ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

“ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ & ΥΠΗΡΕΣΙΕΣ”

Τεχνικές Data Mining και προβλεπτική αναλυτική Σακχαρώδους Διαβήτη

Από
Γεννούζη Γεράσιμο -Ιάκωβο
ΑΜ: ΜΕ2203

Υποβάλλεται
για την εκπλήρωση των προϋποθέσεων λήψης
Μεταπτυχιακού Διπλώματος
στην ειδίκευση «ΜΔΑ»
του ΠΜΣ “Πληροφοριακά Συστήματα & Υπηρεσίες”
στο
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
03 2024

Επιβλέπων: Φιλιππάκης Μιχαήλ
Ακαδημαϊκή Θέση: Καθηγητής

ΣΕΛΙΔΑ ΕΓΚΥΡΟΤΗΤΑΣ

Ονοματεπώνυμο Φοιτητή/Φοιτήτριας: Γεννούζης Γεράσιμος Ιάκωβος

Τίτλος Μεταπτυχιακής Διπλωματικής Εργασίας: Τεχνικές data mining και προβλεπτική αναλυτική σακχαρώδους διαβήτη.

Η παρούσα Μεταπτυχιακή Διπλωματική Εργασία υποβάλλεται ως μερική εκπλήρωση των απαιτήσεων του Προγράμματος Μεταπτυχιακών Σπουδών "Πληροφοριακά Συστήματα & Υπηρεσίες" του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς και εγκρίθηκε στις …… [ημερομηνία έγκρισης] από τα μέλη της Εξεταστικής Επιτροπής.

Εξεταστική Επιτροπή

Επιβλέπων (Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Πειραιώς) Φιλιππάκης Μιχαήλ, Καθηγητής

Μέλος Εξεταστικής Επιτροπής: Χαλκίδη Μαρία, Αναπληρώτρια Καθηγήτρια

Μέλος Εξεταστικής Επιτροπής: Κυριαζής Δημοσθένης, Καθηγητής

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΑΥΘΕΝΤΙΚΟΤΗΤΑΣ

Ο Γεννούζης Γεράσιμος Ιάκωβος, γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπεύθυνα ότι η παρούσα εργασία με τίτλο «Τεχνικές data mining και προβλεπτική αναλυτική σακχαρώδους διαβήτη», αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές που έχω χρησιμοποιήσει, έχουν δηλωθεί κατάλληλα στις βιβλιογραφικές παραπομπές και αναφορές. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.
Επιπλέον *δηλώνω υπεύθυνα ότι η συγκεκριμένη Μεταπτυχιακή Διπλωματική Εργασία έχει συγγραφεί από εμένα προσωπικά και δεν έχει υποβληθεί ούτε έχει αξιολογηθεί στο πλαίσιο κάποιου άλλου μεταπτυχιακού ή προπτυχιακού τίτλου σπουδών, στην Ελλάδα ή στο εξωτερικό. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου. Σε κάθε περίπτωση, αναληθούς ή ανακριβούς δηλώσεως, υπόκειμαι στις συνέπειες που προβλέπονται τις διατάξεις που προβλέπει η Ελληνική και Κοινοτική Νομοθεσία περί πνευματικής ιδιοκτησίας.*

Ο ΔΗΛΩΝ

Ονοματεπώνυμο: Γεννούζης Γεράσιμος Ιάκωβος
Αριθμός Μητρώου: ΜΕ2203
Υπογραφή:

Ευχαριστίες

Με την παρούσα διπλωματική εργασία ολοκληρώνεται το «ταξίδι» στο μεταπτυχιακό πρόγραμμα σπουδών «Πληροφοριακά Συστήματα & Υπηρεσίες» του Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς. Η παρούσα διπλωματική εργασία εκπονήθηκε από τον μεταπτυχιακό φοιτητή Γεννούζη Γεράσιμο Ιάκωβο, κατά το ακαδημαϊκό έτος 2023 – 2024, υπό την επίβλεψη του καθηγητή Φιλιππάκη Μιχαήλ. Θα ήθελα να εκφράσω τις ευχαριστίες και την βαθιά ευγνωμοσύνη μου στον καθηγητή μου για την ανάθεση του θέματος, τις γνώσεις και την βοήθεια που μου παρείχε με κάθε μέσο καθώς και τον χρόνο που διέθεσε ώστε να πραγματοποιηθεί με επιτυχία η διεκπεραίωση της. Ένα επιπλέον ευχαριστώ και για τον χρόνο φοίτησης μου στο εν λόγω πρόγραμμα, καθώς ο εν λόγω καθηγητής ήταν δίπλα στους φοιτητές παρέχοντας κάθε δυνατή υποστήριξη.

Ιδιαίτερες ευχαριστίες στην οικογένεια μου και στους φίλους μου για την κατανόηση που έδειξαν για τον περιορισμένο χρόνο που διέθετα καθώς και την συνεχή υποστήριξη σε οικονομικό και ψυχολογικό επίπεδο.

Πίνακας Περιεχομένων

Περίληψη

Ο σακχαρώδης διαβήτης είναι μια διαδεδομένη χρόνια νόσος με σημαντικές επιπτώσεις στη δημόσια υγεία και την ατομική ευημερία. Τα τελευταία χρόνια, οι τεχνικές μηχανικής μάθησης (ΜΜ) έχουν αναδειχθεί ως ισχυρά εργαλεία για την πρόβλεψη των αποτελεσμάτων του διαβήτη, προσφέροντας τη δυνατότητα βελτίωσης της έγκαιρης ανίχνευσης. Αυτή η εργασία παρουσιάζει μια ολοκληρωμένη αξιολόγηση διαφόρων μοντέλων ΜΜ, συμπεριλαμβανομένης της λογιστικής παλινδρόμησης (LR), των μηχανών διανυσμάτων υποστήριξης (SVM), του τυχαίου δάσους (RF), των k-πλησιέστερων γειτόνων (KNN), των πολυστρωματικών perceptrons (MLPs) και του μοντέλου gradient boosting (GB), στο πλαίσιο της πρόβλεψης του διαβήτη. Αξιοποιώντας εκτενώς τις δυνατότητες της βιβλιοθήκης Scikit-learn στην Python, αναλύουμε την απόδοση αυτών των μοντέλων χρησιμοποιώντας το σύνολο δεδομένων PIMA και διερευνούμε τον αντίκτυπο των τεχνικών κλιμάκωσης όπως η κανονικοποίηση (Normalization), η προτοτυποποίηση(Standardization) και η μηχανική χαρακτηριστικών (Feature Engineer- τεχνικές Binning και One hot encoding). Τα ευρήματά μας αναδεικνύουν ότι το μοντέλο τυχαίου δάσους (RF) ως τον πιο αποτελεσματικό αλγόριθμο, επιτυγχάνοντας ακρίβεια 85% όταν συνδυάζεται με προτοτυποποίηση και μηχανική χαρακτηριστικών (Feature Engineer- τεχνικές Binning και One hot encoding). Αυτή η έρευνα συμβάλλει στον αυξανόμενο όγκο γνώσεων σχετικά με τις αναλύσεις υγειονομικής περίθαλψης που βασίζονται σε ΜΜ παρέχοντας πληροφορίες για τα δυνατά σημεία και τους περιορισμούς διαφορετικών αλγορίθμων για την πρόβλεψη του διαβήτη. Επιπλέον, η μελέτη μας προσφέρει πρακτικές οδηγίες για επαγγελματίες υγείας και ερευνητές, διευκολύνοντας την έγκαιρη ανίχνευση και εξατομικευμένες παρεμβάσεις για βελτιωμένη διαχείριση του διαβήτη. Μέσω του προσδιορισμού μελλοντικών κατευθύνσεων έρευνας, συμπεριλαμβανομένης της εξατομικευμένης ιατρικής και της βελτίωσης των μεθόδων προεπεξεργασίας, αυτή η εργασία στοχεύει να τονώσει τη συνεχή καινοτομία στον τομέα των αναλύσεων υγειονομικής περίθαλψης που βασίζονται σε ΜΜ και να προωθήσει τη διεπιστημονική συνεργασία για την αντιμετώπιση των προκλήσεων της διαχείρισης χρόνιων ασθενειών.

Λέξεις κλειδιά: Σακχαρώδης διαβήτης, μηχανική μάθηση, λογιστική παλινδρόμηση,μηχανών διανυσμάτων υποστήριξης , τυχαίο δάσος, k-πλησιέστερων γειτόνων, πολυστρωματικών perceptrons gradient boosting, κανονικοποίηση, προτοτυποποίηση, τεχνική κλιμάκωσης, μηχανική χαρακτηριστικών

Abstract

Diabetes mellitus is a prevalent chronic disease with significant implications for public health and individual well-being. In recent years, machine learning (ML) techniques have emerged as powerful tools for predicting diabetes outcomes, offering the potential to improve early detection. This paper presents a comprehensive evaluation of various ML models, including logistic regression (LR), support vector machines (SVM), random forest (RF), k-nearest neighbors (KNN), multilayer perceptrons (MLPs), and of the gradient boosting (GB) model, in the context of diabetes prediction. Making extensive use of the capabilities of the Scikit-learn library in Python, we analyze the performance of these models using the PIMA dataset and investigate the impact of preprocessing techniques such as Normalization, Standardization, and Feature Engineer (Binning and One hot encoding techniques). Our findings highlight the random forest (RF) model as the most efficient algorithm, achieving 85% accuracy when combined with feature engineering (Binning and One hot encoding techniques). This research contributes to the growing body of knowledge on ML-based healthcare analytics by providing insight into the strengths and limitations of different algorithms for diabetes prediction. Furthermore, our study offers practical guidance for healthcare professionals and researchers, facilitating early detection and personalized interventions for improved diabetes management. By identifying future research directions, including personalized medicine, and improving pretreatment methods, this work aims to stimulate continued innovation in the field of ML-based healthcare analytics and promote interdisciplinary collaboration to address the challenges of chronic disease management.

Keywords: Diabetes mellitus, machine learning, logistic regression, support vector machines , random forest, k-nearest neighbors, multilayer perceptrons , gradient boosting, normalization, standardization, feature scaling, feature engineering

Πίνακας Εικόνων

Πίνακας Σχεδίων

Πίνακας Εξισώσεων

Πίνακας γραφικών παραστάσεων

Κεφάλαιο 1: Εισαγωγή

Σε αυτό το κεφάλαιο πραγματοποιείται η εισαγωγή του προβλήματος που θα ασχοληθούμε στην παρούσα εργασία, γίνεται ο ορισμός του προβλήματος καθώς και παρουσιάζεται η δομή της εργασίας. Τέλος παρουσιάζεται και η πιθανή μελλοντική συνεισφορά της.

1.1. Εισαγωγή

Στις μέρες μας μεγάλες διαστάσεις έχει πάρει η χρήση της τεχνολογίας στην υγειονομική περίθαλψη. Ο συνδυασμός προηγμένης τεχνολογίας και ιατρικής επιστήμης έχει προαναγγείλει μια νέα εποχή ιατρικής ακριβείας και προγνωστικής ανάλυσης. Μεταξύ των πολλών εφαρμογών της τεχνολογίας στον τομέα της υγείας, η μηχανική μάθηση (ΜΜ) ξεχωρίζει ως μετασχηματιστικό εργαλείο για την πρόβλεψη, τη διάγνωση και την πρόγνωση ασθενειών. Συγκεκριμένα, η διασταύρωση της μηχανικής μάθησης και της νόσου του διαβήτη αποτελεί μια συναρπαστική ευκαιρία να φέρουμε επανάσταση στον τρόπο με τον οποίο κατανοούμε, διαχειριζόμαστε και αποτρέπουμε αυτήν την ευρέως διαδεδομένη χρόνια πάθηση[9].

Ο σακχαρώδης διαβήτης, που χαρακτηρίζεται από μη φυσιολογικά επίπεδα γλυκόζης στο αίμα, αντιπροσωπεύει μια παγκόσμια επιδημία υγείας με βαθιές επιπτώσεις στη δημόσια υγεία και την ατομική ευημερία. Αποτελεί αναμφισβήτητα έναν συνδυασμό γενετικών, περιβαλλοντικών παραγόντων και παραγόντων του τρόπου ζωής. Η νόσος του του διαβήτη θέτει σημαντικές προκλήσεις για την έγκαιρη ανίχνευση και την αποτελεσματική διαχείριση. Ωστόσο, οι πρόσφατες εξελίξεις στις τεχνικές μηχανικής μάθησης προσφέρουν άνευ προηγουμένου δυνατότητες αποκρυπτογράφησης των περίπλοκων μοτίβων που κρύβονται πίσω από την εμφάνιση και την εξέλιξη του διαβήτη [9].

Ο στόχος της παρούσας εργασίας είναι να διερευνήσει την εφαρμογή των αλγορίθμων μηχανικής μάθησης στην πρόβλεψη του διαβήτη, αξιοποιώντας τις γνώσεις που προέκυψαν από περιεκτικές αναλύσεις διαφορετικών συνόλων δεδομένων και προηγμένες μεθοδολογίες μοντελοποίησης. Αξιοποιώντας τη δύναμη της προβλεπτικής αναλυτικής, αυτή η έρευνα προσπαθεί να αναπτύξει ισχυρά μοντέλα ικανά να εντοπίζουν άτομα υψηλού κινδύνου να αναπτύξουν διαβήτη και να διευκολύνει την

έγκαιρη παρέμβαση. Τέλος να θέσει τα θεμέλια για μελλοντικές έρευνες και να δημιουργήσει ιδέες π.χ. για εξατομικευμένη φροντίδα.

1.2. Ορισμός του προβλήματος

Ο διαβήτης είναι μια χρόνια μεταβολική διαταραχή που επηρεάζει εκατομμύρια σε όλο τον κόσμο, έχει αναδειχθεί ως σημαντική ανησυχία για την υγεία τον 21ο αιώνα. Ο διαβήτης χαρακτηρίζεται από αυξημένα επίπεδα σακχάρου στο αίμα, ο διαβήτης διαταράσσει την ικανότητα του σώματος να επεξεργάζεται σωστά τη γλυκόζη, οδηγώντας σε πολλές επιπλοκές εάν αφεθεί ανεξέλεγκτη. Η νόσος αυτή φαίνεται να αυξάνεται σταθερά, τροφοδοτούμενος από την καθιστική ζωή, τις κακές διατροφικές επιλογές και τη γενετική προδιάθεση, ο διαβήτης επιβάλλει σημαντική επιβάρυνση σε άτομα, οικογένειες και συστήματα υγειονομικής περίθαλψης παγκοσμίως. Η κατανόηση της πολύπλευρης φύσης του διαβήτη, συμπεριλαμβανομένων των τύπων, των παραγόντων κινδύνου και των στρατηγικών διαχείρισης, είναι ζωτικής σημασίας για την αποτελεσματική πρόληψη, την έγκαιρη ανίχνευση και την ολοκληρωμένη θεραπεία [1].

1.3. Δομή εργασίας

Η παρούσα εργασία αποτελείται από 7 κεφάλαια.

Στο κεφάλαιο 1 γίνεται μια εισαγωγή στην μηχανική μάθηση και η συνάφειά της στην υγειονομική περίθαλψη, ιδιαίτερα στο πλαίσιο της πρόβλεψης του διαβήτη. Ο διαβήτης εισάγεται ως μια σημαντική ανησυχία για τη δημόσια υγεία, παρακινώντας την εξερεύνηση τεχνικών ΜΜ για προγνωστικά μοντέλα σε αυτόν τον τομέα. Οι στόχοι της μελέτης περιγράφονται, δίνοντας έμφαση στην ανάγκη αξιολόγησης διαφόρων αλγορίθμων ML για την αποτελεσματικότητά τους στην πρόβλεψη του διαβήτη

Στο κεφάλαιο 2 γίνεται μια πιο μια ολοκληρωμένη διερεύνηση του διαβήτη ως ασθένειας, καλύπτοντας την αιτιολογία, τα συμπτώματα και τις επιπλοκές του. Στη συνέχεια, παρέχεται μια λεπτομερής επισκόπηση των τεχνικών μηχανικής εκμάθησης, εστιάζοντας σε προσεγγίσεις εποπτευόμενης μάθησης.

Αναλύονται επιπλέον συγκεκριμένα μοντέλα εποπτευόμενης ΜΜ, που θα χρησιμοποιηθούν για την πρόβλεψη διαβήτη. Επιπλέον, διευκρινίζει τις διαφορές μεταξύ δυαδικής ταξινόμησης, ταξινόμησης πολλαπλών κλάσεων και εργασιών παλινδρόμησης. Τέλος παρουσιάζονται τα κοινά μέτρα αξιολόγησης που χρησιμοποιούνται για την αξιολόγηση της απόδοσης του μοντέλου.

Στο κεφάλαιο 3 πραγματοποιείται μια διεξοδική ανασκόπηση της υπάρχουσας βιβλιογραφίας που σχετίζεται με την πρόβλεψη του διαβήτη χρησιμοποιώντας μεθοδολογίες μηχανικής μάθησης. Δίνεται έμφαση σε μελέτες που χρησιμοποιούν σύνολα δεδομένων διαβήτη και το ευρέως αναφερόμενο σύνολο δεδομένων PIMA

Το κεφάλαιο 4 περιγράφει το περιβάλλον στο οποίο αναπτύχθηκε στην παρούσα εργασία, με επίκεντρο τη χρήση της βιβλιοθήκης Scikit-learn στην Python για την υλοποίηση μοντέλων. Στην συνέχεια παρουσιάζονται συνοπτικά τα βήματα της μεθοδολογίας που θα ακολουθηθούν για την υλοποίηση της εργασίας.

Το κεφάλαιο 5 περιγράφει αναλυτικά την μεθοδολογική προσέγγιση που αναπτύχθηκε στην παρούσα εργασία. Παρουσιάζεται η διαδικασία προεπεξεργασίας δεδομένων, εκπαίδευσης μοντέλων και αξιολόγησης, με ιδιαίτερη έμφαση στους αλγόριθμους ML που επιλέχθηκαν για πειραματισμό. Τέλος, σε αυτό το κεφάλαιο παρουσιάζονται διάφορες τεχνικές προεπεξεργασίας, όπως η κανονικοποίηση, η τυποποίηση και η μηχανική χαρακτηριστικών, που χρησιμοποιούνται για τη βελτίωση της απόδοσης του μοντέλου.

Στο κεφάλαιο 6 παρουσιάζονται τα αποτελέσματα που προέρχονται από την εφαρμογή μοντέλων ML στο σύνολο δεδομένων PIMA. Παρέχεται μια ολοκληρωμένη σύγκριση της απόδοσης του μοντέλου, που διευκολύνεται από την αξιολόγηση διαφόρων μετρήσεων. Στην συνέχεια επισημαίνονται τα μοντέλα με τις καλύτερες επιδόσεις και εξετάζει τον αντίκτυπο των τεχνικών προεπεξεργασίας, συμπεριλαμβανομένης της κανονικοποίησης, της τυποποίησης και της μηχανικής χαρακτηριστικών, στις διάφορες μετρικής απόδοσης.

Στο κεφάλαιο 7 παρουσιάζονται τα συμπεράσματα και οι μελλοντικές κατευθύνσεις. Στην διάρκεια των συμπερασμάτων γίνεται και η συζήτηση πάνω στα αποτελέσματα.

Τέλος υπάρχει η βιβλιογραφία που χρησιμοποιήσαμε προκειμένου να επιτύχουμε την δημιουργία της παρούσας εργασίας καθώς και το παράρτημα που περιλαμβάνει όλο τον κώδικα που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας.

1.4. Συνεισφορά

Με την παρούσα εργασία θα προσπαθήσουμε να πετύχουμε τα μέγιστα δυνατά αποτελέσματα προκειμένου να συμβάλλουμε στην διαδικασία της πρόβλεψης του διαβήτη, επιθυμώντας παράλληλα να αναδείξουμε την μέγιστη σημασία που έχουν οι τεχνικές μηχανικής χαρακτηριστικών και κλιμάκωσης χαρακτηριστικών για την βελτίωση της απόδοσης του εκάστοτε μοντέλου. Έτσι θα συμβάλουμε όχι μόνο στην πρόβλεψη του διαβήτη αλλά και σε παρόμοιες εργασίες που θα ασχοληθούν με ιατρικά δεδομένα.

Κεφάλαιο 2: Βιβλιογραφική επισκόπηση

Σε αυτό το κεφάλαιο γίνεται πλήρης περιγραφή του προβλήματος μέσα από την ανασκόπηση της βιβλιογραφίας που υπάρχει διαθέσιμη. Επιπλέον αναλύονται οι τεχνικές της μηχανικής μάθησης, των τεχνικών εξόρυξης δεδομένων καθώς και τα metrics εκείνα που χρησιμοποιούνται για την αξιολόγηση των μοντέλων.

2.1. Διαβήτης

Ο σακχαρώδης διαβήτης είναι μεταβολική ασθένεια η οποία χαρακτηρίζεται από αύξηση της συγκέντρωσης του σακχάρου στο αίμα (υπεργλυκαιμία) και από διαταραχή του μεταβολισμού της γλυκόζης, των λιπιδίων και των πρωτεϊνών, είτε ως αποτέλεσμα ελαττωμένης έκκρισης ινσουλίνης είτε λόγω ελάττωσης της ευαισθησίας των κυττάρων του σώματος στην ινσουλίνη.[1][6]

Υπάρχουν τρεις κυρίως τύποι σακχαρώδη διαβήτη: ο τύπος 1, ο τύπος 2 και ο διαβήτης της κύησης, ενώ υπάρχουν και άλλοι ειδικοί τύποι (MODY, παγκρεατικός διαβήτης, φαρμακευτικός διαβήτης, διαβήτης από δηλητήρια κλπ). [1][6]

Η γλυκόζη στο αίμα των ατόμων με σακχαρώδη διαβήτη είναι αυξημένη διότι δεν ελαττώνεται με φυσιολογικό τρόπο, δηλαδή δεν μπορεί να χρησιμοποιηθεί από τα κύτταρα του σώματος (των μυών και του λίπους κυρίως). Φυσιολογικά η γλυκόζη είναι το κύριο “καύσιμο” του σώματός μας και χρησιμοποιείται για την παραγωγή της ενέργειας για την κίνησή μας και την λειτουργία των οργάνων μας. Η γλυκόζη στο αίμα αυξάνει μέχρις ενός ορίου μετά το γεύμα και μετά ελαττώνεται. Η γλυκόζη που δεν χρησιμοποιείται φυσιολογικά αποθηκεύεται στο ήπαρ (ως γλυκογόνο) και μπορεί να μετατρέπεται με χημικές αντιδράσεις σε λίπος. Οι κυριότεροι μηχανισμοί είναι οι εξής:

Το κλειδί για την είσοδο της γλυκόζης στα κύτταρα είναι η λειτουργία της ινσουλίνης: αν η παραγωγή της ινσουλίνης από το σώμα μας (η ινσουλίνη παράγεται στο πάγκρεας) δεν είναι ικανοποιητική τότε η

γλυκόζη παραμένει στο αίμα και προκαλεί τα συμπτώματα του διαβήτη. Στην περίπτωση του διαβήτη τύπου 1 ινσουλίνη δεν παράγεται, είτε παράγεται σε ελάχιστη ποσότητα. Στην περίπτωση του διαβήτη τύπου 2 παράγεται μεν αρχικά ινσουλίνη, αλλά συχνά σε μεγάλη ποσότητα και έτσι το πάγκρεας εξαντλεί την παραγωγή του, οπότε σε όψιμο στάδιο, μπορεί να χρειάζεται η χορήγηση της ινσουλίνης με ένεση. Η γλυκόζη στο αίμα αυξάνει υπερβολικά μετά το γεύμα, όταν δεν υπάρχει αρκετή ινσουλίνη για να την "βάλει στα κύτταρα".[1][3][6]

Άλλος λόγος αύξησης της γλυκόζης στο αίμα είναι η αντίσταση των κυττάρων που θα την χρησιμοποιήσουν. Αυτή η παθολογική κατάσταση ονομάζεται "αντίσταση στην ινσουλίνη" και είναι συχνή σε υπέρβαρα και παχύσαρκα άτομα, σε άτομα που δεν ασκούνται και σε καταστάσεις σημαντικού σωματικού και ψυχικού στρες. Ο μηχανισμός αυτός ισχύει κυρίως για το διαβήτη τύπου 2. [1][6]

Ένας ακόμα λόγος αυξημένης γλυκόζης στο αίμα είναι η υπερβολική παραγωγή γλυκόζης από το συκώτι. Φυσιολογικά, όταν δεν τρώμε, το συκώτι παράγει γλυκόζη από το γλυκογόνο των αποθηκών του, όταν στο αίμα μας η γλυκόζη είναι λίγη. Στην περίπτωση του διαβήτη το συκώτι παράγει γλυκόζη ανεξάρτητα από τη γλυκόζη στο αίμα. Για αυτό συχνά τις πρωινές ώρες η γλυκόζη στο αίμα είναι αυξημένη, ενώ δεν έχει προηγηθεί γεύμα. [1][6]

Οι μηχανισμοί αυτοί επηρεάζονται από τον τρόπο ζωής κυρίως στην περίπτωση του διαβήτη τύπου 2 (τί ποσότητες και τί ποιότητας τροφές καταναλώνουμε, πόσο ασκούμαστε, πόσο στρες επιβαρύνει το σώμα μας), αλλά και από την κληρονομικότητα.

Τα πρώτα συμπτώματα του σακχαρώδη διαβήτη είναι [1][2][6]:

πολυουρία

- μπορεί να εκδηλώνεται ως συχνουρία

πολυδιψία

- πολλοί ασθενείς αναφέρουν μεγάλη κατανάλωση χυμών και αναψυκτικών για να καταπολεμήσουν τη δίψα τους, πίνοντας έτσι μεγάλες ποσότητες ζάχαρης

πείνα

- ακόμα κι αν ο ασθενής τρώει πολύ συχνά

απώλεια βάρους

- χωρίς αυτή να επιδιώκεται με δίαιτα (αφορά κυρίως το διαβήτη τύπου 1) απόπνοια οξόνης

μεγάλη κούραση

Συχνά ακόμη μπορεί να υπάρχουν [1][6]:

- θολή όραση
- καθυστερημένη θεραπεία πληγών
- πόνοι ή μουδιασμένα χέρια ή πόδια (συχνότερα σε άτομα με διαβήτη τύπου 2.
- Μερικές φορές αρχική εκδήλωση μπορεί να είναι το “σύνδρομο καρπιαίου σωλήνα” με μούδιασμα ή πόνους στα χέρια, ιδίως τη νύχτα.

Μερικά άτομα με διαβήτη τύπου 2 μπορεί να μην εμφανίζουν πολύ έντονα συμπτώματα ή άλλοτε να τα εμφανίζουν κι άλλοτε όχι.

Ειδικότερα στην περίπτωση του διαβήτη τύπου 1, τα συμπτώματα αυτά μπορεί να είναι ιδιαίτερα έντονα και μετά από σύντομο χρονικό διάστημα, αν δεν αντιμετωπιστεί ο διαβήτης, να οδηγήσουν σε μια βαριά και επικίνδυνη κατάσταση για τη ζωή του ασθενούς, την διαβητική κετοξέωση, που μπορεί να οδηγήσει σε κώμα αν δεν αντιμετωπιστεί με χορήγηση ορού και ινσουλίνης από τη φλέβα. [1][6]

Η έγκαιρη διάγνωση του διαβήτη με αναγνώριση των συμπτωμάτων είναι πολύ σημαντική, ώστε να δοθεί η κατάλληλη θεραπεία και να υποχωρήσουν τα συμπτώματα, αλλά και να προληφθούν οι διαβητικές επιπλοκές. [1][6]

Τα συμπτώματα του διαβήτη εμφανίζονται διότι το σώμα προσπαθεί να αποβάλει τις υπερβολικές ποσότητες γλυκόζης που έχει στο αίμα. Η γλυκόζη αποβάλλεται με τα ούρα μαζί με πολύ νερό (πολυουρία) και ο ασθενής αφυδατώνεται (δίψα, απώλεια βάρους, κούραση). Όταν η γλυκόζη δεν χρησιμοποιείται φυσιολογικά, και προκειμένου να βρει την αναγκαία ενέργεια για τη λειτουργία του, ο οργανισμός καταναλώνει λίπος. Παράγωγο της κατανάλωσης του λίπους είναι οι κετόνες (οξόνη), που προκαλούν τη χαρακτηριστική δύσοσμη αναπνοή, αλλά και την αυξημένη κόπωση, όταν αυξάνονται στο αίμα. [1][6][9]

Η γλυκόζη αίματος μετριέται με mg/dl (χιλιοστόγραμμα ανά δεκατόλιτρο ή μιλιγκράμ ανά ντι-ελ ή μιλιγκράμ τοις εκατό). Για τη διάγνωση του σακχαρώδη διαβήτη αξιολογούνται μόνον οι μετρήσεις του εργαστηρίου και όχι του μετρητή γλυκόζης.

Εικόνα 1: Οικιακός μετρητής γλυκόζης στο αίμα

Η δοκιμασία της ανοχής γλυκόζης είναι μια εξέταση για τη διάγνωση του σακχαρώδη διαβήτη. Γίνεται με λήψη από το στόμα διαλύματος 75 γραμμαρίων γλυκόζης σε ένα ποτήρι νερό – περίπου 200 ml (χιλιοστά του λίτρου) και η μέτρηση γλυκόζης στο φλεβικό αίμα πριν από τη λήψη και δύο ώρες μετά. Η ζυγισμένη σκόνη γλυκόζης πρέπει να διαλύεται αμέσως πριν από την εξέταση στο νερό και με προσοχή (γιατί μπορεί να σχηματίσει σβώλους). Γλυκόζη σε έτοιμο πυκνό διάλυμα σε διάφορες γεύσεις, που μπορεί να κυκλοφορεί στην αγορά, δεν γίνεται δεκτή. Ο εξεταζόμενος περιμένει ήρεμος στο χώρο του εργαστηρίου μέχρι την επόμενη αιμοληψία, δεν λαμβάνει άλλη τροφή, δεν επιτρέπεται να καπνίσει, ούτε να υποβληθεί σε κάποιο σωματικό ή ψυχικό στρες. Κατά τις τρεις ημέρες που προηγούνται ο ασθενής πρέπει να λαμβάνει σχετικά αυξημένες ποσότητες υδατανθράκων, γιατί η αποχή από τους υδατάνθρακες μπορεί να δώσει λανθασμένα αποτελέσματα. Μερικές φορές ζητείται και ενδιάμεση μέτρηση (στη μία ώρα), είτε ζητείται παράλληλα μέτρηση και άλλων ουσιών στο αίμα (όπως ινσουλίνης). [1][6]

Αν στις δύο ώρες βρεθεί γλυκόζη άνω του 200 mg/dl προκύπτει η διάγνωση του διαβήτη, αφ' όσον επιβεβαιωθεί με ένα άλλο κριτήριο, ή με επανάληψη της δοκιμασίας. Τιμές γλυκόζης κάτω του 140 mg/dl είναι φυσιολογικές. Τιμές γλυκόζης μεταξύ 140 και 200 mg/dl χαρακτηρίζονται ως "παθολογική ανοχή στη γλυκόζη" ή προδιαβήτης.

Η δοκιμασία ανοχής γλυκόζης δεν έχει ανεπιθύμητες ενέργειες (παρενέργειες) και μπορεί μόνον να προκαλέσει ναυτία από την λήψη της γλυκόζης. Αν προκύψει παθολογική ανοχή στη γλυκόζη πρέπει να επαναληφθεί αργότερα, ανάλογα με τη σύσταση του ιατρού. Δεν χρειάζεται να γίνεται δοκιμασία ανοχής γλυκόζης εφόσον υπάρχει διάγνωση του διαβήτη. [1][6]

Ο διαβήτης τύπου 1 χαρακτηρίζεται από πλήρη έλλειψη παραγωγής ινσουλίνης από το πάγκρεας του ασθενούς όπως αναφέρεται και παραπάνω. Αυτό γίνεται διότι τα κύτταρα του παγκρέατος που παράγουν ινσουλίνη έχουν καταστραφεί με "ανοσολογικό μηχανισμό" δηλαδή μέσω ουσιών που παράγει το σώμα των ασθενών και στρέφονται εναντίον κυττάρων του ίδιου του οργανισμού (αντισώματα). Συνήθως αφορά νέα άτομα (παιδιά, εφήβους, νέους ενήλικες) και παλαιότερα ονομαζόταν νεανικός διαβήτης, αλλά μπορεί να εμφανιστεί σε κάθε ηλικία, οπότε η ονομασία νεανικός διαβήτης έχει πλέον καταργηθεί. Εκδηλώνεται με τα τυπικά συμπτώματα είτε με διαβητική κετοξέωση, μια βαριά και επικίνδυνη κατάσταση για τη ζωή του ασθενούς, που μπορεί να οδηγήσει σε κώμα αν δεν αντιμετωπιστεί με

χορήγηση ορού και ινσουλίνης από τη φλέβα. Για την επιβεβαίωση της διάγνωσης του διαβήτη τύπου 1 μπορεί να χρειαστεί μέτρηση αντισωμάτων στο αίμα του ασθενούς ή και μέτρηση ινσουλίνης. [1][6][8]

Για το διαβήτη τύπου 1 το μόνο φάρμακο που μπορεί να ρυθμίσει το σάκχαρο είναι η ινσουλίνη, που χορηγείται με ένεση στο δέρμα (υποδόρια ένεση) και μάλιστα σε 3 ή 4 ενέσεις, μία για το σάκχαρο νηστείας και από μία για κάθε γεύμα. Αυτό λέγεται εντατικό σχήμα ινσουλίνης και απαιτεί 3 ή τέσσερεις μετρήσεις την ημέρα και λήψη ινσουλίνης ανάλογα με τις μετρήσεις, την ποσότητα τροφής και τη σωματική δραστηριότητα. Η ινσουλίνη στο διαβήτη τύπου 1 χορηγείται αμέσως μόλις διαγνωστεί ο διαβήτης, αφού ο ασθενής με διαβήτη τύπου 1 κινδυνεύει από κετοξέωση. Με τα σημερινά δεδομένα η χορήγηση ινσουλίνης από το δέρμα (με ένεση ή με αντλία ινσουλίνης) δίδεται δια βίου. Ο διαβήτης τύπου 1 είναι περίπου το 10% των περιπτώσεων διαβήτη.

Ο διαβήτης τύπου 2 αφορά συνήθως άτομα μεγαλύτερης ηλικίας και συχνά παχύσαρκα, που χαρακτηρίζονται από “αντίσταση στην ινσουλίνη” και μπορεί αρχικά να έχουν σημαντική παραγωγή ινσουλίνης, αργότερα όμως έχουν μικρότερη. Παλαιότερα ονομαζόταν διαβήτης των ενηλίκων, αλλά μπορεί να εμφανιστεί σε κάθε ηλικία, ακόμα και σε παιδιά, αν και σπάνια, οπότε η ονομασία διαβήτης των ενηλίκων καταργήθηκε. Ο διαβήτης τύπου 2 μπορεί να εκδηλωθεί με τα τυπικά συμπτώματα σε έντονο η λιγότερο έντονο βαθμό, είτε να μην υπάρχουν συμπτώματα για χρόνια και η διάγνωση να γίνει τυχαία ή με αφορμή μια νοσηλεία ή όταν εκδηλωθεί κάποια διαβητική επιπλοκή. Για τη θεραπεία του διαβήτη τύπου 2, εκτός από την αλλαγή στον τρόπο ζωής (δίαιτα και άσκηση) αρχίζουμε με τη χορήγηση του φαρμάκου μετφορμίνη, που ελαττώνει την αντίσταση στην ινσουλίνη. Αν ο ασθενής δεν φθάσει τους στόχους (γλυκοζυλιωμένης αιμοσφαιρίνης, με καλές τιμές σακχάρου κατά τις μετρήσεις πριν και μετά τα γεύματα), τότε προσθέτουμε και δεύτερο ή και τρίτο φάρμακο, που λειτουργεί με διαφορετικό τρόπο, ή ινσουλίνη ή τελικά χρησιμοποιούμε το εντατικό σχήμα ινσουλίνης, όπως στο διαβήτη τύπου 1. Ο διαβήτης τύπου 2 είναι περίπου το 90% των περιπτώσεων διαβήτη. [1][6][8]

Και οι δύο τύποι διαβήτη μπορεί να είναι κληρονομικοί, αλλά ο διαβήτης τύπου 2 είναι περισσότερο κληρονομικός από το διαβήτη τύπου 1.

Ο διαβήτης τύπου 1 είναι μια πάθηση που οφείλεται στην καταστροφή των κυττάρων του παγκρέατος που παράγουν ινσουλίνη (ονομάζονται βήτα κύτταρα). Η ινσουλίνη είναι η υπεύθυνη ορμόνη για να εισέρχεται η γλυκόζη (το σάκχαρο) από το αίμα στα κύτταρα. Έλλειψη ινσουλίνης έχει ως συνέπεια την παραμονή της γλυκόζης στο αίμα. Ο διαβήτης τύπου 1 αντιμετωπίζεται μόνο με ενέσεις ινσουλίνης. Το ίδιο το σώμα του ατόμου με σακχαρώδη διαβήτη παράγει ουσίες που καταστρέφουν τα κύτταρα που παράγουν ινσουλίνη. Αυτό γίνεται πιθανόν γιατί το αμυντικό σύστημα του πάσχοντος δημιουργεί ουσίες (αντισώματα) εναντίον κάποιου ιού, και τα αντισώματα αυτά καταστρέφουν τα β κύτταρα του παγκρέατος. Αυτή η παθολογική κατάσταση ονομάζεται αυτοανοσία και τα αντισώματα που στρέφονται κατά των κυττάρων του ίδιου του ατόμου που τα παράγει ονομάζονται αυτοαντισώματα, τα οποία μπορεί να ανιχνευθούν στο αίμα. Ο διαβήτης τύπου 1 είναι αυτοάνοσο νόσημα. Πολλά ερωτήματα παραμένουν ακόμη χωρίς απάντηση σχετικά με τα αίτια των αυτοάνοσων νοσημάτων. Άλλοι παράγοντες, όπως το γάλα της αγελάδας και διάφορες τοξίνες, έχουν επίσης κατηγορηθεί. [1][6][8]

Μια εξέταση αίματος για αυτοαντισώματα μπορεί να αποδείξει ότι ένα άτομο με διαβήτη έχει πράγματι σακχαρώδη διαβήτη τύπου 1. Τα αυτοαντισώματα μπορεί να υπάρχουν πριν εκδηλωθεί ο διαβήτης και για ένα χρονικό διάστημα μετά και να εξαφανισθούν αργότερα. Άλλα άτομα (συνήθως συγγενείς ατόμων με διαβήτη) χωρίς διαβήτη μπορεί να έχουν επίσης όμοια αυτοαντισώματα στο αίμα τους, και να μην εμφανίσουν ποτέ διαβήτη. [1][6][8]

Μπορεί ακόμα να γίνει εξέταση αίματος για την παρουσία ινσουλίνης στο αίμα του ασθενούς, όπως και για μια ουσία που ονομάζεται C-πεπτίδιο και παράγεται μαζί με την ινσουλίνη από τα β-κύτταρα. Αν η ινσουλίνη ή το C-πεπτίδιο βρίσκονται σε πολύ μικρά ποσά στο αίμα του εξεταζόμενου, τότε υπάρχει ισχυρή ένδειξη ότι το άτομο αυτό έχει διαβήτη τύπου 1. Η απόδειξη όμως είναι η παρουσία των ειδικών αυτοαντισωμάτων. [1][6][8]

Υπάρχει κληρονομικότητα στο διαβήτη τύπου 1, αλλά δεν είναι μεγάλη. Αν πάσχει ο πατέρας από διαβήτη τύπου 1 η πιθανότητα να εμφανίσει το παιδί διαβήτη τύπου 1 είναι περίπου 6%, και άν πάσχει η μητέρα η πιθανότητα είναι 3%, ενώ αν πάσχουν και οι δύο γονείς, είναι 30%. Αν ένας μονοζυγωτικός δίδυμος έχει διαβήτη τύπου 1, ο όμοιος δίδυμός του θα εμφανίσει επίσης διαβήτη τύπου 1 σε ποσοστό 50% μέχρι να γίνει σαράντα ετών. [1][6][8]

Ο διαβήτης τύπου 1 εκδηλώνεται στα παιδιά συνήθως με πολλά ούρα, νυκτερινή ούρηση, δίψα και αφυδάτωση και απώλεια βάρους η οποία δεν μπορεί να εξηγηθεί, αφού δεν προηγείται δίαιτα. Το άρρωστο παιδί δεν μπορεί να χρησιμοποιήσει τη γλυκόζη για την ενέργεια που χρειάζεται το σώμα του και είναι κουρασμένο. Το σώμα χρησιμοποιεί εναλλακτικά το λίπος για παραγωγή ενέργειας. Από το λίπος παράγονται κετόνες (οξόνη). Οι κετόνες και η αφυδάτωση μπορεί να προκαλέσουν μια σοβαρή επιπλοκή, την διαβητική κετοξέωση. Αν η διαβητική κετοξέωση δεν αντιμετωπισθεί εγκαίρως με ορούς και ινσουλίνη, μπορεί να προκαλέσει μεγάλη αδυναμία, εμέτους και πόνο στην κοιλιά, διαταραχές της συνείδησης και κώμα.[9]

Πρέπει να ρυθμίζουμε το διαβήτη διότι έτσι περιορίζονται τα συμπτώματά του, αλλά και διότι προστατεύουμε σε σημαντικό βαθμό τον ασθενή από τις επιπλοκές του διαβήτη.

