ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Διπλωματική Εργασία

ΕΦΑΡΜΟΓΗ ΑΛΓΟΡΙΘΜΩΝ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΓΙΑ ΤΗΝ ΟΜΑΔΟΠΟΙΗΣΗ ΠΕΛΑΤΩΝ

του

ΜΟΣΧΟΤΟΓΛΟΥ ΣΤΥΛΙΑΝΟΥ

Επιβλέπων

ΚΑΘΗΓΗΤΗΣ ΤΑΡΑΜΠΑΝΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

Υποβλήθηκε ως απαιτούμενο για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στα Πληροφοριακά Συστήματα

Θεσσαλονίκη,

Μάρτιος 2024

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή Ταραμπάνη Κωνσταντίνο που δέχθηκε να αναλάβει την επίβλεψη της διπλωματικής μου εργασίας, για τις πολύτιμες και χρήσιμες γνώσεις που αποκόμισα στο πεδίο της Επιστήμης των Δεδομένων κατά την διάρκεια των διαλέξεων του και κατά την διάρκεια εκπλήρωσης της εργασίας αυτής, αλλά και για την συνεχή υποστήριξη και εμπιστοσύνη που έλαβα καθ' όλη την διάρκεια των μεταπτυχιακών σπουδών μου.

Επίσης θα ήθελα να ευχαριστήσω την οικογένειά μου, για την καθημερινή προσφορά τους και τον αδιάκοπο αγώνα τους όλα αυτά τα χρόνια.

ΠΕΡΙΛΗΨΗ

Η μηχανική μάθηση αποτελεί σημαντικό κεφάλαιο των τεχνολογικών επιστημών τα τελευταία χρόνια. Καθώς ο όγκος των δεδομένων συνεχώς αυξάνεται, η ανάγκη για μελέτη τους καθίσταται αναγκαία για την εξαγωγή χρήσιμων πληροφοριών. Ως βασικός πυλώνας της τεχνητής νοημοσύνης, η μηχανική μάθηση μπορεί να εφαρμοστεί σε πάρα πολλούς κλάδους και να συνεισφέρει αποτελεσματικά στην ποιότητα των εργασιών.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η διερεύνηση και εφαρμογή αλγορίθμων μηχανικής μάθησης, η αξιολόγησή τους και έπειτα η πρακτική εφαρμογή του πιο κατάλληλου και αποδοτικού μοντέλου, η οποία θα συμβάλλει στην αποτελεσματικότερη ταξινόμηση πελατών σε επιχειρήσεις. Αρχικά, μια λεπτομερή επισκόπηση των βασικών αρχών και τεχνικών της μηχανικής μάθησης παρουσιάζεται, εστιάζοντας σε δημοφιλείς αλγορίθμους όπως Logistic Regression, Decision Trees, Random Forest, K-Nearest Neighbors κι ο αλγόριθμος XGBoost. Στη συνέχεια, επικεντρώνεται στην πρακτική εφαρμογή αυτών των μεθόδων, με την ανάπτυξη ενός μοντέλου μηχανικής μάθησης για μια συγκεκριμένη μελέτη περίπτωσης, αυτή της κατηγοριοποίησης.

Τέλος, παρέχει μια συνολική αξιολόγηση της αποτελεσματικότητας του μοντέλου, με σκοπό την επιλογή του κατάλληλου αλγορίθμου. Με βάση τα αποτελέσματα ανάλυσης των ανωτέρων αλγορίθμων, ο αλγόριθμος XGBoost απέδωσε καλύτερα, λαμβάνοντας υπόψη όλες τις μετρικές αξιολόγησης ενός μοντέλου, όπως είναι η Ορθότητα, η Ακρίβεια, η Ανάκληση και η μετρική F1-Score. Με τις κατάλληλες τεχνικές παραμετροποίησης των υπερπαραμέτρων του αλγορίθμου είναι φυσικό πως το μοντέλο μπορεί να παρουσιάζει καλύτερα αποτελέσματα. Ως εκ τούτου, το συγκεκριμένο μοντέλο θεωρείται ένα μοντέλο με μεγάλη και ικανή προβλεπτική ισχύ.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ ΠΙΝΑΚΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ ΕΞΙΣΩΣΕΩΝ

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

Στον σύγχρονο κόσμο της επιχειρηματικότητας, η δυνατότητα να προβλέπει κανείς την αποχώρηση πελατών αποτελεί μια κρίσιμη πτυχή για τη διατήρηση της ανταγωνιστικότητας και την ενίσχυση της βιωσιμότητας μιας επιχείρησης. Η έγκαιρη ανίχνευση και πρόβλεψη της πιθανής αποχώρησης πελατών επιτρέπει στις επιχειρήσεις να αναλάβουν προληπτικές δράσεις, να βελτιώσουν την εμπειρία του πελάτη και να εφαρμόσουν πιο αποδοτικές στρατηγικές διατήρησης τους. Σε αυτό το πλαίσιο, οι αλγόριθμοι μηχανικής μάθησης προσφέρουν ένα ισχυρό εργαλείο για την εξαγωγή πολύτιμων πληροφοριών από μεγάλα σύνολα δεδομένων, παρέχοντας τη δυνατότητα να προβλέψουν με σημαντική ακρίβεια την αποχώρηση πελατών.

Η παρούσα διπλωματική εργασία επικεντρώνεται στην εφαρμογή και την αξιολόγηση διαφόρων αλγορίθμων μηχανικής μάθησης για την πρόβλεψη της αποχώρησης πελατών σε εταιρείες τηλεπικοινωνιών. Συγκεκριμένα, αναλύουμε διάφορες τεχνικές και μοντέλα μηχανικής μάθησης, όπως τον την Λογιστική Παλινδρόμηση, τον αλγόριθμο Τυχαίου Δάσους, τον αλγόριθμο XGBoost, το Δέντρο Απόφασης και τον αλγόριθμο Κ-Κοντινότεροι Γείτονες, με σκοπό την εύρεση της πιο αποτελεσματικής μεθόδου για την περίπτωση μας.

Στο πρώτο μέρος πραγματοποιείται βιβλιογραφική ανασκόπηση, εξετάζοντας τους τύπους επιβλεπόμενης, μη επιβλεπόμενης κι ενισχυτικής μάθησης, ενώ πραγματοποιείται αναφροά των σημαντικότερων αλγορίθμων για προβλήματα ταξινόμησης όπως ο αλγόριθμος λογιστικής παλινδρόμησης (logistic regression), ο αλγόριθμος XGBoost (Extreme Gradient Boosting), ο αλγόριθμος τυχαίου δάσους (random forest), ο αλγόριθμος δέντρου απόφασης (decision tree) καθώς και ο αλγόριθμος Κ-Κοντινότεροι Γείτονες (K-Nearest Neighbors). Έπειτα γίνεται αναφορά στις μετρικές αξιολόγησης που χρησιμοποιούνται για την μέτρηση της απόδοσης ενός μοντέλου, αλλά και την ροή εργασιών που ακολουθείται για τη δημιουργία ενός μοντέλου. Η ροή εργασιών συνήθως περιλαμβάνει την προ-επεξεργασία των δεδομένων, συμπεριλαμβανομένων των διαδικασιών για τον καθαρισμό των δεδομένων από ελλείπουσες, διπλότυπες κι ακραίες τιμές, την επιλογή των χαρακτηριστικών, την κωδικοποίηση και την κλιμάκωση των τιμών και τον διαχωρισμό των δεδομένων σε δεδομένα εκπαίδευσης και δοκιμής. Στη συνέχεια γίνεται αναφορά στις υπερπαραμέτρους των αλγορίθμων και στις

μεθόδους επιλογής των κατάλληλων τιμών με σκοπό την βελτίωση της απόδοσης ενός μοντέλου. Τεχνικές όπως τυχαία αναζήτηση, αναζήτηση πλέγματος, μη αυτόματη αναζήτηση και Μπεϋζιανή ρύθμιση περιγράφονται στην ενότητα αυτή.

Στο δεύτερο μέρος της διπλωματικής εργασίας, γίνεται περιγραφή της μεθοδολογίας που ακολουθήθηκε, προκειμένου να δημιουργήσουμε μοντέλα μηχανικής μάθησης αξιοποιώντας τους αλγορίθμους ταξινόμησης που αναφέρθηκαν προηγουμένως. Γίνεται αναφορά στο περιβάλλον που χρησιμοποιήθηκε για την ανάπτυξη της έρευνας, καθώς και στα εργαλεία που αξιοποιήθηκαν. Εργαλεία όπως η γλώσσα προγραμματισμού Python έπαιξαν κυρίαρχο ρόλο σε όλα τα στάδια της μεθοδολογίας. Ταυτόχρονα, γίνεται περιγραφή του τρόπου επεξεργασίας των δεδομένων, από την συλλογή αυτών, έως και την κατάλληλη επεξεργασία και προετοιμασία τους, ώστε να μπορούν να είναι αξιοποιήσιμα από τα μοντέλα. Στο τελευταίο σημείο της μεθοδολογίας περιγράφεται η διερευνητική ανάλυση των δεδομένων που έγινε για την καλύτερη κατανόησή τους, αλλά και ο τρόπος δημιουργίας κι αξιολόγησης της απόδοσης των μοντέλων. Μετρικές όπως η ορθότητα (accuracy), f1-score, ακρίβεια (precision) και ανάκληση (recall) χρησιμοποιήθηκαν για την ανάλυση και αξιολόγηση της προβλεπτικής ισχύς του εκάστοτε μοντέλου.

Έπειτα, στο τρίτο μέρος της διπλωματικής εργασίας, παρουσιάζεται ο τρόπος υλοποίησης για κάθε στάδιο της μεθοδολογίας, αξιοποιώντας όλα τα διαθέσιμα εργαλεία του περιβάλλοντος ανάπτυξης αλλά και τα αποτελέσματα των μοντέλων. Βιβλιοθήκες της γλώσσας προγραμματισμού Python όπως pandas, NumPy, matplotlib, seaborn και scikit-learn βοήθησαν στην παραγωγή χρήσιμων περιγραφικών στατιστικών και οστεοποιήσεων, όπως και στην δημιουργία των μοντέλων και εξαγωγή των αποτελεσμάτων. Στο τέλος του τρίτου μέρους, γίνεται αξιολόγηση όλων των αποτελεσμάτων του κάθε αλγορίθμου, με τη χρήση των μετρικών και των διαγραμμάτων καμπύλης μάθησης. Ο αλγόριθμος XGBoost ήταν αυτός που είχε την καλύτερη απόδοση συγκριτικά με τους υπόλοιπους, με τα διαγράμματα καμπύλης χαρακτηριστικών δέκτη (ROC-Curve), SHAP (SHapley Additive exPlanations), αλλά και του πίνακα σύγχυσης (confusion matrix), να ακολουθούν με σκοπό την περαιτέρω ανάλυση της απόδοσής του.

Στο τελευταίο μέρος, αναφέρονται τα συμπεράσματα που προκύπτουν καθώς και οι τρόποι για την βελτίωση της απόδοσης των μοντέλων. Η δοκιμή διαφόρων τεχνικών όπως τα νευρωνικά δίκτυα, μπορούν να βελτιώσουν την προβλεπτική ισχύ. Ταυτόχρονα, η απόδοση των μοντέλων μπορεί να βελτιωθεί με τεχνικές υψηλής υπερπαραμετροποίησης, κάτι που συνιστά μεγάλη υπολογιστική ισχύ.

ΚΕΦΑΛΑΙΟ 2: ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΑΝΑΣΚΟΠΗΣΗ

2.1. ΟΡΙΣΜΟΣ ΤΗΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Η μηχανική μάθηση μπορεί να οριστεί ως η διαδικασία εκμάθησης μιας μηχανής ώστε να σκέφτεται ως άνθρωπος για να εκτελέσει μια συγκεκριμένη εργασίας χωρίς να προγραμματίζεται [1]. Η μηχανική μάθηση επιτρέπει στους αλγόριθμους που αξιοποιεί να μαθαίνουν μόνοι τους χωρίς την ανθρώπινη παρέμβαση [2]. Στόχος είναι η δημιουργία αλγορίθμων που μπορούν να χειριστούν δεδομένα εισόδου για την πρόβλεψη μιας εξόδου ενώ νέα δεδομένα γίνονται συνεχώς διαθέσιμα. Στη μηχανική μάθηση, τα δεδομένα εισόδου αναφέρονται ως «χαρακτηριστικά» ενώ τα δεδομένα εξόδου αναφέρονται ως «στόχος» [3]. Αυτά τα σύνολα δεδομένων μπορεί να είναι τεράστια και να αποτελούνται από εκατομμύρια δεδομένα που δεν μπορούν να επεξεργαστούν μόνο από το ανθρώπινο μυαλό [4].

2.2. ΜΕΛΕΤΕΣ ΠΕΡΙΠΤΩΣΗΣ

Η ανάπτυξη μεθόδων μηχανικής μάθησης (machine learning) στις τηλεπικοινωνίες έχει οδηγήσει σε μια ποικιλία καινοτόμων εφαρμογών και πρακτικών, οι οποίες συμβάλλουν στη βελτίωση της λειτουργικής αποδοτικότητας, την εμπειρία των πελατών και την απόδοση του δικτύου. Παρακάτω, αναλύονται ορισμένες μελέτες περίπτωσης περιγράφοντας λεπτομερώς τις μεθόδους που χρησιμοποιήθηκαν και τα αποτελέσματα που επιτεύχθηκαν.

Οι Joolfoo et al. (2020) πρότειναν μια υβριδική προσέγγιση που συνδυάζει τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Network – ANN) και K-Nearest Neighbors (KNN) για την πρόβλεψη της αποχώρησης από εταιρείες τηλεπικοινωνιών. Αυτή η καινοτόμος προσέγγιση αποσκοπούσε στην αξιοποίηση των πλεονεκτημάτων και των δύο αλγορίθμων για τη βελτίωση της ακρίβειας της πρόβλεψης. Το τελικό αποτέλεσμα της έρευνας δεν παρουσιάστηκε λεπτομερώς, ωστόσο, η συγκεκριμένη έρευνα υπογράμμισε τις δυνατότητες των υβριδικών μοντέλων για τη βελτίωση των δυνατοτήτων πρόβλεψης της αποχώρησης στον τομέα των τηλεπικοινωνιών.

