβειας και της ανάκλησης. Αυτό το καθιστά ιδανικό για περιπτώσεις όπου είναι σημαντικό να ισορροπηθούν η ακρίβεια και η ανάκληση του μοντέλου. Το F1-Score κυμαίνεται από 0 έως 1, όπου 1 είναι ο τέλειος συνδυασμός ακρίβειας και ανάκλησης, ενώ το 0 είναι η χειρότερη δυνατή απόδοση.

2.7.5 Receiver Operating Characteristic (ROC)

Το ROC είναι ένα γράφημα που χρησιμοποιείται για να αξιολογήσει την απόδοση ενός μοντέλου ταξινόμησης. Συνήθως χρησιμοποιείται σε προβλήματα με δύο κλάσεις. Το ROC γράφημα απεικονίζει τη σύγκριση μεταξύ του ποσοστού των σωστών προβλέψεων(TPR) και του ποσοστού των λανθασμένων προβλέψεων(FPR). Το ROC γράφημα αποτελείται από μία καμπύλη που αντιπροσωπεύει διάφορα επίπεδα κατωφλίου που χρησιμοποιούνται για να κατατάξουν τα παραδείγματα στις κλάσεις. Η καμπύλη ROC μπορεί να δείξει τη σχέση

μεταξύ του TPR και FPR για διάφορα κατώφλια και μπορεί να παρέχει πληροφορίες σχετικά με την ικανότητα του μοντέλου να ξεχωρίσει μεταξύ των κλάσεων.

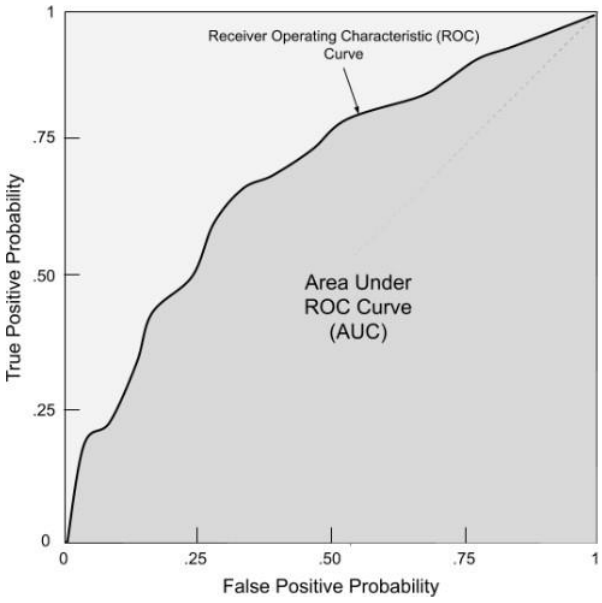


αλείο για την αξιολόγηση της απόδοσης οηθήσει στην επιλογή του κατάλληλου τις κλάσεις.

2.7.6 Area under the curve (AUC)

Το AUC είναι μία μετρική που είναι άμεσα συσχετισμένη με την καμπύλη ROC. Συγκεκριμένα η περιοχή κάτω από την καμπύλη ROC είναι μία συνηθισμένη μετρική για τη συγκρισιμότητα της απόδοσης μεταξύ των διαφορετικών μοντέλων.

Το AUC μετράει το εμβαδόν κάτω από την καμπύλη ROC. Αυτό σημαίνει ότι όσο μεγαλύτεροείναι το εμβαδόν τόσο καλύτερη είναι η ικανότητα του μοντέλου στη διάκριση μεταξύ των κλάσεων. Με άλλα λόγια όσο μεγαλύτερο είναι το AUC τόσο καλύτερη είναι η ικανότητα του μοντέλου στην ταξινόμηση των παραδειγμάτων.



Γράφημα **2.1****4** Υπολογισμός μετρικής AUC
Πηγή: https://www.ml-science.com/

Ένα AUC ίσο με 1 σημαίνει ότι το μοντέλο έχει τέλεια διάκριση μεταξύ των κλάσεων ενώ AUC ίσο με 0.5 υποδεικνύει ότι το μοντέλο δεν έχει καθόλου διάκριση και κάνει προβλέψεις τυχαίας φύσης.

3. Αποτελέσματα

Στο κεφάλαιο αυτό γίνεται η ανάλυση των αλγορίθμων που αναλύσαμε στα προηγούμενα κεφάλαια και η σύγκριση μεταξύ αυτών ως προς την ακρίβεια και τον χρόνο εκτέλεσης (εδώ να σημειώσουμε ότι ο χρόνος εκτέλεσης ενός αλγορίθμου εξαρτάται από το μέγεθος των δεδομένων αλλά και από την υπολογιστική ισχύ του υπολογιστή μας).

3.1 Δεδομένα

Το dataset που μελετήσαμε αποτελείται από κλινικά δεδομένα 100.000 ασθενών μαζί με την κατάσταση του διαβήτη τους (θετική ή αρνητική) . Το dataset ονομάζεται diabetes_prediction_dataset και το συλλέξαμε από τη πλατφόρμα δεδομένων Kaggle η οποία ξεκίνησε ως κοινότητα για data scientists και machine learning engineers όπου μπορούν να βρίσκουν και να δημοσιεύουν datasets , να λύνουν προβλήματα και να μοιράζονται ιδέες. Τα δεδομένα περιλαμβάνουν χαρακτηριστικά όπως η ηλικία, το φύλο, ο δείκτης μάζας σώματος (ΔΜΣ), η υπέρταση, οι καρδιακές παθήσεις, το ιστορικό καπνίσματος, το επίπεδο HbA1c και το επίπεδο γλυκόζης στο αίμα (αναλύονται στο πίνακα 1 παρακάτω) .