Για όλα τα άτομα με διαβήτη η αλλαγή του τρόπου ζωής (δίαιτα και άσκηση) είναι ο πρώτος στόχος. Ο ασθενής πρέπει να μάθει να διακρίνει ποια τρόφιμα ανεβάζουν το σάκχαρό του και να περιορίσει την κατανάλωσή τους. Πρέπει ακόμα να αυξήσει τη σωματική του δραστηριότητα, ελαττώνοντας έτσι την αντίσταση στην ινσουλίνη, ιδίως αν είναι υπέρβαρος ή παχύσαρκος. [1][6][8]

Πρέπει ακόμα να μετρά τη γλυκόζη στο αίμα του και να επιδιώκει μικρότερες τιμές γλυκόζης, με στόχους που εξηγούνται από το γιατρό του. Δυστυχώς η δίαιτα και η σωματική άσκηση δεν αρκούν. Γι αυτό συνιστάται η έναρξη φαρμάκων.

Για το διαβήτη τύπου 1 το μόνο φάρμακο που μπορεί να ρυθμίσει το διαβήτη είναι η ινσουλίνη, που χορηγείται με ένεση στο δέρμα (υποδόρια ένεση) και μάλιστα σε 3 ή 4 ενέσεις, μία για το σάκχαρο νηστείας και από μία για κάθε γεύμα. Αυτό λέγεται εντατικό σχήμα ινσουλίνης και απαιτεί 3 ή τέσσερεις μετρήσεις την ημέρα και λήψη ινσουλίνης ανάλογα με τις μετρήσεις, την ποσότητα τροφής και τη σωματική δραστηριότητα. Η ινσουλίνη στο διαβήτη τύπου 1 χορηγείται αμέσως μόλις διαγνωστεί ο διαβήτης, αφού ο ασθενής με διαβήτη τύπου 1 δεν παράγει ινσουλίνη και κινδυνεύει από κετοξέωση. Για το διαβήτη τύπου 2, αρχίζουμε με τη μετφορμίνη, ένα φάρμακο που ελαττώνει την αντίσταση στην ινσουλίνη. Αν δεν φθάσει στους στόχους, τότε προσθέτουμε και δεύτερο ή και τρίτο φάρμακο, που λειτουργεί με διαφορετικό τρόπο, ή ινσουλίνη ή τελικά χρησιμοποιούμε το εντατικό σχήμα ινσουλίνης, όπως στο διαβήτη τύπου 1. [1],[8]

Εικόνα 2: Σκευάσματα φαρμακευτικού περιεχομένου

Η ρύθμιση του διαβήτη αποδεικνύεται, εκτός από τις καθημερινές μετρήσεις, από μια εξέταση αίματος, τη γλυκοζυλιωμένη αιμοσφαιρίνη, που πρέπει να μετράται τακτικά (κάθε λίγους μήνες), για όλη τη ζωή του ατόμου με διαβήτη. [1]

Στις ΗΠΑ η γλυκοζυλιωμένη αιμοσφαιρίνη (άνω του 6,5%) χρησιμοποιείται ως κριτήριο για τη διάγνωση του διαβήτη, αλλά σε εργαστήρια που έχουν έλεγχο πιστοποίησης. Στην Ελλάδα η γλυκοζυλιωμένη αιμοσφαιρίνη δεν αποτελεί κριτήριο για τη διάγνωση του διαβήτη.

Προδιαβήτης: το στάδιο πρόληψης του διαβήτη

Ενώ στο διαβήτη τύπου 1, που εκδηλώνεται κυρίως στη νεανική ζωή, η πρόληψη δεν είναι εφικτή, στο διαβήτη τύπου 2 υπάρχει αναγνωρισμένο προκλινικό στάδιο, κατά το οποίο η νόσος μπορεί να διαγνωστεί με αξιόπιστες εξετάσεις.

Στις περισσότερες περιπτώσεις προηγείται του διαβήτη μια κατάσταση που συχνά ονομάζεται προδιαβήτης. Αυτή χαρακτηρίζεται είτε από παθολογική γλυκόζη νηστείας [σάκχαρο αίματος μεγαλύτερο από 100 χιλιοστά του 0γραμμαρίου ανά δέκατο του λίτρου (mg/dl), αλλά μικρότερο από 126, οπότε πρόκειται για σακχαρώδη διαβήτη, αν βρεθεί σε δύο μετρήσεις], είτε από παθολογική ανοχή στη γλυκόζη (γλυκόζη αίματος μεταξύ 140 και 200 mg/dl, 2 ώρες μετά από λήψη 75 γραμμαρίων γλυκόζης – αυτή η εξέταση λέγεταικαμπύλη γλυκόζης. Πάνω από 200 είναι διαβήτης). Σε κάθε μία από αυτές τις περιπτώσεις υπάρχει πιθανότητα 30% περίπου να εκδηλωθεί διαβήτης μέσα στα επόμενα 2 χρόνια, αν δεν συνυπάρχουν, και 50% αν συνυπάρχουν. Πάντως τα άτομα με προδιαβήτη έχουν αυξημένο καρδιαγγειακό κίνδυνο (στεφανιαία νόσο, αγγειακό εγκεφαλικό επεισόδιο, αρτηριοπάθεια) και μεγαλύτερη πιθανότητα να πάσχουν από υπέρταση, δυσλιπιδαιμία και παχυσαρκία. Άτομα με προδιαβήτη έχουν «αντίσταση στην ινσουλίνη». Αυτό σημαίνει ότι η ινσουλίνη που παράγουν στο πάγκρεάς τους δεν επιτυγχάνει την αποτελεσματική ελάττωση του σακχάρου αίματος. Γι’ αυτό, τα άτομα με αντίσταση στην ινσουλίνη παράγουν όλο και περισσότερη, μέχρι να εξαντλήσουν το πάγκρεάς τους. Η αντίσταση στην ινσουλίνη προδιαθέτει σε καρδιαγγειακή νόσο [1][6][8].

Γενικά, προδιάθεση για διαβήτη τύπου 2 έχουν άτομα που είναι μεγαλύτερης ηλικίας, παχύσαρκα, που κάνουν καθιστική ζωή και δεν ασκούνται, που έχουν οικογενειακό ιστορικό διαβήτη, γυναίκες που γέννησαν νεογνά μεγαλύτερα από 4 κιλά, γυναίκες που έχουν πολυκυστικές ωοθήκες, και άτομα που έχουν αυξημένη χοληστερίνη ή/και τριγλυκερίδια ή αυξημένο λόγο HDL/LDL χοληστερόλης. Αυτά τα άτομα πρέπει να ελέγχουν το σάκχαρο νηστείας τους (συνήθως πριν από το πρωινό) τουλάχιστο μια φορά το χρόνο και αν το βρουν μεγαλύτερο από 100 mg/dl, πρέπει να εξεταστούν με καμπύλη γλυκόζης [1][6][8].

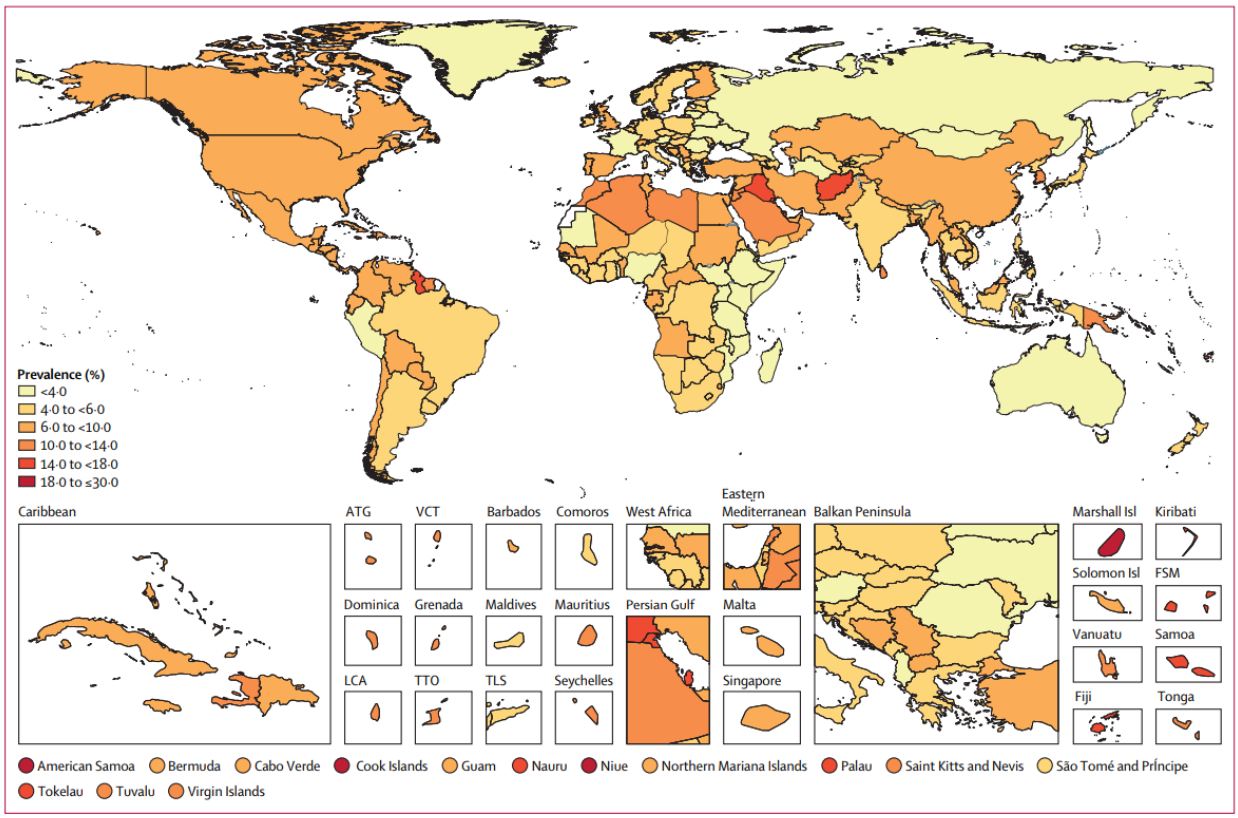
Μελέτες πρόληψης

Σύμφωνα με μια έγκυρη μελέτη από τη Φινλανδία, η απώλεια 5-7% του αρχικού βάρους με δίαιτα και άσκηση (150 λεπτά την εβδομάδα), μείωσε κατά 58% τον κίνδυνο εμφάνισης διαβήτη σε άτομα με παθολογική ανοχή στη γλυκόζη. Αυτή η μείωση ήταν η σημαντικότερη που βρέθηκε σε σοβαρές κλινικές μελέτες, και ήταν σαφώς πιο σημαντική από την μείωση που παρατηρήθηκε μετά από χρήση φαρμάκων (χωρίς αλλαγή τρόπου ζωής)[4] .

Δοκιμάστηκαν φάρμακα που βελτιώνουν την αντίσταση στην ινσουλίνη (η μετφορμίνη και η τρογλιταζόνη) ή που καθυστερούν την απορρόφηση των υδατανθράκων (η ακαρβόζη), η ορλιστάτη (φάρμακο κατά της παχυσαρκίας), ακόμα και αντιυπερτασικά (η ραμιπρίλη και η λοζαρτάνη) με διάφορα αποτελέσματα. Αναμένονται, πάντως, με μεγάλο ενδιαφέρον, αποτελέσματα μεγάλων μελετών που είναι σε εξέλιξη και χρησιμοποιούν άλλα φάρμακα [4][5].

Οι προσπάθειες για την πρόληψη του διαβήτη επικεντρώνονται στην εκπαίδευση του πληθυσμού, προκειμένου όλοι να αντιληφθούν τη σοβαρότητα του διαβήτη και να επιδιώξουν αλλαγή του τρόπου ζωής τους, με βάση συγκεκριμένες υγιεινοδιαιτητικές συμβουλές.

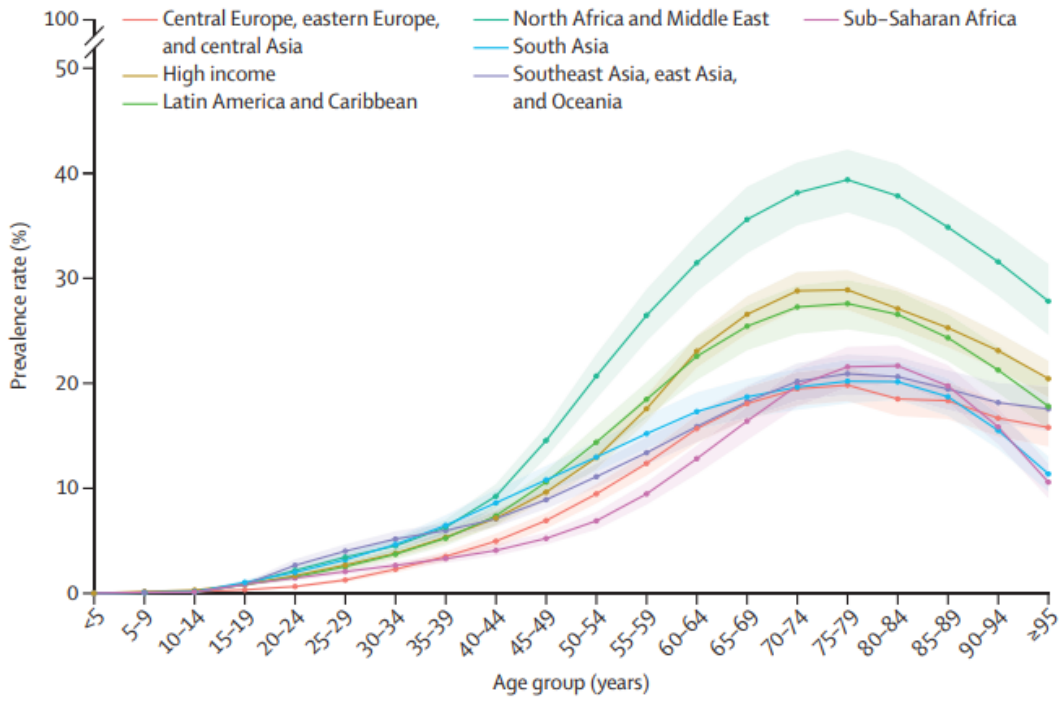
Με περισσότερα από μισό δισεκατομμύριο άτομα να αντιμετωπίζουν τον διαβήτη παγκοσμίως, αυτή η εκτενής υγειονομική ανησυχία ξεπερνά τα όρια, επηρεάζοντας άνδρες, γυναίκες και παιδιά όλων των ηλικιών σε κάθε χώρα. Οι προβλέψεις δείχνουν μια εκπληκτική άνοδο σε 1,3 δισεκατομμύρια ανθρώπους τα επόμενα 30 χρόνια, ενώ κάθε χώρα παρατηρεί αύξηση, όπως δημοσιεύτηκε πρόσφατα στο The Lancet [4][5][6].



Εικόνα 3:Συνολικά ποσοστά επικράτησης διαβήτη βάσει ηλικίας το 2021

Οι πιο πρόσφατοι και πλήρεις υπολογισμοί δείχνουν ότι η τρέχουσα παγκόσμια ποσοστιαία επιδημιολογία είναι στο 6,1%, καθιστώντας τον διαβήτη έναν από τους 10 κύριους λόγους θανάτου και αναπηρίας. Σε γεωγραφικό επίπεδο, ο υψηλότερος ρυθμός είναι 9,3% στη Βόρεια Αφρική και τη Μέση Ανατολή, με προβλεπόμενη αύξηση στο 16,8% έως το 2050. Στη Λατινική Αμερική και στην Καραϊβική, προβλέπεται αύξηση στο 11,3%.

Αυτή η ασθένεια είναι παρουσιάζεται σε άτομα ηλικίας 65 ετών και άνω, καταγράφοντας ποσοστό εμφάνισης άνω του 20% παγκοσμίως, με το υψηλότερο ποσοστό στην ηλικιακή ομάδα 75 έως 79 ετών. Αναλύοντας τα δεδομένα κατά γεωγραφική περιοχή, η Βόρεια Αφρική και η Μέση Ανατολή έχουν το υψηλότερο ποσοστό σε αυτήν την ηλικιακή ομάδα, φτάνοντας το 39,4%, ενώ η Κεντρική Ευρώπη, η Ανατολική Ευρώπη και η Κεντρική Ασία έχουν το χαμηλότερο ποσοστό στο 19,8% [4][5][6].



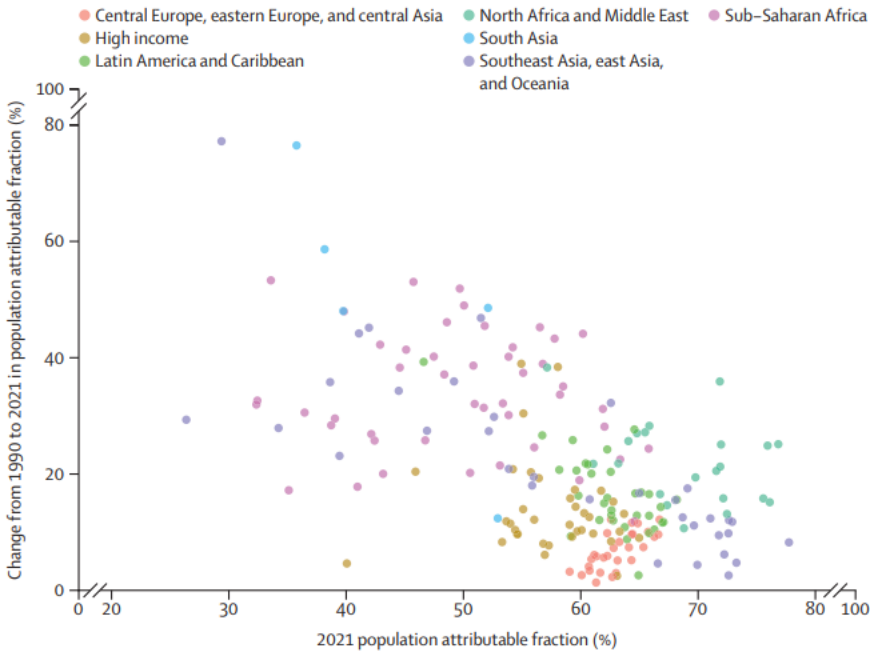
Εικόνα 4: Επικράτηση του διαβήτη ανά ηλικία και γεωγραφική περιοχή (2021)

Ο Διαβήτης τύπου 2 αποτελεί σημαντική πλειονότητα, αφού καλύπτει το 96% των παγκόσμιων περιστατικών. Όλοι οι 16 παράγοντες κινδύνου που μελετήθηκαν συσχετίζονται με το Διαβήτη τύπου 2. Ειδικότερα, ο υψηλός δείκτης μάζας σώματος (BMI) είναι ο κύριος κίνδυνος, συνεισφέροντας κατά 52,2% στην αναπηρία και τη θνησιμότητα από Διαβήτη τύπου 2. Άλλοι παράγοντες περιλαμβάνουν διατροφικούς κινδύνους, περιβαλλοντικούς/επαγγελματικούς κινδύνους, κάπνισμα, χαμηλή σωματική δραστηριότητα και χρήση αλκοόλ[4][5][7].

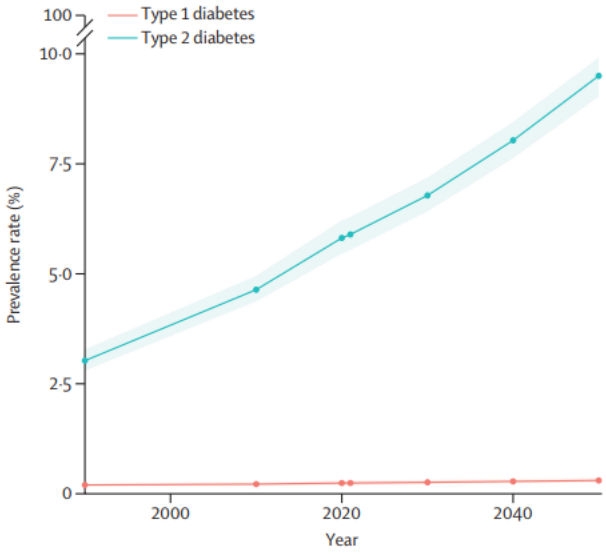
Ο ταχύς ρυθμός ανάπτυξης του διαβήτη, αποτελεί πρόκληση για τα παγκόσμια συστήματα υγείας. Ενώ συχνά συνδέεται με παχυσαρκία, έλλειψη άσκησης και κακή διατροφή, η πολυπλοκότητα της πρόληψης και ελέγχου του διαβήτη υπογραμμίζεται από γενετικούς, λογιστικούς, κοινωνικούς και οικονομικούς παράγοντες, ιδίως σε χώρες χαμηλού και μεσαίου εισοδήματος. Η νόσος αυξάνει επίσης τον κίνδυνο ισχαιμικής καρδιοπάθειας και εγκεφαλικού επεισοδίου.

Κάποιοι ερευνητές υποστηρίζουν μια ολιστική προσέγγιση, λαμβάνοντας υπόψη τις συνθήκες μέσα στις οποίες ζουν οι άνθρωποι, καθώς αυτές συμβάλλουν στις παγκόσμιες ανισότητες σε θέματα εξέτασης, πρόσβασης σε θεραπεία και διαθεσιμότητας υπηρεσιών υγείας. Αυτό είναι ακριβώς γιατί χρειαζόμαστε μια πιο πλήρη εικόνα του πώς ο διαβήτης επηρεάζει τους πληθυσμούς σε λεπτομερές επίπεδο [4][5][7].

Χρησιμοποιώντας τη μελέτη Global Burden of Disease (GBD) 2021, οι ερευνητές εξέτασαν την επιδημιολογία, τη νοσηρότητα και τη θνησιμότητα του διαβήτη σε 204 χώρες και εδάφη ανά ηλικία και φύλο από το 1990 έως το 2021 και προέβλεψαν την επιδημιολογία του διαβήτη έως το 2050. Παρείχαν επίσης εκτιμήσεις για τον διαβήτη τύπου 1 και τον διαβήτη τύπου 2, καθώς και το ποσοστό του βάρους του διαβήτη τύπου 2 που αποδίδεται σε 16 παράγοντες κινδύνου. Η ομάδα εργασίας περιελάβανε ερευνητές από το IHME και συνεργάτες του GBD 2021 από όλο τον κόσμο [4].



Εικόνα 5: Αλλαγή από το 1990 έως το 2021 στο κλάσμα που αποδίδεται στον πληθυσμό για υψηλό ΔΜΣ (BMI) σε σχέση με τον τύπο 2 διαβήτη, κατά υπερ-περιοχή GBD



Εικόνα 6: Παγκόσμια επικράτηση βάση ηλικίας του διαβήτη τύπου 1 και τύπου 2

Η σκιασμένη περιοχή αντιπροσωπεύει διαστήματα αβεβαιότητας 95%. Ο συνολικός διαβήτης είναι το άθροισμα του διαβήτη τύπου 1 και τύπου 2 [4].

2.2. Μηχανική Μάθηση (Machine Learning) & Οι Τεχνικές της

Η Μηχανική Μάθηση (ΜΜ) ορίζεται ως ένα πεδίο της τεχνητής νοημοσύνης (ΤΝ), όπου αλγόριθμοι, εκπαιδευμένοι σε τεράστια σύνολα δεδομένων, προωθούν τη δημιουργία αυτομάτων μοντέλων που επιτρέπουν στις μηχανές να εκτελούν εργασίες που θα ήταν δυνατές μόνο για ανθρώπους. Η παρούσα ανασκόπηση εξερευνά την ουσία της μηχανικής μάθησης, όχι μόνο για την εφαρμογή της στην πρόβλεψη του σακχαρώδους διαβήτη, αλλά και τον ευρύτερο χώρο της τεχνητής νοημοσύνης (AI), καθώς και μια λεπτομερή εξέταση των αλγορίθμων μηχανικής μάθησης [10].

Εικόνα 7: Παρομοίαση για εγκέφαλο υπολογιστή

2.2.1. Η Τύποι Μηχανικής Μάθησης

Η Μηχανική Μάθηση εξελίσσεται ως μια προοδευτική μεθοδολογία της τεχνητής νοημοσύνης, βασιζόμενη σε στατιστικά μοντέλα και αλγόριθμους για την ανάλυση και την εύρεση ενδιαφέρον προτύπων στα δεδομένα.

Οι Τύποι Μηχανικής Μάθησης είναι:

2.2.1.1. Supervised Learning: Οι αλγόριθμοι εκπαιδεύονται σε labeled δεδομένα, δηλαδή δεδομένα που έχουν ήδη ταξινομηθεί σε κλάση. Ο αλγόριθμος μαθαίνει να αναγνωρίζει πρότυπα στα δεδομένα και μπορεί να χρησιμοποιηθεί για την ταξινόμηση νέων, μη ταξινομημένων δεδομένων. Κάποιοι διάσημοι αλγόριθμοι για supervised learning είναι [10][11][13]:

Linear Regression

- Ένας εποπτευόμενος αλγόριθμος μηχανικής μάθησης που χρησιμοποιείται για εργασίες παλινδρόμησης. Μοντελοποιεί τη σχέση μεταξύ μιας εξαρτημένης μεταβλητής και μιας ή περισσότερων ανεξάρτητων μεταβλητών προσαρμόζοντας μια γραμμική εξίσωση στα δεδομένα.

Logistic Regression

- Ένας εποπτευόμενος αλγόριθμος μηχανικής μάθησης που χρησιμοποιείται για εργασίες ταξινόμησης. Μοντελοποιεί την πιθανότητα ενός δυαδικού αποτελέσματος με βάση μία ή περισσότερες ανεξάρτητες μεταβλητές χρησιμοποιώντας μια λογιστική συνάρτηση.

Decision Trees

- Ένας τύπος αλγόριθμου μηχανικής μάθησης που μπορεί να χρησιμοποιηθεί τόσο για εργασίες ταξινόμησης όσο και για εργασίες παλινδρόμησης. Λειτουργεί με αναδρομικό διαχωρισμό των δεδομένων σε μικρότερα υποσύνολα με βάση τις τιμές των χαρακτηριστικών και δημιουργώντας μια δομή δέντρου όπου κάθε εσωτερικός κόμβος αντιπροσωπεύει ένα χαρακτηριστικό, κάθε κλάδος αντιπροσωπεύει έναν κανόνα απόφασης και κάθε κόμβος φύλλου αντιπροσωπεύει μια πρόβλεψη.

Random Forests

- Μια μέθοδος μάθησης συνόλου που συνδυάζει πολλαπλά δέντρα αποφάσεων για να βελτιώσει την ακρίβεια και την ευρωστία των προβλέψεων. Λειτουργεί κατασκευάζοντας ένα πλήθος δέντρων απόφασης κατά τον χρόνο εκπαίδευσης και λαμβάνοντας την πλειοψηφία των εκροών των μεμονωμένων δέντρων.

Support Vector Machines

- Ένας εποπτευόμενος αλγόριθμος μηχανικής μάθησης που χρησιμοποιείται για εργασίες ταξινόμησης. Λειτουργεί βρίσκοντας το υπερεπίπεδο που διαχωρίζει τα δεδομένα σε διαφορετικές κλάσεις. Μπορεί να χρησιμοποιηθεί τόσο με γραμμικά όσο και με μη γραμμικά όρια απόφασης.

K - nearest neighbors

- Ένας μη παραμετρικός αλγόριθμος που κατηγοριποιεί ή προβλέπει με βάση την πλειοψηφική κλάση των k πλησιέστερων γειτόνων του στον χώρο των δεδομέων. Δεν προϋποθέτει κάποια συγκεκριμένη κατανομή δεδομένων.

Multi-Layer Perceptrons (MLPs)

- Είναι νευρωνικά δίκτυα με πολλαπλά κρυφά επίπεδα, που τους επιτρέπουν να μαθαίνουν πολύπλοκα μοτίβα στα δεδομένα. Χρησιμοποιούν συναρτήσεις ενεργοποίησης και είναι ζωτικής σημασίας για την επίλυση μη γραμμικών προβλημάτων στη μηχανική μάθηση.

2.2.1.2 Unsupervised Learning: Ο αλγόριθμος εκπαιδεύεται σε δεδομένα που δεν έχουν ταξινομηθεί. Ο αλγόριθμος μαθαίνει να αναγνωρίζει πρότυπα στα δεδομένα και μπορεί να χρησιμοποιηθεί για τον ομαδοποιημένο προσδιορισμό δεδομένων που είναι παρόμοια. Κάποιες τεχνικές unsupervised learning είναι [10][11][13]:

K-Means Clustering

- Πρόκειται για έναν αλγόριθμο μηχανικής μάθησης χωρίς επίβλεψη που χρησιμοποιείται για την ομαδοποίηση εργασιών. Λειτουργεί διαιρώντας τα δεδομένα σε k συμπλέγματα με βάση την ομοιότητα των σημείων δεδομένων. Ο αλγόριθμος εκχωρεί επαναληπτικά κάθε σημείο δεδομένων στο πλησιέστερο κέντρο συστάδας και ενημερώνει τα κεντροειδή μέχρι τη σύγκλιση.

Principal component analysis (PCA)

- Είναι ένας αλγόριθμος μηχανικής μάθησης χωρίς επίβλεψη που χρησιμοποιείται για τη μείωση διαστάσεων. Λειτουργεί βρίσκοντας τα πιο σημαντικά χαρακτηριστικά στα δεδομένα και προβάλλοντας τα δεδομένα σε χώρο χαμηλότερης διάστασης διατηρώντας παράλληλα τη διακύμανση.

Εκμάθηση κανόνων συσχέτισης

- Αποτελεί έναν αλγόριθμο μηχανικής μάθησης χωρίς επίβλεψη που χρησιμοποιείται για την εύρεση προτύπων σε δεδομένα. Λειτουργεί εντοπίζοντας συχνά σύνολα στοιχείων στα δεδομένα και δημιουργώντας κανόνες συσχέτισης που περιγράφουν τις σχέσεις μεταξύ των στοιχείων

2.2.1.3. Reinforcement Learning: Ο αλγόριθμος μαθαίνει διαδραστικά αλληλεπιδρώντας με ένα περιβάλλον και λαμβάνοντας ανατροφοδότηση υπό τη μορφή ανταμοιβών ή ποινών [10][11][13].

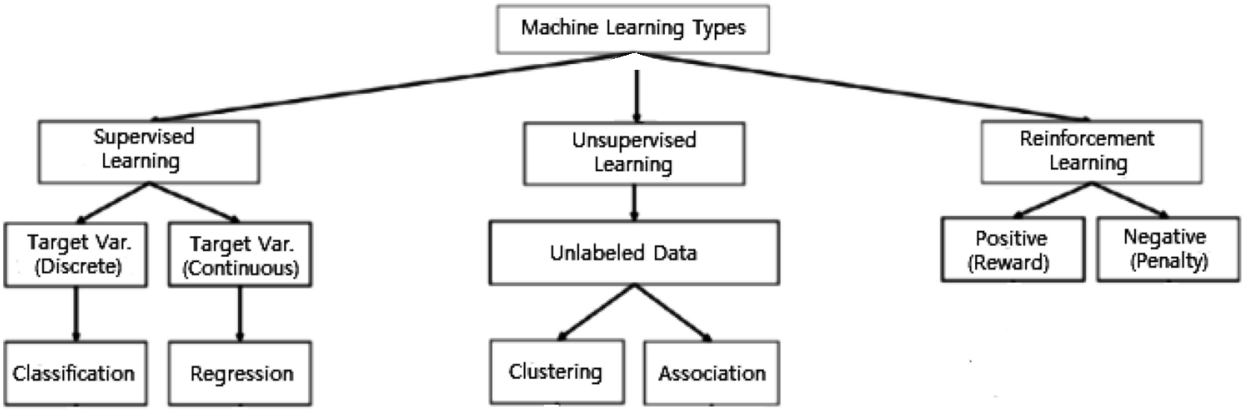
Q-Learning

- Χρησιμοποιείται για τη μάθηση μιας βέλτιστης στρατηγικής για έναν πράκτορα σε ένα περιβάλλον.

Deep reinforcement learning

- Χρησιμοποιεί νευρωνικά δίκτυα για την μάθηση βέλτιστων πολιτικών για έναν πράκτορα (agent).

Αυτά είναι μερικά παραδείγματα από την ποικιλία των αλγορίθμων και των εφαρμογών της ΜΜ. Η επιλογή ενός αλγορίθμου εξαρτάται από το συγκεκριμένο πρόβλημα και τα διαθέσιμα δεδομένα [11].



Εικόνα 8: Διάγραμμα κατηγοριών μηχανικής μάθησης

2.2.3. Ανάλυση αλγορίθμων Εποπτευόμενης Μηχανικής Μάθησης

Logistic Regression

Η LR είναι ένας δημοφιλής αλγόριθμος εποπτευόμενης μηχανικής μάθησης που χρησιμοποιείται για εργασίες δυαδικής ταξινόμησης, όπου ο στόχος είναι η πρόβλεψη της πιθανότητας εμφάνισης ενός συμβάντος με βάση μία ή περισσότερες μεταβλητές πρόβλεψης. Η συνεισφορά της χρήσης του γίνεται ευρέως σε διάφορους τομείς, όπως η υγειονομική περίθαλψη, τα οικονομικά και το μάρκετινγκ.[13][16]

Εννοιολογική επισκόπηση:

1. Σιγμοειδής συνάρτηση **:** Η LR χρησιμοποιεί τη σιγμοειδή συνάρτηση (επίσης γνωστή ως λογιστική συνάρτηση) για να αντιστοιχίσει τις προβλεπόμενες τιμές στις πιθανότητες. Η σιγμοειδής συνάρτηση διασφαλίζει ότι η έξοδος του μοντέλου θα είναι μεταξύ 0 και 1, το οποίο είναι κατάλληλο για την αναπαράσταση πιθανοτήτων.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Εξίσωση 1: Μαθηματικός τύπος για Λογιστική παλινδρόμηση

z: Αυτή είναι η είσοδος της σιγμοειδούς συνάρτησης. Είναι συνήθως ένας γραμμικός συνδυασμός χαρακτηριστικών εισόδου και των αντίστοιχων βαρών τους στην LR.

$$z = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_mx_m$$

w_0 : είναι ο όρος μεροληψίας ή παρεμπόδισης

$w_1, w_2, \ldots, w_m$: είναι τα coefficients των χαρακτηριστικών

$x_1 + x_2 + \ldots + x_m$: είναι τα χαρακτηριστικά που δίνονται ως είσοδο

σ(z): είναι η σιγμοειδή συνάρτηση

Οι παράμετροι $w_0, w_1, w_2, \ldots, w_m$ μαθαίνονται από τα δεδομένα εκπαίδευσης χρησιμοποιώντας τεχνικές βελτιστοποίησης όπως η gradient descent για την ελαχιστοποίηση της συνάρτησης απώλειας.

$1 + \mathrm{e}^{-z}$: Ο παρονομαστής της σιγμοειδούς συνάρτησης, ο οποίος διασφαλίζει ότι η έξοδος της σιγμοειδούς συνάρτησης είναι πάντα μεταξύ 0 και 1.

2. Όριο απόφασης: Το μοντέλο της LR χρησιμοποιεί ένα όριο απόφασης για την ταξινόμηση των instances σε διαφορετικές κλάσεις. Αυτό το όριο διαχωρίζει τις κλάσεις με βάση τις προβλεπόμενες πιθανότητες.

3. Εκτίμηση μέγιστης πιθανότητας: Η LR κάνει εκτίμηση τις παραμέτρους (coefficients) του μοντέλου μεγιστοποιώντας τη συνάρτηση πιθανότητας. Το μοντέλο μαθαίνει τις παραμέτρους που ταιριάζουν καλύτερα στα δεδομένα εκπαίδευσης.

Linear Regression

Η γραμμική παλινδρόμηση (Linear Regression) είναι ένας θεμελιώδης αλγόριθμος εποπτευόμενης μηχανικής μάθησης που χρησιμοποιείται για εργασίες παλινδρόμησης, όπου ο στόχος είναι να προβλέψει μια μεταβλητή συνεχούς αποτελέσματος με βάση μία ή περισσότερες μεταβλητές πρόβλεψης. Μοντελοποιεί τη σχέση μεταξύ μιας εξαρτημένης μεταβλητής και μιας ή περισσότερων ανεξάρτητων μεταβλητών προσαρμόζοντας μια γραμμική εξίσωση στα δεδομένα. [13][17]

Εννοιολογική επισκόπηση:

1. Γραμμικότητα: Η Linear Regression προϋποθέτει μια γραμμική σχέση μεταξύ των μεταβλητών πρόβλεψης και της μεταβλητής στόχου. Μοντελοποιεί αυτή τη σχέση προσαρμόζοντας μια ευθεία γραμμή στα δεδομένα.

2. Ελαχιστοποίηση σφαλμάτων: Η Linear Regression στοχεύει στην ελαχιστοποίηση της διαφοράς μεταξύ των παρατηρούμενων τιμών και των τιμών που προβλέπονται από το γραμμικό μοντέλο. Συνήθως χρησιμοποιεί μια συνάρτηση απώλειας όπως το μέσο τετραγωνικό σφάλμα (MSE) για να ποσοτικοποιήσει αυτή τη διαφορά.