Στο έργο τους οι De και Prabu (2022) με τη βιβλιογραφική ανασκόπηση που διεξήγαγαν, ανέλυσαν 38 πρωτογενείς μελέτες που δημοσιεύθηκαν μεταξύ του 2018 και 2021, εντοπίζοντας δημοφιλείς τεχνικές μηχανικής μάθησης και προτείνοντας νέες κατευθύνσεις για μελλοντική έρευνα. Η ανασκόπηση τόνισε την έλλειψη ικανότητας γενίκευσης σε διάφορους βιομηχανικούς τομείς και υπογράμμισε τις δυνατότητες ενσωμάτωσης χαρακτηριστικών ανάλυσης κοινωνικών δικτύων στα μοντέλα churn. Επισημάνθηκε επίσης ο υποεξερευνημένος τομέας της αξιοποίησης των δεδομένων αλληλεπίδρασης μεταξύ πελατών και εταιρειών για την πρόβλεψη της αποχώρησης [49]. Οι De και Pradu (2022) επισημαίνουν μια αυξανόμενη τάση στην έρευνα για την πρόβλεψη της αποχώρησης, η οποία περιλαμβάνει την ενσωμάτωση χαρακτηριστικών ανάλυσης κοινωνικών δικτύων. Η συγκεκριμένη προσέγγιση αξιοποιεί δεδομένα σχετικά με τις κοινωνικές συνδέσεις και αλληλεπιδράσεις μεταξύ των πελατών για τη βελτίωση της ακρίβειας των προβλέψεων αποχώρησης. Αυτή η έννοια αντανακλά μια ευρύτερη στροφή προς τη χρήση ολοκληρωμένων συνόλων δεδομένων που περιλαμβάνουν την κοινωνική δυναμική ως παράγοντα πρόβλεψης της συμπεριφοράς των πελατών. Η ανάλυση των κοινωνικών δικτύων υποδεικνύει ότι οι κοινωνικοί παράγοντες μπορούν να επηρεάσουν σημαντικά τη συμπεριφορά αποχώρησης. Τα αποτελέσματα αυτά υποδηλώνουν τη σημασία της συνεκτίμησης των κοινωνικών σχέσεων και των επιδράσεων δικτύου στην ανάπτυξη πιο εξελιγμένων μοντέλων πρόβλεψης της αποχώρησης.

Ο τομέας της πρόβλεψης αποχώρησης στις τηλεπικοινωνίες έχει δει ένα ευρύ φάσμα μελετών που χρησιμοποιήθηκαν διάφορες τεχνικές μηχανικής μάθησης για την ενίσχυση της ακρίβειας και της αποτελεσματικότητας της πρόβλεψης της αποχώρησης των πελατών. Η πρόβλεψη αποχώρησης των πελατών στις τηλεπικοινωνίες με χρήση μηχανικής μάθησης των Patil & Wankhade (2023) εξετάζει το κρίσιμο σημείο της πρόβλεψης αποχώρησης πελατών στις εταιρείες τηλεπικοινωνιών με χρήση τεχνικών μηχανικής μάθησης. Η μελέτη δίνει έμφαση στην ανάλυση ιστορικών δεδομένων πελατών, συμπεριλαμβανομένων δημογραφικών πληροφοριών, προτύπων χρήσης, λεπτομερειών χρέωσης και ιστορικού υπηρεσιών, για την πρόβλεψη μελλοντικών πιθανοτήτων αποχώρησης. Η διαδικασία περιλαμβάνει προ-επεξεργασία δεδομένων, μηχανική χαρακτηριστικών, επιλογή μοντέλου και αξιολόγηση για τη μείωση των ποσοστών αποχώρησης και τη βελτίωση της ικανοποίησης των πελατών [56].

Η πρόβλεψη αποχώρησης πελατών στις τηλεπικοινωνίες των Huang et al. (2012) παρουσιάζει ένα σύνολο νέων χαρακτηριστικών για την πρόβλεψη αποχώρησης πελατών σταθερής τηλεφωνίας, συμπεριλαμβανομένης της τμηματοποίησης, των λεπτομερειών κλήσης και των δημογραφικών χαρακτηριστικών. Η συγκεκριμένη μελέτη εντοπίζει και εφαρμόζει επτά τεχνικές πρόβλεψης, όπως Logistic Regressions, Naive Bayes και Decision Trees μεταξύ άλλων, και διεξάγει συγκριτικά πειράματα για την αξιολόγηση αυτών των μεθόδων για την πρόβλεψη αποχώρησης πελατών. Τα αποτελέσματα της έρευνας δείχνουν την αποτελεσματικότητα των νέων χαρακτηριστικών και τεχνικών μοντελοποίησης στην πρόβλεψη της αποχώρησης στον τομέα των τηλεπικοινωνιακών υπηρεσιών [50].

Η βελτιωμένη πρόβλεψη της αποχώρησης στη βιομηχανία τηλεπικοινωνιών μέσω της ανάλυσης ενός μεγάλου δικτύου επικεντρώνεται στη διατήρηση των πελατών μέσω της πρόβλεψης των πιθανών πελατών που θα αποχωρήσουν, λαμβάνοντας υπόψη τις σχέσεις μεταξύ των πελατών. Η μελέτη των Kim et al. (2012) εισάγει μια νέα διαδικασία για την πρόβλεψη της αποχώρησης εξετάζοντας τα πρότυπα επικοινωνίας και λαμβάνοντας υπόψη μια διαδικασία διάδοσης σε ένα δίκτυο με βάση τα αρχεία λεπτομερειών κλήσεων. Η προτεινόμενη μεθοδολογία βελτιώνει σημαντικά την απόδοση του μοντέλου πρόβλεψης ενσωματώνοντας χαρακτηριστικά κοινωνικού δικτύου και παραδοσιακά προσωπικά χαρακτηριστικά [54].

Η βελτιωμένη πρόβλεψη της αποχώρησης στη βιομηχανία τηλεπικοινωνιών με τη χρήση τεχνικών εξόρυξης δεδομένων των A. Keramati et al. (2014) διερευνά την εφαρμογή των αλγορίθμων Decision Trees, Τεχνητά Νευρωνικά Δίκτυα (ANN), K-Nearest Neighbors (KNN) και Μηχανή Διανυσμάτων Υποστήριξης (SVM) για την πρόβλεψη της αποχώρησης. Χρησιμοποιώντας δεδομένα από μια ιρανική εταιρεία κινητής τηλεφωνίας, η μελέτη συγκρίνει αυτές τις τεχνικές και εισάγει μια υβριδική μεθοδολογία που βελτιώνει σημαντικά τις τιμές των μετρικών αξιολόγησης, επιτυγχάνοντας ακρίβεια άνω του 95% στην Ανάκληση και την Ακρίβεια. Η μεθοδολογία εισάγει επίσης μια νέα προσέγγιση για την εξαγωγή χαρακτηριστικών με επιρροή, υπογραμμίζοντας τη σημασία της επιλογής χαρακτηριστικών στα μοντέλα πρόβλεψης αποχώρησης [53]. Η πρόβλεψη αποχώρησης με διατήρηση του απορρήτου από τους Xu et al. (2009) ασχολείται με την κρίσιμη πτυχή του απορρήτου των πελατών στην πρόβλεψη αποχώρησης. Η μελέτη καταδεικνύει πώς οι τεχνικές παραμόρφωσης δεδομένων μπορούν να

καλύψουν σύνολα δεδομένων πελατών τηλεπικοινωνιών για την πρόβλεψη αποχώρησης, διατηρώντας παράλληλα την ακρίβεια των προβλέψεων. Η προσέγγιση αυτή διασφαλίζει το απόρρητο των πελατών τροποποιώντας σημαντικά τα δεδομένα, αλλά επιτρέπει την αποτελεσματική πρόβλεψη της αποχώρησης μέσω των προτεινόμενων μεθόδων. Η μελέτη συγκρίνει τις επιδόσεις διαφόρων μεθόδων παραμόρφωσης δεδομένων, προσφέροντας πληροφορίες σχετικά με την εξισορρόπηση των ανησυχιών για την προστασία της ιδιωτικής ζωής με την ακρίβεια της πρόβλεψης [57].

Η πρόβλεψη της αποχώρησης στη βιομηχανία τηλεπικοινωνιών με την προσέγγιση των αδρών συνόλων από τους Amin et al. (2015) αξιοποιεί τη θεωρία των αδρών συνόλων, μια τεχνική λήψης αποφάσεων που βασίζεται σε κανόνες, για τη διαμόρφωση μοντέλων πρόβλεψης της αποχώρησης. Πειραματιζόμενοι με διαφορετικούς αλγορίθμους (Exhaustive, Genetic, Covering και LEM2) για τη δημιουργία κανόνων, η μελέτη εντοπίζει ότι η ταξινόμηση του τραχύ συνόλου με βάση τον γενετικό αλγόριθμο παρέχει την καταλληλότερη απόδοση. Η προσέγγιση αυτή υπογραμμίζει τη δυνατότητα χρήσης εναλλακτικών θεωρητικών πλαισίων για την ενίσχυση της ακρίβειας πρόβλεψης της αποχώρησης και προσφέρει στρατηγικές ιδέες στους υπεύθυνους λήψης αποφάσεων στον τομέα των τηλεπικοινωνιών [48].

Η μελέτη των Lin et al (2014) τονίζει τη σημασία της επιλογής χαρακτηριστικών και της μείωσης των δεδομένων ως δύο κρίσιμα βήματα προ-επεξεργασίας δεδομένων στην πρόβλεψη της αποχώρησης. Η παρούσα μελέτη διερευνά τον τρόπο με τον οποίο διαφορετικές προτεραιότητες προ-επεξεργασίας δεδομένων επηρεάζουν την απόδοση ενός μοντέλου τεχνητού νευρωνικού δικτύου που εκπαιδεύεται σε ένα πραγματικό σύνολο δεδομένων αποχώρησης πελατών τηλεπικοινωνιών. Τα ευρήματα υπογραμμίζουν ότι η επιλογή των σωστών τεχνικών προ-επεξεργασίας μπορεί να βελτιώσει σημαντικά την ποιότητα πρόβλεψης του μοντέλου, αναδεικνύοντας τον κρίσιμο ρόλο της προετοιμασίας των δεδομένων στην πρόβλεψη της αποχώρησης [55]. Η πρόβλεψη αποχώρησης στις τηλεπικοινωνίες με τη χρήση τεχνολογίας εξόρυξης δεδομένων των Jadhav και Pawar (2011) αποσκοπεί στη δημιουργία ενός συστήματος υποστήριξης αποφάσεων για την πρόβλεψη αποχώρησης σε μια εταιρεία τηλεπικοινωνιών με τη χρήση τεχνολογίας εξόρυξης δεδομένων. Η παρούσα εργασία περιγράφει τη διαδικασία σχεδιασμού ενός συστήματος υποστήριξης αποφάσεων που είναι ικανό να προβλέπει τη

συμπεριφορά αποχώρησης πελατών αρκετά νωρίτερα. Το προτεινόμενο μοντέλο αντιμετωπίζει το κρίσιμο πρόβλημα της αυξανόμενης αποχώρησης πελατών, το οποίο αποτελεί σημαντική απώλεια εσόδων για τις εταιρείες τηλεπικοινωνιών. Χρησιμοποιώντας προηγμένες τεχνικές εξόρυξης δεδομένων, η παρούσα μελέτη συμβάλλει στην ανάπτυξη εξελιγμένων και προσαρμοσμένων συστημάτων υποστήριξης αποφάσεων για την αποτελεσματική διαχείριση της αποχώρησης [51].

Η διερεύνηση της πρόβλεψης της αποχώρησης στον τομέα των τηλεπικοινωνιών αποκαλύπτει ένα δυναμικό και εξελισσόμενο πεδίο που χαρακτηρίζεται από την εφαρμογή ενός ευρέος φάσματος τεχνικών μηχανικής μάθησης και εξόρυξης δεδομένων. Από τη χρήση παραδοσιακών αλγορίθμων όπως τα δέντρα αποφάσεων, οι μηχανές διανυσμάτων υποστήριξης και τα νευρωνικά δίκτυα έως πιο καινοτόμες προσεγγίσεις, συμπεριλαμβανομένων μεθόδων συνόλου και μοντέλων βαθιάς μάθησης, οι ερευνητικές προσπάθειες αναδεικνύουν μια συνεχή αναζήτηση για τη βελτίωση της ακρίβειας και της αποτελεσματικότητας των μοντέλων πρόβλεψης της αποχώρησης.
Τα βασικά συμπεράσματα από τις μελέτες που συζητήθηκαν περιλαμβάνουν:

- Η αποτελεσματικότητα των υβριδικών μοντέλων που συνδυάζουν διαφορετικές τεχνικές μηχανικής μάθησης για την αξιοποίηση των πλεονεκτημάτων τους και τον μετριασμό των αδυναμιών τους, βελτιώνοντας έτσι την ακρίβεια της πρόβλεψης.
- Η σημασία της επιλογής χαρακτηριστικών και της προ-επεξεργασίας δεδομένων για την ενίσχυση της απόδοσης του μοντέλου, συμπεριλαμβανομένων νέων μεθόδων για την αντιμετώπιση των προκλήσεων των μεγάλων, αραιών και ανισόρροπων συνόλων δεδομένων.
- Το δυναμικό της ανάλυσης κοινωνικών δικτύων και της δυναμικής ανάλυσης συμπεριφοράς για την παροχή βαθύτερης πληροφόρησης σχετικά με τις σχέσεις και τις συμπεριφορές των πελατών, προσφέροντας νέους δρόμους για τον εντοπισμό σημάτων αποχώρησης.
- Η διερεύνηση τεχνικών διατήρησης της ιδιωτικότητας στην πρόβλεψη της αποχώρησης, εξισορροπώντας την ανάγκη για ακριβείς προβλέψεις με την επιτακτική ανάγκη προστασίας των δεδομένων των πελατών.