Α/Α	Μεταβλητή	Ερμηνεία	Τιμές
1	Age	Ηλικία	0 = Δεν έχει διαβήτη
2	HbA1c_level	Μ.Ο σακχάρου	1 = Έχει διαβήτη
3	Blood_glucose_level	Επίπεδο γλυκόζης αίματος	
4	Bmi	Δείκτης μάζας σώματος	
5	Gender	Φύλο	
6	Hypertension	Υπέρταση	
7	Heart_disease	Πάθηση καρδιάς	
8	Smoking_history	Ιστορικό καπνίσματος	

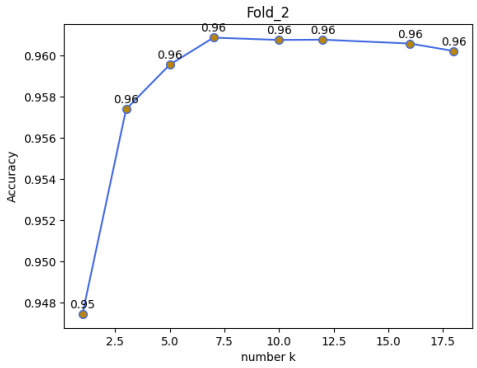
Πίνακας 3.1 Ορισμός των μεταβλητών diabetes_prediction dataset

3.2 Αποτελέσματα των αλγορίθμων

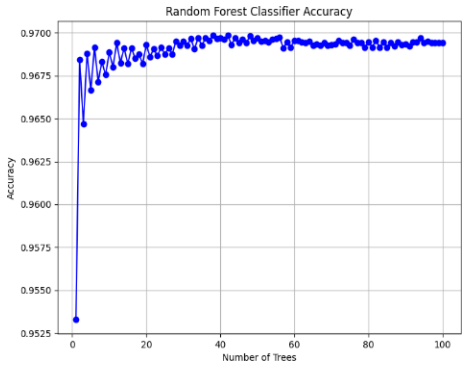
Τα αποτελέσματα που προέκυψαν από την ανάλυση των αλγορίθμων μας παρουσιάζονται στο πίνακα 2 καθώς και στα γραφήματα 1,2 και 3 όπου μπορούμε να διακρίνουμε τη διακύμανση της ακρίβειας για διαφορετικές παραμέτρους σε κάθε μία από τις μεθόδους μας. Η εξέταση των αποτελεσμάτων έδειξε υψηλή ακρίβεια σε όλες τις μεθόδους με τον αλγόριθμο των τυχαίων δασών να εμφανίζει την υψηλότερη ακρίβεια . Ομοίως υψηλό ήταν και το μέτρο AUC για το ROC . Η μετρική Log Loss στη μέθοδο KNN παρουσίασε χαμηλότερη ακρίβεια σε ελάχιστο ποσοστό βέβαια συγκριτικά με τις άλλες μεθόδους εκτός του αλγορίθμου των δέντρων απόφασης διότι είναι μία μετρική που συνήθως χρησιμοποιείται σε προβλήματα πιθανοτικής ταξινόμησης κάτι το οποίο δεν συμβαίνει στα δέντρα αποφάσεων διότι αυτά παράγουν αποφάσεις με βάση τα κατώφλια των χαρακτηριστικών όπως έχει ήδη αναφερθεί στο κεφάλαιο 2. Αντιθέτως οι μετρικές MAE δηλαδή το μέσο απόλυτο σφάλμα και το RMSE δηλαδή το μέσο τετραγωνικό σφάλμα σε όλες τις μεθόδους υποδεικνύουν μικρές απόλυτες διαφορές μεταξύ των προβλέψεων και των πραγματικών τιμών τους με καλύτερη σε αποτέλεσμα τη μέθοδο Random Forest. Ο αλγόριθμος των τυχαίων δασών επίσης φαίνεται να έχει ισορροπημένες τιμές precision και recall υποδεικνύοντας ότι μπορεί να είναι καλός στο να αναγνωρίζει και τις δύο κλάσεις. Τέλος, οι χρόνοι εκτέλεσης των μεθόδων είχαν μεγάλες διαφορές όπως παρατηρούμε με τη καλύτερη σε χρόνο τη μέθοδο της λογιστικής παλινδρόμησης.

	Random Forest	**KNN**	**SVM**	**LogRe**	**Decision_Trees**
Accuracy	0.9694	0.96087	0.95826	0.96025	0.95245
F1	0.793661	0.721369	0.666146	0.73151	0.72538
Precision	0.929699	0.9165911	0.997663	0.85884	0.71242
Recall	0.6923529	0.6923529	0.5	0.63705	0.73882
Log Loss	0.141229	0.6410573	0.10429	0.11277	1.70745
MAE	0.0306	0.03905	0.0428	0.03975	0.04755
RMSE	0.174928	0.1976107	0.206881	0.19937	0.21805
AUC	0.9646827	0.89239054	0.962162	0.96195	0.85653
TRP	0.6923529	0.59470588	0.491217	0.63705	0.73882
FPR	0.0048633	0.00076502	0.94505	0.00972	0.02770
Χρόνος_Εκτελ	6.824755907	0.323138713	782.50689	0.11878	0.22226