3. Συντελεστές: Η Linear Regression κάνει εκτίμηση τις παραμέτρους (coefficients) για κάθε μεταβλητή πρόβλεψης, αντιπροσωπεύοντας την ισχύ και την κατεύθυνση της σχέσης μεταξύ των προγνωστικών παραγόντων και της μεταβλητής στόχου.

$$\hat{z} = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_m x_m$$

Εξίσωση 2: Μαθηματικός τύπος για γραμμική παλινδρόμηση

$\hat{z}$: είναι η τιμή πρόβλεψης

w_0 : είναι ο όρος μεροληψίας ή παρεμπόδισης

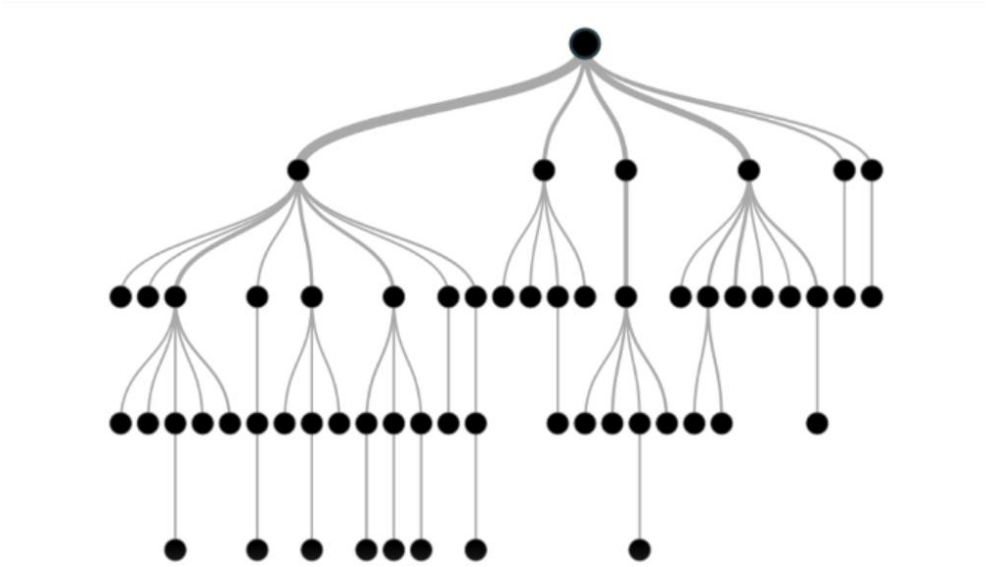
$w_1, w_2, \ldots, w_m$: είναι τα coefficients των χαρακτηριστικών

$x_1 + x_2 + \ldots + x_m$: είναι τα χαρακτηριστικά που δίνονται ως είσοδο

Οι παράμετροι $w_0, w_1, w_2, \ldots, w_m$ μαθαίνονται από τα δεδομένα εκπαίδευσης χρησιμοποιώντας τεχνικές βελτιστοποίησης όπως τα ordinary least squares (OLS) ή gradient descent για την ελαχιστοποίηση της συνάρτησης απώλειας.

Decision Trees

Τα δέντρα απόφασης (Decision Trees-DT) είναι ευέλικτοι και ευρέως χρησιμοποιούμενοι αλγόριθμοι εποπτευόμενης μάθησης, ικανοί να εκτελούν εργασίες ταξινόμησης και παλινδρόμησης.[18]



Σχήμα 1: Γραφική αναπαράσταση Decision Trees

Εννοιολογική επισκόπηση:

1. Δομή δέντρου: Ένα δέντρο αποφάσεων χωρίζει αναδρομικά τον χώρο χαρακτηριστικών σε περιοχές, με κάθε διαίρεση να βασίζεται σε έναν κανόνα απόφασης που εφαρμόζεται σε ένα από τα χαρακτηριστικά εισόδου.

2. Κριτήρια διαχωρισμού: Τα δέντρα αποφάσεων λαμβάνουν αποφάσεις σε κάθε κόμβο επιλέγοντας το χαρακτηριστικό που διαχωρίζει καλύτερα τα δεδομένα. Αυτός ο διαχωρισμός βασίζεται σε κριτήρια όπως το Gini impurity ή το κέρδος πληροφοριών (εντροπία), τα οποία στοχεύουν στη μεγιστοποίηση της ομοιογένειας (καθαρότητας) των υποσυνόλων που προκύπτουν.

3. Κόμβοι φύλλων: Οι τερματικοί κόμβοι του δέντρου αποφάσεων, που ονομάζονται κόμβοι φύλλων, αντιπροσωπεύουν τις τελικές προβλέψεις ή αποτελέσματα.

Ένα δέντρο απόφασης αποτελείται από κόμβους και κλάδους. Κάθε κόμβος αντιπροσωπεύει μια απόφαση που βασίζεται σε ένα χαρακτηριστικό και κάθε κλάδος αντιπροσωπεύει ένα αποτέλεσμα αυτής

της απόφασης. Η διαδικασία δημιουργίας ενός δέντρου αποφάσεων περιλαμβάνει τον αναδρομικό διαχωρισμό των δεδομένων με βάση τις τιμές των χαρακτηριστικών.

Ο αλγόριθμος του δέντρου απόφασης τυπικά διατυπώνεται χρησιμοποιώντας αναδρομική κατάτμηση. Σε κάθε κόμβο, ο αλγόριθμος επιλέγει το χαρακτηριστικό που διαχωρίζει καλύτερα τα δεδομένα, με βάση ένα κριτήριο διαχωρισμού. Για εργασίες δυαδικής ταξινόμησης, τα κοινά κριτήρια διαχωρισμού περιλαμβάνουν το Gini impurity ή το κέρδος πληροφοριών (εντροπία).

$$\text{Gini impurity} = \sum_{i=1}^{c} p_i^2$$

$$\text{Information Gain} = \text{entropy(parent)} - \sum_{i=1}^{k} \frac{N_i}{N} \text{entropy}(\text{child}_i)$$

Εξίσωση 3: Μαθηματικοί τύποι για τα κριτήρια διαχωρισμού στα Decision Trees

c : είναι ο αριθμός των κλάσεων

p_i: είναι η πιθανότητα της κλάσης i μέσα στον κόμβο

k: είναι ο αριθμός των παιδιών κόμβων μετά το διαμοιρασμό

N_i : είναι ο αριθμός δειγμάτων στο $i^{ου}$ παιδιού κόμβου

N: είναι ο συνολικός αριθμός δειγμάτων στον γονικό κόμβο.

Entropy: Η εντροπία είναι ένα μέτρο της αταξίας στον κόμβο.

Random Forest

Μια μέθοδος εκμάθησης συνόλου που συνδυάζει πολλαπλά δέντρα αποφάσεων για να βελτιώσει την ακρίβεια και την ευρωστία των προβλέψεων. Λειτουργεί κατασκευάζοντας ένα πλήθος δέντρων απόφασης κατά τη διάρκεια της εκπαίδευσης και εξάγει τον τρόπο (για ταξινόμηση) ή τη μέση πρόβλεψη (για παλινδρόμηση) από τα μεμονωμένα δένδρα [23].

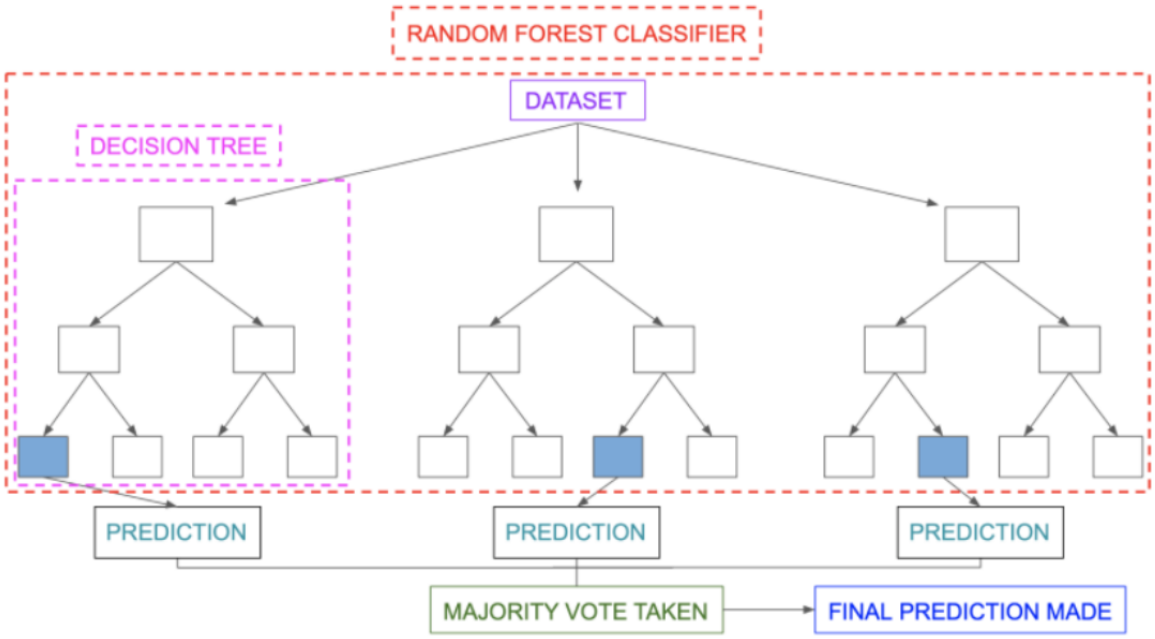
Εννοιολογική επισκόπηση:

1. Συλλογική μάθηση: Το Random Forest (RF) είναι μια τεχνική εκμάθησης συνόλου που συνδυάζει τις προβλέψεις πολλαπλών μεμονωμένων δέντρων αποφάσεων για τη βελτίωση της συνολικής απόδοσης και ευρωστίας του μοντέλου.

2. Αθροιστική εκκίνηση(Bootstrap Aggregating)-Bagging: Κάθε δέντρο απόφασης στο RF εκπαιδεύεται σε ένα τυχαίο υποσύνολο των δεδομένων εκπαίδευσης, δειγματοληψία με αντικατάσταση. Αυτή η διαδικασία είναι γνωστή ως bootstrapping. Το Bagging βοηθά στη μείωση της υπερπροσαρμογής προάγοντας την ποικιλομορφία μεταξύ των δέντρων.

3. Τυχαία επιλογή χαρακτηριστικών: Σε κάθε διαχωρισμό στο RF, μόνο ένα τυχαίο υποσύνολο χαρακτηριστικών λαμβάνεται υπόψη για διαχωρισμό. Αυτό προάγει περαιτέρω την ποικιλομορφία μεταξύ των δέντρων και εμποδίζει τα μεμονωμένα δέντρα να κυριαρχήσουν στο σύνολο.

4. Κατηγοριοποίηση ή παλινδρόμηση: Στις εργασίες ταξινόμησης, η τελική πρόβλεψη του τυχαίου δάσους καθορίζεται με πλειοψηφία μεταξύ των μεμονωμένων δέντρων. Στις εργασίες παλινδρόμησης, η τελική πρόβλεψη είναι η μέση πρόβλεψη όλων των δέντρων.



Εικόνα 9: Απεικόνιση διαδικασίας πρόβλεψης για το RF

Το RF συνδυάζει τις προβλέψεις πολλαπλών δέντρων αποφάσεων $T_1, T_2, \dots T_n$ προκειμένου να πραγματοποιήσει την τελική του πρόβλεψη.

Για την ταξινόμηση (classification) λαμβάνεται ο τρόπος των προβλέψεων, ενώ για εργασίες παλινδρόμησης λαμβάνεται υπόψη ο μέσος όρος των προβλέψεων.

Classification: $\hat{y} = \text{mode}\,(T_1(x), T_2(x), \dots, T_n(x))$

Regression: $\hat{y} = \frac{1}{n}\sum_{i=1}^{n} T_i\,(x)$

Εξίσωση 4: Μαθηματικοί τύπου του RF ανάλογα την εργασία Classification ή Regression

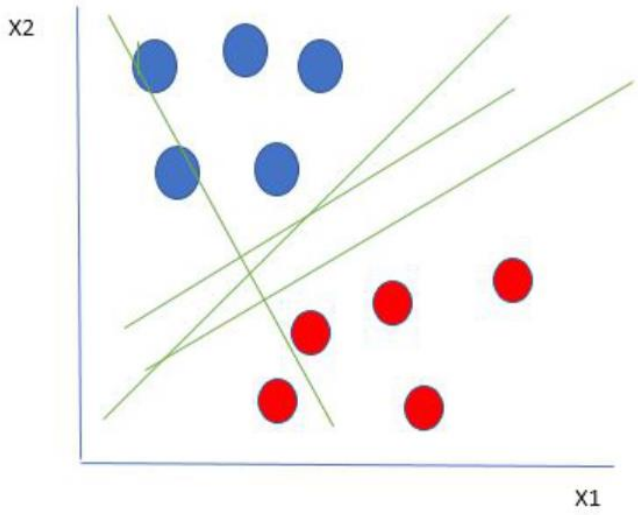
$\hat{y}$: Είναι η προβλεπόμενη μεταβλητή

$T_i(x)$: είναι η πρόβλεψη του $i^{ου}$ δένδρου απόφασης για είσοδο μεταβλητής χ

n: είναι ο αριθμός των δένδρων αποφάσεων που χρησιμοποιεί ο αλγόριθμος RF

Support Vector Machines (SVM)

Το Support Vector Machine (SVM) είναι ένας εποπτευόμενος αλγόριθμος μηχανικής μάθησης που χρησιμοποιείται κυρίως για ταξινόμηση (Classification), αν και μπορεί να προσαρμοστεί για εργασίες παλινδρόμησης (Regression). Στοχεύει στην εύρεση του βέλτιστου υπερεπίπεδου σε ένα χώρο N-διαστάσεων που διαχωρίζει αποτελεσματικά σημεία δεδομένων που ανήκουν σε διαφορετικές κλάσεις. Το υπερεπίπεδο είναι τοποθετημένο έτσι ώστε να μεγιστοποιεί το περιθώριο μεταξύ των πλησιέστερων σημείων των διαφορετικών κλάσεων. Η διάσταση του υπερεπίπεδου αντιστοιχεί στον αριθμό των χαρακτηριστικών εισόδου, με μια γραμμή για δύο χαρακτηριστικά, ένα δισδιάστατο επίπεδο για τρία χαρακτηριστικά και γίνεται πιο περίπλοκη για μεγαλύτερες διαστάσεις. Στην περίπτωση δύο χαρακτηριστικών εισόδου (x1, x2) και μιας δυαδικής εξαρτημένης μεταβλητής (μπλε κύκλος ή κόκκινος κύκλος), μπορούν να σχεδιαστούν πολλαπλές γραμμές για να διαχωριστούν τα σημεία δεδομένων, που αντιπροσωπεύουν πιθανά υπερεπίπεδα.



Εικόνα 10: Απεικόνιση γραφικής διαχωρισιμότητας στο SVM

Η πρόκληση βρίσκεται στον προσδιορισμό του καλύτερου υπερεπιπέδου για την ακριβή ταξινόμηση των σημείων δεδομένων.[19]

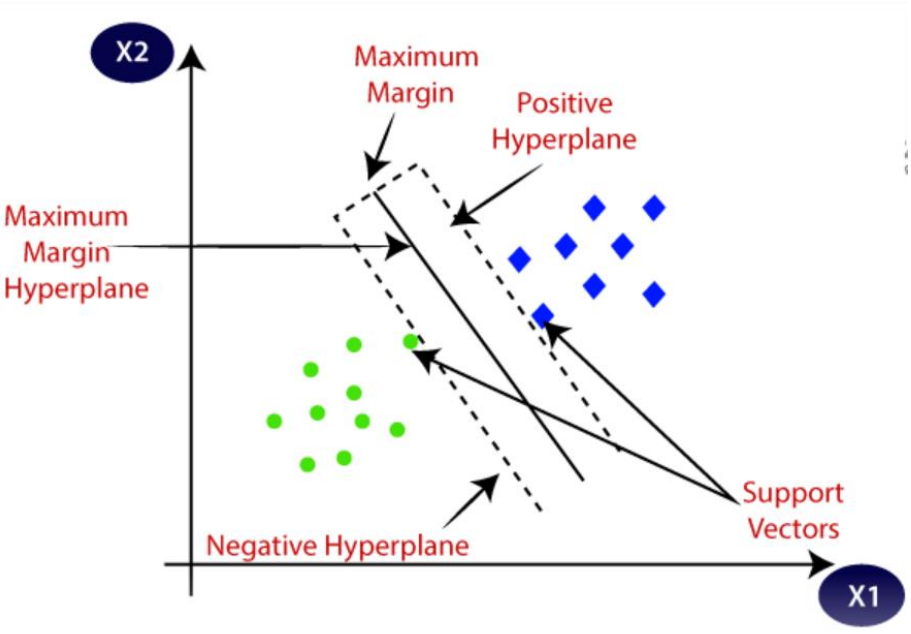
Εννοιολογική επισκόπηση:

1. Γραμμική Διαχωρισιμότητα: Το SVM βασίζεται στην ιδέα της εύρεσης του υπερεπίπεδου που διαχωρίζει καλύτερα τα δεδομένα σε διαφορετικές κλάσεις, ενώ μεγιστοποιεί το περιθώριο, που είναι η απόσταση μεταξύ του υπερεπίπεδου και των πλησιέστερων σημείων δεδομένων από κάθε κατηγορία.

2. Πυρήνας: Το SVM μπορεί να χειριστεί αποτελεσματικά μη γραμμικά διαχωρίσιμα δεδομένα αντιστοιχίζοντας τα χαρακτηριστικά εισόδου σε έναν χώρο υψηλότερης διάστασης χρησιμοποιώντας μια συνάρτηση πυρήνα. Σε αυτόν τον χώρο υψηλότερης διάστασης, τα δεδομένα μπορεί να γίνουν γραμμικά διαχωρίσιμα, επιτρέποντας στο SVM να βρει ένα διαχωριστικό υπερεπίπεδο.

3. Διανύσματα υποστήριξης: Τα διανύσματα υποστήριξης είναι τα σημεία δεδομένων που βρίσκονται πιο κοντά στο υπερεπίπεδο. Παίζουν καθοριστικό ρόλο στον προσδιορισμό του βέλτιστου υπερεπίπεδου και του περιθωρίου.

4. Περιθώριο: Το περιθώριο είναι η απόσταση μεταξύ του υπερεπίπεδου και των πλησιέστερων σημείων δεδομένων από κάθε κλάση. Το SVM στοχεύει να μεγιστοποιήσει αυτό το περιθώριο, καθώς πιστεύεται ότι οδηγεί σε καλύτερη απόδοση γενίκευσης.



Εικόνα 11: Απεικόνιση των κρίσιμων όρων του SVM

Τύποι SVM

Γραμμικό SVM: Χρησιμοποιεί μια ευθεία γραμμή ή υπερεπίπεδο για να διαχωρίσει τα σημεία δεδομένων διαφορετικών κλάσεων όταν τα δεδομένα μπορούν να διαχωριστούν με ακρίβεια γραμμικά.

Μη Γραμμικό SVM: Χειρίζεται δεδομένα που δεν μπορούν να διαχωριστούν με ευθεία γραμμή. Χρησιμοποιεί συναρτήσεις πυρήνα για να μετατρέψει δεδομένα σε χώρο υψηλότερης διάστασης όπου είναι δυνατός ο γραμμικός διαχωρισμός.

Δημοφιλείς συναρτήσεις πυρήνα

Γραμμικός πυρήνας (Linear SVM): Αντιπροσωπεύει ένα απλό σύνολο κουκίδων διανυσμάτων, κατάλληλο για γραμμικά διαχωρίσιμα δεδομένα.

Πολυωνυμικός πυρήνας (Polynomial Kernel): Αντιστοιχίζει δεδομένα σε χώρο υψηλότερης διάστασης χρησιμοποιώντας πολυωνυμικές συναρτήσεις, επιτρέποντας πιο πολύπλοκα όρια απόφασης.

Πυρήνας Radial Bass Function (RBF): Μετασχηματίζει δεδομένα με βάση την απόσταση από ένα σημείο αναφοράς, επιτρέποντας στο SVM να χειρίζεται μη γραμμικά δεδομένα.

Sigmoid πυρήνας: Εφαρμόζει μια υπερβολική εφαπτομενική συνάρτηση στο γινόμενο κουκίδων των διανυσμάτων. Είναι κατάλληλη για εργασίες δυαδικής ταξινόμησης.

Αυτές οι συναρτήσεις πυρήνα βοηθούν τα SVM να χειρίζονται διάφορους τύπους δεδομένων αντιστοιχίζοντας τα σε χώρους υψηλότερων διαστάσεων όπου είναι δυνατός ο γραμμικός διαχωρισμός, ακόμη και όταν τα αρχικά δεδομένα δεν είναι γραμμικά διαχωρισμένα.

Σε ένα σύνολο δεδομένων εκπαίδευσης $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ όπου το x_i είναι τα χαρακτηριστικά εισόδου και y_iαντιπροσωπεύει τις ετικέτες κλάσεων (-1 ή 1 για δυαδική ταξινόμηση). Το SVM στοχεύει στο να βρει το βέλτιστο υπερεπίπεδο που αντιπροσωπεύεται από $w^T x + b = 0$ όπου w είναι το διάνυσμα βάρους και b είναι ο όρος μεροληψίας.

Η συνάρτηση SVM για τη γραμμικά διαχωρίσιμη περίπτωση :

$$min_{w,b} \frac{1}{2}\|w\|^2 \text{ με } y_i(w^T x_i + b) \geq 1 \text{ για όλα τα } i$$

Εξίσωση 5: Μαθηματικός τύπος για SVM γραμμικά διαχωρίσιμη περίπτωση

Η συνάρτηση SVM για τη μη γραμμικά διαχωρίσιμη περίπτωση περιλαμβάνει και μια slack μεταβλητή :

$$min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_i^n \xi_i$$

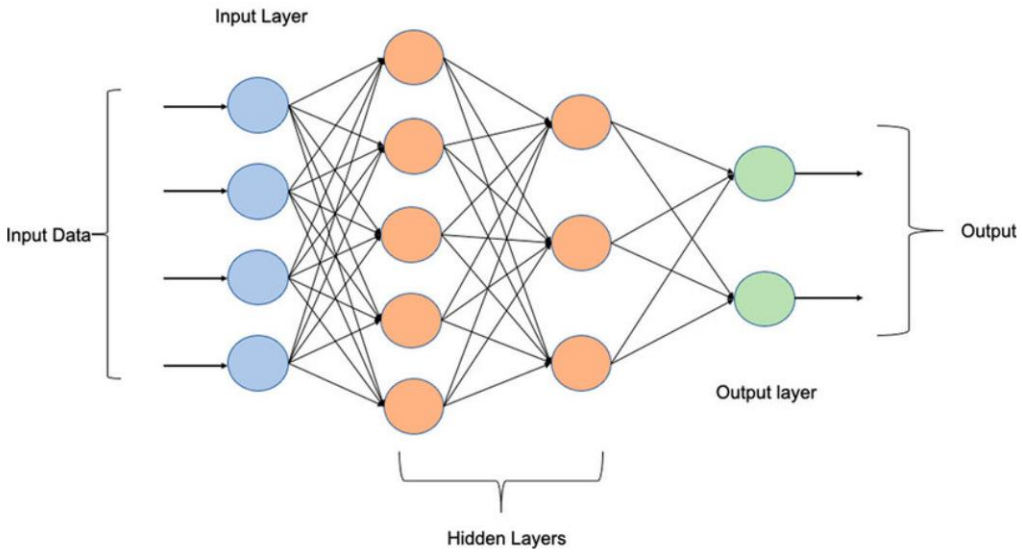
$$\text{με } y_i(w^T x_i + b) \geq 1 - \xi_i \text{ για όλα τα } i$$

Εξίσωση 6: Μαθηματικός τύπος για SVM μη γραμμικά διαχωρίσιμη περίπτωση

όπου C είναι η παράμετρος τακτοποίησης που ελέγχει την αντιστάθμιση μεταξύ της μεγιστοποίησης του περιθωρίου και της ελαχιστοποίησης του σφάλματος ταξινόμησης.[20]

Multi-Layer Perceptrons (MLPs)

Τα Multi-Layer Perceptrons (MLPs) είναι ένας τύπος τεχνητού νευρωνικού δικτύου που χρησιμοποιείται συνήθως για εργασίες ταξινόμησης (Classification) και παλινδρόμησης (Regression) και υπάγεται στην κατηγορία των αλγορίθμων της εποπτευόμενης μάθησης.[21]



Εικόνα 12: Βασική αρχιτεκτονική Multi-layer Perceptrons

Εννοιολογική επισκόπηση:

1. Δομή Νευρωνικού Δικτύου: Τα MLP αποτελούνται από πολλαπλά στρώματα διασυνδεδεμένων κόμβων (νευρώνες), συμπεριλαμβανομένου ενός στρώματος εισόδου, ενός ή περισσότερων κρυφών επιπέδων και ενός στρώματος εξόδου. Κάθε νευρώνας σε ένα στρώμα συνδέεται με κάθε νευρώνα στα γειτονικά στρώματα.

2. Συνάρτηση ενεργοποίησης (Activation Function): Μη γραμμικές συναρτήσεις ενεργοποίησης (όπως ReLU, σιγμοειδές ή tanh) εφαρμόζονται στην έξοδο κάθε νευρώνα στα κρυφά επίπεδα. Αυτές οι συναρτήσεις ενεργοποίησης εισάγουν μη γραμμικότητα στο μοντέλο, επιτρέποντάς του να μαθαίνει πολύπλοκα μοτίβα στα δεδομένα.

3. Προώθηση διάδοσης: Κατά τη φάση της προς τα εμπρός διάδοσης, τα δεδομένα εισόδου διαβιβάζονται μέσω του δικτύου και οι υπολογισμοί εκτελούνται στρώμα προς στρώμα έως ότου φτάσει στο επίπεδο εξόδου. Κάθε νευρώνας υπολογίζει ένα σταθμισμένο άθροισμα των εισόδων του, εφαρμόζει μια συνάρτηση ενεργοποίησης και περνά το αποτέλεσμα στο επόμενο στρώμα.

4. Backpropagation: Η Backpropagation είναι ένας αλγόριθμος που χρησιμοποιείται για την εκπαίδευση των MLP. Περιλαμβάνει επαναληπτική προσαρμογή των βαρών των συνδέσεων στο δίκτυο για να ελαχιστοποιηθεί η διαφορά μεταξύ των προβλεπόμενων εξόδων και των πραγματικών ετικετών. Αυτό γίνεται συνήθως χρησιμοποιώντας τεχνικές βελτιστοποίησης Gradient Descent.

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)} \text{ όπου } a^{(l)} = \sigma^{(l)}(z^{(l)})$$

Εξίσωση 7: Μαθηματικός τύπος για MLPs

$z^{(l)}$: είναι το σταθμισμένο άθροισμα των inputs για το επίπεδο l

$a^{(l)}$: είναι το εξαγόμενο από το στρώμα l μετά την εφαρμογή της συνάρτησης ενεργοποίησης

$W^{(l)}$: είναι το βάρος του στρώματος l

$b^{(l)}$: είναι το διάνυσμα μεροληψίας του στρώματος l

$\sigma^{(l)}$: είναι η συνάρτηση ενεργοποίησης

K-Nearest Neighbors (KNN)

Ο K-Nearest Neighbors (KNN) είναι ένας απλός αλλά ισχυρός αλγόριθμος που χρησιμοποιείται στη μηχανική μάθηση για εργασίες ταξινόμησης και παλινδρόμησης. Ανήκει στην κατηγορία των αλγορίθμων της μη εποπτευόμενης μηχανικής μάθησης. Θεωρείται ένας μη παραμετρικός και τεμπέλης αλγόριθμος μάθησης επειδή δεν κάνει υποθέσεις σχετικά με την υποκείμενη κατανομή δεδομένων και δεν μαθαίνει ένα συγκεκριμένο μοντέλο κατά τη διάρκεια της εκπαίδευσης. Αντίθετα, αποθηκεύει όλα τα διαθέσιμα σημεία δεδομένων και κάνει προβλέψεις με βάση το μέτρο ομοιότητας μεταξύ των νέων δεδομένων και των υπαρχόντων. [22]

Εννοιολογική επισκόπηση:

1.Εκμάθηση: Ο KNN είναι ένας αλγόριθμος που βασίζεται σε παραδείγματα, δεν μαθαίνει ρητά ένα μοντέλο. Αντίθετα, αποθηκεύει όλα τα διαθέσιμα στιγμιότυπα-δεδομένα και κάνει προβλέψεις με βάση την ομοιότητά τους με τα νέα στιγμιότυπα-δεδομένα.

2. Κ-Κοντινότεροι γείτονες: Για κάθε νέο στιγμιότυπο, ο KNN προσδιορίζει το k πλησιέστεροι γείτονες από το σύνολο δεδομένων εκπαίδευσης με βάση την απόσταση (συνήθως Ευκλείδεια απόσταση). Η

κατηγορία ή η τιμή του νέου στιγμιότυπου προσδιορίζεται στη συνέχεια με πλειοψηφία (για ταξινόμηση) ή με μέσο όρο (για παλινδρόμηση) μεταξύ των k πλησιέστερων γειτόνων.

3. Μέτρηση απόστασης: Ο KNN βασίζεται σε μια μέτρηση απόστασης για τη μέτρηση της ομοιότητας μεταξύ των στιγμιοτύπων. Η επιλογή του τύπου απόστασης (π.χ. Ευκλείδεια, Μανχάταν ή συνημιτονική απόσταση) μπορεί να επηρεάσει την απόδοση του αλγορίθμου.

4. Υπερπαράμετρος k: Ο αριθμός των γειτόνων k, πρόκειται για μια υπερπαράμετρο η οποία ορίζεται πριν εφαρμοστεί ο αλγόριθμος. Δηλώνει το πλήθος των γειτόνων που πρέπει να ελεγχθούν ώστε το σημείο να ταξινομηθεί σε κλάση.

$$\text{distance}(\mathrm{x}, \mathrm{x_i}) = \sqrt{\sum_{\mathrm{j}=1}^{\mathrm{m}} (\mathrm{x_{j-x_{i_j}}})^2}$$

Εξίσωση 8: Μαθηματικός τύπος για τον KNN

x: νέο στιγμιότυπο

$\mathrm{x_i}$: στιγμιότυπο που υπάρχει μέσα στο σύνολο εκπαίδευσης

x_j: είναι το $\mathrm{j}^{\sigma\tau o}$ χαρακτηριστικό του x στιγμιότυπου

$\mathrm{x_{i_j}}$: είναι το $j^{\sigma\tau o}$ χαρακτηριστικό του στιγμιότυπου $\mathrm{x_i}$

2.2.3. Εφαρμογές της Μηχανικής Μάθησης:

Η Μηχανική Μάθηση εφαρμόζεται σε διάφορους τομείς, όπως η υγεία, οι οικονομικές υπηρεσίες, ο τομέας των μεταφορών, η επεξεργασία φυσικής γλώσσας, η αναγνώριση εικόνας και η αναγνώριση ομιλίας [10].

2.2.4. Προκλήσεις της Μηχανικής Μάθησης

Παρά τη δυναμική της, η Μηχανική Μάθηση αντιμετωπίζει προκλήσεις, όπως η ανάγκη για υψηλής ποιότητας δεδομένα, η ανάγκη για ειδικούς επαγγελματίες και ηθικοί περιορισμοί σχετικά με πιθανές προκαταλήψεις και οι επιπτώσεις στην απασχόληση [10].

2.2.5. Η Μηχανική Μάθηση στην Πρόβλεψη του Διαβήτη

Η μηχανική μάθηση αποτελεί ισχυρό εργαλείο για την πρόβλεψη της έναρξης του σακχαρώδους διαβήτη. Οι αλγόριθμοι αναλύουν εκτεταμένα σύνολα δεδομένων ασθενών, λαμβάνοντας υπόψη διάφορους παράγοντες για τη δημιουργία εξατομικευμένων προφίλ κινδύνου [24].

2.2.5.1 Αλγόριθμοι για την Πρόβλεψη του Διαβήτη

Καθοριστικοί αλγόριθμοι περιλαμβάνουν τον αλγόριθμο Random Forest, χρησιμοποιώντας δέντρα απόφασης για ακριβείς προβλέψεις, και τον αλγόριθμο Support Vector Machine, που αξιοποιείται στην κατηγοριοποίηση δεδομένων [24].

2.2.6. Εφαρμογές της Μηχανικής Μάθησης στον τομέα της Υγείας

Οι εφαρμογές της μηχανικής μάθησης στον τομέα της υγείας υπερβαίνουν την πρόβλεψη του σακχαρώδους διαβήτη, περιλαμβάνοντας τον προγνωστικό χαρακτήρα των νόσων, την εξατομίκευση της ιατρικής, και τη βελτιστοποίηση της φροντίδας του ασθενούς.

2.3. Τεχνικές Εξόρυξης Δεδομένων

2.3.1. Ανασκόπηση και Εφαρμογές

Η εξόρυξη δεδομένων (data mining) αποτελεί ένα πολύπλευρο εργαλείο που λειτουργεί ως γέφυρα μεταξύ των δεδομένων και των πρακτικών αποφάσεων. Μέσω της εξόρυξης μοτίβων, πληροφοριών και γνώσεων από μεγάλα σύνολα δεδομένων, οι εταιρίες και οργανισμοί λαμβάνουν ενημερωμένες αποφάσεις σε διάφορους τομείς. Τα κύρια χαρακτηριστικά της ανάλυσης δεδομένων, είναι η δυαδική κατηγοριοποίηση (binary classification), η κατηγοριοποίηση (classification) και η παλινδρόμηση (regression) [25],[27].

Η εξόρυξη δεδομένων ξεκινά με τη συλλογή δεδομένων από διάφορες πηγές. Αυτά τα δεδομένα μπορεί να είναι δομημένα (π.χ. βάσεις δεδομένων, φύλλα εργασίας) ή μη δομημένα (π.χ. κείμενο, εικόνες). Η πλούσια και ποικιλόμορφη φύση των δεδομένων που συλλέγονται αποτελεί τη βάση για την επόμενη ανάλυση.

Πριν από την ανάλυση, η προεπεξεργασία (preprocessing) των δεδομένων είναι απαραίτητη για να εξασφαλιστεί η ποιότητα και η αξιοπιστία των δεδομένων [25][26].

1. Καθαρισμός (Cleaning) : περιλαμβάνει την αφαίρεση ανομοιογενειών, τιμών που λείπουν και θορύβου από τα δεδομένα, εξασφαλίζοντας ότι τα δεδομένα είναι απαλλαγμένα από ανωμαλίες που θα μπορούσαν να αλλοιώσουν τα αποτελέσματα.

2. Μετατροπή (Transformation): περιλαμβάνει τη μετατροπή ακατέργαστων δεδομένων σε κατάλληλη μορφή για ανάλυση.

3. Ενσωμάτωση (Integration): περιλαμβάνει τη συνένωση δεδομένων από διαφορετικές πηγές, εξασφαλίζοντας ότι όλες οι σχετικές πληροφορίες συγκεντρώνονται για ανάλυση.

4. Μείωση διαστάσεων (Reduction): Τεχνικές μείωσης των διαστάσεων των δεδομένων όπως η Ανάλυση Κυρίων Συνιστωσών (PCA) ή μέθοδοι επιλογής χαρακτηριστικών χρησιμοποιούνται για να διευκολύνουν την ανάλυση.

2.3.2. Τεχνικές Εξόρυξης Δεδομένων

2.3.2.1. Δυαδική Κατηγοριοποίηση

Στη δυαδική κατηγοριοποίηση, ο στόχος είναι να κατηγοριοποιηθούν τα δεδομένα σε μια από τις δύο προκαθορισμένα κλάσεις. Τεχνικές όπως η λογιστική παλινδρόμηση, οι μηχανές υποστήριξης διανυσμάτων (SVM) και οι δέντρα απόφασης χρησιμοποιούνται συχνά για αυτόν τον σκοπό.

Αυτή η τεχνική βρίσκει εκτεταμένες εφαρμογές σε διάφορους τομείς, που κυμαίνονται από το πεδίο των χρηματοοικονομικών έως την υγειονομική περίθαλψη και όχι μόνο. Στα χρηματοοικονομικά, η δυαδική κατηγοριοποίηση χρησιμοποιείται συχνά για τη βαθμολόγηση της πιστοληπτικής ικανότητας, όπου τα άτομα ταξινομούνται είτε ως αξιόπιστα είτε ως μη αξιόπιστα με βάση το οικονομικό τους ιστορικό και άλλους σχετικούς παράγοντες. Στην υγειονομική περίθαλψη, η δυαδική ταξινόμηση παίζει κρίσιμο ρόλο στην ιατρική διάγνωση, όπως η διάκριση μεταξύ καλοήθων και κακοήθων όγκων σε ακτινολογικές εικόνες. Επιπλέον, η δυαδική ταξινόμηση χρησιμοποιείται ευρέως στον εντοπισμό ανεπιθύμητων μηνυμάτων ηλεκτρονικού ταχυδρομείου, όπου τα μηνύματα ηλεκτρονικού ταχυδρομείου ταξινομούνται είτε ως ανεπιθύμητα είτε ως όχι με βάση το περιεχόμενο και τα χαρακτηριστικά τους [27][28][29].

2.3.2.2. Κατηγοριοποίηση

Πέραν της δυαδικής κατηγοριοποίησης, η κατηγοριοποίηση (classification) περιλαμβάνει την ανάθεση προκαθορισμένων ετικετών σε στιγμιότυπα δεδομένων ανάμεσα σε πολλαπλές κλάσεις. Αλγόριθμοι όπως ο k-nearest neighbors (KNN), η random forest και η naive Bayes χρησιμοποιούνται ευρέως σε τέτοιες εργασίες.

Αυτή η τεχνική βρίσκει εφαρμογές σε διάφορα πεδία όπως η αναγνώριση εικόνας, η επεξεργασία φυσικής γλώσσας και η ανάλυση συναισθημάτων. Στην αναγνώριση εικόνων, οι αλγόριθμοι κατηγοριοποίησης χρησιμοποιούνται για την κατηγοριοποίηση αντικειμένων ή σκηνών που απεικονίζονται σε εικόνες, επιτρέποντας εφαρμογές όπως αυτόνομα οχήματα και συστήματα αναγνώρισης προσώπου. Στην επεξεργασία φυσικής γλώσσας, η κατηγοριοποίηση χρησιμοποιείται για την ανάλυση συναισθήματος, προσδιορίζοντας το συναίσθημα (θετικό, αρνητικό ή ουδέτερο) που

εκφράζεται σε δεδομένα κειμένου, το οποίο είναι πολύτιμο για εργασίες όπως η παρακολούθηση μέσων κοινωνικής δικτύωσης και η ανάλυση σχολίων πελατών. Επιπλέον, οι αλγόριθμοι κατηγοριοποίησης εφαρμόζονται στην ταξινόμηση εγγράφων, όπου τα έγγραφα κατηγοριοποιούνται σε προκαθορισμένα θέματα ή κατηγορίες, βοηθώντας στην ανάκτηση πληροφοριών και σε εργασίες οργάνωσης [27][28][29].

2.3.2.3. Παλινδρόμηση

Η ανάλυση παλινδρόμησης περιλαμβάνει την πρόβλεψη αριθμητικών τιμών με βάση τα χαρακτηριστικά των δεδομένων. Τεχνικές όπως η γραμμική παλινδρόμηση, η πολυωνυμική παλινδρόμηση και η παλινδρόμηση gradient boosting χρησιμοποιούνται για την πρόβλεψη των σχέσεων μεταξύ των μεταβλητών.

