



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επεξηγούμενη τεχνητή νοημοσύνη (Explainable Artificial Intelligence - xAI)
σε Electronic Health Records (EHR)

Υποψήφιος : Θεόδωρος Κοντός

Π2019193

p19kont1@ionio.gr

Επιβλέπων: Αριστείδης Βραχάτης

2023-2024

Τριμελής Επιτροπή

Αριστείδης Βραχάτης, Επίκουρος Καθηγητής

Παναγιώτης Βλάμος, Καθηγητής

Θεμιστοκλής Έξαρχος, Αναπληρωτής Καθηγητής

Περίληψη

Στην παρούσα πτυχιακή εργασία, γίνεται χρήση της επεξηγούμενης τεχνητής νοημοσύνης σε αρχεία ηλεκτρονικών φακέλων υγείας, χρησιμοποιώντας και συγκρίνοντας τους δύο ταξινομητές XGBoost και CatBoost. Έτσι, γίνεται μία προσπάθεια εντοπισμού των διαφορών μεταξύ των δύο ταξινομητών, μεταξύ των εργαλείων της επεξηγούμενης τεχνητής νοημοσύνης, αλλά και μεταξύ των διάφορων κλάσεων που προκύπτουν στο τελικό dataset.

Στο 1^ο κεφάλαιο, γίνεται μία εισαγωγή στην εργασία με αναφορά στους ορισμούς των Ηλεκτρονικών Φακέλων Υγείας αλλά και της επεξηγούμενης τεχνητής νοημοσύνης. Μεταξύ άλλων αναφέρονται τα πλεονεκτήματα και τα μειονεκτήματα των Ηλεκτρονικών Φακέλων Υγείας καθώς και πως μπορούν αυτοί να συνδυαστούν με την επεξηγούμενη τεχνητή νοημοσύνη.

Στο 2^ο κεφάλαιο, παρουσιάζεται μία εκτενής βιβλιογραφική ανασκόπηση 11 επιστημονικών άρθρων, με κύριο θέμα τη χρήση επεξηγούμενης τεχνητής νοημοσύνης σε Ηλεκτρονικούς Φάκελους Υγείας. Επίσης, αναφέρονται οι σημαντικότερες ελλείψεις στη βιβλιογραφία.

Το 3^ο κεφάλαιο περιέχει την μεθοδολογία που ακολουθήθηκε για την εκπόνηση της πτυχιακής εργασίας, δηλαδή του πως χρησιμοποιήθηκαν τα δεδομένα, πως αυτά επεξεργάστηκαν και πως έγινε η τελική τους απεικόνιση.

Στο 4^ο κεφάλαιο, έχουμε την παρουσίαση των διαγραμμάτων τόσο για την οπτικοποίηση των δεδομένων όσο και για την επεξηγούμενη μηχανική μάθηση. Επιπλέον, παρουσιάζεται και ένα μοντέλο διάγνωσης Προδιαβήτη/Διαβήτη που προέκυψε από την επεξεργασία των δεδομένων.

Στο 5^ο κεφάλαιο, γίνεται μία συζήτηση περί των αποτελεσμάτων που προέκυψαν από την εφαρμογή μηχανικής μάθησης στα αρχεία, γίνεται μία σύγκριση με την υπάρχουσα βιβλιογραφία ενώ παρουσιάζονται τόσο οι καινοτομίες όσο και οι αδυναμίες της εργασίας.

Τέλος, στο 6^ο κεφάλαιο προκύπτουν τα συμπεράσματα της πτυχιακής εργασίας και προτείνονται κάποιες μελλοντικές προοπτικές.

Φυσικά, μετά το κύριο μέρος της πτυχιακής παρατίθενται η βιβλιογραφία και ο κώδικας που αναπτύχθηκε τόσο για την οπτικοποίηση των δεδομένων, όσο και για την μηχανική μάθηση.

Λέξεις-κλειδιά: Ηλεκτρονικοί φάκελοι υγείας, μηχανική μάθηση, επεξηγούμενη τεχνητή νοημοσύνη, οπτικοποίηση δεδομένων, κατηγοριοποίηση δεδομένων, μοντέλο διάγνωσης ασθένειας

Abstract

In this thesis, we use of explainable artificial intelligence in electronic health record files, using and comparing the two classifiers XGBoost and CatBoost. Thus, an attempt is made to identify the differences between the two classifiers, between the tools of the explainable AI, and between the different classes that result in the final dataset.

In chapter 1, an introduction to the thesis is given with reference to the definitions of Electronic Health Records and of the explainable AI. Among others, the advantages and disadvantages of Electronic Health Records are mentioned as well as how they can be combined with explainable artificial intelligence.

In Chapter 2, an extensive literature review of 11 scientific articles is presented, focusing on the use of explainable artificial intelligence in Electronic Health Records. Also, the most important shortcomings in the literature are mentioned.

Chapter 3 contains the methodology followed for the thesis, how the data were used, how they were processed and how they were finally visualized.

In chapter 4, we have the presentation of the diagrams for both data visualization and explainable machine learning. In addition, we also present a model for the diagnosis of Prediabetes/Diabetes that resulted from the data processing.

In Chapter 5, there is a discussion of the results obtained by applying machine learning to the files, a comparison with the existing literature is made and both the innovations and weaknesses of the work are presented.

Finally, Chapter 6 draws the conclusions of the thesis and suggests some future perspectives.

Of course, after the main part of the thesis the literature and the code developed for both data visualization and machine learning are listed.

Keywords: Electronic health records, machine learning, explainable artificial intelligence, , data visualization, data classification, disease diagnosis model

Περιεχόμενα

Τριμελής Επιτροπή.....	2
Περίληψη	3
Abstract.....	4
Περιεχόμενα.....	5
Κατάλογος γραφημάτων	6
1. Εισαγωγή.....	8
1.1 Ηλεκτρονικοί Φάκελοι Υγείας (Electronic Health Records).....	8
1.1.1 Πλεονεκτήματα των ΗΦΥ	9
1.1.2 Μειονεκτήματα των ΗΦΥ.....	9
1.1.3 Μελλοντικός σχεδιασμός των ΗΦΥ.....	10
1.2 ΗΦΥ και επεξηγούμενη τεχνητή νοημοσύνη (xAI)	11
1.2.1 Ο ρόλος της επεξηγούμενης μηχανικής μάθησης	11
1.2.2 Προκλήσεις στην ενσωμάτωση των ΗΦΥ με την επεξηγούμενη μηχανική μάθηση.....	12
2. Παρόμοιες μελέτες.....	13
2.1 Βιβλιογραφική ανασκόπηση.....	13
2.2 Περιορισμοί και ελλείψεις στην βιβλιογραφία	23
3. Μεθοδολογία	24
3.1 Δεδομένα	26
3.2 Ένωση των dataframes	27
3.3 Απεικόνιση των δεδομένων	28
3.4 Extreme Gradient Boosting (XGBoost)	28
3.5 CatBoost	28
3.6 SHapley Additive exPlanations (SHAP)	28
3.7 Local Interpretable Model-agnostic Explanations (LIME).....	28
4. Αποτελέσματα	29
4.1 Plotting.....	29
4.2 Classification.....	35
4.2.1 Γενικός πίνακας αποτελεσμάτων	35
4.2.2 Classification ως προς το φύλο (gender)	35
4.2.3 Classification ως προς την κλάση colonoscopy	45

4.2.4	Classification ως προς την κλάση Td vax (Εμβόλιο για τέτανο και διφθερίτιδα)	54
4.3	Μοντέλο διάγνωσης Προδιαβήτη/Διαβήτη	63
4.3.1	XGBoost	63
4.3.2	CatBoost	65
4.3.3	Εφαρμογή διάγνωσης Προδιαβήτη/Διαβήτη	67
5.	Συζήτηση	68
5.1	Σχόλια	68
5.2	Σύγκριση με Άλλους Ερευνητές	70
5.3	Παρουσίαση Αδυναμιών	70
5.4	Προβληματισμοί	70
5.5	Παρουσίαση Σπουδαιότητας και Καινοτομίας	71
6.	Συμπεράσματα	72
6.1	Σημαντικά στοιχεία που αναπτύχθηκαν, ανοικτές ερωτήσεις	72
6.2	Μελλοντικές προοπτικές	72
7.	Βιβλιογραφία	73
	Παράρτημα Α – Ο κώδικας	76

Κατάλογος γραφημάτων

Γράφημα 1.	Δημογραφικά στοιχεία (φύλο, έτος γέννησης ανά ασθενή, έτος γέννησης ανά φύλο)	29
Γράφημα 2.	Ποσοστό ασθενών με ολοκληρωμένο ή ενεργό σχέδιο περίθαλψης	29
Γράφημα 3.	Ποσοστό ανά είδος περιστατικού (σε εξωτερικά ιατρεία, χωρίς διανυκτέρευση, στα επείγοντα, ενδονοσοκομειακά, ημερήσια)	30
Γράφημα 4.	Ιατρικές διαδικασίες (συνταγογράφηση, τοκετός, ανοσοθεραπεία κλπ.)	30
Γράφημα 5.	Ιατρικές παρατηρήσεις (ύψος, βάρος, αρτηριακή πίεση, δείκτης μάζας σώματος κ.α.)	31
Γράφημα 6.	Ιατρικές παρατηρήσεις στο σύνολο των ασθενών (Displot με KDE Curve)	31
Γράφημα 7.	Ιατρικές παρατηρήσεις στο σύνολο των ασθενών (Violinplot)	32
Γράφημα 8.	Violin Plots ιατρικών παρατηρήσεων χωρισμένων κατά ομάδες	33
Γράφημα 9.	Ποσοστό ιατρικών παρατηρήσεων ενός τυχαίου ασθενή	33
Γράφημα 10.	Μεταβολή τιμών των ιατρικών παρατηρήσεων δύο τυχαίων ασθενών ανά έτος	34
Γράφημα 11.	Μεταβολή τιμών των ιατρικών παρατηρήσεων δύο τυχαίων ασθενών ανά έτος	34
Γράφημα 12.	BarPlot του XGBoost (feature selected: gender)	35
Γράφημα 13.	WaterfallPlot και BeeswarmPlot του XGBoost (feature selected: gender)	36
Γράφημα 14.	SHAP DependencePlot του XGBoost για την ηλικία και το βάρος (feature selected: gender)	36

Γράφημα 15. SHAP DependencePlot του XGBoost για την ηλικία και το βάρος (feature selected: gender)	37
Γράφημα 16. SHAP DependencePlot του XGBoost για διάφορα χαρακτηριστικά (feature selected: gender)	38
Γράφημα 17. SHAP DependencePlot του XGBoost για διάφορα χαρακτηριστικά (feature selected: gender)	39
Γράφημα 18. LIME plot για την θετική και αρνητική επίδραση κάθε χαρακτηριστικού (feature selected: gender)	39
Γράφημα 19. SHAP BarPlot του CatBoost (feature selected: gender)	40
Γράφημα 20. WaterfallPlot και BeeswarmPlot του XGBoost (feature selected: gender)	40
Γράφημα 21. SHAP DependencePlots του CatBoost για διάφορα χαρακτηριστικά (feature selected: gender)	42
Γράφημα 22. SHAP DependencePlots του CatBoost για διάφορα χαρακτηριστικά (feature selected: gender)	43
Γράφημα 23. SHAP DependencePlots του CatBoost για διάφορα χαρακτηριστικά (feature selected: gender)	44
Γράφημα 24. LIME plot για την θετική και αρνητική επίδραση κάθε χαρακτηριστικού (feature selected: gender)	44
Γράφημα 25. SHAP BarPlot του XGBoost (feature selected: colonoscopy)	45
Γράφημα 26. SHAP BeeswarmPlot και WaterfallPlot του XGBoost (feature selected: colonoscopy)	45
Γράφημα 27. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)	47
Γράφημα 28. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)	47
Γράφημα 29. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)	48
Γράφημα 30. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)	49
Γράφημα 31. LIME plot για την θετική και αρνητική επίδραση κάθε χαρακτηριστικού (feature selected: colonoscopy)	49
Γράφημα 32. SHAP BarPlot του CatBoost (feature selected: colonoscopy)	50
Γράφημα 33. SHAP BeeswarmPlot και WaterfallPlot του CatBoost (feature selected: colonoscopy)	50
Γράφημα 34. SHAP DependencePlots του CatBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)	51
Γράφημα 35. SHAP DependencePlots του CatBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)	52
Γράφημα 36. SHAP DependencePlots του CatBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)	53
Γράφημα 37. LIME plot για την θετική και αρνητική επίδραση κάθε χαρακτηριστικού (feature selected: colonoscopy)	54
Γράφημα 38. SHAP BarPlot του XGBoost (feature selected: Td vax)	54
Γράφημα 39. SHAP BeeswarmPlot και WaterfallPlot του XGBoost (feature selected: Td vax)	55

Γράφημα 40. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax).....	55
Γράφημα 41. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax).....	56
Γράφημα 42. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax).....	57
Γράφημα 43. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax).....	58
Γράφημα 44. LIME plot για την θετική και αρνητική επίδραση κάθε χαρακτηριστικού (feature selected: Td vax).....	58
Γράφημα 45. SHAP BarPlot του CatBoost (feature selected: Td vax)	59
Γράφημα 46. SHAP BeeswarmPlot και WaterfallPlot του CatBoost (feature selected: Td vax) .	59
Γράφημα 47. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax).....	60
Γράφημα 48. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax).....	61
Γράφημα 49. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax).....	62
Γράφημα 50. LIME plot για την θετική και αρνητική επίδραση κάθε χαρακτηριστικού (feature selected: Td vax).....	63
Γράφημα 51. Αποτελέσματα του XGBoost για το μοντέλο διάγνωσης Προδιαβήτη/Διαβήτη	63
Γράφημα 52. SHAP BarPlot για τα χαρακτηριστικά του μοντέλου με XGBoost	64
Γράφημα 53. SHAP Beeswarm για τα χαρακτηριστικά του μοντέλου με XGBoost.....	64
Γράφημα 54. SHAP WaterfallPlot για τα χαρακτηριστικά του μοντέλου με XGBoost.....	64
Γράφημα 55. SHAP ScatterPlots για την ηλικία με XGBoost	65
Γράφημα 56. Αποτελέσματα του CatBoost για το μοντέλο διάγνωσης Προδιαβήτη/Διαβήτη	65
Γράφημα 57. SHAP BarPlot για τα χαρακτηριστικά του μοντέλου με CatBoost.....	65
Γράφημα 58. SHAP Beeswarm για τα χαρακτηριστικά του μοντέλου με CatBoost.....	66
Γράφημα 59. SHAP WaterfallPlot για τα χαρακτηριστικά του μοντέλου με CatBoost	66
Γράφημα 60. SHAP ScatterPlots για την ηλικία και το φύλο με CatBoost	66
Γράφημα 61. Για τα συγκεκριμένα στοιχεία εισόδου η εφαρμογή προβλέπει ότι το άτομο έχει προδιαβήτη.	67
Γράφημα 62. Για τα συγκεκριμένα στοιχεία εισόδου η εφαρμογή προβλέπει ότι το άτομο έχει διαβήτη.	67

1. Εισαγωγή

1.1 Ηλεκτρονικοί Φάκελοι Υγείας (Electronic Health Records)

Η χρήση των συστημάτων ηλεκτρονικών φακέλων υγείας (στο εξής ΗΦΥ) στην υγειονομική περίθαλψη αυξάνεται, με αποτέλεσμα την αφθονία ηλεκτρονικών κλινικών δεδομένων. Η υγειονομική περίθαλψη με γνώμονα τα δεδομένα επιδιώκει να αξιοποιήσει τα δεδομένα των ΗΦΥ για την παροχή εξατομικευμένης περίθαλψης και τη βελτίωση των αποτελεσμάτων.

Οι ΗΦΥ έχουν καταστεί ακρογωνιαίος λίθος της σύγχρονης υγειονομικής περίθαλψης, με στόχο τον εξορθολογισμό των κλινικών ροών εργασίας και τη βελτίωση της συνολικής αποτελεσματικότητας των συστημάτων υγείας.

Οι ΗΦΥ είναι ψηφιακές εκδόσεις των χάρτινων διαγραμμάτων των ασθενών και έχουν σχεδιαστεί για να είναι ολοκληρωμένα αρχεία σε πραγματικό χρόνο που παρέχουν πλήρη εικόνα του ιστορικού υγείας ενός ασθενούς. Περιλαμβάνουν μια ποικιλία τύπων δεδομένων, όπως:

- Δημογραφικά στοιχεία του ασθενούς
- Ιατρικό ιστορικό
- Λίστες φαρμάκων και αλλεργιών
- Εργαστηριακά και απεικονιστικά αποτελέσματα
- Σχέδια θεραπείας και σημειώσεις προόδου

Ο πρωταρχικός στόχος των ΗΦΥ είναι να βελτιώσουν την ποιότητα της περίθαλψης καθιστώντας τις πληροφορίες των ασθενών εύκολα προσβάσιμες στους παρόχους υγειονομικής περίθαλψης, διευκολύνοντας έτσι την καλύτερη λήψη κλινικών αποφάσεων. [1]

1.1.1 Πλεονεκτήματα των ΗΦΥ

Πολυάριθμες μελέτες έχουν αναδείξει τα δυνητικά οφέλη των ΗΦΥ, μεταξύ άλλων:

- Βελτίωση της ποιότητας της περίθαλψης: Οι ΗΦΥ μπορούν να μειώσουν τα ιατρικά σφάλματα, βελτιώνοντας την ακρίβεια των διαγνώσεων και διευκολύνοντας τη λήψη αποφάσεων βάσει τεκμηρίων. Μια ανασκόπηση εμπειρικών μελετών διαπίστωσε ότι το 78% των μελετών ανέφερε αύξηση της ποιότητας της υγειονομικής περίθαλψης μετά την εφαρμογή των ΗΦΥ.

- Μείωση του κόστους: Ορισμένες μελέτες έχουν αναφέρει ότι οι ΗΦΥ μπορούν να οδηγήσουν σε εξοικονόμηση κόστους με τη μείωση της ανάγκης για φυσική αποθήκευση χάρτινων αρχείων και την ελαχιστοποίηση των διπλών εξετάσεων. Περίπου το 56% των μελετών ανέφερε μείωση του κόστους που συνδέεται με τη χρήση ΗΦΥ. [2]

- Αυξημένη αποδοτικότητα: Οι ΗΦΥ εξορθολογίζουν τις ροές εργασίας με την αυτοματοποίηση διοικητικών εργασιών, όπως ο προγραμματισμός και η τιμολόγηση, γεγονός που μπορεί να οδηγήσει σε εξοικονόμηση χρόνου για τους παρόχους υγειονομικής περίθαλψης. Η έρευνα των Poissant L, Pereira J et. Al [3] έδειξε ότι οι ΗΦΥ μπορούν να μειώσουν τον χρόνο που δαπανάται για εργασίες τεκμηρίωσης κατά περίπου 12.6% έως 4% ανά συνάντηση με τον ασθενή.

1.1.2 Μειονεκτήματα των ΗΦΥ

Παρά τα πλεονεκτήματα, η εφαρμογή των ΗΦΥ είναι γεμάτη με προκλήσεις που μπορεί να εμποδίσουν την αποτελεσματικότητά τους:

- Αντίσταση στην καινοτομία: Συχνά υπάρχει αντίσταση από τους παρόχους υγειονομικής περίθαλψης οι οποίοι μπορεί να είναι απρόθυμοι να υιοθετήσουν νέες τεχνολογίες. Ένας σημαντικός παράγοντας αυτής της απροθυμίας είναι η αντιληπτή αύξηση του φόρτου εργασίας και ο χρόνος που απαιτείται για την εκπαίδευση.

- Ανησυχίες σχετικά με το απόρρητο: Η μετάβαση σε ψηφιακά αρχεία εγείρει ανησυχίες σχετικά με την ασφάλεια των δεδομένων και την ιδιωτικότητα των ασθενών. Ζητήματα όπως η μη εξουσιοδοτημένη πρόσβαση σε ευαίσθητες πληροφορίες υγείας μπορούν να υπονομεύσουν την εμπιστοσύνη στα συστήματα ΗΦΥ [4].

- Τεχνικά εμπόδια: Ζητήματα που σχετίζονται με τη λειτουργικότητα του συστήματος, την ακρίβεια των δεδομένων και το σχεδιασμό της διεπαφής χρήστη (User interface) μπορεί να περιπλέξουν τη χρήση του ΗΦΥ. Οι ιατροί συχνά εκφράζουν απογοήτευση με τα συστήματα ΗΦΥ που δεν ενσωματώνονται καλά στις υπάρχουσες ροές εργασίας ή που απαιτούν υπερβολικό χρόνο για την εισαγωγή δεδομένων [5].

Η εφαρμογή των ΗΦΥ δεν είναι απλώς μια τεχνολογική αλλαγή- είναι ένας κοινωνικοτεχνικός μετασχηματισμός που επηρεάζει τόσο τις κλινικές ροές εργασίας όσο και τις οργανωτικές δομές. Η ενσωμάτωση των ΗΦΥ μεταβάλλει τον τρόπο με τον οποίο οι πάροχοι υγειονομικής περίθαλψης αλληλεπιδρούν μεταξύ τους και με τους ασθενείς, οδηγώντας σε νέες ροές εργασίας και πρότυπα επικοινωνίας. Αυτός ο μετασχηματισμός μπορεί να οδηγήσει σε απρόβλεπτες συνέπειες, όπως:

- Διαταραχή της ροής εργασίας: Εάν τα συστήματα ΗΦΥ δεν είναι ευθυγραμμισμένα με τις υπάρχουσες ροές εργασίας, μπορεί να δημιουργήσουν αναποτελεσματικότητα αντί να εξορθολογήσουν τις διαδικασίες. Για παράδειγμα, οι κακώς σχεδιασμένες διεπαφές των ΗΦΥ μπορούν να οδηγήσουν σε αύξηση του χρόνου τεκμηρίωσης και εξουθένωση του υγειονομικού προσωπικού [1].

- Γνωστικό φορτίο: Η μετάβαση στους ΗΦΥ μπορεί να αυξήσει το γνωστικό φορτίο των κλινικών γιατρών, καθώς πρέπει να περιηγηθούν σε πολύπλοκα συστήματα ενώ παράλληλα παρέχουν φροντίδα στους ασθενείς. Αυτό μπορεί να μειώσει την ποιότητα των αλληλεπιδράσεων με τους ασθενείς και να οδηγήσει σε μειωμένη επαγγελματική ικανοποίηση των παρόχων υγειονομικής περίθαλψης [4].

Οι εμπειρίες των κλινικών ιατρών με τους ΗΦΥ είναι κρίσιμες για τη διαμόρφωση της αποτελεσματικότητάς τους. Η μελέτη των A. Karim Jabali, Abu Waris et. al [5] αποκάλυψε μικτές αντιλήψεις μεταξύ των παρόχων υγειονομικής περίθαλψης σχετικά με τους ΗΦΥ. Ενώ ορισμένοι κλινικοί ιατροί εκτιμούν την προσβασιμότητα των πληροφοριών των ασθενών και τη δυνατότητα βελτιωμένου συντονισμού της περίθαλψης, άλλοι βρίσκουν τους ΗΦΥ δυσκίνητους και χρονοβόρους. Οι νοσηλεύτες, ειδικότερα, ανέφεραν θετικές εμπειρίες που σχετίζονται με την αποτελεσματικότητα, ενώ οι γιατροί εξέφρασαν ανησυχίες σχετικά με τον αντίκτυπο των ΗΦΥ στη ροή εργασίας τους.

1.1.3 Μελλοντικός σχεδιασμός των ΗΦΥ

Για να μεγιστοποιηθούν τα οφέλη των ΗΦΥ, μπορούν να χρησιμοποιηθούν διάφορες στρατηγικές:

- Σχεδιασμός με επίκεντρο τον χρήστη: Τα μελλοντικά συστήματα ΗΦΥ θα πρέπει να δίνουν προτεραιότητα στην εμπειρία του χρήστη, ενσωματώνοντας τα σχόλια των γιατρών για τη δημιουργία πιο διαισθητικών διεπαφών που ευθυγραμμίζονται με τις υπάρχουσες ροές εργασίας.

- Εκπαίδευση και υποστήριξη: Η συνεχής κατάρτιση και υποστήριξη των παρόχων υγειονομικής περίθαλψης είναι απαραίτητες για να διασφαλιστεί ότι μπορούν να χρησιμοποιούν

αποτελεσματικά τα συστήματα ΗΦΥ. Αυτό περιλαμβάνει όχι μόνο την αρχική κατάρτιση αλλά και τη συνεχή εκπαίδευση καθώς τα συστήματα εξελίσσονται [4].

- Πρότυπα διαλειτουργικότητας: Η καθιέρωση ισχυρών προτύπων διαλειτουργικότητας θα διευκολύνει την καλύτερη ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων ΗΦΥ, ενισχύοντας το συντονισμό της περίθαλψης και βελτιώνοντας τα αποτελέσματα των ασθενών.

- Εστίαση στην ουσιαστική χρήση: Η έμφαση στην έννοια της «ουσιαστικής χρήσης» -η αποτελεσματική χρήση των ΗΦΥ για τη βελτίωση της φροντίδας των ασθενών- θα βοηθήσει να διασφαλιστεί ότι τα συστήματα ΗΦΥ δεν υιοθετούνται απλώς, αλλά χρησιμοποιούνται στο μέγιστο των δυνατοτήτων τους για την προώθηση βελτιώσεων στην ποιότητα και την αποτελεσματικότητα της υγειονομικής περίθαλψης [5].

1.2 ΗΦΥ και επεξηγούμενη τεχνητή νοημοσύνη (xAI)

Η ενσωμάτωση των Ηλεκτρονικών Φακέλων Υγείας (ΗΦΥ) με τις τεχνολογίες μηχανικής μάθησης (Machine Learning), ιδίως την επεξηγούμενη μηχανική μάθηση (xAI), έχει αναδειχθεί ως μια καίρια εξέλιξη στην πληροφορική της υγειονομικής περίθαλψης. Η σχέση αυτή όχι μόνο ενισχύει τη λήψη κλινικών αποφάσεων, αλλά αντιμετωπίζει επίσης την κρίσιμη ανάγκη για διαφάνεια στις προβλέψεις που βασίζονται στην Τεχνητή Νοημοσύνη.

1.2.1 Ο ρόλος της επεξηγούμενης μηχανικής μάθησης

Η επεξηγούμενη μηχανική μάθηση σχετίζεται με την έννοια της εξήγησης ως διεπαφής μεταξύ των ανθρώπων και ενός συστήματος τεχνητής νοημοσύνης. Περιλαμβάνει συστήματα τεχνητής νοημοσύνης που είναι ακριβή και κατανοητά από τον άνθρωπο (Gilpin et al., 2018) [9]. Η xAI στοχεύει στην παροχή πληροφοριών σχετικά με τον τρόπο με τον οποίο τα μοντέλα καταλήγουν σε προβλέψεις, ενισχύοντας την εμπιστοσύνη και διευκολύνοντας τη λήψη καλύτερων κλινικών αποφάσεων.

Οι βασικές πτυχές της χρήσης επεξηγούμενης μηχανικής μάθησης σε ΗΦΥ είναι:

- Διαφάνεια: Οι τεχνικές xAI βοηθούν στην απομυθοποίηση των πολύπλοκων αλγορίθμων, επιτρέποντας στους κλινικούς γιατρούς να κατανοήσουν το σκεπτικό πίσω από τις προβλέψεις. Αυτό είναι ιδιαίτερα σημαντικό σε περιβάλλοντα υψηλού κινδύνου όπως η υγειονομική περίθαλψη, όπου οι αποφάσεις που βασίζονται σε αδιαφανή μοντέλα μπορεί να οδηγήσουν σε δυσμενή αποτελέσματα.

- Ερμηνευσιμότητα: Με τη διάσπαση των προβλέψεων του μοντέλου σε κατανοητά στοιχεία, η xAI επιτρέπει στους επαγγελματίες του τομέα της υγειονομικής περίθαλψης να επικυρώνουν και να πλαισιώνουν τις συστάσεις της τεχνητής νοημοσύνης στο πλαίσιο της κλινικής πρακτικής.

- Εμπιστοσύνη: Οι κλινικοί ιατροί είναι πιο πιθανό να υιοθετήσουν εργαλεία τεχνητής νοημοσύνης όταν μπορούν να κατανοήσουν και να εμπιστευτούν τις υποκείμενες διαδικασίες. Αυτή η εμπιστοσύνη είναι απαραίτητη για την επιτυχή ενσωμάτωση της τεχνητής νοημοσύνης στις κλινικές ροές εργασίας.

Η σχέση μεταξύ των ΗΦΥ και της xAI είναι πολύπλευρη, με διάφορους βασικούς τομείς αλληλεπίδρασης:

- Αξιοποίηση δεδομένων

Οι ΗΦΥ παρέχουν πλούσια, δομημένα σύνολα δεδομένων που είναι απαραίτητα για την εκπαίδευση μοντέλων μηχανικής μάθησης. Τα δεδομένα που καταγράφονται στους ΗΦΥ, συμπεριλαμβανομένων των κλινικών σημειώσεων, των εργαστηριακών αποτελεσμάτων και των δημογραφικών στοιχείων των ασθενών, μπορούν να αξιοποιηθούν για την ανάπτυξη προγνωστικών μοντέλων για διάφορες εφαρμογές υγειονομικής περίθαλψης. Για παράδειγμα, τα μοντέλα πρόβλεψης μπορούν να προβλέπουν τα αποτελέσματα των ασθενών, να εντοπίζουν πιθανές επιπλοκές ή να προτείνουν εξατομικευμένα σχέδια θεραπείας με βάση ιστορικά δεδομένα [6].

Ωστόσο, η αποτελεσματικότητα αυτών των μοντέλων εξαρτάται από την ποιότητα και την πληρότητα των δεδομένων που εξάγονται από τους ΗΦΥ. Οι ασυνεπείς πρακτικές εισαγωγής δεδομένων, οι πληροφορίες που λείπουν και οι παραλλαγές στις μορφές δεδομένων μπορούν να υπονομεύσουν την αξιοπιστία των μοντέλων μηχανικής μάθησης [4][5].

- Ενίσχυση της υποστήριξης των κλινικών αποφάσεων

Η xAI μπορεί να ενισχύσει σημαντικά τα συστήματα υποστήριξης κλινικών αποφάσεων (CDSS) που χρησιμοποιούν δεδομένα ΗΦΥ. Παρέχοντας εξηγήσεις για τις προβλέψεις, η xAI βοηθά τους κλινικούς ιατρούς να κατανοήσουν τη βάση των συστάσεων, διευκολύνοντας την τεκμηριωμένη λήψη αποφάσεων.

Η έρευνα των Payrovnaziri SN et. al δείχνει ότι η ενσωμάτωση του xAI στο CDSS μπορεί να βελτιώσει τη δέσμευση των κλινικών ιατρών και την τήρηση των συστάσεων, οδηγώντας τελικά σε καλύτερα αποτελέσματα για τους ασθενείς. Επιπλέον, καθιστώντας τη διαδικασία λήψης αποφάσεων πιο διαφανή, το xAI μπορεί να συμβάλει στην αντιμετώπιση των ανησυχιών σχετικά με την αλγοριθμική μεροληψία και να διασφαλίσει ότι οι συστάσεις είναι δίκαιες σε διαφορετικά σύνολα ασθενών.

- Αντιμετώπιση ηθικών και νομικών προκλήσεων

Η ενσωμάτωση της τεχνητής νοημοσύνης στην υγειονομική περίθαλψη εγείρει ηθικά και νομικά ζητήματα, ιδίως όσον αφορά τη λογοδοσία και τη συγκατάθεση μετά από ενημέρωση. Το xAI μπορεί να διαδραματίσει κρίσιμο ρόλο στον μετριασμό αυτών των ανησυχιών παρέχοντας σαφείς εξηγήσεις για τον τρόπο λήψης των αποφάσεων.

Για παράδειγμα, σε περιπτώσεις όπου η σύσταση ενός μοντέλου τεχνητής νοημοσύνης οδηγεί σε αρνητικό αποτέλεσμα, η κατανόηση του σκεπτικού πίσω από την απόφαση μπορεί να βοηθήσει στην αποσαφήνιση της ευθύνης. Αυτή η διαφάνεια είναι απαραίτητη για την ενίσχυση της εμπιστοσύνης τόσο μεταξύ των ασθενών όσο και μεταξύ των παρόχων υγειονομικής περίθαλψης [5][6].

1.2.2 Προκλήσεις στην ενσωμάτωση των ΗΦΥ με την επεξηγούμενη μηχανική μάθηση

Παρά την πολλά υποσχόμενη σχέση μεταξύ των ΗΦΥ και της xAI, εξακολουθούν να υπάρχουν αρκετές προκλήσεις:

- Ποιότητα δεδομένων και τυποποίηση: Η αποτελεσματικότητα των μοντέλων xAI εξαρτάται από την ποιότητα των δεδομένων που προέρχονται από τους ΗΦΥ. Η ασυνεπής καταχώρηση δεδομένων, η έλλειψη τυποποίησης και οι ελλιπείς εγγραφές μπορούν να εμποδίσουν την απόδοση και την επεξηγηματικότητα των μοντέλων.

- Πολυπλοκότητα των κλινικών δεδομένων: Η ετερογενής φύση των κλινικών δεδομένων δημιουργεί προκλήσεις για την ερμηνευσιμότητα των μοντέλων. Οι ΗΦΥ περιέχουν αδόμητα δεδομένα, όπως κλινικές σημειώσεις, τα οποία απαιτούν προηγμένες τεχνικές επεξεργασίας φυσικής γλώσσας για την εξαγωγή ουσιαστικών συμπερασμάτων.

- Ρυθμιστικά και ηθικά ζητήματα: Η ενσωμάτωση της ΤΝ στην υγειονομική περίθαλψη εγείρει ρυθμιστικές και ηθικές προκλήσεις, ιδίως όσον αφορά την ιδιωτικότητα των ασθενών και την ασφάλεια των δεδομένων. Η διασφάλιση της συμμόρφωσης με τους κανονισμούς κατά την αξιοποίηση των δεδομένων ΗΦΥ για μηχανική μάθηση είναι ένα πολύπλοκο έργο που απαιτεί προσεκτική εξέταση.[6]

2. Παρόμοιες μελέτες

2.1 Βιβλιογραφική ανασκόπηση

Οι Yu Cheng, Fei Wang et. al [10] στο επιστημονικό άρθρο με τίτλο «Risk Prediction with Electronic Health Records: A Deep Learning Approach» τονίζουν τη σημασία της αξιοποίησης τεχνικών βαθιάς μάθησης (deep learning) για την εξαγωγή πολύτιμων πληροφοριών από τους ΗΦΥ των ασθενών, αντιμετωπίζοντας παράλληλα τις προκλήσεις που σχετίζονται με την πολυπλοκότητα και τη μεταβλητότητα των δεδομένων σε περιβάλλοντα υγειονομικής περίθαλψης.

Το μοντέλο τους αναπαριστά τους ΗΦΥ των ασθενών ως χρονικούς πίνακες και χρησιμοποιεί ένα συνελικτικό νευρωνικό δίκτυο τεσσάρων επιπέδων για την εξαγωγή και πρόβλεψη φαινοτύπων. Διερευνούνται επίσης διαφορετικές στρατηγικές χρονικής συγχώνευσης για την ενσωμάτωση των χρονικών πληροφοριών στη διαδικασία μάθησης.

Η χρήση βαθιάς μάθησης και συνελικτικών νευρωνικών δικτύων προσφέρει σύμφωνα με τους ερευνητές μια πολλά υποσχόμενη προσέγγιση για την ανάλυση δεδομένων ΗΦΥ για την πρόβλεψη κινδύνου και την εξατομικευμένη υγειονομική περίθαλψη.

Όσον αφορά τα μοντέλα που χρησιμοποιήθηκαν, το μοντέλο Slow Fusion CNN (SF-CNN) επιτυγχάνει σταθερά την καλύτερη απόδοση σε διαφορετικά σύνολα δεδομένων. Το μοντέλο Basic CNN (BS-CNN) και το Temporal Early Fusion CNN (EF-CNN) παρουσιάζουν συγκρίσιμες επιδόσεις, ενώ το Late Fusion CNN (LF-CNN) έχει ελαφρώς καλύτερες επιδόσεις από το BS-CNN και το EF-CNN σε ορισμένες περιπτώσεις.

Για την πρόβλεψη της χρόνιας καρδιακής ανεπάρκειας, το SF-CNN επιδεικνύει την υψηλότερη απόδοση, βελτιώνοντας την ακρίβεια πρόβλεψης κατά 5.2% με 90% training data σε σύγκριση με το βασικό μοντέλο.

Για την πρόβλεψη της Χρόνιας Αποφρακτικής Πνευμονοπάθειας (ΧΑΠ), το SF-CNN υπερτερεί επίσης έναντι άλλων μοντέλων, αυξάνοντας την AUC πρόβλεψης κατά 5.3% σε σχέση με το βασικό μοντέλο. Το LF-CNN παρουσιάζει διαφορετική απόδοση ανάλογα με το σύνολο δεδομένων.

Όλα τα μοντέλα επωφελούνται από μεγαλύτερα σύνολα δεδομένων εκπαίδευσης, γεγονός που υποδεικνύει τη σημασία της συλλογής περισσότερων δεδομένων από τον πραγματικό κόσμο για ισχυρή απόδοση του μοντέλου. Τα μοντέλα που βασίζονται στα συνελικτικά νευρωνικά δίκτυα

παρουσιάζουν βελτιωμένη απόδοση με την αύξηση των δεδομένων εκπαίδευσης, γεγονός που υποδηλώνει επεκτασιμότητα και γενικευσιμότητα με μεγαλύτερα σύνολα δεδομένων.

Οι Boxwala, Aziz A., et al. [11] αξιολογούν την αποτελεσματικότητα στατιστικών μεθόδων και μεθόδων μηχανικής μάθησης στον εντοπισμό δυνητικά ακατάλληλης πρόσβασης σε ΗΦΥ.

Οι ερευνητές εξήγαγαν 26 χαρακτηριστικά από τα αρχεία καταγραφής πρόσβασης σε ΗΦΥ και άλλα οργανωτικά δεδομένα που συλλέχθηκαν κατά τη διάρκεια μιας περιόδου 2 μηνών για τον εντοπισμό ύποπτων προσβάσεων. Τα μοντέλα λογιστικής παλινδρόμησης (LR) και διανυσματικής μηχανής υποστήριξης (SVM) εκπαιδεύτηκαν σε 1291 συμβάντα χρησιμοποιώντας 10πλό cross-validation. Η ευαισθησία των τελικών μοντέλων αξιολογήθηκε σε ένα εξωτερικό σύνολο 58 συμβάντων που αναγνωρίστηκαν ως πραγματικά ακατάλληλα.

Τα μοντέλα LR και SVM επέδειξαν ισχυρές επιδόσεις, με περιοχή κάτω από την καμπύλη λειτουργικής χαρακτηριστικής του δέκτη 0.91 για το LR και 0.95 για το SVM στο σύνολο των δεδομένων. Όταν αξιολογήθηκαν σε 58 πραγματικά ακατάλληλα συμβάντα, η ευαισθησία ήταν 0.76 για το LR και 0.79 για το SVM, υπερβαίνοντας τη βασική μέθοδο.

Η δυνατότητα γενίκευσης των μοντέλων LR και SVM μπορεί να είναι περιορισμένη λόγω των διαφορών στις οργανωτικές δομές και τις ροές εργασίας μεταξύ των ιδρυμάτων. Ωστόσο, η προσέγγιση της χρήσης τεχνικών στατιστικής και μηχανικής μάθησης μπορεί να εφαρμοστεί ευρύτερα. Η μελέτη σημείωσε επίσης τον περιορισμό του σχετικά μικρού μεγέθους δείγματος για το σύνολο εκπαίδευσης λόγω της προσπάθειας που απαιτείται για τη δημιουργία του.

Τα ευρήματα υποδηλώνουν ότι οι στατιστικές μέθοδοι και οι μέθοδοι μηχανικής μάθησης μπορούν να είναι πολύτιμες για την υποβοήθηση των υπευθύνων προστασίας της ιδιωτικής ζωής στον εντοπισμό ύποπτων προσβάσεων στους ΗΦΥ, αναδεικνύοντας τον πιθανό ρόλο αυτών των τεχνικών στην ενίσχυση της ασφάλειας των δεδομένων και της ιδιωτικής ζωής σε περιβάλλοντα υγειονομικής περίθαλψης.

Οι Barak-Corren, Yuval, et al. [12] επικεντρώνονται στην πρόβλεψη της αυτοκτονικής συμπεριφοράς με τη χρήση διαχρονικών ΗΦΥ. Οι ερευνητές ανέπτυξαν μια προσέγγιση μοντελοποίησης με βάση τα δεδομένα που αξιοποιεί όλο το φάσμα των φαινοτυπικών δεδομένων που είναι διαθέσιμα στους ΗΦΥ, αντί να βασίζεται αποκλειστικά σε κοινώς χρησιμοποιούμενους παράγοντες κινδύνου.

Το προγνωστικό μοντέλο πέτυχε AUC 0.77 και επέδειξε ευαισθησία 45% με ποσοστό ψευδώς θετικών αποτελεσμάτων 10%, ξεπερνώντας σημαντικά τα απλούστερα μοντέλα που βασίζονται σε παραδοσιακούς παράγοντες κινδύνου, τα οποία είχαν χαμηλότερη ευαισθησία (π.χ. 29% μόνο για την κατάθλιψη).

Η θετική προγνωστική αξία του μοντέλου ήταν χαμηλή έως μέτρια (5%-27% με 95% specificity), γεγονός αναμενόμενο δεδομένης της χαμηλής αρχικής πιθανότητας αυτοκτονίας (1.2%). Ωστόσο, το μοντέλο παρείχε εμπλουτισμό κατά 4.5 έως 6.5 φορές στην πρόβλεψη του κινδύνου αυτοκτονίας σε σύγκριση με το βασικό ποσοστό.

Οι συγγραφείς προτείνουν ότι η εφαρμογή του μοντέλου σε συγκεκριμένους πληθυσμούς ασθενών με υψηλότερο κίνδυνο αυτοκτονίας, όπως αυτοί που βρίσκονται σε ψυχιατρική περίθαλψη, θα μπορούσε να βελτιώσει την προγνωστική του αξία. Η μελέτη αναδεικνύει τη δυνατότητα εντοπισμού τόσο των καθιερωμένων όσο και των προηγούμενων μη υποψιασμένων

παραγόντων κινδύνου με τη χρήση ολοκληρωμένων δεδομένων ΗΦΥ, γεγονός που ενισχύει την αποτελεσματικότητα του μοντέλου σε σύγκριση με τις παραδοσιακές μεθόδους.

Η μελέτη περιελάμβανε ασθενείς με τρεις ή περισσότερες επισκέψεις, με αποτέλεσμα ένα δείγμα 1.728.549 ατόμων. Αναπτύχθηκαν χωριστά μοντέλα για άνδρες και γυναίκες για να ληφθούν υπόψη οι διαφορές μεταξύ των δύο φύλων. Η κοόρτη χωρίστηκε σε σύνολα εκπαίδευσης και δοκιμών και χρησιμοποιήθηκαν μοντέλα ταξινομητών naïve Bayesian για την εκτίμηση του κινδύνου αυτοκτονικής συμπεριφοράς. Η μέθοδος αυτή πλεονεκτεί λόγω της επεκτασιμότητάς της και της ικανότητάς της να χειρίζεται πολλές ανεξάρτητες μεταβλητές.

Η απόδοση των μοντέλων αξιολογήθηκε αναδρομικά χρησιμοποιώντας ένα ανεξάρτητο σύνολο δοκιμών, εστιάζοντας στην ευαισθησία, την ειδικότητα και την ικανότητα πρόβλεψης της αυτοκτονικής συμπεριφοράς εκ των προτέρων.

Αυτές οι μέθοδοι είχαν συλλογικά ως στόχο τη δημιουργία ενός ισχυρού προγνωστικού μοντέλου που θα μπορούσε να χρησιμεύσει ως εργαλείο υποστήριξης κλινικών αποφάσεων για τον εντοπισμό ατόμων με υψηλό κίνδυνο αυτοκτονίας. Το μοντέλο ήταν σε θέση να προβλέψει την αυτοκτονική συμπεριφορά κατά μέσο όρο 3 έως 4 χρόνια νωρίτερα, γεγονός που είναι σημαντικό για την έγκαιρη παρέμβαση.

Το μοντέλο εντόπισε τόσο γνωστούς παράγοντες κινδύνου (όπως η κατάχρηση ουσιών και οι ψυχιατρικές διαταραχές) όσο και λιγότερο συμβατικούς παράγοντες (όπως ορισμένοι τραυματισμοί και χρόνιες παθήσεις), υποδεικνύοντας μια ολοκληρωμένη προσέγγιση για τη δημιουργία προφίλ κινδύνου.

Στο επιστημονικό άρθρο των Deo et. al [13] εξετάζεται η εφαρμογή της μηχανικής μάθησης στην πρόβλεψη της καρδιακής ανεπάρκειας (HF) με τη χρήση ΗΦΥ. Δίνει έμφαση στα δυνητικά οφέλη και τις προκλήσεις που σχετίζονται με την αξιοποίηση των μεγάλων δεδομένων στην υγειονομική περίθαλψη.

Οι συγγραφείς ξεκινούν τονίζοντας τη σημασία της κατανόησης των διαφορών μεταξύ ασθενών με και χωρίς συγκεκριμένες ασθένειες, καθώς οι διαφορές αυτές μπορούν να αποκαλύψουν τροποποιήσιμους παράγοντες κινδύνου και να βελτιώσουν τις στρατηγικές θεραπείας. Κατά την τελευταία δεκαετία, έχουν δημοσιευτεί πάνω από 28 μοντέλα πρόβλεψης για την καρδιακή ανεπάρκεια, τα οποία αποδεικνύουν καλές διακριτικές ιδιότητες, με στατιστικά στοιχεία C που κυμαίνονται από 0.70 έως 0.89. Αυτά τα μοντέλα χρησιμοποιούν συνήθως έναν μέτριο αριθμό προγνωστικών παραγόντων (λιγότερους από 15) και συχνά βασίζονται σε γραμμικά μοντέλα.

Ένα από τα κρίσιμα βήματα της μελέτης ήταν ο καθορισμός της μεταβλητής απόκρισης, όπου οι συγγραφείς επέλεξαν τη δυαδική ταξινόμηση των ασθενών ως έχοντες ή όχι καρδιακή ανεπάρκεια. Η επιλογή αυτή έγινε για να ξεπεραστούν οι περιορισμοί που σχετίζονται με τη διαθεσιμότητα αλγορίθμων μηχανικής μάθησης για αναλύσεις time-to-event. Οι συγγραφείς ασχολήθηκαν επίσης με την πρόκληση του χειρισμού των προγνωστικών χαρακτηριστικών, σημειώνοντας ότι τα ακατέργαστα χαρακτηριστικά συχνά δεν μπορούν να εισαχθούν απευθείας σε μοντέλα λόγω ζητημάτων όπως η σπανιότητα και η διαστατικότητα. Για να μετριάσουν αυτά τα ζητήματα, ομαδοποίησαν τα χαρακτηριστικά σε «υπερ-χαρακτηριστικά», μειώνοντας τον χώρο των χαρακτηριστικών από δεκάδες χιλιάδες σε μερικές εκατοντάδες.

Στη μελέτη χρησιμοποιήθηκαν δύο καθιερωμένοι αλγόριθμοι: η παραλλαγή lasso της λογιστικής παλινδρόμησης και τα Random Forests. Ενώ τα Random Forests υπερέχαν ελαφρώς της λογιστικής παλινδρόμησης lasso, η τελευταία υπερέχει λόγω του χαμηλότερου υπολογιστικού

κόστους της. Η απόδοση του μοντέλου, με περιοχή κάτω από την καμπύλη λειτουργικής χαρακτηριστικής του δέκτη ήταν περίπου 0.79 και ήταν συγκρίσιμη με τα υπάρχοντα μοντέλα, αλλά δεν τα ξεπέρασε σημαντικά. Οι συγγραφείς κατέληξαν στο συμπέρασμα ότι, ενώ οι προσεγγίσεις μηχανικής μάθησης μπορεί να μην βελτιώνουν δραστικά την προγνωστική απόδοση, προσφέρουν πλεονεκτήματα όσον αφορά τη σχέση κόστους-αποτελεσματικότητας και την ενσωμάτωση στην κλινική λήψη αποφάσεων.

Η εγκυρότητα των προγνωστικών μοντέλων εξαρτάται σε μεγάλο βαθμό από τον ακριβή ορισμό των περιπτώσεων και των ελέγχων. Επισημαίνουν ότι οι ηλεκτρονικοί φάκελοι υγείας, αν και χρήσιμοι, μπορεί να μην παρέχουν πάντα μια ολοκληρωμένη αναπαράσταση των διαγνώσεων ενός ασθενούς λόγω των περιορισμών της διαγνωστικής κωδικοποίησης. Ο κίνδυνος ψευδώς αρνητικών αποτελεσμάτων είναι ιδιαίτερα ανησυχητικός, καθώς θα μπορούσε να οδηγήσει σε λανθασμένη ταξινόμηση ασθενών που έχουν ήδη καρδιακή ανεπάρκεια.