Αυτές οι προσπάθειες υπογραμμίζουν συλλογικά τον κρίσιμο ρόλο της πρόβλεψης της αποχώρησης, που επιτρέπει στις εταιρείες τηλεπικοινωνιών να σχεδιάζουν αποτελεσματικές στρατηγικές διατήρησης πελατών. Με τον έγκαιρο εντοπισμό των δυνητικών αποχωρήσεων, οι εταιρείες μπορούν να προσαρμόσουν τις παρεμβάσεις για τη διατήρηση αυτών των πελατών, διασφαλίζοντας έτσι τα έσοδά τους και το ανταγωνιστικό τους πλεονέκτημα. Επιπλέον, η συνεχής καινοτομία στις μεθοδολογίες πρόβλεψης της αποχώρησης αναδεικνύει τη σημασία της υιοθέτησης μιας προληπτικής, βασισμένης στα δεδομένα προσέγγισης στη διαχείριση των πελατειακών σχέσεων.

Το εύρος των μεθοδολογιών και το βάθος των γνώσεων που αποκτήθηκαν από αυτές τις μελέτες καταδεικνύουν το πλούσιο δυναμικό για περαιτέρω έρευνα στον τομέα αυτό. Καθώς οι εταιρείες τηλεπικοινωνιών περιηγούνται σε ένα ολοένα και πιο ανταγωνιστικό τοπίο, η ικανότητα πρόβλεψης και πρόληψης της αποχώρησης πελατών θα παραμείνει βασικός παράγοντας της επιτυχίας τους.

2.3. ΚΑΤΗΓΟΡΙΕΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Η μηχανική μάθηση έχει πολλούς αλγορίθμους, τεχνικές και μεθοδολογίες που μπορούν να χρησιμοποιηθούν για τη δημιουργία μοντέλων και για την επίλυση των προβλημάτων [5]. Μπορούμε να κατηγοριοποιήσουμε την μηχανική μάθηση σε τρεις βασικές κατηγορίες:

1. Επιβλεπόμενη Μηχανική Μάθηση (Supervised Machine Learning)
2. Μη Επιβλεπόμενη Μηχανική Μάθηση (Unsupervised Machine Learning)
3. Ενισχυτική μάθηση (Reinforcement Learning)

Κάθε μία από τις κατηγορίες χρησιμοποιείται για συγκεκριμένο σκοπό και τα δεδομένα που χρησιμοποιούνται διαφέρουν μεταξύ τους [6].

Πίνακας 1 *Διαφορές επιβλεπόμενης και μη-επιβλεπόμενης μηχανικής μάθησης [1].*

Επιβλεπόμενη Μάθηση	**Μη Επιβλεπόμενη Μηχανική Μάθηση**
Χρησιμοποιεί δεδομένα με ετικέτα.	Χρησιμοποιεί δεδομένα χωρίς ετικέτα.

Δεν απαιτεί υπερβολικά δεδομένα για ακρίβεια.	Απαιτεί αρκετά δεδομένα για ακρίβεια.
Μικρή υπολογιστική πολυπλοκότητα	Μεγάλη υπολογιστική πολυπλοκότητα
Δεν αναγνωρίζει μοτίβα μόνο του από το σύνολο των δεδομένων	Αναγνωρίζει μοτίβα μόνο του από το σύνολο των δεδομένων.

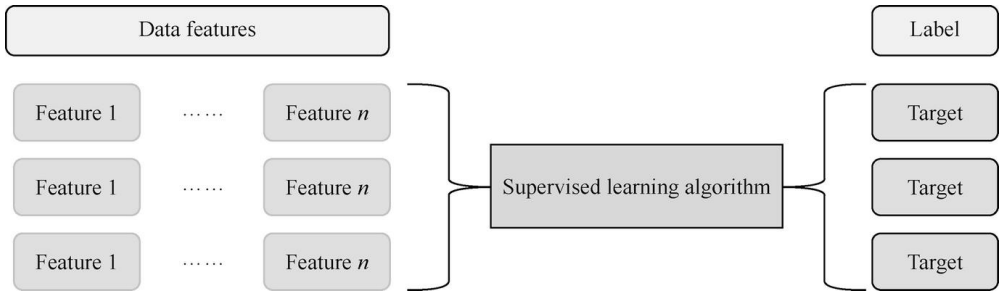
2.3.1. Επιβλεπόμενη Μάθηση (Supervised Learning)

Στόχος της επιβλεπόμενης μάθησης είναι να αντιστοιχήσει τις εισόδους σε εξόδους, όπως φαίνεται στην ακόλουθη εξίσωση [1]:

$$y = f(x)$$

Τα δεδομένα εκπαίδευσης καθοδηγούν την ανάπτυξη του αλγορίθμου. Ένα μοντέλο δημιουργείται μέσω μίας διαδικασίας εκπαίδευσης κατά την οποία το μοντέλο κάνει προβλέψεις και διορθώνεται όταν οι προβλέψεις είναι εσφαλμένες. Η εκπαίδευση συνεχίζεται μέχρι το μοντέλο να επιτύχει το επιθυμητό επίπεδο ακρίβειας στα δεδομένα εκπαίδευσης. Οι αλγόριθμοι της επιβλεπόμενης μάθησης μπορούν να εφαρμόσουν όσα έχουν μάθει σε νέα δεδομένα χρησιμοποιώντας δεδομένα με ετικέτα για να προβλέψουν μελλοντικά γεγονότα [7]. Υπάρχουν αρκετοί αλγόριθμοι που μπορούν να χρησιμοποιηθούν για να λύσουν προβλήματα μηχανικής μάθησης με τη βοήθεια της επιβλεπόμενης μάθησης. Οι αλγόριθμοι αυτοί μπορούν να χωριστούν σε δύο κατηγορίες:

- **Αλγόριθμοι ταξινόμησης (Classification algorithms)**
  Αυτοί οι αλγόριθμοι περιέχουν εξόδους που είναι κατηγορικές. Για παράδειγμα τα χρώματα (πορτοκαλί, μωβ, μπλε), τα συναισθήματα (χαρούμενος/η, λυπημένος/η, θυμωμένος/η), το φύλο (άνδρας, γυναίκα) κλπ.
- **Αλγόριθμοι παλινδρόμησης (Regression algorithms)**
  Αυτοί οι αλγόριθμοι περιέχουν εξόδους που είναι πραγματικές ή μετρήσιμες. Για παράδειγμα το ύψος, η ηλικία, η τιμή κλπ. [1].



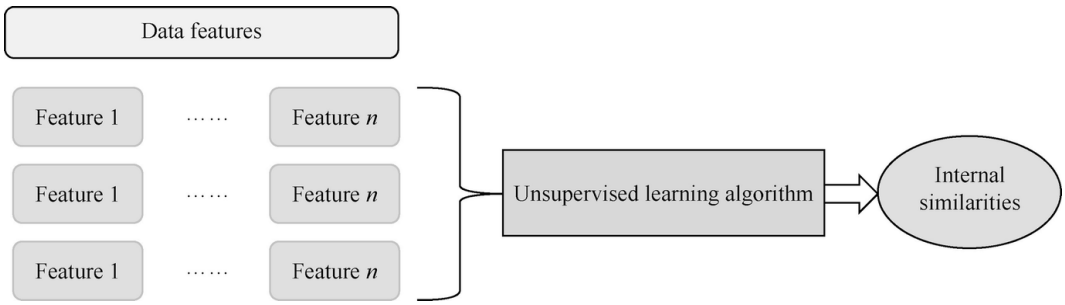
Σχήμα 1 *Επιβλεπόμενη μηχανική μάθηση [9].*

2.3.2. Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)

Στην μη επιβλεπόμενη μάθηση, δεν υπάρχει έξοδος για κάθε είσοδο. Ο αλγόριθμος προσπαθεί να αναγνωρίσει κρυμμένα μοτίβα ή δομές και ομαδοποιήσεις μέσα στα σύνολα δεδομένων [9]. Ο αλγόριθμος δεν έχει προηγούμενη πληροφόρηση για τα μοτίβα και τις δομές αυτές. Υπάρχουν επίσης αρκετοί αλγόριθμοι που μπορούν να χρησιμοποιηθούν για την επίλυση προβλημάτων μηχανικής μάθησης με την βοήθεια της μη επιβλεπόμενης μάθησης. Οι αλγόριθμοι αυτοί χωρίζονται κι αυτοί σε δύο κύριες κατηγορίες:

- **Ανάλυση συστάδων (Cluster analysis)**
  Η προσέγγιση αυτή βρίσκει ομοιότητες μεταξύ των δεδομένων και στη συνέχεια ομαδοποιεί τα κοινά δεδομένα σε συστάδες (clusters).

- **Μείωση διαστάσεων (Dimensionality reduction)**
  Η προσέγγιση αυτή προσπαθεί να μειώσει την πολυπλοκότητα των δεδομένων [1]. Μπορεί να επιτευχθεί μέσω της επιλογής κι εξαγωγής χαρακτηριστικών. Η μείωση διαστάσεων μπορεί να αναπαραστήσει ή να μετασχηματίσει τα δεδομένα [7].



Σχήμα 2 *Μη επιβλεπόμενη μάθηση [9].*

2.3.3. Ενισχυτική Μάθηση (Reinforcement Learning)

Η ενισχυτική μάθηση χρησιμοποιείται κυρίως για την επίλυση προβλημάτων λήψης αποφάσεων πολλαπλών βημάτων, όπως είναι τα βιντεοπαιχνίδια και η οπτική πλοήγηση. Σε αντίθεση με τα προβλήματα που μελετώνται από την επιβλεπόμενη και μη επιβλεπόμενη μάθηση, στα προβλήματα αυτά είναι συχνά δύσκολο να βρεθούν ακριβείς απαντήσεις. Τα πιο σημαντικά μέρη ενός αλγορίθμου ενισχυτικής μάθησης είναι το μοντέλο και το περιβάλλον. Σε διαφορετικά περιβάλλοντα, το μοντέλο μπορεί να καθορίσει τις δικές του ενέργειες και διαφορετικές ενέργειες μπορούν να έχουν διαφορετικές επιπτώσεις στο περιβάλλον [9]. Αυτός ο τύπος εκμάθησης θεωρείται ως η αλληλεπίδραση μεταξύ του περιβάλλοντος κι ενός ατζέντη. Το περιβάλλον επηρεάζει έντονα την απόφαση του ατζέντη. Παίζει τον ρόλο του κριτή, κρίνοντας τη δράση του ατζέντη. Η ενισχυτική μάθηση λειτουργεί εντός ενός συστήματος ανταμοιβής [6]. Ο ατζέντης δέχεται μια είσοδο από το περιβάλλον και δημιουργεί μια δράση ως έξοδο. Για τη δράση αυτή δίνεται μια ανταμοιβή ή μια ποινή. Εάν δοθεί ανταμοιβή, η έξοδος ορίζεται ως επιθυμητή, διαφορετικά, η έξοδος ενημερώνεται [10]. Τα επιμέρους κομμάτια της εξηγούν το σύστημα αυτό:

- **Αυτόνομος ατζέντης (Autonomous agent)**
  Είναι ο κύριος χαρακτήρας αυτής της διαδικασίας μάθησης και είναι υπεύθυνος για τη πράξη ενεργειών. Για παράδειγμα, εάν πρόκειται για παιχνίδι, ο ατζέντης κάνεις τις απαραίτητες κινήσεις για να τερματίσει ή να φτάσει στον τελικό στόχο.
- **Ενέργειες (Actions)**

Πρόκειται για σύνολα πιθανών βημάτων που μπορεί να κάνει ένας ατζέντης για να προχωρήσει στην διαδικασία. Κάθε ενέργεια θα έχει κάποια επίδραση στην κατάσταση του ατζέντη και μπορεί να οδηγήσει είτε σε ανταμοιβή (reward) είτε σε ποινή (penalty). Για παράδειγμα, σε ένα παιχνίδι τένις, οι ενέργειες μπορεί να είναι το σερβίρισμα, η επιστροφή, η μετακίνηση αριστερά η δεξιά κλπ .

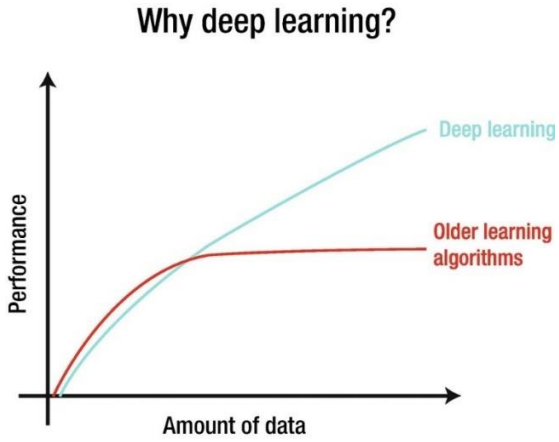
- **Επιβράβευση (Reward)**
  Αυτό είναι το κλειδί της ενισχυτικής μάθησης. Οι ανταμοιβές δίνουν τη δυνατότητα στους ατζέντες να προβαίνουν σε ενέργειες με βάση το εάν πρόκειται για θετικές ανταμοιβές ή ποινές.
- **Περιβάλλον (Environment)**
  Είναι η περιοχή στην οποία μπορεί να ενεργεί ένας ατζέντης. Το περιβάλλον αποφασίζει εάν οι ενέργειες που κάνει ο πράκτορας έχουν ως αποτέλεσμα ανταμοιβές ή ποινές.
- **Κατάσταση (State)**
  Η θέση στην οποία βρίσκεται ο ατζέντης σε οποιαδήποτε δεδομένη χρονική στιγμή ορίζει την κατάστασή του. Για να προχωρήσεις ή να φτάσει στον τελικό στόχο, πρέπει να συνεχίζει να αλλάζει καταστάσεις προς θετική κατεύθυνση για να μεγιστοποιήσει τις ανταμοιβές [6].