Στα χρηματοοικονομικά, η ανάλυση παλινδρόμησης χρησιμοποιείται για την πρόβλεψη των τιμών των μετοχών, όπου ιστορικά δεδομένα μετοχών και άλλες σχετικές μεταβλητές χρησιμοποιούνται για την πρόβλεψη μελλοντικών τιμών μετοχών, βοηθώντας τους επενδυτές στη λήψη αποφάσεων. Στα οικονομικά, η ανάλυση παλινδρόμησης διαδραματίζει κρίσιμο ρόλο στην πρόβλεψη ζήτησης, όπου αναλύονται οικονομικοί παράγοντες και τάσεις της αγοράς για να προβλεφθεί η μελλοντική ζήτηση για προϊόντα και υπηρεσίες, βοηθώντας τις επιχειρήσεις στην κατανομή πόρων και τον προγραμματισμό. Επιπλέον, η ανάλυση παλινδρόμησης χρησιμοποιείται στην υγειονομική περίθαλψη για την πρόβλεψη της έκβασης των ασθενών, όπου τα δεδομένα του ασθενούς και το ιατρικό ιστορικό αξιοποιούνται για την πρόβλεψη των αποτελεσμάτων και των κινδύνων για την υγεία, επιτρέποντας εξατομικευμένα σχέδια θεραπείας και παρεμβάσεις [27][28][29].

2.3.3. Κατασκευή και Αξιολόγηση Μοντέλων

Η επιλογή ενός κατάλληλου μοντέλου εξαρτάται από τη φύση του προβλήματος που αντιμετωπίζεται και τα χαρακτηριστικά του συνόλου δεδομένων. Τα μοντέλα εκπαιδεύονται χρησιμοποιώντας δεδομένα με ετικέτες, και η απόδοσή τους αξιολογείται χρησιμοποιώντας μετρικές όπως η ορθότητα (accuracy), ακρίβεια (precision) η ανάκληση (recall) και F1-score.

Η εξόρυξη δεδομένων διευκολύνει στην εξαγωγή νοήματος από τα μοντέλα, αποκαλύπτοντας κρυμμένες ενδείξεις μέσα στα δεδομένα. Η ερμηνεία των αποτελεσμάτων είναι κρίσιμη για την παραγωγή ενεργειών και την κατανόηση των επιπτώσεών τους στην πραγματική ζωή. Η γνώση που προέρχεται από την ανάλυση δεδομένων εφαρμόζεται σε διάφορους τομείς, συμπεριλαμβανομένων των επιχειρηματικών αποφάσεων, της υγείας, της χρηματοοικονομικής και λοιπών.

Η αποτελεσματική διαχείριση μαζικών συνόλων δεδομένων δημιουργεί σημαντικές προκλήσεις σε θέματα αποθήκευσης, επεξεργασίας και ανάλυσης. Η διασφάλιση του απορρήτου και της ασφάλειας των δεδομένων είναι ζωτικής σημασίας, ειδικά όταν ασχολούμαστε με ευαίσθητες πληροφορίες..

Τέλος, η εξόρυξη δεδομένων είναι ένα ισχυρό εργαλείο για την εξαγωγή συμπερασμάτων από μεγάλα σύνολα δεδομένων, επιτρέποντας σε εταιρίες και οργανισμούς να λαμβάνουν βέλτιστες αποφάσεις και να προωθούν την καινοτομία. Μέσω της αξιοποίησης μιας ποικιλίας τεχνικών από τη δυαδική κατηγοριοποίησης μέχρι την ανάλυση παλινδρόμησης, η εξόρυξη δεδομένων αποκαλύπτει κρυφά patterns και γνώσεις, μετατρέποντας τελικά τα ακατέργαστα δεδομένα σε λήψη αποφάσεων [26][27].

2.4. Evaluation Metrics στην Μηχανική Μάθηση

Οι μετρήσεις αξιολόγησης είναι ζωτικής σημασίας για την αξιολόγηση της αποτελεσματικότητας και της απόδοσης των μοντέλων μηχανικής μάθησης. Παρέχουν μετρήσιμους υπολογισμούς ορθότητας (accuracy) και ακρίβειας (precision), ανάκλησης (Recall) και άλλων σχετικών παραμέτρων, προσφέροντας γνώσεις σχετικά με την ικανότητα ενός μοντέλου να γενικεύει σε μη ορατά δεδομένα και να εκπληρώνει τους επιθυμητούς στόχους [30][31].

2.4.1. Μετρήσεις ταξινόμησης (Classification Metrics)

2.4.1.1. Ορθότητα (Accuracy)

Η ορθότητα ταξινόμησης υπολογίζει την αναλογία των σωστών προβλέψεων προς τον συνολικό αριθμό των δειγμάτων εισόδου. Ενώ είναι απλό και διαισθητικό, μπορεί να παραπλανήσει παρουσία μη ισορροπημένων τάξεων, καθώς δεν λαμβάνει υπόψη τα ψευδώς θετικά και τα ψευδώς αρνητικά.

2.4.1.2. Ακρίβεια (Precision)

Η ακρίβεια μετρά το ποσοστό των αληθινών θετικών προβλέψεων από όλες τις θετικές προβλέψεις που γίνονται από το μοντέλο. Υπολογίζεται ως ο λόγος των αληθινών θετικών προς το άθροισμα των αληθινών θετικών και των ψευδώς θετικών. Η ακρίβεια (precision) είναι σημαντικής σημασίας όταν η ελαχιστοποίηση των ψευδών θετικών είναι ζωτικής σημασίας.

2.4.1.3. Ανάκληση (Recall)

Η ανάκληση, γνωστή και ως ευαισθησία ή αληθινός θετικός ρυθμός, ποσοτικοποιεί την αναλογία των πραγματικών θετικών που προσδιορίζονται σωστά από το μοντέλο. Υπολογίζεται ως ο λόγος των αληθινών θετικών προς το άθροισμα των αληθινών θετικών και των ψευδών αρνητικών. Η ανάκληση είναι ιδιαίτερα σημαντική όταν η ελαχιστοποίηση των ψευδώς αρνητικών είναι κρίσιμη.

2.4.1.4. F1 Score

Το σκορ F1 είναι το αρμονικό μέσο ακρίβειας και ανάκλησης. Παρέχει μια ενιαία μέτρηση που εξισορροπεί τόσο την ακρίβεια όσο και την ανάκληση. Η βαθμολογία F1 φτάνει την καλύτερη τιμή της στο 1 (τέλεια ακρίβεια και ανάκληση) και τη χειρότερη στο 0. Είναι μια χρήσιμη μέτρηση όταν υπάρχει ανομοιόμορφη κατανομή κατηγορίας ή όταν τα ψευδώς θετικά και τα ψευδώς αρνητικά είναι εξίσου σημαντικά.

2.4.1.5. Λογαριθμική απώλεια (Log Loss)

Η λογαριθμική απώλεια «τιμωρεί» τις ψευδείς ταξινομήσεις και είναι ιδιαίτερα χρήσιμη για προβλήματα ταξινόμησης πολλαπλών κλάσεων. Η ελαχιστοποίηση της Log loss οδηγεί σε υψηλότερη ακρίβεια για τον ταξινομητή, καθώς μετρά την αβεβαιότητα των προβλεπόμενων πιθανοτήτων σε σύγκριση με τις πραγματικές ετικέτες κλάσης.

2.4.1.6. Περιοχή κάτω από την καμπύλη (AUC)

Η AUC χρησιμοποιείται ευρέως για εργασίες δυαδικής ταξινόμησης και αντιπροσωπεύει την πιθανότητα ένας ταξινομητής να κατατάξει ένα τυχαία επιλεγμένο θετικό στιγμιότυπο υψηλότερα από ένα αρνητικό στιγμιότυπο. Λαμβάνει υπόψη διάφορες πτυχές απόδοσης όπως το πραγματικό θετικό ποσοστό, το πραγματικό αρνητικό ποσοστό και το ποσοστό ψευδώς θετικών, παρέχοντας μια ολοκληρωμένη αξιολόγηση της διακριτικής ισχύος του μοντέλου.

2.4.2. Μετρήσεις παλινδρόμησης

2.4.2.1. Μέσο απόλυτο σφάλμα (MAE)

Το MAE μετρά τη μέση απόλυτη διαφορά μεταξύ των προβλεπόμενων και των πραγματικών τιμών. Παρέχει μια απλή ερμηνεία του μέσου μεγέθους των σφαλμάτων που γίνονται από το μοντέλο.

2.4.2.2. Μέσο τετραγωνικό σφάλμα (MSE)

Το MSE υπολογίζει τη μέση τετραγωνική διαφορά μεταξύ των προβλεπόμενων και των πραγματικών τιμών. Ενισχύει τον αντίκτυπο των μεγάλων σφαλμάτων λόγω της λειτουργίας τετραγωνισμού, καθιστώντας το πιο ευαίσθητο σε ακραίες τιμές σε σύγκριση με το MAE.

2.4.2.3. Μέσο τετράγωνο σφάλμα ρίζας (RMSE)

Το RMSE είναι η τετραγωνική ρίζα του MSE και παρέχει μια πιο ερμηνεύσιμη μέτρηση, καθώς βρίσκεται στις ίδιες μονάδες με το μεταβλητό στόχο. Προσφέρει ένα μέτρο της τυπικής απόκλισης των προβλέψεων από τις πραγματικές τιμές.

2.4.2.4. Βαθμολογία R2 (Συντελεστής Προσδιορισμού)

Η βαθμολογία R2 υποδεικνύει την αναλογία διακύμανσης στην εξαρτημένη μεταβλητή που εξηγείται από τις ανεξάρτητες μεταβλητές. Κυμαίνεται από 0 έως 1, όπου το 1 υποδηλώνει τέλεια εφαρμογή και το 0 δείχνει ότι το μοντέλο δεν εξηγεί καμία διακύμανση.

2.5. Επισκόπηση της Τεχνητής Νοημοσύνης

Η τεχνητή νοημοσύνη αναγνωρίζει την ικανότητα του υπολογιστικού συστήματος να εκτελεί εργασίες που απαιτούν ευφυΐα, όπως τη μάθηση, τη σκέψη, την επίλυση προβλημάτων, και τη χρήση γλώσσας.

Η τεχνητή νοημοσύνη έχει εφαρμογές σε πολλούς τομείς, όπως η αναγνώριση ομιλίας, η αναγνώριση εικόνας, η επεξεργασία φυσικής γλώσσας και η αυτόνομη οδήγηση. Ενώ η τεχνητή νοημοσύνη φέρνει τεράστια οφέλη, όπως την αυτονομία οχημάτων και τη βελτίωση των συστημάτων αναγνώρισης, αντιμετωπίζει προκλήσεις όπως η διαφάνεια, η ασφάλεια και η ηθική.

Με αυτήν τη συνολική ανασκόπηση, αποκτούμε μια εκτενή κατανόηση της Μηχανικής Μάθησης και των τεχνικών της, καλύπτοντας την ουσία, τις εφαρμογές, την Τεχνητή Νοημοσύνη και τους διάφορους τύπους αλγορίθμων που τη διαμορφώνουν.

Κεφάλαιο 3 : Μελέτη σχετικών εργασιών

Σε αυτό το κεφάλαιο θα παρουσιάσουμε το πρόβλημα του διαβήτη με πιο ειδικούς όρους και ταυτόχρονα θα παρουσιαστούν σχετικές εργασίες που έχουν πραγματοποιηθεί για την πρόβλεψη του με βάση τη μηχανική μάθηση. Έχουν χρησιμοποιηθεί διάφορα σύνολα δεδομένων καθώς και το σύνολο δεδομένων PIMA που είναι και το σύνολο δεδομένων που χρησιμοποιήθηκε για την εκπόνηση αυτής της εργασίας.

3.1. Σχετικές εργασίες πρόβλεψης διαβήτη με Μηχανική Μάθηση

Η μηχανική μάθηση (Machine Learning) στις μέρες μας έχει αναδειχθεί ως η μετασχηματιστική δύναμη σε διάφορους επιστημονικούς τομείς της ανθρωπότητας, προσφέροντας καινοτόμες λύσεις σε διάφορα περίπλοκα προβλήματα. Η συνεισφορά της στον τομέα της υγείας είναι εντυπωσιακή, καθώς δύναται να γίνει πρόβλεψη ή και ακόμα διάγνωση σε διάφορες σπάνιες ασθένειες. Μια από τις πολλές εφαρμογές της είναι η πρόβλεψη της ασθένειας του διαβήτη. Ο διαβήτης, μια χρόνια μεταβολική διαταραχή που χαρακτηρίζεται από αυξημένα επίπεδα γλυκόζης στο αίμα, αποτελεί σημαντική παγκόσμια πρόκληση για την υγεία. Οι τεχνικές της Μηχανικής Μάθησης έχουν αποδειχθεί καθοριστικές για την αξιοποίηση των τεράστιων συνόλων δεδομένων για τη διάκριση μοτίβων, την εξαγωγή πολύτιμων γνώσεων και την πρόβλεψη της πιθανότητας ανάπτυξης διαβήτη. Υπάρχουν πάρα πολλές έρευνες από την ακαδημαϊκή κοινότητα που προσπαθούν να εφαρμόσουν διάφορους αλγορίθμους Μηχανικής Μάθησης και να αποτυπώσουν τα μέγιστα δυνατά αποτελέσματα.

Ενδεικτικά υπάρχει το paper “ Diabetes Prediction Using Machine Learning ” της KM Jyoti Rani [32], το οποίο στοχεύει στην πρόβλεψη ασθενών με διαβήτη χρησιμοποιώντας τεχνικές μηχανικής μάθησης. Τα δεδομένα που χρησιμοποιήθηκαν είναι από το νοσοκομείο της Φρανκφούρτης της Γερμανίας. Στο εν λόγω papper η ερευνήτρια συγκρίνει διάφορους αλγόριθμους ταξινόμησης, συμπεριλαμβανομένων του Support Vector Machine (SVM),του k-Nearest Neighbour (KNN), του Random Forest, της Logistic Regression και των Decision Trees. Καταλήγοντας στο συμπέρασμα πως το καλύτερο μοντέλο είναι το Decision Tree επιτυγχάνοντας μέγιστο Accuracy 99%.

Επιπλέον σημαντικό paper αποτελεί το “ Classification and prediction of diabetes disease using machine learning paradigm ” των Md. Maniruzzaman, Md. Jahanur Rahman, Benojir Ahammed, και Md. Menhazul

Abedin [33]. Στο εν λόγω paper οι ερευνητές χρησιμοποιούν λογιστική παλινδρόμηση (Logistic Regression) για τον εντοπισμό των παραγόντων κινδύνου του διαβήτη με βάση τις τιμές p και τις αναλογίες πιθανοτήτων. Γίνεται χρήση τεσσάρων αλγορίθμων, του Naive Bayes, του δέντρου αποφάσεων (Decision Trees), του Adaboost (AB) και του Random Forest, οι οποίοι υιοθετούνται για πρόβλεψη. Το σύνολο δεδομένων που χρησιμοποιούν προέρχεται από την Εθνική Έρευνα Εξέτασης Υγείας και Διατροφής (2009–2012) των Ηνωμένων πολιτειών Αμερικής. Καταλήγοντας πως ο συνδυασμός Logistic Regression με επιλεγμένα τα πιο σημαντικά χαρακτηριστικά με το Random Forest επιτυγχάνουν την καλύτερη δυνατή ακρίβεια 94.25%.

Τέλος ξεχωρίζει το paper “Machine Learning and Data Mining Methods in Diabetes Research” των Ioannis Kavakiotis, Olga Tsave, Athanasios Salifoglou, Nicos Maglaveras, Ioannis Vlahavas, και Ioanna Chouvarda [34], στο οποίο γίνεται διερεύνηση των εφαρμογών της μηχανικής μάθησης καθώς και των τεχνικών εξόρυξης γνώσης για το πεδίο της πρόβλεψης διαβήτη. Οι συγγραφείς αναλύουν ένα ευρύ φάσμα αλγορίθμων μηχανικής μάθησης, που έχουν χρησιμοποιηθεί σε διάφορες προσεγγίσεις, διαπιστώνοντας πως το 85% των αλγορίθμων που χρησιμοποιούνται υπάγεται στις προσεγγίσεις της εποπτευόμενης μάθησης, ενώ το 15% να βασίζεται σε μη εποπτευόμενη μάθηση. Τέλος καταλήγουν πως το Support Vector Machines (SVM) αναδεικνύεται ως ο πιο επιτυχημένος και ευρέως χρησιμοποιούμενος αλγόριθμος στα δεδομένα για πρόβλεψη διαβήτη.

3.2. Σχετικές εργασίες για το σύνολο δεδομένων PIMA

Όπως γίνεται κατανοητό, υπάρχουν πολλά σύνολα δεδομένων που χρησιμοποιούνται για την πρόβλεψη του διαβήτη με τεχνικές ML. Το πιο διάσημο σύνολο δεδομένων στο οποίο έχουν στηριχθεί πάρα πολλές έρευνες είναι το PIMA. Το συγκεκριμένο σύνολο δεδομένων περιέχει διάφορα χαρακτηριστικά που σχετίζονται με την υγεία γυναικών της Ινδίας Pima ηλικίας 21 ετών και άνω, και έχει συλλεχθεί από το Εθνικό Ινστιτούτο Διαβήτη και Πεπτικών και Νεφροπαθειών των Ηνωμένων πολιτειών Αμερικής. Παρακάτω θα δούμε κάποια ενδεικτικά papers που συνέβαλαν στην πρόβλεψη του διαβήτη με μηχανική μάθηση.

Αρχικά το paper “Prediction of Diabetes using Classification Algorithms” των Deepti Sisodiaa και Dilip Singh Sisodiab [35], χρησιμοποιεί το σύνολο δεδομένων PIMA και αξιολογεί τρείς αλγορίθμους μηχανικής μάθησης. Οι αλγόριθμοι αυτοί είναι το Decision Tree, το Support Vector Machine(SVM) καθώς και ο Naive

Bayes. Συγκεκριμένα ο Naive Bayes πέτυχε την υψηλότερη ακρίβεια 76,30% ξεπερνώντας τις επιδόσεις των άλλων μοντέλων.

Στην συνέχεια θα μελετήσουμε το paper “A comparison of machine learning algorithms for diabetes prediction” των Jobeda Jamal Khanam και Simon Y. Foo [36]. Οι συγγραφείς διερευνούν την πρόβλεψη του διαβήτη χρησιμοποιώντας τρεις αλγόριθμους μηχανικής μάθησης: Support Vector Machine (SVM), Naive Bayes και Random Forest στο σύνολο δεδομένων PIMA. Οι ερευνητές αξιολογούν την επίδοση των αλγορίθμων και καταλήγουν πως τόσο το SVM όσο και το Random Forest πέτυχαν ακρίβεια πάνω από 80% .

Τέλος θα αναλύσουμε και την επιστημονική μελέτη που δημοσιεύτηκε στο Journal of Diabetes & Metabolic Disorders των ερευνητών Huma Naz και Sachin Ahuja με τίτλο “ Deep learning approach for diabetes prediction using PIMA Indian dataset ” [37]. Σε αυτή τη μελέτη οι μελετητές συνέκριναν την απόδοση ποικίλων αλγόριθμων μηχανικής μάθησης χρησιμοποιώντας το σύνολο δεδομένων PIMA.Οι αλγόριθμοι που εφαρμόστηκαν είναι: το Artificial Neural Network (ANN), ο Naïve Bayes (NB), τα Decision Trees (DT) και τεχνικές Deep Learning (DL). Τα αποτελέσματα έδειξαν ότι η ακρίβεια αυτών των παραπάνω αλγορίθμων κυμαινόταν από 90% έως 98%. Τέλος οι ερευνητές καταλήγουν πως χρησιμοποιώντας τεχνικές DL η πρόβλεψη εμφάνισης διαβήτη είχε εντυπωσιακή ακρίβεια 98,07%.

Συνοψίζοντας θα πρέπει να σημειώσουμε πως στην βιβλιογραφία οι περισσότερες επιστημονικές έρευνες έχουν στηριχθεί στο σύνολο δεδομένων PIMA, για την αξιολόγηση των διαφόρων ταξινομητών, με απώτερο σκοπό τα καλύτερα evaluation metrics. Στην παρούσα εργασία επιλέχθηκε το ίδιο σύνολο δεδομένων.

Κεφάλαιο 4 :Περιγραφή εργαλείου

Σε αυτό το κεφάλαιο θα παρουσιαστεί το περιβάλλον υλοποίησης, η γλώσσα που επιλέχθηκε καθώς και οι βιβλιοθήκες που χρησιμοποιήθηκαν. Τέλος θα παρουσιαστούν συνοπτικά τα βήματα της μεθοδολογίας που θα ακολουθήσουμε για την υλοποίηση.

4.1. Περιβάλλον υλοποίησης

Για την δημιουργία της εργασίας επιλέχθηκε η γλώσσα προγραμματισμού Python που είναι η πιο διαδεδομένη γλώσσα στις μέρες μας, για πολλούς επαγγελματίες και ερευνητές για την υλοποίηση εφαρμογών της μηχανικής μάθησης (ML). Υπερέχει λόγω της απλότητας, ευελιξίας και της διαθεσιμότητας πολύ ισχυρών βιβλιοθηκών.

Η Python είναι μια υψηλού επιπέδου και διερμηνευμένη (interpreted) γλώσσα προγραμματισμού, γνωστή για την ευελιξία της σε διάφορους τομείς, όπως η μηχανική μάθηση (ML), η τεχνητή νοημοσύνη (AI), η ανάπτυξη ιστού και η ανάλυση δεδομένων. Είναι δωρεάν και ανοικτού κώδικα Αναπτύχθηκε από τον Guido van Rossum τη δεκαετία του 1980, η Python δίνει προτεραιότητα στην αναγνωσιμότητα και στη μείωση της πολυπλοκότητας του κώδικα. Η πρώτη έκδοση έκανε το ντεμπούτο το 1991, προσφέροντας βασικές λειτουργίες και τύπους δεδομένων. Καθώς η Python κέρδισε την έλξη, οι επόμενες εκδόσεις εισήγαγαν πρόσθετα χαρακτηριστικά για την κάλυψη των εξελισσόμενων αναγκών. Η Python 1.5 (1997) εισήγαγε βελτιώσεις, ενώ η Python 2.0 (2000) και η Python 3.0 (2008) έφεραν σημαντικές ενημερώσεις και επεκτάθηκαν λειτουργίες. Καθ' όλη τη διάρκεια της εξέλιξής της, η Python έχει συγκεντρώσει μια μεγάλη κοινότητα προγραμματιστών αφιερωμένη στη συνεχή βελτίωση και συντήρησή της. Σήμερα είναι διαθέσιμη η έκδοση 3.12.2 και υποστήριξη υπάρχει από την έκδοση 3.8.x και μετά. [38]

Χρησιμοποιήθηκε το περιβάλλον Anacoda (anaconda individual edition 2023.09-64bits), το οποίο είναι ένα περιβάλλον ελεύθερης διανομής και βρίσκεται διαθέσιμο δωρεάν στην επίσημη ιστοσελίδα του Anacoda και συγκεκριμένα στο σύνδεσμο https://www.anaconda.com/download . Επιπλέον η έκδοση της python που χρησιμοποιήθηκε είναι η 3.11.5. Το περιβάλλον Anacoda περιλαμβάνει ένα ολοκληρωμένο

σύστημα διαχείρισης πακέτων και ένα ευρύ φάσμα βιβλιοθηκών και εργαλείων που χρησιμοποιούνται συνήθως για εργασίες επιστήμης δεδομένων και μηχανικής μάθησης.

Τέλος τα αρχεία με κώδικα δημιουργήθηκαν στο jupyter, το οποίο είναι μια διαδικτυακή εφαρμογή ανοιχτού κώδικα που χρησιμοποιείται για τη δημιουργία και κοινή χρήση εγγράφων που ονομάζονται σημειωματάρια (notebooks), τα οποία συνδυάζουν ζωντανό κώδικα, οπτικοποιήσεις, κείμενο και εξισώσεις. Υποστηρίζει πολλές γλώσσες προγραμματισμού, συμπεριλαμβανομένων των Python, R και Julia, επιτρέποντας διαδραστικό υπολογισμό, αυτό πρακτικά σημαίνει πως οι χρήστες μπορούν να εκτελέσουν κελιά κώδικα μεμονωμένα και να δουν τα αποτελέσματα στο ίδιο έγγραφο. Τα σημειωματάρια Jupyter χρησιμοποιούνται ευρέως στην ανάλυση δεδομένων, στους επιστημονικούς υπολογιστές, στη μηχανική μάθηση και στην εκπαίδευση. Το Jupyter είναι μια διαδικτυακή εφαρμογή ελεύθερης διανομής και βρίσκεται διαθέσιμο δωρεάν στην επίσημη ιστοσελίδα του Jupyter και συγκεκριμένα στο σύνδεσμο https://jupyter.org/.

4.1.1. Βιβλιοθήκες Python

Οι βιβλιοθήκες της Python που χρησιμοποιήθηκαν για τις ανάγκες της υλοποίησης είναι ανοικτού κώδικα και αυτές και είναι διαθέσιμες δωρεάν. Θα τις αναλύσουμε σύντομα παρακάτω:

- Pandas

Η Pandas είναι μια ισχυρή βιβλιοθήκη για χειρισμό και ανάλυση δεδομένων στην Python. Περιλαμβάνει δομές δεδομένων όπως Series και DataFrame, οι οποίες είναι ιδανικές για το χειρισμό δομημένων δεδομένων. Το Pandas επιτρέπει τον καθαρισμό δεδομένων, τον χειρισμό, τη συγχώνευση, τον διαχωρισμό και πολλά άλλα. Χρησιμοποιείται ευρέως στην επιστήμη δεδομένων και στην μηχανική μάθηση. [40]

- NumPy

Η NumPy είναι το βασικό πακέτο για αριθμητικούς υπολογισμούς με Python. Παρέχει υποστήριξη για πολυδιάστατους πίνακες, μαζί με μια συλλογή μαθηματικών συναρτήσεων για αποτελεσματικές πράξεις σε αυτούς τους πίνακες. Το NumPy χρησιμοποιείται ευρέως για αριθμητικούς υπολογισμούς,

συμπεριλαμβανομένης της γραμμικής άλγεβρας, των μετασχηματισμών Fourier, της δημιουργίας τυχαίων αριθμών κ.λπ. [40]

- Matplotlib

Η Matplotlib είναι μια ολοκληρωμένη βιβλιοθήκη για τη δημιουργία στατικών, κινούμενων και διαδραστικών απεικονίσεων στην Python. Προσφέρει μια μεγάλη ποικιλία γραφικών παραστάσεων και επιλογών προσαρμογής, επιτρέποντας στους χρήστες να δημιουργούν διαγράμματα ποιότητας που μπορεί να γίνουν ως και δημοσίευση για επιστημονική έρευνα ακόμα και για παρουσιάσεις. Το Matplotlib μπορεί να δημιουργήσει γραφήματα όπως γραμμικά διαγράμματα, διαγράμματα διασποράς, ιστογράμματα, γραφήματα ράβδων και πολλά άλλα. [39]

- SciPy

Η SciPy είναι μια συλλογή επιστημονικών υπολογιστικών εργαλείων που έχουν δημιουργηθεί στηριζόμενη στη βιβλιοθήκη NumPy. Περιλαμβάνει εργαλεία για βελτιστοποίηση, ολοκλήρωση, παρεμβολή, γραμμική άλγεβρα, επεξεργασία σήματος και εικόνας, στατιστικές συναρτήσεις και πολλά άλλα. Το SciPy επεκτείνει τη λειτουργικότητα του NumPy και χρησιμοποιείται εκτενώς σε επιστημονικές και μηχανολογικές εφαρμογές.[40]

- Scikit-learn

Το Scikit-learn είναι μια βιβλιοθήκη μηχανικής μάθησης στην Python που παρέχει απλά και αποτελεσματικά εργαλεία για εξόρυξη δεδομένων και ανάλυση δεδομένων. Διαθέτει όλους τους σημαντικούς εποπτευόμενους και μη εποπτευόμενους αλγόριθμους μάθησης, που χρησιμοποιούνται γιαταξινόμηση, παλινδρόμηση, ομαδοποίηση, μείωση διαστάσεων και τέλος και τα ίδια τα μοντέλα. Το Scikit-learn έχει σχεδιαστεί για να είναι εύκολο στη χρήση και να ενσωματώνεται με άλλες επιστημονικές και αριθμητικές βιβλιοθήκες όπως οι NumPy και SciPy. [39]

- Missingno

Η Missingno είναι μια βιβλιοθήκη Python που χρησιμοποιείται για την οπτικοποίηση δεδομένων που λείπουν σε σύνολα δεδομένων. Παρέχει διαγράμματα από τα οποία μπορεί εύκολα να προκύψει

πληροφορία, όπως ραβδώσεις, θερμικούς χάρτες και δενδρογράμματα για να βοηθήσει τους χρήστες να κατανοήσουν τα μοτίβα των τιμών που λείπουν στα δεδομένα τους. Η Missingno είναι χρήσιμο για την προ επεξεργασία δεδομένων και την αξιολόγηση της ποιότητας, καθιστώντας δυνατό τον εντοπισμό και την αποτελεσματική διαχείριση των δεδομένων που λείπουν.[41]

- Seaborn

Η Seaborn είναι μια βιβλιοθήκη οπτικοποίησης στατιστικών δεδομένων που βασίζεται στη Matplotlib. Παρέχει μια διεπαφή υψηλού επιπέδου για τη σχεδίαση ελκυστικών στατιστικών διαγραμμάτων, από τα οποία εξάγεται πληροφορία . Το Seaborn απλοποιεί τη διαδικασία δημιουργίας σύνθετων απεικονίσεων όπως διαγράμματα βιολιού (violin), διαγράμματα ανάμεσα σε ζευγάρια χαρακτηριστικών (per plots),διαγράμματα joint που δείχνουν τις σχέσεις μεταξύ δύο μεταβλητών σε σχέση με την μεταβλητή στόχο και πολλά άλλα, με ελάχιστο κώδικα. Συχνά χρησιμοποιείται σε συνδυασμό με τα Pandas και NumPy για την διερεύνηση - ανάλυση δεδομένων και την παρουσίαση αποτελεσμάτων. [39]

4.2. Μεθοδολογία υλοποίησης

Η πρόβλεψη της ασθένειας του διαβήτη στους ανθρώπους συγκαταλέγεται στα προβλήματα που μπορεί να επιλυθούν με την βοήθεια της επιβλεπόμενης μηχανικής μάθησης, καθώς στις μέρες μας διαθέτουμε πολλά ιατρικά δεδομένα. Το αποτέλεσμα της πρόβλεψης μπορεί να χωριστεί σε δύο κλάσεις, πρώτη κλάση «Να αποκτήσει διαβήτη- 1» και δεύτερη κλάση «Να μην αποκτήσει διαβήτη- 0», επομένως εντάσσεται στην κατηγορία της δυαδικής ταξινόμησης.

Προκειμένου να υλοποιηθεί η μηχανική μάθηση για την πρόβλεψη του διαβήτη θα πρέπει να [42]:

1. Σύνολο δεδομένων: Στην επιβλεπόμενη μηχανική μάθηση καθοριστικό ρόλο κατέχει η επιλογή του συνόλου δεδομένων σύμφωνα με το οποίο θα πραγματοποιηθεί η εκπαίδευση του μοντέλου. Το σύνολο δεδομένων διέπεται από διάφορες διαδικασίες-τεχνικές προ επεξεργασίας με στόχο οι τιμές να γίνουν όσο πιο «καθαρές». Πιο αναλυτικά γίνεται ανάλυση και διαχείριση των κενών τιμών στο δείγμα, των N/A τιμών, των πολύ ακραίων τιμών καθώς και μετατρέπονται οι κατηγορικές τιμές σε αριθμητικές τιμές.

2. Διαχωρισμός δεδομένων και κλιμάκωση χαρακτηριστικών: Στη συνέχεια επιλέγεται το ποσοστό των δεδομένων εκείνων που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου καθώς και για την

αξιολόγηση του. Επιπλέον επιλέγεται η κατάλληλη τεχνική κλιμάκωσης των χαρακτηριστικών (Feature scaling) που θα χρησιμοποιηθεί προκειμένου τα μοντέλα να έχουν την μέγιστη δυνατή απόδοση.

3. Επιλογή μοντέλου: Σε αυτό το στάδιο γίνεται η επιλογή του αλγορίθμου ταξινόμησης με τον οποίο θα γίνει η εκπαίδευση και η αξιολόγηση. Οι πιο γνωστοί αλγόριθμοι ταξινόμησης είναι linear regression, decision trees, random forest ,support vector machines, neural networks και KNN.

4. Εκπαίδευση μοντέλου: Το επιλεγμένο μοντέλο εκπαιδεύεται στα δεδομένα εκπαίδευσης. Κατά τη διάρκεια της εκπαίδευσης, το μοντέλο μαθαίνει τα πρότυπα και τις σχέσεις στα δεδομένα.

5. Αξιολόγηση μοντέλου: Μετά την εκπαίδευση, η απόδοση του μοντέλου αξιολογείται χρησιμοποιώντας το σετ δοκιμών. Οι πιο διαδεδομένες μετρήσεις αξιολόγησης περιλαμβάνουν το Accuracy, το Precision, το recall, το F1-score, κ.λπ. Αυτό το βήμα βοηθά στην αξιολόγηση του πόσο καλά το μοντέλο σε μελλοντικά δεδομένα εισόδου.

6. Επιλογή υπερπαραμέτρων (Hyperparameters): Πολλά μοντέλα έχουν παραμέτρους που δεν επιλέγονται αυτόματα κατά τη διάρκεια της εκπαίδευσης, αλλά πρέπει να ρυθμιστούν εκ των προτέρων. Αυτές οι παράμετροι ονομάζονται υπερπαράμετροι (Hyperparameters). Η επιλογή των υπερπαραμέτρων περιλαμβάνει την προσαρμογή αυτών των υπερπαραμέτρων για τη βελτιστοποίηση της απόδοσης του μοντέλου μέσα από πολλαπλές δοκιμές.

7. Χρήση μοντέλου: Μόλις το μοντέλο ικανοποιήσει τις απαιτήσεις για την απόδοση του, μπορεί να χρησιμοποιηθεί για την πραγματοποίηση προβλέψεων σε νέα, αόρατα δεδομένα. Μπορεί να ενσωματωθεί σε κάποια εφαρμογή ή σύστημα.

Κεφάλαιο 5 :Μεθοδολογία υλοποίησης

Σε αυτό το κεφάλαιο θα πραγματοποιηθεί ανάλυση της μεθοδολογίας που ακολουθήθηκε για την υλοποίηση της εργασίας. Επιπροσθέτως θα γίνει ανάλυση του συνόλου δεδομένων που επιλέχθηκε. Τέλος θα παρουσιαστούν οι αλγόριθμοι που επιλέχθηκαν και υλοποιήθηκαν μαζί με τους βέλτιστους παραμέτρους.

5.1 Σύνολο δεδομένων

Σε αυτή την ενότητα θα γίνει η επιλογή του συνόλου δεδομένων καθώς και διερευνητική ανάλυση δεδομένων (Exploratory Data Analysis- EDA). Η EDA είναι ένα κρίσιμο βήμα στη διαδικασία ανάλυσης δεδομένων, όπου στοχεύετε στο να γίνει κατανόηση των δεδομένων, να εντοπιστούν πρότυπα καθώς και να εντοπιστούν ανωμαλίες στα δεδομένα.

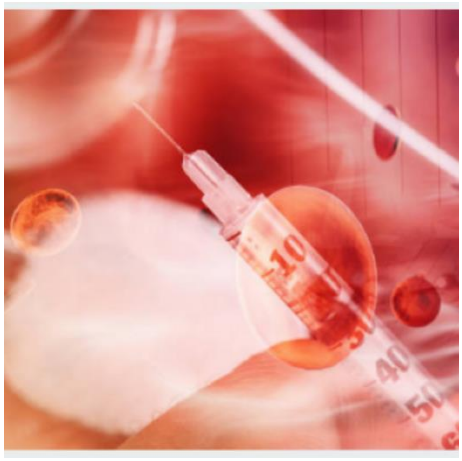
5.1.1 Επιλογή συνόλου δεδομένων

Το σύνολο δεδομένων Pima Indian Diabetes, που προέρχεται από το Εθνικό Ινστιτούτο Διαβήτη και Πεπτικών και Νεφροπαθειών των Ηνωμένων Πολιτειών της Αμερικής, περιέχει πληροφορίες για 768 γυναίκες από έναν πληθυσμό κοντά στο Φοίνιξ, Αριζόνα, ΗΠΑ. Διαβήτη είχαν 258 ενώ 500 δεν είχαν διαβήτη. Επομένως, υπάρχει μία μεταβλητή στόχος (εξαρτώμενη) και τα 8 χαρακτηριστικά: εγκυμοσύνη, γλυκόζη αίματος, αρτηριακή πίεση, πάχος δέρματος, ινσουλίνη, BMI (Δείκτης Μάζας Σώματος), ηλικία, γενεαλογική λειτουργία διαβήτη . Ο πληθυσμός του Pima μελετάται από το Εθνικό Ινστιτούτο Διαβήτη και Πεπτικών και Νεφρικών Νόσων σε διαστήματα 2 ετών από το 1965. Καθώς τα επιδημιολογικά στοιχεία δείχνουν ότι ο Διαβήτης τύπου 2 προκύπτει από αλληλεπίδραση γενετικών και περιβαλλοντικών παραγόντων, το σύνολο δεδομένων για τον διαβήτη των Ινδιάνων Pima περιλαμβάνει πληροφορίες σχετικά με τα χαρακτηριστικά που θα μπορούσε και θα έπρεπε να σχετίζονται με την εμφάνιση του διαβήτη και τις μελλοντικές επιπλοκές του.

Περιγραφή του PIMA dataset

Το σύνολο δεδομένων PIMA Indian Diabetes [43],[44] αποτελεί μια συλλογή υγειονομικών δεδομένων για γυναίκες Ινδιάνων PIMA, σχεδιασμένο για προβλεπτικές εργασίες μοντελοποίησης, ιδίως στον τομέα της έρευνας για τον διαβήτη. Αυτό το σύνολο δεδομένων περιλαμβάνει διάφορες παραμέτρους, καθεμία από τις οποίες παρέχει ενδιαφέρουσες πληροφορίες για το υγειονομικό προφίλ των ατόμων.

Παρακάτω παρέχεται μια λεπτομερής περιγραφή κάθε παραμέτρου που περιλαμβάνεται στο σύνολο δεδομένων PIMA:



Εικόνα 13: Απεικόνιση νοσοκομειακής σύριγγας

1. Εγκυμοσύνες

- *Περιγραφή:* Η παράμετρος "Εγκυμοσύνες" αντιπροσωπεύει τον αριθμό των φορών που μια γυναίκα εγκυμονεί. Είναι μια ακέραιη τιμή που χρησιμεύει ως ένδειξη της αναπαραγωγικής ιστορίας των ατόμων στο σύνολο δεδομένων. Ο αριθμός των εγκυμοσύνων είναι σημαντικός για τον κατανόηση των πιθανών επιπτώσεων των αναπαραγωγικών παραγόντων στον διαβήτη [43],[44].

2. Γλυκόζη

- *Περιγραφή:* Η παράμετρος "Γλυκόζη" μετρά τη συγκέντρωση γλυκόζης στο πλάσμα μετά από δοκιμασία ανοχής στη γλυκόζη διάρκειας 2 ωρών. Είναι ένα καθοριστικό μέτρο για την αξιολόγηση των επιπέδων γλυκόζης στο αίμα των ατόμων. Αυξημένα επίπεδα γλυκόζης μπορεί να υποδεικνύουν αντίσταση στην ινσουλίνη ή διαταραχές στον μεταβολισμό της γλυκόζης, που συνδέονται συχνά με τον διαβήτη [43],[44].

3. Αρτηριακή Πίεση

- *Περιγραφή:* "Αρτηριακή Πίεση" αντιπροσωπεύει τη διαστολική αρτηριακή πίεση των ατόμων, μετρημένη σε χιλιοστά στη στήλη (mm Hg). Η διαστολική αρτηριακή πίεση είναι ένα σημαντικό

καρδιαγγειακό μέτρο και παρέχει πληροφορίες για την συνολική κυκλοφορική υγεία των ατόμων [43],[44].