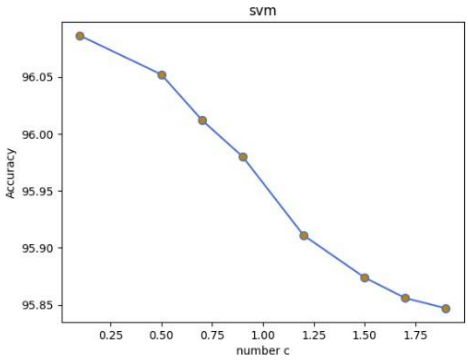
Πίνακας 3.2 Αποτελέσματα αλγορίθμων



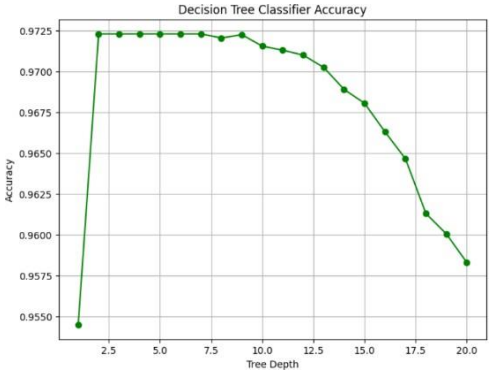
Γράφημα 3.1 Ακρίβεια KNN



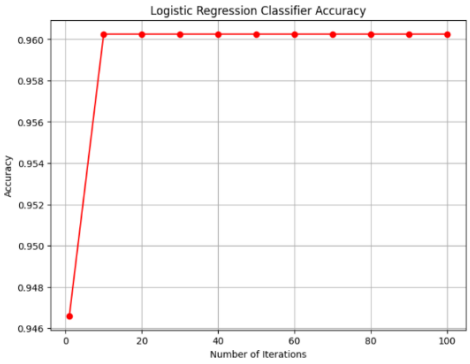
Γράφημα 3.2 Ακρίβεια Random Forest



Γράφημα 3.3 Ακρίβεια SVM



Γράφημα 3.4 Ακρίβεια Decision Tree



Γράφημα 3.5 Ακρίβεια Logistic Regression

Όπως παρατηρούμε ο κάθε αλγόριθμος εμφανίζει τη μέγιστη ακρίβεια του σε ένα συγκεκριμένο αριθμό της παραμέτρου που αντιστοιχεί στον κάθε έναν από αυτούς. Δηλαδή στον αλγόριθμο των μηχανών αναζήτησης αποφάσεων για c=0.1 έχουμε την υψηλότερη ακρίβεια γι αυτόν, όπου το c αποτελεί την παράμετρο που ρυθμίζει την αυστηρότητα του μοντέλου, στον αλγόριθμο knn για κ=7 κέντρα έχουμε την μέγιστη ακρίβεια, στον αλγόριθμο random forest για n_estimators =35- 40 δέντρα έχουμε τη μέγιστη ακρίβεια, στον αλγόριθμο των δέντρων απόφασης για tree_depth =2 αριθμούς επιπέδων που μπορεί να έχει το δέντρο και στον αλγόριθμο της λογιστικής παλινδρόμησης για number_of_iterations=10 επαναλήψεις.

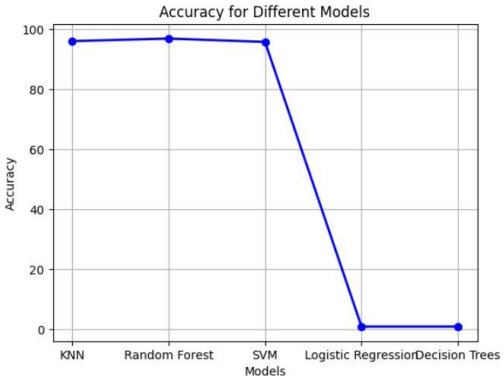
Στο παρακάτω πίνακα 3.3 φαίνονται τα αποτελέσματα που προέκυψαν από τον πίνακα confusion matrix για την εκτίμηση της απόδοσης κάθε μοντέλου όπως προέκυψαν.

	TP	**TN**	**FP**	**FN**	**N**
Random Forest	1177	18211	89	523	20000
KNN	1011	18208	92	689	20000
SVM	854	18290	2	854	20000
Decision Tree	1256	17793	507	444	20000
Logistic Regression	1083	18122	178	617	20000

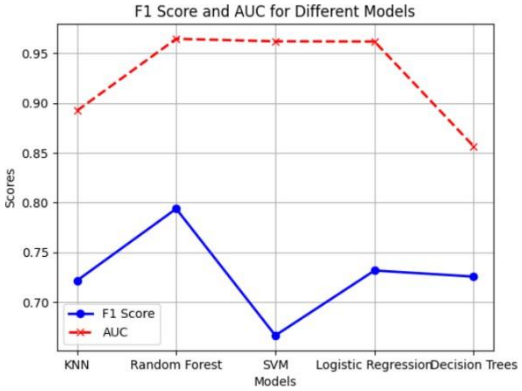
Πίνακας 3.3 Τιμές ταξινομητών

Όπως παρατηρούμε ο αλγόριθμος τυχαίων δασών (Random Forest) εμφανίζει υψηλό αριθμό δειγμάτων που ταξινομούνται ως θετικά και είναι πράγματι θετικά (TP), υψηλό επίσης αριθμό δειγμάτων που ταξινομούνται ως αρνητικά και είναι πράγματι αρνητικά (TN) καθώς και ο αριθμός των αρνητικών δειγμάτων που είναι λαθεμένα αρνητικά είναι μη αμελητέος (FN). Αντίστοιχα ο αλγόριθμοw των κ πλησιέστερων γειτόνων (KNN) έχει χαμηλό αριθμό FP αλλά ο αριθμός των FN είναι αρκετά υψηλός. Στον αλγόριθμο των μηχανών διανυσματικής υποστήριξης (SVM) έχει λιγότερα FN αλλά ο αριθμός των FP είναι πολύ χαμηλός. Στον αλγόριθμο των δέντρων αποφάσεων (Decision Tree) έχει σχετικά υψηλό αριθμό TP αλλά ο αριθμός των FN είναι αρκετά υψηλός. Τέλος ο αλγόριθμος της λογιστικής παλινδρόμησης (Logistic Regression) έχει μικρό αριθμό FN αλλά ο αριθμός των FP είναι μεγαλύτερος σε σύγκριση με τον SVM. Φυσικά για την αξιολόγηση του κάθε αλγορίθμου πρέπει να ληφθούν υπόψη και μετρικές όπως η ακρίβεια και η ευαισθησία.

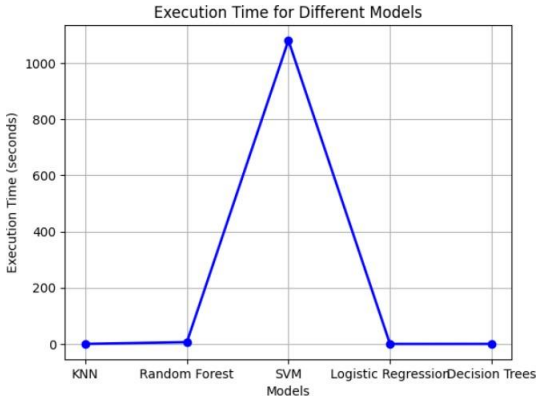
Παρακάτω στα γραφήματα 3.4, 3.5, 3.6 και 3.7 παρουσιάζονται αντίστοιχα η σύγκριση των μεθόδων ως προς διάφορες μετρικές όπως την ακρίβεια, το F1 score, το MAE και το RMSE καθώς και το χρόνο εκτέλεσης.