Σχήμα 3 *Διαδικασία ενισχυτικής μάθησης [6].*

2.4. ΒΑΘΙΑ ΜΑΘΗΣΗ (DEEP LEARNING)

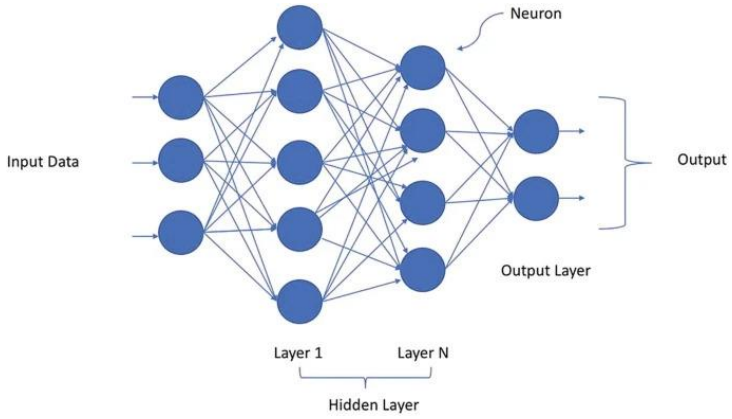
Η βαθιά μάθηση είναι ένα υποσύνολο της μηχανικής μάθησης που ασχολείται με αλγορίθμους εμπνευσμένους από τη βιολογική δομή και λειτουργία ενός εγκεφάλου. Τα μοντέλα βαθιάς μάθησης σχεδιάστηκαν για να χρησιμοποιούν την αρχιτεκτονική των νευρωνικών δικτύων. Ένα

νευρωνικό δίκτυο είναι μια ιεραρχική οργάνωση νευρώνων (παρόμοια με τους νευρώνες του ανθρώπινου εγκεφάλου) με συνδέσεις με άλλους νευρώνες. Αυτοί οι νευρώνες περνούν ένα μήνυμα ή σήμα σε άλλους νευρώνες με βάση την είσοδο και σχηματίζουν ένα πολύπλοκο δίκτυο που μαθαίνει με κάποιο μηχανισμό ανάδρασης [11].



Σχήμα 4 *Επίδοση αλγορίθμων μηχανικής μάθησης και αλγορίθμων βαθιάς μάθησης [12].*

Όπως φαίνεται στην Σχήμα 1.5, τα δεδομένα εισόδου εισέρχονται στο πρώτο στρώμα νευρώνων οι οποίοι στην συνέχεια παρέχουν μια έξοδο για το επόμενο στρώμα. Αυτό συνεχίζεται μέχρι την τελική έξοδο. Κάθε στρώμα μπορεί να έχει έναν ή και περισσότερους νευρώνες, και κάθε ένας από αυτούς κάνει έναν υπολογισμό. Η σύνδεση μεταξύ δύο νευρώνων από διαδοχικά στρώματα περιλαμβάνει ένα σχετικό βάρος. Το βάρος καθορίζει την επίδραση της εισόδου στην έξοδο για τον επόμενο νευρώνα και τελικά για τη συνολική τελική έξοδο.



Σχήμα 5 *Αναπαράσταση νευρωνικού δικτύου [11].*

Ενώ η μηχανική μάθηση λειτουργεί πολύ καλά για μια ποικιλία προβλημάτων, αποτυγχάνει σε ορισμένες περιπτώσεις που φαίνονται πολύ εύκολες για τον άνθρωπό όπως η ταξινόμηση μιας εικόνας ως γάτα ή σκύλο, η διάκριση ενός ήχου ως ανδρικής η γυναικείας φωνής κλπ. Η μηχανική μάθηση δεν έχει καλή απόδοση με εικόνες και άλλους μη δομημένους τύπους δεδομένων.

Σήμερα μπορούμε να αξιοποιήσουμε την βαθιά μάθηση για σχεδόν όλες τις περιπτώσεις χρήσης που είχαν επιλυθεί προηγουμένως με τη χρήση της μηχανικής μάθησης [11]. Πλέον, μπορούμε να δούμε την υιοθέτηση της βαθιάς μάθησης σε πολλές περιπτώσεις όπως τη λειτουργία αυτοοδήγησης στα αυτοκίνητα, τις προβλέψεις της επόμενης λέξης στο σύστημα ανταλλαγής μηνυμάτων του κινητού τηλεφώνου, τους προσωπικούς ψηφιακούς βοηθούς που ανταποκρίνονται ως άνθρωποι με εφαρμογές όπως την αναγνώριση φωνής, την μετάφραση γλωσσών, την δημιουργία εικόνων κλπ. [11, 12].

2.5. ΑΛΓΟΡΙΘΜΟΙ ΕΠΙΒΛΕΠΟΜΕΝΗΣ ΜΑΘΗΣΗΣ

Όπως αναφέρθηκε στην ενότητα 1.2.1., οι αλγόριθμοι της επιβλεπόμενης μάθησης προσπαθούν να αντιστοιχήσουν τις εισόδους σε κάποιες συναφείς εξόδους κατά τη διάρκεια εκπαίδευσης ενός μοντέλου. Ονομάζεται επιβλεπόμενη μάθηση επειδή το μοντέλο μαθαίνει σε δεδομένα όπου οι επιθυμητές ετικέτες εξόδου είναι ήδη γνωστές κατά τη φάση της εκπαίδευσης. Όπως επίσης αναφέρθηκε στο κεφάλαιο, οι αλγόριθμοι της επιβλεπόμενης μάθησης μπορούν να χωριστούν σε δύο κύριες κατηγορίες ανάλογα με το πρόβλημα που καλούνται να επιλύσουν που είναι η κατηγοριοποίηση και η παλινδρόμηση [5].

Στο κεφάλαιο αυτό θα περιγράψουμε μερικούς από τους βασικούς αλγόριθμους που χρησιμοποιούνται στην επιβλεπόμενη μάθηση και για προβλήματα ταξινόμησης όπως είναι η λογιστική παλινδρόμηση, τα δέντρα αποφάσεων, ο ταξινομητής XGBoost, το τυχαίο δέντρο και οι K-κοντινότεροι γείτονες.

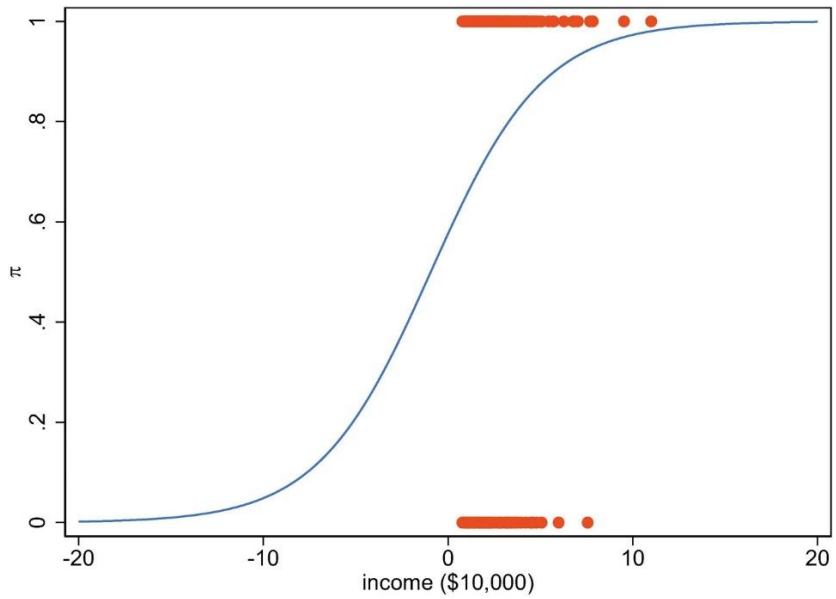
2.5.1. Λογιστική Παλινδρόμηση (Logistic Regression)

Η λογιστική παλινδρόμηση εφευρέθηκε στα τέλη του 1930 από τους στατιστικολόγους Ronald Fisher και Frank Yates [13]. Χρησιμοποιείται κατά την εξέταση της σχέσης μιας κατηγορικής μεταβλητής στόχου και μιας ή περισσότερων μεταβλητών εισόδου. Η μεταβλητή στόχος μπορεί να είναι είτε δυαδική, είτε πολυονομαστική είτε τακτική. Η μεταβλητές εισόδου μπορεί να είναι είτε συνεχείς (αριθμητικές) είτε κατηγορικές. Με λίγα λόγια, η λογιστική παλινδρόμηση παρέχει την πιθανότητα να συμβεί ένα συγκεκριμένο γεγονός προβλέποντας ταυτόχρονα και την αξία του αποτελέσματος. Η φύση της μεταβλητής στόχου (κλάσης/κατηγορίας) καθορίζει ποιος από τους τρεις τύπους λογιστικής παλινδρόμησης θα χρησιμοποιηθεί, δυαδική λογιστική παλινδρόμηση (binary logistic regression), πολυονομαστική λογιστική παλινδρόμηση (multi-nominal logistic regression) και τακτική λογιστική παλινδρόμηση (ordinal logistic regression)

Στο κεφάλαιο αυτό, εστιάζουμε στην δυαδική λογιστική παλινδρόμηση η οποία χρησιμοποιείται όταν η μεταβλητή στόχος είναι δυαδική, δηλαδή έχει δύο πιθανά αποτελέσματα. Για παράδειγμα την επιτυχία ή την αποτυχία ενός μαθήματος, την απάντηση «ναι» ή «όχι» σε ένα μάθημα κλπ. Όπως αναφέρθηκε προηγουμένως, η δυαδική λογιστική παλινδρόμηση μπορεί να είναι μονομεταβλητή (μια μεταβλητή κατηγορίας στόχου και μια μεταβλητή εισόδου), είτε πολυπαραγοντική (μια μεταβλητή κατηγορίας στόχου και δύο ή περισσότερες μεταβλητές εισόδου).

Η δυαδική λογιστική παλινδρόμηση προβλέπει τις πιθανότητες ενός συμβάντος με βάση τις τιμές των μεταβλητών εισόδου. Έτσι αντικρούει τις παραδοχές τις γραμμικής παλινδρόμησης, ιδιαίτερα την υπόθεση ότι τα υπολείμματα κατανέμονται κανονικά. Ο λόγος για αυτό είναι ότι η λογιστική παλινδρόμηση χρησιμοποιείται για την πρόβλεψη μεταβλητών-στόχων που ανήκουν σε μια από ένα πεπερασμένο σύνολο κατηγοριών αντί μια μεταβλητή στόχο που έχει συνεχές αποτέλεσμα. Με αυτόν τον τρόπο, υπολογίζει τις πιθανότητες εμφάνισης σε διαφορετικά επίπεδα καθεμιάς από τις μεταβλητές εισόδου και στη συνέχεια παίρνει τον λογάριθμό της για να δημιουργήσει μια συνεχή συνθήκη ως μετασχηματισμένο τύπο της μεταβλητής στόχου.

Επειδή η μεταβλητή στόχος είναι πεπερασμένη (πχ. 1 εάν υπάρχει συνθήκη, 0 εάν η συνθήκη δεν υπάρχει), η γραφική παράσταση μιας δυαδικής λογιστικής παλινδρόμησης λαμβάνει τη μορφή ενός σιγμοειδούς σχήματος [14].



Σχήμα 6 *Η σιγμοειδής καμπύλη [15].*

Το μοντέλο λαμβάνει την παρακάτω μορφή, αυτή της λογιστικής καμπύλης (logistic curve) η οποία δίνεται από την παρακάτω εξίσωση:

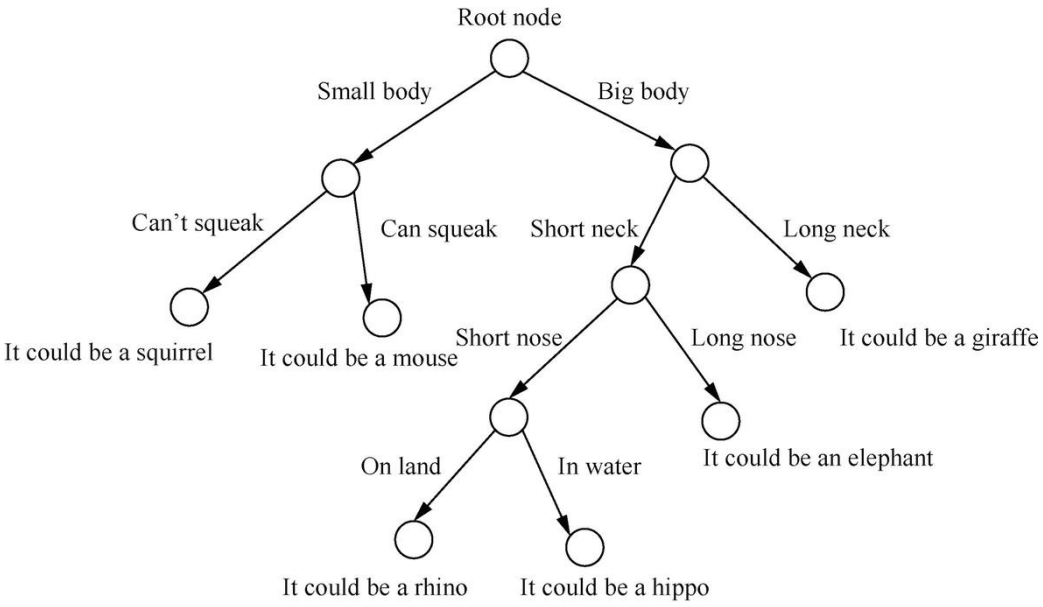
$$p(X) = \frac{1}{1 + e^{-(\alpha+\beta_0 X_0+\cdots+\beta_k X_k)}}$$

Εξίσωση 1 *Η λογιστική συνάρτηση [16].*

2.5.2. Δέντρα Απόφασης (Decision Trees)

Το δέντρο απόφασης αποτελεί έναν ταξινομητή με δομή δέντρου όπως φαίνεται στην εικόνα 2.2 [9]. Ένα δέντρο απόφασης αποτελείται από τους κλάδους (branches), τη ρίζα (root node), τους εσωτερικούς κόμβους (internal nodes) και τα φύλλα (leaf nodes) [17]. Κάθε εσωτερικός κόμβος

αντιπροσωπεύει μια δοκιμή σε ένα χαρακτηριστικό. Τα φύλλα ή αλλιώς κόμβοι φύλλων (leaf nodes) αντιπροσωπεύουν μια ετικέτα μιας κλάσης, ενώ ένας κλάδος αντιπροσωπεύει το αποτέλεσμα της δοκιμής [18].