4. Πάχος Δέρματος

- *Περιγραφή:* "Πάχος Δέρματος" αναφέρεται στο πάχος του δέρματος στην περιοχή της τρικέφαλης μέτρησης σε χιλιοστά. Αυτή η μέτρηση αξιολογεί την υποδοχή υποδοχείου, παρέχοντας πληροφορίες σχετικά με τη σύνθεση του σώματος και τις πιθανές συσχετίσεις με τη μεταβολική υγεία [43],[44].

5. Ινσουλίνη

- *Περιγραφή:* Η παράμετρος "Ινσουλίνη" μετρά τη συγκέντρωση ινσουλίνης στο πλάσμα 2 ώρες μετά από το δοκιμαστικό διάστημα. Η ινσουλίνη είναι ένα ουσιώδες ορμόνη για τον μεταβολισμό της γλυκόζης και μη ομαλές τιμές ινσουλίνης μπορεί να υποδεικνύουν αντίσταση στην ινσουλίνη, μια κατάσταση συχνά συνδεδεμένη με τον διαβήτη [43],[44].

6. Δείκτης Μάζας Σώματος (ΔΜΣ)

- *Περιγραφή:* Ο "Δείκτης Μάζας Σώματος" (ΔΜΣ) υπολογίζεται ως ο λόγος βάρους προς ύψος. Είναι μια αριθμητική τιμή που προκύπτει από το βάρος και το ύψος του ατόμου. Ο ΔΜΣ λειτουργεί ως ένδειξη της συνολικής σωματικής δομής και χρησιμοποιείται συχνά για την αξιολόγηση του κινδύνου σχετιζόμενου με την παχυσαρκία, συμπεριλαμβανομένου του διαβήτη [43],[44].

7. Δείκτης Οικογενειακής Ιστορίας Διαβήτη

- *Περιγραφή:* Ο "Δείκτης Οικογενειακής Ιστορίας Διαβήτη" είναι μια αριθμητική τιμή που αντιπροσωπεύει τον αριθμό των συγγενών του ατόμου που έχουν διαβήτη. Αυτός ο δείκτης παρέχει πληροφορίες σχετικά με το γενετικό υπόβαθρο του διαβήτη και την πιθανή προδιάθεση των ατόμων για τη νόσο [43],[44].

8. Ηλικία

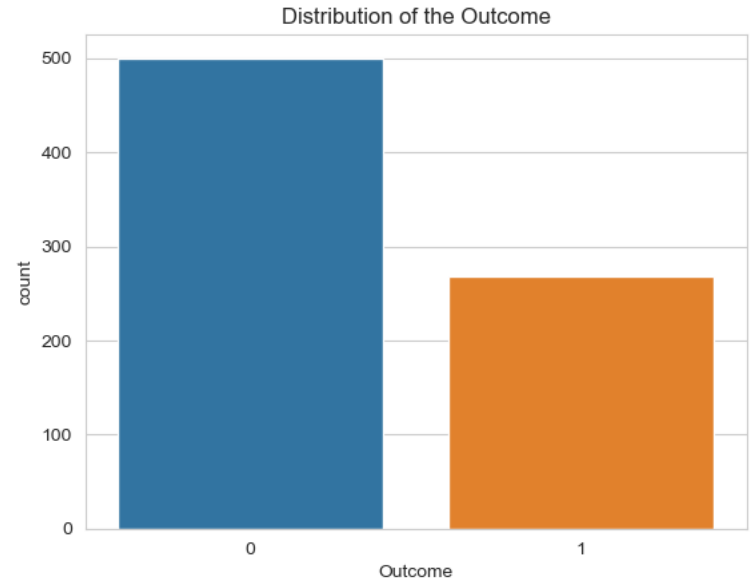
- *Περιγραφή:* Η "Ηλικία" αντιπροσωπεύει την ηλικία των ατόμων, μετρημένη σε έτη. Η ηλικία αποτελεί σημαντική παράμετρο για την κατανόηση των συνδέσεων μεταξύ της ηλικίας και του διαβήτη, καθώς η νόσος συχνά επηρεάζει περισσότερο τις ενήλικες ηλικίας [43],[44].

9. Αποτέλεσμα-Διαβήτης

- *Περιγραφή:* Το πεδίο "Αποτέλεσμα" αντιπροσωπεύει την παρουσία ή απουσία διαβήτη, με το "1" να υποδηλώνει την παρουσία του διαβήτη και το "0" την απουσία. Αυτή είναι η βασική ετικέτα κλάσης που χρησιμοποιείται για τον στόχο των προβλέψεων στα προβλεπτικά μοντέλα [43],[44].

Το πλήρες σύνολο δεδομένων PIMA παρέχει ένα ευρύ φάσμα πληροφοριών που καλύπτουν τον γενετικό, φυσιολογικό, και γενικό υγειονομικό χαρακτήρα των ατόμων, και αποτελεί ένα χρήσιμο σύνολο δεδομένων για την ανάλυση και την ανάπτυξη προβλεπτικών μοντέλων στον τομέα της υγείας.

Όπως φαίνεται παρακάτω στο σχήμα 1 από το σύνολο των δεδομένων των 768 στιγμιοτύπων διαβήτη έχουν τα 268 στιγμιότυπα ενώ 500 στιγμιότυπα δεν έχουν.



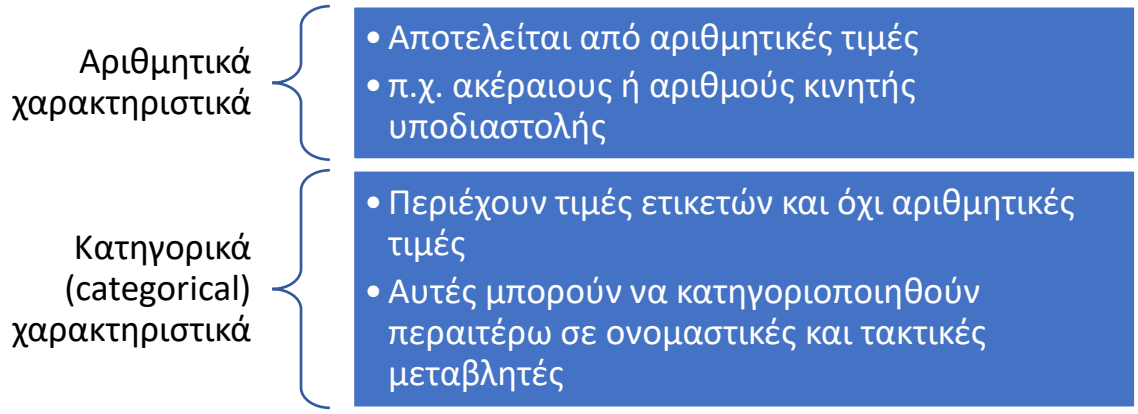

```
Outcome
0    500
1    268
```

Γραφική παράσταση 1: Διάγραμμα διασποράς δείγματος στιγμιοτύπων με διαβήτη ή όχι

5.1.2. Μηχανική χαρακτηριστικών (Feature Engineering)

Η μηχανική χαρακτηριστικών είναι η τέχνη της μετατροπής ακατέργαστων δεδομένων (μεταβλητές εισόδου) σε σημαντικά χαρακτηριστικά που μπορούν να χρησιμοποιηθούν αποτελεσματικά από μοντέλα μηχανικής μάθησης. Περιλαμβάνει την επιλογή, την εξαγωγή και τη μετατροπή σχετικών πληροφοριών από τα διαθέσιμα δεδομένα. Ο στόχος είναι να βελτιωθεί η ακρίβεια του μοντέλου. Η καλή μηχανική χαρακτηριστικών μπορεί να επηρεάσει σημαντικά την απόδοση ενός μοντέλου μηχανικής μάθησης, συχνά περισσότερο από την επιλογή του ίδιου του μοντέλου.[14]

Στη μηχανική μάθηση, ένα χαρακτηριστικό αντιπροσωπεύει μια μεμονωμένη μετρήσιμη ιδιότητα ή χαρακτηριστικό ενός σημείου δεδομένων. Τα χαρακτηριστικά μπορεί να είναι διαφορετικών τύπων:

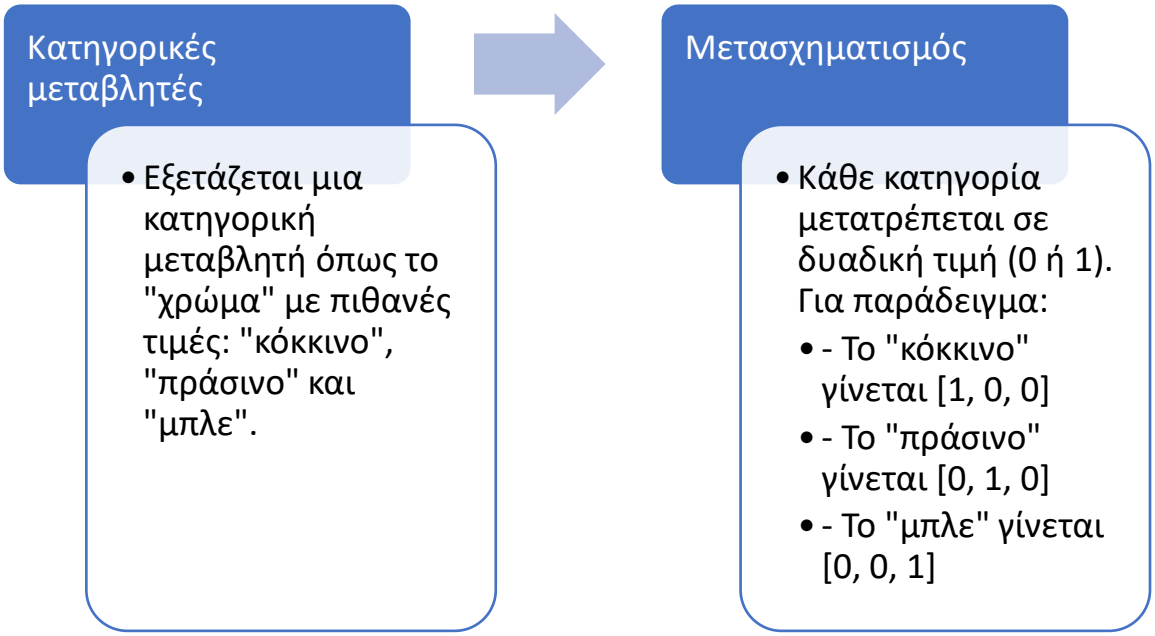


Σχήμα 2: Σχήμα με Αριθμητικά και Κατηγορικά χαρακτηριστικά

Τεχνικές που χρησιμοποιούνται στη μηχανική χαρακτηριστικών καθώς και στην παρούσα εργασία:

- *One-Hot Κωδικοποίηση*

Η One-hot κωδικοποίηση είναι μια τεχνική που χρησιμοποιείται για τη μετατροπή κατηγορικών μεταβλητών σε αριθμητικές τιμές που μπορούν να χρησιμοποιηθούν απευθείας από μοντέλα μηχανικής εκμάθησης. Δουλεύει ακολούθως:

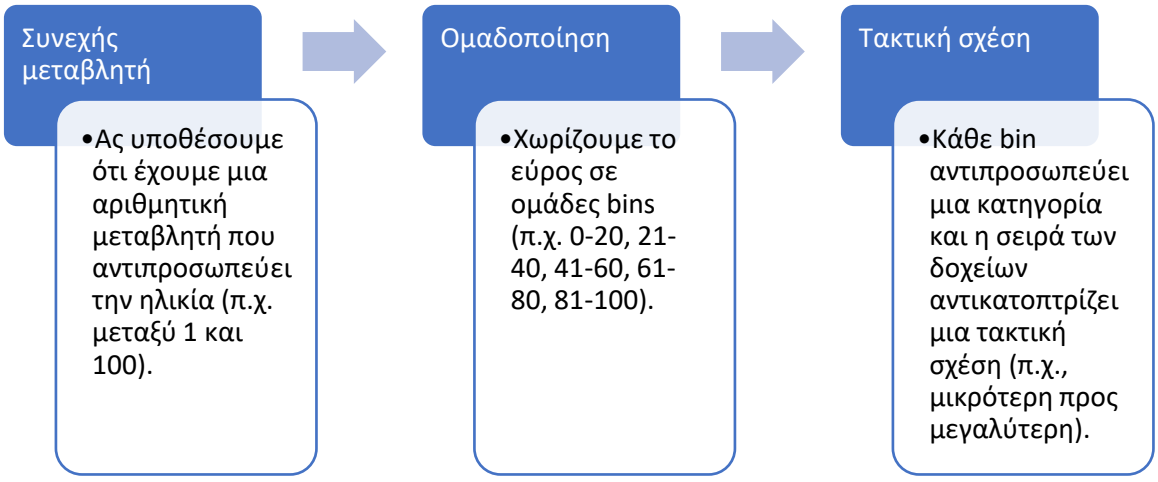


Σχήμα 3: Σχήμα αναπαράστασης One Hot Encoding

Αυτός ο μετασχηματισμός διασφαλίζει ότι το μοντέλο μπορεί να χειριστεί αποτελεσματικά κατηγορικά δεδομένα. Η βιβλιοθήκη scikit-learn παρέχει λειτουργίες που βοηθούν στην κωδικοποίηση one-hot.

➢ *Binning (Ομαδοποίηση)*

Το Binning (ομαδοποίηση), γνωστό και ως discretization, περιλαμβάνει τη διαίρεση μιας συνεχούς αριθμητικής μεταβλητής σε διακριτά bins ή ομάδες. Αυτή η μέθοδος είναι χρήσιμη όταν θέλουμε να μετατρέψουμε ένα αριθμητικό χαρακτηριστικό σε μια τακτική μεταβλητή. Δουλεύει με τον παρακάτω τρόπο:



Σχήμα 4: Σχήμα αναπαράστασης τεχνικής Binning

Με το binning, δημιουργούμε μια νέα δυνατότητα που καταγράφει την ουσία της αρχικής συνεχούς μεταβλητής ενώ την καθιστά κατάλληλη για ορισμένους αλγόριθμους.

- Επεξεργασία δεδομένων κειμένου (Text Data Processing)

  Προεπεξεργασία και μετατροπή δεδομένων κειμένου σε αριθμητικά χαρακτηριστικά που μπορούν να χρησιμοποιηθούν σε μοντέλα μηχανικής εκμάθησης. Οι τεχνικές περιλαμβάνουν το tokenization, το stemming/lemmatization και τη διανυσματοποίηση TF-IDF (Term Frequency-Inverse Document Frequency).

- Καταλογισμός (imputation)

  Χειρισμός τιμών που λείπουν στο σύνολο δεδομένων συμπληρώνοντάς τες με μια συγκεκριμένη τιμή (π.χ. μέση τιμή, διάμεσος, τρόπος λειτουργίας) ή χρησιμοποιώντας πιο προηγμένες τεχνικές όπως K-πλησιέστεροι γείτονες ή παρεμβολή.

- Επιλογή χαρακτηριστικών (Feature Selection)

Προσδιορισμός και επιλογή των πιο σχετικών χαρακτηριστικών που συμβάλλουν περισσότερο στην προγνωστική απόδοση του μοντέλου. Αυτό μπορεί να γίνει χρησιμοποιώντας τεχνικές όπως ανάλυση συσχέτισης, βαθμολογίες σπουδαιότητας χαρακτηριστικών (π.χ. με βάση τα δέντρα απόφασης) ή επιλογή βάσει μοντέλου.

Αυτές οι τεχνικές χρησιμοποιούνται συχνά σε συνδυασμό και η επιλογή των τεχνικών εξαρτάται από τη φύση του συνόλου δεδομένων και το συγκεκριμένο πρόβλημα που αντιμετωπίζεται. Η αποτελεσματική μηχανική χαρακτηριστικών απαιτεί βαθιά κατανόηση των δεδομένων και του τομέα του προβλήματος, καθώς και πειραματισμό για τον προσδιορισμό των τεχνικών που αποδίδουν τα καλύτερα αποτελέσματα. Στην παρούσα εργασία έχει γίνει χρήση των τεχνικών :

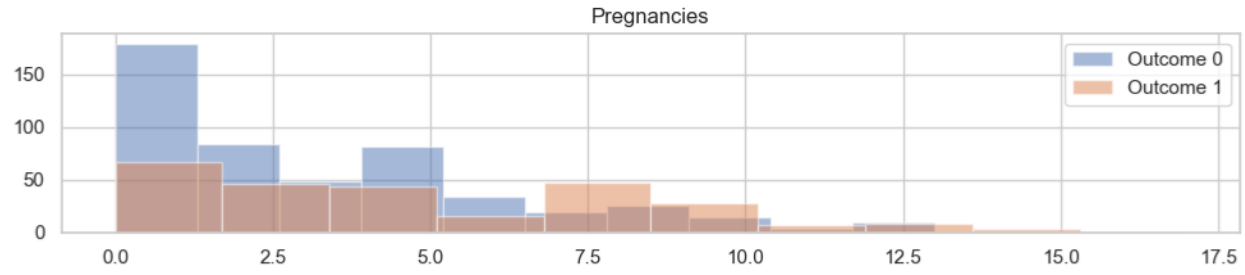
1. Καταλογισμός (imputation)
2. Ομαδοποίηση (Binning)
3. *One-Hot Κωδικοποίηση*

5.1.3. Διερευνητική Ανάλυση Δεδομένων (EDA)

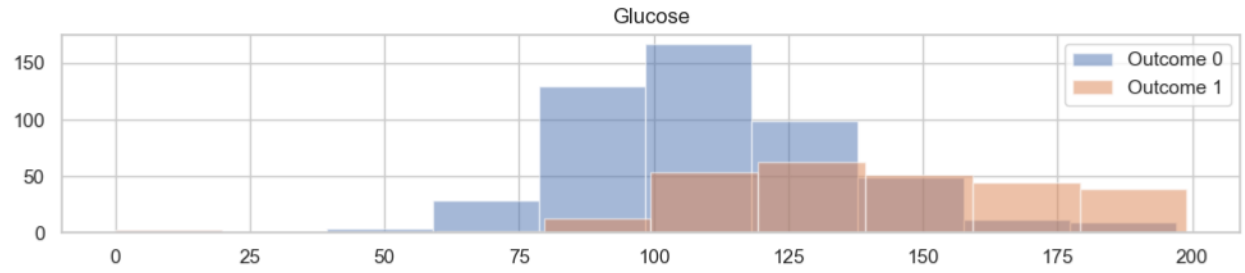
Πρώτο βήμα για την κατανόηση των συνόλων δεδομένων είναι η διερευνητική ανάλυση δεδομένων (EDA) , χρησιμοποιώντας οπτικές και στατιστικές τεχνικές για την αποκάλυψη μοτίβων, ανωμαλιών και σχέσεων μέσα στα δεδομένα. Με αυτό τον τρόπο θα αποκτηθεί χρήσιμη γνώση σχετικά με τη δομή και τα χαρακτηριστικά του συνόλου δεδομένων.

5.1.3.1. Κατανομές χαρακτηριστικών με βάση τη κλάση ταξινόμησης

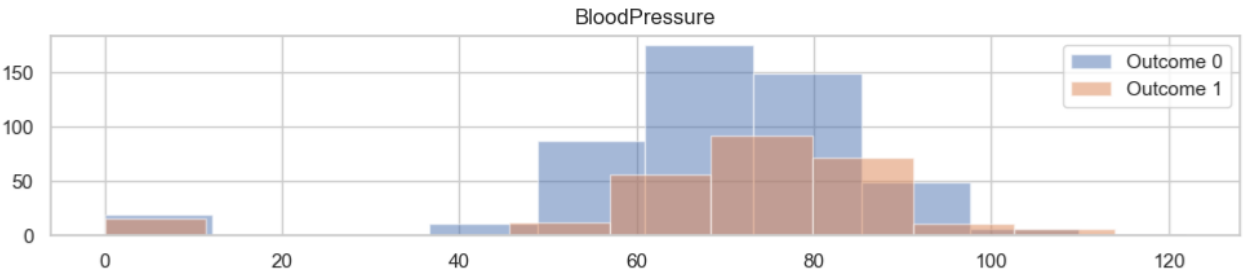
Σε αυτή την ενότητα παρουσιάζεται η συσχέτιση κάθε χαρακτηριστικού ως προς τη κλάση ταξινόμησης.



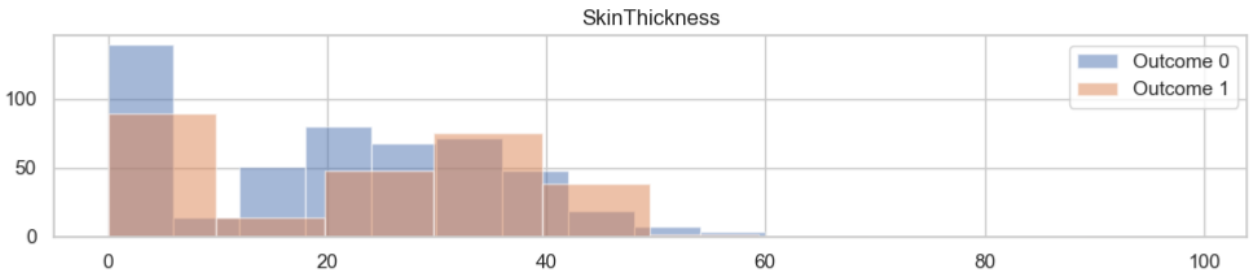
Γραφική παράσταση 2: Εγκυμοσύνες ως προς κλάση



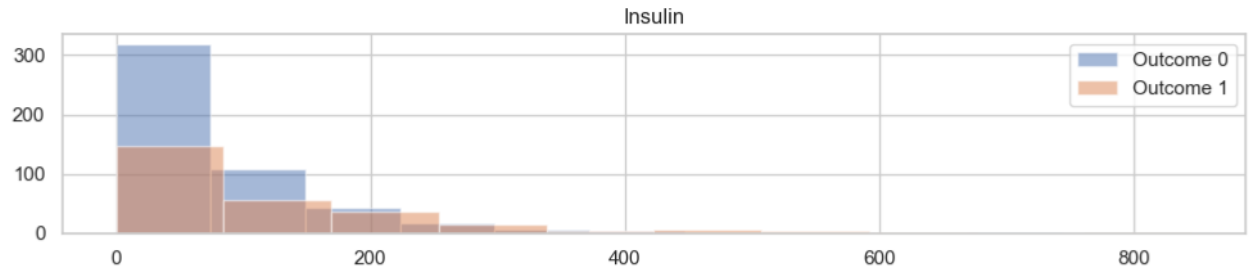
Γραφική παράσταση 3: Γλυκόζη ως προς την κλάση



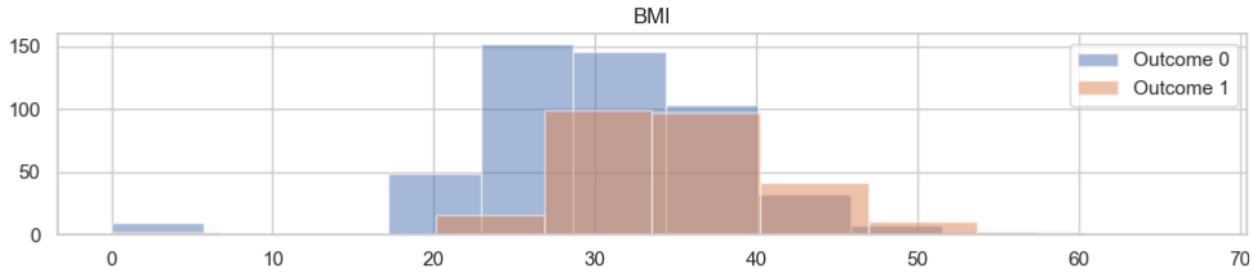
Γραφική παράσταση 4: Πίεση αίματος ως προς την κλάση



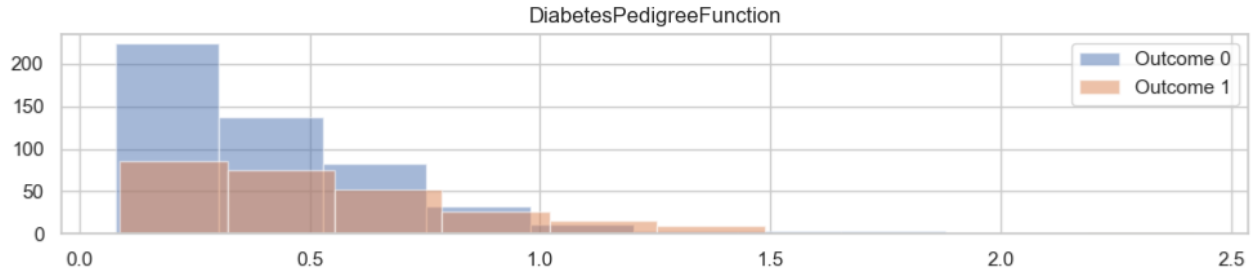
Γραφική παράσταση 5: Πάχος δέρματος ως προς την κλάση



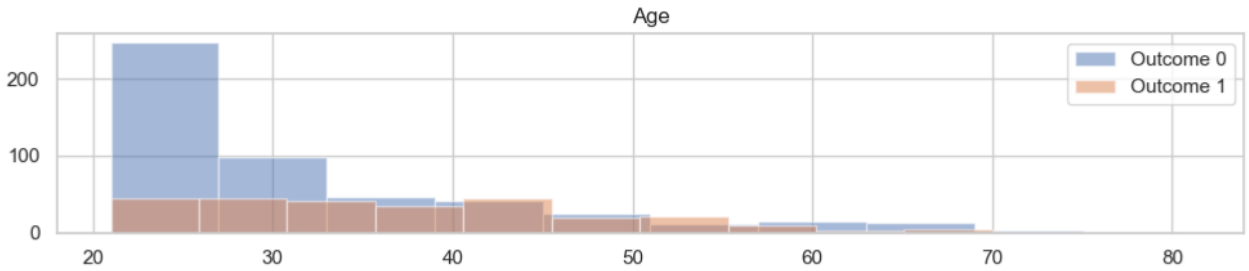
Γραφική παράσταση 6: Ινσουλίνη ως προς την κλάση



Γραφική παράσταση 7: Δείκτης μάζας σώματος ως προς την κλάση



Γραφική παράσταση 8: Δείκτης Οικογενειακής Ιστορίας Διαβήτη ως προς την κλάση



Γραφική παράσταση 9: Ηλικία ως προς την κλάση

Παρατηρούμε πως όσες γυναίκες είχαν πολλές εγκυμοσύνες έχουν παραπάνω πιθανότητες να παρουσιάσουν διαβήτη, το ίδιο ισχύει και για το πάχος του δέρματος τους, όσο πιο παχύ είναι το δέρμα έχουν παραπάνω πιθανότητες να αναπτύξουν διαβήτη. Τέλος η μεταβλητή της ινσουλίνης φαίνεται να μην επηρεάζει τόσο πολύ την κλάση, το ίδιο ισχύει και για τις μεταβλητές του οικογενειακού ιστορικού και ηλικίας.

5.1.3.2. Διαγράμματα pair plots

Αρχικά ψάχνουμε να βρούμε ποια από τα χαρακτηριστικά έχουν συσχέτιση μεταξύ τους. Κατασκευάζουμε pairplots όπως φαίνεται στο Σχήμα 5.

Γραφική παράσταση 10: Pairplots μεταβλητών

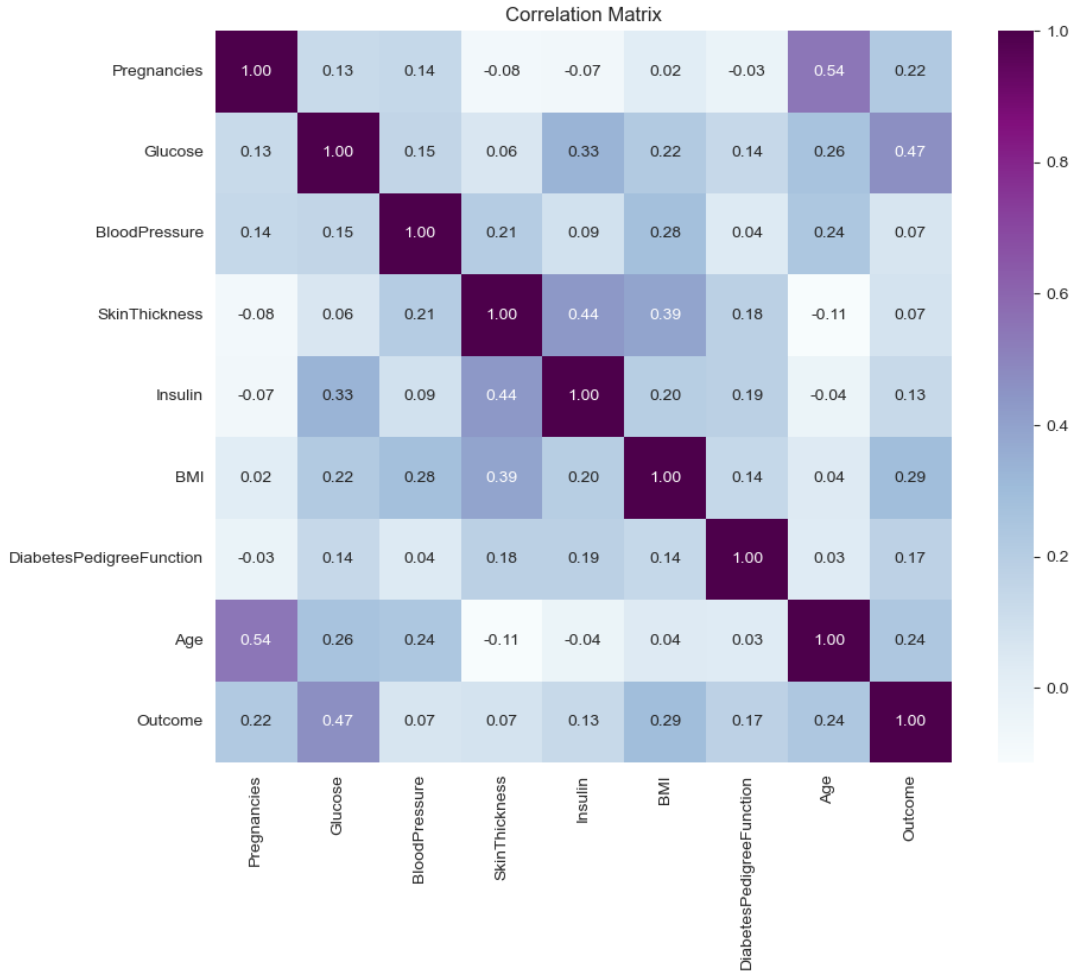
Θετική γραμμική σχέση φαίνεται πως έχουν τα χαρακτηριστικά BMI με το πάχος του δέρματος (SkinThickness) και Εγκυμοσύνες (Pregnancies) με την Ηλικία (Age). Η γλυκόζη (Glucose) με την ινσουλίνη (Insulin) θα μπορούσαν να έχουν θετική γραμμική σχέση. Τέλος φαίνεται πως δεν υπάρχουν μη γραμμικές σχέσεις.

5.1.3.3. Συσχετίσεις χαρακτηριστικών (Correlation matrix)

Σε αυτή την ενότητα δημιουργούμε ένα correlation matrix σε ένα διάγραμμα «θερμότητας» (Heat map), προκειμένου να δούμε το μητρώο συσχέτισης όλων των δεδομένων [45].

Ένας πίνακας συσχέτισης είναι ένας πίνακας που δείχνει συντελεστές συσχέτισης μεταξύ μεταβλητών. Κάθε κελί στον πίνακα αντιπροσωπεύει τον συντελεστή συσχέτισης μεταξύ δύο μεταβλητών. Ο συντελεστής συσχέτισης μετρά την ισχύ και την κατεύθυνση μιας γραμμικής σχέσης μεταξύ δύο μεταβλητών. Κυμαίνεται από -1 έως 1, όπου:

- Το 1 δείχνει μια τέλεια θετική γραμμική σχέση
- Το -1 δείχνει μια τέλεια αρνητική γραμμική σχέση
- Το 0 υποδηλώνει ότι δεν υπάρχει γραμμική σχέση



Σχήμα 5: Correlation matrix σε Heat map

Τα αποτελέσματα του πίνακα συσχετίσεων επιβεβαιώνουν το αποτέλεσμα από τα pair plots καθώς θετική γραμμική συσχέτιση φαίνεται πως έχουν τα χαρακτηριστικά BMI με το πάχος του δέρματος (SkinThickness) και Εγκυμοσύνες (Pregnancies) με την Ηλικία (Age). Η γλυκόζη (Glucose) με την ινσουλίνη (Insulin) φαίνεται πως έχουν θετική έχουν γραμμική σχέση. Τέλος η μεταβλητή γλυκόζη (Glucose) φαίνεται πως έχει μεγάλο βαθμό συσχέτισης με το να έχει κάποιος διαβήτη ή όχι.

5.2. Διαχωρισμός δεδομένων και κλιμάκωση χαρακτηριστικών

5.2.1. Διαχωρισμός δεδομένων

Στη μηχανική μάθηση, ο διαχωρισμός δεδομένων αναφέρεται στη διαίρεση ενός συνόλου δεδομένων σε διαφορετικά υποσύνολα για διαφορετικούς σκοπούς, όπως εκπαίδευση, επικύρωση και δοκιμή. Στην παρούσα εργασία γίνεται διαχωρισμός δεδομένων ως εξής:

Σύνολο εκπαίδευσης (Training set): Αυτό το υποσύνολο δεδομένων χρησιμοποιείται για την εκπαίδευση του μοντέλου μηχανικής εκμάθησης. Το μοντέλο μαθαίνει μοτίβα και σχέσεις από αυτά τα δεδομένα. Σε κάθε αλγόριθμο που θα δούμε παρακάτω γίνεται χρήση του 80% του συνόλου δεδομένων για την εκπαίδευση του.

Σετ επικύρωσης (Validation set): Μετά την εκπαίδευση του μοντέλου, είναι απαραίτητο να αξιολογηθεί η απόδοσή του σε δεδομένα που δεν έχει δει στο παρελθόν. Το σύνολο επικύρωσης βοηθά στον συντονισμό των υπερπαραμέτρων και στην αξιολόγηση της ικανότητας γενίκευσης του μοντέλου. Χρησιμοποιείται για την πρόβλεψη των εναπομεινάντων δειγμάτων από το σύνολο εκπαίδευσης.

5.2.2. Κλιμάκωση χαρακτηριστικών (Feature scaling)

Η κλιμάκωση χαρακτηριστικών είναι μια κρίσιμη τεχνική στη μηχανική μάθηση που διασφαλίζει την ισάξια συμβολή όλων των χαρακτηριστικών εξίσου στο μοντέλο μετατρέποντας τις τιμές τους σε παρόμοια κλίμακα. Είναι απαραίτητη διαδικασία όταν τα σύνολα δεδομένων περιέχουν χαρακτηριστικά

με διαφορετικά εύρη, μονάδες ή μεγέθη. Οι πιο κοινές μέθοδοι είναι η κανονικοποίηση (Normalization) και η προτυποποίηση (Standardization), οι οποίες προσαρμόζουν τις τιμές χαρακτηριστικών διατηρώντας παράλληλα τις σχέσεις τους. Η κλιμάκωση επιτρέπει ακριβείς συγκρίσεις μεταξύ των χαρακτηριστικών, βελτιώνει τη σύγκλιση του μοντέλου και αποτρέπει την κυριαρχία οποιουδήποτε μεμονωμένου στοιχείου με βάση αποκλειστικά το μέγεθός του. [15]

Η τεχνική της κανονικοποίησης (Normalization) χρησιμοποιεί την συνάρτηση MinMaxScaler, με την οποία μετατρέπει τις τιμές των χαρακτηριστικών σε ένα σταθερό εύρος, που κυμαίνεται μεταξύ 0 και 1. Ο τύπος που χρησιμοποιείται είναι : $X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}}$

Εξίσωση 9: Μαθηματικός τύπος κανονικοποίησης

Η τεχνική της προτοτυποποίησης (Standardization) χρησιμοποιεί την συνάρτηση StandardScaler, με την οποία μετατρέπει έτσι τις τιμές των χαρακτηριστικών ώστε να έχουν μέση τιμή ίση με το 0 και τυπική απόκλιση ίση με το 1. Ο τύπος που χρησιμοποιεί είναι :

$$X_{new} = \frac{X - \mu}{\sigma}$$

Εξίσωση 10: Μαθηματικός τύπος κανονικοποίησης

Τέλος και οι δύο τεχνικές κλιμάκωσης βρίσκονται στην Scikit-Learn βιβλιοθήκη της Python, και θα τις εφαρμόσουμε στην παρούσα εργασία.

5.3. Επιλογή μοντέλου – εκπαίδευση μοντέλου

Σε αυτό το κεφάλαιο θα πραγματοποιηθεί αναφορά των μοντέλων που επιλέχθηκαν καθώς και των παραμέτρων που χρησιμοποιήθηκαν για κάθε μοντέλο.

5.3.1. Επιλογή μοντέλου δυαδικής ταξινόμησης

Κύριος στόχος στην παρούσα εργασία είναι η διερεύνηση της αποτελεσματικότητας διαφορετικών αλγορίθμων μηχανικής μάθησης, στην πρόβλεψη εμφάνισης διαβήτη με βάση τα χαρακτηριστικά που

είναι διαθέσιμα στο σύνολο δεδομένων PIMA. Στην συνέχεια επιλέχθηκαν να χρησιμοποιηθούν τα πιο γνωστά μοντέλα δυαδικής ταξινόμησης όπως:

1. Λογιστική παλινδρόμηση (Logistic Regression)
2. Μηχανές διανυσμάτων υποστήριξης (SVM)
3. Τυχαία δάση (Random Forest)
4. Κ-Κοντινότεροι γείτονες (KNN)
5. Gradient Boosting
6. Μοντέλο Perceptron πολλών στρωμάτων (Multilayer Perceptron -MLP).

Τα παραπάνω μοντέλα βρίσκονται στην βιβλιοθήκη Scikit-Learn της Python, που όπως έχουμε αναφέρει εκεί βρίσκονται οι περισσότεροι αλγόριθμοι που χρησιμοποιούνται για την μηχανική μάθηση.

5.3.2. Εκπαίδευση μοντέλου

Για την εκπαίδευση του κάθε μοντέλου από τα παραπάνω θα γίνεται χρήση του 80% του συνόλου δεδομένων PIMA. Αυτό θα είναι το σύνολο εκπαίδευσης (Training set). Το υπόλοιπο 20% θα χρησιμοποιηθεί για την αξιολόγηση του. Κατά τη διάρκεια της εκπαιδευτικής διαδικασίας το μοντέλο προσαρμόζει τις παραμέτρους του επαναληπτικά με βάση τις πληροφορίες που περιέχονται στο σύνολο εκπαίδευσης, με στόχο να ελαχιστοποιήσει την απόκλιση μεταξύ των προβλέψεων του και των πραγματικών τιμών-στόχων. Η ποιότητα του σετ εκπαίδευσης παίζει καθοριστικό ρόλο στην απόδοση του μοντέλου, καθώς και στην ικανότητα της γενίκευσης του μοντέλου. Για αυτό τον λόγο είναι απαραίτητο να εφαρμοστούν τεχνικές προ επεξεργασίας των δεδομένων πριν την εκπαίδευση. Τέλος ένα καλά προετοιμασμένο σετ εκπαίδευσης θέτει τις βάσεις για ένα ισχυρό και αποτελεσματικό μοντέλο μηχανικής μάθησης.