Το άρθρο των Rahimian, Salimi-Khorshidi et al. [14] παρουσιάζει μια μελέτη που επικεντρώνεται στην πρόβλεψη του κινδύνου επείγουσών εισαγωγών σε νοσοκομεία με τη χρήση τεχνικών μηχανικής μάθησης που εφαρμόζονται σε ΗΦΥ. Η μελέτη αποσκοπεί στη βελτίωση του εντοπισμού των ασθενών που διατρέχουν υψηλό κίνδυνο επείγουσας εισαγωγής, γεγονός που μπορεί να βοηθήσει τους παρόχους υγειονομικής περίθαλψης να εφαρμόσουν προληπτικά μέτρα και να βελτιστοποιήσουν την κατανομή των πόρων.

Η έρευνα χρησιμοποίησε ένα σύνολο δεδομένων που περιλάμβανε πάνω από 4 εκατομμύρια ασθενείς από γενικά ιατρεία στο Ηνωμένο Βασίλειο, με ημερομηνία βάσης την 1η Ιανουαρίου 2010. Το πρωταρχικό αποτέλεσμα ενδιαφέροντος ήταν η πρώτη επείγουσα εισαγωγή στο νοσοκομείο μετά από αυτή την ημερομηνία βάσης, η οποία παρακολουθήθηκε σε διάφορα χρονικά παράθυρα (12, 24, 36, 48 και 60 μήνες). Η μελέτη χρησιμοποίησε μια σειρά μεταβλητών πρόβλεψης, συμπεριλαμβανομένων δημογραφικών πληροφοριών, κλινικών διαγνώσεων, προηγούμενης χρήσης υγειονομικής περίθαλψης και αποτελεσμάτων εργαστηριακών εξετάσεων.

Για την ανάπτυξη προγνωστικών μοντέλων, οι ερευνητές αναπαρήγαγαν αρχικά ένα μοντέλο Cox Proportional Hazards (CPH) ως σημείο αναφοράς. Στη συνέχεια συνέκριναν αυτό το μοντέλο με δύο προσεγγίσεις μηχανικής μάθησης: Gradient Boosting Classifier και Random Forest. Και τα δύο μοντέλα είναι μέθοδοι συνόλου που βασίζονται σε δέντρα απόφασης, με τον GBC να επικεντρώνεται στην ενίσχυση των αδύναμων μαθητών διαδοχικά και τον RF να χρησιμοποιεί τεχνικές bagging για να ενισχύσει την ακρίβεια πρόβλεψης. Η επιλογή αυτών των μοντέλων βασίστηκε στην αποδεδειγμένη αποτελεσματικότητά τους στο χειρισμό μεγάλων συνόλων δεδομένων και στην ευρωστία τους σε διάφορες εφαρμογές.

Τα ευρήματα της μελέτης έδειξαν ότι τα μοντέλα μηχανικής μάθησης υπερέιχαν σημαντικά έναντι των παραδοσιακών στατιστικών μεθόδων όσον αφορά τη διάκριση και τη βαθμονόμηση για την πρόβλεψη επείγουσών εισαγωγών. Τα μοντέλα διατήρησαν σταθερή απόδοση σε διάφορα χρονικά παράθυρα πρόβλεψης και επικυρώθηκαν χρησιμοποιώντας ένα ξεχωριστό σύνολο δεδομένων από γεωγραφικά διαφορετικές περιοχές, εξασφαλίζοντας τη γενικευσιμότητα των αποτελεσμάτων.

Οι βασικές μεταβλητές που επηρέασαν την πρόβλεψη περιλάμβαναν προηγούμενες εισαγωγές σε επείγοντα περιστατικά, τον αριθμό επισκέψεων σε γενικούς ιατρούς, κλινικές διαγνώσεις (όπως διαβήτης, καρδιαγγειακά νοσήματα και ψυχικές καταστάσεις) και διάφορα κλινικά μέτρα (όπως η αρτηριακή πίεση και τα επίπεδα χοληστερόλης). Η συμπερίληψη χρονικών

πληροφοριών, όπως ο χρόνος από την τελευταία εμφάνιση ενός συμβάντος ή μιας διάγνωσης, ενίσχυσε περαιτέρω τις προγνωστικές ικανότητες των μοντέλων.

Η μελέτη ανέδειξε επίσης τη σημασία της διαστρωμάτωσης των αποτελεσμάτων ανάλογα με το φύλο, αν και δεν διαπιστώθηκαν σημαντικές διαφορές, με αποτέλεσμα να γίνει συνδυασμένη ανάλυση για λόγους συντομίας. Οι ερευνητές υπογράμμισαν ότι τα βελτιωμένα μοντέλα πρόβλεψης θα μπορούσαν να καθοδηγήσουν την πολιτική και την πρακτική της υγειονομικής περίθαλψης, μειώνοντας ενδεχομένως την επιβάρυνση των μη προγραμματισμένων εισαγωγών και βελτιώνοντας τα αποτελέσματα των ασθενών.

Οι Steele, Denaxas et. al [15] παρουσιάζουν μια μελέτη που διερευνά την αποτελεσματικότητα των μοντέλων μηχανικής μάθησης σε σύγκριση με τα παραδοσιακά μοντέλα επιβίωσης στην πρόβλεψη της θνησιμότητας από κάθε αιτία με τη χρήση ηλεκτρονικών αρχείων υγείας (ΗΦΥ). Η μελέτη βασίζεται σε ένα μεγάλο σύνολο δεδομένων από το πρόγραμμα CALIBER, το οποίο περιλαμβάνει κλινικά δεδομένα από περισσότερους από 80.000 ασθενείς με στεφανιαία νόσο. Η έρευνα αποσκοπεί στην αντιμετώπιση των περιορισμών των συμβατικών στατιστικών μεθόδων στο χειρισμό δεδομένων υψηλής διάστασης και ελλειπουσών τιμών, οι οποίες είναι συχνές στα ΗΦΥ.

Οι συγγραφείς τονίζουν ότι τα παραδοσιακά μοντέλα επιβίωσης, όπως το μοντέλο αναλογικών κινδύνων Cox, έχουν χρησιμοποιηθεί ευρέως σε κλινικές ρυθμίσεις, αλλά συχνά απαιτούν πλήρη δεδομένα και σημαντική συμβολή εμπειρογνομόνων για την κατασκευή τους. Αυτά τα μοντέλα ενδέχεται να μην αξιοποιούν πλήρως τον πλούτο των διαθέσιμων δεδομένων, οδηγώντας σε πιθανές μεροληψίες και περιορισμούς στη γενικευσιμότητα. Αντίθετα, οι τεχνικές μηχανικής μάθησης, ιδίως τα τυχαία δάση και άλλοι αλγόριθμοι, μπορούν να χειριστούν πιο αποτελεσματικά σύνολα δεδομένων υψηλής διάστασης και ελλείπουσες τιμές, οδηγώντας ενδεχομένως σε βελτιωμένη προβλεπτική απόδοση.

Η μελέτη χρησιμοποιεί διάφορους αλγορίθμους μηχανικής μάθησης, συμπεριλαμβανομένων των τυχαίων δασών, για τη μοντελοποίηση του χρόνου μέχρι την ολική θνησιμότητα. Η απόδοση αυτών των μοντέλων αξιολογείται με βάση τη διάκριση και τη βαθμονόμηση. Η διάκριση αναφέρεται στην ικανότητα του μοντέλου να διακρίνει μεταξύ των ασθενών που θα βιώσουν το γεγονός (θνησιμότητα) και εκείνων που δεν θα βιώσουν, ενώ η βαθμονόμηση αξιολογεί πόσο καλά οι προβλεπόμενες πιθανότητες ευθυγραμμίζονται με τα πραγματικά αποτελέσματα.

Τα αποτελέσματα δείχνουν ότι ενώ τα μοντέλα μηχανικής μάθησης, ιδίως τα τυχαία δάση, επιδεικνύουν συγκρίσιμες επιδόσεις διάκρισης με τα καλύτερα μοντέλα Cox, συχνά δυσκολεύονται με τη βαθμονόμηση. Η μελέτη διαπίστωσε ότι τα τυχαία δάση υπολείπονταν του μοντέλου Cox όσον αφορά τη βαθμονόμηση, γεγονός που υποδηλώνει ότι ενώ μπορούν να εντοπίσουν ασθενείς υψηλού κινδύνου, οι προβλεπόμενες πιθανότητες μπορεί να μην είναι αξιόπιστες. Αυτή η ασυμφωνία εγείρει ανησυχίες σχετικά με την κλινική εφαρμοσιμότητα των μοντέλων μηχανικής μάθησης, καθώς οι ακριβείς εκτιμήσεις πιθανοτήτων είναι ζωτικής σημασίας για την ενημέρωση των αποφάσεων θεραπείας.

Οι συγγραφείς συζητούν επίσης τις προκλήσεις που σχετίζονται με την ερμηνεία των μοντέλων μηχανικής μάθησης. Τα μοντέλα υψηλής διάστασης μπορεί να είναι πολύπλοκα και δύσκολα κατανοητά, καθιστώντας δύσκολη την αιτιολόγηση αποφάσεων που βασίζονται σε αυτά τα μοντέλα από τους κλινικούς γιατρούς. Αντίθετα, τα παραδοσιακά μοντέλα, ενώ είναι πιο ερμηνεύσιμα, ενδέχεται να μην αξιοποιούν πλήρως τις δυνατότητες των δεδομένων. Η μελέτη υπογραμμίζει την ανάγκη για ισορροπία μεταξύ της πολυπλοκότητας των μοντέλων και της

ερμηνευσιμότητας, ώστε να διασφαλιστεί ότι οι κλινικοί γιατροί μπορούν να εμπιστεύονται και να χρησιμοποιούν αποτελεσματικά αυτά τα μοντέλα στην πράξη.

Επιπλέον, η έρευνα ασχολείται με το ζήτημα των ελλιπών δεδομένων, το οποίο είναι διαδεδομένο στους ΗΦΥ. Οι συγγραφείς διερευνούν διάφορες τεχνικές υπολογισμού για τον χειρισμό των ελλιπών τιμών και αξιολογούν τον αντίκτυπό τους στην απόδοση του μοντέλου. Σημειώνουν ότι ο μη υπολογισμός των ελλιπών τιμών μπορεί να περιπλέξει την ερμηνεία των συντελεστών του μοντέλου, καθώς οι συντελεστές αυτοί μπορεί να αντανakλούν τους λόγους της έλλειψης.

Η μελέτη καταλήγει στο συμπέρασμα ότι ενώ τα μοντέλα μηχανικής μάθησης έχουν τη δυνατότητα να υπερτερούν των παραδοσιακών μοντέλων επιβίωσης σε ορισμένες πτυχές, παραμένουν προκλήσεις όσον αφορά τη βαθμονόμηση, την ερμηνευσιμότητα και τον χειρισμό των ελλιπών δεδομένων. Οι συγγραφείς υποστηρίζουν την περαιτέρω έρευνα για την αναπαραγωγή των ευρημάτων σε διάφορα περιβάλλοντα και την τελειοποίηση των τεχνικών μηχανικής μάθησης για κλινική χρήση. Προτείνουν ότι η ενσωμάτωση της μηχανικής μάθησης με τις παραδοσιακές μεθόδους μπορεί να προσφέρει μια πολλά υποσχόμενη προσέγγιση για την ενίσχυση της προγνωστικής μοντελοποίησης στην υγειονομική περίθαλψη.

Η έρευνα που διεξήχθη από τους Mohamadlou et al. [16] επικεντρώνεται στην ανάπτυξη και την αξιολόγηση ενός αλγορίθμου μηχανικής μάθησης (MLA) που έχει σχεδιαστεί για την πρόβλεψη της οξείας νεφρικής βλάβης (ONB) σε νοσηλευόμενους ασθενείς με τη χρήση δεδομένων ηλεκτρονικού φακέλου υγείας (ΗΦΥ). Η έρευνα στοχεύει στην ενίσχυση της έγκαιρης ανίχνευσης της ONB, η οποία είναι ζωτικής σημασίας για την έγκαιρη παρέμβαση και τη βελτίωση των αποτελεσμάτων των ασθενών.

Οι συγγραφείς χρησιμοποίησαν δεδομένα από δύο μεγάλα ιατρικά κέντρα: Beth Israel Deaconess Medical Center (BIDMC) και Stanford University Medical Center. Η μελέτη περιελάμβανε συνολικά 68.319 ασθενείς, με 48.582 από το BIDMC και 19.737 από το Stanford. Ο πληθυσμός των ασθενών περιορίστηκε σε άτομα ηλικίας 18 ετών και άνω και η συλλογή δεδομένων τηρούσε τους κανονισμούς HIPAA, διασφαλίζοντας το απόρρητο των ασθενών.

Το MLA που χρησιμοποιήθηκε σε αυτή τη μελέτη βασίζεται σε δέντρα ενισχυμένης κλίσης (gradient boosted trees), μια εξελιγμένη τεχνική μηχανικής μάθησης που συνδυάζει πολλαπλά δέντρα απόφασης για να δημιουργήσει ένα σκορ πρόβλεψης κινδύνου. Ο αλγόριθμος σχεδιάστηκε για να προβλέπει την έναρξη της AKI 12, 24, 48 και 72 ώρες νωρίτερα. Οι ερευνητές συνέκριναν την απόδοση του MLA με τη βαθμολογία Sequential Organ Failure Assessment (SOFA), ένα ευρέως χρησιμοποιούμενο κλινικό εργαλείο για την αξιολόγηση της λειτουργίας των οργάνων και την πρόβλεψη του κινδύνου AKI.

Τα αποτελέσματα έδειξαν ότι η MLA υπερείχε της βαθμολογίας SOFA σε διάφορες μετρήσεις, συμπεριλαμβανομένης της περιοχής κάτω από τη χαρακτηριστική καμπύλη λειτουργίας δέκτη (AUROC), της ακρίβειας και του διαγνωστικού λόγου πιθανοτήτων (DOR) για την ανίχνευση σοβαρής AKI. Αυτό υποδηλώνει ότι η MLA θα μπορούσε να παρέχει μια πιο αξιόπιστη και έγκαιρη πρόβλεψη της AKI, επιτρέποντας στους κλινικούς γιατρούς να εφαρμόζουν προληπτικά μέτρα όπως η αναζωογόνηση όγκου ή η αποφυγή νεφροτοξικών φαρμάκων.

Παρά τα πολλά υποσχόμενα αποτελέσματα, η μελέτη αναγνωρίζει αρκετούς περιορισμούς. Καθώς πρόκειται για αναδρομική ανάλυση, τα ευρήματα ενδέχεται να μην μεταφραστούν πλήρως σε πραγματικές κλινικές ρυθμίσεις. Ο αλγόριθμος δοκιμάστηκε μόνο σε ενήλικες

ασθενείς από τα δύο συγκεκριμένα νοσοκομεία, γεγονός που μπορεί να περιορίσει τη γενίκευσή του σε άλλους πληθυσμούς ή ρυθμίσεις. Επιπλέον, η μελέτη δεν έλαβε υπόψη διάφορους παράγοντες, όπως η αιτία εισαγωγής ή οι συννοσηρότητες των ασθενών, οι οποίοι θα μπορούσαν να επηρεάσουν την αποτελεσματικότητα του αλγορίθμου σε διαφορετικούς υποπληθυσμούς.

Οι συγγραφείς σημείωσαν επίσης ότι η προγνωστική απόδοση του αλγορίθμου θα μπορούσε να διαφέρει ανάλογα με τους ορισμούς της AKI που χρησιμοποιούνται σε διάφορα κλινικά περιβάλλοντα. Η εξάρτηση της μελέτης από αποπροσδιορισμένα δεδομένα σήμαινε ότι δεν απαιτήθηκαν δεοντολογικές εγκρίσεις, αλλά περιορίσε επίσης τη δυνατότητα διερεύνησης των αποχρώσεων της φροντίδας των ασθενών και των αποτελεσμάτων με μεγαλύτερη λεπτομέρεια.

Η εργασία των Xiao, Choi, Sun [17] παρουσιάζει μια συστηματική ανασκόπηση των μοντέλων βαθιάς μάθησης που εφαρμόζονται σε δεδομένα ηλεκτρονικών αρχείων υγείας (ΗΦΥ), υπογραμμίζοντας τις πρόσφατες εξελίξεις, τις προκλήσεις και τις μελλοντικές ερευνητικές κατευθύνσεις. Οι συγγραφείς διεξήγαγαν μια ολοκληρωμένη βιβλιογραφική αναζήτηση σε πολλαπλές βάσεις δεδομένων, συμπεριλαμβανομένων των PubMed και Google Scholar, εστιάζοντας σε μελέτες που δημοσιεύθηκαν μεταξύ Ιανουαρίου 2010 και Ιανουαρίου 2018. Από την αναζήτηση αυτή προέκυψαν 361 άρθρα, τα οποία φιλτράρισαν σε 98 σχετικές μελέτες μετά την αφαίρεση των διπλοτύπων και την αξιολόγηση της συνάφειας του θέματος.

Η ανασκόπηση κατηγοριοποιεί τα επιλεγμένα άρθρα με βάση τρεις κύριες πτυχές: τον τόπο δημοσίευσης (ιατρική, πληροφορική ή επιστήμη των υπολογιστών), τη χρήση δεδομένων EXAE και τις συγκεκριμένες εργασίες, τα μοντέλα και τις μετρήσεις επιδόσεων που αναφέρθηκαν. Οι συγγραφείς εντόπισαν πέντε κύριες κατηγορίες εργασιών ανάλυσης στη βιβλιογραφία:

- Ανίχνευση/ταξινόμηση ασθενειών: Πρόκειται για τον εντοπισμό του κατά πόσον συγκεκριμένες ασθένειες μπορούν να επιβεβαιωθούν με τη χρήση δεδομένων ΗΦΥ.
- Διαδοχική πρόβλεψη κλινικών συμβάντων: Αυτή η εργασία επικεντρώνεται στην πρόβλεψη μελλοντικών κλινικών συμβάντων με βάση ιστορικά δεδομένα.
- Ενσωμάτωση εννοιών: Αφορά την εξαγωγή αναπαραστάσεων χαρακτηριστικών κλινικών εννοιών ή φαινοτύπων από δεδομένα ΗΦΥ.
- Ενίσχυση δεδομένων: Πρόκειται για τη δημιουργία ρεαλιστικών αρχείων ασθενών με βάση υπάρχοντα δεδομένα ΗΦΥ.
- Απόρρητο δεδομένων ΗΦΥ: Αυτή η κατηγορία αφορά τις προκλήσεις της διατήρησης του απορρήτου των ασθενών κατά τη χρήση δεδομένων ΗΦΥ για έρευνα.

Η ανασκόπηση δίνει έμφαση στις μοναδικές προκλήσεις που θέτουν τα δεδομένα ΗΦΥ, οι οποίες περιλαμβάνουν ζητήματα που σχετίζονται με την ποιότητα των δεδομένων, όπως τα ελλιπή ή ακανόνιστα δειγματοληπτικά δεδομένα, και την πολυπλοκότητα των κλινικών συμβάντων που μπορεί να διαφέρουν σημαντικά σε διαφορετικούς πληθυσμούς ασθενών. Οι συγγραφείς σημειώνουν ότι, ενώ τα μοντέλα βαθιάς μάθησης έχουν δείξει υποσχέσεις σε διάφορες εφαρμογές, δεν υπερτερούν πάντα έναντι των παραδοσιακών μοντέλων μηχανικής μάθησης, ιδίως όταν τα μεγέθη των δειγμάτων είναι μικρά.

Στο άρθρο εξετάζονται επίσης οι τύποι αρχιτεκτονικών βαθιάς μάθησης που χρησιμοποιήθηκαν στις μελέτες, συμπεριλαμβανομένων των επαναλαμβανόμενων νευρωνικών δικτύων (RNN) και των νευρωνικών δικτύων συνέλιξης (CNN). Η απόδοση αυτών των μοντέλων αξιολογείται συχνά με τη χρήση μετρικών όπως η περιοχή κάτω από την καμπύλη λειτουργικής χαρακτηριστικής

καμπύλης δέκτη (AUC), με πολλές μελέτες να αναφέρουν τιμές AUC γύρω στο 0.8 ή υψηλότερες για συγκεκριμένες εργασίες.

Εκτός από τη συνοπτική παρουσίαση της τρέχουσας κατάστασης της έρευνας, οι συγγραφείς εντοπίζουν αρκετές ανοιχτές προκλήσεις και πιθανές μελλοντικές κατευθύνσεις για τη βαθιά μάθηση στην υγειονομική περίθαλψη. Αυτές περιλαμβάνουν την ανάγκη για υψηλής απόδοσης, φορητές υπολογιστικές μεθόδους για κλινικό φαινότυπο, την ενσωμάτωση διαφορετικών τύπων δεδομένων (όπως απεικονιστικά και γενετικά δεδομένα) με δεδομένα ΗΦΥ και την ανάπτυξη ισχυρών μοντέλων που μπορούν να χειριστούν τον εγγενή θόρυβο και την αραιότητα των κλινικών δεδομένων.

Η ανασκόπηση καταλήγει τονίζοντας τη σημασία της διεπιστημονικής συνεργασίας μεταξύ κλινικών ιατρών, επιστημόνων δεδομένων και ερευνητών για την προώθηση του πεδίου της βαθιάς μάθησης στην υγειονομική περίθαλψη. Με την αντιμετώπιση των εντοπισμένων προκλήσεων και την αξιοποίηση των πλεονεκτημάτων της βαθιάς μάθησης, οι συγγραφείς πιστεύουν ότι μπορούν να επιτευχθούν σημαντικές βελτιώσεις στη λήψη κλινικών αποφάσεων, στα αποτελέσματα των ασθενών και στη συνολική παροχή υγειονομικής περίθαλψης.

Οι Mehta N, Devarakonda MV [18] εξετάζουν την ενσωμάτωση της τεχνητής νοημοσύνης (AI), ιδίως της μηχανικής μάθησης (ML) και της επεξεργασίας φυσικής γλώσσας (NLP), στην υγειονομική περίθαλψη, δίνοντας έμφαση στις δυνατότητές τους να βελτιώσουν τη φροντίδα των ασθενών και να εξορθολογήσουν τις κλινικές ροές εργασίας. Επισημαίνει τον αυξανόμενο όγκο δεδομένων που παράγονται μέσω των ηλεκτρονικών φακέλων υγείας (ΗΦΥ) και της ψηφιακής απεικόνισης, τα οποία μπορούν να αξιοποιηθούν για τη βελτίωση της κλινικής λήψης αποφάσεων και των αποτελεσμάτων των ασθενών.

Οι συγγραφείς αρχίζουν με την περιγραφή των δυνατοτήτων του NLP και της ML. Το NLP επιτρέπει στους υπολογιστές να επεξεργάζονται και να κατανοούν την ανθρώπινη γλώσσα, επιτρέποντας την ανάλυση κλινικών σημειώσεων και αναφορών ελεύθερου κειμένου. Αυτό είναι ζωτικής σημασίας, καθώς πολλές από τις πολύτιμες πληροφορίες στην υγειονομική περίθαλψη είναι αδόμητες. Το ML, από την άλλη πλευρά, περιλαμβάνει αλγορίθμους που μαθαίνουν από δεδομένα, οι οποίοι μπορούν να κατηγοριοποιηθούν σε επιβλεπόμενη και μη επιβλεπόμενη μάθηση. Η μάθηση με επίβλεψη χρησιμοποιεί σύνολα δεδομένων με ετικέτες για την εκπαίδευση αλγορίθμων, ενώ η μάθηση χωρίς επίβλεψη εντοπίζει μοτίβα σε δεδομένα χωρίς προκαθορισμένες ετικέτες.

Το άρθρο παραπέμπει στο Unified Medical Language System (UMLS) της Εθνικής Βιβλιοθήκης Ιατρικής, το οποίο παρέχει έναν ολοκληρωμένο ιατρικό μεταθESAυρό και εργαλεία για την αντιστοίχιση κειμένου σε ιατρικές έννοιες. Αυτός ο πόρος είναι ζωτικής σημασίας για την προώθηση των εφαρμογών NLP στην υγειονομική περίθαλψη, επιτρέποντας τον εντοπισμό ασθενειών, φαρμάκων και συμπτωμάτων μέσα σε κλινικά κείμενα. Οι συγγραφείς αναφέρουν επίσης την αρχιτεκτονική διαχείρισης μη δομημένων πληροφοριών (UIMA), που αναπτύχθηκε από την IBM, η οποία διευκολύνει την ανάλυση μη δομημένων δεδομένων. Το σύστημα κλινικής ανάλυσης κειμένου και εξαγωγής γνώσης (cTAKES) της Mayo Clinic επισημαίνεται ως μια επιτυχημένη εφαρμογή της UIMA, η οποία επιτρέπει την εξαγωγή σχετικών πληροφοριών από τα ΗΦΥ.

Σημαντική έμφαση στο άρθρο δίνεται στις δυνατότητες της τεχνητής νοημοσύνης για την αυτοματοποίηση των γνωστικών εργασιών που παραδοσιακά εκτελούνται από τους κλινικούς ιατρούς. Οι συγγραφείς περιγράφουν ένα σύστημα που παρακολουθεί δεδομένα ασθενών,

όπως τα επίπεδα κρεατινίνης, παράλληλα με συμβάντα που επηρεάζουν τη νεφρική λειτουργία. Το σύστημα αυτό μπορεί να εξορύσσει την ιατρική βιβλιογραφία για να παραμένει ενημερωμένο σχετικά με τα αίτια της νεφρικής βλάβης και να παρουσιάζει σχετικές κλινικές σημειώσεις στους παρόχους υγειονομικής περίθαλψης. Τέτοιες δυνατότητες θα μπορούσαν να μειώσουν το γνωστικό φορτίο των κλινικών ιατρών, επιτρέποντάς τους να επικεντρωθούν στη φροντίδα των ασθενών και όχι στα διοικητικά καθήκοντα.

Το άρθρο εξετάζει επίσης τις επιπτώσεις των εξελίξεων της τεχνητής νοημοσύνης για τους επαγγελματίες του τομέα της υγειονομικής περίθαλψης. Αυτοματοποιώντας τα τετριμμένα καθήκοντα ανάκτησης πληροφοριών και τεκμηρίωσης, η ΤΝ μπορεί να συμβάλει στη μείωση της εξουθένωσης των κλινικών ιατρών και στη βελτίωση της αποτελεσματικότητας. Οι συγγραφείς οραματίζονται ένα μέλλον όπου δεδομένα σε πραγματικό χρόνο από διάφορες πηγές, συμπεριλαμβανομένης της γονιδιωματικής, της πρωτεομικής και των ατομικών αισθητήρων υγείας, μπορούν να ενσωματωθούν για την αντιμετώπιση πολύπλοκων αναγκών υγειονομικής περίθαλψης. Αυτό θα μπορούσε να οδηγήσει σε εξελίξεις στην προγνωστική μοντελοποίηση, την ιατρική ακριβείας και τη διαχείριση της υγείας του πληθυσμού.

Το επιστημονικό άρθρο των Garriga et. al [19] εξετάζει μια μελέτη που επικεντρώθηκε στη βελτίωση της διαχείρισης κρίσεων ψυχικής υγείας μέσω της χρήσης αλγορίθμων μηχανικής μάθησης που εφαρμόζονται σε ηλεκτρονικά αρχεία υγείας (ΗΦΥ). Η μελέτη υπογραμμίζει τη σημαντική παγκόσμια επιβάρυνση από τις διαταραχές της ψυχικής υγείας, οι οποίες επηρεάζουν σχεδόν 1 δισεκατομμύριο ανθρώπους, και τονίζει την επιδείνωση αυτών των ζητημάτων λόγω της πανδημίας COVID-19. Οι συγγραφείς υποστηρίζουν ότι η έγκαιρη παρέμβαση είναι ζωτικής σημασίας για την πρόληψη κρίσεων ψυχικής υγείας, οι οποίες συχνά οδηγούν σε καταστάσεις επείγουσας περίθαλψης.

Η έρευνα αποσκοπεί στην ανάπτυξη ενός συστήματος προγνωστικής βαθμολογίας κινδύνου (PRS), το οποίο μπορεί να εντοπίζει ασθενείς που διατρέχουν κίνδυνο επιδείνωσης, επιτρέποντας έτσι προληπτικές παρεμβάσεις. Η μελέτη εξήγαγε 198 χαρακτηριστικά από τους ΗΦΥ, συμπεριλαμβανομένων των κωδικοποιημένων διαγνώσεων, των πρόσφατων αξιολογήσεων και του χρόνου που έχει παρέλθει από τα σημαντικά γεγονότα, για να δημιουργήσει ένα ολοκληρωμένο σύνολο δεδομένων για ανάλυση. Δοκιμάστηκαν διάφορες τεχνικές μηχανικής μάθησης, συμπεριλαμβανομένων δέντρων απόφασης, πιθανοτικών μοντέλων, μεθόδων συνόλου και ταξινομητών βαθιάς μάθησης. Μεταξύ αυτών, η μέθοδος XGBoost (eXtreme Gradient Boosting) επέδειξε ανώτερη απόδοση, ξεπερνώντας σημαντικά τις άλλες μεθόδους στην πρόβλεψη του κινδύνου των ασθενών.

Η εφαρμογή του αλγορίθμου αξιολογήθηκε μέσω ημιδομημένων συνεντεύξεων με κλινικές ομάδες, αποκαλύπτοντας ότι το 64% των προβλέψεων κρίθηκε χρήσιμο για τον εντοπισμό της επιδείνωσης του ασθενούς και τη διευκόλυνση έγκαιρων παρεμβάσεων. Ειδικότερα, ο αλγόριθμος αναφέρθηκε ως κλινικά πολύτιμος για την πρόληψη κρίσεων στο 19% των περιπτώσεων και για τον εντοπισμό επιδεινούμενων καταστάσεων στο 17% των περιπτώσεων. Ωστόσο, μία κοινοτική ομάδα ψυχικής υγείας (CMHT) δεν ανέφερε καμία προστιθέμενη αξία από τον αλγόριθμο, γεγονός που υποδεικνύει τη διακύμανση της αντιλαμβανόμενης αποτελεσματικότητάς του σε διαφορετικές ομάδες.

Η μελέτη εξετάζει επίσης τις ηθικές εκτιμήσεις σχετικά με τη χρήση ευαίσθητων δεδομένων ασθενών. Λόγω της ευπάθειας των εμπλεκόμενων πληθυσμών, τα δεδομένα του ΗΦΥ δεν μπορούν να δημοσιοποιηθούν και τα αιτήματα πρόσβασης πρέπει να εγκρίνονται από την

αρμόδια επιτροπή διακυβέρνησης. Ο κώδικας που αναπτύχθηκε για τη μελέτη είναι προσαρμοσμένος στη συγκεκριμένη νοσοκομειακή βάση δεδομένων και δεν είναι διαθέσιμος στο κοινό, περιορίζοντας τη χρησιμότητά του χωρίς πρόσβαση στα υποκείμενα δεδομένα.

Το άρθρο με τίτλο «Early Detection of Heart Failure Using Electronic Health Records» των Kenney Ng et. al [20] διερευνά τις δυνατότητες των ηλεκτρονικών αρχείων υγείας (ΗΦΥ) για την ενίσχυση της έγκαιρης ανίχνευσης της καρδιακής ανεπάρκειας (HF) μέσω της προγνωστικής ανάλυσης. Οι συγγραφείς τονίζουν τον αυξανόμενο όγκο και την ποικιλομορφία των διαχρονικών δεδομένων ασθενών που είναι διαθέσιμα στους ΗΦΥ, τα οποία μπορούν να υποστηρίξουν σημαντικά τις αναλύσεις σε επίπεδο πληθυσμού και την εξατομικευμένη φροντίδα των ασθενών.

Η μελέτη υπογραμμίζει τη σημασία των προγνωστικών μοντέλων για τον εντοπισμό ασθενών που διατρέχουν κίνδυνο ΚΑ, με στόχο τη μείωση της αποφευκτέας περίθαλψης, όπως οι εισαγωγές στο νοσοκομείο, και τη βελτίωση της κλινικής λήψης αποφάσεων. Οι συγγραφείς σημειώνουν ότι, ενώ η ποσοτική απόδοση των προγνωστικών μοντέλων, που συχνά μετράται με την περιοχή κάτω από τη χαρακτηριστική καμπύλη λειτουργίας δέκτη (AUC), αποτελεί κρίσιμη εστίαση, δεν είναι ο μοναδικός καθοριστικός παράγοντας της χρησιμότητας ενός μοντέλου στην κλινική πρακτική. Άλλοι παράγοντες, όπως η συνοχή του μοντέλου, η ερμηνευσιμότητα και η ευκολία ενσωμάτωσης στις κλινικές ροές εργασίας, είναι εξίσου σημαντικοί για την επιτυχή εφαρμογή.

Η έρευνα διερευνά τον τρόπο με τον οποίο η ποσότητα και η ποικιλομορφία των δεδομένων του ΗΦΥ επηρεάζουν την απόδοση του μοντέλου. Οι συγγραφείς διαπίστωσαν ότι η απόδοση του μοντέλου επηρεάζεται σημαντικά από την ποικιλομορφία των δεδομένων, την κατασκευή των χαρακτηριστικών και το μήκος του παραθύρου παρατήρησης. Τα δεδομένα ΗΦΥ είναι εγγενώς ποικίλα, καθώς περιλαμβάνουν διάφορους κωδικούς ασθενειών, εντολές φαρμακευτικής αγωγής και εργαστηριακές μετρήσεις. Η μελέτη δείχνει ότι, ενώ η αύξηση του παραθύρου παρατήρησης μπορεί να βελτιώσει την απόδοση του μοντέλου, μειώνει επίσης τον αριθμό των ασθενών που είναι επιλέξιμοι για την εφαρμογή του μοντέλου, δημιουργώντας ένα συμβιβασμό που πρέπει να διαχειριστεί.

Ένα από τα βασικά ευρήματα είναι η ευαισθησία της απόδοσης του μοντέλου στο μέγεθος του συνόλου εκπαίδευσης. Οι συγγραφείς παρατήρησαν σημαντική βελτίωση της ακρίβειας πρόβλεψης όταν το μέγεθος του συνόλου εκπαίδευσης αυξήθηκε από 1.000 σε 4.000 εγγραφές ασθενών. Το εύρημα αυτό υπογραμμίζει την πρόκληση της εφαρμογής μοντέλων πρόβλεψης σε μικρότερες κλινικές πρακτικές, όπου η διαθεσιμότητα δεδομένων μπορεί να είναι περιορισμένη. Η μελέτη υποδηλώνει ότι η συγκέντρωση δεδομένων από πολλαπλές πρακτικές θα μπορούσε να βελτιώσει την απόδοση του μοντέλου και να διευκολύνει την ευρύτερη εφαρμογή του.

Το άρθρο συζητά επίσης τις μεθοδολογικές προκλήσεις που θέτει η ετερογενής φύση των δεδομένων ΗΦΥ. Σε αντίθεση με τα δομημένα επιδημιολογικά δεδομένα που συλλέγονται μέσω τυποποιημένων πρωτοκόλλων, τα δεδομένα ΗΦΥ συγκεντρώνονται με μη προγραμματισμένο τρόπο, γεγονός που οδηγεί σε μεταβλητότητα στη συχνότητα και τον τύπο των δεδομένων που συλλέγονται. Αυτή η ασυνέπεια μπορεί να περιπλέξει τις προσπάθειες προγνωστικής μοντελοποίησης. Οι συγγραφείς υπογραμμίζουν την ανάγκη για προσεκτική επιμέλεια των δεδομένων και την ανάπτυξη μοντέλων που μπορούν να προσαρμοστούν στα συγκεκριμένα χαρακτηριστικά των δεδομένων των διάφορων κλινικών ρυθμίσεων.

Όσον αφορά την απόδοση του μοντέλου, η μελέτη αξιολογεί τον αντίκτυπο των διαφορετικών μηκών παραθύρων πρόβλεψης και παρατήρησης. Διαπιστώνει ότι τα μικρότερα παράθυρα

πρόβλεψης (έως ένα έτος) αποδίδουν καλύτερες επιδόσεις, με τιμές AUC άνω του 0,80, ενώ οι επιδόσεις μειώνονται για μεγαλύτερα παράθυρα πρόβλεψης. Οι συγγραφείς προτείνουν ότι τα σήματα για την έγκαιρη ανίχνευση της HF μπορεί να είναι πιο εμφανή εντός ενός έως τριών ετών πριν από τη διάγνωση.

Η έρευνα υποστηρίζεται από επιχορήγηση των Εθνικών Ινστιτούτων Υγείας και συμβάλλει στον αυξανόμενο όγκο βιβλιογραφίας σχετικά με τη χρήση των ΗΦΥ για προγνωστική ανάλυση στην υγειονομική περίθαλψη. Οι συγγραφείς καταλήγουν στο συμπέρασμα ότι ενώ οι ΗΦΥ παρουσιάζουν σημαντικές ευκαιρίες για τη βελτίωση της έγκαιρης ανίχνευσης της HF, η επιτυχής εφαρμογή των προγνωστικών μοντέλων θα απαιτήσει την αντιμετώπιση των προκλήσεων που σχετίζονται με την ποικιλομορφία των δεδομένων, το μέγεθος του συνόλου εκπαίδευσης και την προσαρμοστικότητα του μοντέλου σε διαφορετικά κλινικά περιβάλλοντα.

2.2 Περιορισμοί και ελλείψεις στην βιβλιογραφία

Η βιβλιογραφία σχετικά με τη χρήση της εξηγήσιμης μηχανικής μάθησης (xAI) στους ΗΦΥ δείχνει διάφορες ελλείψεις που μπορούν να εμποδίσουν την αποτελεσματική ενσωμάτωση αυτών των τεχνολογιών στην κλινική πρακτική.

- Περιορισμένα πλαίσια αξιολόγησης

Ένα σημαντικό κενό είναι η έλλειψη τυποποιημένων πλαισίων αξιολόγησης για το xAI σε πλαίσια υγειονομικής περίθαλψης. Ενώ οι μελέτες αναδεικνύουν διάφορες τεχνικές xAI, όπως η απόσταση γνώσης και οι μηχανισμοί προσοχής, δεν υπάρχει επαρκής εστίαση στον τρόπο με τον οποίο αξιολογούνται αυτές οι τεχνικές όσον αφορά την κλινική τους εφαρμοσιμότητα και αποτελεσματικότητα. Μια συστηματική επισκόπηση προσδιορίζει ότι πολλές μελέτες δεν αντιμετωπίζουν επαρκώς την αναπαραγωγιμότητα των ευρημάτων τους, η οποία είναι κρίσιμη για την επικύρωση των μοντέλων xAI σε πραγματικές συνθήκες [21]. Η απουσία ισχυρών μετρικών αξιολόγησης περιορίζει τη δυνατότητα σύγκρισης διαφορετικών μεθόδων xAI και την κατανόηση των πρακτικών επιπτώσεών τους στη λήψη κλινικών αποφάσεων.

- Προκλήσεις ποιότητας και ολοκλήρωσης δεδομένων

Ένα άλλο κρίσιμο ζήτημα είναι η ποιότητα των δεδομένων εντός των ΗΦΥ. Η βιβλιογραφία τονίζει ότι οι ΗΦΥ περιέχουν τόσο δομημένα όσο και αδόμητα δεδομένα, με τα τελευταία να θέτουν σημαντικές προκλήσεις για τις εφαρμογές μηχανικής μάθησης. Οι τεχνικές επεξεργασίας φυσικής γλώσσας (NLP) χρησιμοποιούνται συχνά για τη δόμηση δεδομένων ελεύθερου κειμένου, αλλά οι μέθοδοι αυτές μπορούν να εισάγουν προκαταλήψεις και ανακρίβειες εάν δεν τύχουν κατάλληλης διαχείρισης [22]. Επιπλέον, η διακύμανση της ποιότητας των δεδομένων σε διάφορα συστήματα ΗΦΥ περιπλέκει την ενσωμάτωση των μοντέλων xAI, καθώς οι ασυνέπειες μπορεί να οδηγήσουν σε αναξιόπιστες προβλέψεις και εξηγήσεις.

- Ηθικές ανησυχίες

Η βιβλιογραφία επισημαίνει επίσης ηθικές και κανονιστικές προκλήσεις που σχετίζονται με τη χρήση του xAI στην υγειονομική περίθαλψη. Για παράδειγμα, ο Γενικός Κανονισμός για την Προστασία Δεδομένων (GDPR) δίνει έμφαση στο «δικαίωμα στην παροχή εξηγήσεων», το οποίο επιβάλλει τη διαφάνεια στις αυτοματοποιημένες διαδικασίες λήψης αποφάσεων. Ωστόσο, πολλά μοντέλα xAI εξακολουθούν να λειτουργούν ως «μαύρα κουτιά», αποτυγχάνοντας να ανταποκριθούν σε αυτές τις κανονιστικές απαιτήσεις. Αυτή η έλλειψη συμμόρφωσης μπορεί να

υπονομεύσει την εμπιστοσύνη μεταξύ των παρόχων υγειονομικής περίθαλψης και των ασθενών, περιορίζοντας ενδεχομένως την υιοθέτηση των τεχνολογιών ΧΑΙ στην κλινική πρακτική [23].

- Συμβιβασμός πολυπλοκότητας και ερμηνευσιμότητας

Ένας άλλος κρίσιμος περιορισμός είναι ο εγγενής συμβιβασμός μεταξύ της πολυπλοκότητας του μοντέλου και της ερμηνευσιμότητας. Πολλά προηγμένα μοντέλα μηχανικής μάθησης, όπως οι αλγόριθμοι βαθιάς μάθησης, επιτυγχάνουν υψηλή ακρίβεια αλλά συχνά περιγράφονται ως «μαύρα κουτιά» λόγω της πολυπλοκότητάς τους. Η βιβλιογραφία τονίζει ότι η αύξηση της διαφάνειας ενός μοντέλου μπορεί να οδηγήσει σε μείωση της προβλεπτικής του απόδοσης, δημιουργώντας ένα δίλημμα για τους επαγγελματίες που απαιτούν τόσο ακριβείς προβλέψεις όσο και κατανοητές εξηγήσεις [25].

- Απουσία plotting σε ΗΦΥ και σύγκρισης μεταξύ των αλγορίθμων

Στην πτυχιακή εργασία επιχειρείται η οπτικοποίηση ενός μεγάλου συνόλου δεδομένων των ΗΦΥ. Κάτι τέτοιο στην παρούσα βιβλιογραφία δεν έχει παρουσιαστεί ιδιαίτερα, όπως επίσης και η σύγκριση της αποτελεσματικότητας μεταξύ των δύο αλγορίθμων στο ίδιο σύνολο δεδομένων.

Ενώ η ενσωμάτωση της xAI στους ΗΦΥ υπόσχεται πολλά, υπάρχουν σημαντικοί περιορισμοί. Σε αυτούς περιλαμβάνονται η ανάγκη για τυποποιημένα πλαίσια αξιολόγησης, οι προκλήσεις που σχετίζονται με την ποιότητα και την ενσωμάτωση δεδομένων, η αντιστάθμιση πολυπλοκότητας-ερμηνευσιμότητας, οι ελλείψεις στον σχεδιασμό με επίκεντρο τον χρήστη και οι δεοντολογικές ανησυχίες. Η αντιμετώπιση αυτών των ελλείψεων είναι απαραίτητη για την επιτυχή εφαρμογή του ΧΑΙ στην υγειονομική περίθαλψη, ενισχύοντας τελικά τη φροντίδα των ασθενών και τη λήψη κλινικών αποφάσεων.

3. Μεθοδολογία

Η Python είναι μια δημοφιλής γλώσσα προγραμματισμού που έχει αποκτήσει ευρεία αποδοχή στην επιστημονική κοινότητα λόγω της απλότητας, της αναγνωσιμότητάς της και του εκτεταμένου οικοσυστήματος βιβλιοθηκών. Είναι μια διεργασμένη, υψηλού επιπέδου, γενικού σκοπού γλώσσα προγραμματισμού γνωστή για την ευελιξία και την ευκολία χρήσης της. [26]

Στον τομέα της επιστήμης των δεδομένων, αρκετές βιβλιοθήκες Python έχουν αναδειχθεί ως βασικά εργαλεία για αριθμητικούς υπολογισμούς, επεξεργασία δεδομένων και οπτικοποίηση. Μεταξύ αυτών, οι NumPy, Pandas, Matplotlib και Seaborn είναι ιδιαίτερα αξιοσημείωτες για τις ισχυρές λειτουργίες τους και την ευρεία υιοθέτησή τους. Οι συγκεκριμένες βιβλιοθήκες χρησιμοποιούνται εκτενώς στο κομμάτι της επεξεργασίας των δεδομένων αλλά και στην οπτικοποίησή τους.

Η NumPy (Numerical Python) χρησιμεύει ως η θεμελιώδης βιβλιοθήκη για επιστημονικούς υπολογισμούς στην Python. Παρέχει υποστήριξη για μεγάλους, πολυδιάστατους πίνακες και πίνακες, μαζί με μια συλλογή μαθηματικών συναρτήσεων για την αποτελεσματική λειτουργία αυτών των πινάκων. Τα βασικά χαρακτηριστικά περιλαμβάνουν:

- Αντικείμενο πίνακα: Το βασικό χαρακτηριστικό είναι το αντικείμενο N-διάστατων πινάκων (ndarray), το οποίο επιτρέπει γρήγορες και αποδοτικές λειτουργίες πινάκων.

- Μαθηματικές συναρτήσεις: Προσφέρει ένα ευρύ φάσμα μαθηματικών συναρτήσεων, συμπεριλαμβανομένων τριγωνομετρικών, στατιστικών και αλγεβρικών ρουτινών.

- Ενσωμάτωση: Η NumPy ενσωματώνεται απρόσκοπτα με άλλες βιβλιοθήκες όπως η Pandas και η Matplotlib, ενισχύοντας τη χρησιμότητά της σε εφαρμογές επιστήμης δεδομένων [27].

Η Pandas είναι μια βιβλιοθήκη επεξεργασίας και ανάλυσης δεδομένων ανοικτού κώδικα που έχει δημιουργηθεί πάνω στη NumPy. Παρέχει δομές δεδομένων και συναρτήσεις που απαιτούνται για τον απρόσκοπτο χειρισμό δομημένων δεδομένων. Τα βασικά χαρακτηριστικά περιλαμβάνουν:

- DataFrames: Η κύρια δομή δεδομένων, DataFrame, είναι μια δισδιάστατη δομή δεδομένων με ετικέτες που επιτρέπει ποικίλους τύπους δεδομένων.

- Χειρισμός δεδομένων: Τα εργαλεία για τη συγχώνευση, την αναδιαμόρφωση, την επιλογή και τον καθαρισμό δεδομένων αποτελούν αναπόσπαστο μέρος της λειτουργικότητάς του.

- Χειρισμός χρονοσειρών: Το Pandas προσφέρει ισχυρή υποστήριξη για δεδομένα χρονοσειρών, καθιστώντας το ανεκτίμητο για την ανάλυση χρονικών δεδομένων.

- Εισαγωγή/εξαγωγή δεδομένων: Υποστηρίζει την ανάγνωση και εγγραφή δεδομένων από διάφορες μορφές, όπως CSV, Excel και SQL, διευκολύνοντας την εύκολη ενσωμάτωση δεδομένων [27].

Η Matplotlib είναι μια ολοκληρωμένη βιβλιοθήκη για τη δημιουργία στατικών, κινούμενων και διαδραστικών απεικονίσεων στην Python. Είναι ιδιαίτερα παραμετροποιήσιμη και υποστηρίζει ένα ευρύ φάσμα τύπων γραφικών παραστάσεων. Τα βασικά χαρακτηριστικά της περιλαμβάνουν:

- Ποικιλία γραφικών παραστάσεων: Υποστηρίζει διάφορους τύπους γραφικών παραστάσεων, όπως διαγράμματα γραμμής, ράβδου, διασποράς, ιστογράμματος και πίτας.

- Προσαρμογή: Οι χρήστες μπορούν να ελέγχουν κάθε πτυχή του διαγράμματος, από τα χρώματα έως τις ετικέτες, επιτρέποντας τη λεπτομερή οπτική αναπαράσταση.

- Ενσωμάτωση: Η δυνατότητα ενσωμάτωσης των γραφικών παραστάσεων σε ένα πρόγραμμα είναι πολύ σημαντική: Το Matplotlib συνεργάζεται καλά με το NumPy και το Pandas, καθιστώντας το βασικό εργαλείο για την οπτικοποίηση δεδομένων στην επιστημονική πληροφορική.

- Διαδραστικά γραφήματα: Υποστηρίζει διαδραστικά γραφήματα και κινούμενα σχέδια, ενισχύοντας την εμπλοκή του χρήστη με τα οπτικά δεδομένα [28].