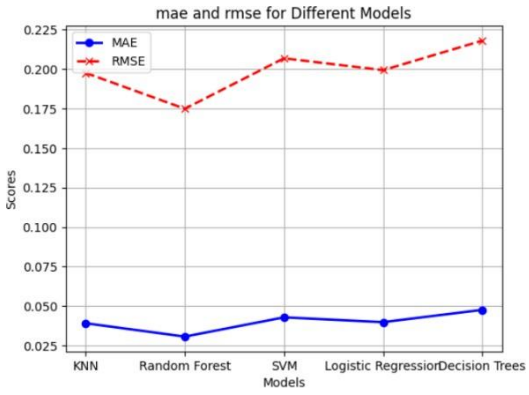
Γράφημα 3.6 Σύγκριση μεθόδων ως προς την ακρίβεια



Γράφημα 3.7 Σύγκριση μεθόδων ως προς F1 Score



Γράφημα 3.8 Σύγκριση μεθόδων ως προς χρόνο εκτέλεσης



Γράφημα 3.9 Σύγκριση μεθόδων ως προς MAE και RMSE

Όπως παρατηρούμε ο αλγόριθμος των τυχαίων δασών έχει την υψηλότερη ακρίβεια , το υψηλότερο F1 score, έχει τα χαμηλότερα MAE και RMSE υποδηλώνοντας μικρότερο σφάλμα πρόβλεψης και ενδιάμεσο χρόνο εκτέλεσης συγκριτικά με τους άλλους αλγορίθμους.

4. Συμπεράσματα

4.1 Συμπεράσματα ανάλυσης αλγορίθμων

Τα αποτελέσματα των πέντε αλγορίθμων που προέκυψαν έπειτα από την ανάλυση μας είναι αρκετά ικανοποιητικά. Φυσικά υπάρχουν αρκετά σημεία που χρειάστηκε να δώσουμε έμφαση διότι και οι πέντε μέθοδοι εμφανίζουν αρκετά καλά αποτελέσματα ως προς την ακρίβεια, οπότε έπρεπε να συνυπολογίσουμε και άλλους παράγοντες για να καταλήξουμε στον αλγόριθμο που προσφέρει τη καλύτερη ισορροπία μεταξύ ακρίβειας και επίδοσης. Η τελική επιλογή του αλγορίθμου εξαρτάται από το περιεχόμενο του dataset που εξετάζουμε αλλά και από το πλήθος των παρατηρήσεων που περιέχει αυτό. Πάραυτα υπάρχουν και άλλοι παράγοντες που αλλάζουν την ταχύτητα των αλγορίθμων και την ακρίβεια τους και αυτοί είναι οι παράμετροι που χρησιμοποιούνται κατά την εφαρμογή τους.

Τα αποτελέσματα της εξέτασης των πέντε αλγορίθμων Random Forest, KNN, SVM, Logistic Regression και Decision Tree έδειξαν ότι ο Random Forest έχει την υψηλότερη ακρίβεια σε σχέση με τους άλλους 4 αλγορίθμους. Βεβαίως η αμέσως επόμενη υψηλότερη ακρίβεια ανήκει στην KNN όπου συγκριτικά με τον Random Forest η διαφορά είναι ελάχιστη αλλά συνυπολογίζοντας και τις υπόλοιπες μετρικές ο Random Forest είναι ο πιο αποτελεσματικός αλγόριθμος.

Βάση των μετρικών των σφαλμάτων ο αλγόριθμος Random Forest δείχνει το χαμηλότερο MAE και RMSE συγκριτικά με τους άλλους 2 αλγορίθμους και αυτό υποδεικνύει μικρότερη διακύμανση μεταξύ των πραγματικών και προβλεπόμενων τιμών.

Το τελικό συμπέρασμα που προκύπτει από την εξέταση των αποτελεσμάτων μας είναι ότι ο αλγόριθμος Random Forest ξεχωρίζει για τη συνολική του επίδοση, ενώ ο αλγόριθμος SVM εμφανίζει τη μικρότερη αποτελεσματικότητα στον εντοπισμό των θετικών δειγμάτων .Τέλος ο αλγόριθμος KNN παρέχει μία λιγότερη χρονοβόρα εναλλακτική με μέτρια ακρίβεια.

4.2 Μελλοντικές βελτιώσεις

Με βάση την ανάλυσή μας, υπάρχουν μερικά βήματα που μπορούν να βελτιώσουν την επίδοση των αλγορίθμων. Αρχικά, η προσαρμογή των παραμέτρων των αλγορίθμων μπορεί να οδηγήσει σε βελτίωση της ακρίβειας. Ο εκσυγχρονισμός του Random Forest μέσω δοκιμής διαφορετικών τιμών παραμέτρων όπως τον αριθμό των δέντρων ή το βάθος τους μπορεί να βελτιώσει τα αποτελέσματα.

Επίσης, η χρήση τεχνικών εξισορρόπησης δεδομένων σε περιπτώσεις ανισορροπίας κλάσεων μπορεί να βελτιώσει την απόδοση του SVM στον εντοπισμό θετικών δειγμάτων.

Πέραν αυτού, η χρήση μιας προ-επεξεργασίας των δεδομένων μπορεί να βελτιστοποιήσει την απόδοση του KNN. Αυτό μπορεί να περιλαμβάνει κλιμακοποίηση των χαρακτηριστικών ή επιλογή σημαντικών χαρακτηριστικών για να μειωθεί η διαστατικότητα των δεδομένων.

Συνολικά, η συνεχής παραμετροποίηση, προ-επεξεργασία δεδομένων και εξέταση της συνδυασμένης χρήσης αλγορίθμων μπορούν να οδηγήσουν σε βελτίωση της απόδοσης του μοντέλου, ενώ ο συνυπολογισμός πολλών μετρικών θα πρέπει να ληφθεί υπόψη για την οριστική επιλογή του βέλτιστου αλγορίθμου.