5.4. Αξιολόγηση μοντέλου

Κατά τη σύγκριση των αποτελεσμάτων των προβλέψεων που γίνονται από ένα μοντέλο με την πραγματική κλάση ταξινόμησης, μπορούν να προκύψουν διάφορα σενάρια αποτελεσμάτων, καθένα από τα οποία παρέχει πολύτιμες πληροφορίες για την απόδοση του μοντέλου. Αυτά τα αποτελέσματα αναλύονται τυπικά χρησιμοποιώντας έναν πίνακα σύγχυσης (Confusion Matrix) , ο οποίος οργανώνει τις προβλέψεις σε τέσσερις κατηγορίες: Αληθινό θετικό (True Positive-TP), Αληθινό αρνητικό (True Negative -TN), ψευδώς θετικό (False Positive -FP) και ψευδώς αρνητικό (False Negative -FN).

Ο πίνακας σύγχυσης συνοψίζει την απόδοση ενός αλγορίθμου ταξινόμησης παρουσιάζοντας τις μετρήσεις των αληθινών θετικών, των αληθινών αρνητικών, των ψευδώς θετικών και των ψευδώς αρνητικών. Από τον πίνακα σύγχυσης, μπορούν να προκύψουν διάφορες μετρήσεις απόδοσης, συμπεριλαμβανομένης της ορθότητας (Accuracy), της ανάκλησης (Recall) , της βαθμολογίας F1 (F1- score) και της ακρίβειας (Precision), παρέχοντας μια λεπτομερή κατανόηση της απόδοσης του μοντέλου σε διαφορετικές κατηγορίες.

Με άλλα λόγια, μας βοηθά να καταλάβουμε πόσο καλά τα πηγαίνει το μοντέλο μας όσον αφορά την πραγματοποίηση σωστών και εσφαλμένων προβλέψεων.

Είναι ιδιαίτερα χρήσιμο σε προβλήματα δυαδικής κατηγοριοποίησης (όπου υπάρχουν δύο κατηγορίες, π.χ. διαβήτης και όχι διαβήτης).Ο πίνακας σύγχυσης αποτελείται από τέσσερα βασικά στοιχεία:

Αληθινά θετικά (TP)	Αληθινά αρνητικά (TN)	Ψευδώς Θετικά (FP)	Ψεδώς Αρνητικά (FN)
• Ο αριθμός των περιπτώσεων που έχουν προβλεφθεί σωστά ως θετικές (π.χ. σωστή αναγνώριση των γυναικών που έχουν διαβήτη).	• Ο αριθμός των περιπτώσεων που έχουν προβλεφθεί σωστά ως αρνητικές (π.χ. σωστή αναγνώριση των γυναικών που δεν έχουν διαβήτη).	• Ο αριθμός των περιπτώσεων που έχουν προβλεφθεί εσφαλμένα ως θετικές (π.χ. ταξινόμηση μιας μη διαβητικής γυναίκας ως διαβητική).	• Ο αριθμός των περιπτώσεων που προβλέφθηκαν λανθασμένα ως αρνητικές (π.χ. ταξινόμηση μιας διαβητικής γυναίκας ως μη διαβητική).

Ορθότητα (Accuracy): Η ορθότητα είναι μια κοινή μέτρηση που αξιολογεί τη συνολική ακρίβεια του μοντέλου. Υπολογίζεται ως εξής:

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

Εξίσωση 11: Μαθηματικός τύπος για το Accuracy

Ωστόσο, η ορθότητα από μόνη της μπορεί να είναι παραπλανητική, ειδικά όταν οι κλάσεις είναι μη ισορροπημένες.

Ακρίβεια (Precision): Η ακρίβεια μετρά πόσες από τις θετικές προβλέψεις που έγιναν από το μοντέλο είναι πραγματικά σωστές. Υπολογίζεται ως εξής:

$$Precision = \frac{TP}{TP + FP}$$

Εξίσωση 12: Μαθηματικός τύπος για το Pcecision

Υψηλή ακρίβεια σημαίνει ότι όταν το μοντέλο προβλέπει, θα ταξινομήσει συνήθως σωστά στην θετική κλάση (π.χ. διαβήτης).

Ανάκληση (Recall): Η ανάκληση (γνωστή και ως ευαισθησία ή πραγματικό θετικό ποσοστό) μετρά πόσο καλά το μοντέλο καταγράφει όλες τις θετικές περιπτώσεις. Υπολογίζεται ως εξής:

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

Εξίσωση 13: Μαθηματικός τύπος για το Recall

Η υψηλή ανάκληση σημαίνει ότι το μοντέλο είναι καλό στον εντοπισμό θετικών περιπτώσεων.

Specificity (πραγματικό αρνητικό ποσοστό): Το Specificity μετρά πόσο καλά το μοντέλο εντοπίζει αρνητικές περιπτώσεις. Υπολογίζεται ως εξής:

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

Εξίσωση 14: Μαθηματικός τύπος για το Specificity

Βαθμολογία F1: Η βαθμολογία F1 συνδυάζει την ακρίβεια και την ανάκληση σε μια ενιαία μέτρηση:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Εξίσωση 15: Μαθηματικός τύπος για το F1 Score

ένα είδος ισορροπίας μεταξύ της ακρίβειας και την ανάκλησης, χρησιμοποιούμε μια άλλη μετρική που συνδυάζει τις δύο μετρικές, τον αρμονικό μέσο όρο της ακρίβειας και της ανάκλησης που ονομάζεται F1 αποτέλεσμα (F1 score)

Εξισορροπεί την ακρίβεια και την ανάκληση, καθιστώντας τη χρήσιμη όταν και τα δύο είναι σημαντικά. Το F1 αποτέλεσμα είναι ο αρμονικός μέσος όρος της ακρίβειας και της ανάκλησης.

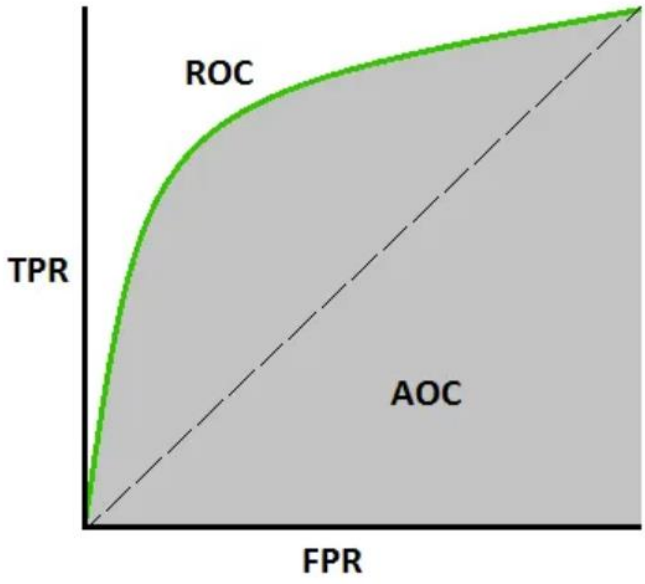
Βαθμολογία ROC-AUC:

Η καμπύλη ROC (Receiver Operating Characteristic) είναι μια μέτρηση απόδοσης που χρησιμοποιείται συχνά σε προβλήματα ταξινόμησης για την απεικόνιση της αντιστάθμισης μεταξύ True Positive Rate (TPR) και False Positive Rate (FPR). Είναι μια καμπύλη πιθανότητας που απεικονίζει την ικανότητα του μοντέλου να διακρίνει σωστά τις κλάσεις των δεδομένων. Το AUC (Εμβαδόν κάτω από την καμπύλη) ή AUC-PR είναι ένα αριθμητικό μέτρο της διαχωρισιμότητας των κλάσεων που αντιπροσωπεύεται από την καμπύλη ROC. Μια υψηλότερη AUC υποδηλώνει ότι το μοντέλο είναι καλύτερο στο να αναγνωρίζει σωστά τις περιπτώσεις και των δύο κλάσεων.

$$TPR = \frac{TP}{TP+FN} \qquad \text{και} \qquad FPR = \frac{FP}{TN+FP}$$

Εξίσωση 16: Μαθηματικός τύπος για TPR και FPR

Στην ουσία, η καμπύλη ROC απεικονίζει την ευαισθησία του μοντέλου (True Positive Rate - TPR) έναντι του specificity (True Negative Rate - TNR), με το TPR να απεικονίζεται στον άξονα y και το FPR στον άξονα x. Αυτή η γραφική αναπαράσταση βοηθά στην κατανόηση του πόσο καλά το μοντέλο μπορεί να διαφοροποιήσει μεταξύ των 2 κλάσεων (Θετικής και αρνητικής πχ. Έχει διαβήτη και δεν έχει διαβήτη).[10]



Εικόνα 14: Απεικόνιση AUC – ROC

5.5. Επιλογή υπερπαραμέτρων (Hyperparameters)

Η αποτελεσματικότητα ενός μοντέλου δεν καθορίζεται μόνο από την αρχιτεκτονική του ή το πλήθος των δεδομένων εκπαίδευσης του, αλλά και από την προσεκτική επιλογή των υπερπαραμέτρων. Αυτές οι προσαρμόσιμες ρυθμίσεις είναι συχνά ανεπαίσθητες αλλά καθοριστικές καθώς, υπαγορεύουν τη συμπεριφορά και την απόδοση του μοντέλου κατά τη διάρκεια της εκπαίδευσης αλλά και των συμπερασμάτων.

Σε όλους τους αλγόριθμους εφαρμόστηκαν οι βέλτιστοι υπερπαράμετροι που βρέθηκαν με την βοήθεια της Grid search SV η οποία είναι μια τεχνική που χρησιμοποιείται για την ρύθμιση υπερπαραμέτρων σε μοντέλα μηχανικής μάθησης, που υπάρχει στην python στην βιβλιοθήκη scikit learn. Με άλλα λόγια είναι μια μέθοδος εξαντλητικής αναζήτησης μέσω ενός καθορισμένου υποσυνόλου υπερπαραμέτρων για ένα μοντέλο.[4]

- *Logistic Regression Hyperparameters*

  Οι υπερπαράμετροι που ορίστηκαν στην παρούσα εργασία για την Logistic Regression είναι οι εξής:

LogisticRegression(C=0.1,penalty='l2',random_state=0,solver='lbfgs',max_iter=100,fit_intercept=True,tol=1e-4,class_weight=None,dual=False,verbose=0,warm_start=False)

- SVM Hyperparameters

  Οι υπερπαράμετροι που ορίστηκαν στην παρούσα εργασία για την SVM είναι οι εξής:

SVC(kernel='linear',C=0.1,gamma='scale',random_state=0,probability=False,shrinking=True,tol=1e-3,cache_size=200,class_weight=None,verbose=False,max_iter=-1,decision_function_shape='ovr', break_ties=False)

- *Random Forest Hyperparameters*

  Οι υπερπαράμετροι που ορίστηκαν στην παρούσα εργασία για το Random Forest είναι οι εξής:

*RandomForestClassifier(n_estimators=100,max_depth=10,min_samples_split=5,min_samples_leaf=2,random_state=0,bootstrap=True,criterion='gini',max_features=None,max_leaf_nodes=None,min_impurity_*

decrease=0.0,min_weight_fraction_leaf=0.0,n_jobs=None,oob_score=False,verbose=0,warm_start=False,class_weight=None,ccp_alpha=0.0,max_samples=None)

- *KNN Hyperparameters*

Οι υπερπαράμετροι που ορίστηκαν στην παρούσα εργασία για το KNN είναι οι εξής:

KNeighborsClassifier(n_neighbors=7,p=1,weights='distance',algorithm='auto',leaf_size=30,metric='minkowski', metric_params=None,n_jobs=None)

- *Gradient Boosting Hyperparameters*

Οι υπερπαράμετροι που ορίστηκαν στην παρούσα εργασία για το Gradient Boosting είναι οι εξής:

GradientBoostingClassifier(learning_rate=0.01,n_estimators=100,max_depth=3,min_samples_split=2,min_samples_leaf=4,random_state=0,loss='log_loss',criterion='friedman_mse',min_weight_fraction_leaf=0.0,subsample=1.0,max_features=None,max_leaf_nodes=None,validation_fraction=0.1,n_iter_no_change=None,tol=0.0001,ccp_alpha=0.0,init=None,verbose=0,warm_start=False)

- *Multilayer Perceptron (MLP) Hyperparameters*

Οι υπερπαράμετροι που ορίστηκαν στην παρούσα εργασία για το Multilayer Perceptron είναι οι εξής:

MLPClassifier(alpha=0.001,hidden_layer_sizes=(50,),max_iter=1000,random_state=0,activation='relu',solver='adam',batch_size='auto',learning_rate='constant',learning_rate_init=0.001,power_t=0.5,momentum=0.9,nesterovs_momentum=True,early_stopping=False,validation_fraction=0.1,beta_1=0.9,beta_2=0.999,epsilon=1e-08,n_iter_no_change=10)

5.6. Χρήση μοντέλου

Τέλος όταν το μοντέλο έχει επιτύχει την μέγιστη απόδοση τότε μπορεί να αναπτυχθεί στην παραγωγή ή να ενσωματωθεί σε κάποια εφαρμογή. Έτσι θα μπορούν οι χρήστες να εκτελούν την επιθυμητή εργασία πραγματοποίησης προβλέψεων για την νόσο του διαβήτη εισάγοντας τα απαραίτητα δεδομένα.

Κεφάλαιο 6: Αποτελέσματα Υλοποίησης

Στο κεφάλαιο αυτό θα παρουσιαστούν αναλυτικά τα αποτελέσματα της εκτέλεσης των επιλεγμένων αλγορίθμων μηχανικής μάθησης στο σύνολο δεδομένων PIMA. Στην συνέχεια θα γίνει σύγκριση τους και θα επιλεχθεί το μοντέλο με τις τεχνικές εκείνες (feature engineering και feature scaling) που οδηγεί στα μέγιστα δυνατά αποτελέσματα για την πρόβλεψη του διαβήτη. Τέλος θα πραγματοποιηθεί και ανάλυση πάνω στα αποτελέσματα.

6.1. Αποτελέσματα μοντέλων

Μετά την επιτυχή εκτέλεση των μοντέλων που επιλέχθηκαν έχει πραγματοποιηθεί η καταγραφή των αποτελεσμάτων τους στον πίνακα 1.

Στα πλαίσια της παρούσας εργασίας τα μοντέλα που αξιολογήθηκαν με τις διάφορες τεχνικές είναι τα ακόλουθα:

1. Λογιστική παλινδρόμηση (Logistic Regression) με κανονικοποίηση (Normalization) ή προτοτυποποίηση (Standardization) με μηχανική χαρακτηριστικών (Binning και One hot encoding) και δίχως μηχανική χαρακτηριστικών

2. Μηχανές διανυσμάτων υποστήριξης (SVM) με κανονικοποίηση (Normalization) ή προτοτυποποίηση (Standardization) με μηχανική χαρακτηριστικών (Binning και One hot encoding) και δίχως μηχανική χαρακτηριστικών

3. Τυχαία δάση (Random Forest) με κανονικοποίηση (Normalization) ή προτοτυποποίηση (Standardization) με μηχανική χαρακτηριστικών (Binning και One hot encoding) και δίχως μηχανική χαρακτηριστικών

4. Κ-Κοντινότεροι γείτονες (KNN) με κανονικοποίηση (Normalization) ή προτοτυποποίηση (Standardization) με μηχανική χαρακτηριστικών (Binning και One hot encoding) και δίχως μηχανική χαρακτηριστικών

5. Gradient Boosting με κανονικοποίηση (Normalization) ή προτοτυποποίηση (Standardization) με μηχανική χαρακτηριστικών (Binning και One hot encoding) και δίχως μηχανική χαρακτηριστικών

6. Μοντέλο Perceptron πολλών στρωμάτων (Multilayer Perceptron -MLP) με κανονικοποίηση (Normalization) ή προτοτυποποίηση (Standardization) με μηχανική χαρακτηριστικών (Binning και One hot encoding) και δίχως μηχανική χαρακτηριστικών.

Scaling & Feature Engineer	Models	Accuracy	Precision	Recall	F1 Score	AUC-ROC	AUC-PR
Normalization with Feature Engineer	Multilayer Perceptron (MLP)	79,90%	67,40%	65,90%	66,70%	87,00%	69,10%
Normalization with Feature Engineer	Random Forest	81,80%	70,20%	70,20%	70,20%	86,60%	73,80%
Normalization with Feature Engineer	Gradient Boosting	79,20%	74,10%	48,90%	58,90%	84,70%	72,70%
Normalization with Feature Engineer	K-NN	81,20%	75,00%	57,40%	65,00%	77,90%	63,30%
Normalization with Feature Engineer	SVM	81,80%	75,70%	59,60%	66,70%	86,00%	70,00%
Normalization with Feature Engineer	Logistic Regression	81,20%	78,10%	53,20%	63,30%	84,90%	67,40%
Normalization without Feature Engineer	K-NN	78,60%	68,40%	55,30%	61,20%	78,30%	65,10%
Normalization without Feature Engineer	Random Forest	82,50%	72,70%	68,00%	70,30%	85,90%	70,50%
Normalization without Feature Engineer	Gradient Boosting	80,50%	77,40%	51,00%	61,50%	85,00%	73,00%
Normalization without Feature Engineer	SVM	79,20%	77,70%	44,70%	56,80%	86,20%	69,80%
Normalization without Feature Engineer	Multilayer Perceptron (MLP)	82,50%	77,70%	59,60%	67,50%	87,70%	72,10%
Normalization without Feature Engineer	Logistic Regression	78,60%	88,90%	34,00%	49,20%	85,50%	70,00%
Standardization with Feature Engineer	K-NN	78,60%	69,40%	53,10%	60,20%	78,50%	62,00%
Standardization with Feature Engineer	Gradient Boosting	79,20%	74,10%	48,90%	58,90%	84,70%	72,60%
Standardization with Feature Engineer	Multilayer Perceptron (MLP)	82,40%	75,00%	63,80%	68,90%	84,90%	68,70%
Standardization with Feature Engineer	Logistic Regression	82,50%	75,00%	63,80%	68,90%	88,20%	72,20%
Standardization with Feature Engineer	Random Forest	85,00%	75,00%	76,50%	75,80%	87,70%	74,80%
Standardization with Feature Engineer	SVM	83,10%	75,60%	65,90%	70,40%	88,40%	72,60%
Standardization without Feature Engineer	K-NN	79,90%	70,00%	59,60%	64,40%	78,30%	63,20%
Standardization without Feature Engineer	Random Forest	82,50%	72,70%	68,00%	70,30%	85,70%	70,50%
Standardization without Feature Engineer	SVM	81,20%	73,70%	59,60%	65,90%	87,20%	71,70%
Standardization without Feature Engineer	Multilayer Perceptron (MLP)	82,50%	76,30%	61,70%	68,20%	85,20%	69,90%
Standardization without Feature Engineer	Logistic Regression	82,50%	76,30%	61,70%	68,20%	87,00%	71,60%
Standardization without Feature Engineer	Gradient Boosting	80,50%	77,40%	51,00%	61,50%	84,90%	73,00%

Πίνακας 1: Αποτελέσματα εκτέλεσης μοντέλων μηχανικής μάθησης

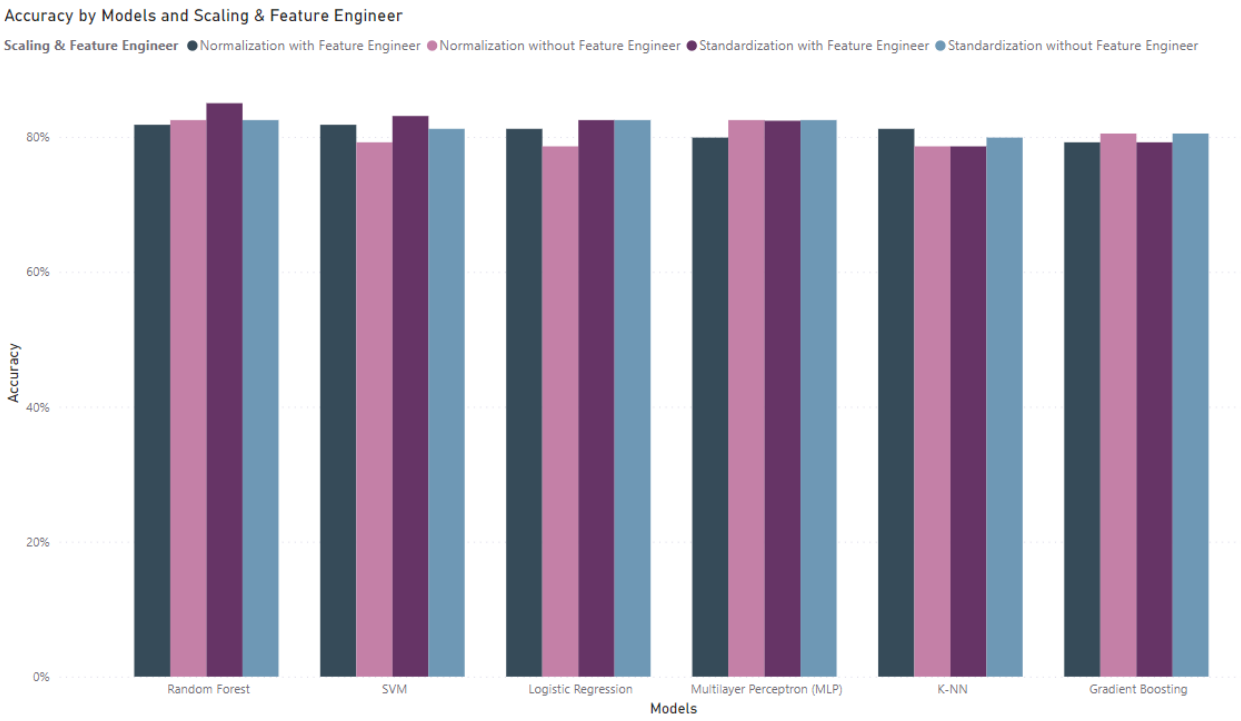
Όπως γίνεται εύκολα αντιληπτό οι τεχνικές κλιμάκωσης (Feature Scaling) και μηχανικής (Feature Engineer) των χαρακτηριστικών επιδρούν διαφορετικά ανάλογα το μοντέλο που θα επιλεχθεί.

Για κάθε μοντέλο η καλύτερη απόδοση επιτεύχθηκε με:

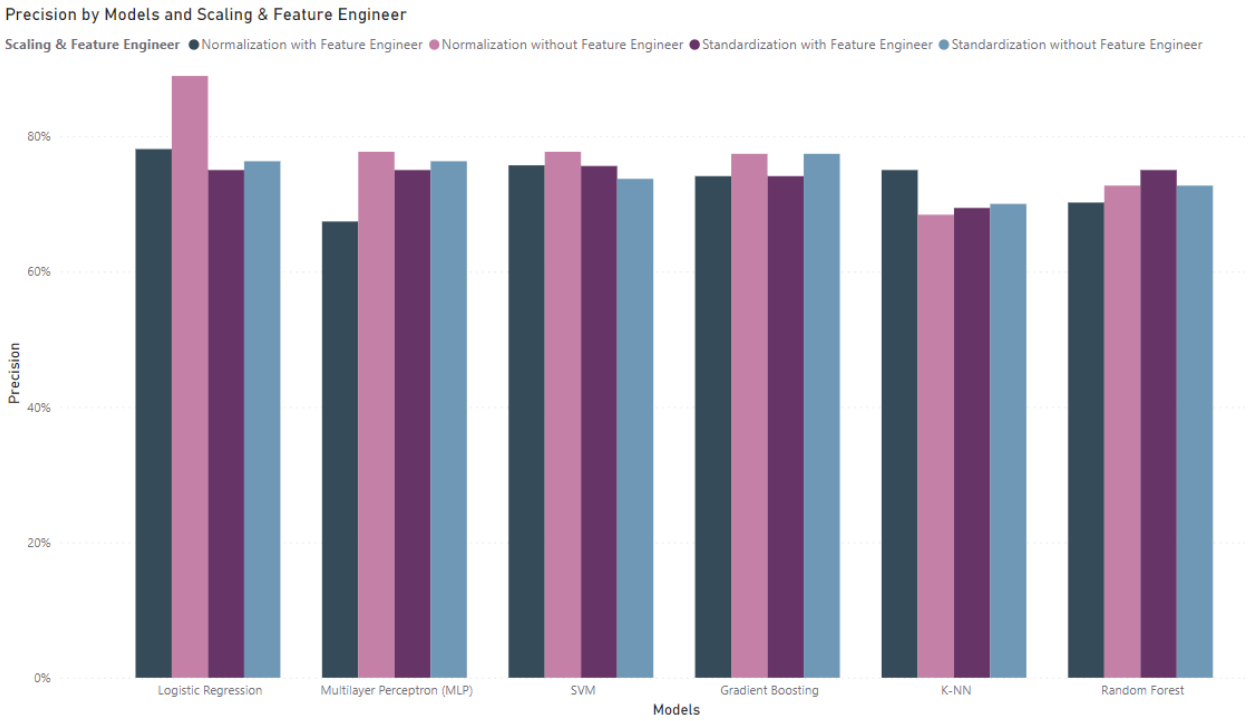
1. Λογιστική παλινδρόμηση (Logistic Regression)- Με τεχνική κλιμάκωσης προτοτυποποίηση (Standardization) και τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding).
2. Μηχανές διανυσμάτων υποστήριξης (SVM)- Με τεχνική κλιμάκωσης προτοτυποποίηση (Standardization) και τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding)

3. Τυχαία δάση (Random Forest)- Με τεχνική κλιμάκωσης προτοτυποποίηση (Standardization) και τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding)
4. Κ-Κοντινότεροι γείτονες (KNN) - Με τεχνική κλιμάκωσης κανονικοποίηση (Normalization) και τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding)
5. Gradient Boosting- Χωρίς τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding) ανεξάρτητα από την τεχνική κλιμάκωσης.
6. Μοντέλο Perceptron πολλών στρωμάτων (Multilayer Perceptron -MLP)- Χωρίς τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding) ανεξάρτητα από την τεχνική κλιμάκωσης.

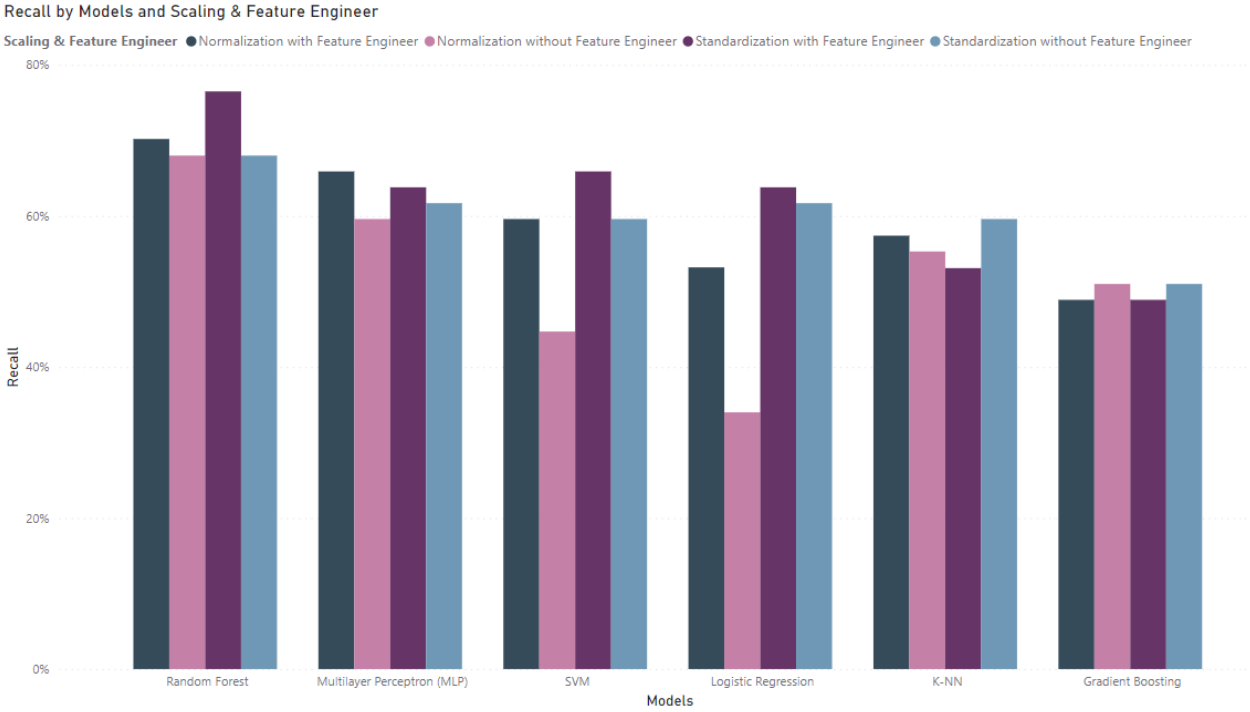
Στην συνέχεια παρουσιάζονται οι πίνακες (2-7) για κάθε μέτρο απόδοσης για όλα τα σενάρια των μοντέλων και τεχνικών προκειμένου να γίνουν εύκολα αντιληπτά τα παραπάνω αποτελέσματα.



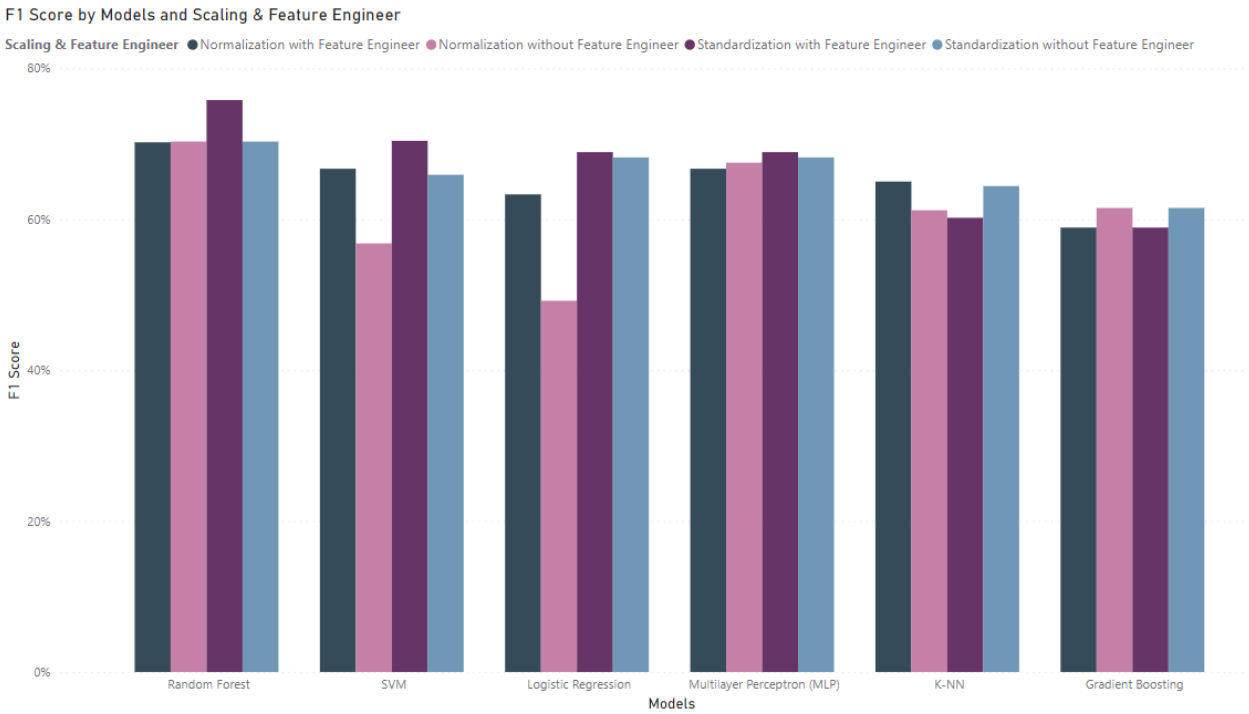
Γραφική παράσταση 11: Ορθότητα ανά μοντέλο



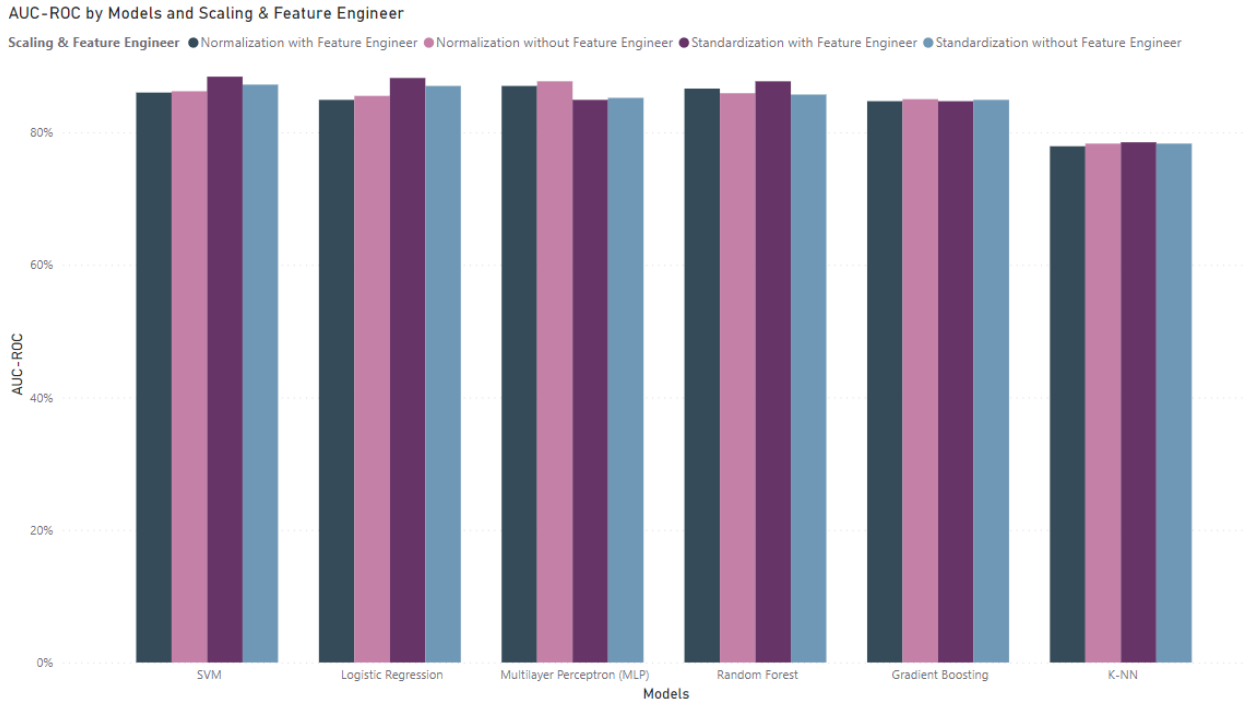
Γραφική παράσταση 12: Ακρίβεια ανά μοντέλο



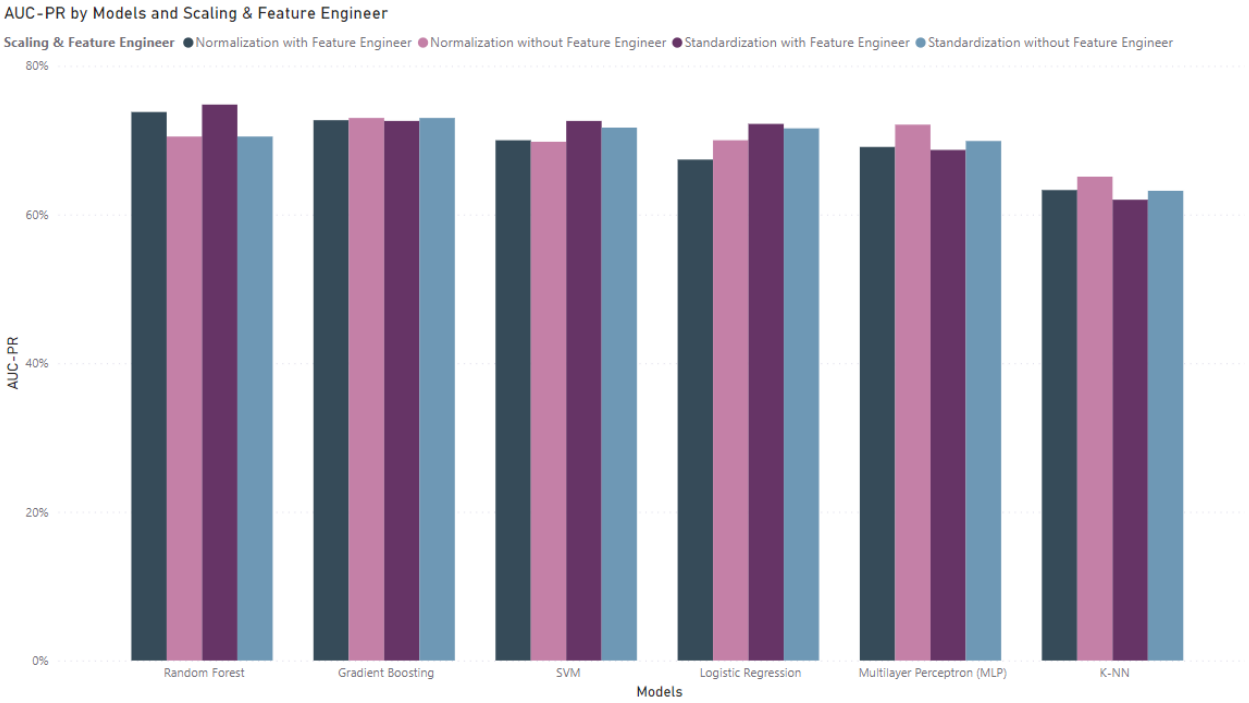
Γραφική παράσταση 13: Ανάκληση ανά μοντέλο



Γραφική παράσταση 14: Βαθμολογία F1 ανά μοντέλο



Γραφική παράσταση 15: Βαθμολογία AUC-ROC ανά μοντέλο



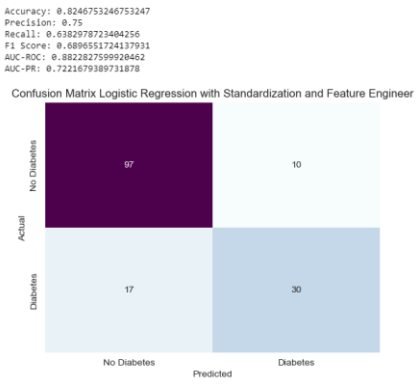
Γραφική παράσταση 16: Βαθμολογία AUC-PR ανά μοντέλο

6.2. Πίνακας σύγχυσης (Confusion Matix)

Σε αυτή την ενότητα θα γίνει παρουσίαση των πινάκων σύγχυσης των μοντέλων με τις τεχνικές εκείνες που πέτυχαν την καλύτερη απόδοση.

Επομένων ο πίνας σύγχυσης για το μοντέλο:

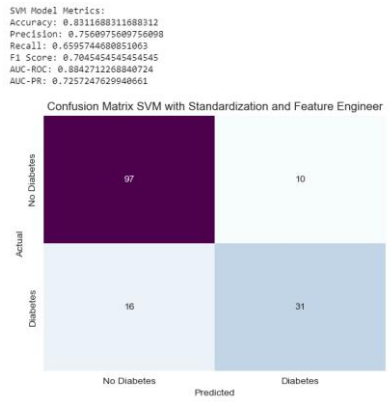
1. Λογιστική παλινδρόμηση (Logistic Regression) με τεχνική κλιμάκωσης προτοτυποποίηση (Standardization) και τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding).