Τέλος, η Seaborn είναι μια βιβλιοθήκη οπτικοποίησης που βασίζεται στην Matplotlib, σχεδιασμένη για τη δημιουργία ελκυστικών και κατατοπιστικών στατιστικών γραφικών. Απλοποιεί τη διαδικασία δημιουργίας σύνθετων οπτικοποιήσεων. Τα βασικά χαρακτηριστικά της περιλαμβάνουν:

- Στατιστικά γραφήματα: Το Seaborn υποστηρίζει μια σειρά από στατιστικά διαγράμματα, συμπεριλαμβανομένων των διαγραμμάτων box, των διαγραμμάτων violin και των διαγραμμάτων ζεύγους, τα οποία είναι απαραίτητα για τη διερευνητική ανάλυση δεδομένων.

- Βελτιωμένη αισθητική: Παρέχει μια διεπαφή υψηλού επιπέδου για τη σχεδίαση ελκυστικών γραφικών, αντιστοιχίζοντας αυτόματα τιμές δεδομένων σε οπτικά χαρακτηριστικά όπως το χρώμα και το μέγεθος.
- Ενσωμάτωση με το Pandas: Το Seaborn είναι ιδιαίτερα κατάλληλο για τη συνεργασία με Pandas DataFrames, βελτιώνοντας τη διαδικασία οπτικοποίησης.
- Στατιστική εκτίμηση: Περιλαμβάνει ενσωματωμένες συναρτήσεις για στατιστική εκτίμηση και εξαγωγή συμπερασμάτων, όπως μοντέλα γραμμικής παλινδρόμησης [28].

Το Scikit-learn ενσωματώνει ένα ευρύ φάσμα αλγορίθμων μηχανικής μάθησης τελευταίας τεχνολογίας για προβλήματα μεσαίας κλίμακας με επίβλεψη και χωρίς επίβλεψη. Επικεντρώνεται στο να φέρει τη μηχανική μάθηση σε μη ειδικούς χρησιμοποιώντας μια γλώσσα υψηλού επιπέδου γενικού σκοπού. Δίνεται έμφαση στην ευκολία χρήσης, την απόδοση, την τεκμηρίωση και τη συνοχή του API [29].

3.1 Δεδομένα

Χρησιμοποιήσαμε το Synthea Dataset Jsons – EHR το οποίο είναι διαθέσιμο στο kaggle [30]. Το dataset περιέχει συνολικά 129218 αρχεία json. Τα αρχεία είναι χωρισμένα σε 255 φακέλους και 4080 υποφακέλους. Προκειμένου να γίνει ολοκληρωμένη εξερεύνηση του συνόλου δεδομένων χρησιμοποιήθηκε η βιβλιοθήκη pandas το οποίο σύμφωνα με τους McKinney, Wes et al. έχει τον ευρύτερο στόχο να γίνει το πιο ισχυρό και ευέλικτο εργαλείο ανάλυσης/χειρισμού δεδομένων ανοικτού κώδικα που είναι διαθέσιμο σε οποιαδήποτε γλώσσα [31].

Η βασική δομή δεδομένων του Pandas είναι το DataFrame. Πρόκειται για μια δομή πίνακα που αποτελείται από στήλες και γραμμές. Οι στήλες έχουν έναν τύπο δεδομένων και ένα προαιρετικό όνομα που μπορεί να χρησιμοποιηθεί για την αναφορά στις στήλες. Επιπλέον, ένα DataFrame μπορεί να έχει μια στήλη δείκτη, η οποία μπορεί να χρησιμοποιηθεί για την άμεση πρόσβαση σε σειρές με βάση την τιμή [32].

Δημιουργήσαμε δύο συναρτήσεις: τη `process_one_file` η οποία επεξεργάζεται ένα προσωρινό DataFrame με μικτά ιατρικά δεδομένα και διαχωρίζει τα δεδομένα με βάση τον τύπο του πόρου και τα προσθέτει στα αντίστοιχα DataFrames και τη `clean_and_rename` που χρησιμοποιείται για τον καθαρισμό και τη μετονομασία των στηλών των διαφόρων DataFrames, ώστε να είναι ευκολότερη η επεξεργασία τους στη συνέχεια.

Στα οκτώ DataFrames που προέκυψαν έγινε η απαραίτητη επεξεργασία που αφορούσε αλλαγές στα ονόματα των στηλών αλλά και διαχωρισμός ή διαγραφή ορισμένων στηλών οι οποίες δεν χρειάζονται για το κομμάτι της μηχανικής μάθησης (π.χ. το όνομα και η εθνικότητα ενός ασθενή). Επίσης, το κάθε DataFrame αποθηκεύτηκε και σε ένα csv αρχείο.

Το δείγμα αφορά το 2.14% των συνολικών δεδομένων και περιλαμβάνει:

2773 ασθενείς, 103680 ιατρικές παρατηρήσεις, 10184 αναφορές κλινικής κατάστασης, 25607 αναφορές ιατρικών συναντήσεων, 5158 σχέδια περίθαλψης, 11843 αναφορές ιατρικών διαδικασιών, 6637 διαγνωστικές αναφορές και 18538 αναφορές εμβολιασμών.

Κρατήσαμε όμως μόνο ασθενείς με τιμές σε όλες τις στήλες, καθώς και χαρακτηριστικά με ισορροπημένα αποτελέσματα, οπότε το τελικό dataframe περιείχε 662 ασθενείς.

Ας δούμε τις στήλες που θα χρειαστούμε για κάθε DataFrame (σε παρένθεση ο τύπος των δεδομένων) :

patient <i>Στοιχεία ασθενή</i>	resource.id: id ασθενή (αλφαριθμητικό) resource.gender: φύλο (0: άνδρες, 1: γυναίκες) age: ηλικία (ακέραιος)
observation <i>Ιατρικές παρατηρήσεις</i>	resource.subject.reference: id ασθενή (αλφαριθμητικό) Body Height: ύψος (σε εκατοστά, δεκαδικός) Body Mass Index: Δείκτης Μάζας Σώματος (σε kg/m ² , δεκαδικός) Body Weight: βάρος (σε kg, δεκαδικός) Calcium: ασβέστιο (σε mg/dL, δεκαδικός) Carbon Dioxide: Διοξείδιο του άνθρακα (σε mmol/L, ακέραιος) Chloride: Χλωρίδιο (σε mmol/L, ακέραιος) Glucose: Γλυκόζη (σε mg/dL, ακέραιος) Hemoglobin A1c/Hemoglobin.total in Blood: ολική αιμοσφαιρίνη στο αίμα (σε %, δεκαδικός) High Density Lipoprotein Cholesterol: «Καλή» χοληστερόλη (σε mg/dL, ακέραιος) Low Density Lipoprotein Cholesterol: «Κακή» χοληστερόλη (σε mg/dL, ακέραιος) Potassium: Κάλιο (σε mg/dL, δεκαδικός) Sodium: Νάτριο (σε mmol/L, ακέραιος) Total Cholesterol: Ολική χοληστερόλη (σε mg/dL, ακέραιος) Triglycerides: Τριγλυκερίδια (σε mg/dL, ακέραιος) Urea Nitrogen: Αζωτο ουρίας (σε mg/dL, ακέραιος)
condition <i>Κλινική κατάσταση</i>	resource.subject.reference: id ασθενή (αλφαριθμητικό) Acute viral pharyngitis: Οξεία ιογενής φαρυγγίτιδα (0 αν δεν έχει, 1 αν έχει, δυαδικός) Hypertension: Υπέρταση (0 αν δεν έχει, 1 αν έχει, δυαδικός) Viral sinusitis: Ιογενής ιγμορίτιδα (0 αν δεν έχει, 1 αν έχει, δυαδικός)
encounter <i>Ιατρική συνάντηση</i>	resource.patient.reference: id ασθενή (αλφαριθμητικό) emergency: έκτακτη ιατρική συνάντηση (0 αν δεν συνέβη, 1 αν συνέβη, δυαδικός)
careplan <i>Σχέδιο περίθαλψης</i>	resource.subject.reference: id ασθενή (αλφαριθμητικό) Respiratory therapy: αναπνευστική θεραπεία (0 αν δεν συνέβη, 1 αν συνέβη, δυαδικός)
procedure <i>Ιατρικές διαδικασίες</i>	resource.subject.reference: id ασθενή (αλφαριθμητικό) Colonoscopy: Κολονοσκόπηση (0 αν δεν συνέβη, 1 αν συνέβη, δυαδικός)
immunization <i>Εμβολιασμοί</i>	resource.patient.reference: id ασθενή (αλφαριθμητικό) Td (adult) preservative free: Εμβόλιο Td για τέτανο και διφθερίτιδα (0 αν δεν εμβολιάστηκε, 1 αν εμβολιάστηκε, δυαδικός)

3.2 Ένωση των dataframes

Επειδή το κάθε dataframe δεν περιέχει πολλές στήλες, η ένωση των dataframes έπρεπε να γίνει οπωσδήποτε, προκειμένου να λειτουργήσει το machine learning.

Στα περισσότερα dataframe για παράδειγμα στο observation, υπήρχε η στήλη resource.code.display που περιείχε τα ονόματα των χαρακτηριστικών και η στήλη

`resource.valueQuantity.value` που περιείχε την τιμή του χαρακτηριστικού. Για την σωστή επεξεργασία των δεδομένων, κρατήσαμε τα πολυπληθέστερα χαρακτηριστικά και δημιουργήσαμε μία στήλη για το κάθε ένα με χρήση της εντολής `pandas.pivot_table`. Στη συνέχεια ενώσαμε τα `dataframes` με το `pandas.merge`, χρησιμοποιώντας ως κλειδί το `id` των ασθενών, που υπήρχε σε κάθε `dataset`. Το `dataframe diagnostic_reports` αποφασίστηκε να μην χρησιμοποιηθεί, καθώς είχε πολλά `missing values` και μη ισορροπημένα χαρακτηριστικά.

3.3 Απεικόνιση των δεδομένων

Με τα `DataFrames` στην τελική τους μορφή, προχωρήσαμε σε απεικόνιση(plotting) των δεδομένων με χρήση των δύο βιβλιοθηκών `matplotlib` και `seaborn`.

Η `Matplotlib` είναι μια δημοφιλής βιβλιοθήκη γραφικών για πακέτο οπτικοποίησης δεδομένων στην `Python`, η οποία υποστηρίζεται εύκολα με τις `NumPy`, `Pandas` και άλλες σχετικές βιβλιοθήκες [33].

Η `seaborn` είναι μια βιβλιοθήκη για την κατασκευή στατιστικών γραφικών στην `Python`. Παρέχει μια υψηλού επιπέδου διασύνδεση με την `matplotlib` και ενσωματώνεται στενά με τις δομές δεδομένων του `pandas`. Οι συναρτήσεις της βιβλιοθήκης `seaborn` εκθέτουν ένα δηλωτικό, προσανατολισμένο στα σύνολα δεδομένων `API`, που καθιστά εύκολη τη μετάφραση ερωτήσεων σχετικά με τα δεδομένα σε γραφικά που μπορούν να τις απαντήσουν [34].

3.4 Extreme Gradient Boosting (XGBoost)

Το `Extreme Gradient Boosting`, σε συντομογραφία `XGBoost`, είναι ένα κλιμακούμενο σύστημα μηχανικής μάθησης για την ενίσχυση δέντρων (`tree boosting`). Το σύστημα είναι διαθέσιμο ως πακέτο ανοικτού κώδικα. Ο αντίκτυπος του συστήματος έχει αναγνωριστεί ευρέως σε διάφορες προκλήσεις μηχανικής μάθησης και εξόρυξης δεδομένων [35].

3.5 CatBoost

Ο `CatBoost` (`categorical boost` - κατηγορική ενίσχυση) βασίζεται σε δέντρα αποφάσεων και στο `gradient boosting`, αλλά με μειωμένες παραμέτρους. Ως εκ τούτου, ο `CatBoost` απαιτεί λιγότερο χρόνο εκπαίδευσης και δοκιμής σε σύγκριση με τα άλλα μοντέλα συνόλου. Αυτό το μοντέλο δίνει συνήθως καλές επιδόσεις με τις προεπιλεγμένες παραμέτρους. Ο `CatBoost` μπορεί να εφαρμοστεί στα αριθμητικά δεδομένα μετά την εκτέλεση της κωδικοποίησης των δεδομένων [36].

3.6 SHapley Additive exPlanations (SHAP)

Ο `SHAP` είναι μια προσέγγιση που βασίζεται στη θεωρία παιγνίων για την περιγραφή της απόδοσης ενός μοντέλου μηχανικής μάθησης. Για την παραγωγή ενός ερμηνεύσιμου μοντέλου, ο `SHAP` χρησιμοποιεί μια προσθετική μέθοδο απόδοσης χαρακτηριστικών, δηλαδή ένα μοντέλο εξόδου ορίζεται ως γραμμική πρόσθεση μεταβλητών εισόδου [37].

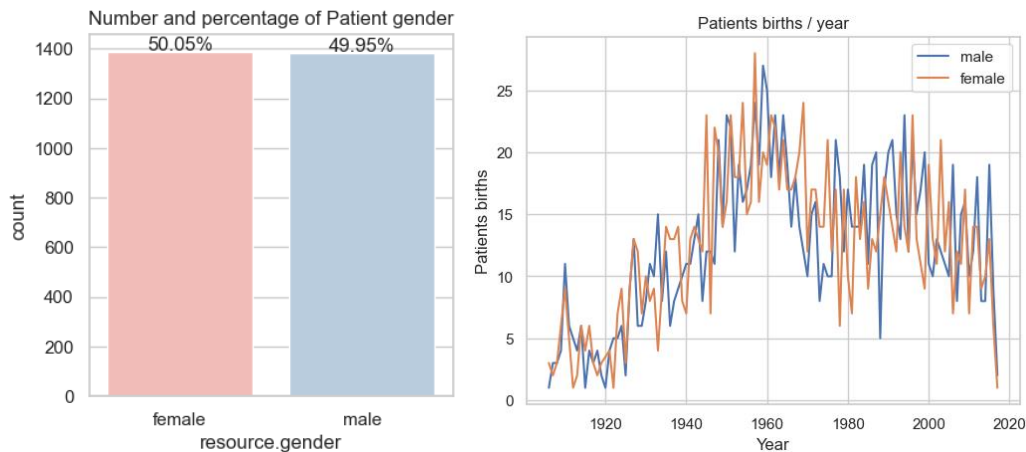
3.7 Local Interpretable Model-agnostic Explanations (LIME)

Ο `LIME` είναι ένας αλγόριθμος που παρέχει επεξηγήσεις με βάση το παράδειγμα στις προβλέψεις οποιουδήποτε ταξινομητή. Αυτές οι εξηγήσεις είναι τοπικά πιστές ανά περίπτωση, ανεξάρτητες από τον τύπο του μοντέλου του ταξινομητή, και μαθαίνονται πάνω σε ερμηνεύσιμες αναπαραστάσεις της περίπτωσης [38].

4. Αποτελέσματα

4.1 Plotting

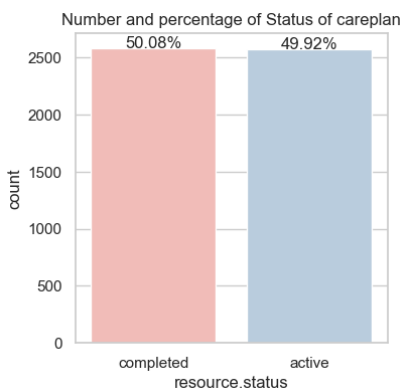
Σε σύνολο 2733 ασθενών παρατηρούμε ότι το ποσοστό ανδρών και γυναικών είναι σχεδόν ίδιο.



Γράφημα 1. Δημογραφικά στοιχεία (φύλο, έτος γέννησης ανά ασθενή, έτος γέννησης ανά φύλο)

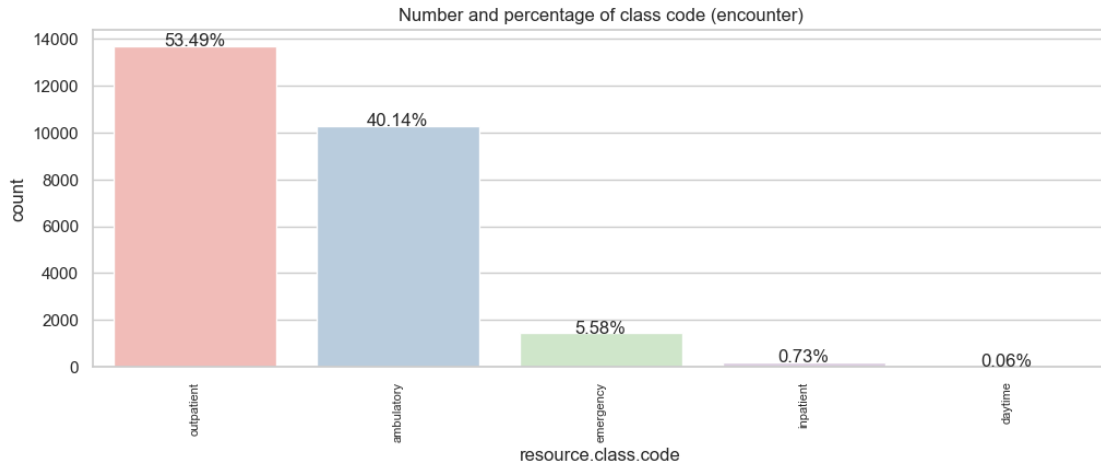
Όσον αφορά το έτος γέννησης, οι περισσότεροι ασθενείς είναι γεννηθείς μεταξύ των ετών 1950-1960, ενώ δεν διαφέρουν σημαντικά τα στοιχεία ανάμεσα στα δύο φύλα.

Παρόμοιο ποσοστό έχουμε μεταξύ των ασθενών με ολοκληρωμένο σχέδιο περίθαλψης και των ασθενών με ενεργό σχέδιο περίθαλψης, όπως φαίνεται στο παρακάτω διάγραμμα.



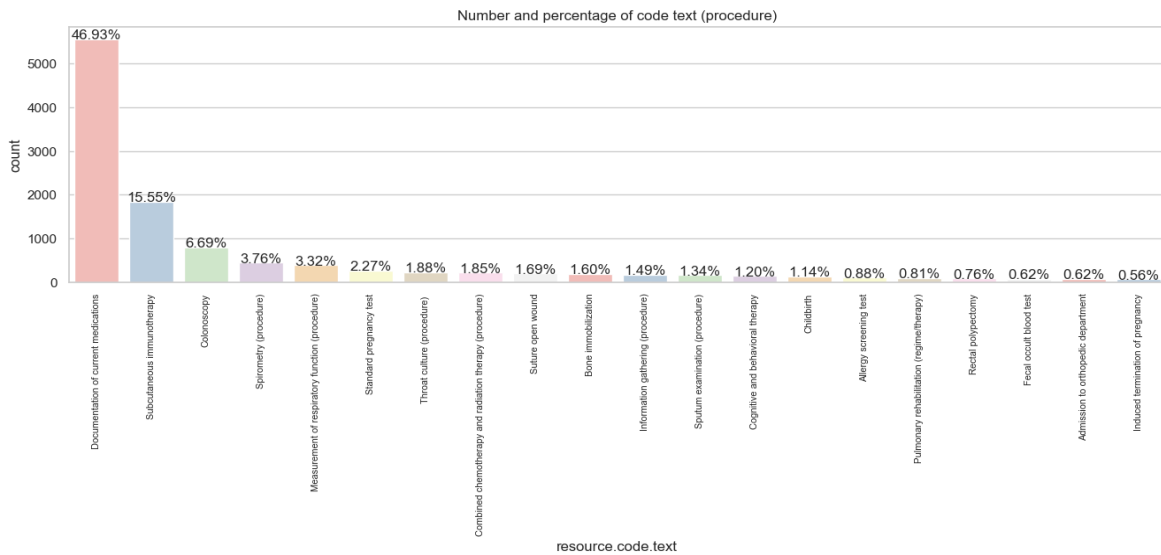
Γράφημα 2. Ποσοστό ασθενών με ολοκληρωμένο ή ενεργό σχέδιο περίθαλψης.

Όσον αφορά τις κλάσεις στις ιατρικές συναντήσεις, υψηλό ποσοστό έχουν οι συναντήσεις στα εξωτερικά ιατρεία αλλά και για περιπατητικούς ασθενείς (ατόμων δηλαδή που δεν χρειάζονται βοήθεια για να κινηθούν). Χαμηλά τα ποσοστά των επειγόντων περιστατικών, αλλά και των ενδονοσοκομειακών και ημερησίων περιστατικών.



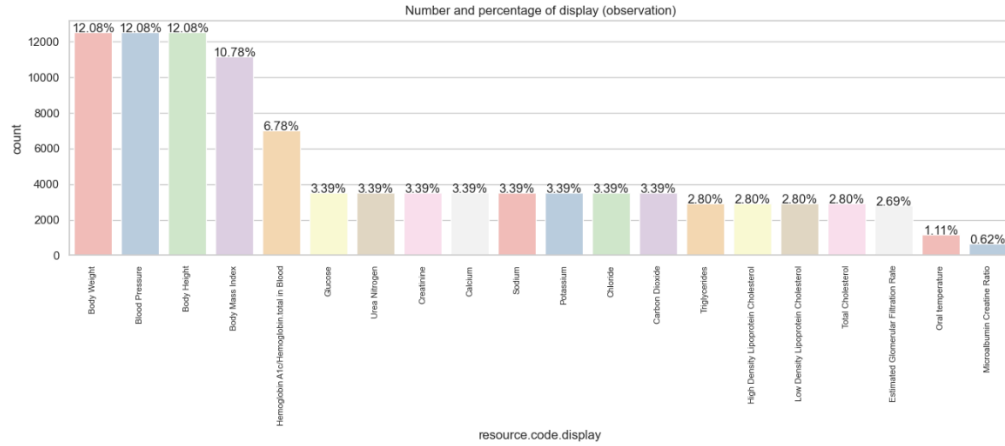
Γράφημα 3. Ποσοστό ανά είδος περιστατικού (σε εξωτερικά ιατρεία, χωρίς διανυκτέρευση, στα επείγοντα, ενδονοσοκομειακά, ημερήσια)

Στις ιατρικές διαδικασίες, η συνταγογράφηση συγκεντρώνει με διαφορά το υψηλότερο ποσοστό, με την υποδόρια ανοσοθεραπεία να ακολουθεί. Στο dataset καταγράφονται πολλές άλλες διαδικασίες αλλά με αρκετά μικρά ποσοστά.

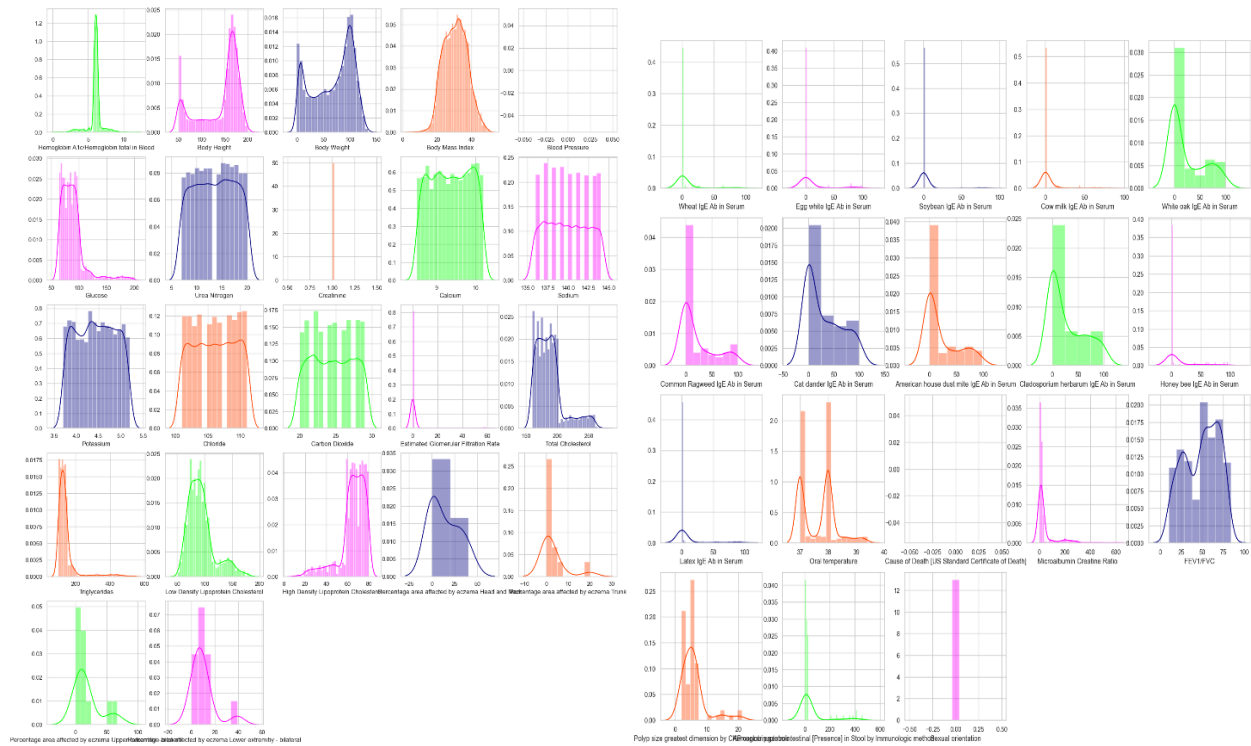


Γράφημα 4. Ιατρικές διαδικασίες (συνταγογράφηση, τοκετός, ανοσοθεραπεία κλπ.)

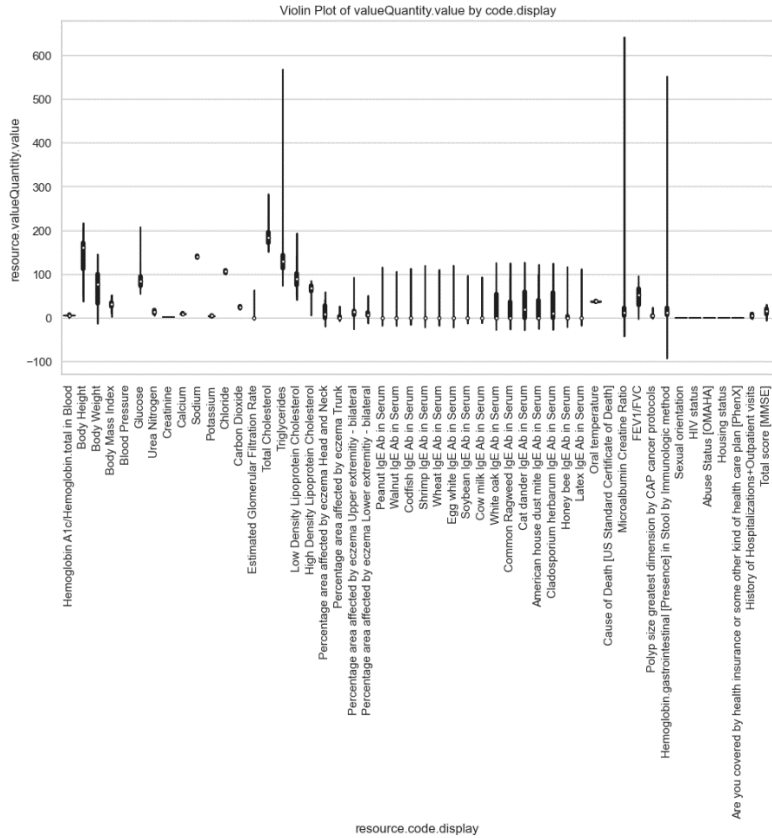
Το σημαντικότερο dataframe από αυτά που δημιουργήθηκαν είναι αυτό των ιατρικών παρατηρήσεων. Πάνω από 12000 καταγραφές ύψους, βάρους και αρτηριακής πίεσης υπάρχουν στο dataframe.



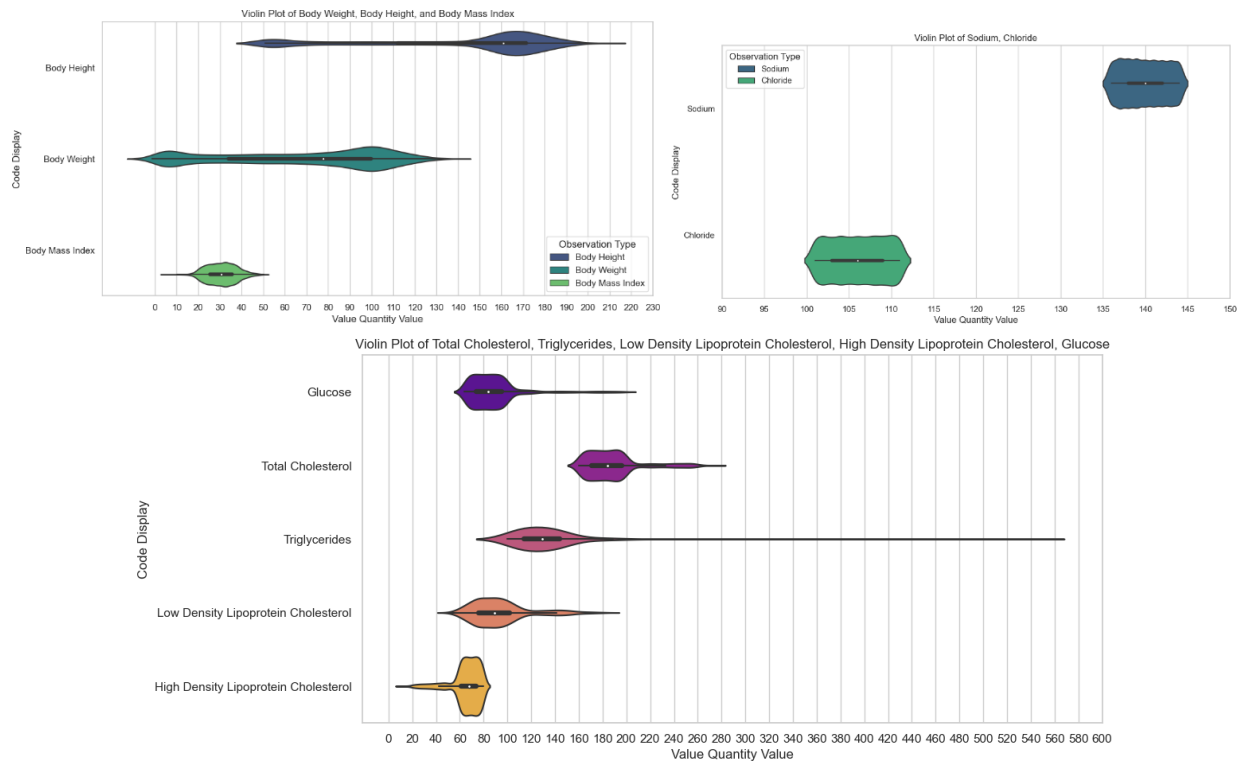
Γράφημα 5. Ιατρικές παρατηρήσεις (ύψος, βάρος, αρτηριακή πίεση, δείκτης μάζας σώματος κ.α.)

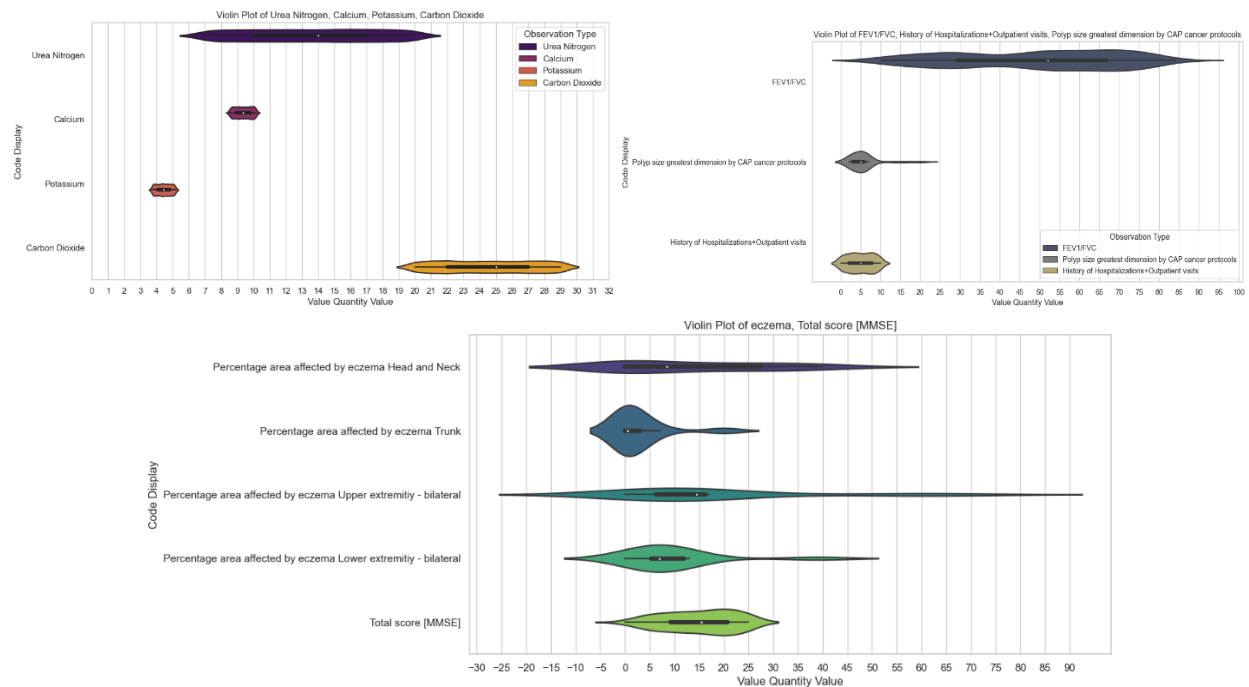


Γράφημα 6. Ιατρικές παρατηρήσεις στο σύνολο των ασθενών (Displot με KDE Curve)



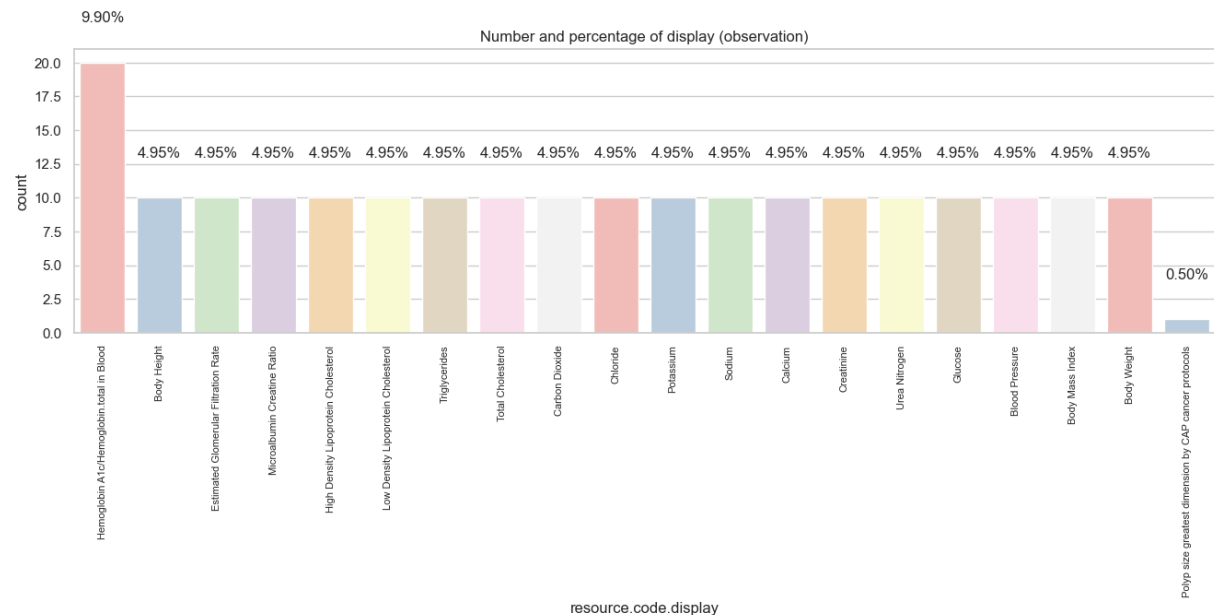
Γράφημα 7. Ιατρικές παρατηρήσεις στο σύνολο των ασθενών (Violinplot)





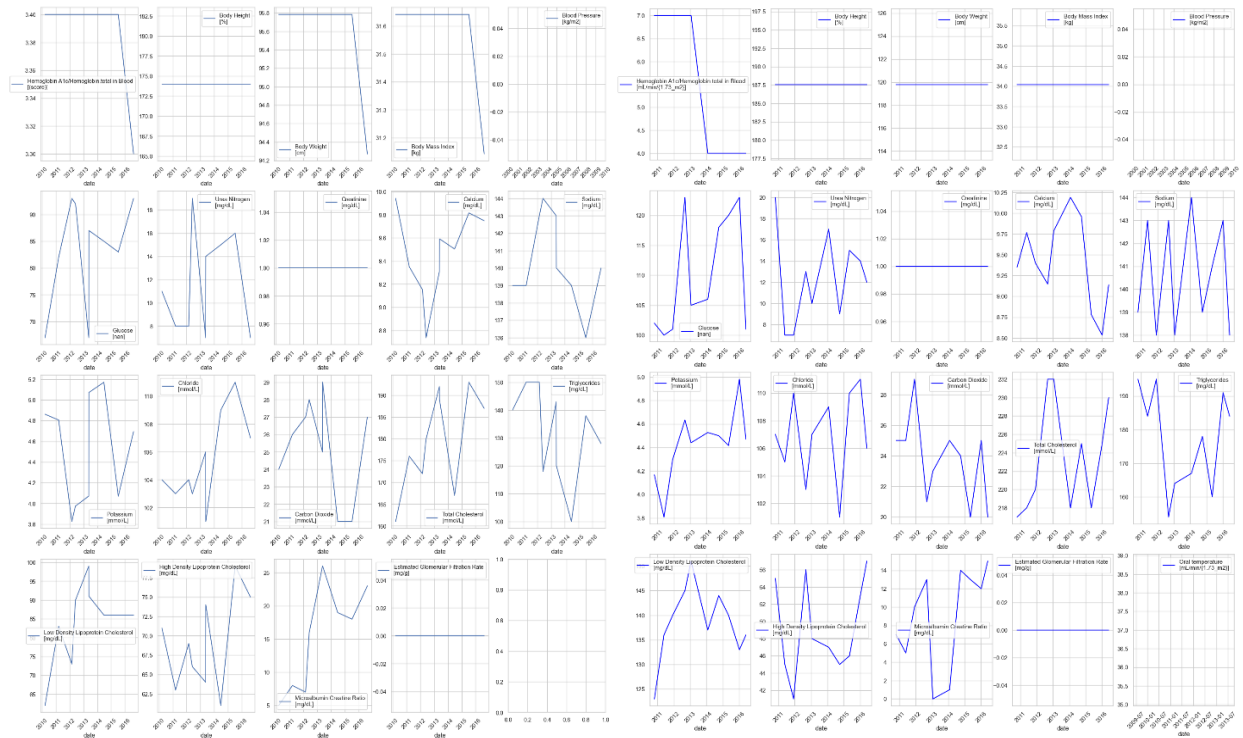
Γράφημα 8. Violin Plots ιατρικών παρατηρήσεων χωρισμένων κατά ομάδες.

Σε κάθε ασθενή, οι ιατρικές παρατηρήσεις ποικίλουν και ένας τύπος παρατήρησης μπορεί να έχει καταγραφεί παραπάνω από μία φορά. Στον παρακάτω ασθενή, έχει γίνει μέτρηση αιμοσφαιρίνης τουλάχιστον 20 φορές.

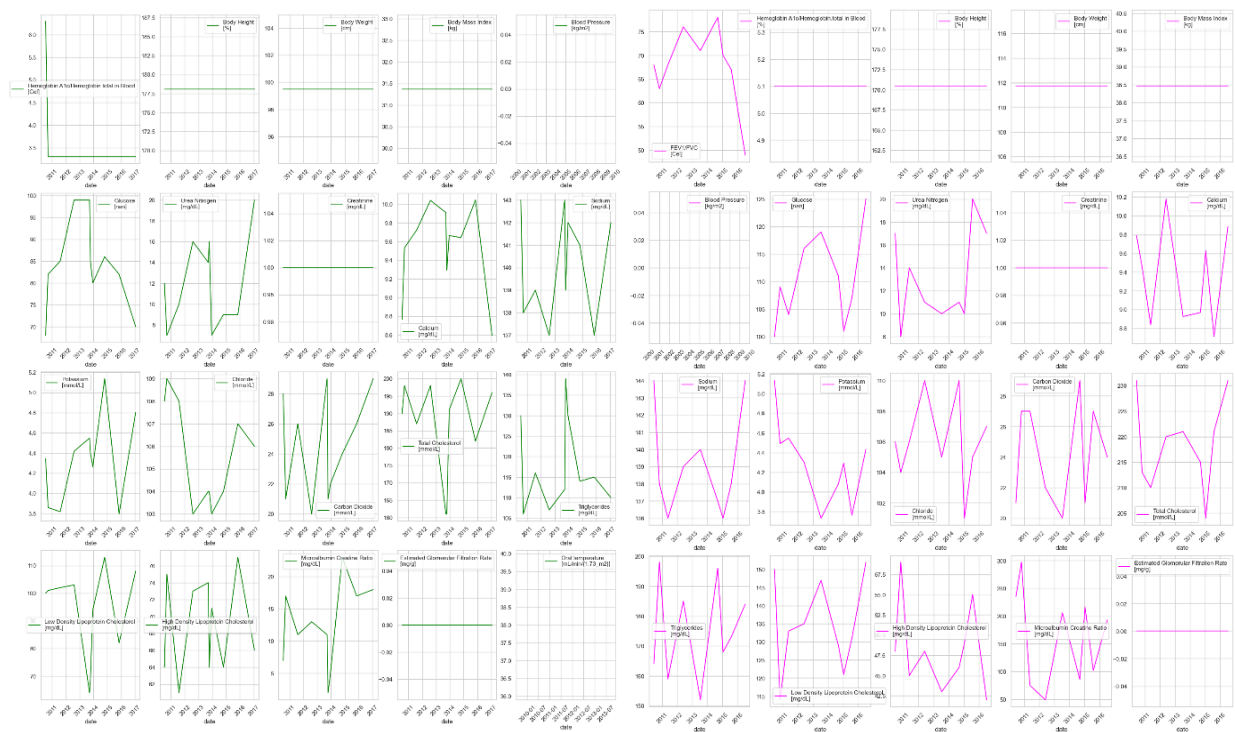


Γράφημα 9. Ποσοστό ιατρικών παρατηρήσεων ενός τυχαίου ασθενή.

Για πολλούς από τους ασθενείς, μπορούμε να φτιάξουμε διαγράμματα σαν αυτά που παρουσιάζονται παρακάτω, τα οποία δείχνουν την μεταβολή των ιατρικών παρατηρήσεων σε βάθος χρόνου.



Γράφημα 10. Μεταβολή τιμών των ιατρικών παρατηρήσεων δύο τυχαίων ασθενών ανά έτος.



Γράφημα 11. Μεταβολή τιμών των ιατρικών παρατηρήσεων δύο τυχαίων ασθενών ανά έτος.

4.2 Classification

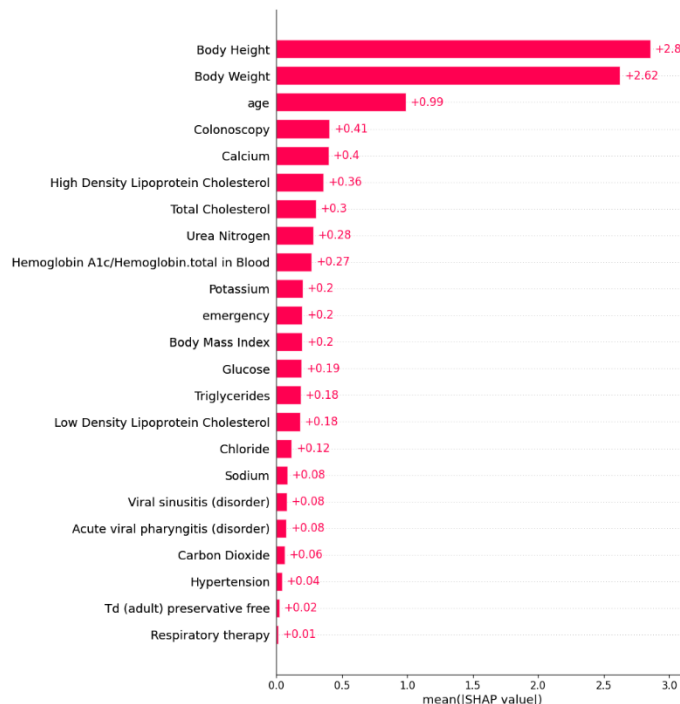
4.2.1 Γενικός πίνακας αποτελεσμάτων

Αλγόριθμος	Explainer	Target column	Accuracy	Precision	Recall	F1 score	roc_auc score
XGBoost	SHAP LIME	gender	0.83	0.83	0.83	0.83	0.83
CatBoost	SHAP LIME	gender	0.86	0.86	0.87	0.86	0.87
XGBoost	SHAP LIME	Colonoscopy	0.77	0.78	0.78	0.77	0.78
CatBoost	SHAP LIME	Colonoscopy	0.77	0.79	0.78	0.77	0.78
XGBoost	SHAP LIME	Td vaccine	0.86	0.86	0.83	0.84	0.83
CatBoost	SHAP LIME	Td vaccine	0.83	0.84	0.77	0.79	0.77

4.2.2 Classification ως προς το φύλο (gender)

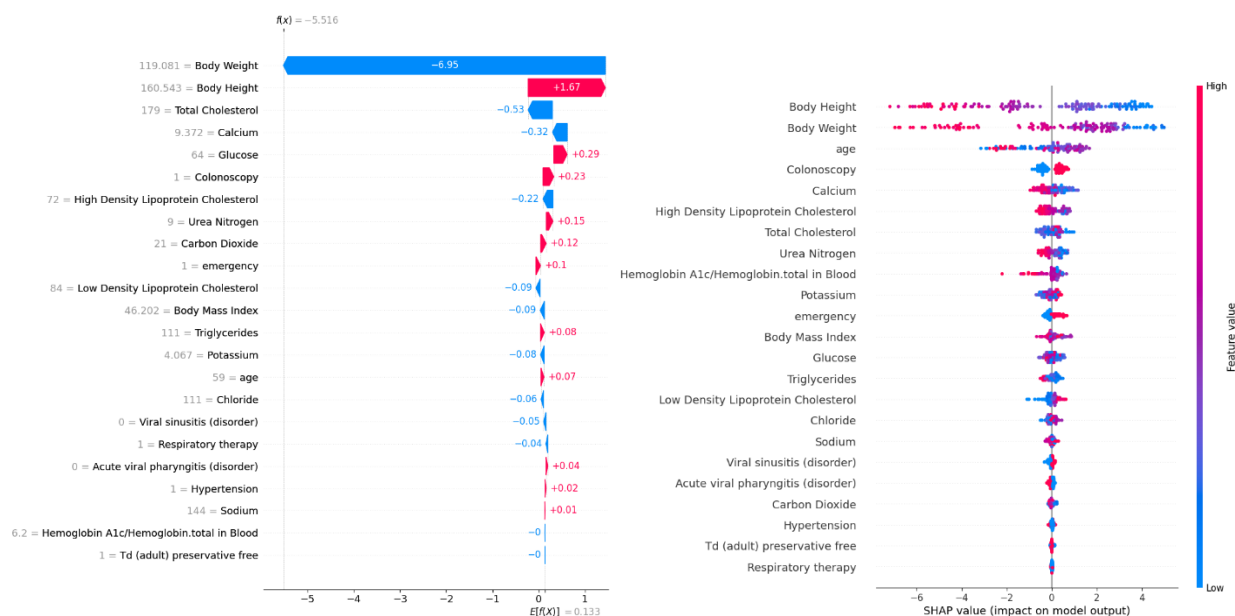
Στο 0.83 ήταν το ποσοστό επιτυχίας για τον XGBoost, ελαφρώς υψηλότερα στο 0.86 για τον CatBoost, ο οποίος για επιλεγμένο χαρακτηριστικό το φύλο, πέτυχε τα μεγαλύτερα σκορ σε σχέση με τα άλλα δύο χαρακτηριστικά. Η επιλεγμένη κλάση παίρνει δύο τιμές, 0 για το ανδρικό φύλο και 1 για το γυναικείο.