Βιβλιογραφία

1) Machine Learning "What it is and why it matters", Διαθέσιμο στην ηλεκτρονική πλατφόρμα : https://www.sas.com/en_us/insights/analytics/machine-learning.html

2) Διαβήτης, Διαθέσιμο στην ηλεκτρονική πλατφόρμα: https://el.wikipedia.org/wiki/%CE%94%CE%B9%CE%B1%CE%B2%CE%AE%CF%84%CE%B7%CF%82_(%CE%B1%CF%83%CE%B8%CE%AD%CE%BD%CE%B5%CE%B9%CE%B

3) Τύποι Διαβήτη, Διαθέσιμο στην ηλεκτρονική πλατφόρμα: https://www.diabetes.ascensia.gr/my-diabetes/tyiloi_iabhth/#Sub-Menu-1

4) Διαβήτης, Ενημερώσου και μην αγνοείς τις ενδείξεις!, Διαθέσιμο στην ηλεκτρονική πλατφόρμα: https://www.doctoranytime.gr/p/diavitis

5) Support Vector Machines, What makes SVM very Powerful and Special?, Διαθέσιμο στην ηλεκτρονική πλατφόρμα: https://www.mltut.com/svm-machine-learning/

6) K-Nearest Neighbor(KNN) Algorithm for Machine Learning, Διαθέσιμο στην ηλεκτρονική πλατφόρμα: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

7) Random Forest Algorithm, Διαθέσιμο στην ηλεκτρονική πλατφόρμα: https://www.javatpoint.com/machine-learning-random-forest-algorithm

8) Logistic Regression in Machine Learning, Διαθέσιμο στην ηλεκτρονική πλατφόρμα: https://www.javatpoint.com/logistic-regression-in-machine-learning

9) Decision Tree Classification Algorithm, Διαθέσιμο στην ηλεκτρονική πλατφόρμα: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

10) Confusion Matrix, Derived Values, Sensitivity/Recall, Precision, F1 Score, Area Under ROC Curve (AUC), Receiver Operating Characteristic (ROC), Διαθέσιμο στην ηλεκτρονική πλατφόρμα: https://www.ml-science.com/confusion-matrix

Παράρτημα - Κώδικας

Εφαρμογή Random Forest

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import time

from sklearn import svm

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, classification_report, accuracy_score

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import cross_val_score

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy_score, precision_score, recall_score, log_loss,
mean_absolute_error, mean_squared_error, roc_auc_score, confusion_matrix


csv =

r'C:\Users\Lenovo\Desktop\Documents\Papei\Διπλωματική\diabetes_prediction_dataset.cs
v'

data = pd.read_csv(csv, sep=',', header = 0)

print(data.head())


#Check for missing values

missing_values = data.isna().sum()

print(missing_values)
```

```
# Coding of the categorical variables "gender" and "smoking_history"

label_encoder = LabelEncoder()

data['gender'] = label_encoder.fit_transform(data['gender'])

data['smoking_history'] = label_encoder.fit_transform(data['smoking_history'])


# Define the variables as characteristics (x) and target (y)

X = data[['gender', 'age','hypertension', 'heart_disease', 'smoking_history', 'smoking_history',
'bmi', 'HbA1c_level', 'blood_glucose_level']]  # Customize the columns of the attributes


# Replace 'target_column' with the actual target column

y = data['diabetes']

#splitting data into training and test sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)


#standardizing data

sc = StandardScaler()

X_train_std = sc.fit_transform(X_train)

X_test_std = sc.transform(X_test)


# Step 1: Create a Random Forest Classifier

rf_classifier = RandomForestClassifier(random_state=42)


start_time_randf = time.time()

# Step 2: Train the Random Forest model on the training data

rf_classifier.fit(X_train_std, y_train)


end_time_randf = time.time()

execution_time = end_time_randf - start_time_randf

print(f"Ο χρόνος εκτέλεσης ήταν: {execution_time} δευτερόλεπτα")
```

```
# Make predictions on the test set
y_pred = rf_classifier.predict(X_test_std)
y_pred_proba = rf_classifier.predict_proba(X_test_std)[:, 1]

# Calculate overall accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Overall Accuracy: {accuracy}')

# Calculate overall precision
precision = precision_score(y_test, y_pred)
print(f'Overall Precision: {precision}')

# Calculate overall recall
recall = recall_score(y_test, y_pred)
print(f'Overall Recall: {recall}')

# Calculate log loss
logloss = log_loss(y_test, y_pred_proba)
print(f'Log Loss: {logloss}')

# Calculate Mean Absolute Error
mae = mean_absolute_error(y_test, y_pred)
print(f'MAE: {mae}')

# Calculate Root Mean Squared Error
rmse = mean_squared_error(y_test, y_pred, squared=False)
print(f'RMSE: {rmse}')

# Calculate AUC
auc = roc_auc_score(y_test, y_pred_proba)
print(f'AUC: {auc}')
```

```
# Calculate F1-score for class 1
f1 = 2 * (precision * recall) / (precision + recall)
print(f'F1-score : {f1}')

# Calculate True Positive Rate and False Positive Rate
conf_matrix = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = conf_matrix.ravel()
tpr = tp / (tp + fn)
fpr = fp / (fp + tn)
print(f'True Positive Rate (TPR): {tpr}')
print(f'False Positive Rate (FPR): {fpr}')

# Create a diagram for the accuracy
plt.figure(figsize=(8, 6))
plt.title('Random Forest Classifier Accuracy')
plt.xlabel('Number of Trees')
plt.ylabel('Accuracy')
plt.grid(True)

# Calculate and plot accuracy for different numbers of trees
trees_range = range(1, 101)
accuracy_scores = []

for n_trees in trees_range:
    rf_classifier = RandomForestClassifier(n_estimators=n_trees, random_state=42)
    rf_classifier.fit(X_train_std, y_train)
    y_pred = rf_classifier.predict(X_test_std)
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)
```

```
plt.plot(trees_range, accuracy_scores, marker='o', linestyle='-', color='b')