Σχήμα 6: Confusion matrix LR

Από το παραπάνω έχουμε Αληθινό θετικό (True Positive-TP)= 30 , Αληθινό αρνητικό (True Negative -TN)= 97, ψευδώς θετικό (False Positive -FP)= 10 και ψευδώς αρνητικό (False Negative -FN)= 17.

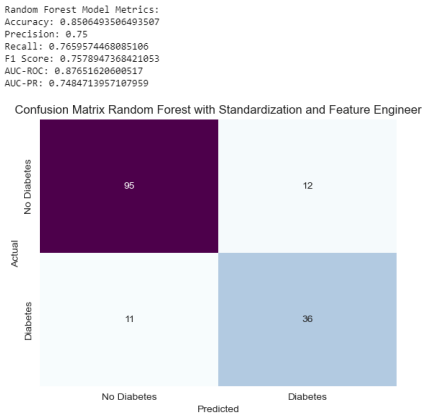
2. Μηχανές διανυσμάτων υποστήριξης (SVM) με τεχνική κλιμάκωσης προτοτυποποίηση (Standardization) και τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding)



Σχήμα 7: Confusion matrix SVM

Από το παραπάνω έχουμε Αληθινό θετικό (True Positive-TP)= 31 , Αληθινό αρνητικό (True Negative -TN)= 97, ψευδώς θετικό (False Positive -FP)= 10 και ψευδώς αρνητικό (False Negative -FN)= 16.

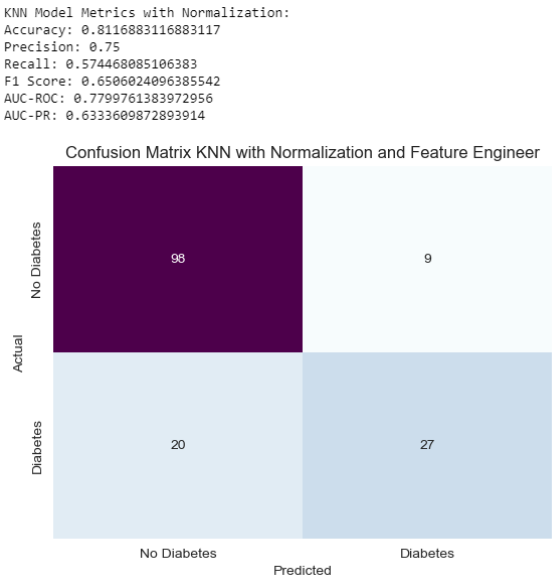
3. Τυχαία δάση (Random Forest) με τεχνική κλιμάκωσης προτοτυποποίηση (Standardization) και τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding)



Σχήμα 8: Confusion matrix RF

Από το παραπάνω έχουμε Αληθινό θετικό (True Positive-TP)= 36 , Αληθινό αρνητικό (True Negative -TN)= 95, ψευδώς θετικό (False Positive -FP)= 12και ψευδώς αρνητικό (False Negative -FN)= 11.

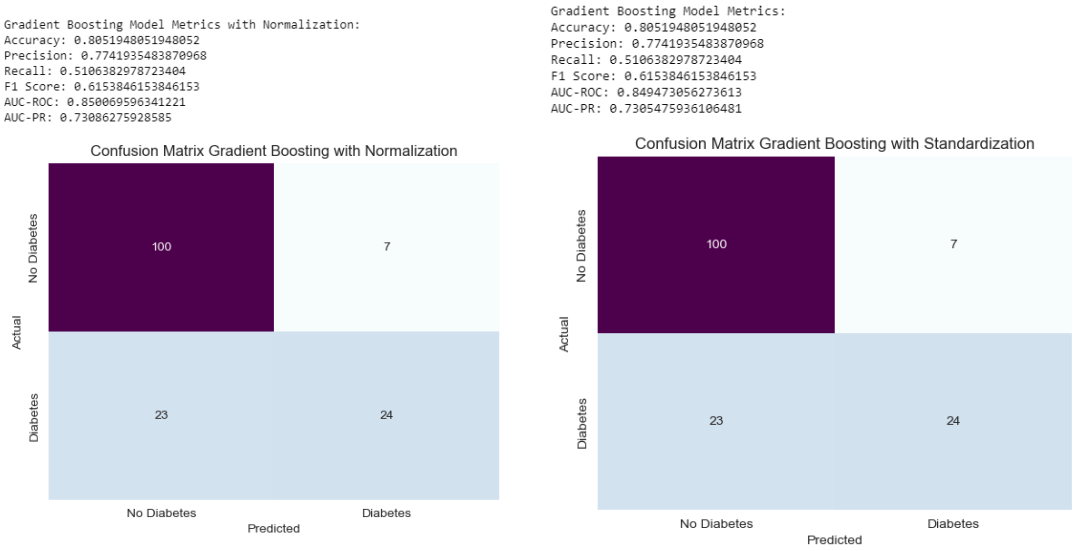
4. Κ-Κοντινότεροι γείτονες (KNN) με τεχνική κλιμάκωσης κανονικοποίηση (Normalization) και τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding)



Σχήμα 9: Confusion matrix KNN

Από το παραπάνω έχουμε Αληθινό θετικό (True Positive-TP)= 27 , Αληθινό αρνητικό (True Negative -TN)= 96, ψευδώς θετικό (False Positive -FP)= 9 και ψευδώς αρνητικό (False Negative -FN)= 20.

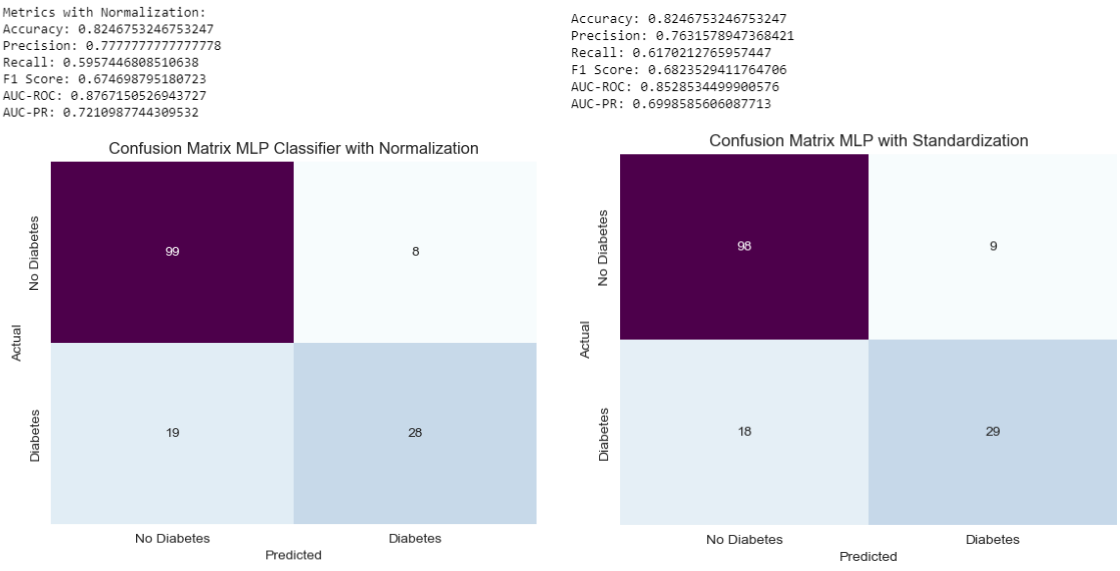
5. Gradient Boosting χωρίς τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding) ανεξάρτητα από την τεχνική κλιμάκωσης.



Σχήμα 10: Confusion matrixes GB

Από τα παραπάνω έχουμε Αληθινό θετικό (True Positive-TP)= 24 , Αληθινό αρνητικό (True Negative -TN)= 100, ψευδώς θετικό (False Positive -FP)= 7 και ψευδώς αρνητικό (False Negative -FN)= 23 αντίστοιχα και στα δυο confusion matrix.

6. Μοντέλο Perceptron πολλών στρωμάτων (Multilayer Perceptron -MLP) χωρίς τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding) ανεξάρτητα από την τεχνική κλιμάκωσης.



Σχήμα 11: Confusion matrixes MLPs

Από τα παραπάνω στην περίπτωση της κανονικοποίησης έχουμε Αληθινό θετικό (True Positive-TP)= 28 , Αληθινό αρνητικό (True Negative -TN)= 99, ψευδώς θετικό (False Positive -FP)= 8 και ψευδώς αρνητικό (False Negative -FN)= 19 και για την περίπτωση της προτοτυποποίησης έχουμε Αληθινό θετικό (True Positive-TP)= 29 , Αληθινό αρνητικό (True Negative -TN)= 98, ψευδώς θετικό (False Positive -FP)= 9 και ψευδώς αρνητικό (False Negative -FN)= 18.

6.3 Επιλογή αποδοτικότερου αλγορίθμου

Παρατηρούμε ότι για το σύνολο δεδομένων PIMA ο αποδοτικότερος αλγόριθμος σχεδόν σε όλα τα σενάρια υλοποίησης είναι το μοντέλο *Random Forest.* Όταν χρησιμοποιηθούν τεχνική κλιμάκωσης προτοτυποποίηση (Standardization) και τεχνικές μηχανικής χαρακτηριστικών (Binning και One hot encoding) το μοντέλο είναι ικανό να πετύχει 85% ορθότητα στην πρόβλεψη διαβήτη σε νέα δεδομένα.

Κεφάλαιο 7 : Συμπεράσματα και Μελλοντικές Κατευθύνσεις

Σε αυτό το κεφάλαιο συνοψίσουμε την παρούσα εργασία και θα εξάγουμε τα συμπεράσματα. Τέλος θα παρουσιαστούν οι μελλοντικές κατευθύνσεις.

7.1 Συμπεράσματα

Με βάση την εκτενή ανάλυση που διεξήχθη για την πρόβλεψη της νόσου του διαβήτη χρησιμοποιώντας διάφορες τεχνικές μηχανικής μάθησης και μετρήσεις αξιολόγησης, είναι προφανές ότι η χρήση εποπτευόμενων μεθόδων μάθησης όπως η λογιστική παλινδρόμηση, οι μηχανές διανυσμάτων υποστήριξης (SVM), το τυχαίο δάσος, οι k-πλησιέστεροι γείτονες (KNN), Τα πολυστρωματικά perceptrons (MLPs) αποδίδουν πολλά υποσχόμενα αποτελέσματα σε εργασίες δυαδικής κατηγοριοποίησης. Μέσω αυστηρής αξιολόγησης, συμπεριλαμβανομένης της αξιολόγησης της ορθότητας, της ανάκλησης, της βαθμολογίας F1 και της ακρίβειας, η απόδοση κάθε μοντέλου εξετάστηκε διεξοδικά.

Συγκεκριμένα, η αξιοποίηση του Scikit-learn στην Python επέτρεψε την αποτελεσματική υλοποίηση και σύγκριση διαφόρων μοντέλων για το σύνολο δεδομένων PIMA, λαμβάνοντας υπόψη παράγοντες όπως η κανονικοποίηση, η προτοτυποποίηση και τεχνικών μηχανικής χαρακτηριστικών (Binning και One Hot Encoding).

Συμπερασματικά μεταξύ των μοντέλων που εξετάστηκαν, το RF εμφανίστηκε ως το πιο αποτελεσματικό, παρουσιάζοντας εντυπωσιακή ακρίβεια 85% σε συνδυασμό με τεχνικές προτοτυποποίησης και μηχανικής χαρακτηριστικών, όπως binning και one-hot encoding. Αυτό δείχνει τη σημασία των μεθοδολογιών προεπεξεργασίας για τη βελτίωση της απόδοσης του μοντέλου και της ακρίβειας πρόβλεψης.

7.2 Μελλοντικές κατευθύνσεις

Στις μέρες μας η μηχανική μάθηση συνεχίζει να εξελίσσεται με ραγδαίους ρυθμούς, υπάρχουν πολλές νέες υποσχόμενες μέθοδοι για την δημιουργία πιο αποτελεσματικών προβλεπτικών μοντέλων.

Αρχιτεκτονικές Deep Learning: Τα μοντέλα βαθιάς μάθησης, συμπεριλαμβανομένων των συνελικτικών νευρωνικών δικτύων (CNN) και των επαναλαμβανόμενων νευρωνικών δικτύων (RNN), προσφέρουν ισχυρές δυνατότητες εκμάθησης πολύπλοκων μοτίβων από δεδομένα υψηλών διαστάσεων. Η διερεύνηση της εφαρμογής αρχιτεκτονικών βαθιάς μάθησης στην πρόβλεψη του διαβήτη θα μπορούσε να αποκαλύψει νέες ιδέες και να βελτιώσει την προγνωστική απόδοση.

Εξατομικευμένη ιατρική: Θα μπορούσε να παρέχεται εξατομικευμένη ιατρική σε ανθρώπους, με την προσαρμογή των προγνωστικών μοντέλων στα μεμονωμένα χαρακτηριστικά του ασθενούς και το ιατρικό ιστορικό μπορεί να επιτευχθεί βελτίωση της ακρίβεια της αξιολόγησης κινδύνου διαβήτη και να επιτρέψει εξατομικευμένες στρατηγικές θεραπείας.

Βιβλιογραφία

[1] Παναγιώτης Τσαπόγας MD «Σακχαρώδης Διαβήτης: Πλήρης Οδηγός»,2016 (Available from : Σακχαρώδης Διαβήτης: Πλήρης Οδηγός - Γ Παθολογική Κλινική Ερρίκος Ντυνάν Hospital Center (pathologia.eu))[accessed Dec 12,2023]

[2] Centers for Disease Control and Prevention « Diabetes Basics» (Available from: https://www.cdc.gov/diabetes/basics/index.html) [accessed Dec 12,2023]

[3] Health Direct «Diabetes»,2023 (Available from: https://www.healthdirect.gov.au/diabetes) [accessed Dec 12,2023]

[4] GBD 2021 Diabetes Collaborator « Global, regional, and national burden of diabetes from 1990 to 2021, with projections of prevalence to 2050: a systematic analysis for the Global Burden of Disease Study 2021»,2023 (Available from: https://www.healthdata.org/news-events/newsroom/news-releases/global-diabetes-cases-soar-529-million-13-billion-2050)[accessed Dec 13,2023]

[5] International Diabetes Federation «Facts & figures» (Available from: https://idf.org/about-diabetes/diabetes-facts-figures) [accessed Dec 13,2023]

[6] World Health Organization «Diabetes», 2023 (Available: https://www.who.int/news-room/fact-sheets/detail/diabetes) [accessed Dec 13,2023]

[7] Centers for Disease Control and Prevention «National Diabetes Statistics Report»,2023 (Available: https://www.cdc.gov/diabetes/data/statistics-report/index.html) [Accessed Dec 13,2023]

[8] Wikipedia the free encyclopedia «Diabetes»,2023 (Available from : https://en.wikipedia.org/wiki/Diabetes) [Accessed Dec 13,2023]

[9] SOS ΙΑΤΡΟΙ “ΣΑΚΧΑΡΩΔΗΣ ΔΙΑΒΗΤΗΣ Ή "ΖΑΧΑΡΟ", ΕΝΑ ΣΥΧΝΟ ΝΟΣΗΜΑ”,2023 (Available from: https://www.sosiatroi.gr/iatrikes-symvoules/pathologika/sakxarodis-diabitis/) [Accessed Dec 13,2023]

[10] Geek for geeks «Machine Learning Algorithms»,2024 (Available from: https://www.geeksforgeeks.org/machine-learning-algorithms/) [Accessed Jan 10,2024]

[11] Iqbal H. Sarker «Machine Learning: Algorithms, Real-World Applications and Research Directions», 2021 (Available from: https://link.springer.com/article/10.1007/s42979-021-00592-x) [Accessed Jan 10,2024]

[12] Afshine Amidi & Shervine Amidi (Available from: https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-supervised-learning) [Accessed Jan 10,2024]

[13] Saimadhu Polamuri «10 MOST POPULAR SUPERVISED LEARNING ALGORITHMS IN MACHINE LEARNING»,2023 (Available from: https://dataaspirant.com/supervised-learning-algorithms/) [Accessed Jan 10,2024]

[14] Geek for geeks « What is Feature Engineering? »,2024 (Available from: https://www.geeksforgeeks.org/what-is-feature-engineering/)[Accessed Feb 15,2024]

[15] Aniruddha Bhandari «Feature Scaling: Engineering, Normalization, and Standardization»,2024 (Available from: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/)[Accessed Feb 15,2024]

[16] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

[17] Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.

[18] Anshul Saini «Decision Tree – A Step-by-Step Guide» ,2024 (Available from: https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/) [Accessed Jan 15,2024]

[19] Geek for geeks «Support Vector Machine (SVM) Algorithm»,2024 (Available from: https://www.geeksforgeeks.org/support-vector-machine-algorithm/) [Accessed Jan 15,2024]

[20] Cortes, C., & Vapnik, V. (1995). «Support-vector networks. Machine learning», 20(3), 273-297.

[21] Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). «Deep Learning. MIT Press».

[22] Cover, T., & Hart, P. (1967). «Nearest neighbor pattern classification» IEEE transactions on information theory, 13(1), 21-27.

[23] Anshul Saini «An Introduction to Random Forest Algorithm for beginners»,2022 (Available from: https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/) [Accessed Jan 20,2024]

[24] Marijn Speeckaert «Application of Machine Learning Models for Early Detection and Accurate Classification of Type 2 Diabetes»,2023 (Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10378239/) [Accessed Jan 20,2024]

[25] Geek for geeks «Data Mining Techniques»,2024 (Available from: https://www.geeksforgeeks.org/data-mining-techniques/) [Accessed Jan 25,2024]

[26] LarryAlton « The 7 Most Important Data Mining Techniques»,2017 https://www.datasciencecentral.com/the-7-most-important-data-mining-techniques/) [Accessed Jan 25,2024]

[27] Ammar Ali «Top 10 Data Mining Techniques»,2023 (Available from: https://www.astera.com/type/blog/top-10-data-mining-techniques/) [Accessed Jan 25,2024]

[28] Neri Van Otten «Regression Vs Classification — Understand How To Choose And Switch Between Them»,2023 (Available from: https://spotintelligence.com/2023/05/02/regression-vs-classification/) [Accessed Jan 28,2024]

[29] Avcontentteam «Regression vs Classification in Machine Learning Explained!»,2023 (Available from: https://www.analyticsvidhya.com/blog/2023/05/regression-vs-classification/) [Accessed Jan 28,2024]

[30] Geek for geeks «Evaluation Metrics in Machine Learning»,2024 (Available from: https://www.geeksforgeeks.org/metrics-for-machine-learning-model/) [Accessed Jan 28,2024]

[31] Geek for geeks «A Tour of Evaluation Metrics for Machine Learning»,2020 (Available from: https://www.analyticsvidhya.com/blog/2020/11/a-tour-of-evaluation-metrics-for-machine-learning/) [Accessed Jan 28,2024]

[32] KM Jyoti Rani « Diabetes Prediction Using Machine Learning »,2020 (Available from: https://ijsrcseit.com/CSEIT206463) [Accessed Jan 28,2024]

[33] Md. Maniruzzaman, Md. Jahanur Rahman, Benojir Ahammed & Md. Menhazul Abedin « Classification and prediction of diabetes disease using machine learning paradigm »,2020 (Available from: https://link.springer.com/article/10.1007/s13755-019-0095-z)[Accessed Feb 5,2024]

[34] Ioannis Kavakiotis, Olga Tsave,c Athanasios Salifoglou,c Nicos Maglaveras,b,d Ioannis Vlahavas,a and Ioanna Chouvarda « Machine Learning and Data Mining Methods in Diabetes Research »,2020 (Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5257026/)[Accessed Feb 5,2024]

[35] Deepti Sisodia, Dilip Singh Sisodia « Prediction of Diabetes using Classification Algorithms »,2018 (Available from: https://www.sciencedirect.com/science/article/pii/S1877050918308548)[Accessed Feb 5,2024]

[36] Jobeda Jamal Khanam, Simon Y. Foo « A comparison of machine learning algorithms for diabetes prediction»,2021(Available from: https://www.sciencedirect.com/science/article/pii/S2405959521000205)[Accessed Feb 5,2024]

[37] Huma Naz & Sachin Ahuja « Deep learning approach for diabetes prediction using PIMA Indian dataset »,2020 (Available from: https://link.springer.com/article/10.1007/s40200-020-00520-5)[Accessed Feb 5,2024]

[38] Geek for geeks « What is Python? »,2024 (Available from: https://www.geeksforgeeks.org/what-is-python/)[Accessed Feb 10,2024]

[39] «»,2020 (Available from: https://www.analyticsvidhya.com/blog/2021/04/top-15-python-libraries-you-must-know-for-data-science-in-2021/)[Accessed Feb 10,2024]

[40] Python Data Analytics: With Pandas, NumPy, and Matplotlib , Fabio Nelli, 2018.

[41] Akshay Gupta « Top 15 Python Libraries you must know for Data Science in 2021 »,2021 (Available from: https://www.geeksforgeeks.org/python-visualize-missing-values-nan-values-using-missingno-library/)[Accessed Feb 10,2024]

[42] Introduction to Machine Learning with Python: A Guide for Data Scientists by Andreas C. Müller & Sarah Guido.

[43] Naz H , Ahuja S « Deep learning approach for diabetes prediction using PIMA Indian dataset »,2020 (Available from: https://europepmc.org/article/PMC/PMC7270283)[Accessed Feb 10,2024]

[44] RDocumentation «pima: Pima Indians Diabetes Data »,2024 (Available from: https://rdocumentation.org/packages/pdp/versions/0.8.1/topics/pima)[Accessed Feb 10,2024]

[45] Tim Bock «What is a Correlation Matrix?»,2024 (Available from: https://www.displayr.com/what-is-a-correlation-matrix/)[Accessed Feb 10,2024]

[46] Rahul Shah «Tune Hyperparameters with GridSearchCV»,2023 (Available from: https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/)[Accessed Feb 15,2024]

[47] Sarang Narkhede «Understanding AUC - ROC Curve»,2020 (Available from: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5)[Accessed Feb 15,2024]

Παράρτημα Α. Κώδικας

Κώδικας Α περίπτωσης Normalization & Feature Engineering (Binning και One Hot Encoding).

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import missingno as msno

from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.neighbors import LocalOutlierFactor
from sklearn.preprocessing import MinMaxScaler, LabelEncoder, StandardScaler, RobustScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, precision_score, recall_score,
f1_score, roc_auc_score, average_precision_score
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn import datasets
from sklearn.neural_network import MLPClassifier
```

```
#Load the pima dataset from my folder

def load_data():

   data = pd.read_csv('C:/Users/geras/OneDrive/Έγγραφα/Μεταπτυχιακό/Εξάμηνο 3 - Διπλωματική Εργασία/diabetes.csv')

   return data


df = load_data()

df.head()


df.isnull().sum()

df.info()

df.shape

df.columns

correlation_matrix = df.corr()

#Correlation Matric using Heatmap

plt.figure(figsize=(10, 8))

sns.heatmap(correlation_matrix, annot=True, cmap='BuPu', fmt='.2f')

plt.title('Correlation Matrix')

plt.show()

#Pairplot for some selected features

selected_features = ["SkinThickness","Insulin","Pregnancies", "Glucose", "BMI", "Age", "Outcome"]

sns.pairplot(df[selected_features], hue="Outcome")

plt.suptitle("Pairplots")

plt.show()

#subplots for boxplots in order to see the outliers

plt.figure(figsize=(14, 10))

sns.set_style(style='whitegrid')



plt.subplot(2, 3, 1)

sns.boxplot(x='Glucose',data=df)
```

```
plt.subplot(2, 3, 2)
sns.boxplot(x='BloodPressure', data=df)

plt.subplot(2, 3, 3)
sns.boxplot(x='Insulin', data=df)

plt.subplot(2, 3, 4)
sns.boxplot(x='BMI', data=df)

plt.subplot(2, 3, 5)
sns.boxplot(x='Age', data=df)

plt.subplot(2, 3, 6)
sns.boxplot(x='SkinThickness', data=df)

plt.show()
#plot the Distribution of the Outcome
sns.countplot(x='Outcome', data=df)
plt.title('Distribution of the Outcome')
plt.show()
outcome_counts = df['Outcome'].value_counts()
print(outcome_counts)
#Plot the distribution of numerical variables
numerical_columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',
'DiabetesPedigreeFunction', 'Age']

for column in numerical_columns:
    sns.histplot(df[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.show()
```

```
numeric_columns = df.select_dtypes(include=['int64', 'float64']).columns

#Plotting histograms for each variable with both outcomes in the same diagram
fig, axes = plt.subplots(nrows=len(numeric_columns), ncols=1, figsize=(10, 20))

for i, column in enumerate(numeric_columns):
    ax = axes[i]
    for outcome in [0, 1]:
        ax.hist(df[df['Outcome'] == outcome][column], alpha=0.5, label=f'Outcome {outcome}')
    ax.set_title(column)
    ax.legend()

plt.tight_layout()
plt.show()
#Handling outliers using the IQR (Interquartile Range) method
def handle_outliers_iqr(series):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    series = series.apply(lambda x: lower_bound if x < lower_bound else (upper_bound if x > upper_bound else x))
    return series

#Apply the IQR
df['Glucose'] = handle_outliers_iqr(df['Glucose'])
df['BloodPressure'] = handle_outliers_iqr(df['BloodPressure'])
df['SkinThickness'] = handle_outliers_iqr(df['SkinThickness'])
df['Insulin'] = handle_outliers_iqr(df['Insulin'])
df['BMI'] = handle_outliers_iqr(df['BMI'])
#Box plots in order to visualize the data without outliers
```

```
plt.figure(figsize=(16, 6))
sns.boxplot(data= df)
plt.title('Box Plot of Pima Dataset Features')
plt.show()
df.describe().T
#Replece zeros with the mean of the features
df['Glucose'].replace(0, df['Glucose'].mean(), inplace=True)
df['BloodPressure'].replace(0, df['BloodPressure'].mean(), inplace=True)
df['SkinThickness'].replace(0, df['SkinThickness'].mean(), inplace=True)
df['Insulin'].replace(0, df['Insulin'].mean(), inplace=True)
df['BMI'].replace(0, df['BMI'].mean(), inplace=True)
df.describe().T

##Logistic Regression with Min-Max Scaling
#Features for (X) and target variable (y)
X = df.drop('Outcome', axis=1)
#X=df[['Glucose','BloodPressure','BMI']] test Feature importance
y = df['Outcome']



#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=0)

#Scaling with Normalization (Min-Max Scaling)
#Initialize the MinMaxScaler
scaler = MinMaxScaler()

X_train_normalized = scaler.fit_transform(X_train)
X_test_normalized = scaler.transform(X_test)

#Initialize the Logistic Regression model using the best hyperparameters using GridSearchCV
```

```
model =
LogisticRegression(C=0.1,penalty='l2',random_state=0,solver='lbfgs',max_iter=100,fit_intercept=True,tol=1e-
4,class_weight=None,dual=False,verbose=0,warm_start=False)


#Train on the normalized training set

model.fit(X_train_normalized, y_train)


#Predictions on the normalized test set

y_pred_normalized = model.predict(X_test_normalized)


#Evaluation of the model with normalization

accuracy_normalized = accuracy_score(y_test, y_pred_normalized)

precision_normalized = precision_score(y_test, y_pred_normalized)

recall_normalized = recall_score(y_test, y_pred_normalized)

f1_normalized = f1_score(y_test, y_pred_normalized)

roc_auc_normalized = roc_auc_score(y_test, model.predict_proba(X_test_normalized)[:, 1])

pr_auc_normalized = average_precision_score(y_test, model.predict_proba(X_test_normalized)[:, 1])

conf_matrix_normalized = confusion_matrix(y_test, y_pred_normalized)


#Print metrics

print("Metrics with Normalization:")

print(f"Accuracy: {accuracy_normalized}")

print(f"Precision: {precision_normalized}")

print(f"Recall: {recall_normalized}")

print(f"F1 Score: {f1_normalized}")

print(f"AUC-ROC: {roc_auc_normalized}")

print(f"AUC-PR: {pr_auc_normalized}")


#Plot the confusion matrix for the model

sns.heatmap(conf_matrix_normalized, annot=True, fmt="d", cmap="BuPu", cbar=False,

        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
```

```
plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix Logistic Regression with Normalization")

plt.show()


##SVM with Min-Max Scaling

#Features for (X) and target variable (y)

X = df.drop("Outcome", axis=1)

#X=df[['Glucose','BloodPressure','BMI']] test Feature importance

y = df["Outcome"]


#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=0)


#Scaling with Normalization (Min-Max Scaling)

#Initialize the MinMaxScaler

scaler = MinMaxScaler()

X_train_normalized = scaler.fit_transform(X_train)

X_test_normalized = scaler.transform(X_test)


#Initialize the SVM model using the best hyperparameters using GridSearchCV

model_svm = SVC(kernel='linear',C=0.1,gamma='scale',random_state=0,probability=False,shrinking=True,tol=1e-
3,cache_size=200,class_weight=None,verbose=False,max_iter=-1,decision_function_shape='ovr', break_ties=False)


#Train on the normalized training set

model_svm.fit(X_train_normalized, y_train)


#Predictions on the normalized test set

y_pred_svm = model_svm.predict(X_test_normalized)


#Evaluation of the model with normalization
```

```
accuracy_svm = accuracy_score(y_test, y_pred_svm)
precision_svm = precision_score(y_test, y_pred_svm)
recall_svm = recall_score(y_test, y_pred_svm)
f1_svm = f1_score(y_test, y_pred_svm)
roc_auc_svm = roc_auc_score(y_test, model_svm.decision_function(X_test_normalized))
pr_auc_svm = average_precision_score(y_test, model_svm.decision_function(X_test_normalized))
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)

#Print metrics
print("SVM Model Metrics with Normalization:")
print(f"Accuracy: {accuracy_svm}")
print(f"Precision: {precision_svm}")
print(f"Recall: {recall_svm}")
print(f"F1 Score: {f1_svm}")
print(f"AUC-ROC: {roc_auc_svm}")
print(f"AUC-PR: {pr_auc_svm}")

#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_svm, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix SVM with Normalization")
plt.show()

##Random Forest with Min-Max Scaling
#Features for (X) and target variable (y)
X = df.drop("Outcome", axis=1)
#X=df[['Glucose','BloodPressure','BMI']] test Feature importance
y = df["Outcome"]
```

```
#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=0)


#Scaling with Normalization (Min-Max Scaling)

#Initialize the MinMaxScaler

scaler = MinMaxScaler()

X_train_normalized = scaler.fit_transform(X_train)

X_test_normalized = scaler.transform(X_test)


#Initialize the Random Forest model using the best hyperparameters using GridSearchCV

rf_model_normalized =
RandomForestClassifier(n_estimators=100,max_depth=10,min_samples_split=5,min_samples_leaf=2,random_stat
e=0,bootstrap=True,criterion='gini',max_features=None,max_leaf_nodes=None,min_impurity_decrease=0.0,min_
weight_fraction_leaf=0.0,n_jobs=None,oob_score=False,verbose=0,warm_start=False,class_weight=None,ccp_alph
a=0.0,max_samples=None)


#Train on the normalized training set

rf_model_normalized.fit(X_train_normalized, y_train)


#Predictions on the normalized test set

y_pred_rf_normalized = rf_model_normalized.predict(X_test_normalized)


#Evaluation of the model with normalization

accuracy_rf_normalized = accuracy_score(y_test, y_pred_rf_normalized)

precision_rf_normalized = precision_score(y_test, y_pred_rf_normalized)

recall_rf_normalized = recall_score(y_test, y_pred_rf_normalized)

f1_rf_normalized = f1_score(y_test, y_pred_rf_normalized)

roc_auc_rf_normalized = roc_auc_score(y_test, rf_model_normalized.predict_proba(X_test_normalized)[:, 1])

pr_auc_rf_normalized = average_precision_score(y_test,
rf_model_normalized.predict_proba(X_test_normalized)[:, 1])

conf_matrix_rf_normalized = confusion_matrix(y_test, y_pred_rf_normalized)
```

```
#Print metrics
print("Random Forest Model Metrics with Normalization:")
print(f"Accuracy: {accuracy_rf_normalized}")
print(f"Precision: {precision_rf_normalized}")
print(f"Recall: {recall_rf_normalized}")
print(f"F1 Score: {f1_rf_normalized}")
print(f"AUC-ROC: {roc_auc_rf_normalized}")
print(f"AUC-PR: {pr_auc_rf_normalized}")

#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_rf_normalized, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix Random Forest with Normalization")
plt.show()

##K-nn with Min-Max Scaling
#Features for (X) and target variable (y)
X = df.drop("Outcome", axis=1)
#X=df[['Glucose','BloodPressure','BMI']] test Feature importance
y = df["Outcome"]

#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=0)

#Scaling with Normalization (Min-Max Scaling)
#Initialize the MinMaxScaler
scaler = MinMaxScaler()
X_train_normalized = scaler.fit_transform(X_train)
```

```
X_test_normalized = scaler.transform(X_test)



#Initialize the KNN Classifier using the best hyperparameters using GridSearchCV

knn_model_normalized = KNeighborsClassifier(n_neighbors=7,
p=1,weights='distance',algorithm='auto',leaf_size=30,metric='minkowski', metric_params=None,n_jobs=None)


#Train on the normalized training set

knn_model_normalized.fit(X_train_normalized, y_train)


#Predictions on the normalized test set

y_pred_knn_normalized = knn_model_normalized.predict(X_test_normalized)


#Evaluation of the model with normalization

accuracy_knn_normalized = accuracy_score(y_test, y_pred_knn_normalized)

precision_knn_normalized = precision_score(y_test, y_pred_knn_normalized)

recall_knn_normalized = recall_score(y_test, y_pred_knn_normalized)

f1_knn_normalized = f1_score(y_test, y_pred_knn_normalized)

roc_auc_knn_normalized = roc_auc_score(y_test, knn_model_normalized.predict_proba(X_test_normalized)[:, 1])

pr_auc_knn_normalized = average_precision_score(y_test,
knn_model_normalized.predict_proba(X_test_normalized)[:, 1])

conf_matrix_knn_normalized = confusion_matrix(y_test, y_pred_knn_normalized)


#Print metrics

print("KNN Model Metrics with Normalization:")

print(f"Accuracy: {accuracy_knn_normalized}")

print(f"Precision: {precision_knn_normalized}")

print(f"Recall: {recall_knn_normalized}")

print(f"F1 Score: {f1_knn_normalized}")

print(f"AUC-ROC: {roc_auc_knn_normalized}")

print(f"AUC-PR: {pr_auc_knn_normalized}")
```

```
#Plot the confusion matrix for the Classifier
sns.heatmap(conf_matrix_knn_normalized, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix KNN with Normalization")
plt.show()

##Gradient Boosting with Min-Max Scaling
#Features for (X) and target variable (y)
X = df.drop("Outcome", axis=1)
#X=df[['Glucose','BloodPressure','BMI']] test Feature importance
y = df["Outcome"]



#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=0)

#Scaling with Normalization (Min-Max Scaling)
#Initialize the MinMaxScaler
scaler = MinMaxScaler()
X_train_normalized = scaler.fit_transform(X_train)
X_test_normalized = scaler.transform(X_test)

#Initialize the GradientBoostingClassifier using the best hyperparameters using GridSearchCV
gb_model_normalized =
GradientBoostingClassifier(learning_rate=0.01,n_estimators=100,max_depth=3,min_samples_split=2,min_samples
_leaf=4,random_state=0,loss='log_loss',criterion='friedman_mse',min_weight_fraction_leaf=0.0,subsample=1.0,m
ax_features=None,max_leaf_nodes=None,validation_fraction=0.1,n_iter_no_change=None,tol=0.0001,ccp_alpha=
0.0,init=None,verbose=0,warm_start=False)
```

```
#Train on the normalized training set
gb_model_normalized.fit(X_train_normalized, y_train)

#Predictions on the normalized test set
y_pred_gb_normalized = gb_model_normalized.predict(X_test_normalized)

#Evaluation of the model with normalization
accuracy_gb_normalized = accuracy_score(y_test, y_pred_gb_normalized)
precision_gb_normalized = precision_score(y_test, y_pred_gb_normalized)
recall_gb_normalized = recall_score(y_test, y_pred_gb_normalized)
f1_gb_normalized = f1_score(y_test, y_pred_gb_normalized)
roc_auc_gb_normalized = roc_auc_score(y_test, gb_model_normalized.decision_function(X_test_normalized))
pr_auc_gb_normalized = average_precision_score(y_test,
gb_model_normalized.decision_function(X_test_normalized))
conf_matrix_gb_normalized = confusion_matrix(y_test, y_pred_gb_normalized)

#Print metrics
print("Gradient Boosting Model Metrics with Normalization:")
print(f"Accuracy: {accuracy_gb_normalized}")
print(f"Precision: {precision_gb_normalized}")
print(f"Recall: {recall_gb_normalized}")
print(f"F1 Score: {f1_gb_normalized}")
print(f"AUC-ROC: {roc_auc_gb_normalized}")
print(f"AUC-PR: {pr_auc_gb_normalized}")

#Plot the confusion matrix for the Classifier
sns.heatmap(conf_matrix_gb_normalized, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix Gradient Boosting with Normalization")
```

```
plt.show()


##Multilayer Perceptron (MLP) with Min-Max Scaling

#Features for (X) and target variable (y)

X = df.drop('Outcome', axis=1)

y = df['Outcome']


#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=0)


#Scaling with Normalization (Min-Max Scaling)

#Initialize the MinMaxScaler

scaler = MinMaxScaler()

X_train_normalized = scaler.fit_transform(X_train)

X_test_normalized = scaler.transform(X_test)


#Initialize the MLPClassifier model using the best hyperparameters using GridSearchCV

mlp =
MLPClassifier(alpha=0.001,hidden_layer_sizes=(50,),max_iter=1000,random_state=0,activation='relu',solver='adam
',batch_size='auto',learning_rate='constant',learning_rate_init=0.001,power_t=0.5,momentum=0.9,nesterovs_mo
mentum=True,early_stopping=False,validation_fraction=0.1,beta_1=0.9,beta_2=0.999,epsilon=1e-
08,n_iter_no_change=10)


#Train on the normalized training set

mlp.fit(X_train_normalized, y_train)


#Predictions on the normalized test set

y_pred_normalized = mlp.predict(X_test_normalized)


#Evaluation of the model with normalization

accuracy_normalized = accuracy_score(y_test, y_pred_normalized)

precision_normalized = precision_score(y_test, y_pred_normalized)
```

```
recall_normalized = recall_score(y_test, y_pred_normalized)
f1_normalized = f1_score(y_test, y_pred_normalized)
roc_auc_normalized = roc_auc_score(y_test, mlp.predict_proba(X_test_normalized)[:, 1])
pr_auc_normalized = average_precision_score(y_test, mlp.predict_proba(X_test_normalized)[:, 1])
conf_matrix_normalized = confusion_matrix(y_test, y_pred_normalized)

#Print metrics
print("Metrics with Normalization:")
print(f"Accuracy: {accuracy_normalized}")
print(f"Precision: {precision_normalized}")
print(f"Recall: {recall_normalized}")
print(f"F1 Score: {f1_normalized}")
print(f"AUC-ROC: {roc_auc_normalized}")
print(f"AUC-PR: {pr_auc_normalized}")

#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_normalized, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix MLP Classifier with Normalization")
plt.show()