***Σχήμα** 7 Δέντρο απόφασης.*

Από τη στιγμή που θα δημιουργηθεί, μπορεί να χρησιμοποιηθεί για να κατηγοριοποιήσει μια παρατήρηση ξεκινώντας από την ρίζα και προχωρώντας προς τους άλλους κόμβους αποφάσεων μέχρι και τον τελικό κόμβο φύλλου που αποτελεί και την πρόβλεψη [19].

Συγκριτικά με τα γραμμικά μοντέλα όπως η λογιστική παλινδρόμηση, τα δέντρα απόφασης δεν απαιτούν κλιμάκωση των χαρακτηριστικών (feature scaling). Είναι σε θέση να διαχειριστούν χαρακτηριστικά που λείπουν και αποδίδουν εξίσου καλά και σε συνεχή (continuous) αλλά και σε κατηγορικά (categorical) χαρακτηριστικά [18].

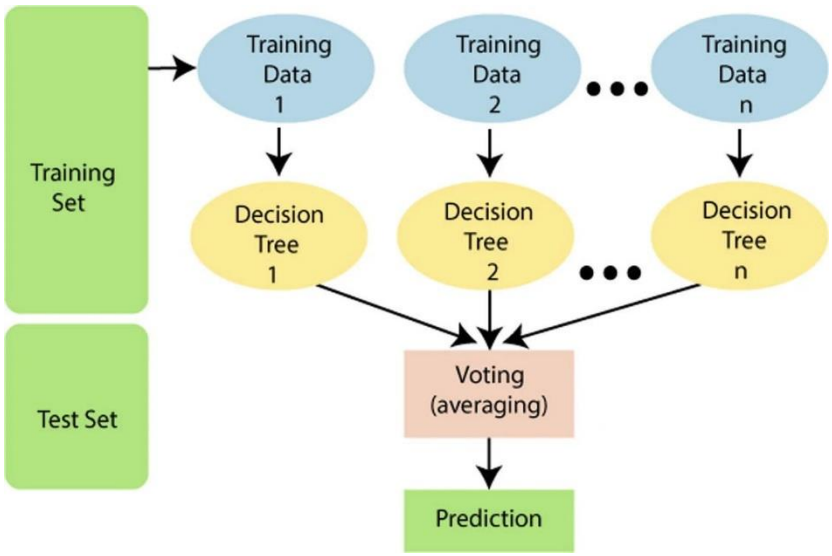
2.5.3. Τυχαίο Δάσος (Random Forest)

Πολλές από τις ελλείψεις στα δέντρα αποφάσεων μπορούν να αντιμετωπιστούν μέσω ενός συνόλου σχετικών αλγορίθμων που βασίζονται σε δέντρα, με τον πιο διάσημο από αυτούς να είναι το τυχαίο δάσος.

Το τυχαίο δάσος αποτελεί μια ομάδα δέντρων και κατ' ουσία πολλαπλά μαθησιακά μοντέλα που συνεργάζονται. Αναπτύχθηκε στα μέσα της δεκαετίας του 1990 και πλέον τα τυχαία δάση θεωρούνται ένας από τους πιο συνηθισμένους αλγορίθμους ταξινόμησης.

Με βάση μια τεχνική που ονομάζεται «bagging», τα αποτελέσματα από πολλαπλά δέντρα αποφάσεων συνδυάζονται για να παράγουν ένα πιο βέλτιστο αποτέλεσμα. Πιο συγκεκριμένα, ο αλγόριθμος επιλέγει ένα τυχαίο αριθμό μεταβλητών πρόβλεψης για να δημιουργήσει πολλά μοναδικά δέντρα (συχνά εκατοντάδες δέντρα). Κάθε δέντρο κάνει τη δική του πρόβλεψη με βάση το υποσύνολο των μεταβλητών και των δεδομένων που του δίνονται. Ο αλγόριθμος τυχαίου δάσους στη συνέχεια συνδυάζει αυτά τα αποτελέσματα για την παραγωγή της τελικής πρόβλεψης [20].

Σκοπός των τυχαίων δασών είναι η πρόληψη της υπερπροσαρμογής (overfitting). Όσο περισσότερα δέντρα αποφάσεων χρησιμοποιούνται μέσα στο τυχαίο δάσος, τόσο πιο ακριβή αποτελέσματα λαμβάνουμε. Όπως και τα δέντρα αποφάσεων, αλγόριθμος τυχαίου δάσους μπορεί να χρησιμοποιηθεί και για προβλήματα ταξινόμησης, αλλά και για προβλήματα παλινδρόμησης [21].



Σχήμα 8 *Τυχαίο δάσος [22].*

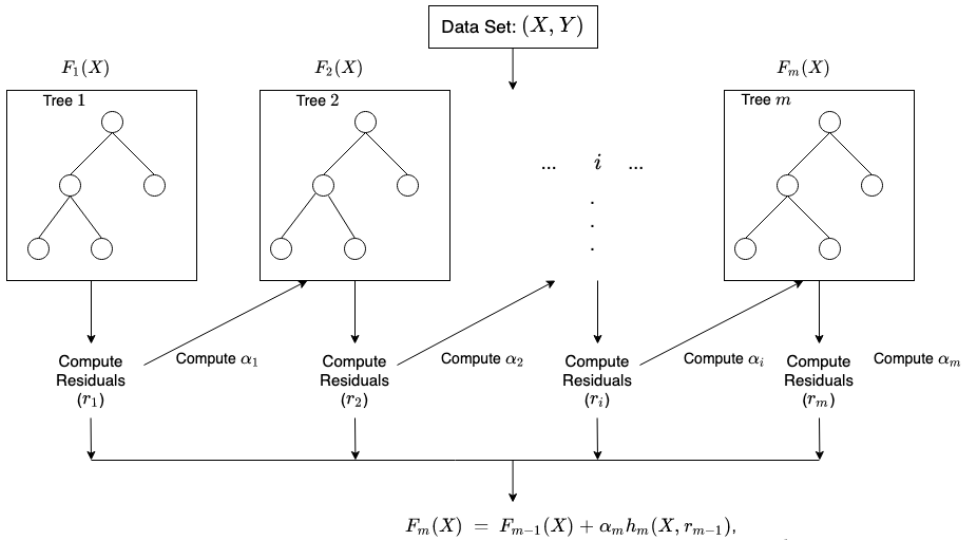
2.5.4. Ταξινομητής XGBoost (XGBoost Classifier)

Αρχικά η ενίσχυση (boosting) είναι μια μέθοδος που παράγει μια συλλογή προγνωστικών μοντέλων διαδοχικά. Δηλαδή, κάθε νέο μοντέλο να μαθαίνει από τα λάθη των προηγούμενων. Οι μέθοδοι ενίσχυσης εφαρμόζονται συνήθως όταν υπάρχει πληθώρα δεδομένων για πρόβλεψη.

Μέσω της επαναληπτικής δημιουργίας νέων μοντέλων, η διαδικασία μετατρέπει τους αδύναμους μαθητές σε δυνατούς. Κάθε επανάληψης της μεθόδου, μαθαίνει τις σχέσεις που προκύπτουν από τα δεδομένα και στη συνέχεια αναλύεται για τυχόν σφάλματα με στόχο την ελαχιστοποίηση του σφάλματος από το προηγούμενο μοντέλο.

Η gradient boosting μέθοδος, συχνά αποκαλούμενη κι ως δέντρο παλινδρόμησης είναι μια τεχνική που προέρχεται από τα δέντρα αποφάσεων με τον Jerome Friedman να εφαρμόζει την τεχνική αυτή στα δέντρα απόφασης το 2001. Στην περίπτωση αυτή, το μοντέλο εκπαιδεύεται διαδοχικά. Κάθε επαναληπτικό μοντέλο επιδιώκει να ελαχιστοποιήσει την συνάρτηση απώλειας. Κάθε νέο δέντρο εκπαιδεύεται με τα δεδομένα που προηγουμένως δεν ήταν σωστά [21].

Μια από τις παραλλαγές της μεθόδου gradient boosting είναι ο αλγόριθμος XGBoost (Extreme Gradient Boosting) που αναπτύχθηκε από τον Tianqi Chen. Ο αλγόριθμος XGBoost χρησιμοποιεί μια σειρά από ενέργειες που τον καθιστούν ταχύτερο και πιο ακριβές από την απλή gradient boosting.



Σχήμα 9 *Ο αλγόριθμος XGBoost.*

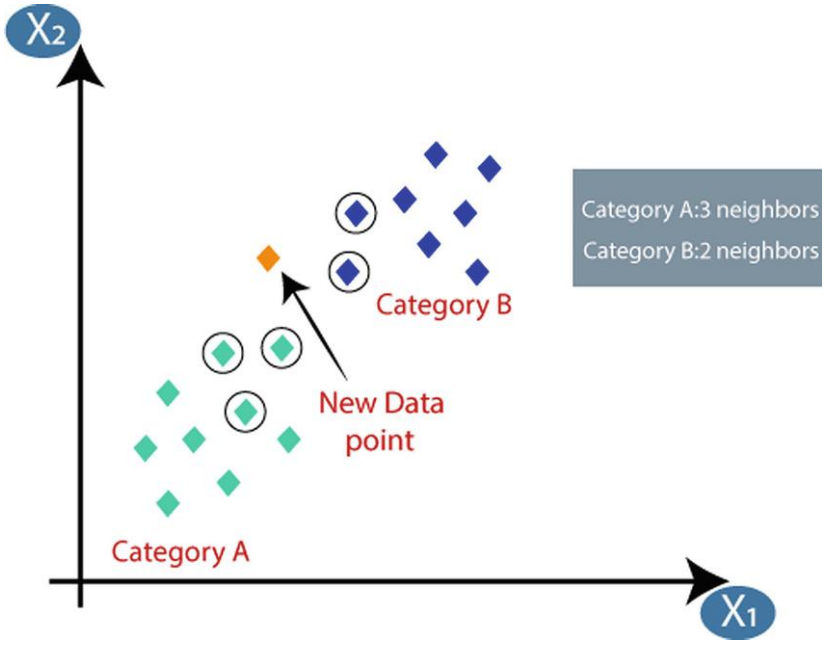
Πηγή: https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html

Είναι εξαιρετικά γρήγορος και αποτελεσματικός λόγω της χρήσης παράλληλων υπολογισμών και χρησιμοποιείται είτε για προβλήματα ταξινόμησης είτε για προβλήματα παλινδρόμησης. Λειτουργεί μόνο με αριθμητικά χαρακτηριστικά και οδηγεί σε υπερπροσαρμογή (overfitting) εάν οι υπερπαράμετροι (hyperparameters) του μοντέλου δεν καθοριστούν σωστά [13].

2.5.5. Κ-Κοντινότεροι Γείτονες (K-Nearest Neighbors)

Ένας ακόμη δημοφιλής αλγόριθμος ταξινόμησης είναι ο Κ-Κοντινότεροι Γείτονες (KNN). Με τον αλγόριθμο αυτό μπορούμε να προβλέψουμε μια κλάση για ένα νέο σημείο δεδομένων με βάση την ομοιότητά του με τα υπάρχοντα σημεία δεδομένων για τα οποία ο αλγόριθμος γνωρίζει ήδη την κλάση (δηλαδή, με βάση αυτά που έμαθε από τα δεδομένα εκπαίδευσης). Η εύρεση της

κατηγορίας ενός σημείου γίνεται με τον υπολογισμό της απόστασης από το σημείο αυτό στους πλησιέστερους γείτονες του, όπως φαίνεται και **στην εικόνα 2.5**. Κατά την αξιολόγηση, ο KNN συγκρίνει το σημείο δεδομένων με άλλα σημεία δεδομένων με παρόμοιες τιμές στις μεταβλητές πρόβλεψης και στη συνέχεια εξετάσει σε ποιες κατηγορίες ανήκουν αυτά τα σημεία. Στη συνέχεια, ο αλγόριθμος χρησιμοποιεί απλώς την πλειοψηφία για να καθορίσει την ταξινόμηση του νέου σημείο σε μία κλάση. Το «K» στον KNN είναι ο προκαθορισμένος αριθμός σημείων με τους οποίους θα συγκριθεί το νέο σημείο [20].



Σχήμα 10 *K-Κοντινότεροι Γείτονες [22].*

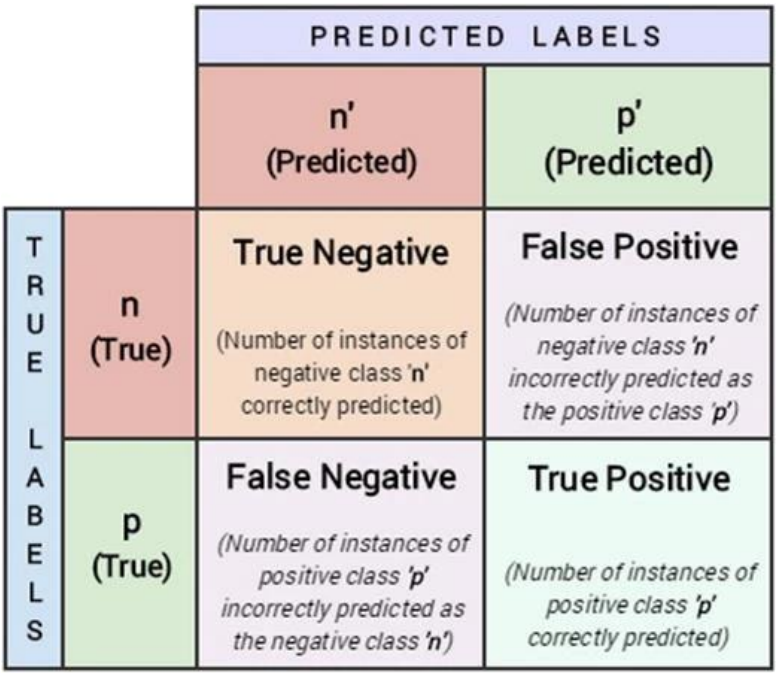
Στον πίνακα 2

2.6. ΜΕΤΡΙΚΕΣ ΑΞΙΟΛΟΓΗΣΗΣ ΜΟΝΤΕΛΟΥ

2.6.1. Πίνακας Σύγχυσης (Confusion Matrix)

Ο πίνακας σύγχυσης (confusion matrix) είναι ένας από τους πιο δημοφιλείς τρόπους αξιολόγησης ενός μοντέλου ταξινόμησης. Η αναπαράσταση του πίνακα μπορεί να χρησιμοποιηθεί για τον καθορισμό μιας ποικιλίας μετρήσεων, οι οποίες καθίστανται σημαντικές σε διαφορετικές περιπτώσεις και σενάρια.