XGBoost

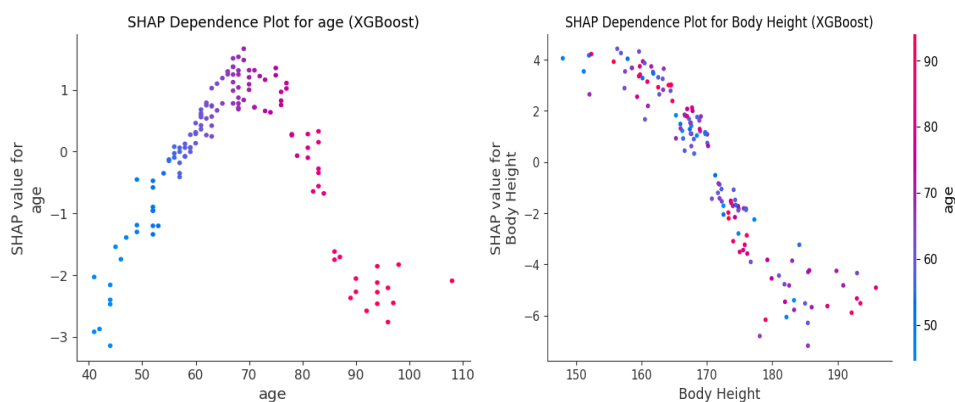


Γράφημα 12. BarPlot του XGBoost (feature selected: gender)

Κατά μέσο όρο το ύψος και το βάρος και το ύψος των ασθενών επιδρά έντονα στο μοντέλο. Για έναν συγκεκριμένο ασθενή, δηλαδή μία συγκεκριμένη πρόβλεψη, το βάρος έχει σημαντικά αρνητικό αντίκτυπο, ενώ θετικό αντίκτυπο έχει το ύψος. Η ηλικία του ασθενή έχει ελαφρώς αρνητικό αντίκτυπο.

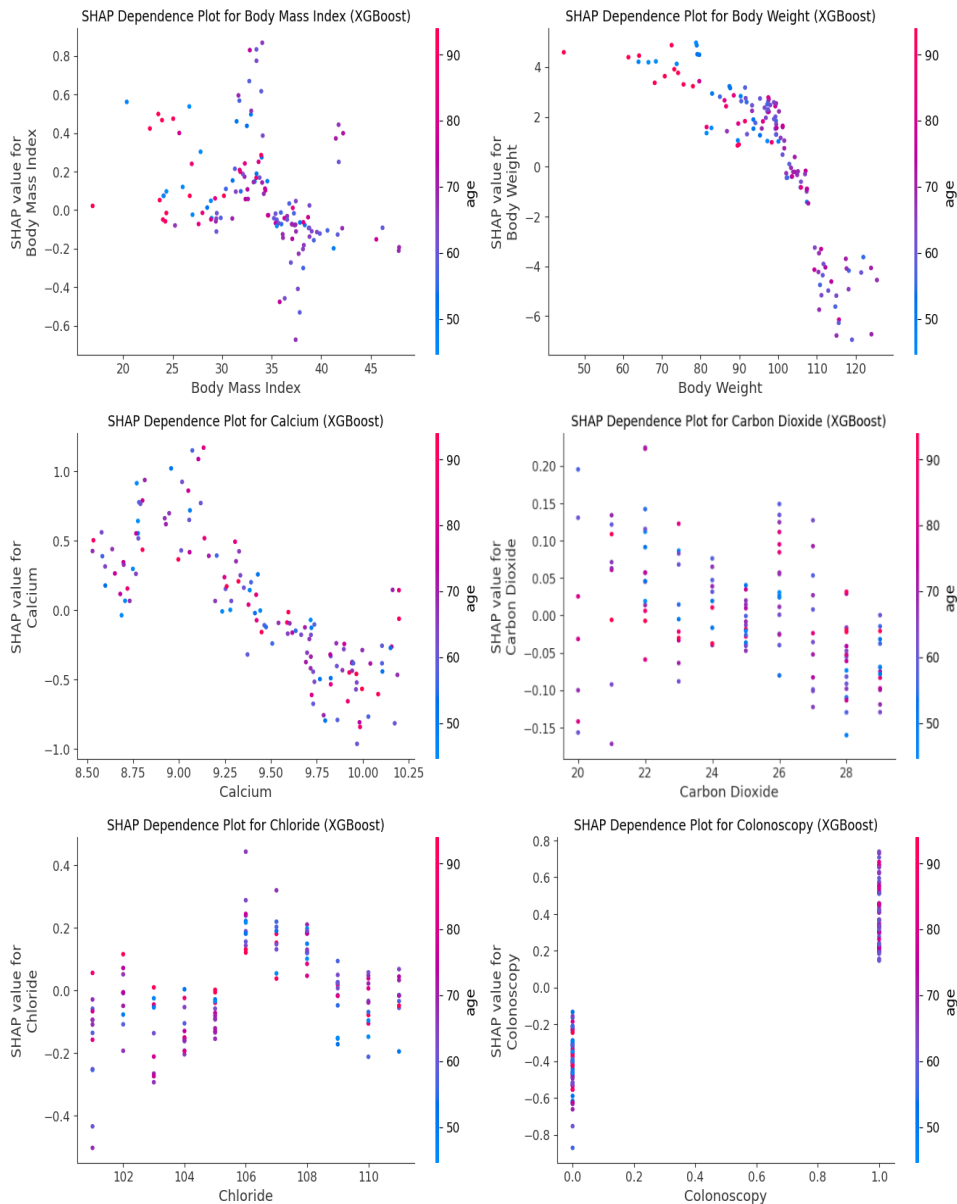


Γράφημα 13. WaterfallPlot και BeeswarmPlot του XGBoost (feature selected: gender)



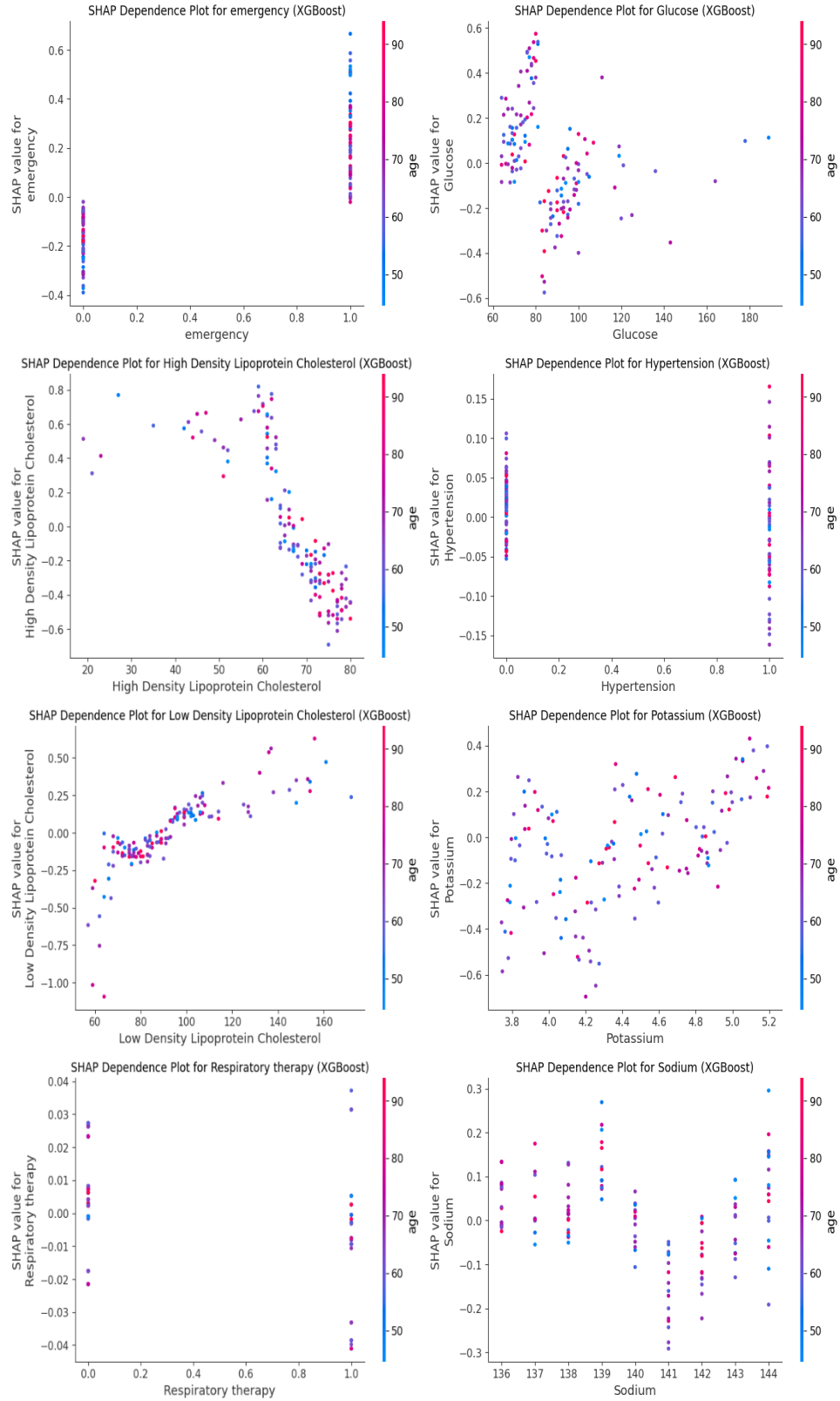
Γράφημα 14. SHAP DependencePlot του XGBoost για την ηλικία και το βάρος (feature selected: gender)

Στα dependence plots, παρατηρούμε ότι οι ηλικίες 60-70 ετών επιδρούν θετικά στο μοντέλο, ενώ νεαροί και ηλικιωμένοι επιδρούν αρνητικά. Επίσης άτομα με ύψος 150-170 εκατοστά επιδρούν πολύ θετικά, σε αντίθεση με άτομα μεγαλύτερου αναστήματος. Ανάλογη είναι η επίδραση και στο βάρος, όπου αυτό όσο αυξάνεται, τόσο χειρότερη είναι η επίδραση στο μοντέλο. Τα υψηλότερα επίπεδα ασβεστίου επίσης μειώνουν την απόδοση της πρόβλεψης. Τα άτομα που δεν υποβλήθηκαν σε κολονοσκόπηση (0) έχουν σταθερά αρνητικές τιμές SHAP, ενώ τα άτομα που υποβλήθηκαν (1) έχουν θετικές τιμές SHAP. Αυτό δείχνει ότι η κολονοσκόπηση αυξάνει την πιθανότητα θετικής έκβασης στο μοντέλο. Το ίδιο ισχύει και για ασθενείς που μπηγαν εκτάκτως στο νοσοκομείο.

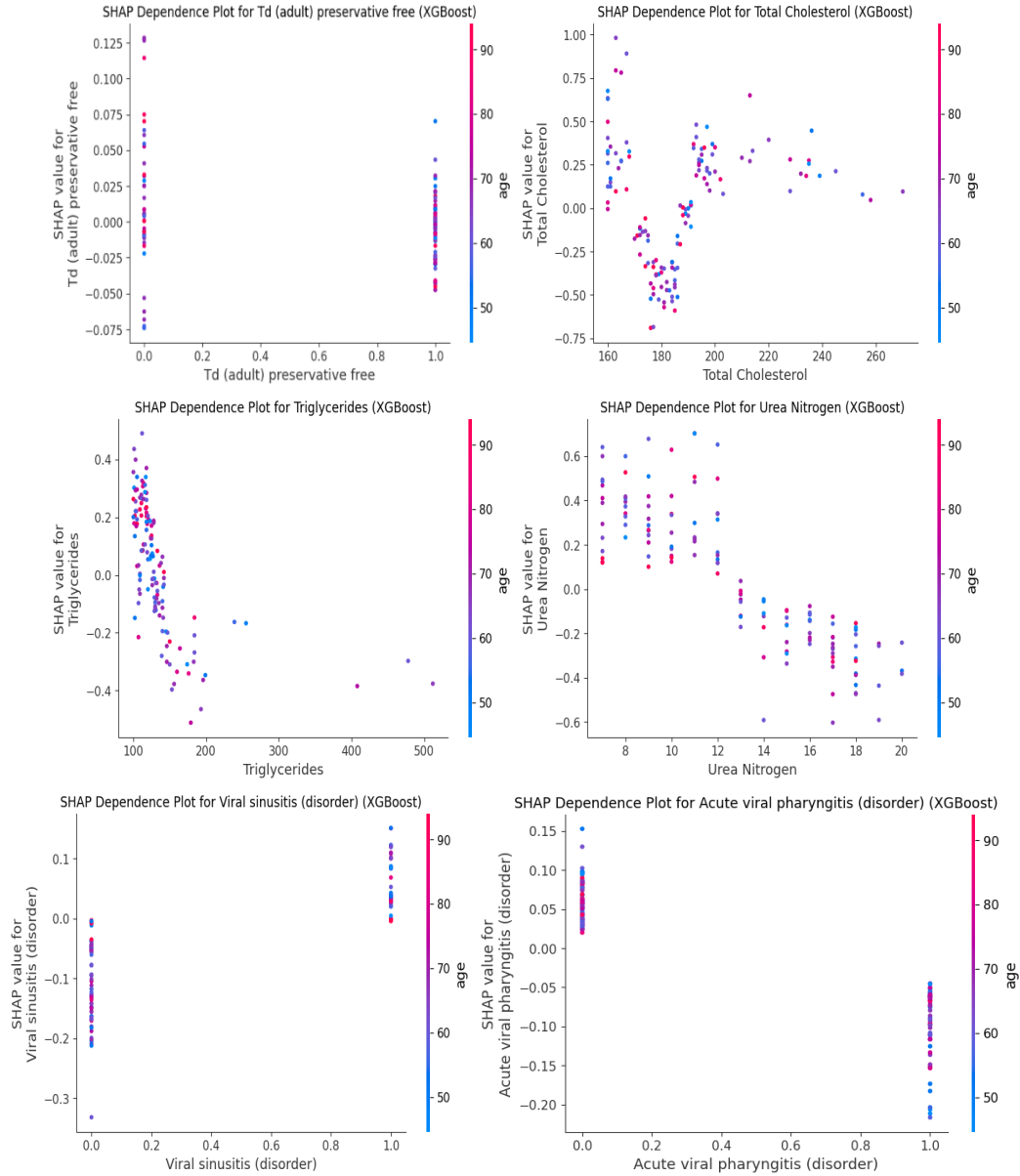


Γράφημα 15. SHAP DependencePlot του XGBoost για την ηλικία και το βάρος (feature selected: gender)

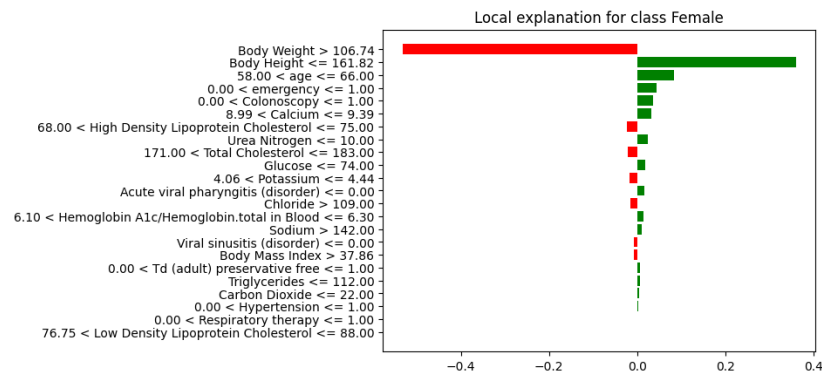
Επίσης συμπεραίνουμε ότι: χαμηλά επίπεδα γλυκόζης επιδρούν κυρίως θετικά στο μοντέλο, υψηλά επίπεδα «καλής» χοληστερόλης έχουν αρνητική επίδραση, ενώ στην «κακή» χοληστερίνη ισχύει το αντίθετο. Η συνολική χοληστερίνη επιδρά κυρίως θετικά με εξαίρεση τις τιμές 170 έως 190. Τριγλυκερίδια γύρω στο 100 έχουν πολύ θετική επιρροή στο μοντέλο, όσο αυξάνονται τα επίπεδα ουρίας μειώνεται η απόδοση του, ύπαρξη ιογενούς ιγμορίτιδας επιφέρει θετικά αποτελέσματα, όπως και η μη ύπαρξη οξείας ιογενούς φαρυγγίτιδας.



Γράφημα 16. SHAP DependencePlot του XGBoost για διάφορα χαρακτηριστικά (feature selected: gender)

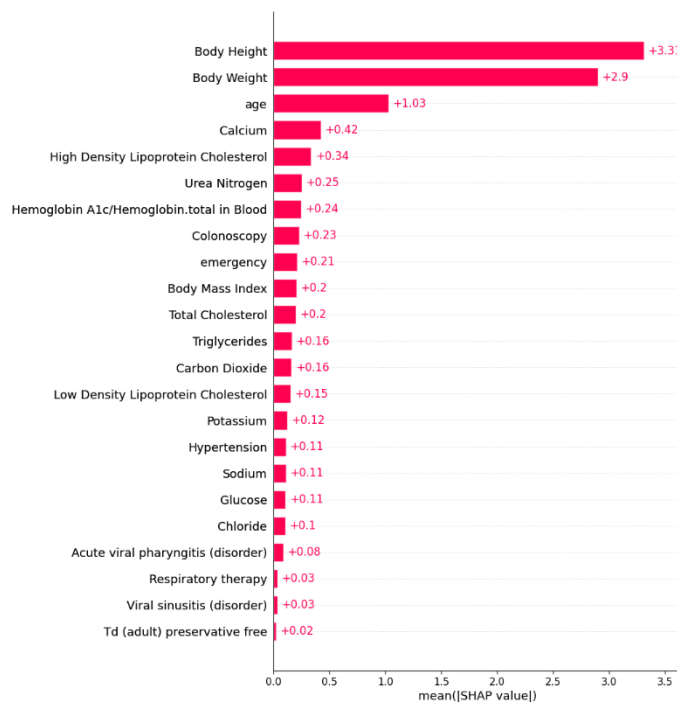


Γράφημα 17. SHAP DependencePlot του XGBoost για διάφορα χαρακτηριστικά (feature selected: gender)

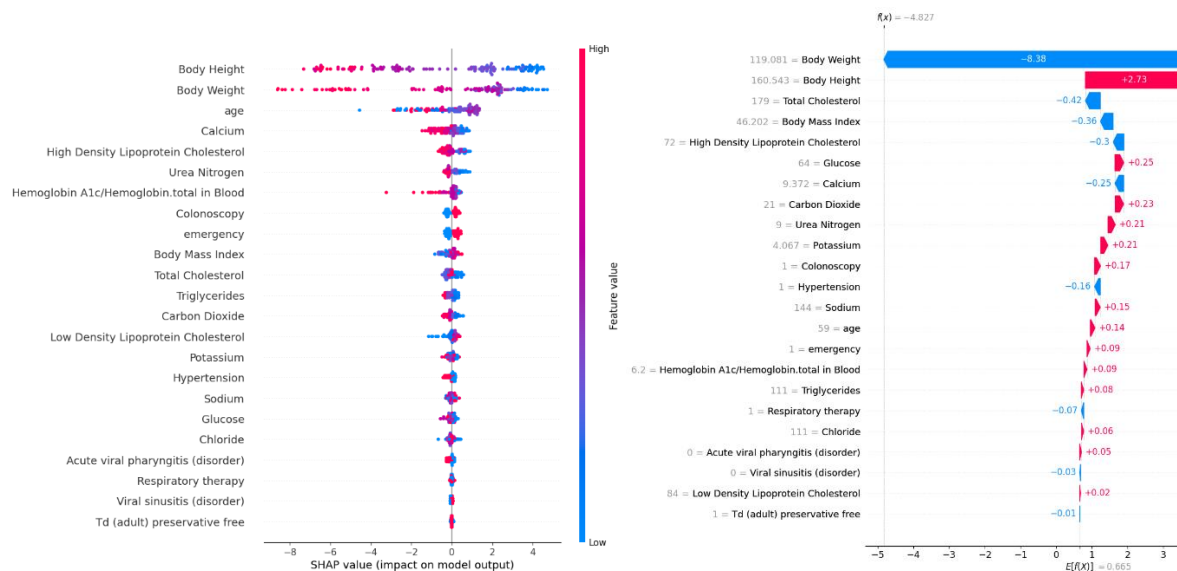


Γράφημα 18. LIME plot για την θετική και αρνητική επίδραση κάθε χαρακτηριστικού (feature selected: gender)

CatBoost



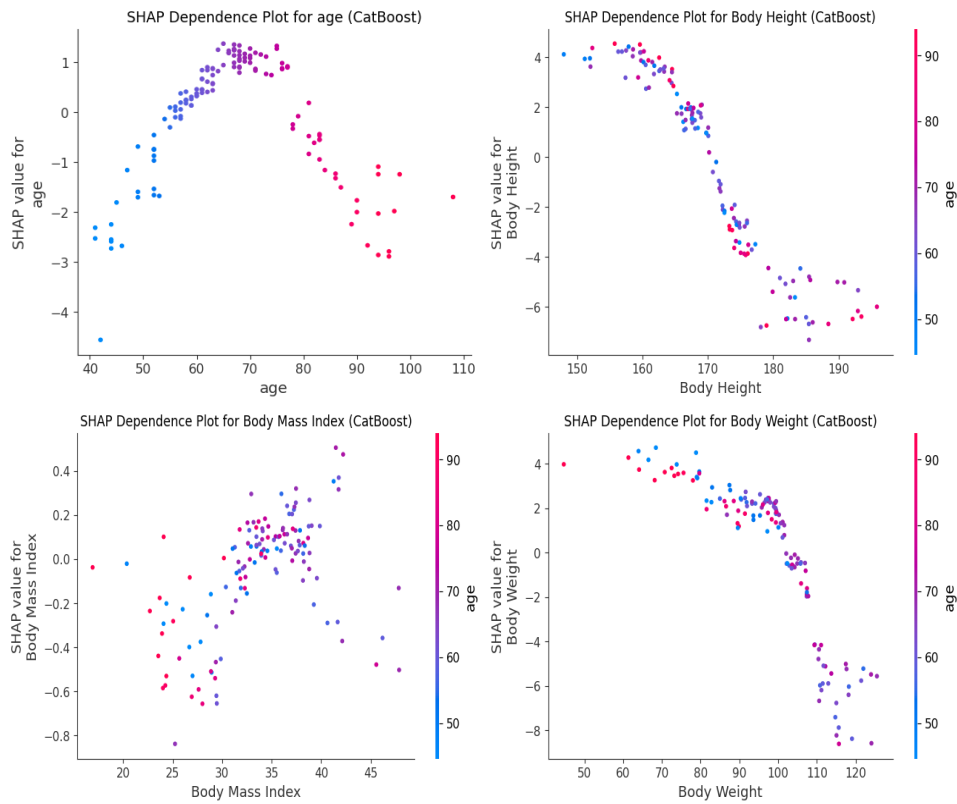
Γράφημα 19. SHAP BarPlot του CatBoost (feature selected: gender)

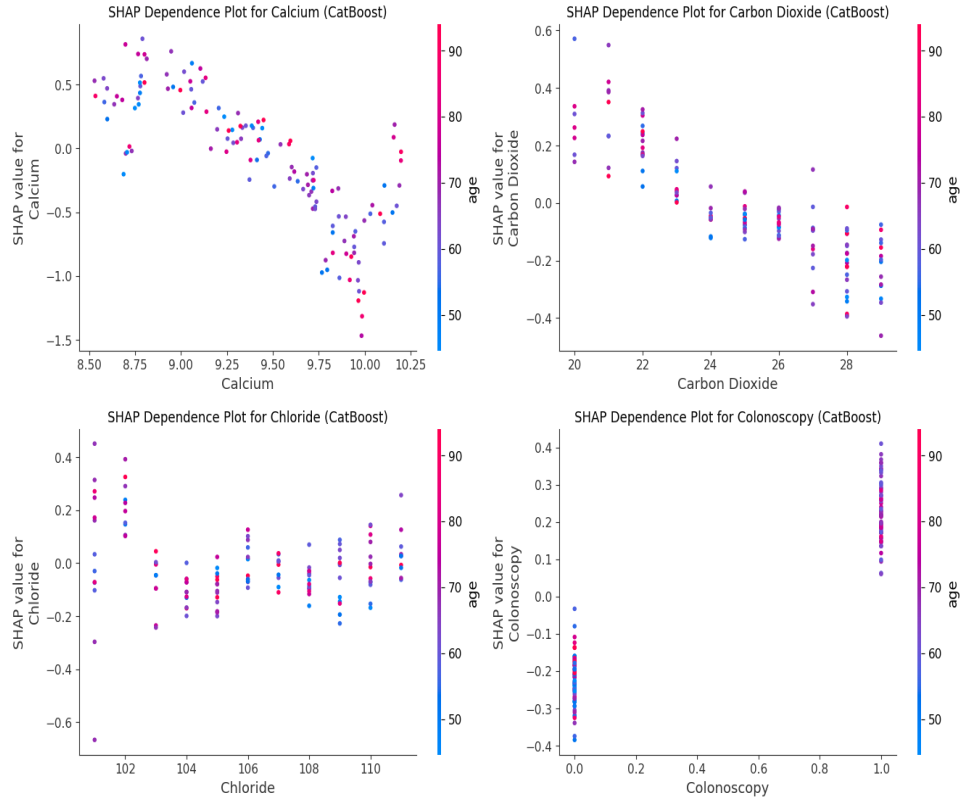


Γράφημα 20. WaterfallPlot και BeeswarmPlot του XGBoost (feature selected: gender)

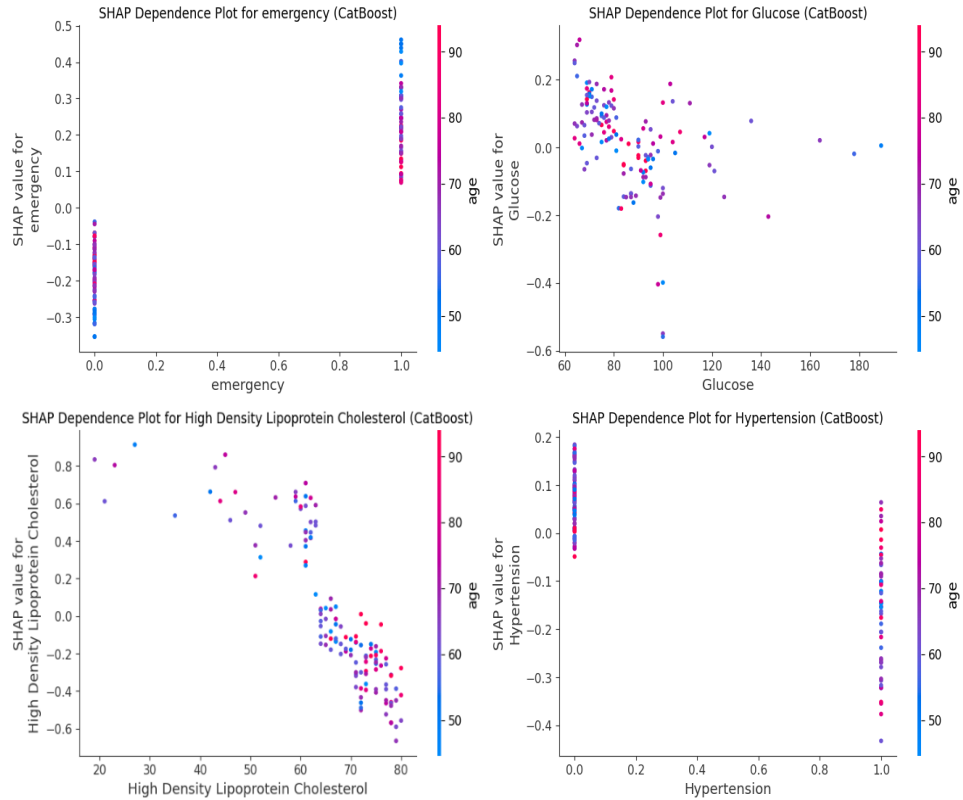
Και στον CatBoost, μεγαλύτερη επίδραση στην τελική πρόβλεψη του μοντέλου έχουν το ύψος και το βάρος των ασθενών, με το βάρος να επιδρά αρνητικά σε μεμονωμένη πρόβλεψη και το ύψος θετικά κυρίως στα άτομα με ανάστημα 150 έως 170 εκατοστά. Χαμηλές τιμές βάρους, ασβεστίου και διοξειδίου του άνθρακα έχουν θετική επιρροή, καθώς και οι μέσες τιμές δείκτη μάζας σώματος. Θετικά επιδρά όπως και στον XGBoost, η υποβολή σε κολονοσκόπηση, η έκτακτη εισαγωγή στο νοσοκομείο, τα πολύ χαμηλά επίπεδα γλυκόζης, οι τιμές «κακής»

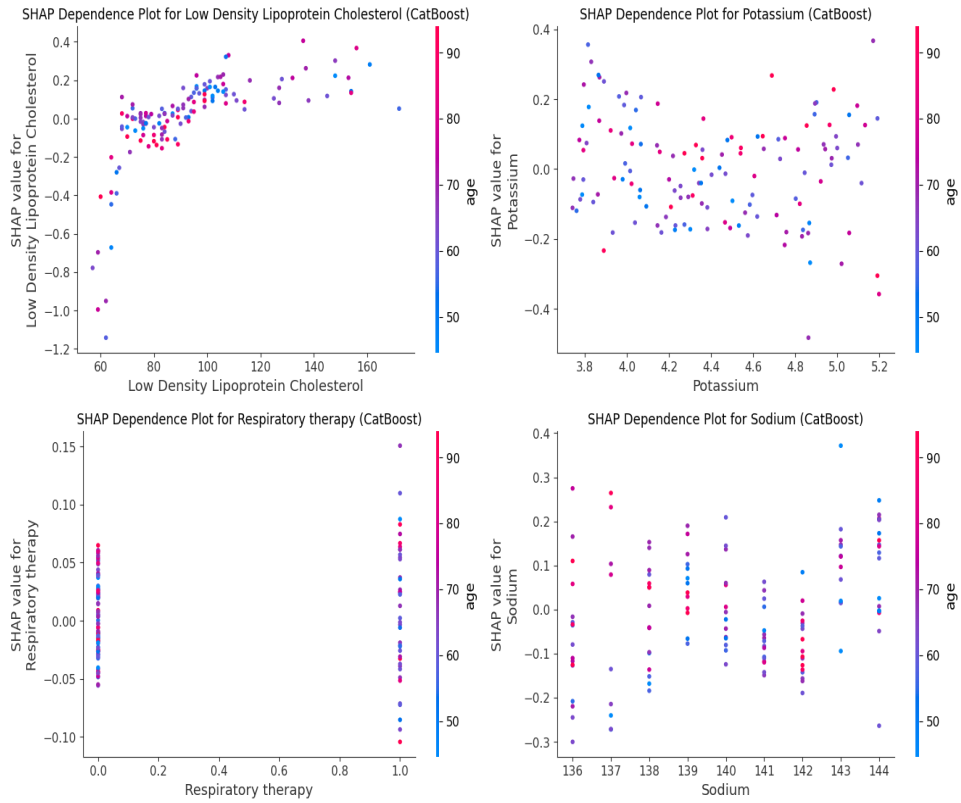
χοληστερόλης στο 80-120, καθώς και τα χαμηλά επίπεδα σε νιτρογενική ουρία και τριγλυκερίδια.



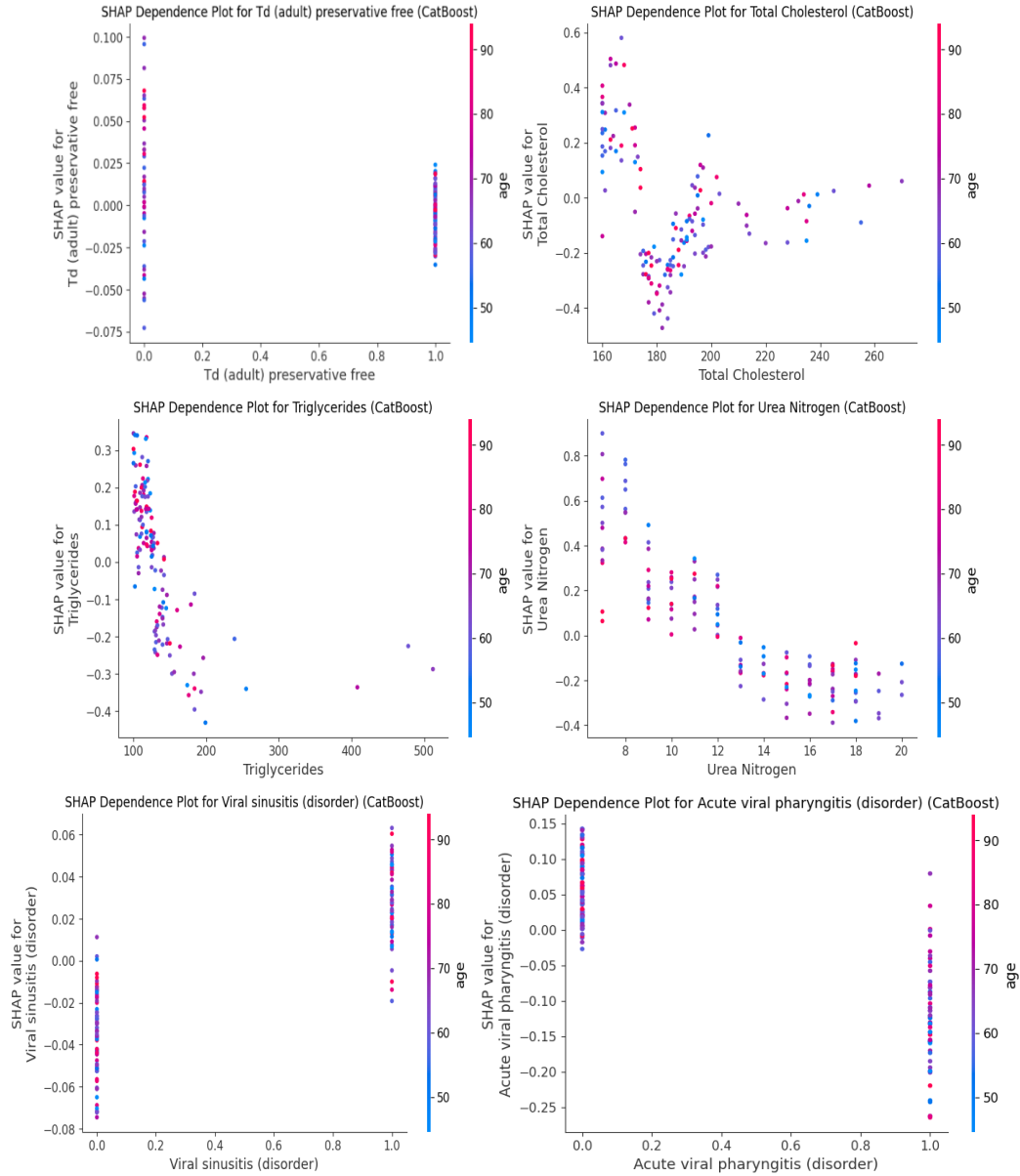


Γράφημα 21. SHAP DependencePlots του CatBoost για διάφορα χαρακτηριστικά (feature selected: gender)





Γράφημα 22. SHAP DependencePlots του CatBoost για διάφορα χαρακτηριστικά (feature selected: gender)



Γράφημα 23. SHAP DependencePlots του CatBoost για διάφορα χαρακτηριστικά (feature selected: gender)

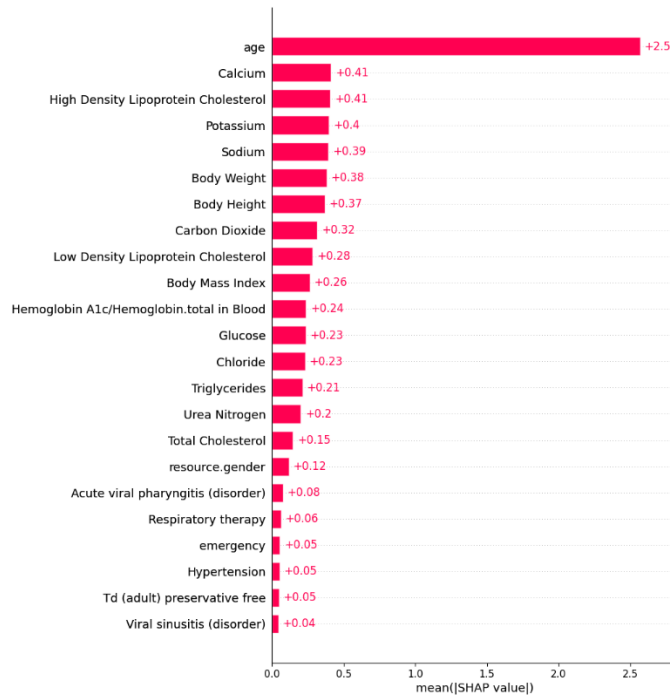


Γράφημα 24. LIME plot για την θετική και αρνητική επίδραση κάθε χαρακτηριστικού (feature selected: gender)

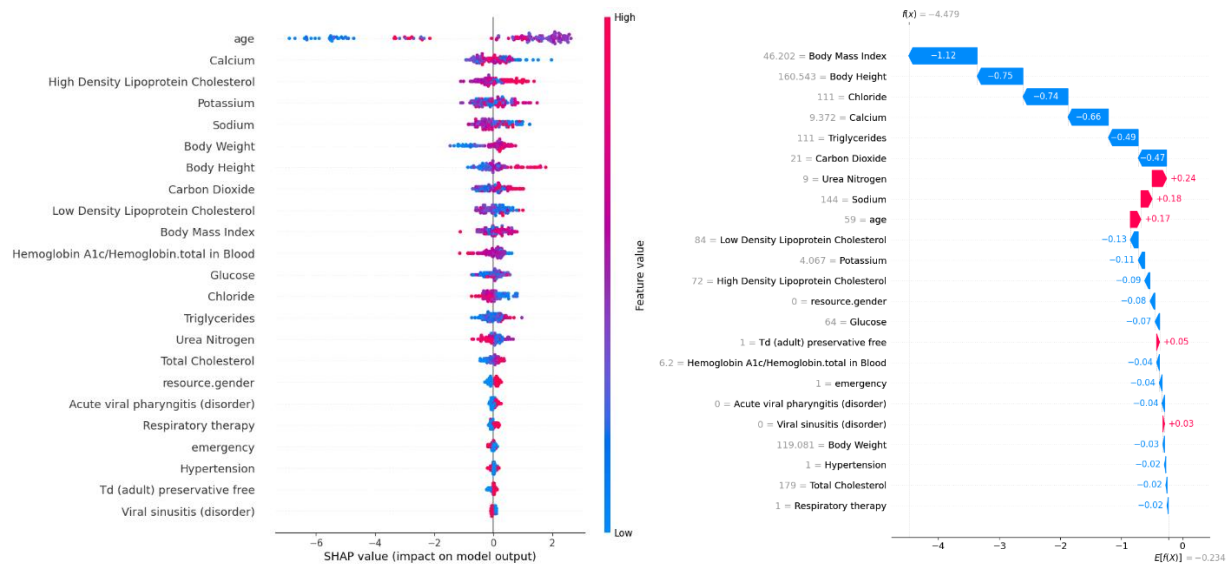
4.2.3 Classification ως προς την κλάση colonoscopy

Παρόμοιο ήταν το ποσοστό επιτυχίας για τους δύο αλγορίθμους. Η επιλεγμένη κλάση παίρνει δύο τιμές, 0 για τα άτομα που δεν έχουν υποβληθεί σε κολονοσκόπηση και 1 για τα άτομα που έχουν υποβληθεί.

XGBoost



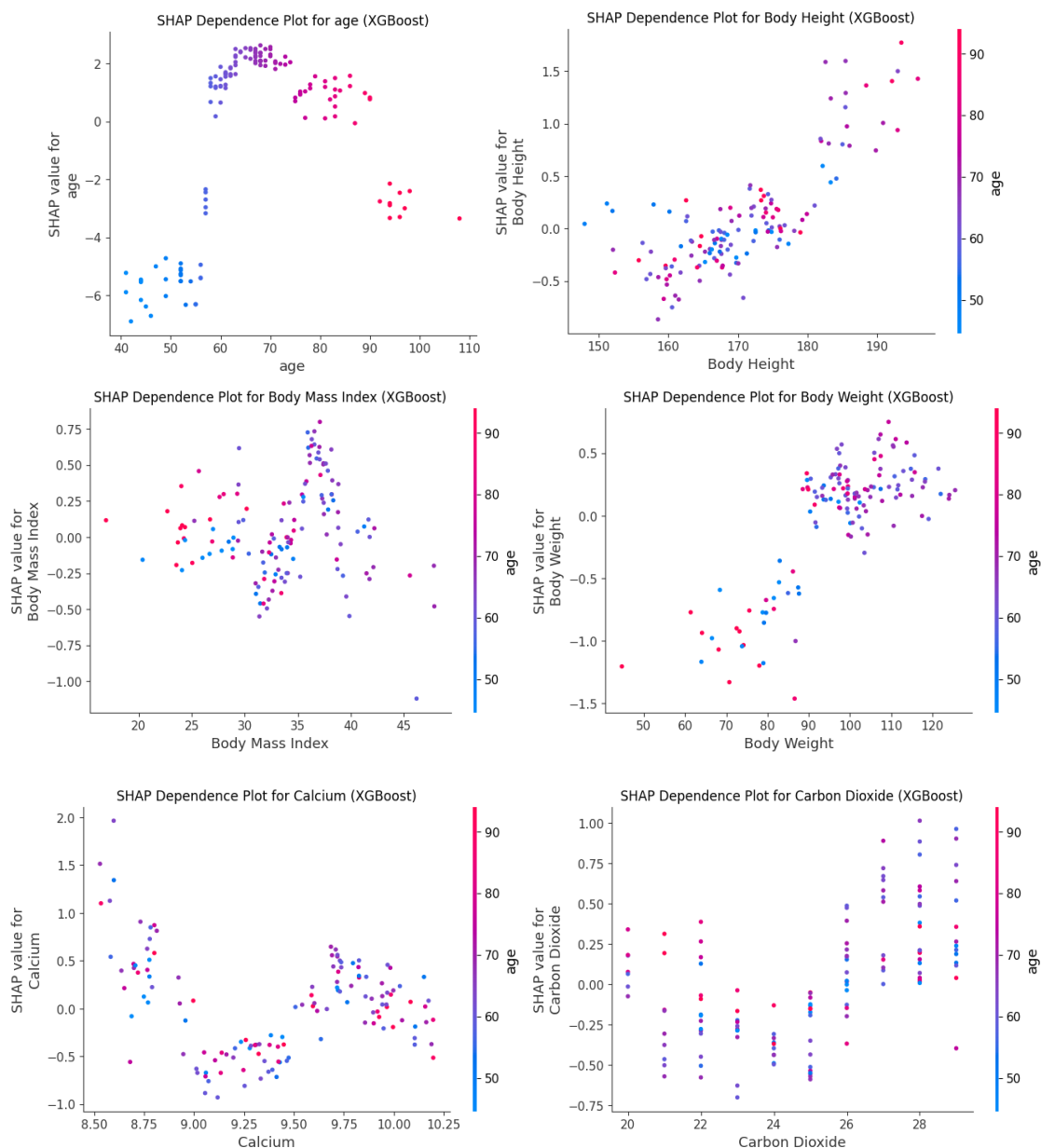
Γράφημα 25. SHAP BarPlot του XGBoost (feature selected: colonoscopy)



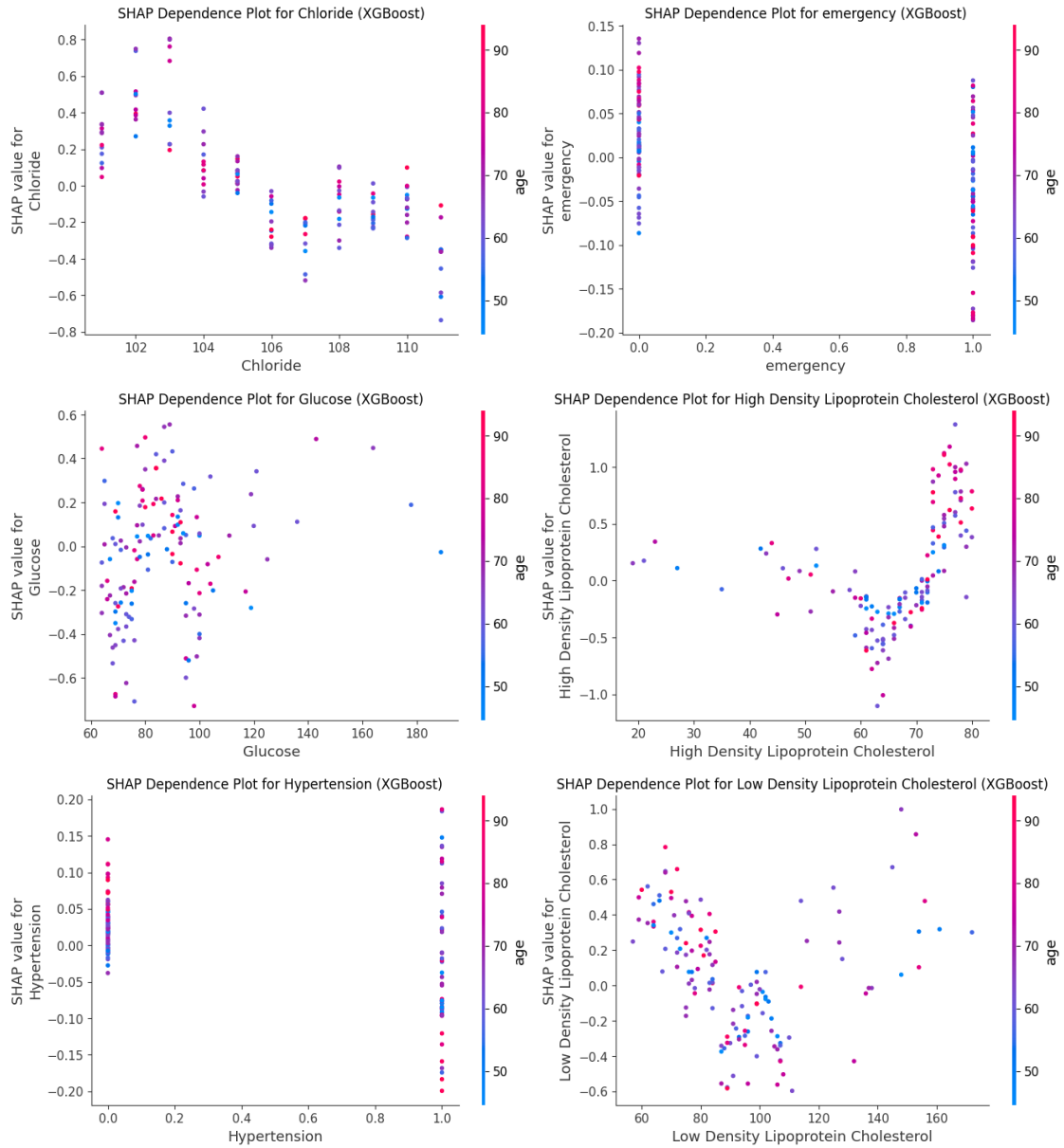
Γράφημα 26. SHAP BeeswarmPlot και WaterfallPlot του XGBoost (feature selected: colonoscopy)

Η ηλικία έχει σημαντική επίδραση στο μοντέλο κυρίως αρνητική, αν και για συγκεκριμένη πρόβλεψη ισορροπείται. Αρνητική επίδραση για μεμονωμένη πρόβλεψη έχουν και εδώ το βάρος και το ύψος.