#Feature Engineering
#Create Bins for BMI and Age because we can capture non-linear relationships with the target variable
bmi_bins = [0, 18.5, 24.9, 29.9, 100]  # BMI categories
bmi_labels = ['Very Underweight', 'Normal Weight', 'Overweight', 'Obese']
df['BMI_Category'] = pd.cut(df['BMI'], bins=bmi_bins, labels=bmi_labels, right=False)

age_bins = [-1, 30, 50, float('inf')]  # Age groups
age_labels = ['Youth', 'Adults', 'Elderly']
```

```
df['Age_Group'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels)

#One hot encoding in order to convert the two categorical variables into numerical
def one_hot_encoder(df, categorical_cols, drop_first=True):
    df = pd.get_dummies(df, columns=categorical_cols, drop_first=drop_first)
    return df

df_one_hot_encoding = one_hot_encoder(df, categorical_cols=['BMI_Category' , 'Age_Group'])
##Logistic Regression with Min-Max Scaling and Feature Engineering (Create Bins and One Hot Encoding)
#Features for (X) and target variable (y) for new df
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]



#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=0)

#Scaling with Normalization (Min-Max Scaling)
#Initialize the MinMaxScaler
scaler = MinMaxScaler()
X_train_normalized = scaler.fit_transform(X_train)
X_test_normalized = scaler.transform(X_test)

# Initialize the Logistic Regression model using the best hyperparameters using GridSearchCV
model_normalized = LogisticRegression(C=0.1, penalty='l2', random_state=0)

#Train on the normalized training set
model_normalized.fit(X_train_normalized, y_train)

#Predictions on the normalized test set
y_pred_normalized = model_normalized.predict(X_test_normalized)
```

```
#Evaluation of the model with normalization
accuracy_normalized = accuracy_score(y_test, y_pred_normalized)
precision_normalized = precision_score(y_test, y_pred_normalized)
recall_normalized = recall_score(y_test, y_pred_normalized)
f1_normalized = f1_score(y_test, y_pred_normalized)
roc_auc_normalized = roc_auc_score(y_test, model_normalized.predict_proba(X_test_normalized)[:, 1])
pr_auc_normalized = average_precision_score(y_test, model_normalized.predict_proba(X_test_normalized)[:, 1])

conf_matrix_normalized = confusion_matrix(y_test, y_pred_normalized)

#Print metrics
print("Metrics with Normalization:")
print(f"Accuracy: {accuracy_normalized}")
print(f"Precision: {precision_normalized}")
print(f"Recall: {recall_normalized}")
print(f"F1 Score: {f1_normalized}")
print(f"AUC-ROC: {roc_auc_normalized}")
print(f"AUC-PR: {pr_auc_normalized}")

#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_normalized, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix Logistic Regression with Normalization and Feature Engineer")
plt.show()
##SVM with Min-Max Scaling and Feature Engineering (Create Bins and One Hot Encoding)
#Features for (X) and target variable (y) for new df
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]
```

```
#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=0)

#Scaling with Normalization (Min-Max Scaling)
#Initialize the MinMaxScaler
scaler = MinMaxScaler()
X_train_normalized = scaler.fit_transform(X_train)
X_test_normalized = scaler.transform(X_test)

#Initialize the SVM model using the best hyperparameters using GridSearchCV
model_svm_normalized = SVC(kernel='linear', C=0.1, gamma='scale', random_state=0)

#Train on the normalized training set
model_svm_normalized.fit(X_train_normalized, y_train)

#Predictions on the normalized test set
y_pred_svm = model_svm_normalized.predict(X_test_normalized)

#Evaluation of the model with normalization
accuracy_svm = accuracy_score(y_test, y_pred_svm)
precision_svm = precision_score(y_test, y_pred_svm)
recall_svm = recall_score(y_test, y_pred_svm)
f1_svm = f1_score(y_test, y_pred_svm)
roc_auc_svm = roc_auc_score(y_test, model_svm_normalized.decision_function(X_test_normalized))
pr_auc_svm = average_precision_score(y_test, model_svm_normalized.decision_function(X_test_normalized))
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)

#Print metrics
print("SVM Model Metrics with Normalization:")
```

```
print(f"Accuracy: {accuracy_svm}")
print(f"Precision: {precision_svm}")
print(f"Recall: {recall_svm}")
print(f"F1 Score: {f1_svm}")
print(f"AUC-ROC: {roc_auc_svm}")
print(f"AUC-PR: {pr_auc_svm}")

#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_svm, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix SVM with Normalization and Feature Engineer")
plt.show()

##Random Forest with Min-Max Scaling and Feature Engineering (Create Bins and One Hot Encoding)
#Features for (X) and target variable (y) for new df
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]

#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=0)

#Scaling with Normalization (Min-Max Scaling)
#Initialize the MinMaxScaler
scaler = MinMaxScaler()
X_train_normalized = scaler.fit_transform(X_train)
X_test_normalized = scaler.transform(X_test)

#Initialize the Random Forest model using the best hyperparameters using GridSearchCV
rf_model_normalized = RandomForestClassifier(
```

```
    n_estimators=100,
    max_depth=10,
    min_samples_split=5,
    min_samples_leaf=2,
    random_state=0
)

#Train on the normalized training set
rf_model_normalized.fit(X_train_normalized, y_train)

#Predictions on the normalized test set
y_pred_rf_normalized = rf_model_normalized.predict(X_test_normalized)

#Evaluation of the model with normalization
accuracy_rf_normalized = accuracy_score(y_test, y_pred_rf_normalized)
precision_rf_normalized = precision_score(y_test, y_pred_rf_normalized)
recall_rf_normalized = recall_score(y_test, y_pred_rf_normalized)
f1_rf_normalized = f1_score(y_test, y_pred_rf_normalized)
roc_auc_rf_normalized = roc_auc_score(y_test, rf_model_normalized.predict_proba(X_test_normalized)[:, 1])
pr_auc_rf_normalized = average_precision_score(y_test,
rf_model_normalized.predict_proba(X_test_normalized)[:, 1])
conf_matrix_rf_normalized = confusion_matrix(y_test, y_pred_rf_normalized)

#Print metrics
print("Random Forest Model Metrics with Normalization:")
print(f"Accuracy: {accuracy_rf_normalized}")
print(f"Precision: {precision_rf_normalized}")
print(f"Recall: {recall_rf_normalized}")
print(f"F1 Score: {f1_rf_normalized}")
print(f"AUC-ROC: {roc_auc_rf_normalized}")
print(f"AUC-PR: {pr_auc_rf_normalized}")
```

```
#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_rf_normalized, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix Random Forest with Normalization and Feature Engineer")
plt.show()
##K-nn with Min-Max Scaling and Feature Engineering (Create Bins and One Hot Encoding)
#Features for (X) and target variable (y) for new df
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]


#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=0)

#Scaling with Normalization (Min-Max Scaling)
#Initialize the MinMaxScaler
scaler = MinMaxScaler()
X_train_normalized = scaler.fit_transform(X_train)
X_test_normalized = scaler.transform(X_test)

#Initialize the KNN Classifier using the best hyperparameters using GridSearchCV
knn_model = KNeighborsClassifier(n_neighbors=7, p=1, weights='distance')

#Train on the normalized training set
knn_model_normalized.fit(X_train_normalized, y_train)

#Predictions on the normalized test set
y_pred_knn_normalized = knn_model_normalized.predict(X_test_normalized)
```

```
#Evaluation of the model with normalization
accuracy_knn_normalized = accuracy_score(y_test, y_pred_knn_normalized)
precision_knn_normalized = precision_score(y_test, y_pred_knn_normalized)
recall_knn_normalized = recall_score(y_test, y_pred_knn_normalized)
f1_knn_normalized = f1_score(y_test, y_pred_knn_normalized)
roc_auc_knn_normalized = roc_auc_score(y_test, knn_model_normalized.predict_proba(X_test_normalized)[:, 1])
pr_auc_knn_normalized = average_precision_score(y_test,
knn_model_normalized.predict_proba(X_test_normalized)[:, 1])
conf_matrix_knn_normalized = confusion_matrix(y_test, y_pred_knn_normalized)

#Print metrics
print("KNN Model Metrics with Normalization:")
print(f"Accuracy: {accuracy_knn_normalized}")
print(f"Precision: {precision_knn_normalized}")
print(f"Recall: {recall_knn_normalized}")
print(f"F1 Score: {f1_knn_normalized}")
print(f"AUC-ROC: {roc_auc_knn_normalized}")
print(f"AUC-PR: {pr_auc_knn_normalized}")

#Plot the confusion matrix for the Classifier
sns.heatmap(conf_matrix_knn_normalized, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix KNN with Normalization and Feature Engineer")
plt.show()
##Gradient Boosting with Min-Max Scaling and Feature Engineering (Create Bins and One Hot Encoding)
#Features for (X) and target variable (y) for new df
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]
```

```
#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=0)

#Scaling with Normalization (Min-Max Scaling)
#Initialize the MinMaxScaler
scaler_normalized = MinMaxScaler()
X_train_normalized = scaler.fit_transform(X_train)
X_test_normalized = scaler.transform(X_test)

#Initialize the GradientBoostingClassifier using the best hyperparameters using GridSearchCV
gb_model_normalized = GradientBoostingClassifier(
    learning_rate=0.01,
    n_estimators=100,
    max_depth=3,
    min_samples_split=2,
    min_samples_leaf=4,
    random_state=0
)

#Train on the normalized training set
gb_model_normalized.fit(X_train_normalized, y_train)

#Predictions on the normalized test set
y_pred_gb_normalized = gb_model_normalized.predict(X_test_normalized)

#Evaluation of the model with normalization
accuracy_gb_normalized = accuracy_score(y_test, y_pred_gb_normalized)
precision_gb_normalized = precision_score(y_test, y_pred_gb_normalized)
recall_gb_normalized = recall_score(y_test, y_pred_gb_normalized)
```

```
f1_gb_normalized = f1_score(y_test, y_pred_gb_normalized)

roc_auc_gb_normalized = roc_auc_score(y_test, gb_model_normalized.decision_function(X_test_normalized))

pr_auc_gb_normalized = average_precision_score(y_test,
gb_model_normalized.decision_function(X_test_normalized))

conf_matrix_gb_normalized = confusion_matrix(y_test, y_pred_gb_normalized)


#Print metrics

print("Gradient Boosting Model Metrics with Normalization:")

print(f"Accuracy: {accuracy_gb_normalized}")

print(f"Precision: {precision_gb_normalized}")

print(f"Recall: {recall_gb_normalized}")

print(f"F1 Score: {f1_gb_normalized}")

print(f"AUC-ROC: {roc_auc_gb_normalized}")

print(f"AUC-PR: {pr_auc_gb_normalized}")


#Plot the confusion matrix for the Classifier

sns.heatmap(conf_matrix_gb_normalized, annot=True, fmt="d", cmap="BuPu", cbar=False,

        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix Gradient Boosting with Normalization and Feature Engineer")

plt.show()

##Multilayer Perceptron (MLP) with Min-Max Scaling and Feature Engineering (Create Bins and One Hot Encoding)

#Features for (X) and target variable (y) for new df

X = df_one_hot_encoding.drop(["Outcome"], axis=1)

y = df_one_hot_encoding["Outcome"]


#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=0)
```

```
#Scaling with Normalization (Min-Max Scaling)
#Initialize the MinMaxScaler
scaler = MinMaxScaler()
X_train_normalized = scaler.fit_transform(X_train)
X_test_normalized = scaler.transform(X_test)

#Initialize the MLPClassifier model using the best hyperparameters using GridSearchCV
mlp = MLPClassifier(alpha=0.001, hidden_layer_sizes=(50,), max_iter=1000, random_state=0)

#Train on the normalized training set
mlp.fit(X_train_normalized, y_train)

#Predictions on the normalized test set
y_pred_normalized = mlp.predict(X_test_normalized)

#Evaluation of the model with normalization
accuracy_normalized = accuracy_score(y_test, y_pred_normalized)
precision_normalized = precision_score(y_test, y_pred_normalized)
recall_normalized = recall_score(y_test, y_pred_normalized)
f1_normalized = f1_score(y_test, y_pred_normalized)
roc_auc_normalized = roc_auc_score(y_test, mlp.predict_proba(X_test_normalized)[:, 1])
pr_auc_normalized = average_precision_score(y_test, mlp.predict_proba(X_test_normalized)[:, 1])
conf_matrix_normalized = confusion_matrix(y_test, y_pred_normalized)

#Print metrics
print("Metrics with Normalization:")
print(f"Accuracy: {accuracy_normalized}")
print(f"Precision: {precision_normalized}")
print(f"Recall: {recall_normalized}")
print(f"F1 Score: {f1_normalized}")
print(f"AUC-ROC: {roc_auc_normalized}")
```

```
print(f"AUC-PR: {pr_auc_normalized}")


#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_normalized, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix MLP Classifier with Normalization")
plt.show()
```

Κώδικας Β περίπτωσης Standardization & Feature Engineering (Binning και One Hot Encoding).

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import missingno as msno
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.neighbors import LocalOutlierFactor
from sklearn.preprocessing import MinMaxScaler, LabelEncoder, StandardScaler, RobustScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, precision_score, recall_score,
f1_score, roc_auc_score, average_precision_score
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import MinMaxScaler
```

```
from sklearn.neighbors import KNeighborsClassifier

from sklearn.neural_network import MLPClassifier


#Load the pima dataset from my folder

def load_data():

   data = pd.read_csv('C:/Users/geras/OneDrive/Έγγραφα/Μεταπτυχιακό/Εξάμηνο 3 - Διπλωματική Εργασία/diabetes.csv')

   return data


df = load_data()

df.head()


##Logistic Regression with Standard Scaling

#Features for (X) and target variable (y)

X = df.drop('Outcome', axis=1)

#X=df[['Glucose','BloodPressure','BMI']] test Feature importance

y = df['Outcome']


#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=0)



#Scaling with Standardization

#Initialize the StandardScaler

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)



#Initialize the Logistic Regression model using the best hyperparameters using GridSearchCV

model = LogisticRegression(C=0.1, penalty='l2', random_state=0)  # #Best parameters using GridSearchCV
```

```
#Train on the standardized training set
model.fit(X_train_scaled, y_train)

#Predictions on the standardized test set
y_pred = model.predict(X_test_scaled)

#Evaluation of the model with standardization
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, model.predict_proba(X_test_scaled)[:, 1])
pr_auc = average_precision_score(y_test, model.predict_proba(X_test_scaled)[:, 1])

conf_matrix = confusion_matrix(y_test, y_pred)

#Print metrics
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"AUC-ROC: {roc_auc}")
print(f"AUC-PR: {pr_auc}")


#Plot the confusion matrix for the model
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

```
plt.title("Confusion Matrix Logistic Regression with Standardization")
plt.show()
##SVM with Standard Scaling
#Features for (X) and target variable (y)
X = df.drop("Outcome", axis=1)
#X=df[['Glucose','BloodPressure','BMI']] test Feature importance
y = df["Outcome"]

#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=0)

#Scaling with Standardization
#Initialize the StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Initialize the SVM model using the best hyperparameters using GridSearchCV
model_svm = SVC(kernel='linear', C=0.1, gamma='scale', random_state=0)

#Train on the standardized training set
model_svm.fit(X_train_scaled, y_train)

#Predictions on the standardized test set
y_pred_svm = model_svm.predict(X_test_scaled)

#Evaluation of the model with standardization
accuracy_svm = accuracy_score(y_test, y_pred_svm)
precision_svm = precision_score(y_test, y_pred_svm)
recall_svm = recall_score(y_test, y_pred_svm)
f1_svm = f1_score(y_test, y_pred_svm)
```

```
roc_auc_svm = roc_auc_score(y_test, model_svm.decision_function(X_test_scaled))
pr_auc_svm = average_precision_score(y_test, model_svm.decision_function(X_test_scaled))
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)

#Print metrics
print("SVM Model Metrics:")
print(f"Accuracy: {accuracy_svm}")
print(f"Precision: {precision_svm}")
print(f"Recall: {recall_svm}")
print(f"F1 Score: {f1_svm}")
print(f"AUC-ROC: {roc_auc_svm}")
print(f"AUC-PR: {pr_auc_svm}")

#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_svm, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix SVM with Standardization")
plt.show()
##Random Forest with Standard Scaling
#Features for (X) and target variable (y)
X = df.drop("Outcome", axis=1)
#X=df[['Glucose','BloodPressure','BMI']] test Feature importance
y = df["Outcome"]

#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=0)

#Scaling with Standardization
#Initialize the StandardScaler
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Initialize the Random Forest Classifier using the best hyperparameters using GridSearchCV
rf_model = RandomForestClassifier(
    n_estimators=100,
    max_depth=10,
    min_samples_split=5,
    min_samples_leaf=2,
    random_state=0
)

#Train on the standardized training set
rf_model.fit(X_train_scaled, y_train)

#Predictions on the standardized test set
y_pred_rf = rf_model.predict(X_test_scaled)

#Evaluation of the model with standardization
accuracy_rf = accuracy_score(y_test, y_pred_rf)
precision_rf = precision_score(y_test, y_pred_rf)
recall_rf = recall_score(y_test, y_pred_rf)
f1_rf = f1_score(y_test, y_pred_rf)
roc_auc_rf = roc_auc_score(y_test, rf_model.predict_proba(X_test_scaled)[:, 1])
pr_auc_rf = average_precision_score(y_test, rf_model.predict_proba(X_test_scaled)[:, 1])
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)

#Print metrics
print("Random Forest Model Metrics:")
print(f"Accuracy: {accuracy_rf}")
```

```
print(f"Precision: {precision_rf}")
print(f"Recall: {recall_rf}")
print(f"F1 Score: {f1_rf}")
print(f"AUC-ROC: {roc_auc_rf}")
print(f"AUC-PR: {pr_auc_rf}")

#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_rf, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix Random Forest with Standardization")
plt.show()
##K-nn with Standard Scaling
#Features for (X) and target variable (y)
X = df.drop("Outcome", axis=1)
#X=df[['Glucose','BloodPressure','BMI']] test Feature importance
y = df["Outcome"]

#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=0)

#Scaling with Standardization
#Initialize the StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Initialize the KNN Classifier using the best hyperparameters using GridSearchCV
knn_model = KNeighborsClassifier(n_neighbors=7, p=1, weights='distance')
```

```
#Train on the standardized training set
knn_model.fit(X_train_scaled, y_train)

#Predictions on the standardized test set
y_pred_knn = knn_model.predict(X_test_scaled)

#Evaluation of the model with standardization
accuracy_knn = accuracy_score(y_test, y_pred_knn)
precision_knn = precision_score(y_test, y_pred_knn)
recall_knn = recall_score(y_test, y_pred_knn)
f1_knn = f1_score(y_test, y_pred_knn)
roc_auc_knn = roc_auc_score(y_test, knn_model.predict_proba(X_test_scaled)[:, 1])
pr_auc_knn = average_precision_score(y_test, knn_model.predict_proba(X_test_scaled)[:, 1])
conf_matrix_knn = confusion_matrix(y_test, y_pred_knn)

#Print metrics
print("KNN Model Metrics:")
print(f"Accuracy: {accuracy_knn}")
print(f"Precision: {precision_knn}")
print(f"Recall: {recall_knn}")
print(f"F1 Score: {f1_knn}")
print(f"AUC-ROC: {roc_auc_knn}")
print(f"AUC-PR: {pr_auc_knn}")

#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_knn, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix KNN with Standardization")
plt.show()
```

```
##Gradient Boosting with Standard Scaling
#Features for (X) and target variable (y)
X = df.drop("Outcome", axis=1)
#X=df[['Glucose','BloodPressure','BMI']] test Feature importance
y = df["Outcome"]

#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=0)

#Scaling with Standardization
#Initialize the StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Initialize the Gradient Boosting Classifier using the best hyperparameters using GridSearchCV
gb_model = GradientBoostingClassifier(
    learning_rate=0.01,
    n_estimators=100,
    max_depth=3,
    min_samples_split=2,
    min_samples_leaf=4,
    random_state=0
)

#Train on the standardized training set
gb_model.fit(X_train_scaled, y_train)

#Predictions on the standardized test set
y_pred_gb = gb_model.predict(X_test_scaled)
```

```
#Evaluation of the model with standardization
accuracy_gb = accuracy_score(y_test, y_pred_gb)
precision_gb = precision_score(y_test, y_pred_gb)
recall_gb = recall_score(y_test, y_pred_gb)
f1_gb = f1_score(y_test, y_pred_gb)
roc_auc_gb = roc_auc_score(y_test, gb_model.decision_function(X_test_scaled))
pr_auc_gb = average_precision_score(y_test, gb_model.decision_function(X_test_scaled))
conf_matrix_gb = confusion_matrix(y_test, y_pred_gb)

#Print metrics
print("Gradient Boosting Model Metrics:")
print(f"Accuracy: {accuracy_gb}")
print(f"Precision: {precision_gb}")
print(f"Recall: {recall_gb}")
print(f"F1 Score: {f1_gb}")
print(f"AUC-ROC: {roc_auc_gb}")
print(f"AUC-PR: {pr_auc_gb}")

#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_gb, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix Gradient Boosting with Standardization")
plt.show()
##Multilayer Perceptron (MLP) with Standard Scaling
#Features for (X) and target variable (y)
X = df.drop('Outcome', axis=1)
y = df['Outcome']

#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=0)

#Scaling with Standardization
#Initialize the StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Initialize the MLPClassifier model using the best hyperparameters using GridSearchCV
mlp = MLPClassifier(alpha=0.001, hidden_layer_sizes=(50,), max_iter=1000, random_state=0)

#Train on the standardized training set
mlp.fit(X_train_scaled, y_train)

#Predictions on the standardized test set
y_pred_scaled = mlp.predict(X_test_scaled)

#Evaluation of the model with standardization
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, mlp.predict_proba(X_test_scaled)[:, 1])
pr_auc = average_precision_score(y_test, mlp.predict_proba(X_test_scaled)[:, 1])

conf_matrix = confusion_matrix(y_test, y_pred)

#Evaluation of the model with standardization
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

```
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, mlp.predict_proba(X_test_scaled)[:, 1])
pr_auc = average_precision_score(y_test, mlp.predict_proba(X_test_scaled)[:, 1])

conf_matrix = confusion_matrix(y_test, y_pred)

#Print metrics
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"AUC-ROC: {roc_auc}")
print(f"AUC-PR: {pr_auc}")


#Plot the confusion matrix for the model
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix MLP with Standardization")
plt.show()
#Feature Engineering
#Create Bins for BMI and Age because we can capture non-linear relationships with the target variable
bmi_bins = [0, 18.5, 24.9, 29.9, 100]  # BMI categories
bmi_labels = ['Very Underweight', 'Normal Weight', 'Overweight', 'Obese']
df['BMI_Category'] = pd.cut(df['BMI'], bins=bmi_bins, labels=bmi_labels, right=False)

age_bins = [-1, 30, 50, float('inf')]  # Age groups
age_labels = ['Youth', 'Adults', 'Elderly']
df['Age_Group'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels)
```

```
#One hot encoding in order to convert the two categorical variables into numerical
def one_hot_encoder(df, categorical_cols, drop_first=True):
    df = pd.get_dummies(df, columns=categorical_cols, drop_first=drop_first)
    return df

df_one_hot_encoding = one_hot_encoder(df, categorical_cols=['BMI_Category' , 'Age_Group'])
##Logistic Regression with Standard Scaling and Feature Engineering (Create Bins and One Hot Encoding)
#Features for (X) and target variable (y) for new df
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]

#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=0)


#Scaling with Standardization
#Initialize the StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Initialize the Logistic Regression model using the best hyperparameters using GridSearchCV
model = LogisticRegression(C=0.1, penalty='l2', random_state=0)

#Train on the standardized training set
model.fit(X_train_scaled, y_train)

#Predictions on the standardized test set
y_pred = model.predict(X_test_scaled)

#Evaluation of the model with standardization
```

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, model.predict_proba(X_test_scaled)[:, 1])
pr_auc = average_precision_score(y_test, model.predict_proba(X_test_scaled)[:, 1])
conf_matrix = confusion_matrix(y_test, y_pred)

#Print metrics
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"AUC-ROC: {roc_auc}")
print(f"AUC-PR: {pr_auc}")



#Plot the confusion matrix for the model
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix Logistic Regression with Standardization and Feature Engineer")
plt.show()
##SVM with Standard Scaling and Feature Engineering (Create Bins and One Hot Encoding)
#Features for (X) and target variable (y) for new df
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]



#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=0)



#Scaling with Standardization
#Initialize the StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Initialize the SVM model using the best hyperparameters using GridSearchCV
model_svm = SVC(kernel='linear', C=0.1, gamma='scale', random_state=0)

#Train on the standardized training set
model_svm.fit(X_train_scaled, y_train)

#Predictions on the standardized test set
y_pred_svm = model_svm.predict(X_test_scaled)

#Evaluation of the model with standardization
accuracy_svm = accuracy_score(y_test, y_pred_svm)
precision_svm = precision_score(y_test, y_pred_svm)
recall_svm = recall_score(y_test, y_pred_svm)
f1_svm = f1_score(y_test, y_pred_svm)
roc_auc_svm = roc_auc_score(y_test, model_svm.decision_function(X_test_scaled))
pr_auc_svm = average_precision_score(y_test, model_svm.decision_function(X_test_scaled))
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)

#Print metrics
print("SVM Model Metrics:")
print(f"Accuracy: {accuracy_svm}")
print(f"Precision: {precision_svm}")
```

```
print(f"Recall: {recall_svm}")
print(f"F1 Score: {f1_svm}")
print(f"AUC-ROC: {roc_auc_svm}")
print(f"AUC-PR: {pr_auc_svm}")

#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_svm, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix SVM with Standardization and Feature Engineer")
plt.show()
##Random Forest with Standard Scaling and Feature Engineering (Create Bins and One Hot Encoding)
#Features for (X) and target variable (y)
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

#Scaling with Standardization
#Initialize the StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


#Initialize the Random Forest Classifier using the best hyperparameters using GridSearchCV
rf_model = RandomForestClassifier(
    n_estimators=100,
    max_depth=None,
```

```
    min_samples_split=5,
    min_samples_leaf=2,
    random_state=0
)


#Train on the standardized training set
rf_model.fit(X_train_scaled, y_train)

#Predictions on the standardized test set
y_pred_rf = rf_model.predict(X_test_scaled)

#Evaluation of the model with standardization
accuracy_rf = accuracy_score(y_test, y_pred_rf)
precision_rf = precision_score(y_test, y_pred_rf)
recall_rf = recall_score(y_test, y_pred_rf)
f1_rf = f1_score(y_test, y_pred_rf)
roc_auc_rf = roc_auc_score(y_test, rf_model.predict_proba(X_test_scaled)[:, 1])
pr_auc_rf = average_precision_score(y_test, rf_model.predict_proba(X_test_scaled)[:, 1])
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)

#Print metrics
print("Random Forest Model Metrics:")
print(f"Accuracy: {accuracy_rf}")
print(f"Precision: {precision_rf}")
print(f"Recall: {recall_rf}")
print(f"F1 Score: {f1_rf}")
print(f"AUC-ROC: {roc_auc_rf}")
print(f"AUC-PR: {pr_auc_rf}")

#Plot the confusion matrix for the model
```

```
sns.heatmap(conf_matrix_rf, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix Random Forest with Standardization and Feature Engineer")
plt.show()
##K-nn with Standard Scaling and Feature Engineering (Create Bins and One Hot Encoding)
#Features for (X) and target variable (y)
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]

#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=0)

#Scaling with Standardization
#Initialize the StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Initialize the KNN Classifier using the best hyperparameters using GridSearchCV
knn_model = KNeighborsClassifier(n_neighbors=7, p=1, weights='distance') #Best parameters with GridSearchCV

#Train on the standardized training set
knn_model.fit(X_train_scaled, y_train)

#Predictions on the standardized test set
y_pred_knn = knn_model.predict(X_test_scaled)

#Evaluation of the model with standardization
accuracy_knn = accuracy_score(y_test, y_pred_knn)
```

```
precision_knn = precision_score(y_test, y_pred_knn)
recall_knn = recall_score(y_test, y_pred_knn)
f1_knn = f1_score(y_test, y_pred_knn)
roc_auc_knn = roc_auc_score(y_test, knn_model.predict_proba(X_test_scaled)[:, 1])
pr_auc_knn = average_precision_score(y_test, knn_model.predict_proba(X_test_scaled)[:, 1])
conf_matrix_knn = confusion_matrix(y_test, y_pred_knn)

#Print metrics
print("KNN Model Metrics:")
print(f"Accuracy: {accuracy_knn}")
print(f"Precision: {precision_knn}")
print(f"Recall: {recall_knn}")
print(f"F1 Score: {f1_knn}")
print(f"AUC-ROC: {roc_auc_knn}")
print(f"AUC-PR: {pr_auc_knn}")

#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_knn, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix KNN with Standardization and Feature Engineer")
plt.show()
##Gradient Boosting with Standard Scaling and Feature Engineering (Create Bins and One Hot Encoding)
#Features for (X) and target variable (y)
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]

#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=0)
```

```
#Scaling with Standardization
#Initialize the StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Initialize the Gradient Boosting Classifier using the best hyperparameters using GridSearchCV
gb_model = GradientBoostingClassifier(
    learning_rate=0.01,
    n_estimators=100,
    max_depth=3,
    min_samples_split=2,
    min_samples_leaf=4,
    random_state=0
)

#Train on the standardized training set
gb_model.fit(X_train_scaled, y_train)

#Predictions on the standardized test set
y_pred_gb = gb_model.predict(X_test_scaled)

#Evaluation of the model with standardization
accuracy_gb = accuracy_score(y_test, y_pred_gb)
precision_gb = precision_score(y_test, y_pred_gb)
recall_gb = recall_score(y_test, y_pred_gb)
f1_gb = f1_score(y_test, y_pred_gb)
roc_auc_gb = roc_auc_score(y_test, gb_model.decision_function(X_test_scaled))
pr_auc_gb = average_precision_score(y_test, gb_model.decision_function(X_test_scaled))
conf_matrix_gb = confusion_matrix(y_test, y_pred_gb)
```

```
#Print metrics
print("Gradient Boosting Model Metrics:")
print(f"Accuracy: {accuracy_gb}")
print(f"Precision: {precision_gb}")
print(f"Recall: {recall_gb}")
print(f"F1 Score: {f1_gb}")
print(f"AUC-ROC: {roc_auc_gb}")
print(f"AUC-PR: {pr_auc_gb}")

#Plot the confusion matrix for the model
sns.heatmap(conf_matrix_gb, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix Gradient Boosting with Standardization and Feature Engineer")
plt.show()
##Multilayer Perceptron (MLP) with Standard Scaling and Feature Engineering (Create Bins and One Hot Encoding)
#Features for (X) and target variable (y) for new df
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]

#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=0)

#Scaling with Standardization
#Initialize the StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Initialize the MLPClassifier model using the best hyperparameters using GridSearchCV
```

```
mlp= MLPClassifier(alpha=0.001, hidden_layer_sizes=(50,), max_iter=1000, random_state=0)

#Train on the standardized training set
mlp.fit(X_train_scaled, y_train)

#Predictions on the standardized test set
y_pred_scaled = mlp.predict(X_test_scaled)

#Evaluation of the model with standardization
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, mlp.predict_proba(X_test_scaled)[:, 1])
pr_auc = average_precision_score(y_test, mlp.predict_proba(X_test_scaled)[:, 1])

conf_matrix = confusion_matrix(y_test, y_pred)

#Evaluation of the model with standardization
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, mlp.predict_proba(X_test_scaled)[:, 1])
pr_auc = average_precision_score(y_test, mlp.predict_proba(X_test_scaled)[:, 1])

conf_matrix = confusion_matrix(y_test, y_pred)

#Print metrics
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
```

```
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"AUC-ROC: {roc_auc}")
print(f"AUC-PR: {pr_auc}")



#Plot the confusion matrix for the model
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="BuPu", cbar=False,
        xticklabels=["No Diabetes", "Diabetes"], yticklabels=["No Diabetes", "Diabetes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix MLP with Standardization and Feature Engineer")
plt.show()
```

Κώδικας GridSearchCV για Normalization και Standardization με ή χωρίς Feature Engineering (Binning και One Hot Encoding).

```
##Logistic Regression
X = df.drop('Outcome', axis=1)
y = df['Outcome']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

scaler = MinMaxScaler() #StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

param_grid = {
   'C': [0.001, 0.01, 0.1, 1, 10, 100],  # Inverse of regularization strength
   'penalty':['l2', 'none']
}
```

```
logreg_model = LogisticRegression()

grid_search = GridSearchCV(logreg_model, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_scaled, y_train)


best_params = grid_search.best_params_
print(f"Best Hyperparameters: {best_params}")


best_logreg_model = LogisticRegression(**best_params)
best_logreg_model.fit(X_train_scaled, y_train)
y_pred = best_logreg_model.predict(X_test_scaled)

##SVM
X = df.drop("Outcome", axis=1)
y = df["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Feature Scaling
scaler = MinMaxScaler() #StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


param_grid_svm = {
   'C': [0.1, 1, 10],'kernel': ['linear', 'rbf', 'poly'], 'gamma': ['scale', 'auto']
}
```

```
model_svm = SVC(random_state=0)

grid_search_svm = GridSearchCV(model_svm, param_grid_svm, cv=5, scoring='accuracy')
grid_search_svm.fit(X_train_scaled, y_train)

best_params_svm = grid_search_svm.best_params_
print(f"Best Hyperparameters for SVM: {best_params_svm}")
##Random Forest
X = df.drop("Outcome", axis=1)
y = df["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
# Feature Scaling
scaler = MinMaxScaler() #StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)



param_grid_rf = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

model_rf = RandomForestClassifier(random_state=0)

grid_search_rf = GridSearchCV(model_rf, param_grid_rf, cv=5, scoring='accuracy')
grid_search_rf.fit(X_train, y_train)
```

```
best_params_rf = grid_search_rf.best_params_
print(f"Best Hyperparameters for Random Forest: {best_params_rf}")

##K-nn
X = df.drop("Outcome", axis=1)
y = df["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)


scaler = MinMaxScaler() #StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


param_grid_knn = {
    'n_neighbors': [3, 5, 7, 9],  # Number of neighbors
    'weights': ['uniform', 'distance'],  # Weight function used in prediction
    'p': [1, 2]  # Power parameter for the Minkowski distance
}

model_knn = KNeighborsClassifier()

grid_search_knn = GridSearchCV(model_knn, param_grid_knn, cv=5, scoring='accuracy')
grid_search_knn.fit(X_train_scaled, y_train)


best_params_knn = grid_search_knn.best_params_
print(f"Best Hyperparameters for K-NN: {best_params_knn}")

#Gradient Boosting
```

```
X = df.drop("Outcome", axis=1)
y = df["Outcome"]



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
scaler = MinMaxScaler() #StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test= scaler.transform(X_test)

param_grid_gb = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize Gradient Boosting model
model_gb = GradientBoostingClassifier(random_state=0)

# Use GridSearchCV to find the best hyperparameters
grid_search_gb = GridSearchCV(model_gb, param_grid_gb, cv=5, scoring='accuracy')
grid_search_gb.fit(X_train, y_train)

# Get the best hyperparameters
best_params_gb = grid_search_gb.best_params_
print(f"Best Hyperparameters for Gradient Boosting: {best_params_gb}")
##Multilayer Perceptron (MLP)
X = df.drop('Outcome', axis=1)
y = df['Outcome']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

scaler = MinMaxScaler()  #StandardScaler()
X_train_normalized = scaler.fit_transform(X_train)
X_test_normalized = scaler.transform(X_test)

param_grid = {
   'hidden_layer_sizes': [(100,), (50, 50), (100, 50), (50,)],
   'max_iter': [200, 500, 1000],
   'alpha': [0.0001, 0.001, 0.01],
   'random_state': [0]
}


mlp = MLPClassifier()


grid_search = GridSearchCV(mlp, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_normalized, y_train)


print("Best Parameters: ", grid_search.best_params_)
print("Best Accuracy: ", grid_search.best_score_)


best_mlp = grid_search.best_estimator_

bmi_bins = [0, 18.5, 24.9, 29.9, 100]  # BMI categories
bmi_labels = ['Very Underweight', 'Normal Weight', 'Overweight', 'Obese']
df['BMI_Category'] = pd.cut(df['BMI'], bins=bmi_bins, labels=bmi_labels, right=False)
```

```
age_bins = [-1, 30, 50, float('inf')]  # Age groups
age_labels = ['Youth', 'Adults', 'Elderly']
df['Age_Group'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels)

def one_hot_encoder(df, categorical_cols, drop_first=True):
    df = pd.get_dummies(df, columns=categorical_cols, drop_first=drop_first)
    return df

df_one_hot_encoding = one_hot_encoder(df, categorical_cols=['BMI_Category' , 'Age_Group'])
##Logistic Regression
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

scaler = MinMaxScaler() #StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],   'penalty':['l2', 'none']}

logreg_model = LogisticRegression()


grid_search = GridSearchCV(logreg_model, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_scaled, y_train)


best_params = grid_search.best_params_
```

```
print(f"Best Hyperparameters: {best_params}")
##SVM
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

scaler = MinMaxScaler() #StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

param_grid_svm = {
    'C': [0.1, 1, 10],'kernel': ['linear', 'rbf', 'poly'],'gamma': ['scale', 'auto']}

model_svm = SVC(random_state=0)


grid_search_svm = GridSearchCV(model_svm, param_grid_svm, cv=5, scoring='accuracy')
grid_search_svm.fit(X_train_scaled, y_train)


best_params_svm = grid_search_svm.best_params_
print(f"Best Hyperparameters for SVM: {best_params_svm}")

##Random Forest
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
scaler = MinMaxScaler() #StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

```
X_test_scaled = scaler.transform(X_test)



param_grid_rf = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

model_rf = RandomForestClassifier(random_state=0)

grid_search_rf = GridSearchCV(model_rf, param_grid_rf, cv=5, scoring='accuracy')
grid_search_rf.fit(X_train, y_train)

best_params_rf = grid_search_rf.best_params_
print(f"Best Hyperparameters for Random Forest: {best_params_rf}")
##K-nn
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)



scaler = MinMaxScaler() #StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

param_grid_knn = {
    'n_neighbors': [3, 5, 7, 9],'weights': ['uniform', 'distance'],'p': [1, 2] }
```

```
model_knn = KNeighborsClassifier()

grid_search_knn = GridSearchCV(model_knn, param_grid_knn, cv=5, scoring='accuracy')
grid_search_knn.fit(X_train_scaled, y_train)

best_params_knn = grid_search_knn.best_params_
print(f"Best Hyperparameters for K-NN: {best_params_knn}")
##Gradient Boosting
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

scaler = MinMaxScaler() #StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

param_grid_gb = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

model_gb = GradientBoostingClassifier(random_state=0)

grid_search_gb = GridSearchCV(model_gb, param_grid_gb, cv=5, scoring='accuracy')
grid_search_gb.fit(X_train, y_train)
```

```
best_params_gb = grid_search_gb.best_params_
print(f"Best Hyperparameters for Gradient Boosting: {best_params_gb}")
##Multilayer Perceptron (MLP)
X = df_one_hot_encoding.drop(["Outcome"], axis=1)
y = df_one_hot_encoding["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

scaler = MinMaxScaler() #StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

param_grid = {
   'hidden_layer_sizes': [(100,), (50, 50), (100, 50), (50,)],
   'max_iter': [200, 500, 1000],
   'alpha': [0.0001, 0.001, 0.01],
   'random_state': [0]
}

mlp = MLPClassifier()

grid_search = GridSearchCV(mlp, param_grid, cv=5, scoring='accuracy')

grid_search.fit(X_train_normalized, y_train)

print("Best Parameters: ", grid_search.best_params_)
print("Best Accuracy: ", grid_search.best_score_)
```