Ο πίνακας σύγχυσης είναι μια δομή πίνακα που περιγράφει τις σωστές και τις εσφαλμένες ταξινομήσεις, όπως φαίνεται και στην παρακάτω εικόνα:



Σχήμα 11 *Πίνακας σύγχυσης.*

Σύμφωνα με τη δομή του πίνακα, παρουσιάζονται τέσσερις (4) μετρήσεις:

- **Αληθώς Θετικά (True Positives - TP):** Το σύνολο των παρατηρήσεων της θετικής κλάσης όπου το μοντέλο προέβλεψε ως θετικά.
- **Ψευδώς Θετικά (Negative Positives - NP):** Το σύνολο των παρατηρήσεων της αρνητικής κλάσης όπου το μοντέλο προέβλεψε λανθασμένα ως θετικά.
- **Αληθώς Αρνητικά (True Negatives - TN):** Το σύνολο των παρατηρήσεων της αρνητικής κλάσης όπου το μοντέλο προέβλεψε σωστά ως αρνητικά.
- **Ψευδώς Αρνητικά (Negative Positives - FN):** Το σύνολο των παρατηρήσεων της θετικής κλάσης όπου το μοντέλο προέβλεψε λανθασμένα ως αρνητικά [23].

Ο πίνακας σύγχυσης αποτελεί τη βάση για την μέτρηση της ορθότητας, της ακρίβειας, της ανάκλησης κλπ [24].

2.6.2. Ορθότητα (Accuracy)

Η ορθότητα (accuracy) είναι η απλούστερη τεχνική που χρησιμοποιείται για να προσδιορίσει αν ένα μοντέλο κάνει σωστές προβλέψεις. Υπολογίζεται ως ποσοστό της σωστής πρόβλεψης επί των συνολικών προβλέψεων που έγιναν [25]:

$$\textbf{Ορθότητα } (\boldsymbol{Accuracy}) = \frac{Αριθμός\ σωστών\ προβλέψεων}{Σύνολο\ προβλέψεων}$$

Εξίσωση 2 *Ορθότητα (Accuracy).*

Η ορθότητα μπορεί να υπολογιστεί κι από τον πίνακα σύγχυσης από την παρακάτω εξίσωση [24]:

$$\textbf{Ορθότητα } (\textbf{Accuracy}) = \frac{TP + TN}{TP + FP + FN + TN}$$

Εξίσωση 3 *Υπολογισμός ορθότητας από πίνακα σύγχυσης.*

2.6.3. Ακρίβεια (Percision)

Η ακρίβεια, γνωστή κι ως θετική προγνωστική τιμή, είναι μια μέτρηση που προκύπτει από τον πίνακα σύγχυσης [23] και χαρακτηρίζεται ως ο αριθμός των αληθή θετικών προβλέψεων από το σύνολο των παρατηρήσεων της θετικής κλάσης [26].

$$\textbf{Ακρίβεια } (\boldsymbol{Percision}) = \frac{TP}{TP + FP}$$

Εξίσωση 4 *Ακρίβεια (Precision).*

2.6.4. Ανάκληση (Recall)

Ανάκληση ή ευαισθησία (sensitivity) είναι ο αριθμός των αληθή θετικών προβλέψεων διαιρούμενος με τον αριθμό όλων των σχετικών παρατηρήσεων δηλαδή των παρατηρήσεων που θα έπρεπε να έχουν προβλεφθεί ως θετικά και υπολογίζεται:

$$\textbf{Ανάκληση } (\boldsymbol{Recall}) = \frac{TP}{TP + FN}$$

Εξίσωση 5 *Ανάκληση (Recall).*

2.6.5. F1-Score

Η μετρική F1-Score είναι ο αρμονικός μέσος όρος της ακρίβειας και της ανάκλησης. Μαθηματικά, το εύρος της είναι από 0 έως 1. Η F1-Score μας δείχνει πόσο ακριβής είναι ο ταξινομητής, δηλαδή πόσες περιπτώσεις ταξινομεί σωστά, καθώς και πόσο ισχυρός είναι (δεν χάνει σημαντικό αριθμό περιπτώσεων). Όσο μεγαλύτερη είναι η F1-Score τόσο καλύτερη είναι η απόδοση του μοντέλου. Ουσιαστικά, η μετρική προσπαθεί να βρει μια ισορροπία μεταξύ ακρίβειας και ανάκλησης. Η F1-Score υπολογίζεται από τον παρακάτω τύπο [24]:

$$\textbf{F1} - \boldsymbol{Score} = 2 * \frac{Ακρίβεια\ (Percision) * Ανάκληση\ (Recall)}{Ακρίβεια\ (Percision) + Ανάκληση\ (Recall)}$$

Εξίσωση 6 *F1-Score.*

Βάση του πίνακα σύγχυσης, η F1-Score μπορεί να υπολογισθεί [26]:

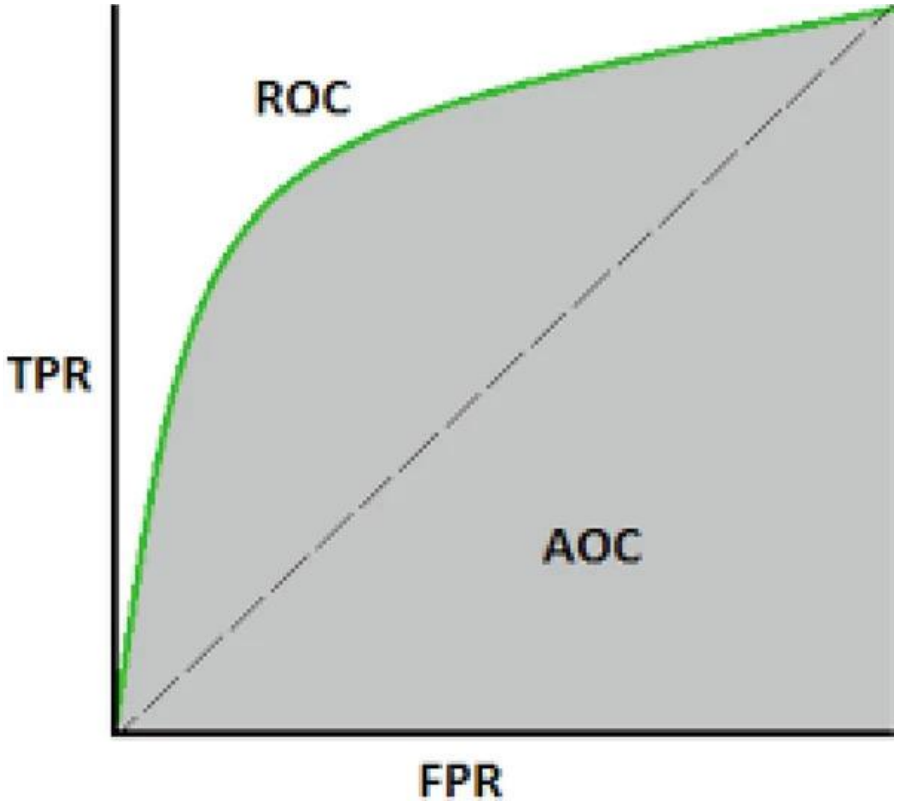
$$\textbf{F1} - \boldsymbol{Score} = 2 * \frac{TP}{TP + \frac{1}{2}.(FP + FN)}$$

Εξίσωση 7 *Υπολογισμός F1-Score από πίνακα σύγχυσης.*

2.6.6. Χαρακτηριστικά Λειτουργίας Δέκτη και Περιοχή Κάτω Από την Καμπύλη (Receiver Operating Characteristics and Area Under Curve)

Η καμπύλη χαρακτηριστικών λειτουργίας δέκτη (ROC Curve) απεικονίζει το ρυθμό των αληθώς θετικών (TPR ή recall) έναντι του ρυθμού των ψευδώς θετικών (FPR ή 1-specificity). Η περιοχή κάτω από την καμπύλη (AUC) υποδεικνύει πόσες σωστές θετικές ταξινομήσεις μπορούν να

επιτευχθούν δίνοντας ανάλογο χώρο κάτω από την καμπύλη [25]. Η περιοχή κάτω από την καμπύλη ενός ταξινομητή είναι ίση με την πιθανότητα ο ταξινομητής να κατατάξει ένα τυχαία επιλεγμένο θετικό σημείο υψηλότερα από ένα τυχαία επιλεγμένο αρνητικό σημείο. Με άλλα λόγια, η καμπύλη χαρακτηριστικών λειτουργίας δέκτη είναι μια καμπύλη πιθανότητας και η AUC μετρά την διαχωρισιμότητα δηλαδή, την ικανότητα του μοντέλου να διακρίνει τις κατηγορίες. Όσο υψηλότερη είναι η AUC, τόσο καλύτερο είναι το μοντέλο.



Σχήμα 12 *Καμπύλη χαρακτηριστικών λειτουργίας δέκτη και περιοχή κάτω από την καμπύλη [24].*

2.7. ΡΟΗ ΕΡΓΑΣΙΩΝ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Ένα πρόβλημα μηχανικής μάθησης συνήθως ακολουθεί τα παρακάτω βήματα:

- **Καθορισμός του προβλήματος:** Πρέπει αρχικά να προσδιοριστεί ποιο ακριβώς είναι το πρόβλημα προτού αρχίσουμε να το λύνουμε. Πρέπει επίσης να καταλάβουμε ποιο είναι

το πρόβλημα, αν είναι εφικτό να χρησιμοποιηθεί η μηχανική μάθηση για την επίλυσή του και ούτω καθεξής

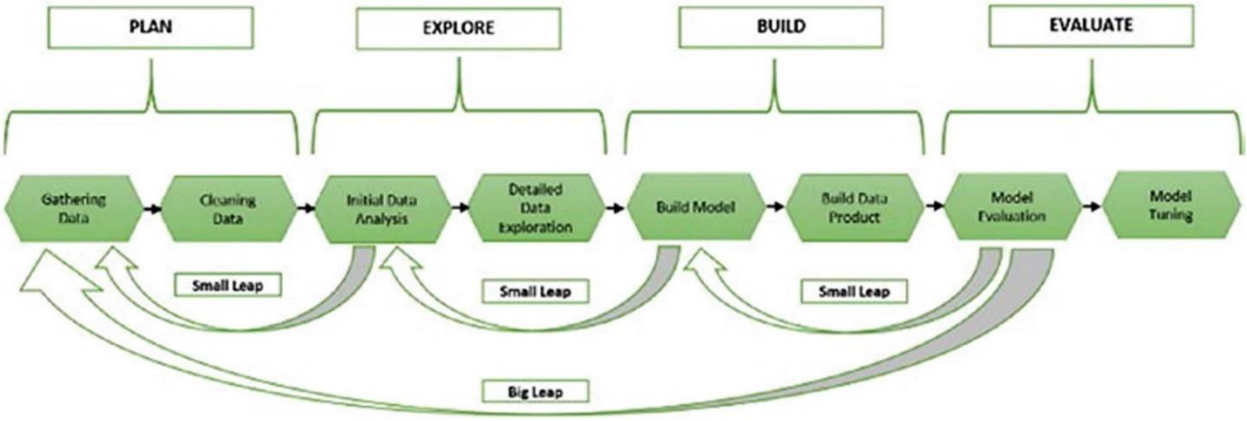
- **Συλλογή των δεδομένων:** Στη συνέχεια, πρέπει να συγκεντρώσουμε τα δεδομένα μας με βάση τον ορισμό του προβλήματος. Τα δεδομένα είναι εξαιρετικά σημαντικά κι πρέπει επομένως να συλλέγονται με προσοχή. Χρειάζεται να βεβαιωθούμε ότι έχουμε δεδομένα που αντιστοιχούν σε όλους τους απαραίτητους παράγοντες που απαιτούνται για την ανάλυση.
- **Προ-επεξεργασία των δεδομένων:** Πρέπει να καθαρίσουμε τα δεδομένα για να τα κάνουμε πιο χρηστικά. Αυτό περιλαμβάνει πολλά βήματα όπως την αφαίρεση ακραίων στοιχείων, τον χειρισμό ελλειπουσών πληροφοριών κλπ. Αυτό γίνεται για να μειωθεί η πιθανότητα σφαλμάτων κατά την ανάλυσή μας.
- **Ανάπτυξη του μοντέλου:** Δημιουργούμε το μοντέλο μηχανικής μάθησης, το οποίο θα χρησιμοποιηθεί για την επίλυση του προβλήματος. Αυτό το μοντέλο παίρνει τα δεδομένα ως είσοδο, εκτελεί διαδικασίες και υπολογισμούς σε αυτά κι έπειτα παράγει κάποια έξοδο για αυτά.
- **Αξιολόγηση του μοντέλου:** Το μοντέλο πρέπει να αξιολογηθεί και να επαληθευτεί ή ακρίβειά του. Αυτό γίνεται χρησιμοποιώντας τις διαδικασίες του μοντέλου σε νέα δεδομένα [1].