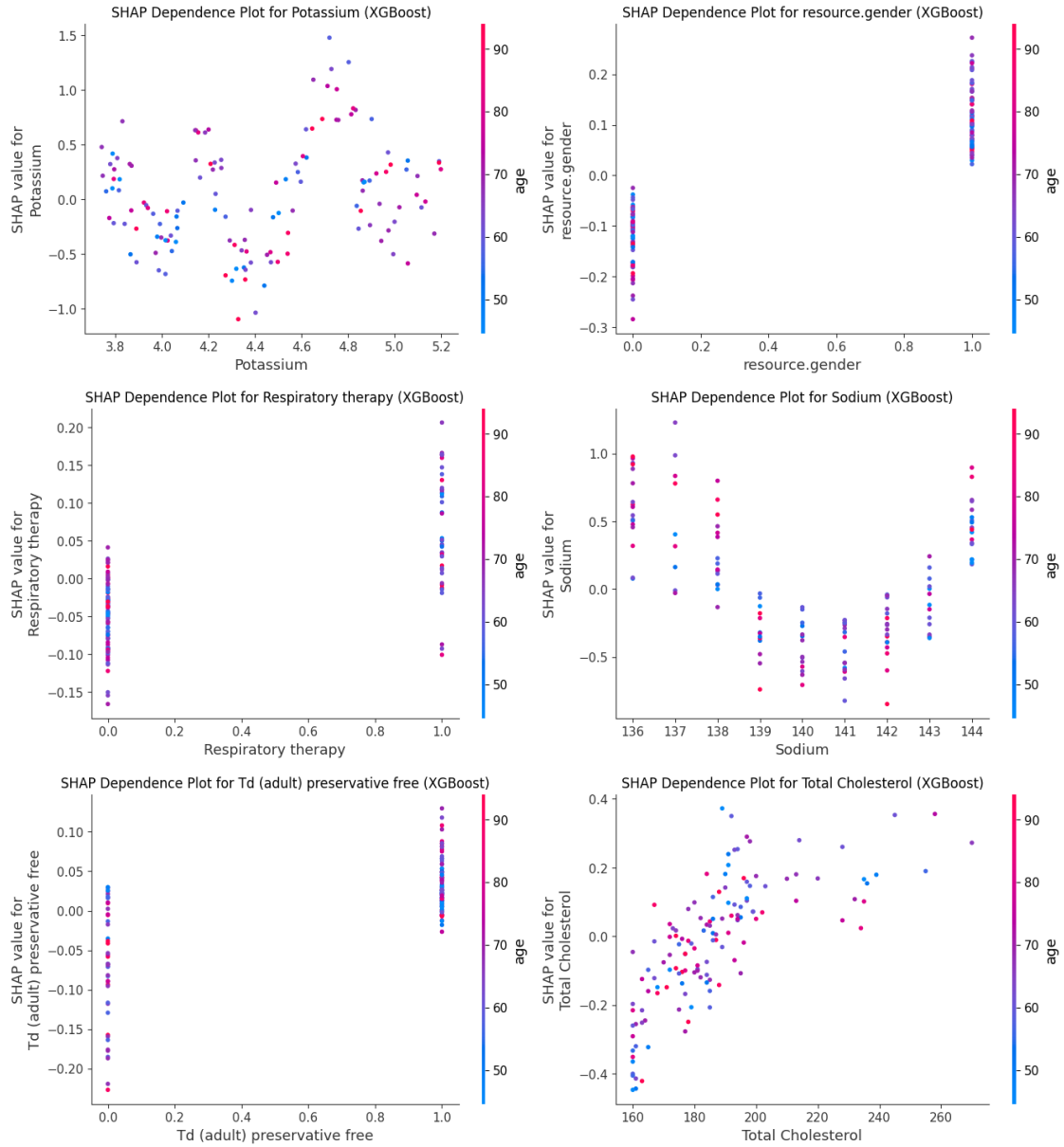
Το μοντέλο επηρεάζεται αρνητικά από τις ηλικίες 40-60 και θετικά από τις ηλικίες 60-80, θετικά από άτομα με υψηλό ανάστημα και βάρος, ενώ οι μέσες τιμές ασβεστίου και διοξειδίου επιδρούν αρνητικά. Θετική επίδραση έχουν: τα χαμηλά επίπεδα χλωριδίου, οι υψηλές τιμές «καλής» χοληστερόλης και οι χαμηλές τιμές «κακής» χοληστερόλης, το γυναικείο φύλο, οι εμβολιασμένοι με το Td vaх, τα άτομα με χαμηλή ουρία και τα άτομα με οξεία ιογενή φαρυγγίτιδα.



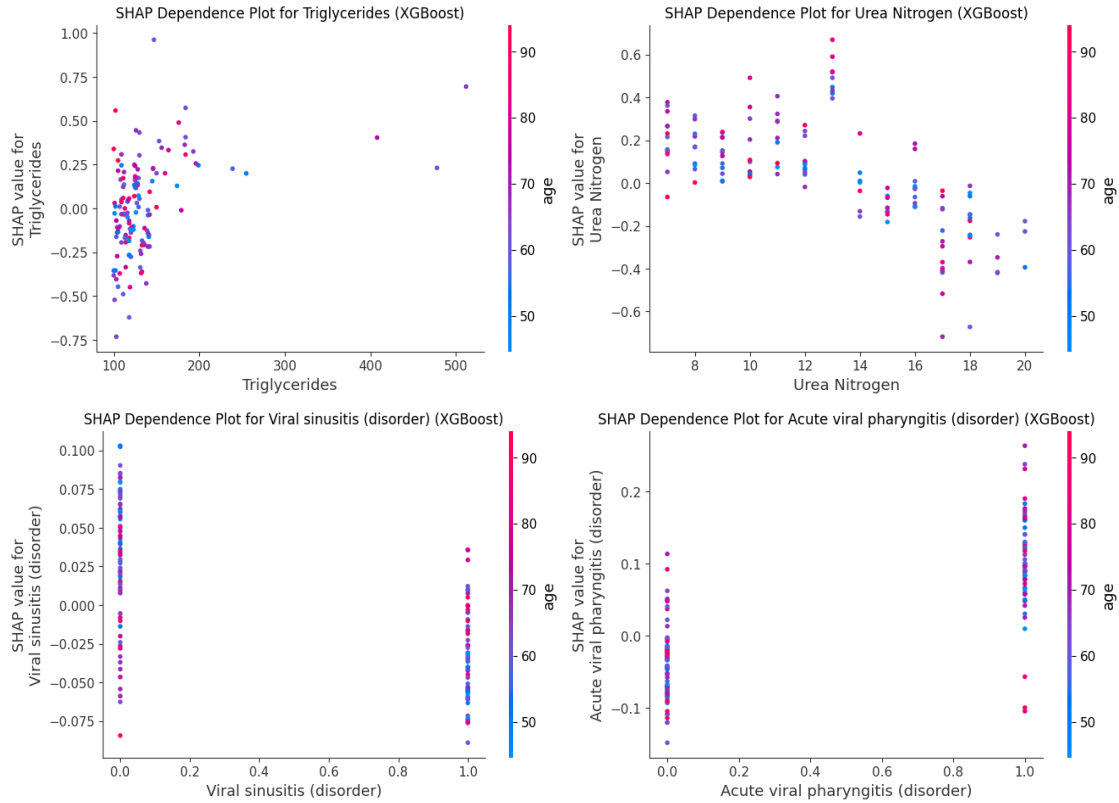
Γράφημα 27. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)



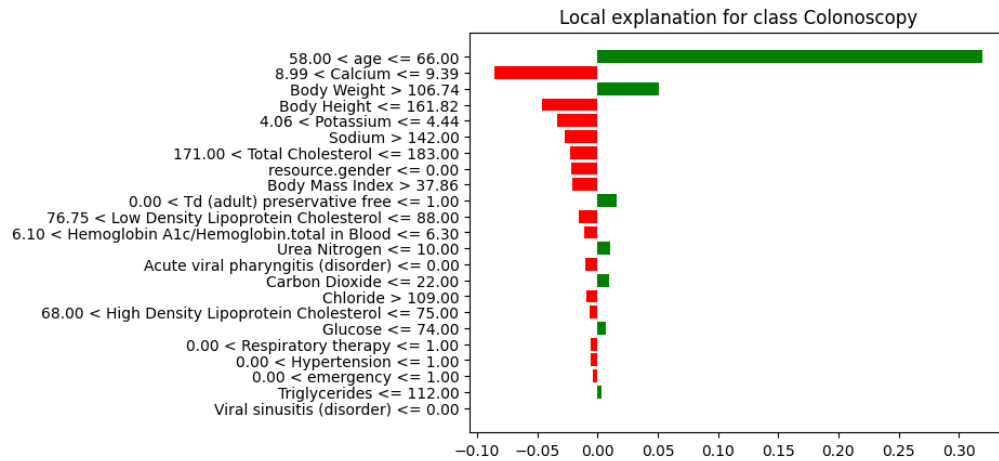
Γράφημα 28. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)



Γράφημα 29. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (*feature selected: colonoscopy*)

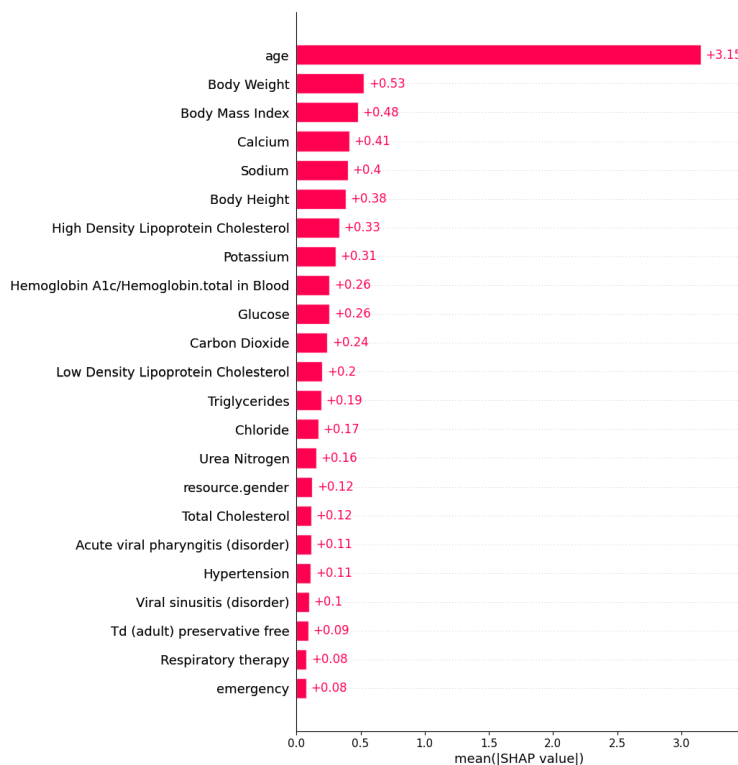


Γράφημα 30. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)

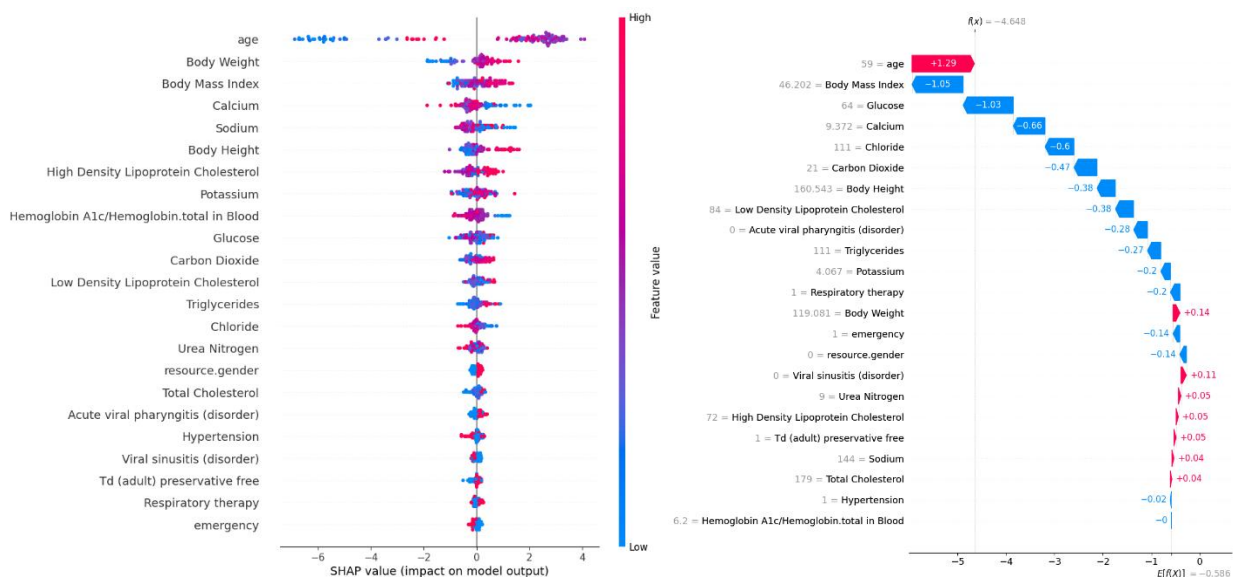


Γράφημα 31. LIME plot για την θετική και αρνητική επίδραση κάθε χαρακτηριστικού (feature selected: colonoscopy)

CatBoost

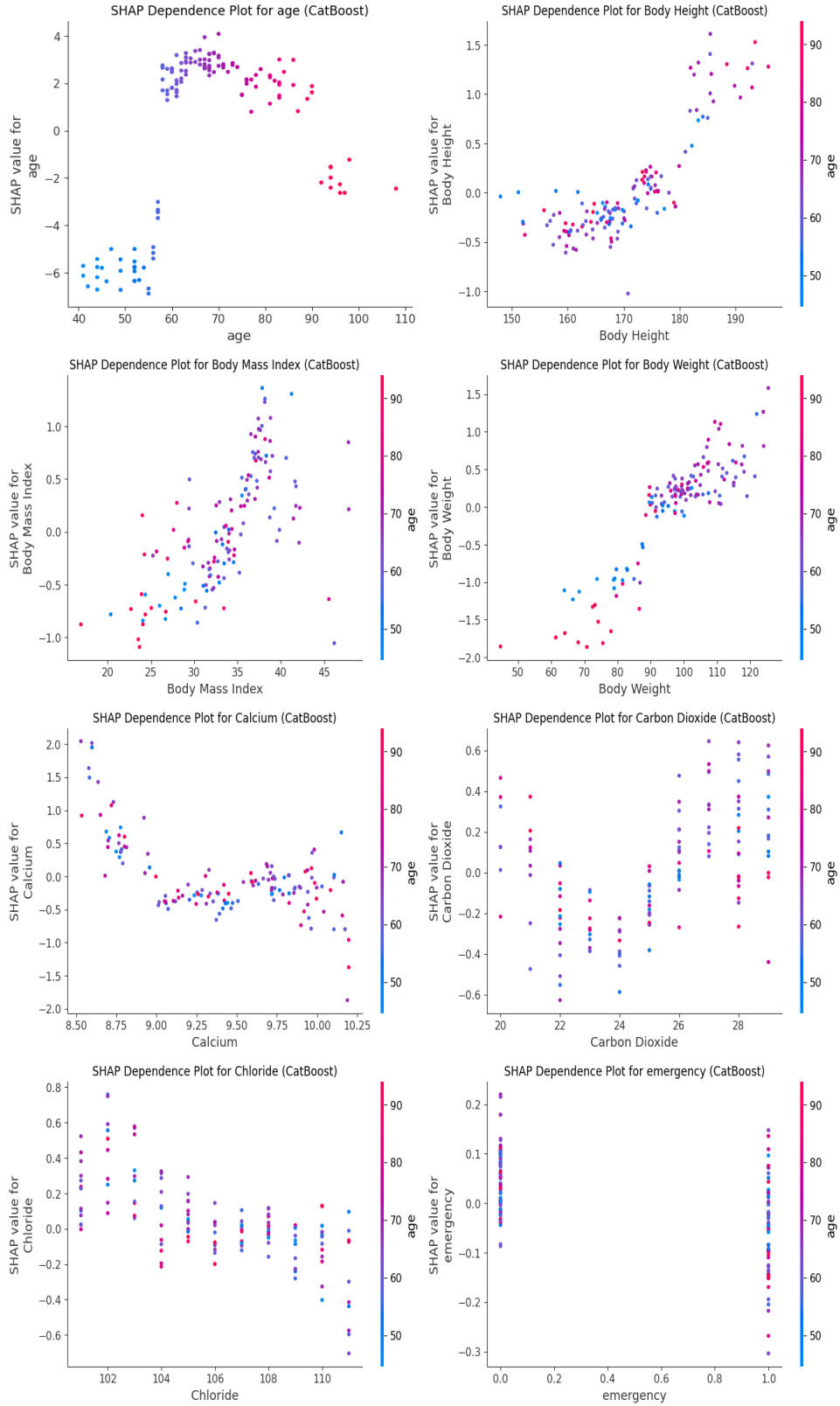


Γράφημα 32. SHAP BarPlot του CatBoost (feature selected: colonoscopy)

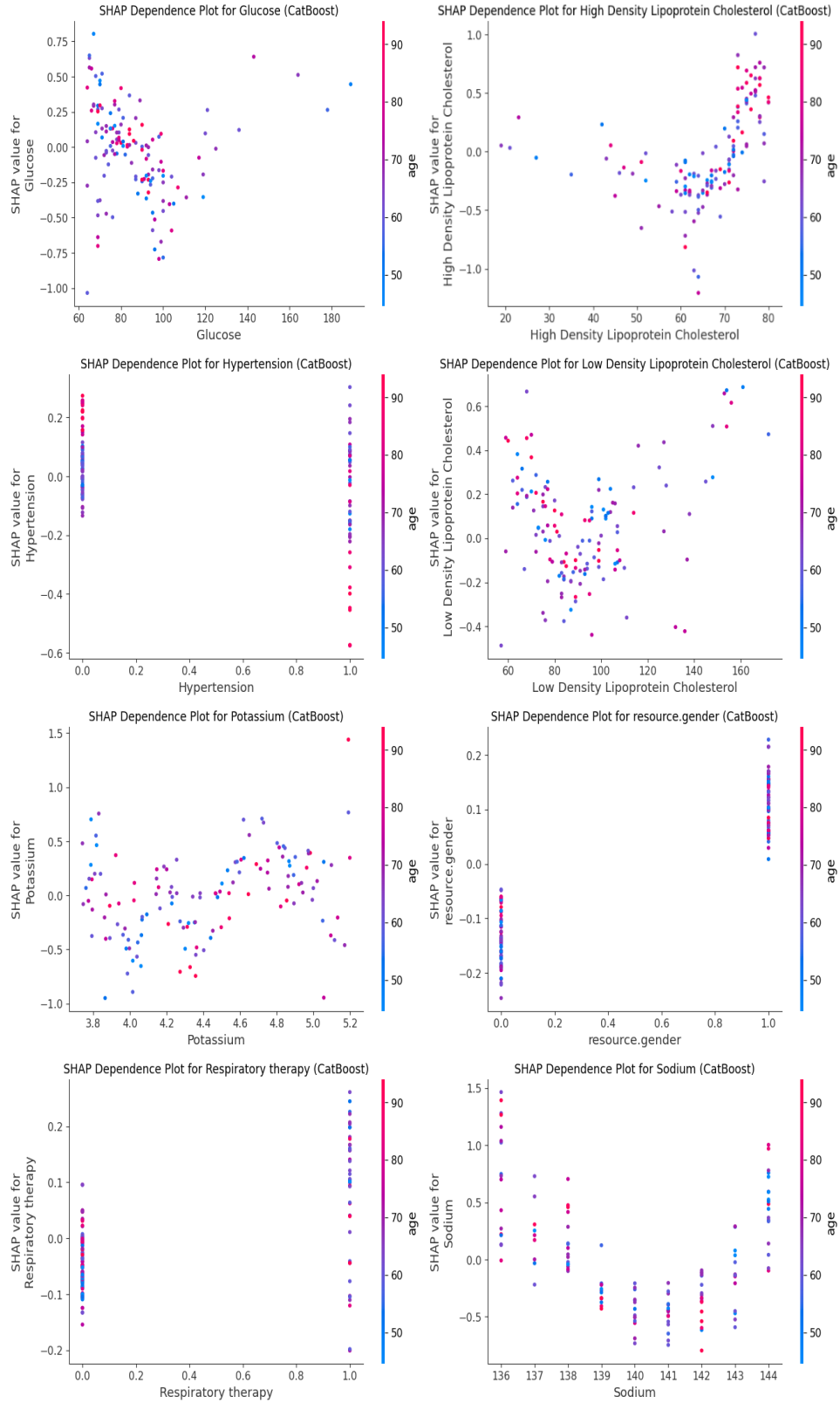


Γράφημα 33. SHAP BeeswarmPlot και WaterfallPlot του CatBoost (feature selected: colonoscopy)

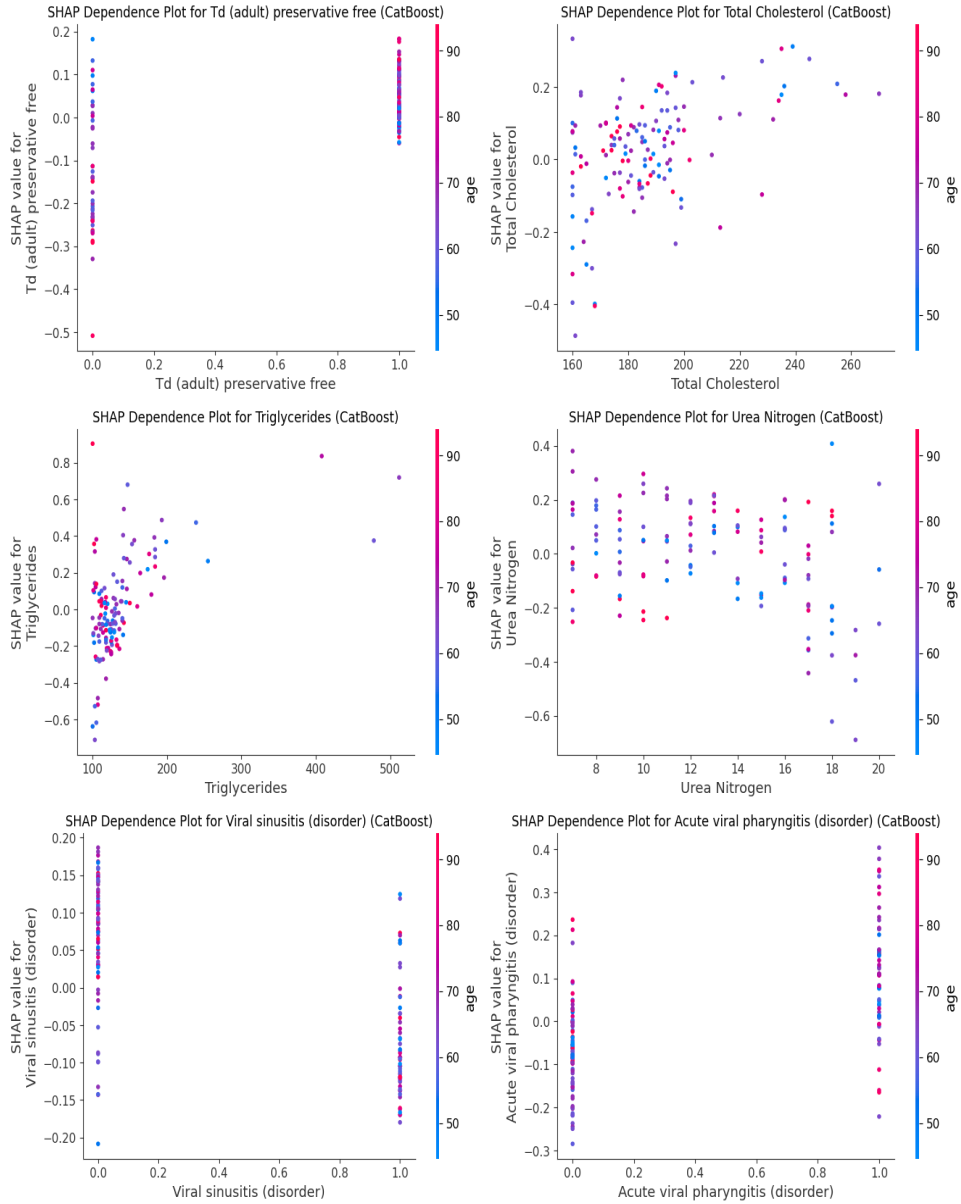
Και στον CatBoost η ηλικία έχει σημαντική επίδραση αλλά εδώ θετική, ενώ τα υπόλοιπα χαρακτηριστικά έχουν χαμηλή συνολική επίδραση. Θετικό αντίκτυπο στο μοντέλο έχουν, οι ηλικίες 60 έως 90, τα άτομα με ύψος 1.75 έως 1.95, τα άτομα με δείκτη μάζας σώματος 35 έως 40, τα άτομα με βάρος άνω των 90, αλλά και οι χαμηλές τιμές ασβεστίου.



Γράφημα 34. SHAP DependencePlots του CatBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)

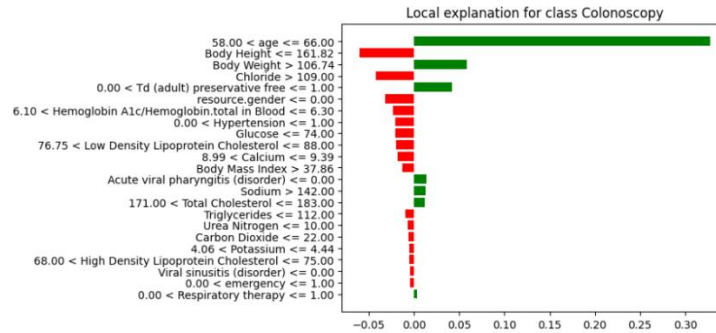


Γράφημα 35. SHAP DependencePlots του CatBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)



Γράφημα 36. SHAP DependencePlots του CatBoost για διάφορα χαρακτηριστικά (feature selected: colonoscopy)

Το γυναικείο φύλο, οι υψηλές τιμές «καλής» χοληστερόλης και τα άτομα με ιογενή ιγμορίτιδα είχαν θετική επιρροή στο μοντέλο.



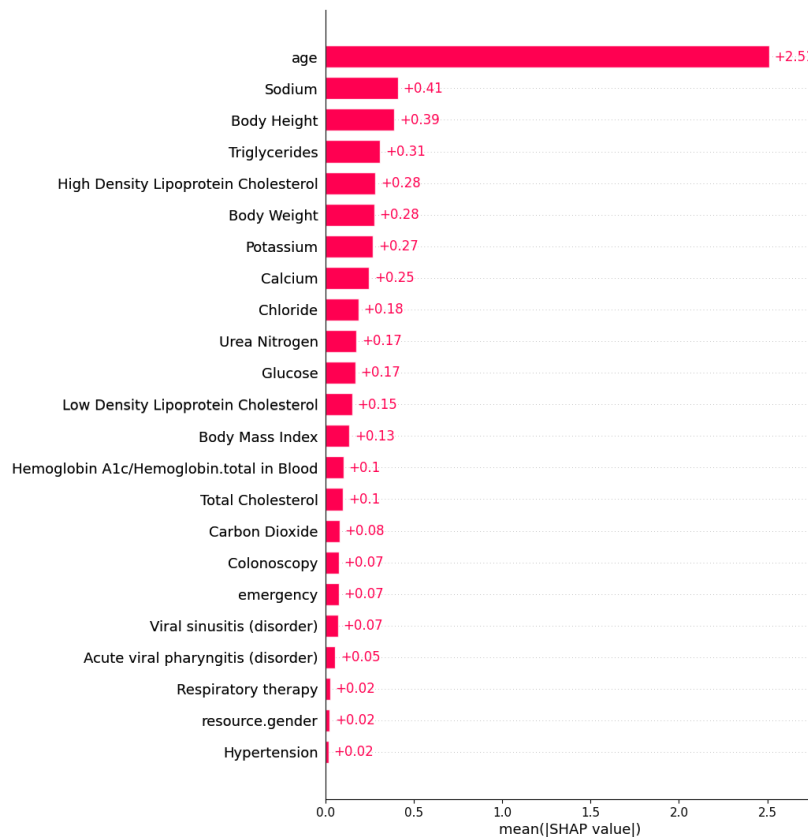
Γράφημα 37. LIME plot για την θετική και αρνητική επίδραση κάθε χαρακτηριστικού (feature selected: colonoscopy)

4.2.4 Classification ως προς την κλάση Td vax (Εμβόλιο για τέτανο και διφθερίτιδα)

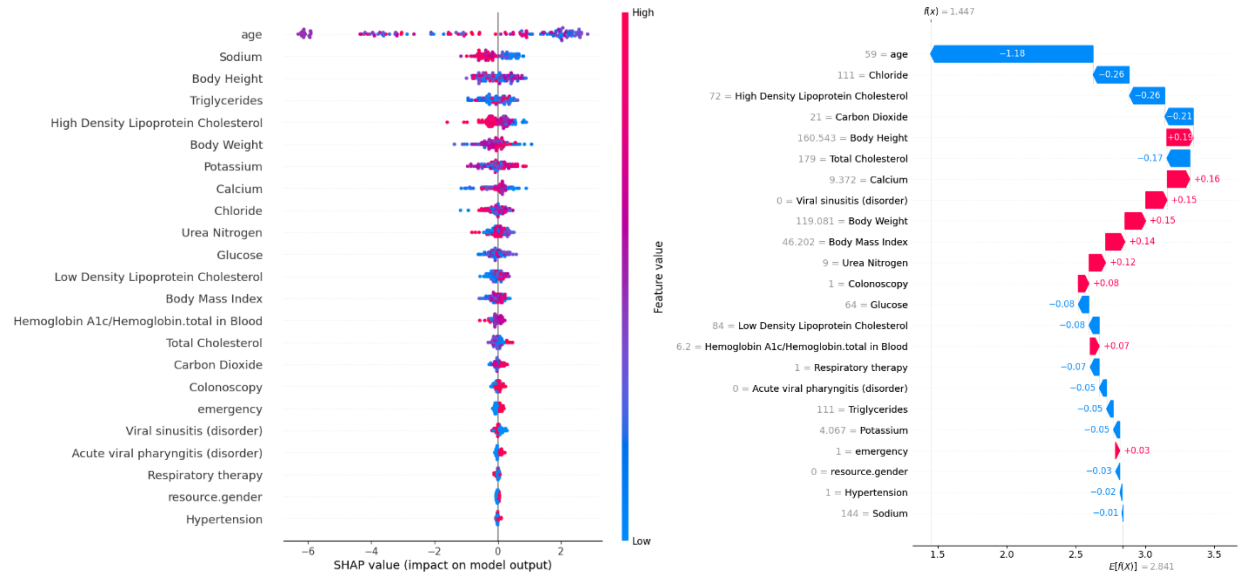
Μεγαλύτερο ήταν το ποσοστό επιτυχίας για τον XGBoost από τον CatBoost, ο οποίος για επιλεγμένο χαρακτηριστικό τον εμβολιασμό με το Td vax, πέτυχε τα μεγαλύτερα σκορ σε σχέση με τα άλλα δύο χαρακτηριστικά. Η επιλεγμένη κλάση παίρνει δύο τιμές, 0 για τους ανεμβολίαστους και 1 για τους εμβολιασμούς.

XGBoost

Μεγαλύτερη επίδραση αλλά ταυτόχρονα αρνητική, έχει η ηλικία, με τα υπόλοιπα χαρακτηριστικά να έχουν μικρότερη επίδραση.

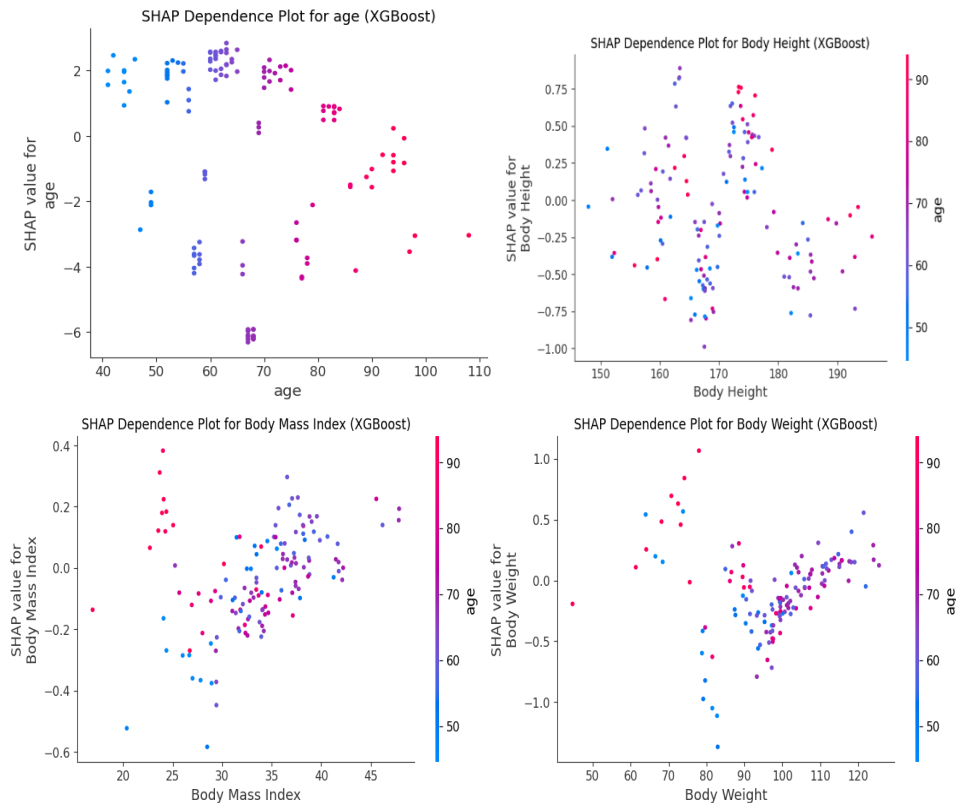


Γράφημα 38. SHAP BarPlot του XGBoost (feature selected: Td vax)

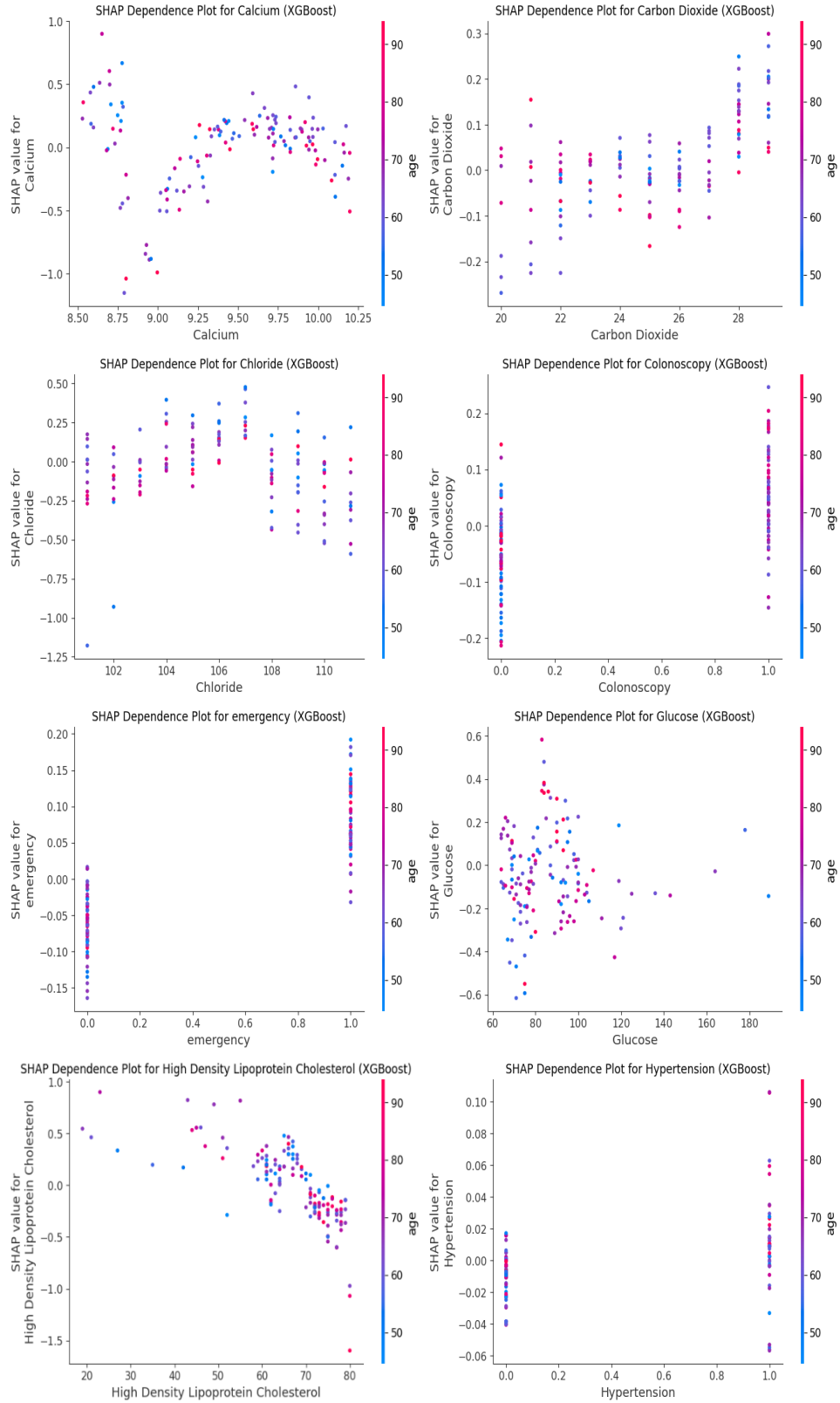


Γράφημα 39. SHAP BeeswarmPlot και WaterfallPlot του XGBoost (feature selected: Td vax)

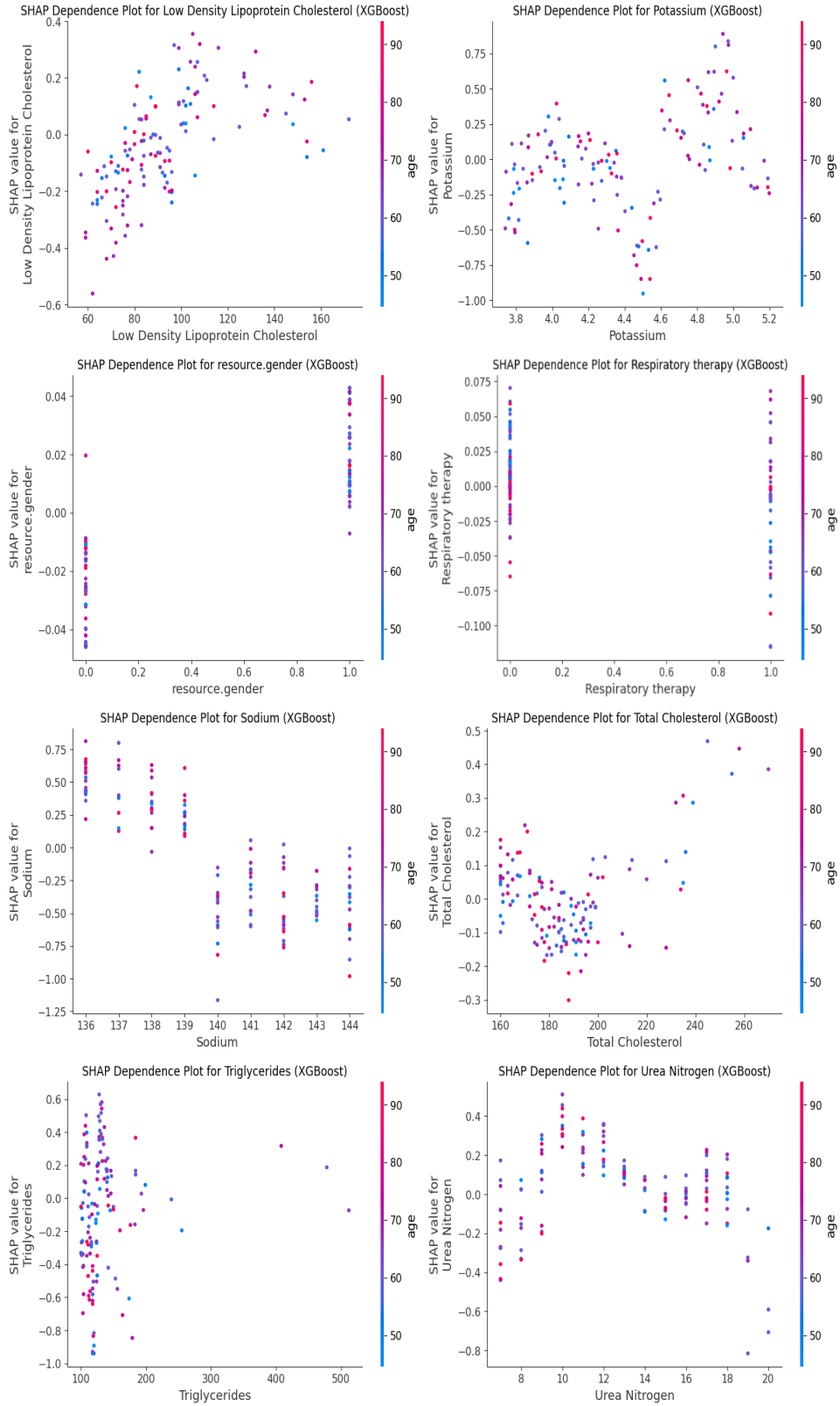
Θετική επιρροή έχουν οι υψηλότερες τιμές διοξειδίου του άνθρακα, η εκτέλεση κολονοσκόπησης, τα έκτακτα περιστατικά, το υψηλό κάλιο, το γυναικείο φύλο και τα χαμηλά επίπεδα νατρίου.



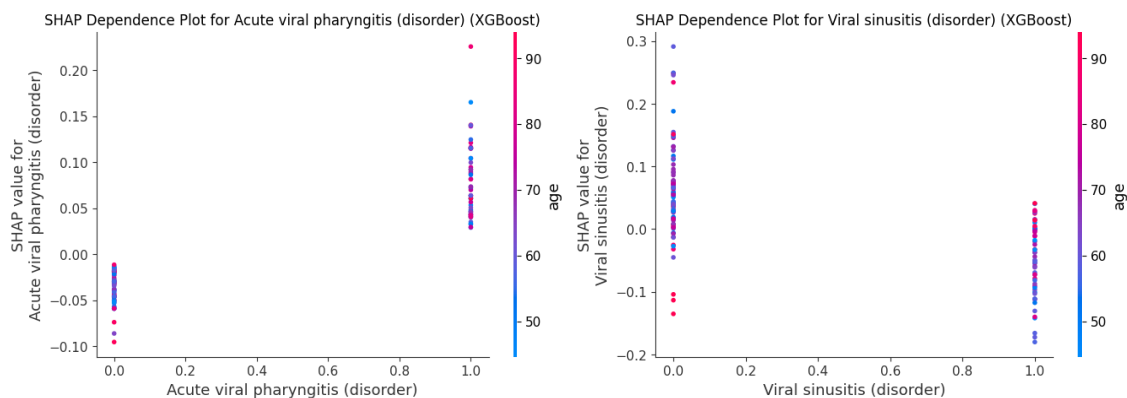
Γράφημα 40. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax)



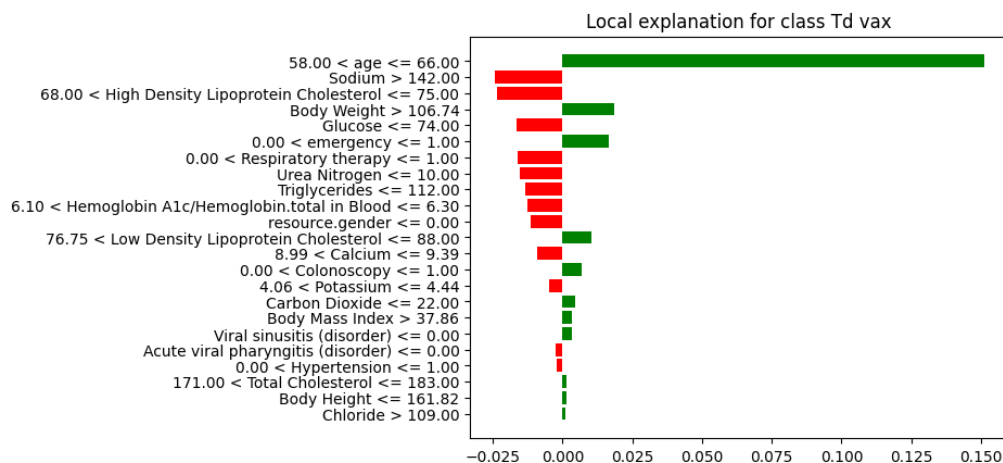
Γράφημα 41. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax)



Γράφημα 42. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax)



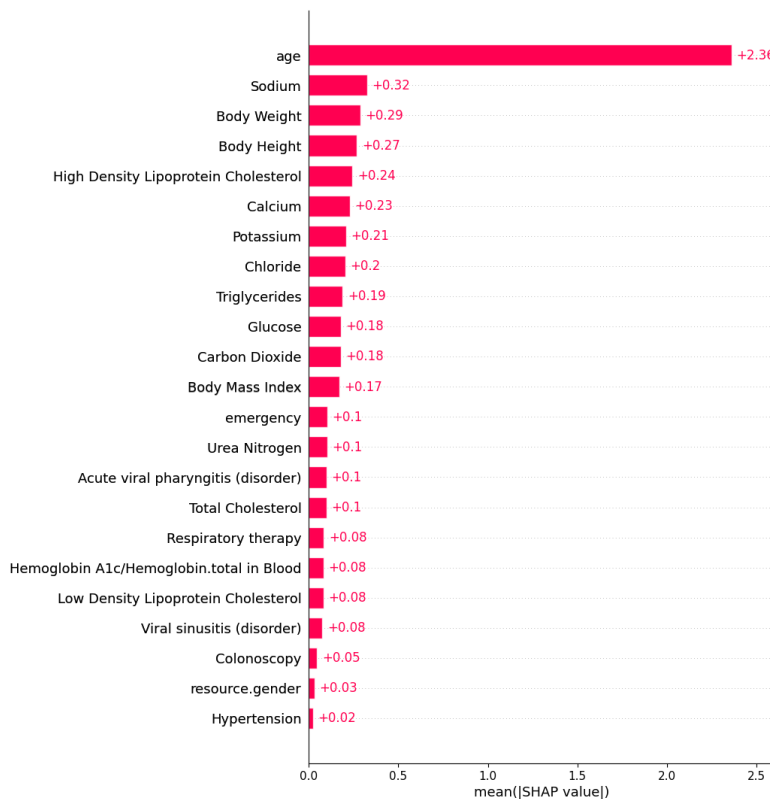
Γράφημα 43. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax)



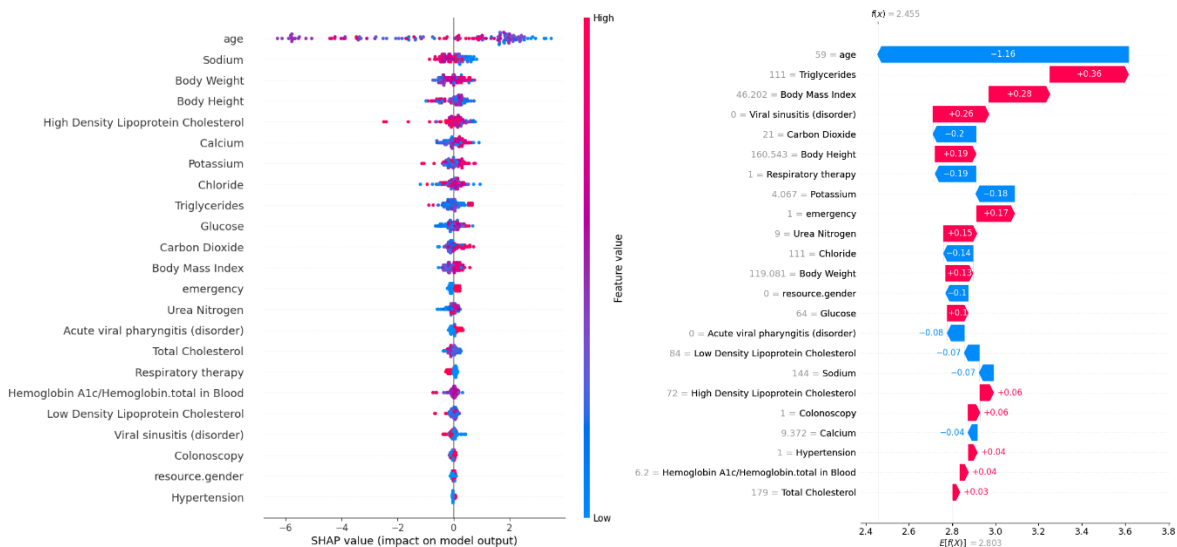
Γράφημα 44. LIME plot για την θετική και αρνητική επίδραση κάθε χαρακτηριστικού (feature selected: Td vax)

Catboost

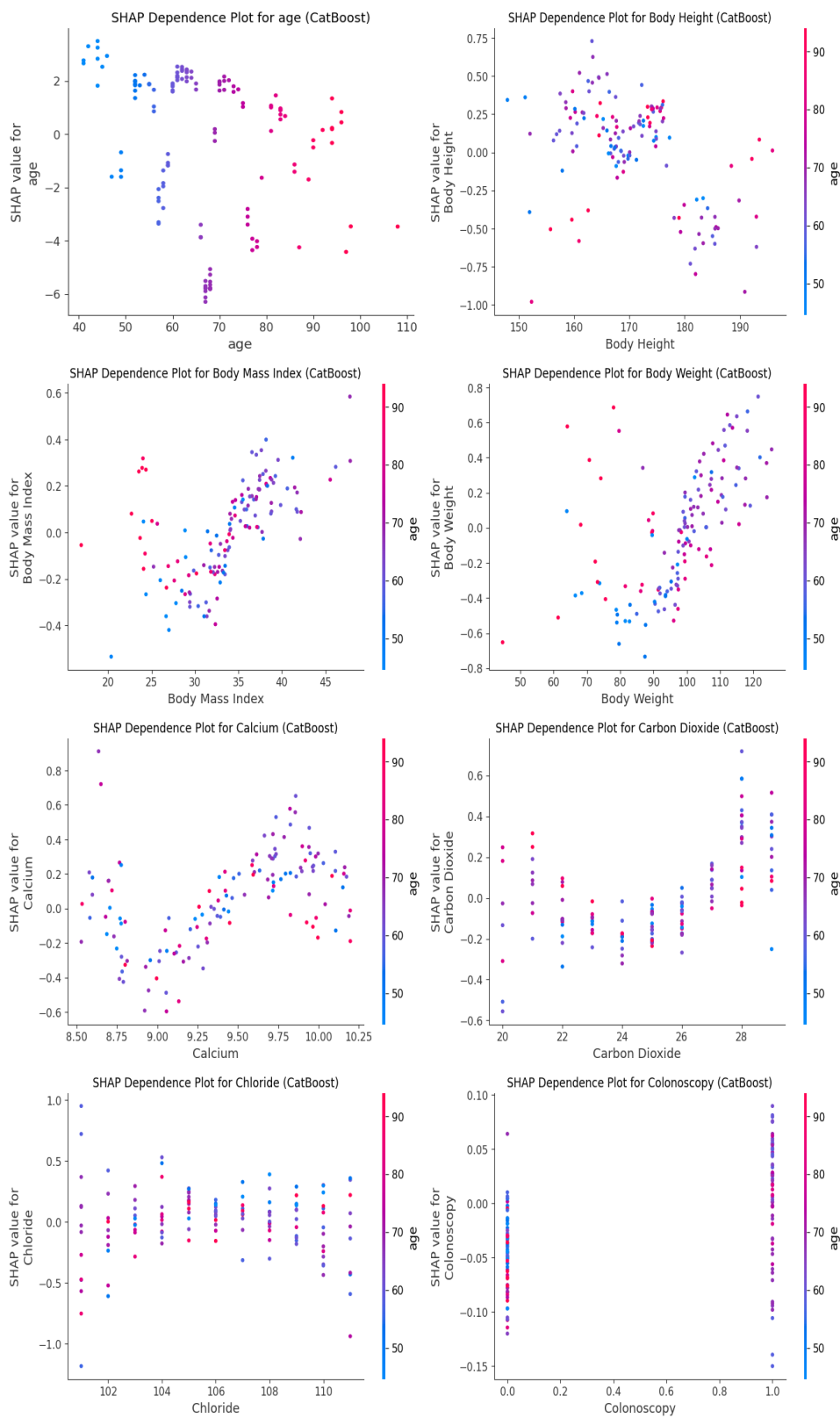
Και στον CatBoost η ηλικία έχει την μεγαλύτερη αλλά ταυτόχρονα και πιο αρνητική επίδραση στο μοντέλο. Το μοντέλο είχε θετική επίδραση για άτομα με μεσαίο ανάστημα και για άτομα με μεγάλο βάρος, για υψηλές τιμές ασβεστίου και διοξειδίου.