plt.show()
```

Εφαρμογή KNN

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import time

from sklearn import svm

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, classification_report, accuracy_score

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import cross_val_score

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy_score, precision_score, recall_score, log_loss,
mean_absolute_error, mean_squared_error, roc_auc_score, confusion_matrix


csv =
r'C:\Users\Lenovo\Desktop\Documents\Papei\Διπλωματική\diabetes_prediction_dataset.cs
v'

data = pd.read_csv(csv, sep=',', header = 0)

print(data.head())


#Check for missing values

missing_values = data.isna().sum()

print(missing_values)


# Coding of the categorical variables "gender" and "smoking_history"

label_encoder = LabelEncoder()

data['gender'] = label_encoder.fit_transform(data['gender'])
```

```
data['smoking_history'] = label_encoder.fit_transform(data['smoking_history'])


# Define the variables as characteristics (x) and target (y)

X = data[['gender', 'age','hypertension', 'heart_disease', 'smoking_history', 'smoking_history',
'bmi', 'HbA1c_level', 'blood_glucose_level']]  # Customize the attribute columns

y = data['diabetes']  # Replace 'target_column' with the actual target column


#splitting data into training and test sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)


#standardizing data

sc = StandardScaler()

X_train_std = sc.fit_transform(X_train)

X_test_std = sc.transform(X_test)


#creating list to store the cross validation scores

cv_scores = []


#performing k-fold cross validation

k_range = (1,3,5,7,10,12,16,18)

for k in k_range:

    knn = KNeighborsClassifier(n_neighbors=k)

    scores = cross_val_score(knn, X_train_std, y_train, cv=2, scoring='accuracy')

    cv_scores.append(scores.mean())




# find the best k value

max_accuracy = max(cv_scores)

optimal_k = cv_scores.index(max_accuracy) + 1

print('The best value of K is {}, with an accuracy of {}'.format(optimal_k, max_accuracy))
```

```
#fitting knn model
knn = KNeighborsClassifier(n_neighbors=optimal_k)
start_time_knn = time.time()
knn.fit(X_train_std, y_train)
end_time_knn = time.time()
execution_time = end_time_knn - start_time_knn
print(f"Ο χρόνος εκτέλεσης ήταν: {execution_time} δευτερόλεπτα")

#predicting the test data
y_pred = knn.predict(X_test_std)

#Confusion matrix code
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
print(confusion_matrix(y_test, y_pred))

# Precision, Recall, F1-score
from sklearn.metrics import precision_score, recall_score, f1_score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# AUC-ROC
from sklearn.metrics import roc_auc_score
y_pred_proba = knn.predict_proba(X_test_std)[:, 1] # Probability estimates of the positive class
```

```
auc_roc = roc_auc_score(y_test, y_pred_proba)
print("AUC-ROC:", auc_roc)

# True Positive Rate (TPR) and False Positive Rate (FPR)
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
print("True Positive Rate (TPR):", tpr)
print("False Positive Rate (FPR):", fpr)

# Log Loss
from sklearn.metrics import log_loss
logloss = log_loss(y_test, y_pred_proba)
print("Log Loss:", logloss)

# Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE)
mae = mean_absolute_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
print("MAE:", mae)
print("RMSE:", rmse)

#plotting the graph
plt.plot(k_range, cv_scores,color='royalblue', linestyle='solid', marker='o',
        markerfacecolor='darkgoldenrod', markersize=7)
plt.xlabel('number k')
plt.ylabel('Accuracy')
plt.title('Fold_2')

for x,y in zip(k_range, cv_scores):
   label = "{:.2f}".format(y)
   plt.annotate(label, # this is the text
              (x,y), # this is the point to label
```

```
            textcoords="offset points", # how to position the text
            xytext=(0,5), # distance from text to points (x,y)
            ha='center') # horizontal alignment can be left, right or center
plt.show()
```

Εφαρμογή SVM

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, accuracy_score
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score, log_loss, mean_absolute_error, mean_squared_error, roc_auc_score, confusion_matrix

csv = r'C:\Users\Lenovo\Desktop\Documents\Papei\Διπλωματική\diabetes_prediction_dataset.csv'
data = pd.read_csv(csv, sep=',', header = 0)
print(data.head())

#Check for missing values
missing_values = data.isna().sum()
print(missing_values)# Κωδικοποίηση των κατηγορικών μεταβλητών "gender" και "smoking_history"
```

```
label_encoder = LabelEncoder()

data['gender'] = label_encoder.fit_transform(data['gender'])

data['smoking_history'] = label_encoder.fit_transform(data['smoking_history'])


#Replace 'Male' with 0 and 'Female' with 1 in the 'Gender' column

#data['gender'] = data['gender'].map({'Male': 0, 'Female': 1})


#Replace 'never' with 0, 'current' with 1, 'former' with 2, 'No Info' with 3 in the 'smoking_historys' column

#data['smoking_history'] = data['smoking_history'].map({'never': 0, 'current': 1, 'former' : 2, 'No Info' : 3})


# Define the variables as characteristics (x) and target (y)

X = data[['gender', 'age','hypertension', 'heart_disease', 'smoking_history', 'smoking_history', 'bmi', 'HbA1c_level', 'blood_glucose_level']]  # Customize the attribute columns

y = data['diabetes']  # Replace 'target_column' with the actual target column


#standardize the data

#sc = StandardScaler()

#X = sc.fit_transform(X)


# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Create an SVM classifier

svc = SVC(kernel='linear')

start_time_svm = time.time()

svc.fit(X, y)

end_time_svm = time.time()

execution_time = end_time_svm - start_time_svm

print(f"Ο χρόνος εκτέλεσης ήταν: {execution_time} δευτερόλεπτα")
```

```
# Conception for the calculation of the probability of correct classification
def predict_accuracy(c_val):
    svc.C = c_val
    svc.fit(X, y)
    return svc.score(X, y)

# Creation of the list of C prices
c_list = (0.1, 0.5, 0.7, 0.9, 1.2, 1.5, 1.7, 1.9)

# Calculation of the probability of correct classification for each value of C
accuracy_list = []
for c_val in c_list:
    accuracy = predict_accuracy(c_val)
    accuracy = accuracy*100
    accuracy_list.append(accuracy)
print(accuracy_list)

# Predict the test data
y_pred = svc.predict(X_test)
y_pred_proba = svc.predict_proba(X_test)[:, 1]  # Probability estimates of the positive class

# Precision, Recall, F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# AUC-ROC
```

```
auc_roc = roc_auc_score(y_test, y_pred_proba)
print("AUC-ROC:", auc_roc)

# Calculate False Positive Rate (FPR) and True Positive Rate (TPR) for ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
print("False Positive Rate (FPR):", fpr)
print("True Positive Rate (TPR):", tpr)