2.7.1. Συλλογή Δεδομένων

Η εύρεση των κατάλληλων δεδομένων είναι το πρώτο βήμα [27]. Η διαδικασία αυτή αφορά την εξαγωγή, την επιμέλεια και τη συλλογή όλων των απαραίτητων δεδομένων που απαιτούνται [5]. Τα δεδομένα μπορούν να ληφθούν από διάφορες πηγές όπως:

- **Ερωτηματολόγια:** Χρησιμοποιούνται για τη συλλογή δεδομένων προκειμένου να αναπτυχθεί ένα συμπέρασμα που μπορεί σε μια ευρύτερη κλίμακα.
- **Δημοσκοπήσεις:** Χρησιμοποιούνται για την κατανόηση των απόψεων ή των συναισθημάτων των ανθρώπων για ένα συγκεκριμένο θέμα.
- **Συνεντεύξεις:** Πρόκειται για δομημένες συνομιλίες στην οποία ένα άτομο κάνει ερωτήσεις ενώ το άλλο άτομο απαντά σε αυτές.

- **Παρατήρηση:** Αφορά τη διαδικασία παρατήρησης ή παρακολούθησης της φυσικής αντίδρασης, ανταπόκρισης και συμπεριφοράς προκειμένου να καταλήξουμε σε κάποιο χρήσιμο συμπέρασμα.



Σχήμα 13 *Η διαδικασία της μηχανικής μάθησης [33].*

Ωστόσο, δεδομένα μπορούν να συλλεχθούν από διαδικτυακές πηγές όπως τα μέσα κοινωνικής δικτύωσης, web scraping, ή η λήψη ήδη συλλεγμένων συνόλων δεδομένων. Εφόσον τα δεδομένα έχουν συλλεχθεί, χρειάζεται να γίνει προ-επεξεργασία αυτών πριν χρησιμοποιηθούν για περαιτέρω εφαρμογές [1].

2.7.2. Προ-επεξεργασία Δεδομένων

Η προ-επεξεργασία δεδομένων είναι ένα κρίσιμο βήμα που απαιτείται για την εκτέλεση της μηχανικής μάθησης, καθώς πρέπει να καθαρίσουμε, να φιλτράρουμε, να συγχωνεύσουμε και να μετατρέψουμε τα δεδομένα μας για να τα φέρουμε στην επιθυμητή μορφή, ώστε να μπορέσουμε να εκπαιδεύσουμε το μοντέλο [28]. Η προ-επεξεργασία και προετοιμασία των δεδομένων είναι συνήθως η πιο χρονοβόρα φάση στον κύκλο ζωής της εξόρυξης δεδομένων και συχνά απαιτεί 60% έως 70% του συνολικού χρόνου. Ωστόσο, αυτή η φάση είναι κρίσιμης σημασίας, δεδομένου ότι λανθασμένα δεδομένα θα οδηγήσουν σε κακά μοντέλα και κατά συνέπεια σε κακές αποδόσεις και αποτελέσματα [5].

2.7.2.1. Καθαρισμός Δεδομένων

Ο καθαρισμός δεδομένων αντιμετωπίζει μη ολοκληρωμένες, διπλότυπες, ελλείπουσες και λανθασμένες τιμές. Σκοπός της εκκαθάρισης των δεδομένων είναι να βελτιώσει την ποιότητα των δεδομένων για την μετέπειτα εκπαίδευση του μοντέλου, την παροχή αναλυτικών στοιχείων, ή τη λήψη μιας απόφασης. Η φάση καθαρισμού δεδομένων περιλαμβάνει συνήθως τις ακόλουθες διαδικασίες [29]:

- Συμπλήρωση ελλειπουσών τιμών.
- Διαχείριση διπλότυπων δεδομένων.
- Διόρθωση τύπων των χαρακτηριστικών.
- Αφαίρεση ακραίων στοιχείων [27, 29].

2.7.2.2. Συμπλήρωση ελλειπουσών τιμών

Η ελλείπουσες τιμές είναι συχνό φαινόμενο. Τα περισσότερα μοντέλα μηχανικής μάθησης απαιτούν ολοκληρωμένα σύνολα δεδομένων [29]. Ελλείπουσες τιμές μπορούν να προκύπτουν από τυχαίο λάθος, λάθος συστήματος, ανθρώπινο λάθος, λανθασμένη αναγνώριση ενός αισθητήρα κλπ. [1].

Υπάρχουν διάφορες προσεγγίσεις για την αντιμετώπιση των ελλειπουσών τιμών. Μια προσέγγιση είναι η αφαίρεση μεμονωμένων ελλειπουσών τιμών ή ακόμα κι ολόκληρες παρατηρήσεις όπου τουλάχιστον λείπει μια τιμή από τα δεδομένα. Αυτή η προσέγγιση ωστόσο, δεν είναι εφικτή σε περίπτωση που αποκλείει μεγάλο αριθμό παρατηρήσεων, καθώς μειώνει τον όγκο των πληροφοριών που δίνονται για ανάλυση. Μια εναλλακτική προσέγγιση είναι η συμπλήρωση των τιμών που λείπουν, διατηρώντας έτσι όσο το δυνατόν περισσότερα δεδομένα, διακινδυνεύοντας ωστόσο την εισαγωγή εσφαλμένων. Οι απλές μέθοδοι συμπλήρωσης είναι η συμπλήρωση της ελλείπουσας τιμής με το αντίστοιχο της προηγούμενης παρατήρησης, συχνά αποκαλούμενη κι ως πλήρωση προς τα εμπρός. Εναλλακτικά, μπορεί να συμπληρωθεί από το επόμενο έγκυρο δεδομένο μιας παρατήρησης. Μια άλλη προσέγγιση είναι η αντικατάσταση των ελλειπουσών τιμών με τον μέσο όρο του χαρακτηριστικού ή τη χρήση τιμών από παρατηρήσεις που περιέχουν κοινά στοιχεία [30].

2.7.2.3. Διαχείριση διπλότυπων δεδομένων

Η διαχείριση διπλότυπων δεδομένων μπορεί να είναι πιο απλή όταν υπάρχουν περισσότερα αντίγραφα των ίδιων δεδομένων, καθώς απαιτείται ο εντοπισμός τους και η αφαίρεσή τους, ενώ σε αντίθεση με τα δεδομένα που δεν είναι εντελώς ταυτόσημα, είναι πιο δύσκολη, καθώς απαιτείται η κατανόηση εάν πράγματι αυτά τα δεδομένα αντιστοιχούν στην ίδια πληροφορία [31].

2.7.2.4. Διαχείριση ακραίων στοιχείων

Ένα ακραίο στοιχείο είναι ένα στοιχείο στα δεδομένα που διαφέρει από τα υπόλοιπα. Ο Hawkins ορίζει τα ακραία στοιχεία:

> *Μια ακραία παρατήρηση είναι μια παρατήρηση που αποκλίνει τόσο πολύ από τις άλλες παρατηρήσεις ώστε να εγείρει υποψίες ότι δημιουργήθηκε από διαφορετικό μηχανισμό.*

Τα ακραία σημεία αναφέρονται επίσης ως ανωμαλίες στη βιβλιογραφία της εξόρυξης δεδομένων και των στατιστικών [32]. Όπως φαίνεται και στον **Πίνακας 2**, η τιμή για το *h* είναι 100. Είναι πολύ μεγαλύτερη σε σύγκριση με τις υπόλοιπες τιμές, οι οποίες είναι εντός του εύρους 1 με 5 [1].

Πίνακας 2 *Υπολογισμός ακραίων παρατηρήσεων.*

	a	b	c	d	e	f	g	h	i	j
X	2	3	1	1	4	3	5	100	5	2

2.7.2.5. Επιλογή Χαρακτηριστικών

Με την επιλογή χαρακτηριστικών, εντοπίζονται τα σχετικά χαρακτηριστικά και φιλτράρονται τα θορυβώδη δεδομένα. Ένα χαρακτηριστικό θεωρείται μη χρήσιμο εάν δεν παρέχει πολύτιμες πληροφορίες και περιττό εάν από αυτό δεν προκύπτουν επιπλέον πληροφορίες [29]. Μέσω της

επιλογής χαρακτηριστικών, αυτά τα περιττά ή άσχετα χαρακτηριστικά μπορούν να εξαλειφθούν, έτσι ώστε το μοντέλο να απλοποιείται και να ερμηνεύεται ευκολότερα από τους χρήστες. Ταυτόχρονα, η επιλογή χαρακτηριστικών, μπορεί να μειώσει αποτελεσματικά τον χρόνο εκπαίδευσης του μοντέλου, να βελτιώσει την απόδοση του και να αποφύγει την υπερπροσαρμογή (overfitting) [9].

2.7.2.6. Κωδικοποίηση (Encoding)

Η κωδικοποίηση μετατρέπει τις κατηγορικές μεταβλητές σε αριθμητικές και μηχανικά αναγνώσιμες [29]. Ορισμένοι αλγόριθμοι όπως ο KNN ή οι αλγόριθμοι παλινδρόμησης δεν μπορούν να λειτουργήσουν με κατηγορικές τιμές, αντιθέτως χρειάζονται αριθμητικές τιμές. Μπορούμε να μετατρέψουμε μια κατηγορική μεταβλητή σε πολλές αριθμητικές δημιουργώντας για κάθε μια από τις κατηγορικές τιμές της ένα νέο δυαδικό αριθμητικό χαρακτηριστικό που λαμβάνει τιμές 0 ή 1. Η συγκεκριμένη τεχνική κωδικοποίησης ονομάζεται one-hot-encoding.

2.7.2.7. Κλιμάκωση (Scaling)

Οι αριθμητικές τιμές των χαρακτηριστικών μπορεί να έχουν διαφορετικά εύρη, όπως για παράδειγμα η ηλικία και το εισόδημα. Το μεγάλο εύρος των τιμών του εισοδήματος μπορεί να επηρεάσει την εκπαίδευση του μοντέλου. Η κλιμάκωση (scaling) αντιμετωπίζει αυτό το πρόβλημα, καθιστώντας τα αριθμητικά χαρακτηριστικά ισοδύναμα ως προς το εύρος τους μετά την κλιμάκωση. Η κλιμάκωση μπορεί να επιτευχθεί με έναν από τους παρακάτω δύο τρόπους:

- **Κανονικοποίηση (Normalization):** Κλιμακώνει όλες τις τιμές σε ένα εύρος μεταξύ 0 και 1. Αυτή η τροποποίηση δεν επηρεάζει την κατανομή του χαρακτηριστικού, αν και λόγω των χαμηλότερων τυπικών αποκλίσεων ενισχύει την επίδραση των ακραίων τιμών. Ως αποτέλεσμα, συνιστάται να αντιμετωπίσετε τις ακραίες τιμές πριν από την κανονικοποίηση.
- **Τυποποποίηση (Standardization):** Η τυποποίηση είναι η διαδικασία κλιμάκωσης των τιμών λαμβάνοντας υπόψη την τυπική απόκλιση για να φτάσουμε σε μια κατανομή με μέση τιμή 0 και τυπική απόκλιση 1 [27].

2.7.3. Διαχωρισμός Δεδομένων

Εφόσον ολοκληρώσουμε την προ επεξεργασία των δεδομένων, θα πρέπει να τα χωρίσουμε σε δύο μέρη:

- **Δεδομένα εκπαίδευσης (Training data):** Τα δεδομένα εκπαίδευσης είναι αυτά που τροφοδοτούνται στο μοντέλο που θα χρησιμοποιηθεί κατά τη διάρκεια της εκπαίδευσης. Αποτελούν το μεγαλύτερο ποσοστό του συνόλου των δεδομένων, καθώς το μοντέλο απαιτεί μεγάλο όγκο δεδομένων κατά την εκπαίδευση για να έχει πιο ακριβή αποτελέσματα.
- **Δεδομένα δοκιμής (Test data):** Τα δεδομένα δοκιμής τροφοδοτούνται στο μοντέλο αφού τελειώσει η εκπαίδευση και καταλήξει στις βέλτιστες παραμέτρους. Αυτό θα είναι το μικρότερο ποσοστό επειδή προορίζεται μόνο για να βοηθήσει το μοντέλο να καθορίσει πόσο ακριβής ή ανακριβής είναι η πρόβλεψή του [34].

Η πιο συχνή πρακτική για τους περισσότερους αλγορίθμους μηχανικής μάθησης είναι να εκπαιδεύονται με το 80% των δεδομένων και να δοκιμάζονται με το υπόλοιπο 20%. Ωστόσο, συχνά βλέπουμε το λόγο εκπαίδευσης/δοκιμής να κυμαίνεται οπουδήποτε από 70/30 έως 85/15 [35].

2.7.4. Επιλογή και Ρύθμιση Υπερπαραμέτρων

Η ρύθμιση υπερπαραμέτρων είναι ένα κρίσιμο βήμα στη ροή εργασίας της Επιστήμης Δεδομένων. Εάν εφαρμοστεί σωστά, μπορεί να μετατρέψει ένα αναποτελεσματικό μοντέλο σε ένα έτοιμο μοντέλο και να λάβει αποφάσεις. Κάθε μοντέλο μηχανικής μάθησης έχει υπερπαραμέτρους και η εύρεση ενός τρόπου για τον αυτόματο προσδιορισμό των καλύτερων τιμών είναι ένα από τα πρώτα βήματα. Κάθε μοντέλο μηχανικής μάθησης παρέχεται με ένα σύνολο υπερπαραμέτρων και στόχος είναι να βρεθεί ο καλύτερος διαθέσιμος συνδυασμός τιμών για να μεγιστοποιηθεί η ακρίβειά του.

Τα μοντέλα μηχανικής μάθησης χαρακτηρίζονται από δύο τύπους παραμέτρων:

- **Παράμετροι του μοντέλου:** Είναι διαφορετικές μεταβλητές που λαμβάνονται υπόψη κατά την εκπαίδευση ενός μοντέλου. Αυτοί οι τύποι παραμέτρων μπορούν να περιλαμβάνουν για παράδειγμα, τις τιμές βάρους στα τεχνητά νευρωνικά δίκτυα και στη γραμμική παλινδρόμηση.
- **Υπερπαράμετροι:** Είναι όλες οι διαφορετικές τιμές παραμέτρων, οι οποίες μπορούν να οριστούν από έναν χρήστη πριν από την εκπαίδευση ενός μοντέλου. Επομένως, αυτές οι παράμετροι μπορούν να δημιουργήσουν διάφορους περιορισμούς σχετικά με τον τρόπο εκπαίδευσης ενός μοντέλου. Για παράδειγμα, δύο Υπερπαράμετροι σε ένα μοντέλο Random Forest είναι ο αριθμός των εκτιμητών συνολικά και το μέγιστο επιτρεπόμενο βάθος σε κάθε εκτιμητή.