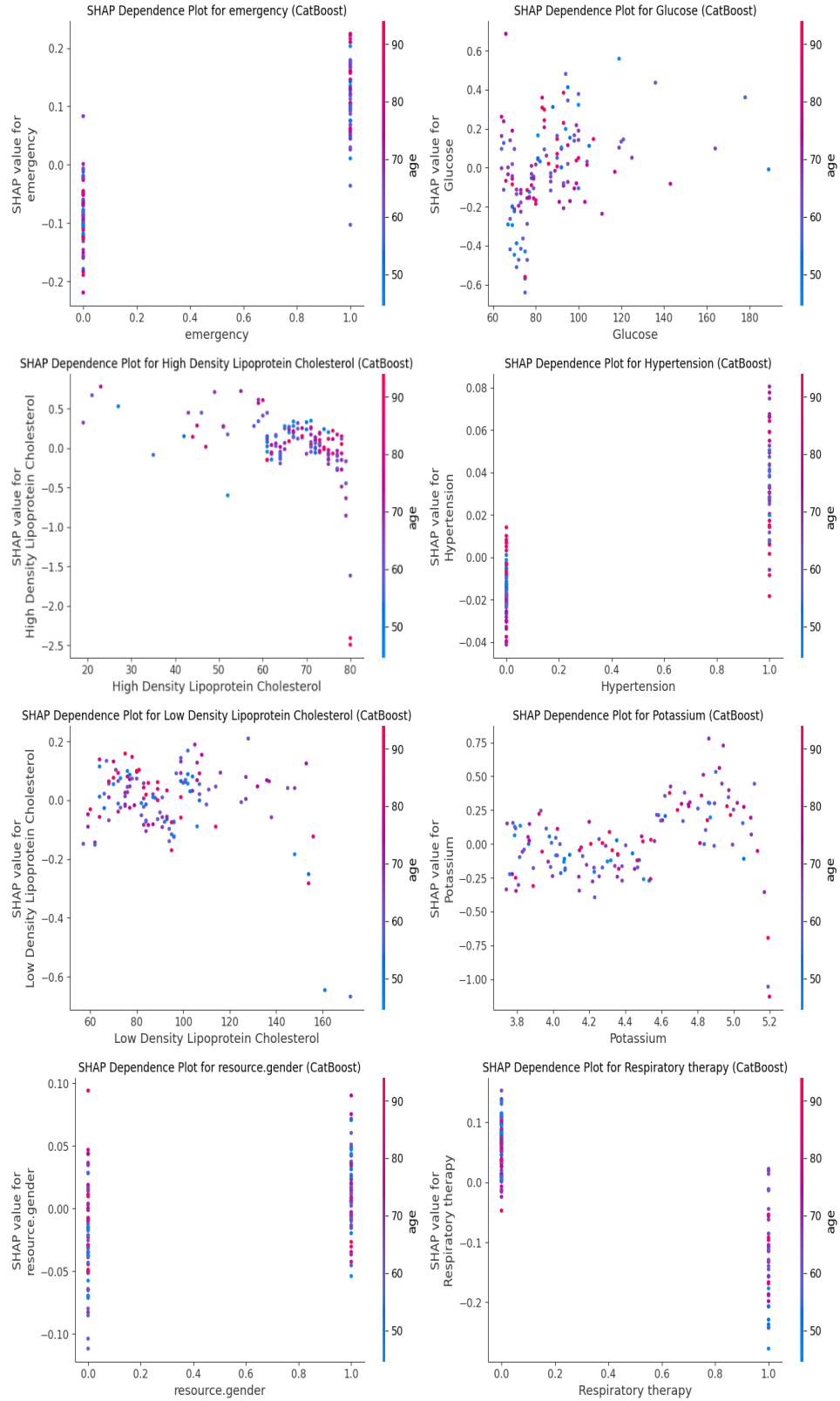
Γράφημα 45. SHAP BarPlot του CatBoost (feature selected: Td vax)



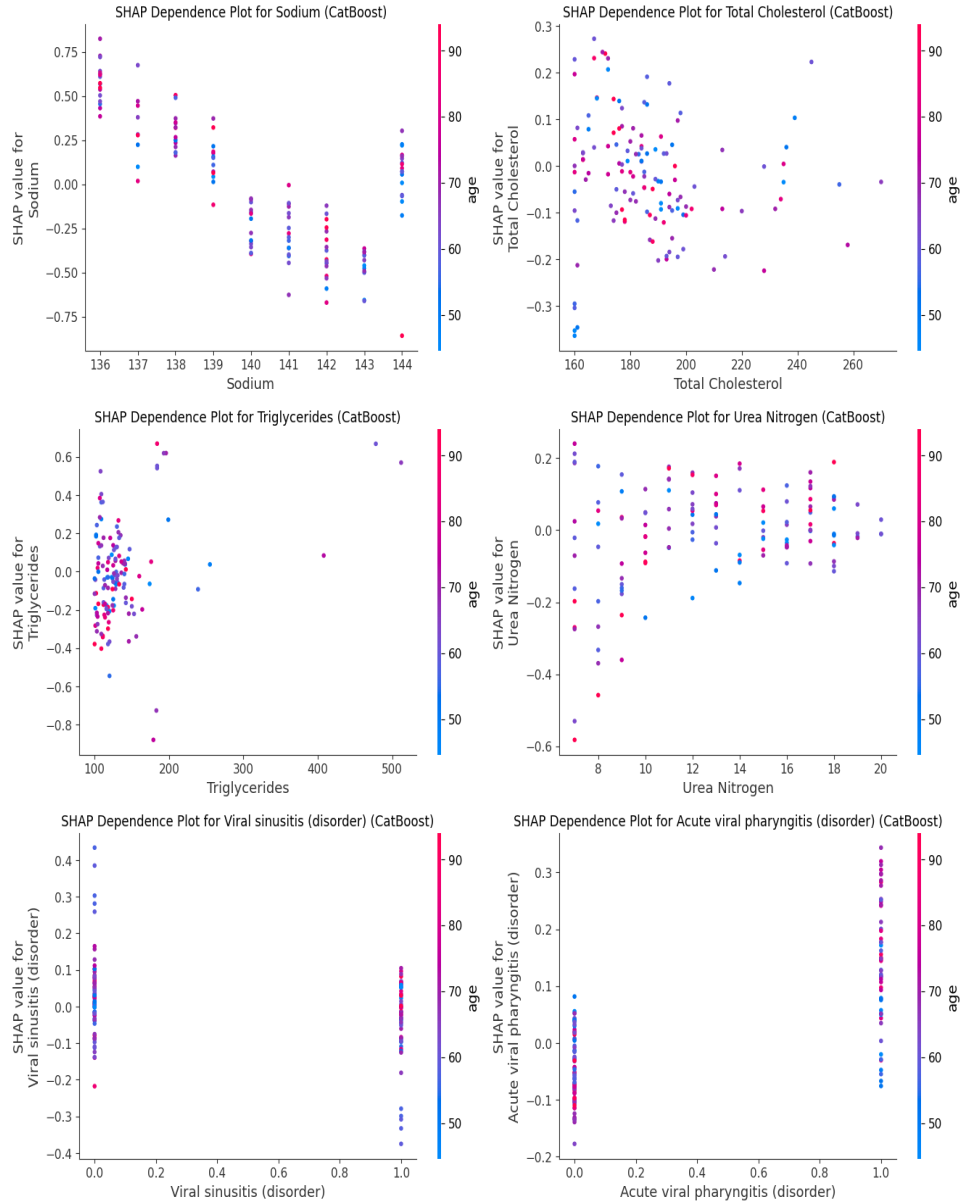
Γράφημα 46. SHAP BeeswarmPlot και WaterfallPlot του CatBoost (feature selected: Td vax)



Γράφημα 47. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax)

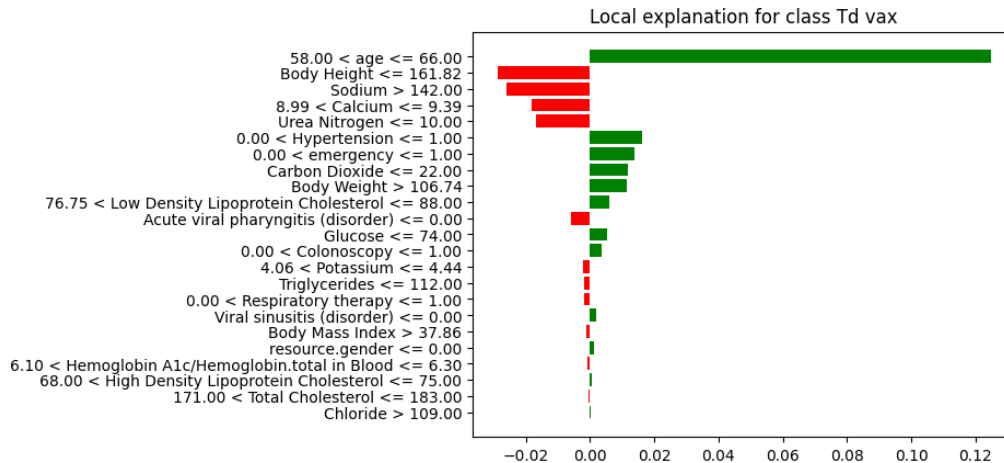


Γράφημα 48. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax)



Γράφημα 49. SHAP DependencePlots του XGBoost για διάφορα χαρακτηριστικά (feature selected: Td vax)

Θετική επίδραση στο μοντέλο είχαν επίσης τα άτομα με έκτακτη εισαγωγή στο νοσοκομείο, οι υψηλές τιμές «καλής» χοληστερόλης και οι σχετικά υψηλές τιμές καλίου. Τα άτομα με υψηλές τιμές νατρίου και αναπνευστική θεραπεία επηρέασαν το τελικό αποτέλεσμα αρνητικά.



Γράφημα 50. LIME plot για την θετική και αρνητική επίδραση κάθε χαρακτηριστικού (feature selected: Td vox)

4.3 Μοντέλο διάγνωσης Προδιαβήτη/Διαβήτη

Στους ΗΦΥ του dataset που χρησιμοποιούμε, υπήρχε ένας ικανοποιητικός αριθμός διαγνώσεων για άτομα που έχουν Προδιαβήτη και Διαβήτη. Μέσω των ιατρικών παρατηρήσεων που έχουν γίνει σε αυτά τα άτομα, δημιουργήσαμε ένα μοντέλο διάγνωσης Προδιαβήτη/Διαβήτη, το οποίο μάλιστα το ενσωματώσαμε σε μία web εφαρμογή (streamlit), όπου ο χρήστης εισάγει ορισμένα στοιχεία και η εφαρμογή του απαντάει αν έχει προδιαβήτη ή διαβήτη.

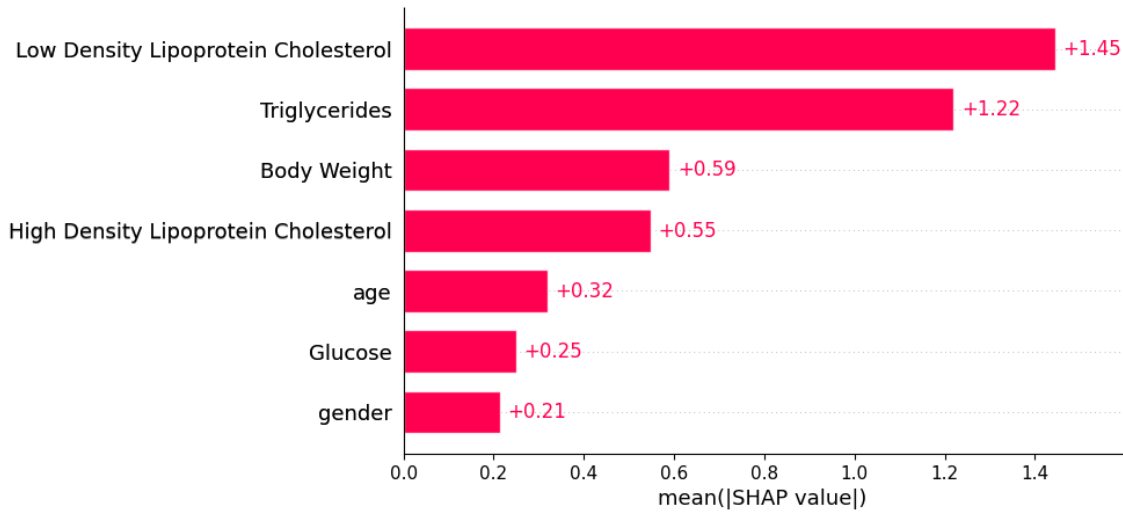
Με αυτόν τον τρόπο δείχνουμε ότι οι ΗΦΥ μπορούν να χρησιμοποιηθούν και στον τομέα των συστημάτων υποστήριξης αποφάσεων (DSS).

4.3.1 XGBoost

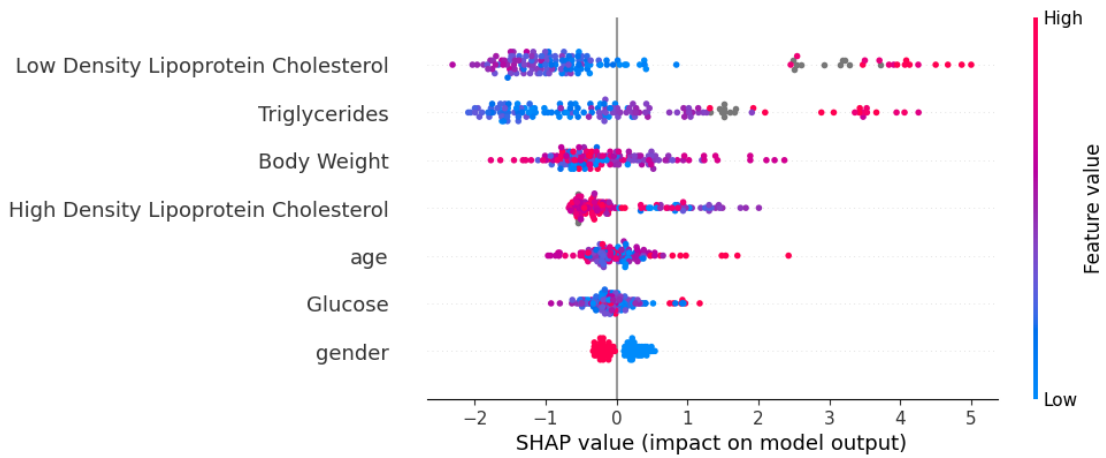
```
Accuracy: 0.9103448275862069
Precision: 0.8401162790697674
Recall: 0.7695852534562212
F1 Score: 0.7986326247195812
roc_auc score: 0.7695852534562212
```

Γράφημα 51. Αποτελέσματα του XGBoost για το μοντέλο διάγνωσης Προδιαβήτη/Διαβήτη

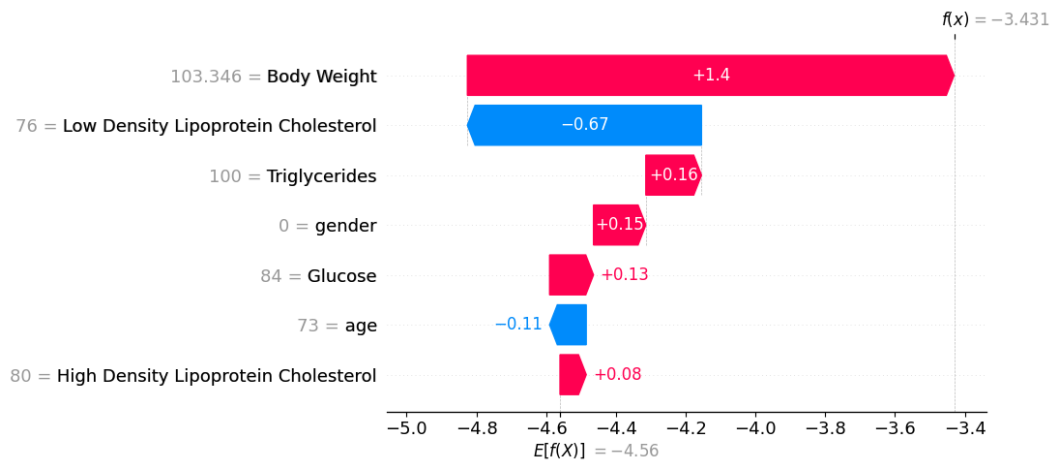
Στον XGBoost, περισσότερο θετικά (αλλά με μικρή διαφορά), επηρέασε το χαρακτηριστικό «Low Density Lipoprotein Cholesterol». Επίσης, άτομα μεγαλύτερης ηλικίας και με χαμηλά επίπεδα γλυκόζης επηρεάζουν αρνητικά το μοντέλο, όπως και τα ηλικιωμένα άτομα αρσενικού γένους.



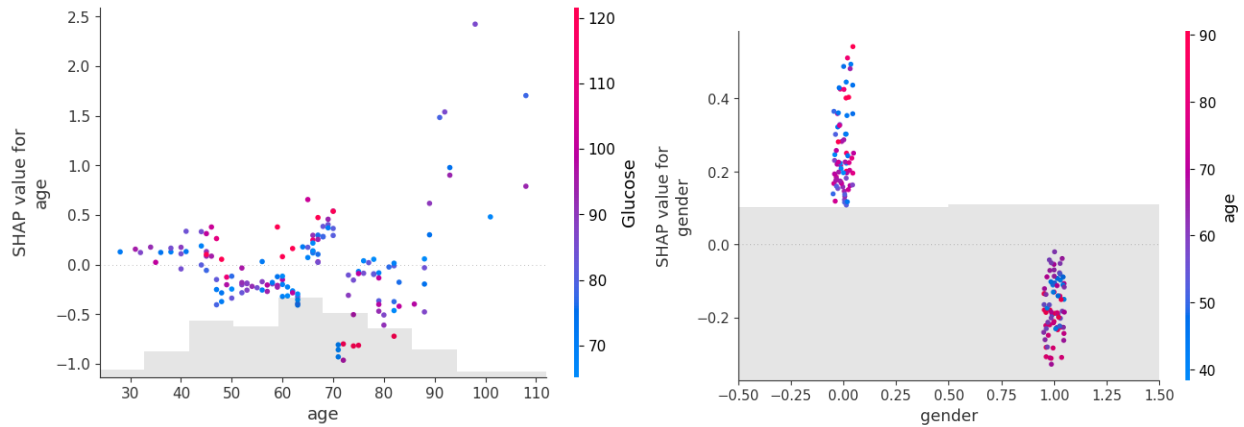
Γράφημα 52. SHAP BarPlot για τα χαρακτηριστικά του μοντέλου με XGBoost



Γράφημα 53. SHAP Beeswarm για τα χαρακτηριστικά του μοντέλου με XGBoost



Γράφημα 54. SHAP WaterfallPlot για τα χαρακτηριστικά του μοντέλου με XGBoost



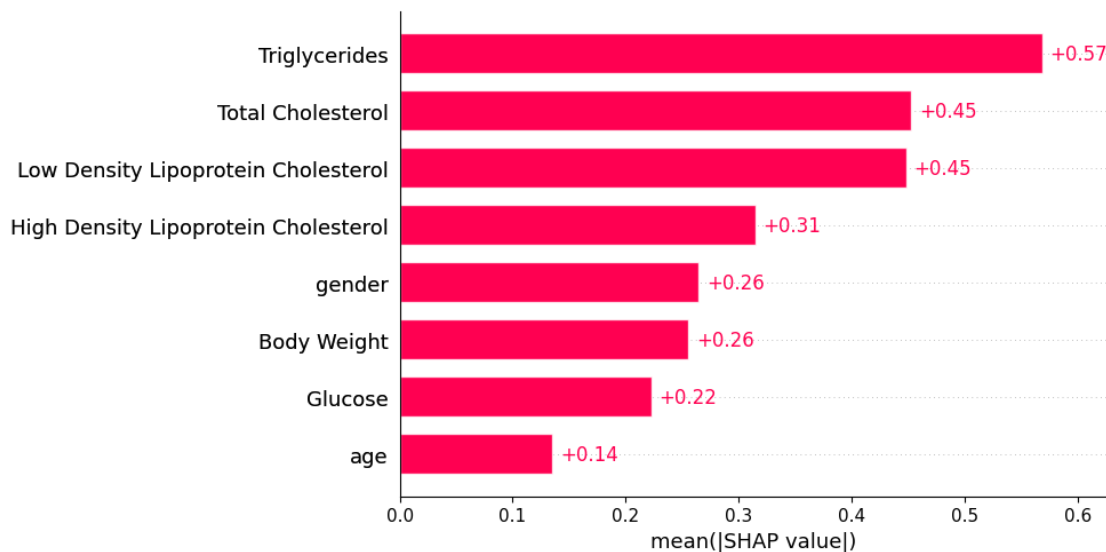
Γράφημα 55. SHAP ScatterPlots για την ηλικία με XGBoost

4.3.2 CatBoost

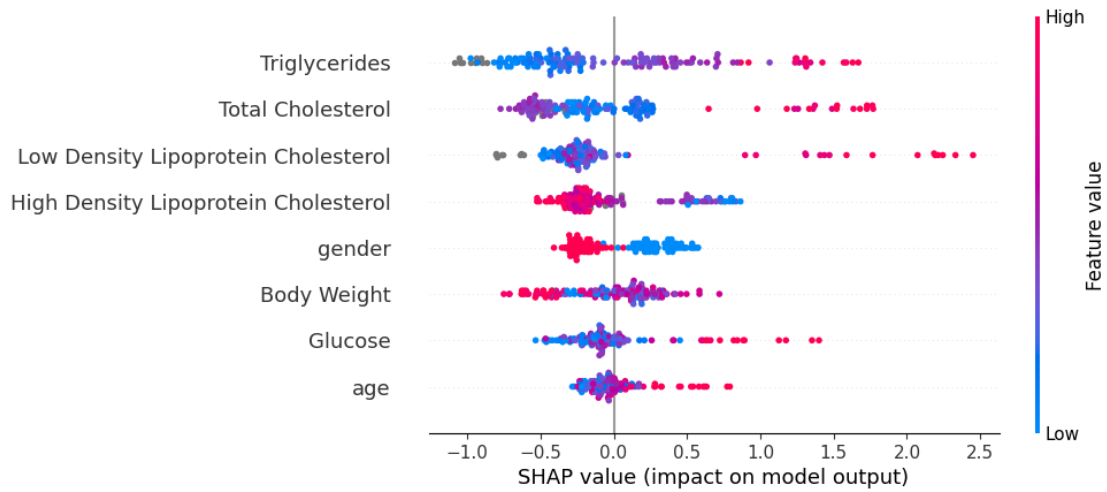
Accuracy: 0.896551724137931
Precision: 0.8049903100775193
Recall: 0.7417434715821813
F1 Score: 0.7676530285225938
roc_auc score: 0.7417434715821812

Γράφημα 56. Αποτελέσματα του CatBoost για το μοντέλο διάγνωσης Προδιαβήτη/Διαβήτη

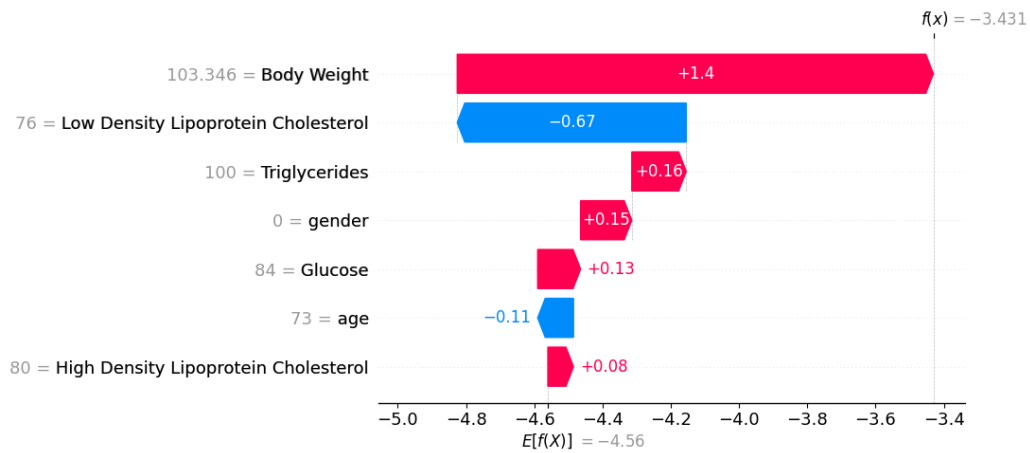
Τα ποσοστά επιτυχίας για τον CatBoost ήταν χαμηλότερα, ενώ δεν διέφεραν σημαντικά τα αποτελέσματα για το ποιο χαρακτηριστικό επηρεάζει περισσότερο το μοντέλο.



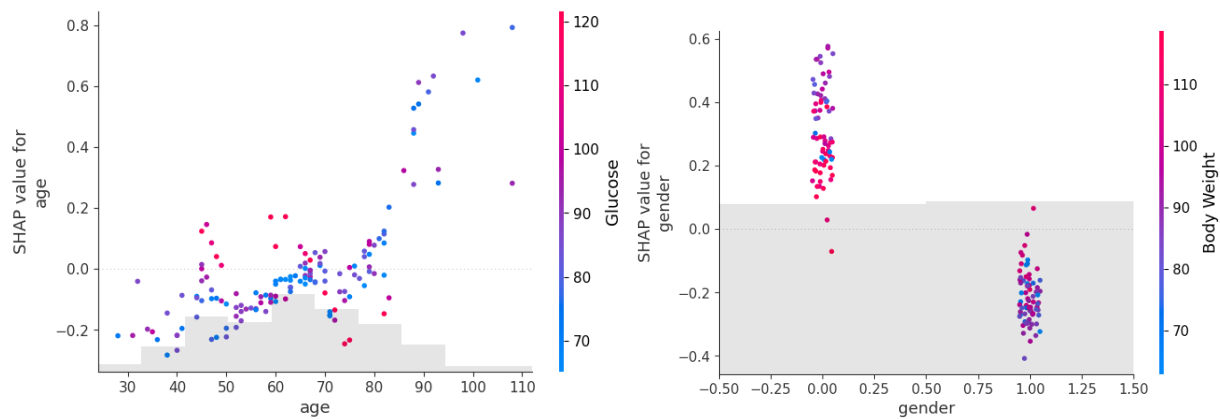
Γράφημα 57. SHAP BarPlot για τα χαρακτηριστικά του μοντέλου με CatBoost



Γράφημα 58. SHAP Beeswarm για τα χαρακτηριστικά του μοντέλου με CatBoost



Γράφημα 59. SHAP WaterfallPlot για τα χαρακτηριστικά του μοντέλου με CatBoost



Γράφημα 60. SHAP ScatterPlots για την ηλικία και το φύλο με CatBoost

4.3.3 Εφαρμογή διάγνωσης Προδιαβήτη/Διαβήτη

Ο χρήστης εισάγει στην εφαρμογή τα εξής στοιχεία: Ηλικία, φύλο, βάρος, Γλυκόζη, «καλή» χοληστερίνη, «κακή» χοληστερίνη και τριγλυκερίδια.

Στη συνέχεια, δίνει την εντολή στο μοντέλο να κάνει την πρόβλεψη και παίρνει απάντηση είτε για διάγνωση προδιαβήτη (σε πράσινο χρώμα), είτε για διάγνωση διαβήτη (κόκκινο χρώμα).

Diabetes/Prediabetes Diagnosis

Enter the following attributes:

Age	60	-	+
Gender	Male		
Body Weight (kg)	70,00	-	+
Glucose (mg/dl)	100,00	-	+
High Density Lipoprotein Cholesterol (mg/dl)	60,00	-	+
Low Density Lipoprotein Cholesterol (mg/dl)	150,00	-	+
Triglycerides (mg/dl)	90,00	-	+
<button>Predict</button>			
The person with the given details has Prediabetes.			

Γράφημα 61. Για τα συγκεκριμένα στοιχεία εισόδου η εφαρμογή προβλέπει ότι το άτομο έχει προδιαβήτη.

Diabetes/Prediabetes Diagnosis

Enter the following attributes:

Age	70	-	+
Gender	Female		
Body Weight (kg)	90,00	-	+
Glucose (mg/dl)	160,00	-	+
High Density Lipoprotein Cholesterol (mg/dl)	90,00	-	+
Low Density Lipoprotein Cholesterol (mg/dl)	150,00	-	+
Triglycerides (mg/dl)	220,00	-	+
<button>Predict</button>			
The person with the given details has Diabetes			

Γράφημα 62. Για τα συγκεκριμένα στοιχεία εισόδου η εφαρμογή προβλέπει ότι το άτομο έχει διαβήτη.

5. Συζήτηση

5.1 Σχόλια

Η χρήση των XGBoost και CatBoost σε ΗΦΥ δείχνει ιδιαίτερα υποσχόμενη για την ανάπτυξη μοντέλων πρόβλεψης σε διάφορους τομείς της υγείας. Αυτοί οι αλγόριθμοι έχουν τη δυνατότητα να διαχειριστούν δεδομένα από διαφορετικές πηγές και με διάφορες μορφές, συμπεριλαμβανομένων δεδομένων κατηγοριών, χρονικών σειρών και συνεχών δεδομένων, τα οποία συχνά περιέχονται σε ΗΦΥ. Επιπλέον, η ενσωμάτωση τεχνικών όπως η αυτόματη κατηγοριοποίηση στο CatBoost και η δυνατότητα επεξεργασίας ελλιπών δεδομένων στο XGBoost καθιστά αυτούς τους αλγόριθμους εξαιρετικά ευέλικτους και αποτελεσματικούς.

Ένα σημαντικό πλεονέκτημα του XGBoost είναι η ταχύτητα εκτέλεσης και η αποδοτικότητα στη μνήμη, κάτι που τον καθιστά κατάλληλο για μεγάλα σύνολα δεδομένων. Από την άλλη πλευρά, ο CatBoost είναι πιο κατάλληλο για δεδομένα με κατηγορίες, τα οποία είναι συνηθισμένα στους ΗΦΥ. Η αποτελεσματική διαχείριση κατηγοριών στον CatBoost μειώνει την ανάγκη για εκτεταμένη προεπεξεργασία των δεδομένων, κάτι που επιταχύνει τη διαδικασία ανάλυσης.

Ο CatBoost ήταν ελαφρώς πιο αποτελεσματικός από τον XGBoost στις κλάσεις που χρησιμοποιήσαμε. Παρακάτω, σημειώνονται οι διαφορές μεταξύ των αλγορίθμων ανά κλάση αλλά και οι διαφορές μεταξύ των ίδιων των κλάσεων.

Διαφορές αλγορίθμων ανά κλάση

Κλάση gender (φύλο)	
XGBoost	CatBoost
• Καλύτερος για υψηλότερες τιμές διοξειδίου του άνθρακα • Το βάρος έχει λιγότερο αρνητική επίδραση	• Καλύτερος για ηλικιωμένους • Καλύτερος για υψηλότερο ΔΜΣ • Καλύτερος για τιμές γλυκόζης • Καλύτερος για άτομα χωρίς υπέρταση • Καλύτερος για τιμές καλίου • Το ύψος είχε θετικότερη επίδραση

Κλάση colonoscopy (κολονοσκόπηση)	
XGBoost	CatBoost
• Καλύτερος για χαμηλό ΔΜΣ • Το βάρος έχει λιγότερο αρνητική επίδραση	• Καλύτερος για μέσες τιμές ασβεστίου • Καλύτερος για μέσες τιμές «κακής» χοληστερόλης • Καλύτερος για χαμηλές τιμές συνολικής χοληστερόλης • η ηλικία έχει θετικότερο αντίκτυπο

Κλάση Td vax (Εμβόλιο για τέτανο και διφθερίτιδα)	
XGBoost	CatBoost
• Καλύτερος για χαμηλά επίπεδα ασβεστίου • Καλύτερος για μέτριες τιμές βάρους	• Καλύτερος για μέσο ανάστημα • Καλύτερος για μεγάλο βάρος • Καλύτερος για υψηλό ασβέστιο

• Συνολικά πιο ισορροπημένος	• Καλύτερος για υψηλή «καλή» χοληστερόλη • Καλύτερος για χαμηλά επίπεδα κακής χοληστερόλης • Καλύτερος για άτομα χωρίς αναπνευστική θεραπεία
--	--

Διαφορές μεταξύ των κλάσεων

Κλάση colonoscopy (κολονοσκόπηση)	Κλάση gender (φύλο)
• το βάρος και το ύψος δεν έχουν τόσο μεγάλο αντίκτυπο • Υψηλό ανάστημα θετικό αντίκτυπο • Χαμηλό βάρος θετικό αντίκτυπο • Καλύτερο για υψηλές τιμές ασβεστίου • Άτομα χωρίς έκτακτη επίσκεψη στο νοσοκομείο επιδρούν θετικά • Καλύτερη επίδοση για άτομα που έχουν κάνει το Td vax • Καλύτερο για μέσες τιμές συνολικής χοληστερόλης	• άτομα χωρίς οξεία φαρυγγίτιδα επιδρούν θετικότερα • Υψηλό ανάστημα αρνητικό αντίκτυπο • Χαμηλό βάρος αρνητικό αντίκτυπο • Άτομα χωρίς έκτακτη επίσκεψη στο νοσοκομείο επιδρούν αρνητικά • Καλύτερη επίδραση σε τιμές «κακής» χοληστερόλης • Αρνητική επίδραση υψηλών τιμών «καλής» χοληστερόλης • Καλύτερο για χαμηλά τριγλυκερίδια

Κλάση Td vax (Εμβόλιο για τέτανο και διφθερίτιδα)	Κλάση gender (φύλο)
• Καλύτερο για άτομα μεγαλύτερου βάρους • Καλύτερη επίδοση για υψηλότερες τιμές διοξειδίου • Καλύτερο για υψηλότερες τιμές καλής χοληστερόλης • Καλύτερο για άτομα που έχουν υποβληθεί αναπνευστική θεραπεία • Καλύτερο για χαμηλότερες τιμές νατρίου • Καλύτερο για υψηλότερες τιμές ουρίας	• επηρεάζεται πολύ πιο αρνητικά από το βάρος • Καλύτερο για άτομα χωρίς οξεία φαρυγγίτιδα • Καλύτερο για άτομα χαμηλότερου ύψους • Καλύτερο για άτομα που έχουν υποβληθεί σε κολονοσκόπηση • Καλύτερο για άτομα χωρίς υπέρταση • Καλύτερο για άτομα με ιογενή ιγμορίτιδα

Κλάση Td vax (Εμβόλιο για τέτανο και διφθερίτιδα)	Κλάση colonoscopy (κολονοσκόπηση)
• Καλύτερο για άτομα με ύψος κάτω από 1.70 • Καλύτερο για άτομα με χαμηλό ασβέστιο • Καλύτερο για άτομα με χαμηλή κακή χοληστερόλη • Καλύτερο για άτομα χωρίς αναπνευστική θεραπεία	• Καλύτερο για ηλικίες 60-80 • Καλύτερο για άτομα με ύψος πάνω από 1.80 • Καλύτερο για άτομα με Δείκτη Μάζας Σώματος 35-40 • Καλύτερο για άτομα χωρίς έκτακτη επίσκεψη στο νοσοκομείο • Καλύτερο για άτομα χωρίς υπέρταση • Καλύτερο για ασθενείς γυναικείου φύλου

5.2 Σύγκριση με Άλλους Ερευνητές

Σε σύγκριση κυρίως με τα επιστημονικά άρθρα που αναλύσαμε στο 2^ο κεφάλαιο της εργασίας, συναντήσαμε σίγουρα αρκετές διαφορές. Οι άλλες ερευνητικές προσεγγίσεις χρησιμοποιούν κυρίως αλγόριθμους όπως ο Random Forest, τα νευρωνικά δίκτυα και οι αλγόριθμοι SVM (Support Vector Machines). Σε σχέση με τον Random Forest, ο XGBoost και ο CatBoost δείχνουν να προσφέρουν καλύτερη απόδοση, ιδίως όταν τα δεδομένα είναι ανισοκατανομημένα ή υπάρχουν πολλοί θόρυβοι στα δεδομένα, όπως συμβαίνει συνήθως με τα δεδομένα των ΗΦΥ.

Όσον αφορά τα νευρωνικά δίκτυα, οι XGBoost και CatBoost μπορούν να επιτύχουν παρόμοια ή και καλύτερα αποτελέσματα σε πολλά προβλήματα χωρίς την ανάγκη για περίπλοκες αρχιτεκτονικές δικτύων και εκτεταμένη εκπαίδευση. Τα νευρωνικά δίκτυα τείνουν να είναι εξαιρετικά αποδοτικά σε συγκεκριμένα προβλήματα που περιλαμβάνουν μη δομημένα δεδομένα, όπως εικόνες ή κείμενα, αλλά στους ΗΦΥ που περιλαμβάνουν κυρίως δομημένα δεδομένα, οι XGBoost και CatBoost μπορεί να είναι πιο επιτυχείς.

5.3 Παρουσίαση Αδυναμιών

Παρά τα πολλά πλεονεκτήματα των αλγορίθμων XGBoost και CatBoost, παρουσιάζουν ορισμένες αδυναμίες. Ένα από τα σημαντικότερα μειονεκτήματα του XGBoost είναι η τάση του να προσαρμόζεται υπερβολικά στα δεδομένα εκπαίδευσης, ειδικά όταν το σύνολο δεδομένων είναι μικρό ή όταν περιέχει πολλές εξαιρέσεις ή θορυβώδη δεδομένα. Αυτό μπορεί να οδηγήσει σε κακή γενίκευση στα δεδομένα δοκιμής, κάτι που υποδεικνύει την ανάγκη για προσεκτική ρύθμιση των υπερπαραμέτρων.

Ο CatBoost, αν και είναι εξαιρετικός για κατηγοριοποίηση, είναι πιο αργός σε σύγκριση με άλλους αλγόριθμους, ιδιαίτερα όταν το σύνολο δεδομένων είναι πολύ μεγάλο ή έχει πολλές κατηγορίες. Επίσης, παρόλο που είναι εξαιρετικά αποδοτικός σε δεδομένα κατηγοριών, μπορεί να υστερεί σε προβλήματα που βασίζονται σε αριθμητικά χαρακτηριστικά, όπου άλλοι αλγόριθμοι μπορούν να αποδώσουν καλύτερα.

Επιπρόσθετα, τρέξαμε τους αλγόριθμους σε ένα πολύ μικρό δείγμα των ΗΦΥ (μόλις το 2.14%), μιας και αν μεγαλώναμε το δείγμα η διαδικασία θα γινόνταν υπερβολικά χρονοβόρα και χωρίς να είναι βέβαιο ότι θα βελτιώσει τα αποτελέσματα.

5.4 Προβληματισμοί

Η εφαρμογή αλγορίθμων μηχανικής μάθησης σε ΗΦΥ εγείρει σίγουρα προβληματισμούς. Ένας από τους βασικότερους προβληματισμούς αφορά την ποιότητα των δεδομένων. Οι ΗΦΥ συχνά περιέχουν ελλιπή, εσφαλμένα ή ακατάλληλα δεδομένα, κάτι που μπορεί να έχει αρνητική επίδραση στην απόδοση και την ακρίβεια των μοντέλων. Παρά το γεγονός ότι αλγόριθμοι όπως ο XGBoost και ο CatBoost είναι σχεδιασμένοι ακόμη και για ελλιπή δεδομένα, εξακολουθεί να απαιτείται ιδιαίτερα προσεκτική προεπεξεργασία για να διασφαλιστεί η ακεραιότητα και η ποιότητα των δεδομένων που χρησιμοποιούνται. Αυτό ενδέχεται να σημαίνει σημαντική επένδυση χρόνου και πόρων για τον καθαρισμό των δεδομένων, κάτι που αποτελεί σημαντική πρόκληση.

Ένας άλλος προβληματισμός συνδέεται με την ερμηνευσιμότητα των μοντέλων μηχανικής μάθησης, κάτι που έχει ιδιαίτερη σημασία όταν μιλάμε για εφαρμογές στον τομέα της υγείας. Παρά το γεγονός ότι οι αλγόριθμοι XGBoost και CatBoost ενσωματώνουν εργαλεία όπως τα

SHAP values και το LIME, που παρέχουν εξηγήσεις για τις προβλέψεις των μοντέλων, η πλήρης κατανόηση των αποτελεσμάτων μπορεί να παραμένει ασαφής, ειδικά όταν τα δεδομένα είναι εξαιρετικά περίπλοκα ή υπάρχουν πολλά αλληλεπιδρώντα χαρακτηριστικά. Οι κλινικές εφαρμογές απαιτούν υψηλό επίπεδο ερμηνευσιμότητας, καθώς οι αποφάσεις που λαμβάνονται βάσει αυτών των μοντέλων μπορεί να επηρεάσουν την υγεία και τη ζωή των ασθενών. Αν τα αποτελέσματα των μοντέλων δεν είναι πλήρως κατανοητά ή δεν μπορούν να εξηγηθούν με ακρίβεια, αυτό μπορεί να περιορίσει την εμπιστοσύνη των επαγγελματιών της υγείας και να καθυστερήσει την υιοθέτηση τέτοιων τεχνολογιών.

Επίσης, πρέπει να σημειωθούν οι δυσκολίες στην ερμηνεία των αποτελεσμάτων παρά τη χρήση μεγάλου αριθμού διαγραμμάτων και την ανάλυση με εργαλεία όπως τα SHAP values και το LIME. Συχνά, ακόμα και με την οπτικοποίηση των δεδομένων και τις επεξηγήσεις, τα συμπεράσματα δεν είναι πάντα ξεκάθαρα. Αυτό είναι ιδιαίτερα προβληματικό σε περιπτώσεις που απαιτείται λήψη αποφάσεων βάσει των αποτελεσμάτων της μηχανικής μάθησης.

Επιπλέον, υπάρχει το ζήτημα της διαφάνειας στην κλινική πράξη. Οι επαγγελματίες υγείας χρειάζονται όχι μόνο ακριβείς προβλέψεις αλλά και σαφείς εξηγήσεις για το πώς προκύπτουν αυτές οι προβλέψεις. Η μηχανική μάθηση, και ειδικότερα οι αλγόριθμοι όπως ο XGBoost και ο CatBoost, μπορεί να προσφέρουν υψηλής ποιότητας προβλέψεις, αλλά η έλλειψη σαφήνειας στην ερμηνεία των προβλέψεων μπορεί να αποτελέσει εμπόδιο στην ευρεία υιοθέτησή τους. Το γεγονός ότι οι κλινικοί γιατροί δυσκολεύονται να κατανοήσουν τις αποφάσεις των μοντέλων μπορεί να προκαλέσει δισταγμό στην εμπιστοσύνη τους σε τέτοιες τεχνολογίες, κάτι που αναδεικνύει την ανάγκη για περαιτέρω ανάπτυξη εργαλείων ερμηνευσιμότητας.

5.5 Παρουσίαση Σπουδαιότητας και Καινοτομίας

Η παρούσα εργασία προσφέρει σημαντική συνεισφορά στη μελέτη της εφαρμογής της μηχανικής μάθησης σε ΗΦΥ μέσω της εκτενούς βιβλιογραφικής ανάλυσης, η οποία καλύπτει λεπτομερώς τις προηγούμενες μελέτες και πρακτικές στον τομέα αυτό. Αυτή η ανασκόπηση προσφέρει μία ολοκληρωμένη συγκριτική παρουσίαση των πιο πρόσφατων και αξιοσημείωτων ερευνητικών άρθρων, συμβάλλοντας στην κατανόηση του τρέχοντος ερευνητικού τοπίου και εντοπίζοντας τα κενά στη βιβλιογραφία. Η ανάλυση αυτή θέτει τα θεμέλια για τις περαιτέρω εξελίξεις, προβάλλοντας τα πλεονεκτήματα και τις προκλήσεις της χρήσης μηχανικής μάθησης στα συστήματα υγειονομικής περίθαλψης, ενώ ταυτόχρονα παρέχει κίνητρα για την περαιτέρω έρευνα.

Ένα από τα πιο καινοτόμα σημεία της εργασίας είναι η έμφαση στην οπτικοποίηση δεδομένων. Ο τεράστιος όγκος δεδομένων που προκύπτει από τους ΗΦΥ παρουσιάζεται μέσω ενός μεγάλου αριθμού διαγραμμάτων, προσφέροντας μια ολοκληρωμένη και σαφή εικόνα των τάσεων, των συσχετίσεων και των μοτίβων που ενδέχεται να αναδειχθούν από τα δεδομένα. Η χρήση πολλαπλών οπτικοποιήσεων παρέχει μία πιο διαισθητική κατανόηση των σύνθετων σχέσεων στα δεδομένα των ΗΦΥ. Είναι αξιοσημείωτο ότι, τουλάχιστον βάσει της βιβλιογραφίας που εξετάστηκε, η παρουσίαση τέτοιου αριθμού και λεπτομέρειας οπτικοποιήσεων δεν είναι ευρέως διαδεδομένη, γεγονός που αποτελεί πιθανώς μία καινοτομία.

Επιπρόσθετα, γίνεται χρήση των δύο από τους πιο σύγχρονους και προηγμένους αλγόριθμους, του XGBoost και του CatBoost, για τη σύγκριση και αξιολόγηση της απόδοσής τους σε διαφορετικά σύνολα δεδομένων. Οι δύο αυτοί αλγόριθμοι ανήκουν στους πιο αποτελεσματικούς για την επεξεργασία και ανάλυση μεγάλων και ετερογενών συνόλων δεδομένων, όπως αυτά που περιλαμβάνονται στους ΗΦΥ. Μέσα από τη σύγκριση των αλγορίθμων αυτών, αναδεικνύεται η δυνατότητα προσαρμογής τους σε διάφορα προβλήματα πρόβλεψης και

κατηγοριοποίησης στον τομέα της υγείας, προτείνοντας έτσι τα βέλτιστα μοντέλα που μπορούν να εφαρμοστούν σε αντίστοιχα περιβάλλοντα.

Ακόμα πιο σημαντικό είναι το γεγονός ότι η παρούσα εργασία προχωρά σε μία πιο βαθιά ανάλυση των αποτελεσμάτων μέσω της χρήσης των εργαλείων SHAP και LIME. Τα εργαλεία αυτά προσφέρουν εξηγήσεις για τις προβλέψεις των μοντέλων, επιτρέποντας έτσι να κατανοηθούν καλύτερα τις αιτίες που οδηγούν σε συγκεκριμένες προβλέψεις. Η εφαρμογή αυτών των εργαλείων δίνει τη δυνατότητα να εντοπιστούν κρίσιμα χαρακτηριστικά που επηρεάζουν τις αποφάσεις των μοντέλων, προσφέροντας έτσι μια πιο διαφανή και αξιόπιστη διαδικασία λήψης αποφάσεων στον τομέα της υγείας.