# Log Loss
logloss = log_loss(y_test, y_pred_proba)
print("Log Loss:", logloss)

# οπτικοποιηση των αποτελεσματων
plt.plot(c_list, accuracy_list,color='royalblue', linestyle='solid', marker='o',
        markerfacecolor='darkgoldenrod', markersize=7)
plt.xlabel('number c')
plt.ylabel('Accuracy')
plt.title('svm')
plt.show()
```

Εφαρμογή Logistic Regression

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score, log_loss,
mean_absolute_error, mean_squared_error, roc_auc_score, confusion_matrix

#csv =
r'C:\Users\Lenovo\Desktop\Documents\Papei\Διπλωματική\diabetes_prediction_dataset.cs
v'
#data = pd.read_csv('/content/drive/MyDrive/diabetes_prediction_dataset.csv')
data = pd.read_csv('/content/sample_data/diabetes_prediction_dataset.csv')

print(data.head())

#Check for missing values
missing_values = data.isna().sum()
print(missing_values)

# Κωδικοποίηση των κατηγορικών μεταβλητών "gender" και "smoking_history"
```

```
label_encoder = LabelEncoder()

data['gender'] = label_encoder.fit_transform(data['gender'])

data['smoking_history'] = label_encoder.fit_transform(data['smoking_history'])


# Ορίστε τις μεταβλητές ως χαρακτηριστικά (X) και τον στόχο (y)

X = data[['gender', 'age','hypertension', 'heart_disease', 'smoking_history', 'smoking_history', 'bmi', 'HbA1c_level', 'blood_glucose_level']] # Προσαρμόστε τις στήλες των χαρακτηριστικών

y = data['diabetes']  # Αντικαταστήστε το 'target_column' με την πραγματική στήλη στόχου


#splitting data into training and test sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)


#standardizing data

sc = StandardScaler()

X_train_std = sc.fit_transform(X_train)

X_test_std = sc.transform(X_test)



from sklearn.linear_model import LogisticRegression


# Step 1: Create a Logistic Regression Classifier

logreg_classifier = LogisticRegression(random_state=42)


start_time_logreg = time.time()

# Step 2: Train the Logistic Regression model on the training data

logreg_classifier.fit(X_train_std, y_train)


end_time_logreg = time.time()

execution_time_logreg = end_time_logreg - start_time_logreg

print(f"Ο χρόνος εκτέλεσης ήταν: {execution_time_logreg} δευτερόλεπτα")
```

```
# Make predictions on the test set
y_pred_logreg = logreg_classifier.predict(X_test_std)
y_pred_proba_logreg = logreg_classifier.predict_proba(X_test_std)[:, 1]

# Calculate overall accuracy
accuracy_logreg = accuracy_score(y_test, y_pred_logreg)
print(f'Overall Accuracy: {accuracy_logreg}')

# Calculate overall precision
precision_logreg = precision_score(y_test, y_pred_logreg)
print(f'Overall Precision: {precision_logreg}')

# Calculate overall recall
recall_logreg = recall_score(y_test, y_pred_logreg)
print(f'Overall Recall: {recall_logreg}')

# Calculate log loss
logloss_logreg = log_loss(y_test, y_pred_proba_logreg)
print(f'Log Loss: {logloss_logreg}')

# Calculate Mean Absolute Error
mae_logreg = mean_absolute_error(y_test, y_pred_logreg)
print(f'MAE: {mae_logreg}')

# Calculate Root Mean Squared Error
rmse_logreg = mean_squared_error(y_test, y_pred_logreg, squared=False)
print(f'RMSE: {rmse_logreg}')

# Calculate AUC
auc_logreg = roc_auc_score(y_test, y_pred_proba_logreg)
```

```
print(f'AUC: {auc_logreg}')

# Calculate F1-score for class 1
f1_logreg = 2 * (precision_logreg * recall_logreg) / (precision_logreg + recall_logreg)
print(f'F1-score: {f1_logreg}')

# Calculate True Positive Rate and False Positive Rate
conf_matrix_logreg = confusion_matrix(y_test, y_pred_logreg)
tn_logreg, fp_logreg, fn_logreg, tp_logreg = conf_matrix_logreg.ravel()
tpr_logreg = tp_logreg / (tp_logreg + fn_logreg)
fpr_logreg = fp_logreg / (fp_logreg + tn_logreg)
print(f'True Positive Rate (TPR): {tpr_logreg}')
print(f'False Positive Rate (FPR): {fpr_logreg}')
print(f': {tp_logreg}')
print(f': {tn_logreg}')
print(f': {fp_logreg}')
print(f': {fn_logreg}')

# Create a diagram for the accuracy
plt.figure(figsize=(8, 6))
plt.title('Logistic Regression Classifier Accuracy')
plt.xlabel('Number of Iterations')
plt.ylabel('Accuracy')
plt.grid(True)

# Define iterations_range
iterations_range = [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

# Calculate and plot accuracy for different iterations
accuracy_scores_logreg = []
```

```
for n_iterations in iterations_range:
    logreg_classifier = LogisticRegression(max_iter=n_iterations, random_state=42)
    logreg_classifier.fit(X_train_std, y_train)
    y_pred_logreg = logreg_classifier.predict(X_test_std)
    accuracy_logreg = accuracy_score(y_test, y_pred_logreg)
    accuracy_scores_logreg.append(accuracy_logreg)

plt.plot(iterations_range, accuracy_scores_logreg, marker='o', linestyle='-', color='r')
plt.show()
```

Εφαρμογή Decision Tree

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score, log_loss,
mean_absolute_error, mean_squared_error, roc_auc_score, confusion_matrix

#csv =
r'C:\Users\Lenovo\Desktop\Documents\Papei\Διπλωματική\diabetes_prediction_dataset.cs
v'
#data = pd.read_csv('/content/drive/MyDrive/diabetes_prediction_dataset.csv')
data = pd.read_csv('/content/sample_data/diabetes_prediction_dataset.csv')
```

```
print(data.head())


#Check for missing values

missing_values = data.isna().sum()

print(missing_values)


# Κωδικοποίηση των κατηγορικών μεταβλητών "gender" και "smoking_history"

label_encoder = LabelEncoder()

data['gender'] = label_encoder.fit_transform(data['gender'])

data['smoking_history'] = label_encoder.fit_transform(data['smoking_history'])


# Ορίστε τις μεταβλητές ως χαρακτηριστικά (X) και τον στόχο (y)

X = data[['gender', 'age','hypertension', 'heart_disease', 'smoking_history', 'smoking_history', 'bmi', 'HbA1c_level', 'blood_glucose_level']] # Προσαρμόστε τις στήλες των χαρακτηριστικών

y = data['diabetes']  # Αντικαταστήστε το 'target_column' με την πραγματική στήλη στόχου