Τέλος, δεν μπορούμε να μην αναφερθούμε στην ανάπτυξη του μοντέλου διάγνωσης Προδιαβήτη/Διαβήτη. Η ανάπτυξη του μοντέλου απέδειξε ότι οι ΗΦΥ μπορούν να χρησιμοποιηθούν στο τομέα των συστημάτων υποστήριξης αποφάσεων για την διάγνωση ή και την πρόγνωση εμφάνισης διάφορων ασθενειών.

6. Συμπεράσματα

6.1 Σημαντικά στοιχεία που αναπτύχθηκαν, ανοικτές ερωτήσεις

Από την εργασία συμπεραίνεται ότι ο αλγόριθμος CatBoost προσφέρει ελαφρώς καλύτερα αποτελέσματα από τον XGBoost. Αυτό προκύπτει από τα αποτελέσματα των μετρικών accuracy, precision, recall, F1 Score και roc_auc score, αλλά και μέσω των επιδράσεων που δέχονται οι δύο αλγόριθμοι ανά χαρακτηριστικό.

Παρόλο που η εργασία καταλήγει σε συγκεκριμένα συμπεράσματα, παραμένουν ορισμένα ανοικτά ερωτήματα. Για παράδειγμα, ποια θα ήταν τα αποτελέσματα αν εφαρμοζόταν βελτιωμένη προεπεξεργασία των δεδομένων; Ποια η μακροπρόθεσμη αποτελεσματικότητα των αλγορίθμων σε πιο σύνθετα ή μεγαλύτερα σύνολα δεδομένων; Επίσης, παρόλο που οι αλγόριθμοι αποδίδουν καλά στις υπάρχουσες μετρήσεις, δεν είναι σαφές πόσο αποτελεσματικοί θα είναι σε άλλες εφαρμογές στον τομέα της υγείας ή σε διαφορετικά περιβάλλοντα. Τέλος, ενώ τα εργαλεία SHAP και LIME παρέχουν ερμηνείες για τις αποφάσεις των μοντέλων, παραμένει το ζήτημα της πλήρους ερμηνευσιμότητας και του αν αυτά τα εργαλεία είναι επαρκή για να καλύψουν τις ανάγκες της κλινικής πρακτικής.

6.2 Μελλοντικές προοπτικές

Για την πλήρη αξιοποίηση του δυναμικού της σχέσης μεταξύ των ΗΦΥ και της επεξηγούμενης μηχανικής μάθησης, μπορούν να εφαρμοστούν διάφορες στρατηγικές:

- Βελτίωση της ποιότητας των δεδομένων: Θα πρέπει να καταβληθούν προσπάθειες για τη βελτίωση της ποιότητας και της τυποποίησης των δεδομένων που καταγράφονται στους ΗΦΥ. Αυτό περιλαμβάνει την εκπαίδευση των παρόχων υγειονομικής περίθαλψης στις βέλτιστες πρακτικές για την εισαγωγή δεδομένων και την εφαρμογή αυτοματοποιημένων τεχνικών επικύρωσης δεδομένων.

- Ανάπτυξη ανθεκτικών τεχνικών XAI: Η συνεχής έρευνα σε μεθοδολογίες XAI είναι απαραίτητη για τη βελτίωση της ερμηνευσιμότητας των μοντέλων μηχανικής μάθησης. Αυτό περιλαμβάνει τη διερεύνηση νέων προσεγγίσεων για την ερμηνευσιμότητα που είναι προσαρμοσμένες στις μοναδικές προκλήσεις των δεδομένων υγειονομικής περίθαλψης.

- Προώθηση της συνεργασίας: Η συνεργασία μεταξύ επιστημόνων δεδομένων, κλινικών ιατρών και διαχειριστών υγειονομικής περίθαλψης είναι ζωτικής σημασίας για την ανάπτυξη εργαλείων ΤΝ που ανταποκρίνονται στις ανάγκες των τελικών χρηστών. Η εμπλοκή των κλινικών ιατρών στο σχεδιασμό και την αξιολόγηση των συστημάτων ΧΑΙ μπορεί να διασφαλίσει ότι τα εργαλεία αυτά είναι πρακτικά και φιλικά προς το χρήστη.

- Ανάλυση περισσότερων δεδομένων: Στη συγκεκριμένη εργασία χρησιμοποιείται το 2.14% των δεδομένων του dataset. Θα μπορούσε να χρησιμοποιηθεί μεγαλύτερο δείγμα και καλύτερα επεξεργασμένο, για ενδεχομένως καλύτερα αποτελέσματα.

Εν τέλει, η ανάγκη για συνεχή βελτίωση των μοντέλων και των εργαλείων που χρησιμοποιούνται στην ανάλυση των δεδομένων των ΗΦΥ. Η κλινική πράξη απαιτεί απόλυτη ακρίβεια και κατανόηση των προβλέψεων, και ενώ τα εργαλεία που χρησιμοποιήθηκαν προσφέρουν σημαντική βοήθεια, παραμένει η πρόκληση της ανάπτυξης μοντέλων που μπορούν να ερμηνεύουν επαρκώς και να προσφέρουν αξιόπιστες προβλέψεις.

7. Βιβλιογραφία

[1] Beasley JW, Holden RJ, Sullivan F. Electronic health records: research into design and implementation. *Br J Gen Pract*. 2011 Oct;61(591):604-5. doi: 10.3399/bjgp11X601244. PMID: 22152827; PMCID: PMC3177114.

[2] Uslu A, Stausberg J, Value of the Electronic Medical Record for Hospital Care: Update From the Literature, *J Med Internet Res* 2021;23(12):e26323, URL: <https://www.jmir.org/2021/12/e26323>, DOI: 10.2196/26323

[3] Poissant L, Pereira J, Tamblyn R, Kawasumi Y. The impact of electronic health records on time efficiency of physicians and nurses: a systematic review. *J Am Med Inform Assoc*. 2005 Sep-Oct;12(5):505-16. doi: 10.1197/jamia.M1700. Epub 2005 May 19. PMID: 15905487; PMCID: PMC1205599.

[4] Upadhyay S, Hu HF. A Qualitative Analysis of the Impact of Electronic Health Records (EHR) on Healthcare Quality and Safety: Clinicians' Lived Experiences. *Health Serv Insights*. 2022 Mar 3;15:11786329211070722. doi: 10.1177/11786329211070722. PMID: 35273449; PMCID: PMC8902175.

[5] A. Karim Jabali, Abu Waris, Diwan Israr Khan, Shakil Ahmed, Rana Jamal Hourani, Electronic health records: Three decades of bibliometric research productivity analysis and some insights, *Informatics in Medicine Unlocked*, Volume 29, 2022, 100872, ISSN 2352-9148, <https://doi.org/10.1016/j.imu.2022.100872>.

[6] Payrovnaziri SN, Chen Z, Rengifo-Moreno P, Miller T, Bian J, Chen JH, Liu X, He Z. Explainable artificial intelligence models using real-world electronic health record data: a systematic scoping review. *J Am Med Inform Assoc*. 2020 Jul 1;27(7):1173-1185. doi: 10.1093/jamia/ocaa053. PMID: 32417928; PMCID: PMC7647281.

[7] Hui Wen Loh, Chui Ping Ooi, Silvia Seoni, Prabal Datta Barua, Filippo Molinari, U Rajendra Acharya, Application of explainable artificial intelligence for healthcare: A systematic review of the last decade (2011–2022), *Computer Methods and Programs in Biomedicine*, Volume 226, 2022, 107161, ISSN 0169-2607, <https://doi.org/10.1016/j.cmpb.2022.107161>.

[8] Shahab S Band, Atefeh Yarahmadi, Chung-Chian Hsu, Meghdad Biyari, Mehdi Sookhak, Rasoul Ameri, Iman Dehzangi, Anthony Theodore Chronopoulos, Huey-Wen Liang, Application of explainable artificial intelligence in medical health: A systematic review of interpretability methods, *Informatics in*

- [9] Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE (pp. 80–89).
- [10] Cheng, Yu, et al. "Risk prediction with electronic health records: A deep learning approach." Proceedings of the 2016 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2016.
- [11] Boxwala, Aziz A., et al. "Using statistical and machine learning to help institutions detect suspicious access to electronic health records." *Journal of the American Medical Informatics Association* 18.4 (2011): 498-505.
- [12] Barak-Corren, Yuval, et al. "Predicting suicidal behavior from longitudinal electronic health records." *American journal of psychiatry* 174.2 (2017): 154-162.
- [13] Deo, Rahul C., and Brahmajee K. Nallamothu. "Learning about machine learning: the promise and pitfalls of big data and the electronic health record." *Circulation: Cardiovascular Quality and Outcomes* 9.6 (2016): 618-620.
- [14] Rahimian, Fatemeh, et al. "Predicting the risk of emergency admission with machine learning: Development and validation using linked electronic health records." *PLoS medicine* 15.11 (2018): e1002695.
- [15] Machine learning models in electronic health records can outperform conventional survival models for predicting patient mortality in coronary artery disease, Andrew J. Steele^{1*}, Spiros C. Denaxas², Anoop D. Shah^{2,3}, Harry Hemingway², Nicholas M. Luscombe
- [16] Prediction of Acute Kidney Injury With a Machine Learning Algorithm Using Electronic Health Record Data Hamid Mohamadlou, Anna Lynn-Palevsky, Christopher Barton, Uli Chettipally, Lisa Shieh, Jacob Calvert, Nicholas R. Saber, and Ritankar Das
- [17] Xiao C, Choi E, Sun J. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *J Am Med Inform Assoc*. 2018 Oct 1;25(10):1419-1428. doi: 10.1093/jamia/ocy068. PMID: 29893864; PMCID: PMC6188527.
- [18] Mehta N, Devarakonda MV. Machine learning, natural language programming, and electronic health records: The next step in the artificial intelligence journey? *J Allergy Clin Immunol*. 2018 Jun;141(6):2019-2021.e1. doi: 10.1016/j.jaci.2018.02.025. Epub 2018 Mar 5. PMID: 29518424.
- [19] Garriga R, Mas J, Abraha S, Nolan J, Harrison O, Tadros G, Matic A. Machine learning model to predict mental health crises from electronic health records. *Nat Med*. 2022 Jun;28(6):1240-1248. doi: 10.1038/s41591-022-01811-5. Epub 2022 May 16. PMID: 35577964; PMCID: PMC9205775.
- [20] Ng K, Steinhubl SR, deFilippi C, Dey S, Stewart WF. Early Detection of Heart Failure Using Electronic Health Records: Practical Implications for Time Before Diagnosis, Data Diversity, Data Quantity, and Data Density. *Circ Cardiovasc Qual Outcomes*. 2016 Nov;9(6):649-658. doi: 10.1161/CIRCOUTCOMES.116.002797. Epub 2016 Nov 8. PMID: 28263940; PMCID: PMC5341145.
- [21] Payrovnaziri SN, Chen Z, Rengifo-Moreno P, Miller T, Bian J, Chen JH, Liu X, He Z. Explainable artificial intelligence models using real-world electronic health record data: a systematic scoping review. *J Am Med Inform Assoc*. 2020 Jul 1;27(7):1173-1185. doi: 10.1093/jamia/ocaa053. PMID: 32417928; PMCID: PMC7647281.

- [22] Honeyford K, Expert P, Mendelsohn EE, Post B, Faisal AA, Glampson B, Mayer EK, Costelloe CE. Challenges and recommendations for high quality research using electronic health records. *Front Digit Health*. 2022 Aug 19;4:940330. doi: 10.3389/fdgth.2022.940330. PMID: 36060540; PMCID: PMC9437583.
- [23] Frasca, Maria, et al. "Explainable and interpretable artificial intelligence in medicine: a systematic bibliometric review." *Discover Artificial Intelligence* 4.1 (2024): 15.
- [24] Loh, Hui Wen, et al. "Application of explainable artificial intelligence for healthcare: A systematic review of the last decade (2011–2022)." *Computer Methods and Programs in Biomedicine* 226 (2022): 107161.
- [25] Hulsén, Tim. "Explainable artificial intelligence (XAI): concepts and challenges in healthcare." *AI* 4.3 (2023): 652-666.
- [26] Dhruv, Akshit J. et al. "Python: The Most Advanced Programming Language for Computer Science Applications." *Proceedings of the International Conference on Culture Heritage, Education, Sustainable Tourism, and Innovation Technologies* (2020): n. pag.
- [27] Han S, Kwak IY. Mastering data visualization with Python: practical tips for researchers. *J Minim Invasive Surg*. 2023 Dec 15;26(4):167-175. doi: 10.7602/jmis.2023.26.4.167. PMID: 38098348; PMCID: PMC10728683.
- [28] Arnav Oberoi, Rahul Chauhan (2019); Visualizing data using Matplotlib and Seaborn libraries in Python for data science; *International Journal of Scientific and Research Publications (IJSRP)* 9(3) (ISSN: 2250-3153), DOI: <http://dx.doi.org/10.29322/IJSRP.9.03.2019.p8733>
- [29] PEDREGOSA, Fabian, et al. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 2011, 12: 2825-2830.
- [30] Krishna Chaitanya (2021). Synthea Dataset Jsons - EHR. from <https://www.kaggle.com/datasets/krsna540/synthea-dataset-jsons-ehr>
- [31] McKinney, Wes, and P. D. Team. "Pandas-Powerful python data analysis toolkit." *Pandas—Powerful Python Data Analysis Toolkit* 1625 (2015).
- [32] Hagedorn, Stefan, Steffen Kläbe, and Kai-Uwe Sattler. "Putting pandas in a box." *Conference on Innovative Data Systems Research (CIDR)*; (Online). 2021.
- [33] Sial, Ali Hassan, Syed Yahya Shah Rashdi, and Abdul Hafeez Khan. "Comparative analysis of data visualization libraries Matplotlib and Seaborn in Python." *International Journal* 10.1 (2021): 45.
- [34] Waskom, Michael L. "Seaborn: statistical data visualization." *Journal of Open Source Software* 6.60 (2021): 3021.
- [35] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016.
- [36] Aslam, Nida, et al. "Interpretable machine learning models for malicious domains detection using explainable artificial intelligence (XAI)." *Sustainability* 14.12 (2022): 7375.
- [37] Mangalathu, Sujith, Seong-Hoon Hwang, and Jong-Su Jeon. "Failure mode and effects analysis of RC members based on machine-learning-based SHapley Additive exPlanations (SHAP) approach." *Engineering Structures* 219 (2020): 110927.
- [38] Mishra, Saumitra, Bob L. Sturm, and Simon Dixon. "Local interpretable model-agnostic explanations for music content analysis." *ISMIR*. Vol. 53. 2017.

Παράρτημα Α – Ο κώδικας

```
import numpy as np
import pandas as pd
import os
import matplotlib
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from tqdm import tqdm
from datetime import datetime as dt

# Suppress warnings and configure pandas to display up to 50 characters in each column
warnings.filterwarnings('ignore')
pd.set_option('display.max_colwidth', 50)

# Exploring file paths and reading the data
file_path_list = [] # Initialize an empty list to store file paths
for dirname, _, filenames in os.walk('fhir'): # Walk through the directory 'fhir'
    for filename in filenames: # For each file in the directory
        file_path_list.append((dirname, filename)) # Add the folder path and file name to the list

# Create a DataFrame from the file path list
metadata_df = pd.DataFrame(file_path_list, columns=["folder", "file"])
print(f"Files: {metadata_df.shape[0]}") # Print the number of files found
metadata_df.head() # Display the first few rows of the DataFrame

# Adding 'group' and 'subgroup' columns by splitting the folder path
metadata_df["group"] = metadata_df["folder"].apply(lambda x: x.split("\\")[-2])
metadata_df["subgroup"] = metadata_df["folder"].apply(lambda x: x.split("\\")[-1])
metadata_df = metadata_df[["folder", "group", "subgroup", "file"]] # Reorder the columns
metadata_df.head() # Display the first few rows of the updated DataFrame

# Print the number of unique folders, groups, subgroups, and files
print(f"Folders: {metadata_df.folder.nunique()}")
print(f"Groups: {metadata_df.group.nunique()}")
print(f"Subgroups: {metadata_df.subgroup.nunique()}")
print(f"Files: {metadata_df.file.nunique()}")

# Read a sample JSON file to understand the structure of the data
sample_df = pd.read_json('fhir/00/00b/00b1aa90-e000-479e-adbd-83f76cd6dc56.json')
print(sample_df.head()) # Print the first few rows of the sample data

# Initialize empty DataFrames for different FHIR resource types
patient_df = pd.DataFrame()
careplan_df = pd.DataFrame()
condition_df = pd.DataFrame()
diagnostic_report_df = pd.DataFrame()
encounter_df = pd.DataFrame()
immunization_df = pd.DataFrame()
observation_df = pd.DataFrame()
procedure_df = pd.DataFrame()

# Function to process one file and populate the respective DataFrames
def process_one_file(sample_df,
                    patient_df,
                    careplan_df,
                    condition_df,
                    diagnostic_report_df,
                    encounter_df,
                    immunization_df,
                    observation_df,
                    procedure_df):

    dataframe_list = [patient_df, careplan_df, condition_df, diagnostic_report_df,
                     encounter_df, immunization_df, observation_df, procedure_df] # List of DataFrames

    # Iterate through each row in the sample DataFrame
    for index, row in sample_df.iterrows():
        resourcetypes=set() # Set to store resource types
        tempdf=pd.json_normalize(row.entry) # Normalize the JSON structure into a flat DataFrame
        resourcetype.add([str(x) for x in tempdf['resource.resourceType']][0]) # Add the resource type to the set

        # Concatenate the data to the appropriate DataFrame based on the resource type
        if str(tempdf['resource.resourceType'][0])=="Patient":
            frames = [patient_df, tempdf]
            patient_df = pd.concat(frames)

        elif str(tempdf['resource.resourceType'][0])=="CarePlan":
            frames = [careplan_df, tempdf]
            careplan_df = pd.concat(frames)

        elif str(tempdf['resource.resourceType'][0])=="Condition":
            frames = [condition_df, tempdf]
            condition_df = pd.concat(frames)

        elif str(tempdf['resource.resourceType'][0])=="DiagnosticReport":
            frames = [diagnostic_report_df, tempdf]
            diagnostic_report_df = pd.concat(frames)

        elif str(tempdf['resource.resourceType'][0])=="Encounter":
            frames = [encounter_df, tempdf]
            encounter_df = pd.concat(frames)

        elif str(tempdf['resource.resourceType'][0])=="Immunization":
            frames = [immunization_df, tempdf]
            immunization_df = pd.concat(frames)

        elif str(tempdf['resource.resourceType'][0])=="Observation":
            frames = [observation_df, tempdf]
            observation_df = pd.concat(frames)

        elif str(tempdf['resource.resourceType'][0])=="Procedure":
            frames = [procedure_df, tempdf]
            procedure_df = pd.concat(frames)
```

```

# Return all updated DataFrames
return patient_df,\
        careplan_df,\
        condition_df,\
        diagnostic_report_df,\
        encounter_df,\
        immunization_df,\
        observation_df,\
        procedure_df

# Function to clean and rename columns in all DataFrames
def clean_and_rename(patient_df,
                    careplan_df,
                    condition_df,
                    diagnostic_report_df,
                    encounter_df,
                    immunization_df,
                    observation_df,
                    procedure_df):
    # Replace periods in column names with underscores and remove 'resource_' prefix
    for df in [patient_df, careplan_df, condition_df, diagnostic_report_df,
              encounter_df, immunization_df, observation_df, procedure_df]:
        df.columns = df.columns.str.replace(".", "_")
        df.columns = df.columns.str.replace("resource_", "")
    # Clean up 'fullUrl' fields in patient, observation, and encounter DataFrames
    for df in [patient_df, observation_df, encounter_df]:
        df['fullUrl'] = df['fullUrl'].str.replace('urn:uuid:', '')
    # Clean up 'resource.subject.reference' and 'context.reference' fields in other DataFrames
    for df in [careplan_df, condition_df, diagnostic_report_df]:
        df['resource.subject.reference'] = df['resource.subject.reference'].str.replace('urn:uuid:', '')
        df['context.reference'] = df['context.reference'].str.replace('urn:uuid:', '')
    # Clean up 'resource.patient.reference' fields in encounter and immunization DataFrames
    for df in [encounter_df, immunization_df]:
        df['resource.patient.reference'] = df['resource.patient.reference'].str.replace('urn:uuid:', '')
    # Clean up 'encounter.reference' fields in immunization, observation, and procedure DataFrames
    for df in [immunization_df]:
        df['encounter.reference'] = df['encounter.reference'].str.replace('urn:uuid:', '')

    for df in [observation_df, procedure_df]:
        df['resource.subject.reference'] = df['resource.subject.reference'].str.replace('urn:uuid:', '')
        df['encounter.reference'] = df['encounter.reference'].str.replace('urn:uuid:', '')

    # Return all cleaned and renamed DataFrames
    return patient_df,\
            careplan_df,\
            condition_df,\
            diagnostic_report_df,\
            encounter_df,\
            immunization_df,\
            observation_df,\
            procedure_df

```

```

# Process the sample file to populate the respective DataFrames
process_one_file(sample_df, patient_df,
                 careplan_df,
                 condition_df,
                 diagnostic_report_df,
                 encounter_df,
                 immunization_df,
                 observation_df,
                 procedure_df)

# Import and handle a group of files
# Select a subset of groups based on their frequency in the metadata
sel_index = list(metadata_df.group.value_counts()[0:5].index)
print(sel_index) # Print the selected group indices
group_df = metadata_df.loc[metadata_df.group.isin(sel_index)] # Filter metadata for selected groups
group_df.shape[0], group_df.shape[0] / metadata_df.shape[0] # Print the number and percentage of selected files

# Reinitialize empty DataFrames for different FHIR resource types
patient_df = pd.DataFrame()
careplan_df = pd.DataFrame()
condition_df = pd.DataFrame()
diagnostic_report_df = pd.DataFrame()
encounter_df = pd.DataFrame()
immunization_df = pd.DataFrame()
observation_df = pd.DataFrame()
procedure_df = pd.DataFrame()

# Iterate over the selected group files and process each file
for index, row in tqdm(group_df.iterrows()):
    folder = row["folder"]
    file = row["file"]
    sample_df = pd.read_json(os.path.join(folder, file)) # Read the JSON file
    # Process the file and update the respective DataFrames
    patient_df, \
    careplan_df, \
    condition_df, \
    diagnostic_report_df, \
    encounter_df, \
    immunization_df, \
    observation_df, \
    procedure_df = \
    process_one_file(sample_df, patient_df,
                    careplan_df,
                    condition_df,
                    diagnostic_report_df,
                    encounter_df,
                    immunization_df,
                    observation_df,
                    procedure_df)

```

```

# Import and handle a group of files
# 2.14% of the data
# Select the top 5 most common groups from metadata_df
sel_index = list(metadata_df.group.value_counts()[0:5].index)
print(sel_index)
# Filter the metadata DataFrame to include only the selected groups
group_df = metadata_df.loc[metadata_df.group.isin(sel_index)]
# Display the number of rows in the filtered DataFrame and its percentage of the total
group_df.shape[0], group_df.shape[0] / metadata_df.shape[0]

# Initialize empty DataFrames for each resource type
patient_df = pd.DataFrame()
careplan_df = pd.DataFrame()
condition_df = pd.DataFrame()
diagnostic_report_df = pd.DataFrame()
encounter_df = pd.DataFrame()
immunization_df = pd.DataFrame()
observation_df = pd.DataFrame()
procedure_df = pd.DataFrame()

# Process each file in the filtered group_df
for index, row in tqdm(group_df.iterrows()):
    folder = row["folder"]
    file = row["file"]
    # Read the JSON file into a DataFrame
    sample_df = pd.read_json(os.path.join(folder, file))
    # Process the file and update the corresponding DataFrames
    patient_df,\
    careplan_df,\
    condition_df,\
    diagnostic_report_df,\
    encounter_df,\
    immunization_df,\
    observation_df,\
    procedure_df = \
    process_one_file(sample_df, patient_df,
    careplan_df,
    condition_df,
    diagnostic_report_df,
    encounter_df,
    immunization_df,
    observation_df,
    procedure_df)

# Print the number of records in each DataFrame
patient_df.shape[0], careplan_df.shape[0], condition_df.shape[0], diagnostic_report_df.shape[0], encounter_df.shape[0], immunization_df.shape[0],\
observation_df.shape[0], procedure_df.shape[0]

# Print the columns of each DataFrame for inspection
for df in [patient_df, careplan_df, condition_df, diagnostic_report_df, encounter_df]:
    print(df.columns)
for df in [immunization_df, observation_df, procedure_df]:
    print(df.columns)

# Example of accessing specific data from the DataFrames
observation_df.iloc[0]["resource.code.coding"]
encounter_df.iloc[0]["resource.type"]

# Function to extract a subitem by its name from a list of items
def extract_subitems_by_name(item_list, name):
    for item in item_list:
        if item[name]:
            return item[name]

# Function to extract the first subitem by its name from a list of items
def extract_one_subitem_by_name(item_list, name):
    return item_list[0][name]

# Extract and process specific fields in patient_df
patient_df["birth_date"] = patient_df["resource.birthDate"].apply(lambda x: dt.strptime(x, '%Y-%m-%d'))
patient_df["birth_year"] = patient_df["birth_date"].apply(lambda x: x.year)
# Extract and process specific fields in observation_df
observation_df["date"] = observation_df["resource.effectiveDateTime"].apply(lambda x: dt.strptime(x[0:10], '%Y-%m-%d'))
observation_df["resource.code.code"] = observation_df["resource.code.coding"].apply(lambda x: extract_one_subitem_by_name(x, 'code'))
observation_df["resource.code.display"] = observation_df["resource.code.coding"].apply(lambda x: extract_one_subitem_by_name(x, 'display'))
# Extract and process specific fields in patient_df
patient_df["resource.extension.url"] = patient_df["resource.extension"].apply(lambda x: extract_one_subitem_by_name(x, 'url'))
# Extract and process specific fields in condition_df
condition_df["resource.code.code"] = condition_df["resource.code.coding"].apply(lambda x: extract_one_subitem_by_name(x, 'code'))
condition_df["resource.code.display"] = condition_df["resource.code.coding"].apply(lambda x: extract_one_subitem_by_name(x, 'display'))
# Extract and process specific fields in careplan_df
careplan_df["resource.category.coding"] = careplan_df["resource.category"].apply(lambda x: extract_one_subitem_by_name(x, 'coding'))
careplan_df["resource.category.coding.code"] = careplan_df["resource.category.coding"].apply(lambda x: extract_one_subitem_by_name(x, 'code'))
careplan_df["resource.category.coding.display"] = careplan_df["resource.category.coding"].apply(lambda x: extract_one_subitem_by_name(x, 'display'))
careplan_df["resource.activity.detail"] = careplan_df["resource.activity"].apply(lambda x: extract_one_subitem_by_name(x, 'detail'))
# Extract and process specific fields in diagnostic_report_df
diagnostic_report_df["resource.performer.display"] = diagnostic_report_df["resource.performer"].apply(lambda x: extract_one_subitem_by_name(x, 'display'))
diagnostic_report_df["resource.code.coding.code"] = diagnostic_report_df["resource.code.coding"].apply(lambda x: extract_one_subitem_by_name(x, 'code'))
diagnostic_report_df["resource.code.coding.display"] = diagnostic_report_df["resource.code.coding"].apply(lambda x: extract_one_subitem_by_name(x, 'display'))
diagnostic_report_df["resource.result.display"] = diagnostic_report_df["resource.result"].apply(lambda x: extract_one_subitem_by_name(x, 'display'))
# Extract and process specific fields in encounter_df
encounter_df["resource.type.text"] = encounter_df["resource.type"].apply(lambda x: extract_one_subitem_by_name(x, 'text'))
# Extract and process specific fields in immunization_df
immunization_df["resource.vaccineCode.coding.code"] = immunization_df["resource.vaccineCode.coding"].apply(lambda x: extract_one_subitem_by_name(x, 'code'))
immunization_df["resource.vaccineCode.coding.display"] = immunization_df["resource.vaccineCode.coding"].apply(lambda x: extract_one_subitem_by_name(x, 'display'))

```

```

# Save each DataFrame to a CSV file
observation_df.to_csv('observations.csv', index=False)
patient_df.to_csv('patients.csv', index=False)
condition_df.to_csv('conditions.csv', index=False)
careplan_df.to_csv('careplans.csv', index=False)
diagnostic_report_df.to_csv('reports.csv', index=False)
encounter_df.to_csv('encounters.csv', index=False)
immunization_df.to_csv('immunizations.csv', index=False)
procedure_df.to_csv('procedures.csv', index=False)

# Columns to drop from the DataFrames
columns_to_drop = ['fullUrl', 'resource.encounter.reference', 'resource.valueQuantity.system', 'resource.component', 'resource.text.div', 'resource.telecom', 'resource.photo', 'resource.context.reference', 'resource.performer']
# Create copies of DataFrames for column dropping
observation_df2 = observation_df.copy()
patient_df2 = patient_df.copy()
condition_df2 = condition_df.copy()
careplan_df2 = careplan_df.copy()
diagnostic_report_df2 = diagnostic_report_df.copy()
encounter_df2 = encounter_df.copy()
immunization_df2 = immunization_df.copy()
procedure_df2 = procedure_df.copy()

# Drop specified columns from each DataFrame if they exist
for column in columns_to_drop:
    if column in observation_df.columns:
        observation_df2.drop(columns=column, inplace=True)
        observation_df2['resource.subject.reference'] = observation_df2['resource.subject.reference'].str.replace('urn:uuid:', '')
    if column in patient_df.columns:
        patient_df2.drop(columns=column, inplace=True)
    if column in condition_df.columns:
        condition_df2.drop(columns=column, inplace=True)
        condition_df2['resource.subject.reference'] = condition_df2['resource.subject.reference'].str.replace('urn:uuid:', '')
    if column in careplan_df.columns:
        careplan_df2.drop(columns=column, inplace=True)
        careplan_df2['resource.subject.reference'] = careplan_df2['resource.subject.reference'].str.replace('urn:uuid:', '')
    if column in diagnostic_report_df.columns:
        diagnostic_report_df2.drop(columns=column, inplace=True)
        diagnostic_report_df2['resource.subject.reference'] = diagnostic_report_df2['resource.subject.reference'].str.replace('urn:uuid:', '')
    if column in encounter_df.columns:
        encounter_df2.drop(columns=column, inplace=True)
        encounter_df2['resource.patient.reference'] = encounter_df2['resource.patient.reference'].str.replace('urn:uuid:', '')
    if column in immunization_df.columns:
        immunization_df2.drop(columns=column, inplace=True)
        immunization_df2['resource.patient.reference'] = immunization_df2['resource.patient.reference'].str.replace('urn:uuid:', '')
    if column in procedure_df.columns:
        procedure_df2.drop(columns=column, inplace=True)
        procedure_df2['resource.subject.reference'] = procedure_df2['resource.subject.reference'].str.replace('urn:uuid:', '')

# Save the filtered DataFrames to new CSV files
observation_df2.to_csv('filt_observations.csv', index=False)
patient_df2.to_csv('filt_patients.csv', index=False)
condition_df2.to_csv('filt_conditions.csv', index=False)
careplan_df2.to_csv('filt_careplans.csv', index=False)
diagnostic_report_df2.to_csv('filt_reports.csv', index=False)
encounter_df2.to_csv('filt_encounters.csv', index=False)
immunization_df2.to_csv('filt_immunizations.csv', index=False)
procedure_df2.to_csv('filt_procedures.csv', index=False)

# PLOTTING
# Function to plot the count of a given feature in a DataFrame
def plot_count(feature, title, df, size=1, ordered=True):
    sns.set_theme(style="whitegrid") # Set the visual theme for the plots
    f, ax = plt.subplots(1,1, figsize=(4*size,4)) # Create a figure with appropriate size
    total = float(len(df)) # Calculate the total number of data points

    # Plot the count of the feature in the DataFrame, ordering by frequency if specified
    if ordered:
        g = sns.countplot(x=feature, data=df, order = df[feature].value_counts().index[:20], palette="Pastell1")
    else:
        g = sns.countplot(x=feature, data=df, palette='Pastell1')

    g.set_title("Number and percentage of {}".format(title)) # Set the plot title

    # Rotate x-axis labels if the size of the plot is larger than 2
    if(size > 2):
        plt.xticks(rotation=90, size=8)

    # Add percentage labels above each bar in the bar plot
    for p in ax.patches:
        height = p.get_height()
        ax.text(p.get_x()+p.get_width()/2.,
                height + 3,
                '{:1.2f}%'.format(100*height/total),
                ha="center") # Position text in the center above the bar
    plt.show()

# Plot the distribution of patient gender
plot_count("resource.gender", "Patient gender", df=patient_df2, size=1)

# Plot the number of patients born per year
agg_year = patient_df2.groupby(["birth_year"])[["resource.gender"].count().reset_index()
agg_year.columns = ["birth_year", "count"] # Rename columns for clarity
plt.plot(agg_year["birth_year"], agg_year["count"]) # Line plot for patient births by year
plt.title("Patients births / year")
plt.xlabel("Year")
plt.ylabel("Patients births")
plt.show()

# Plot the number of patients born per year, separated by gender
agg_year = patient_df2.groupby(["birth_year", "resource.gender"])[["resource.id"].count().reset_index()
agg_year.columns = ["birth_year", "gender", "count"] # Rename columns for clarity
plt.plot(agg_year.loc[agg_year.gender=="male", "birth_year"], agg_year.loc[agg_year.gender=="male", "count"], label="male") # Male births
plt.plot(agg_year.loc[agg_year.gender=="female", "birth_year"], agg_year.loc[agg_year.gender=="female", "count"], label="female") # Female births
plt.title("Patients births / year")
plt.xlabel("Year")
plt.ylabel("Patients births")
plt.legend() # Add a legend to distinguish between male and female
plt.show()

```

```

# Plot the distribution of careplan statuses
plot_count("resource.status", "Status of careplan", df=careplan_df2, size=1)

# Plot the distribution of class codes in encounters, with a large plot size
plot_count("resource.class.code", "class code (encounter)", df=encounter_df2, size=3)

# Plot the distribution of procedure codes with a larger plot size
plot_count("resource.code.text", "code text (procedure)", df=procedure_df2, size=4)

# Plot the distribution of observation display codes with a larger plot size
plot_count("resource.code.display", "display (observation)", df=observation_df2, size=4)

# I present the distribution of data for the observations, categorized by code.display.
# Unique observation display codes to plot
resource_code_display = observation_df2["resource.code.display"].unique()

num_plots = 0 # Counter for the number of plots to be created
colors = ["orangered", "lime", "fuchsia", "navy"] # Define color palette

# Count the number of displays to plot, skipping specific display names
for display in resource_code_display[0:25]:
    if display not in ["Blood Pressure", "Hemoglobin A1c/Hemoglobin.total in Blood", "Creatinine"]:
        num_plots += 1

# Determine the grid layout for the subplots (max 5 columns per row)
num_rows = (num_plots - 1) // 5 + 1
num_cols = min(num_plots, 5)

# Create the figure for subplots
f, ax = plt.subplots(num_rows, num_cols, figsize=(20, 24))

# Iterate through the first 25 display codes, excluding specific ones, and create subplots
i = 0
for display in resource_code_display[0:25]:
    i += 1
    plt.subplot(num_rows, num_cols, i)

    # Filter the data for the specific display and create a distribution plot
    sel_df = observation_df2.loc[observation_df2["resource.code.display"] == display]
    g = sns.displot(sel_df["resource.valueQuantity.value"], color=colors[i % 4])
    g.set(ylabel=None)
    g.set(xlabel=display) # Set x-axis label as the display name

# Remove empty subplots (if any) to avoid plotting issues
for j in range(num_plots + 1, num_rows * num_cols + 1):
    f.delaxes(ax.flatten()[j-1])

# Show the entire grid of plots
plt.show()

```

```

# Get unique observation display codes from the DataFrame
resource_code_display = observation_df2["resource.code.display"].unique()

# Set up the number of rows and columns for the subplot grid
num_rows = 5
num_cols = 5
num_plots = 0

# Define a color palette to be used for the plots
colors = ["orangered", "lime", "fuchsia", "navy"]

# Count the number of valid displays to plot (excluding certain display names)
for display in resource_code_display[26:50]:
    if display not in ["Cause of Death [US Standard Certificate of Death]", "Sexual orientation", "HIV status",
                      "Abuse Status [OMAHA]", "Housing status", "Are you covered by health insurance or some other kind of health care plan [PhenX]"]:
        num_plots += 1

# Create a figure and a grid of subplots to display the distributions
f, ax = plt.subplots(num_rows, num_cols, figsize=(20, 24))
i = 0

# Loop through the display codes and plot the distribution for each (excluding certain codes)
for display in resource_code_display[26:50]:
    i += 1
    plt.subplot(num_rows, num_cols, i) # Create a subplot in the grid
    # Filter the DataFrame for the specific display code
    sel_df = observation_df2.loc[observation_df2["resource.code.display"] == display]
    # Create a distribution plot for the filtered data
    g = sns.displot(sel_df["resource.valueQuantity.value"], color=colors[i % 4])
    g.set(ylabel=None) # Remove y-axis labels
    g.set(xlabel=display) # Set the x-axis label to the display name

# Remove empty subplots if the number of plots is less than the total grid size
for j in range(num_plots+1, num_rows*num_cols+1):
    f.delaxes(ax.flatten()[j-1])

# Show the grid of plots
plt.show()

# Violin plot: Distribution of 'valueQuantity.value' by 'code.display' for all observations
plt.figure(figsize=(12, 6))
sns.violinplot(x='resource.code.display', y='resource.valueQuantity.value', data=observation_df2, palette='viridis')
plt.title('Violin Plot of valueQuantity.value by code.display')
plt.xticks(rotation=90) # Rotate x-axis labels for better readability
plt.show()

# Violin plot: Distribution of 'valueQuantity.value' by 'code.display', this time flipping the axes
plt.figure(figsize=(12, 6))
sns.violinplot(y='resource.code.display', x='resource.valueQuantity.value', data=observation_df2, palette='viridis')
plt.title('Violin Plot of valueQuantity.value by code.display')
plt.xticks(rotation=90) # Rotate x-axis labels for better readability
plt.show()

# Violin plot: Focusing on specific observation types like Body Weight, Height, and BMI
plt.figure(figsize=(12, 6))
sns.violinplot(x='resource.valueQuantity.value', y='resource.code.display', hue='resource.code.display',
               data=observation_df2[observation_df2['resource.code.display'].isin(['Body Weight', 'Body Height', 'Body Mass Index'])],
               palette='viridis', dodge=True)
plt.xticks(range(0, 240, 10)) # Set x-axis ticks range for body measurements
plt.title('Violin Plot of Body Weight, Body Height, and Body Mass Index')
plt.xlabel('Value Quantity Value')
plt.ylabel('Code Display')
plt.legend(title='Observation Type')
plt.show()

```

```

# Violin plot: Focusing on cholesterol and glucose-related observations
plt.figure(figsize=(12, 6))
codes_to_include = ['Total Cholesterol', 'Triglycerides', 'Low Density Lipoprotein Cholesterol',
                    'High Density Lipoprotein Cholesterol', 'Glucose']
filtered_df = observation_df2[observation_df2['resource.code.display'].isin(codes_to_include)]
sns.violinplot(x='resource.valueQuantity.value', y='resource.code.display',
              data=filtered_df, palette='plasma', dodge=True)
plt.xticks(range(0, 620, 20)) # Set x-axis ticks range for cholesterol and glucose measurements
plt.title('Violin Plot of Total Cholesterol, Triglycerides, Low Density Lipoprotein Cholesterol, High Density Lipoprotein Cholesterol, Glucose')
plt.xlabel('Value Quantity Value')
plt.ylabel('Code Display')
plt.show()

# Violin plot: Sodium and Chloride levels
plt.figure(figsize=(12, 6))
sns.violinplot(x='resource.valueQuantity.value', y='resource.code.display', hue='resource.code.display',
              data=observation_df2[observation_df2['resource.code.display'].isin(['Sodium', 'Chloride'])],
              palette='viridis', dodge=True)
plt.xticks(range(90, 155, 5)) # Set x-axis ticks for Sodium and Chloride
plt.title('Violin Plot of Sodium, Chloride')
plt.xlabel('Value Quantity Value')
plt.ylabel('Code Display')
plt.legend(title='Observation Type')
plt.show()

# Violin plot: Urea Nitrogen, Calcium, Potassium, and Carbon Dioxide levels
plt.figure(figsize=(12, 6))
sns.violinplot(x='resource.valueQuantity.value', y='resource.code.display', hue='resource.code.display',
              data=observation_df2[observation_df2['resource.code.display'].isin(['Urea Nitrogen', 'Calcium', 'Potassium', 'Carbon Dioxide'])],
              palette='inferno', dodge=True)
plt.xticks(range(0, 33, 1)) # Set x-axis ticks for these chemical levels
plt.title('Violin Plot of Urea Nitrogen, Calcium, Potassium, Carbon Dioxide')
plt.xlabel('Value Quantity Value')
plt.ylabel('Code Display')
plt.legend(title='Observation Type')
plt.show()

# Violin plot: FEV1/FVC, hospitalizations, and polyp size
plt.figure(figsize=(12, 6))
sns.violinplot(x='resource.valueQuantity.value', y='resource.code.display', hue='resource.code.display',
              data=observation_df2[observation_df2['resource.code.display'].isin(['FEV1/\u200bFVC', 'History of Hospitalizations+\u200bOutpatient visits',
                                      'Polyp size greatest dimension by CAP cancer protocols'])],
              palette='cividis', dodge=True)
plt.xticks(range(0, 105, 5)) # Set x-axis ticks for FEV1/FVC, hospitalizations, and polyp size
plt.title('Violin Plot of FEV1/FVC, History of Hospitalizations+Outpatient visits, Polyp size greatest dimension by CAP cancer protocols')
plt.xlabel('Value Quantity Value')
plt.ylabel('Code Display')
plt.legend(title='Observation Type')
plt.show()

# Violin plot: Eczema percentage area affected and MMSE total scores
plt.figure(figsize=(12, 6))
codes_to_include = ['Percentage area affected by eczema Head and Neck', 'Percentage area affected by eczema Trunk',
                    'Percentage area affected by eczema Upper extremity - bilateral',
                    'Percentage area affected by eczema Lower extremity - bilateral',
                    'Total score [MMSE]']
filtered_df = observation_df2[observation_df2['resource.code.display'].isin(codes_to_include)]
sns.violinplot(x='resource.valueQuantity.value', y='resource.code.display',
              data=filtered_df, palette='viridis', dodge=True)
plt.xticks(range(-30, 95, 5)) # Set x-axis ticks for eczema and MMSE score range
plt.title('Violin Plot of eczema, Total score [MMSE]')
plt.xlabel('Value Quantity Value')
plt.ylabel('Code Display')

# Remove the legend for a cleaner look
plt.legend().remove()
plt.show()

```

```

# Selection of observation data for one patient
# Identify the patient with the most observations available in the dataset
sel_patient = observation_df2["resource.subject.reference"].value_counts().index[0]
print(sel_patient)

# Set number of top patients to select (in this case, selecting 1)
n_patients = 1
top_patients = observation_df2["resource.subject.reference"].value_counts().index[:n_patients]

# Filter observation data for the top patient and save to a CSV file
observation_df3 = observation_df2[observation_df2["resource.subject.reference"].isin(top_patients)]
observation_df3.to_csv('filt_observations3.csv', index=False)

# Repeat the process for the condition data: find the patient with the most conditions
sel_patient = condition_df2["resource.subject.reference"].value_counts().index[0]
print(sel_patient)

# Select the top patient based on conditions and save filtered data to a CSV file
top_patients = condition_df2["resource.subject.reference"].value_counts().index[:n_patients]
condition_df3 = condition_df2[condition_df2["resource.subject.reference"].isin(top_patients)]
condition_df3.to_csv('filt_conditions3.csv', index=False)

# Repeat the process for patient data: find the patient with the most entries
sel_patient = patient_df2["resource.id"].value_counts().index[0]
print(sel_patient)

# Select the top patient based on patient data and save filtered data to a CSV file
top_patients = patient_df2["resource.id"].value_counts().index[:n_patients]
patient_df3 = patient_df2[patient_df2["resource.id"].isin(top_patients)]
patient_df3.to_csv('filt_patients3.csv', index=False)

# Repeat the process for care plan data
sel_patient = careplan_df2["resource.subject.reference"].value_counts().index[0]
print(sel_patient)

# Select the top patient based on care plan data and save filtered data to a CSV file
top_patients = careplan_df2["resource.subject.reference"].value_counts().index[:n_patients]
careplan_df3 = careplan_df2[careplan_df2["resource.subject.reference"].isin(top_patients)]
careplan_df3.to_csv('filt_careplans3.csv', index=False)

# Repeat the process for diagnostic report data
top_patients = diagnostic_report_df2["resource.subject.reference"].value_counts().index[:n_patients]
diagnostic_report_df3 = diagnostic_report_df2[diagnostic_report_df2["resource.subject.reference"].isin(top_patients)]
diagnostic_report_df3.to_csv('filt_reports3.csv', index=False)

# Repeat the process for encounter data
top_patients = encounter_df2["resource.patient.reference"].value_counts().index[:n_patients]
encounter_df3 = encounter_df2[encounter_df2["resource.patient.reference"].isin(top_patients)]
encounter_df3.to_csv('filt_encounters3.csv', index=False)

# Repeat the process for immunization data
top_patients = immunization_df2["resource.patient.reference"].value_counts().index[:n_patients]
immunization_df3 = immunization_df2[immunization_df2["resource.patient.reference"].isin(top_patients)]
immunization_df3.to_csv('filt_immunizations3.csv', index=False)

# Repeat the process for procedure data
top_patients = procedure_df2["resource.subject.reference"].value_counts().index[:n_patients]
procedure_df3 = procedure_df2[procedure_df2["resource.subject.reference"].isin(top_patients)]
procedure_df3.to_csv('filt_procedures3.csv', index=False)

```

```

# Plotting: Observation Data Visualization
# Create count plots for the top patient's observation data based on the code display and value quantity code
plot_count("resource.code.display", "display (observation)", df=observation_df3, size=4)
plot_count("resource.valueQuantity.code", "value quantity code (observation)", df=observation_df3, size=4)

# Select columns to be used for further plotting
sel_obs_columns = ["resource.code.display", "resource.effectiveDateTime", "resource.valueQuantity.value",
                  "resource.valueQuantity.unit", "resource.valueQuantity.system", "resource.valueQuantity.code", "date"]

# Extract unique code display values for the top patient
resource_code_display = observation_df3["resource.code.display"].unique()

# Create a grid of subplots to plot each unique code display's value over time
f, ax = plt.subplots(4, 5, figsize=(20, 24))
i = 0

# Loop through each code display and plot the corresponding values over time
for display in resource_code_display[:-2]: # Exclude the last 2 items for some reason
    i += 1
    plt.subplot(4, 5, i) # Create a subplot
    plt.legend(loc=2, prop={'size': 3})

    # Create a label with the display name and the value quantity code
    sel_df = observation_df3.loc[observation_df3["resource.code.display"] == display]
    label_text = f"{display}\n[{sel_df['resource.valueQuantity.code'].values[0]}]"

    # Plot the value quantity over time
    plt.plot(sel_df["date"], sel_df["resource.valueQuantity.value"], label=label_text)
    plt.tick_params(axis='x', rotation=45, size=6) # Rotate x-axis labels for better readability
    plt.xlabel("date")
    plt.legend()

# Show the grid of plots
plt.show()

# Repeating the plotting process for more patients (patients 1 to 4)
for idx in range(1, 5):
    # Select the patient with the idx-th most observations
    sel_patient = observation_df["resource.subject.reference"].value_counts().index[idx]
    print(f"Selected patient: {sel_patient}")

    # Filter the observation data for the selected patient
    obs_subset_df = observation_df.loc[observation_df["resource.subject.reference"] == sel_patient]

    # Get the unique code display values for this patient
    resource_code_display = obs_subset_df["resource.code.display"].unique()

    # Define color palette for the plots
    colors = ["red", "blue", "green", "magenta", "orange"]

    # Create a grid of subplots to plot the selected patient's data
    f, ax = plt.subplots(4, 5, figsize=(20, 24))
    i = 0

    # Loop through each code display and plot the corresponding values over time for the patient
    for display in resource_code_display[0:20]: # Limit to the first 20 code displays
        i += 1
        plt.subplot(4, 5, i) # Create a subplot
        plt.legend(loc=2, prop={'size': 3})

        # Create a label with the display name and value quantity code
        sel_df = obs_subset_df.loc[obs_subset_df["resource.code.display"] == display]
        label_text = f"{display}\n[{sel_df['resource.valueQuantity.code'].values[0]}]"

        # Plot the value quantity over time using a unique color for each patient
        plt.plot(sel_df["date"], sel_df["resource.valueQuantity.value"], label=label_text, color=colors[idx])
        plt.tick_params(axis='x', rotation=45, size=6) # Rotate x-axis labels for better readability
        plt.xlabel("date")
        plt.legend()

    # Show the grid of plots for the selected patient
    plt.show()