#splitting data into training and test sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)


#standardizing data

sc = StandardScaler()

X_train_std = sc.fit_transform(X_train)

X_test_std = sc.transform(X_test)


from sklearn.tree import DecisionTreeClassifier


# Step 1: Create a Decision Tree Classifier

dt_classifier = DecisionTreeClassifier(random_state=42)


start_time_dt = time.time()
```

```
# Step 2: Train the Decision Tree model on the training data
dt_classifier.fit(X_train_std, y_train)

end_time_dt = time.time()
execution_time_dt = end_time_dt - start_time_dt
print(f"Ο χρόνος εκτέλεσης ήταν: {execution_time_dt} δευτερόλεπτα")

# Make predictions on the test set
y_pred_dt = dt_classifier.predict(X_test_std)
y_pred_proba_dt = dt_classifier.predict_proba(X_test_std)[:, 1]

# Calculate overall accuracy
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print(f'Overall Accuracy: {accuracy_dt}')

# Calculate overall precision
precision_dt = precision_score(y_test, y_pred_dt)
print(f'Overall Precision: {precision_dt}')

# Calculate overall recall
recall_dt = recall_score(y_test, y_pred_dt)
print(f'Overall Recall: {recall_dt}')

# Calculate log loss
logloss_dt = log_loss(y_test, y_pred_proba_dt)
print(f'Log Loss: {logloss_dt}')

# Calculate Mean Absolute Error
mae_dt = mean_absolute_error(y_test, y_pred_dt)
print(f'MAE: {mae_dt}')
```

```
# Calculate Root Mean Squared Error
rmse_dt = mean_squared_error(y_test, y_pred_dt, squared=False)
print(f'RMSE: {rmse_dt}')

# Calculate AUC
auc_dt = roc_auc_score(y_test, y_pred_proba_dt)
print(f'AUC: {auc_dt}')

# Calculate F1-score for class 1
f1_dt = 2 * (precision_dt * recall_dt) / (precision_dt + recall_dt)
print(f'F1-score: {f1_dt}')

# Calculate True Positive Rate and False Positive Rate
conf_matrix_dt = confusion_matrix(y_test, y_pred_dt)
tn_dt, fp_dt, fn_dt, tp_dt = conf_matrix_dt.ravel()
tpr_dt = tp_dt / (tp_dt + fn_dt)
fpr_dt = fp_dt / (fp_dt + tn_dt)
print(f'True Positive Rate (TPR): {tpr_dt}')
print(f'False Positive Rate (FPR): {fpr_dt}')
print(f': {tp_dt}')
print(f': {tn_dt}')
print(f': {fp_dt}')
print(f': {fn_dt}')

# Create a diagram for the accuracy
plt.figure(figsize=(8, 6))
plt.title('Decision Tree Classifier Accuracy')
plt.xlabel('Tree Depth')
plt.ylabel('Accuracy')
plt.grid(True)
```

```
# Calculate and plot accuracy for different tree depths
depths_range = range(1, 21)
accuracy_scores_dt = []

for depth in depths_range:
    dt_classifier = DecisionTreeClassifier(max_depth=depth, random_state=42)
    dt_classifier.fit(X_train_std, y_train)
    y_pred_dt = dt_classifier.predict(X_test_std)
    accuracy_dt = accuracy_score(y_test, y_pred_dt)
    accuracy_scores_dt.append(accuracy_dt)

plt.plot(depths_range, accuracy_scores_dt, marker='o', linestyle='-', color='g')
plt.show()
```

Συγκρίσεις μοντέλων ως προς διάφορες μετρικές

```
import matplotlib.pyplot as plt

# Υποθέτουμε ότι έχουμε τους χρόνους εκτέλεσης σε λίστες execution_time_knn, execution_time_randomforest, execution_time_svm

models = ['KNN', 'Random Forest', 'SVM', 'Logistic Regression', 'Decision Trees']
execution_times = [execution_time_knn, execution_time_rf, execution_time_svm, execution_time_logreg, execution_time_dt]

plt.plot(models, execution_times, marker='o', linestyle='-', color='blue', linewidth=2)
plt.xlabel('Models')
plt.ylabel('Execution Time (seconds)')
plt.title('Execution Time for Different Models')
plt.grid(True)
plt.show()
```

```
import matplotlib.pyplot as plt

models = ['KNN', 'Random Forest', 'SVM', 'Logistic Regression', 'Decision Trees']
accuracies = [accuracy_knn*100, accuracy_rf*100, accuracy_svm, accuracy_logreg, accuracy_dt]

plt.plot(models, accuracies, marker='o', linestyle='-', color='blue', linewidth=2)
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy for Different Models')
plt.grid(True)
plt.show()

import matplotlib.pyplot as plt

models = ['KNN', 'Random Forest', 'SVM', 'Logistic Regression', 'Decision Trees']
f1 = [f1_knn, f1_rf, f1_svm, f1_logreg, f1_dt]
auc = [auc_roc_knn, auc_rf, auc_roc_svm, auc_logreg, auc_dt]

plt.plot(models, f1, marker='o', linestyle='-', color='blue', linewidth=2, label='F1 Score')
plt.plot(models, auc, marker='x', linestyle='--', color='red', linewidth=2, label='AUC')

plt.xlabel('Models')
plt.ylabel('Scores')
plt.title('F1 Score and AUC for Different Models')
plt.legend()
plt.grid(True)
plt.show()

import matplotlib.pyplot as plt
```

```
models = ['KNN', 'Random Forest', 'SVM', 'Logistic Regression', 'Decision Trees']
mae = [mae_knn, mae_rf, mae_svm, mae_logreg, mae_dt]
rmse = [rmse_knn, rmse_rf, rmse_svm, rmse_logreg, rmse_dt ]

plt.plot(models, mae, marker='o', linestyle='-', color='blue', linewidth=2, label='MAE')
plt.plot(models, rmse, marker='x', linestyle='--', color='red', linewidth=2, label='RMSE')

plt.xlabel('Models')
plt.ylabel('Scores')
plt.title('mae and rmse for Different Models')
plt.legend()
plt.grid(True)
plt.show()
```