```

```

import pandas as pd

# Load the patient data CSV file into a pandas DataFrame
file_path = 'filt_patients.csv'
df = pd.read_csv(file_path)
# Display the first few rows of the patient data
df.head()
# Drop unnecessary columns from the DataFrame related to patient information
df.drop(columns=['resource.text.status', 'resource.extension', 'resource.identifier',
                 'resource.name', 'resource.multipleBirthBoolean', 'resource.birthDate',
                 'resource.resourceType', 'resource.address', 'resource.deceasedDateTime',
                 'resource.multipleBirthInteger', 'resource.maritalStatus.coding',
                 'resource.extension.url'], inplace=True)

# Display the first few rows after dropping columns
df.head()
# Check for missing values (NaN) in each column and print the count
nan_counts = df.isna().sum()
print(nan_counts)
# Save the cleaned patient data to a new CSV file
df.to_csv("filt_patients2.csv", index=False)

# Load the observations data CSV file into a pandas DataFrame
file_path = 'filt_observations.csv'
df = pd.read_csv(file_path)
df.head()
# Display a specific value from the first row to check the data
df.iloc[0]["resource.code.coding"]
# Drop unnecessary columns from the observation data
df.drop(columns=['resource.id', 'resource.status', 'resource.code.coding',
                 'resource.effectiveDateTime', 'resource.valueQuantity.unit',
                 'resource.valueQuantity.code', 'resource.resourceType',
                 'resource.valueCodeableConcept.coding'], inplace=True)
# Remove rows with missing values
df.dropna(inplace=True)
df.head()
# Select the top n observations based on frequency in 'resource.code.display'
n_observations = 16
top_observations = df["resource.code.display"].value_counts().index[:n_observations]
# Filter the DataFrame to keep only the rows corresponding to the top observations
df = df[df["resource.code.display"].isin(top_observations)]
# Save the filtered observations data to a new CSV file
df.to_csv("filt_observations2.csv", index=False)
# Pivot the observations DataFrame to have one row per patient and columns for each observation type
df = df.pivot_table(index=['resource.subject.reference'],
                    columns='resource.code.display',
                    values='resource.valueQuantity.value',
                    aggfunc='first').reset_index()
# Check for missing values (NaN) in each column after pivoting
nan_counts = df.isna().sum()
print(nan_counts)
# Remove rows with any missing values after pivoting
df.dropna(inplace=True)
# Save the pivoted observations data to a new CSV file
df.to_csv("filt_observations2.csv", index=False)

# Load the conditions data CSV file into a pandas DataFrame
file_path = 'filt_conditions.csv'
df = pd.read_csv(file_path)
df.head()
# Display a specific value from the first row to check the data
df.iloc[0]["resource.code.coding"]
# Drop unnecessary columns from the conditions data
df.drop(columns=['resource.id', 'resource.clinicalStatus', 'resource.verificationsStatus',
                 'resource.code.coding', 'resource.onsetDateTime', 'resource.resourceType',
                 'resource.abatementDateTime', 'resource.code.code'], inplace=True)
df.head()

```

```

# Select the top n conditions based on frequency in 'resource.code.display'
n_conditions = 16
top_conditions = df["resource.code.display"].value_counts().index[:n_conditions]
# Filter the DataFrame to keep only the rows corresponding to the top conditions
df = df[df["resource.code.display"].isin(top_conditions)]
df.head()
# Remove rows with missing values
df.dropna(inplace=True)
# Save the filtered conditions data to a new CSV file
df.to_csv("filt_conditions2.csv", index=False)

# Load the encounters data CSV file into a pandas DataFrame
file_path = 'filt_encounters.csv'
df = pd.read_csv(file_path)
df.head()
# Print the unique classes of encounters in the dataset
unique_classes = df['resource.class.code'].unique()
print(unique_classes)
# Drop unnecessary columns from the encounters data
df.drop(columns=['resource.id', 'resource.status', 'resource.type',
                 'resource.period.start', 'resource.period.end',
                 'resource.resourceType', 'resource.reason.coding'], inplace=True)
df.head()
# Add a presence column to indicate whether an encounter occurred (set to True)
df['presence'] = True
# Pivot the encounters DataFrame to have one row per patient and columns for each encounter class
df = df.pivot_table(index='resource.patient.reference',
                    columns='resource.class.code',
                    values='presence',
                    aggfunc='max',
                    fill_value=False) # Replace NaNs with False (condition not present)
# Reset the index to make 'resource.subject.reference' a column again
df = df.reset_index()
# Replace boolean values with 1 (True) and 0 (False) for easier analysis
df.iloc[:, 1:] = df.iloc[:, 1:].astype(int)
# Remove rows with missing values
df.dropna(inplace=True)
# Rename 'resource.patient.reference' column to 'resource.subject.reference' to match other datasets
df.rename(columns={'resource.patient.reference': 'resource.subject.reference'}, inplace=True)
# Save the processed encounters data to a new CSV file
df.to_csv("filt_encounters2.csv", index=False)

# Load the care plans data CSV file into a pandas DataFrame
file_path = 'filt_careplans.csv'
df = pd.read_csv(file_path)
# Display a specific value from the first row to check the data
df.iloc[0]["resource.addresses"]
# Drop unnecessary columns from the care plans data
df.drop(columns=['resource.status', 'resource.category', 'resource.period.start',
                 'resource.addresses', 'resource.activity', 'resource.resourceType',
                 'resource.period.end', 'resource.category.coding',
                 'resource.category.coding.code', 'resource.activity.detail'], inplace=True)
df.head()
# Select the top n care plans based on frequency in 'resource.category.coding.display'
n_careplans = 16
top_careplans = df["resource.category.coding.display"].value_counts().index[:n_careplans]
# Filter the DataFrame to keep only the rows corresponding to the top care plans
df = df[df["resource.category.coding.display"].isin(top_careplans)]
# Add a presence column to indicate whether a care plan was provided (set to True)
df['presence'] = True
# Pivot the care plans DataFrame to have one row per patient and columns for each care plan category
df = df.pivot_table(index='resource.subject.reference',
                    columns='resource.category.coding.display',
                    values='presence',
                    aggfunc='max',
                    fill_value=False) # Replace NaNs with False (care plan not provided)
# Reset the index to make 'resource.subject.reference' a column again

```

```

df = df.reset_index()
# Replace boolean values with 1 (True) and 0 (False)
df.iloc[:, 1:] = df.iloc[:, 1:].astype(int)
# Remove rows with missing values
df.dropna(inplace=True)
# Save the processed care plans data to a new CSV file
df.to_csv("filt_careplans2.csv", index=False)

# Load the procedures data CSV file into a pandas DataFrame
file_path = 'filt_procedures.csv'
df = pd.read_csv(file_path)
df.head()
# Drop unnecessary columns from the procedures data
df.drop(columns=['resource.status', 'resource.code.coding', 'resource.performedDateTime',
                 'resource.resourceType', 'resource.performedPeriod.start',
                 'resource.performedPeriod.end', 'resource.reasonReference.reference'], inplace=True)
df.head()
# Select the top n procedures based on frequency in 'resource.code.text'
n_procedures = 16
top_procedures = df["resource.code.text"].value_counts().index[:n_procedures]
# Filter the DataFrame to keep only the rows corresponding to the top procedures
df = df[df["resource.code.text"].isin(top_procedures)]
# Add a presence column to indicate whether a procedure was performed (set to True)
df['presence'] = True
# Pivot the procedures DataFrame to have one row per patient and columns for each procedure
df = df.pivot_table(index='resource.subject.reference',
                    columns='resource.code.text',
                    values='presence',
                    aggfunc='max',
                    fill_value=False) # Replace NaNs with False (procedure not performed)
# Reset the index to make 'resource.subject.reference' a column again
df = df.reset_index()
# Replace boolean values with 1 (True) and 0 (False)
df.iloc[:, 1:] = df.iloc[:, 1:].astype(int)
# Remove rows with missing values
df.dropna(inplace=True)
# Save the processed procedures data to a new CSV file
df.to_csv("filt_procedures2.csv", index=False)

# Load the immunizations data CSV file into a pandas DataFrame
file_path = 'filt_immunizations.csv'
df = pd.read_csv(file_path)
df.head()
# Display a specific value from the first row to check the data
df.iloc[0]["resource.vaccineCode.coding"]
# Drop unnecessary columns from the immunizations data
df.drop(columns=['resource.status', 'resource.date', 'resource.vaccineCode.coding',
                 'resource.wasNotGiven', 'resource.primarySource', 'resource.resourceType',
                 'resource.vaccineCode.coding.code'], inplace=True)
df.head()
# Add a presence column to indicate whether an immunization was administered (set to True)
df['presence'] = True
# Pivot the immunizations DataFrame to have one row per patient and columns for each vaccine type
df = df.pivot_table(index='resource.patient.reference',
                    columns='resource.vaccineCode.coding.display',
                    values='presence',
                    aggfunc='max',
                    fill_value=False) # Replace NaNs with False (vaccine not administered)
# Reset the index to make 'resource.subject.reference' a column again
df = df.reset_index()
# Replace boolean values with 1 (True) and 0 (False)
df.iloc[:, 1:] = df.iloc[:, 1:].astype(int)
# Remove rows with missing values
df.dropna(inplace=True)
# Rename 'resource.patient.reference' column to 'resource.subject.reference' to match other datasets
df.rename(columns={'resource.patient.reference': 'resource.subject.reference'}, inplace=True)

# Save the processed immunizations data to a new CSV file
df.to_csv("filt_immunizations2.csv", index=False)

```

```

import pandas as pd

# Load the processed patient data
file_path = 'filt_patients2.csv'
p_df = pd.read_csv(file_path)
# Load the processed observations data
file_path = 'filt_observations2.csv'
o_df = pd.read_csv(file_path)
# Load the processed conditions data
file_path = 'filt_conditions2.csv'
c_df = pd.read_csv(file_path)
# Load the processed encounters data
file_path = 'filt_encounters2.csv'
e_df = pd.read_csv(file_path)
# Load the processed care plans data
file_path = 'filt_careplans2.csv'
care_df = pd.read_csv(file_path)
# Load the processed procedures data
file_path = 'filt_procedures2.csv'
proc_df = pd.read_csv(file_path)
# Load the processed immunizations data
file_path = 'filt_immunizations2.csv'
imm_df = pd.read_csv(file_path)
# Merge observations with patient data on the patient reference
merged_df = pd.merge(o_df, p_df[['resource.id', 'resource.gender', 'age']], left_on='resource.subject.reference', right_on='resource.id', how='left')
# Drop the redundant 'resource.id' column after merging
merged_df.drop(columns=['resource.id'], inplace=True)
# Convert 'resource.gender' to integers: male -> 0, female -> 1
merged_df['resource.gender'] = merged_df['resource.gender'].replace({'male': 0, 'female': 1}).astype(int)
# Save the first merged DataFrame to a CSV file
merged_df.to_csv("merging1.csv", index=False)

# Merge with conditions data on 'resource.subject.reference'
merged_df = pd.merge(merged_df, c_df, on='resource.subject.reference')
# Display the first few rows of the merged DataFrame and save it to a CSV file
merged_df.head()
merged_df.to_csv("merging2.csv", index=False)

# Merge with encounters data on 'resource.subject.reference'
merged_df = pd.merge(merged_df, e_df, on='resource.subject.reference')
# Display the first few rows of the merged DataFrame and save it to a CSV file
merged_df.head()
merged_df.to_csv("merging3.csv", index=False)

# Merge with care plans data on 'resource.subject.reference'
merged_df = pd.merge(merged_df, care_df, on='resource.subject.reference')
# Display the first few rows of the merged DataFrame and save it to a CSV file
merged_df.head()
merged_df.to_csv("merging4.csv", index=False)

# Merge with procedures data on 'resource.subject.reference'
merged_df = pd.merge(merged_df, proc_df, on='resource.subject.reference')
# Display the first few rows of the merged DataFrame and save it to a CSV file
merged_df.head()
merged_df.to_csv("merging5.csv", index=False)

# Merge with immunizations data on 'resource.subject.reference'
merged_df = pd.merge(merged_df, imm_df, on='resource.subject.reference')
# Display the first few rows of the merged DataFrame and save it to a CSV file
merged_df.head()
merged_df.to_csv("merging6.csv", index=False)

# Load the final merged data from the CSV file
file_path = 'merging6.csv'
df = pd.read_csv(file_path)
# Loop through each column and get the value counts to understand the distribution of data
-

```

```

for column in df.columns:
    print(f"Unique value counts for column: {column}")
    print(df[column].value_counts())
    print("\n")

# Specify a list of columns to drop based on previous analysis (removing irrelevant features)
columns_to_drop = [
    "resource.subject.reference", "Creatinine", "Acute bacterial sinusitis (disorder)", "Acute bronchitis (disorder)",
    "Appendicitis", "Chronic sinusitis (disorder)", "Diabetes", "History of appendectomy",
    "Normal pregnancy", "Otitis media", "Polyp of colon", "Prediabetes", "Sprain of ankle",
    "Streptococcal sore throat (disorder)", "Stroke", "ambulatory", "daytime", "inpatient",
    "outpatient", "Angina self management plan", "Asthma self management", "Burn care",
    "Chronic obstructive pulmonary disease clinical management plan", "Diabetes self management plan",
    "Fracture care", "Head injury rehabilitation", "Musculoskeletal care", "Physical therapy procedure",
    "Routine antenatal care", "Self care", "Skin condition care", "Terminal care",
    "Urinary tract infection care", "Wound care", "Allergy screening test", "Bone immobilization",
    "Childbirth", "Cognitive and behavioral therapy", "Combined chemotherapy and radiation therapy (procedure)",
    "Documentation of current medications", "Information gathering (procedure)", "Measurement of respiratory function (procedure)",
    "Pulmonary rehabilitation (regime/therapy)", "Spirometry (procedure)", "Sputum examination (procedure)",
    "Standard pregnancy test", "Subcutaneous immunotherapy", "Suture open wound",
    "DTaP", "HPV, quadrivalent", "Hep A, ped/adol, 2 dose", "Hep B, adolescent or pediatric",
    "Hib (PRP-OMP)", "IPV", "Influenza, seasonal, injectable, preservative free",
    "MMR", "Pneumococcal conjugate PCV 13", "Tdap", "meningococcal MCV4P",
    "pneumococcal polysaccharide vaccine, 23 valent", "rotavirus, monovalent",
    "varicella", "zoster"
]

# Drop the specified columns that are no longer needed for the analysis
df = df.drop(columns=columns_to_drop)
# Display the first few rows after dropping the columns
df.head()
# Drop a specific column by its index (23rd column)
df = df.drop(df.columns[23], axis=1)
# Save the final cleaned and merged dataset to a new CSV file
df.to_csv("merging7.csv", index=False)

```

```

#MACHINE LEARNING
from sklearn.model_selection import train_test_split
from lime.lime_tabular import LimeTabularExplainer
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
import shap
import xgboost as xgb
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, f1_score, recall_score, roc_auc_score
from sklearn.inspection import PartialDependenceDisplay
from lime import lime_tabular
import lime

file_path = 'merging7.csv' # Replace with your actual file path
df = pd.read_csv(file_path)
X = df.drop(columns=['resource.gender']) # Replace 'target_column' with your target column name
y = df['resource.gender']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize XGBoost classifier
model = XGBClassifier()
# Train the classifier
model.fit(X_train, y_train)
# Make predictions on the testing data
predictions = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')
roc_auc = roc_auc_score(y_test, predictions)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("roc_auc score:", roc_auc)

explainer = shap.Explainer(model, X_train)
shap_values = explainer(X_test)
# Waterfall plot
plt.figure()
shap.waterfall_plot(shap_values[0], max_display=24)
plt.savefig('xgboost1_waterfall_plot.png', bbox_inches='tight')
plt.close()
# Beeswarm plot
plt.figure()
shap.plots.beeswarm(shap_values, max_display=24)
plt.savefig('xgboost1_beeswarm_plot.png', bbox_inches='tight')
plt.close()
# Bar plot
plt.figure()
shap.plots.bar(shap_values, max_display=24)
plt.savefig('xgboost1_bar_plot.png', bbox_inches='tight')
plt.close()
# Summary plot
plt.figure()
shap.summary_plot(shap_values, X_test, show=False)
plt.savefig('xgboost1_summary_plot.png', bbox_inches='tight')
plt.close()

```

```

columns = [
    "Body Height", "Body Mass Index", "Body Weight", "Calcium", "Carbon Dioxide",
    "Chloride", "Glucose",
    "High Density Lipoprotein Cholesterol", "Low Density Lipoprotein Cholesterol",
    "Potassium", "Sodium", "Total Cholesterol", "Triglycerides", "Urea Nitrogen",
    "age", "Acute viral pharyngitis (disorder)", "Hypertension",
    "Viral sinusitis (disorder)", "emergency", "Respiratory therapy", "Colonoscopy",
    "Td (adult) preservative free"
]

# Loop through each column and generate both SHAP and Partial Dependence plots
for column in columns:
    # SHAP Dependence Plot
    fig = plt.figure(figsize=(8, 6))
    shap.dependence_plot(column, shap_values.values, X_test, interaction_index="age", show=False)
    plt.title(f"SHAP Dependence Plot for {column} (XGBoost)")
    plt.savefig(f'xgboost1_SHAP_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()
    # Partial Dependence Plot
    fig, ax = plt.subplots(figsize=(8, 6))
    PartialDependenceDisplay.from_estimator(model, X_test, [column], ax=ax)
    ax.set_title(f"Partial Dependence Plot for {column} (XGBoost)")
    plt.savefig(f'xgboost1_Partial_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()

# Adding LIME
lime_explainer = lime.lime_tabular.LimeTabularExplainer(
    X_train.values, # Use the training data for LIME background distribution
    feature_names=X.columns,
    class_names=['Male', 'Female'], # Adjusted: 'Male' corresponds to 0, 'Female' corresponds to 1
    discretize_continuous=True
)

# Explaining a single instance from X_test (e.g., the first row)
i = 0 # Index of the instance in X_test to explain
exp = lime_explainer.explain_instance(
    X_test.values[i], # Instance to explain
    model.predict_proba, # Prediction function
    num_features=24 # Number of features to show in explanation
)

# Show the explanation for the single instance
fig = exp.as_pyplot_figure()
plt.savefig(f'xgboost1_LIME_explanation_instance_{i}.png', bbox_inches='tight')
plt.close()

#CATBOOST
file_path = 'merging7.csv' # Replace with your actual file path
df = pd.read_csv(file_path)
# Assuming df is already defined with your dataset
X = df.drop(columns=['resource.gender']) # Replace 'resource.gender' with your target column name
y = df['resource.gender']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize CatBoost classifier
model = CatBoostClassifier(iterations=1000, depth=6, learning_rate=0.1, loss_function='Logloss', cat_features=None, verbose=200)
# Train the classifier
model.fit(X_train, y_train)
# Make predictions on the testing data
predictions = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')

```

```

recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')
roc_auc = roc_auc_score(y_test, predictions)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("roc_auc score:", roc_auc)

# SHAP values for CatBoost
explainer = shap.Explainer(model)
shap_values = explainer(X_test)
# Waterfall plot
plt.figure()
shap.waterfall_plot(shap_values[0], max_display=24)
plt.savefig('catboost1_waterfall_plot.png', bbox_inches='tight')
plt.close()
# Beeswarm plot
plt.figure()
shap.plots.beeswarm(shap_values, max_display=24)
plt.savefig('catboost1_beeswarm_plot.png', bbox_inches='tight')
plt.close()
# Bar plot
plt.figure()
shap.plots.bar(shap_values, max_display=24)
plt.savefig('catboost1_bar_plot.png', bbox_inches='tight')
plt.close()
# Summary plot
plt.figure()
shap.summary_plot(shap_values, X_test, show=False)
plt.savefig('catboost1_summary_plot.png', bbox_inches='tight')
plt.close()

columns = [
    "Body Height", "Body Mass Index", "Body Weight", "Calcium", "Carbon Dioxide",
    "Chloride", "Glucose",
    "High Density Lipoprotein Cholesterol", "Low Density Lipoprotein Cholesterol",
    "Potassium", "Sodium", "Total Cholesterol", "Triglycerides", "Urea Nitrogen",
    "age", "Acute viral pharyngitis (disorder)", "Hypertension",
    "Viral sinusitis (disorder)", "emergency", "Respiratory therapy", "Colonoscopy",
    "Td (adult) preservative free"
]

# Loop through each column and generate both SHAP and Partial Dependence plots
for column in columns:
    # SHAP Dependence Plot
    fig = plt.figure(figsize=(8, 6))
    shap.dependence_plot(column, shap_values.values, X_test, interaction_index="age", show=False)
    plt.title(f"SHAP Dependence Plot for {column} (CatBoost)")
    plt.savefig(f'catboost1_SHAP_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()
    # Partial Dependence Plot
    fig, ax = plt.subplots(figsize=(8, 6))
    PartialDependenceDisplay.from_estimator(model, X_test, [column], ax=ax)
    ax.set_title(f"Partial Dependence Plot for {column} (CatBoost)")
    plt.savefig(f'catboost1_Partial_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()

# Adding LIME
lime_explainer = lime.lime_tabular.LimeTabularExplainer(
    X_train.values, # Use the training data for LIME background distribution
    feature_names=X.columns,
    class_names=['Male', 'Female'], # Adjusted: 'Male' corresponds to 0, 'Female' corresponds to 1
    discretize_continuous=True
)

```

```

# Explaining a single instance from X_test (e.g., the first row)
i = 0 # Index of the instance in X_test to explain
exp = lime_explainer.explain_instance(
    X_test.values[i], # Instance to explain
    model.predict_proba, # Prediction function
    num_features=24 # Number of features to show in explanation
)

# Show the explanation for the single instance
fig = exp.as_pyplot_figure()
plt.savefig(f'catboost1_LIME_explanation_instance_{i}.png', bbox_inches='tight')
plt.close()

#XGBOOST
file_path = 'merging7.csv' # Replace with your actual file path
df = pd.read_csv(file_path)
X = df.drop(columns=['Colonoscopy']) # Replace 'target_column' with your target column name
y = df['Colonoscopy']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize XGBoost classifier
model = XGBClassifier()
# Train the classifier
model.fit(X_train, y_train)
# Make predictions on the testing data
predictions = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')
roc_auc = roc_auc_score(y_test, predictions)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("roc_auc score:", roc_auc)

explainer = shap.Explainer(model, X_train)
shap_values = explainer(X_test)

# Visualize the first prediction's explanation with feature names
# Waterfall plot
plt.figure()
shap.waterfall_plot(shap_values[0], max_display=24)
plt.savefig('xgboost4_waterfall_plot.png', bbox_inches='tight')
plt.close()
# Beeswarm plot
plt.figure()
shap.plots.beeswarm(shap_values, max_display=24)
plt.savefig('xgboost4_beeswarm_plot.png', bbox_inches='tight')
plt.close()
# Bar plot
plt.figure()
shap.plots.bar(shap_values, max_display=24)
plt.savefig('xgboost4_bar_plot.png', bbox_inches='tight')
plt.close()

```

```

columns = [
    "Body Height", "Body Mass Index", "Body Weight", "Calcium", "Carbon Dioxide",
    "Chloride", "Glucose",
    "High Density Lipoprotein Cholesterol", "Low Density Lipoprotein Cholesterol",
    "Potassium", "Sodium", "Total Cholesterol", "Triglycerides", "Urea Nitrogen",
    "age", "resource.gender", "Acute viral pharyngitis (disorder)",
    "Respiratory therapy", "Viral sinusitis (disorder)", "emergency", "Td (adult) preservative free",
    "Hypertension"
]

# Loop through each column and generate both SHAP and Partial Dependence plots
for column in columns:
    # SHAP Dependence Plot
    fig = plt.figure(figsize=(8, 6))
    shap.dependence_plot(column, shap_values.values, X_test, interaction_index="age", show=False)
    plt.title(f"SHAP Dependence Plot for {column} (XGBoost)")
    plt.savefig(f'xgboost4_SHAP_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()
    # Partial Dependence Plot
    fig, ax = plt.subplots(figsize=(8, 6))
    PartialDependenceDisplay.from_estimator(model, X_test, [column], ax=ax)
    ax.set_title(f"Partial Dependence Plot for {column} (XGBoost)")
    plt.savefig(f'xgboost4_Partial_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()
# Adding LIME
lime_explainer = lime_tabular.LimeTabularExplainer(
    X_train.values, # Use the training data for LIME background distribution
    feature_names=X.columns,
    class_names=['No Colonoscopy', 'Colonoscopy'], # Adjusted: 'Male' corresponds to 0, 'Female' corresponds to 1
    discretize_continuous=True
)

# Explaining a single instance from X_test (e.g., the first row)
i = 0 # Index of the instance in X_test to explain
exp = lime_explainer.explain_instance(
    X_test.values[i], # Instance to explain
    model.predict_proba, # Prediction function
    num_features=24 # Number of features to show in explanation
)

# Show the explanation for the single instance
fig = exp.as_pyplot_figure()
plt.savefig(f'xgboost4_LIME_explanation_instance_{i}.png', bbox_inches='tight')
plt.close()

#CATBOOST
file_path = 'merging7.csv' # Replace with your actual file path
df = pd.read_csv(file_path)
# Assuming df is already defined with your dataset
X = df.drop(columns=['Colonoscopy']) # Replace 'resource.gender' with your target column name
y = df['Colonoscopy']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize CatBoost classifier
model = CatBoostClassifier(iterations=1000, depth=6, learning_rate=0.1, loss_function='Logloss', cat_features=None, verbose=200)
# Train the classifier
model.fit(X_train, y_train)

# Make predictions on the testing data
predictions = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')

```

```

f1 = f1_score(y_test, predictions, average='macro')
roc_auc = roc_auc_score(y_test, predictions)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("roc_auc score:", roc_auc)

# SHAP values for CatBoost
explainer = shap.Explainer(model)
shap_values = explainer(X_test)
# Visualize the first prediction's explanation with feature names
# Waterfall plot
plt.figure()
shap.waterfall_plot(shap_values[0], max_display=24)
plt.savefig('catboost4_waterfall_plot.png', bbox_inches='tight')
plt.close()
# Beeswarm plot
plt.figure()
shap.plots.beeswarm(shap_values, max_display=24)
plt.savefig('catboost4_beeswarm_plot.png', bbox_inches='tight')
plt.close()
# Bar plot
plt.figure()
shap.plots.bar(shap_values, max_display=24)
plt.savefig('catboost4_bar_plot.png', bbox_inches='tight')
plt.close()

columns = [
    "Body Height", "Body Mass Index", "Body Weight", "Calcium", "Carbon Dioxide",
    "Chloride", "Glucose",
    "High Density Lipoprotein Cholesterol", "Low Density Lipoprotein Cholesterol",
    "Potassium", "Sodium", "Total Cholesterol", "Triglycerides", "Urea Nitrogen",
    "age", "resource.gender", "Acute viral pharyngitis (disorder)",
    "Respiratory therapy", "Viral sinusitis (disorder)", "emergency", "Td (adult) preservative free",
    "Hypertension"
]

# Loop through each column and generate both SHAP and Partial Dependence plots
for column in columns:
    # SHAP Dependence Plot
    fig = plt.figure(figsize=(8, 6))
    shap.dependence_plot(column, shap_values.values, X_test, interaction_index="age", show=False)
    plt.title(f"SHAP Dependence Plot for {column} (CatBoost)")
    plt.savefig(f'catboost4_SHAP_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()
    # Partial Dependence Plot
    fig, ax = plt.subplots(figsize=(8, 6))
    PartialDependenceDisplay.from_estimator(model, X_test, [column], ax=ax)
    ax.set_title(f"Partial Dependence Plot for {column} (CatBoost)")
    plt.savefig(f'catboost4_Partial_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()

# Adding LIME
lime_explainer = lime_tabular.LimeTabularExplainer(
    X_train.values, # Use the training data for LIME background distribution
    feature_names=X.columns,
    class_names=['No Colonoscopy', 'Colonoscopy'], # Adjusted: 'Male' corresponds to 0, 'Female' corresponds to 1
    discretize_continuous=True
)

```

```

# Explaining a single instance from X_test (e.g., the first row)
i = 0 # Index of the instance in X_test to explain
exp = lime_explainer.explain_instance(
    X_test.values[i], # Instance to explain
    model.predict_proba, # Prediction function
    num_features=24 # Number of features to show in explanation
)

# Show the explanation for the single instance
fig = exp.as_pyplot_figure()
plt.savefig(f'catboost4_LIME_explanation_instance_{i}.png', bbox_inches='tight')
plt.close()# Assuming df is already defined with your dataset
X = df.drop(columns=['Colonoscopy']) # Replace 'resource.gender' with your target column name
y = df['Colonoscopy']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize CatBoost classifier
model = CatBoostClassifier(iterations=1000, depth=6, learning_rate=0.1, loss_function='Logloss', cat_features=None, verbose=200)

# Train the classifier
model.fit(X_train, y_train)
# Make predictions on the testing data
predictions = model.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')
roc_auc = roc_auc_score(y_test, predictions)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("roc_auc score:", roc_auc)

# SHAP values for CatBoost
explainer = shap.Explainer(model)
shap_values = explainer(X_test)
# Visualize the first prediction's explanation with feature names
# Waterfall plot
plt.figure()
shap.waterfall_plot(shap_values[0], max_display=24)
plt.savefig('catboost4_waterfall_plot.png', bbox_inches='tight')
plt.close()
# Beeswarm plot
plt.figure()
shap.plots.beeswarm(shap_values, max_display=24)
plt.savefig('catboost4_beeswarm_plot.png', bbox_inches='tight')
plt.close()
# Bar plot
plt.figure()
shap.plots.bar(shap_values, max_display=24)
plt.savefig('catboost4_bar_plot.png', bbox_inches='tight')
plt.close()

columns = [
    "Body Height", "Body Mass Index", "Body Weight", "Calcium", "Carbon Dioxide",
    "Chloride", "Glucose",
    "High Density Lipoprotein Cholesterol", "Low Density Lipoprotein Cholesterol",
    "Potassium", "Sodium", "Total Cholesterol", "Triglycerides", "Urea Nitrogen",
    "age", "resource.gender", "Acute viral pharyngitis (disorder)",
    "Respiratory therapy", "Viral sinusitis (disorder)", "emergency", "Td (adult) preservative free",
    "Hypertension"
]

```

```

# Loop through each column and generate both SHAP and Partial Dependence plots
for column in columns:
    # SHAP Dependence Plot
    fig = plt.figure(figsize=(8, 6))
    shap.dependence_plot(column, shap_values.values, X_test, interaction_index="age", show=False)
    plt.title(f"SHAP Dependence Plot for {column} (CatBoost)")
    plt.savefig(f'catboost4_SHAP_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()

    # Partial Dependence Plot
    fig, ax = plt.subplots(figsize=(8, 6))
    PartialDependenceDisplay.from_estimator(model, X_test, [column], ax=ax)
    ax.set_title(f"Partial Dependence Plot for {column} (CatBoost)")
    plt.savefig(f'catboost4_Partial_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()

# Adding LIME
lime_explainer = lime_tabular.LimeTabularExplainer(
    X_train.values, # Use the training data for LIME background distribution
    feature_names=X.columns,
    class_names=['No Colonoscopy', 'Colonoscopy'], # Adjusted: 'Male' corresponds to 0, 'Female' corresponds to 1
    discretize_continuous=True
)

# Explaining a single instance from X_test (e.g., the first row)
i = 0 # Index of the instance in X_test to explain
exp = lime_explainer.explain_instance(
    X_test.values[i], # Instance to explain
    model.predict_proba, # Prediction function
    num_features=24 # Number of features to show in explanation
)

# Show the explanation for the single instance
fig = exp.as_pyplot_figure()
plt.savefig(f'catboost4_LIME_explanation_instance_{i}.png', bbox_inches='tight')
plt.close()

#XGBOOST
file_path = 'merging7.csv' # Replace with your actual file path
df = pd.read_csv(file_path)
X = df.drop(columns=['Td (adult) preservative free']) # Replace 'target_column' with your target column name
y = df['Td (adult) preservative free']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize XGBoost classifier
model = XGBClassifier()
# Train the classifier
model.fit(X_train, y_train)
# Make predictions on the testing data
predictions = model.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')
roc_auc = roc_auc_score(y_test, predictions)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("roc_auc score:", roc_auc)

explainer = shap.Explainer(model, X_train)
shap_values = explainer(X_test)

```

```

# Visualize the first prediction's explanation with feature names
# Waterfall plot
plt.figure()
shap.waterfall_plot(shap_values[0], max_display=24)
plt.savefig('xgboost5_waterfall_plot.png', bbox_inches='tight')
plt.close()
# Beeswarm plot
plt.figure()
shap.plots.beeswarm(shap_values, max_display=24)
plt.savefig('xgboost5_beeswarm_plot.png', bbox_inches='tight')
plt.close()
# Bar plot
plt.figure()
shap.plots.bar(shap_values, max_display=24)
plt.savefig('xgboost5_bar_plot.png', bbox_inches='tight')
plt.close()

columns = [
    "Body Height", "Body Mass Index", "Body Weight", "Calcium", "Carbon Dioxide",
    "Chloride", "Glucose",
    "High Density Lipoprotein Cholesterol", "Low Density Lipoprotein Cholesterol",
    "Potassium", "Sodium", "Total Cholesterol", "Triglycerides", "Urea Nitrogen",
    "age", "resource.gender", "Acute viral pharyngitis (disorder)",
    "Respiratory therapy", "Viral sinusitis (disorder)", "emergency", "Colonoscopy",
    "Hypertension"
]

# Loop through each column and generate both SHAP and Partial Dependence plots
for column in columns:
    # SHAP Dependence Plot
    fig = plt.figure(figsize=(8, 6))
    shap.dependence_plot(column, shap_values.values, X_test, interaction_index="age", show=False)
    plt.title(f"SHAP Dependence Plot for {column} (XGBoost)")
    plt.savefig(f'xgboost5_SHAP_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()

    # Partial Dependence Plot
    fig, ax = plt.subplots(figsize=(8, 6))
    PartialDependenceDisplay.from_estimator(model, X_test, [column], ax=ax)
    ax.set_title(f"Partial Dependence Plot for {column} (XGBoost)")
    plt.savefig(f'xgboost5_Partial_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()

# Adding LIME
lime_explainer = lime_tabular.LimeTabularExplainer(
    X_train.values, # Use the training data for LIME background distribution
    feature_names=X.columns,
    class_names=['No Td vax', 'Td vax'], # Adjusted: 'Male' corresponds to 0, 'Female' corresponds to 1
    discretize_continuous=True
)

# Explaining a single instance from X_test (e.g., the first row)
i = 0 # Index of the instance in X_test to explain
exp = lime_explainer.explain_instance(
    X_test.values[i], # Instance to explain
    model.predict_proba, # Prediction function
    num_features=24 # Number of features to show in explanation
)

# Show the explanation for the single instance
fig = exp.as_pyplot_figure()
plt.savefig(f'xgboost5_LIME_explanation_instance_{i}.png', bbox_inches='tight')
plt.close()

#CATBOOST
file_path = 'merging7.csv' # Replace with your actual file path
df = pd.read_csv(file_path)
X = df.drop(columns=['Td (adult) preservative free']) # Replace 'target_column' with your target column name
y = df['Td (adult) preservative free']

```

```

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize CatBoost classifier
model = CatBoostClassifier(iterations=1000, depth=6, learning_rate=0.1, loss_function='Logloss', cat_features=None, verbose=200)

# Train the classifier
model.fit(X_train, y_train)

# Make predictions on the testing data
predictions = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')
roc_auc = roc_auc_score(y_test, predictions)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("roc_auc score:", roc_auc)

# SHAP values for CatBoost
explainer = shap.Explainer(model)
shap_values = explainer(X_test)

# Visualize the first prediction's explanation with feature names
# Waterfall plot
plt.figure()
shap.waterfall_plot(shap_values[0], max_display=24)
plt.savefig('catboost5_waterfall_plot.png', bbox_inches='tight')
plt.close()

# Beeswarm plot
plt.figure()
shap.plots.beeswarm(shap_values, max_display=24)
plt.savefig('catboost5_beeswarm_plot.png', bbox_inches='tight')
plt.close()

# Bar plot
plt.figure()
shap.plots.bar(shap_values, max_display=24)
plt.savefig('catboost5_bar_plot.png', bbox_inches='tight')
plt.close()

columns = [
    "Body Height", "Body Mass Index", "Body Weight", "Calcium", "Carbon Dioxide",
    "Chloride", "Glucose",
    "High Density Lipoprotein Cholesterol", "Low Density Lipoprotein Cholesterol",
    "Potassium", "Sodium", "Total Cholesterol", "Triglycerides", "Urea Nitrogen",
    "age", "resource.gender", "Acute viral pharyngitis (disorder)",
    "Respiratory therapy", "Viral sinusitis (disorder)", "emergency", "Colonoscopy",
    "Hypertension"
]

# Loop through each column and generate both SHAP and Partial Dependence plots
for column in columns:
    # SHAP Dependence Plot
    fig = plt.figure(figsize=(8, 6))
    shap.dependence_plot(column, shap_values.values, X_test, interaction_index="age", show=False)
    plt.title(f"SHAP Dependence Plot for {column} (CatBoost)")
    plt.savefig(f'catboost5_SHAP_dependence_plot_{column}.png', bbox_inches='tight')
    plt.close()

```

```

# Partial Dependence Plot
fig, ax = plt.subplots(figsize=(8, 6))
PartialDependenceDisplay.from_estimator(model, X_test, [column], ax=ax)
ax.set_title(f"Partial Dependence Plot for {column} (CatBoost)")
plt.savefig(f'catboost5_Partial_dependence_plot_{column}.png', bbox_inches='tight')
plt.close()

# Adding LIME
lime_explainer = lime_tabular.LimeTabularExplainer(
    X_train.values, # Use the training data for LIME background distribution
    feature_names=X.columns,
    class_names=['No Td vax', 'Td vax'], # Adjusted: 'Male' corresponds to 0, 'Female' corresponds to 1
    discretize_continuous=True
)

# Explaining a single instance from X_test (e.g., the first row)
i = 0 # Index of the instance in X_test to explain
exp = lime_explainer.explain_instance(
    X_test.values[i], # Instance to explain
    model.predict_proba, # Prediction function
    num_features=24 # Number of features to show in explanation
)

# Show the explanation for the single instance
fig = exp.as_pyplot_figure()
plt.savefig(f'catboost5_LIME_explanation_instance_{i}.png', bbox_inches='tight')
plt.close()

```

```

#PREDIABETES/DIABETES MODEL
import pandas as pd
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
import shap
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import mean_squared_error, accuracy_score, precision_score, f1_score, recall_score, roc_auc_score, max_error
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder
import xgboost as xgb
from lime.lime_tabular import LimeTabularExplainer
import matplotlib.pyplot as plt
import lime
import lime.lime_tabular
import matplotlib.pyplot as plt
import numpy as np
import joblib

file_path = 'merge_observations.csv'
df = pd.read_csv(file_path)

df.drop(columns=['resource.code.coding', 'resource.id.x', 'resource.valueQuantity.unit', 'resource.code.code', 'resource.status'], inplace=True)
df['resource.gender'] = df['resource.gender'].replace({'male': 0, 'female': 1}).astype(int)
df.rename(columns={'resource.code.display.x': 'observations'}, inplace=True)
df.rename(columns={'resource.code.display.y': 'conditions'}, inplace=True)
df.rename(columns={'resource.gender': 'gender'}, inplace=True)
df.rename(columns={'resource.valueQuantity.value': 'observation_value'}, inplace=True)
df.rename(columns={'resource.subject.reference': 'id'}, inplace=True)
df.dropna(inplace=True)

unique_classes = df['observations'].unique()
# Display the unique classes
print(unique_classes)

unique_counts = df['conditions'].value_counts()
print(unique_counts)

n_observations = 16
top_observations = df["observations"].value_counts().index[:n_observations]
df2 = df[df["observations"].isin(top_observations)]
n_conditions = 20
top_conditions = df["conditions"].value_counts().index[:n_conditions]
df2 = df2[df2["conditions"].isin(top_conditions)]
df2.head(16)

df2 = df2[df2["observations"].isin(['Body Weight', 'Glucose', 'Total Cholesterol', 'Triglycerides', 'Low Density Lipoprotein Cholesterol', 'High Density Lipoprotein Cholesterol'])]
df2 = df2[df2["conditions"].isin(['Prediabetes', 'Diabetes'])]

unique_classes = df2['conditions'].unique()
print(unique_classes)
unique_classes2 = df2['observations'].unique()
print(unique_classes2)
num_rows = len(df2)
# Using shape attribute
num_rows = df2.shape[0]
print(f"The DataFrame has {num_rows} rows.")

unique_counts = df2['observations'].value_counts()
print(unique_counts)

```

```

# Aggregate duplicate entries
df2_agg = df2.groupby(['id', 'observations']).agg({'observation_value': 'first', 'age': 'first', 'conditions': 'first', 'gender': 'first'}).reset_index()
# Pivot the DataFrame
pivot_df = df2_agg.pivot(index='id', columns='observations', values='observation_value').reset_index()
# Merge the pivoted DataFrame with the original one
merged_df = df2_agg[['id', 'age', 'gender', 'conditions']].merge(pivot_df, on='id')

# Drop the 'observation_value' column (optional, if you want to remove it)
merged_df.drop(columns=['observation_value'], inplace=True)
merged_df.dropna(inplace=True)
merged_df = merged_df.drop_duplicates()
# Now 'age' and 'conditions' columns will be retained in merged_df
merged_df.to_csv("pivoted2.csv", index=False)
nan_counts = merged_df.isna().sum()
print(nan_counts)
merged_df["conditions"] = merged_df["conditions"].replace({'Prediabetes': 0, 'Diabetes': 1}).astype(int)

X = merged_df.drop(columns=['id', 'conditions'])
y = merged_df['conditions']
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize XGBoost classifier
model = XGBClassifier()
# Train the classifier
model.fit(X_train, y_train)
# Make predictions on the testing data
predictions = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')
roc_auc = roc_auc_score(y_test, predictions)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("roc_auc score:", roc_auc)

explainer = shap.Explainer(model, X_train)
shap_values = explainer(X_test)
# Visualize the first prediction's explanation with feature names
shap.plots.waterfall(shap_values[0])
shap.plots.beeswarm(shap_values)
shap.plots.bar(shap_values)
fig = shap.plots.scatter(shap_values[:, "age"], color=shap_values)
#fig = shap.plots.scatter(shap_values[:, "observation_value"], color=shap_values)
figure = plt.figure()
shap.summary_plot(shap_values, X_test)
fig = shap.plots.scatter(shap_values[:, "gender"], color=shap_values)
# Save the model
joblib.dump(model, 'model1.pkl')

from sklearn.preprocessing import LabelEncoder
# Define Features and Target Variable
X = df2.drop(columns=['id', 'date', 'observations', 'conditions']) # Features
y = df2['conditions'] # Target variable

# Convert categorical target labels to numeric labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
# Split Data into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
# Initialize and Train XGBoost classifier
model2 = XGBClassifier()
model2.fit(X_train, y_train)

```

```

# Make Predictions
predictions = model2.predict(X_test)
# Evaluate Model Performance
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')
# For roc_auc_score, use predict_proba instead of predict (assuming multi-class classification)
try:
    roc_auc = roc_auc_score(y_test, model2.predict_proba(X_test), multi_class='ovr')
except ValueError as e:
    print(f"ROC AUC calculation error: {e}")
    roc_auc = None # In case of an issue with roc_auc calculation

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
if roc_auc is not None:
    print("ROC AUC Score:", roc_auc)
# Initialize LIME Explainer
explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values,
                                                    mode='classification',
                                                    feature_names=X_train.columns.tolist(),
                                                    class_names=label_encoder.classes_.tolist(), # Use the encoded class names
                                                    discretize_continuous=True)

# Choose a random instance for explanation
idx = np.random.randint(0, X_test.shape[0])
exp = explainer.explain_instance(X_test.iloc[idx].values, model2.predict_proba)
# Visualize explanation
exp.show_in_notebook(show_table=True, show_all=False)
# Save the model
joblib.dump(model2, 'model2.pkl')

import pandas as pd
from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, f1_score, recall_score, roc_auc_score
import joblib
import shap

# Load data
file_path = 'merge_observations.csv'
df = pd.read_csv(file_path)

# Preprocess data
df.drop(columns=['resource.code.coding', 'resource.id_x', 'resource.valueQuantity.unit', 'resource.code.code', 'resource.status'], inplace=True)
df["resource.gender"] = df["resource.gender"].replace({'male': 0, 'female': 1}).astype(int)
df.rename(columns={'resource.code.display_x': 'observations'}, inplace=True)
df.rename(columns={'resource.code.display_y': 'conditions'}, inplace=True)
df.rename(columns={'resource.gender': 'gender'}, inplace=True)
df.rename(columns={'resource.valueQuantity.value': 'observation_value'}, inplace=True)
df.rename(columns={'resource.subject.reference': 'id'}, inplace=True)
df.dropna(inplace=True)

# Filter top observations and conditions
df2 = df[df["observations"].isin(['Body Weight', 'Glucose', 'Total Cholesterol', 'Triglycerides',
                                'Low Density Lipoprotein Cholesterol', 'High Density Lipoprotein Cholesterol'])]
df2 = df2[df2["conditions"].isin(['Prediabetes', 'Diabetes'])]

# Aggregate and pivot the data
df2_agg = df2.groupby(['id', 'observations']).agg({'observation_value': 'first', 'age': 'first',
                                                  'conditions': 'first', 'gender': 'first'}).reset_index()

pivot_df = df2_agg.pivot(index='id', columns='observations', values='observation_value').reset_index()

```

```

# Merge pivoted data with the original data
merged_df = df2_agg[['id', 'age', 'gender', 'conditions']].merge(pivot_df, on='id')
merged_df = merged_df.drop_duplicates()
merged_df.to_csv("pivoted2.csv", index=False)

# Check for NaN values
nan_counts = merged_df.isna().sum()
print(nan_counts)

# Prepare data for classification
merged_df["conditions"] = merged_df["conditions"].replace({'Prediabetes': 0, 'Diabetes': 1}).astype(int)

X = merged_df.drop(columns=['id', 'conditions'])
y = merged_df['conditions']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize CatBoost classifier
model = CatBoostClassifier(verbose=0) # Set verbose to 0 to disable output during training

# Train the classifier
model.fit(X_train, y_train)

# Make predictions on the testing data
predictions = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')
roc_auc = roc_auc_score(y_test, predictions)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("roc_auc score:", roc_auc)

# Explainer and SHAP visualization
explainer = shap.Explainer(model, X_train)
shap_values = explainer(X_test)

# Visualize the first prediction's explanation with feature names
shap.plots.waterfall(shap_values[0])
shap.plots.beeswarm(shap_values)
shap.plots.bar(shap_values)
shap.summary_plot(shap_values, X_test)
fig=shap.plots.scatter(shap_values[:, "age"], color=shap_values)
#fig=shap.plots.scatter(shap_values[:, "observation_value"], color=shap_values)
figure = plt.figure()
shap.summary_plot(shap_values, X_test)
fig=shap.plots.scatter(shap_values[:, "gender"], color=shap_values)

# Save the model
joblib.dump(model, 'catboost_model.pkl')

import streamlit as st

import joblib
model = joblib.load('model11.pkl')

def main():
    st.title("Diabetes/Prediabetes Diagnosis")

    # Input form for user data
    st.header("Enter the following attributes:")
    age = st.number_input("Age", min_value=0, max_value=100, step=1)
    gender = st.selectbox("Gender", ("Male", "Female"))
    body_weight = st.number_input("Body Weight (kg)", min_value=30.0, max_value=200.0, step=0.1)
    glucose = st.number_input("Glucose (mg/dl)", min_value=70.0, max_value=200.0, step=0.01)
    high_chol = st.number_input("High Density Lipoprotein Cholesterol (mg/dl)", min_value=30.0, max_value=100.0, step=1.0)
    low_chol = st.number_input("Low Density Lipoprotein Cholesterol (mg/dl)", min_value=50.0, max_value=190.0, step=0.01)
    triglycerides = st.number_input("Triglycerides (mg/dl)", min_value=50.0, max_value=500.0, step=0.01)

    # Make prediction
    if st.button("Predict"):
        # Prepare input data
        input_data = [
            [age, gender, body_weight, glucose, high_chol, low_chol, triglycerides]
        ]

        # Convert categorical attributes to numerical
        input_data[0][1] = 1 if input_data[0][1] == "Female" else 0
        print(input_data[0][1])
        # Predict
        prediction = model.predict(input_data)

        # Display prediction result
        if prediction[0] == 0:
            st.success("The person with the given details has Prediabetes.")
        elif prediction[0] == 1:
            st.error("The person with the given details has Diabetes")

if __name__ == '__main__':
    main()

```