Η βαθιά κατανόηση του τρόπου λειτουργίας των διαφορετικών υπερπαραμέτρων έχει μεγάλη σημασία όταν πρόκειται για τον καθορισμό των τιμών τους.

Πίνακας 3 *Παράμετροι και Υπερπαράμετροι [36].*

Παράμετροι	**Υπερπαράμετροι**
Λαμβάνονται υπόψη κατά τη διάρκεια εκπαίδευσης ενός μοντέλου	Τιμές που ορίζονται πριν από την εκπαίδευση ενός μοντέλου
Εσωτερικά του μοντέλου και αποθηκεύονται μετά την εκπαίδευση	Εξωτερικά του μοντέλου και δεν αποθηκεύονται μετά την εκπαίδευση
Εξαρτώνται από το σύνολο δεδομένων εκπαίδευσης	Ανεξάρτητες από το σύνολο δεδομένων εκπαίδευσης

Η ρύθμιση υπερπαραμέτρων αποτελεί πεδίο έρευνας από τις αρχές της δεκαετίας του 1990 και διαφορετικές μελέτες έχουν αποδείξει ότι διαφορετικές τιμές υπερπαραμέτρων αποδίδουν σταθερά καλύτερα για διαφορετικά σύνολα δεδομένων.

Οι τεχνικές ρύθμισης υπερπαραμέτρων χωρίζονται κυρίως σε δύο διαφορετικές κατηγορίες:

1. Παραδοσιακές προσεγγίσεις που επικεντρώνονται αποκλειστικά στην εξερεύνηση διαφορετικών περιοχών του χώρου αναζήτησης
2. Πιο περίπλοκες προσεγγίσεις που ελπίζουν να επιταχύνουν τη διαδικασία εστιάζοντας κυρίως στους πιο πολλά υποσχόμενους συνδυασμούς τιμών υπερπαραμέτρων.

Στον παρακάτω πίνακα παρουσιάζονται μερικές από τις πιο διαδεδομένες υπερπαραμέτρους των αλγορίθμων ταξινόμησης, όπως αυτοί αναφέρθηκαν στην **ενότητα 2.5.**

Πίνακας 4 *Υπερπαράμετροι αλγορίθμων ταξινόμησης.*

Αλγόριθμος	**Υπερπαράμετρος**	**Περιγραφή**
`Logistic Regression`	`C`	*Ένταση της κανονικοποίησης [61].*
	`class_weight`	*Τα βάρη των κλάσεων [62].*
	`fit_intercept`	*Καθορίζει εάν πρέπει να προστεθεί μια σταθερά στη συνάρτηση απόφασης [62].*
	`max_iter`	*Καθορίζει τον μέγιστο αριθμό επαναλήψεων που πρέπει να εκτελεστούν [62].*
	`multi_class`	*Δυαδική ή πολυωνυμική κλάση [62].*
	`solver`	*Τύπος για την κανονικοποίηση [61].*
	`tol`	*Ανοχή στα κριτήρια παύσης [62].*
	`warm_start`	*Όταν το warm_start είναι αληθές, τα υπάρχοντα χαρακτηριστικά του προσαρμοσμένου μοντέλου χρησιμοποιούνται για την προετοιμασία του νέου μοντέλου [60].*
`XGBoost`	`n_estimators`	*Καθορίζει τον αριθμό των δέντρων που θα χρησιμοποιήσει το μοντέλο. Περισσότερα δέντρα κάνουν το μοντέλο πιο περίπλοκο και πιθανώς πιο ακριβές, αλλά προκαλούν και μεγάλο χρόνο λειτουργίας [58].*

	Learning_rate	*Διαχειρίζεται τη μείωση των βαρών σε κάθε βήμα ενίσχυσης/δέντρο απόφασης. Η μείωση των βαρών βοηθάει στη μείωση της επιρροής μεμονωμένων δέντρων στο σύνολο [58].*
	subsample	*Καθορίζει την ποσότητα και την τυχαία επιλογή των δεδομένων που θα χρησιμοποιηθούν σε κάθε επανάληψη [58].*
	colsample_bytree	*Καθορίζει τον αριθμό και την τυχαία επιλογή των χαρακτηριστικών που επιλέγονται σε δέντρο [58].*
	max_depth	*Καθορίζει το μέγιστο βάθος του κάθε δέντρου/βήματος ενίσχυσης [58].*
	min_child_weight	*Είναι το άθροισμα των ελάχιστων βαρών του δείγματος [59].*
Random Forest	min_samples_split	*Ο ελάχιστος αριθμός των παρατηρήσεων που χρειάζεται για την δημιουργία του επόμενου κόμβου [58].*
	n_estimators	*Καθορίζει τον αριθμό των δέντρων που θα χρησιμοποιήσει το μοντέλο. Περισσότερα δέντρα κάνουν το μοντέλο πιο περίπλοκο και πιθανώς πιο ακριβές, αλλά προκαλούν και μεγάλο χρόνο λειτουργίας [58].*
	min_impurity_decrease	*Ένας κόμβος θα διαχωριστεί εάν ο διαχωρισμός προκαλέσει μείωση της αταξίας μεγαλύτερη η ίση με αυτή την τιμή [60].*
	min_samples_leaf	*Ο ελάχιστος αριθμός των παρατηρήσεων σε ένα κόμβο [58].*
	max_depth	*Το μέγιστο βάθος του δέντρου [58].*

	max_features	*Ο μέγιστο αριθμό των χαρακτηριστικών που λαμβάνονται υπόψη σε ένα διαχωρισμό [60].*
	criterion	*Συνάρτηση που μετρά την ποιότητα σε ένα διαχωρισμού [60].*
	max_depth	*Καθορίζει το μέγιστο βάθος του κάθε δέντρου/βήματος ενίσχυσης [58].*
	boostrap	*Εάν ή όχι, θα χρησιμοποιηθούν δείγματα bootstrap κατά την δημιουργία των δέντρων [60].*
Decision Tree	min_samples_split	*Ο ελάχιστος αριθμός των παρατηρήσεων που χρειάζεται για την δημιουργία του επόμενου κόμβου [1].*
	min_impurity_decrease	*Ένας κόμβος θα διαχωριστεί εάν ο διαχωρισμός προκαλέσει μείωση της αταξίας μεγαλύτερη η ίση με αυτή την τιμή [60]*
	min_samples_leaf	*Ο ελάχιστος αριθμός των παρατηρήσεων σε ένα κόμβο [58].*
	max_depth	*Το μέγιστο βάθος του δέντρου [58].*
	max_features	*Ο μέγιστο αριθμό των χαρακτηριστικών που λαμβάνονται υπόψη σε ένα διαχωρισμό [60].*
	criterion	*Συνάρτηση που μετρά την ποιότητα σε ένα διαχωρισμού [60].*
	max_depth	*Καθορίζει το μέγιστο βάθος του κάθε δέντρου/βήματος ενίσχυσης [58].*
K-Nearest Neighbors	n_neighbors	*Καθορίζει τον αριθμό των γειτόνων που λαμβάνονται υπόψη από το μοντέλο. Σε περιπτώσεις ταξινόμησης, επηρεάζει την ομαλότητα του ορίου απόφασης [58].*

	`p`	*Επηρεάζει το μέτρο απόστασης που χρησιμοποιείται για τον προσδιορισμό των γειτόνων [58].*
	`weights`	*Καθορίζει το βάρος των γειτόνων. Μπορεί αν δοθεί ίση προτεραιότητα στους πλησιέστερους γείτονες, ή να ληφθεί υπόψη η απόσταση από το σημείο ενδιαφέροντος, δηλαδή όσο πιο μακριά είναι ένα σημείο τόσο μικρότερο και το βάρος του [58].*
	`algorithm`	*Καθορίζει τον αλγόριθμο που θα προσδιορίσει τους πλησιέστερους γείτονες [1].*
	`metric`	*Καθορίζει την μετρική που θα χρησιμοποιηθεί για τον υπολογισμό της απόστασης μεταξύ των σημείων [1].*

2.7.4.1. Μη Αυτόματη Αναζήτηση (Manual Search)

Η πιο εύκολη προσέγγιση για τη ρύθμιση υπερπαραμέτρων είναι η δοκιμή διαφορετικών τιμών για μια υπερπαράμετρο και να δείτε πως επηρεάζουν τα αποτελέσματα. Επαναλαμβάνοντας τη διαδικασία αυτή, μπορεί να βελτιστοποιηθεί η αρχική εκτίμηση.

2.7.4.2. Αναζήτηση Πλέγματος (Grid Search)

Στην προσέγγιση αυτή, δημιουργείται ένα πλέγμα τιμών για κάθε υπερπαράμετρο και στη συνέχεια το μοντέλο εκπαιδεύεται/δοκιμάζεται σε σχέση με κάθε πιθανό συνδυασμό τιμών. Η αναζήτηση πλέγματος αναφέρεται μερικές φορές κι ως «πλήρης παραγοντικός σχεδιασμός» καθώς αξιολογεί το καρτεσιανό γινόμενο για κάθε σύνολο τιμών που εκχωρούνται σε κάθε υπερπαράμετρο. Η μέθοδος αυτή επιτρέπει την κάλυψη πολλών πιθανών συνδυασμών και τη διερεύνηση ενός μεγάλου μέρους του διαθέσιμου χώρου αναζήτησης της βελτιστοποίησης. Ένα μειονέκτημα της αναζήτησης πλέγματος, ωστόσο, μπορεί να θεωρηθεί ως εξαιρετικά ακριβό,

τόσο σε άποψη χρόνου όσο και σε των υπολογιστικών πόρων που απαιτούνται, σύγκριση με άλλες προσεγγίσεις.

2.7.4.3. Τυχαία Αναζήτηση (Random Search)

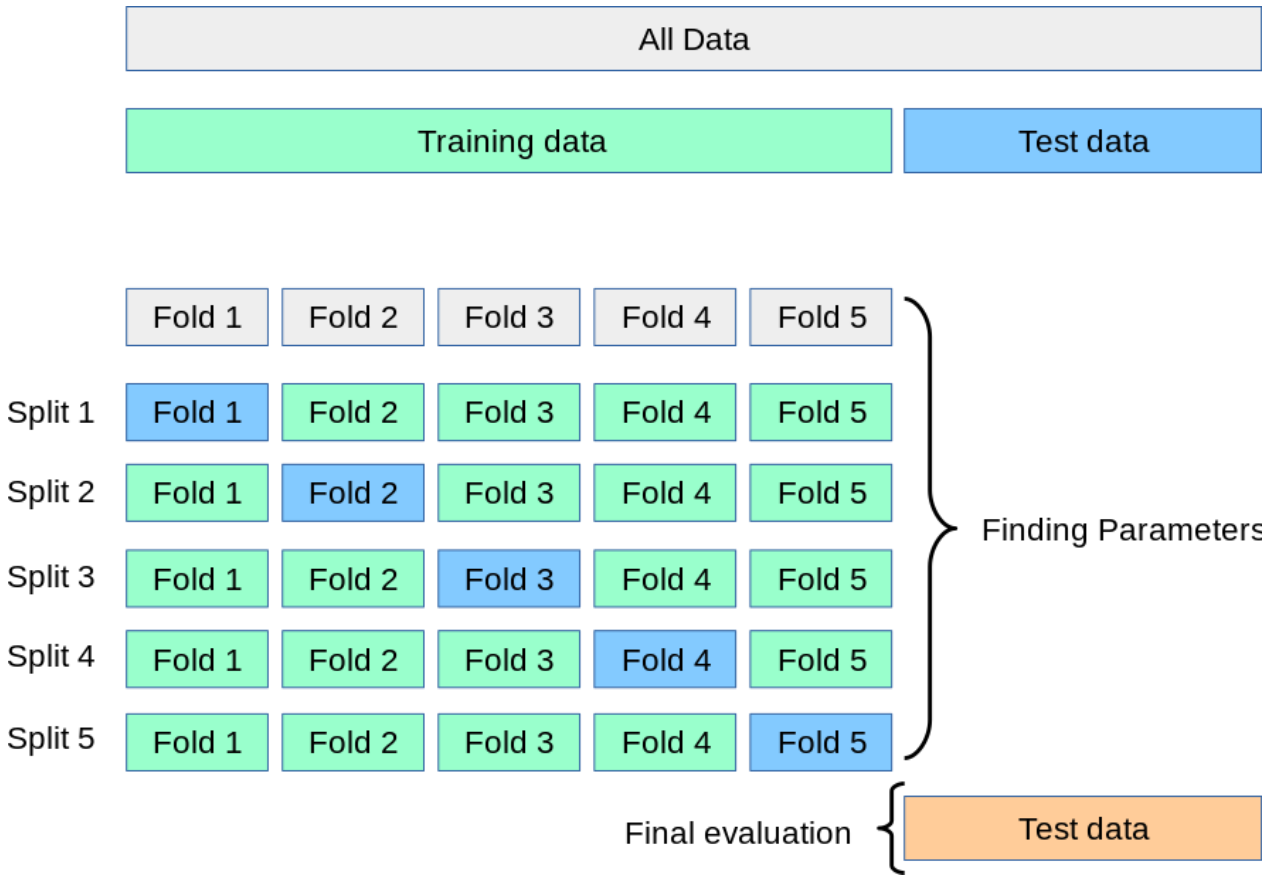
Η μέθοδος αυτή, λαμβάνει συνδυασμούς υπερπαραμέτρων τυχαία από τον χώρο αναζήτησης. Μπορεί να αποτελέσει πιο αποτελεσματική μέθοδος από τις παραδοσιακές προσεγγίσεις μη αυτόματης αναζήτησης (manual search) και αναζήτησης πλέγματος (grid search) όταν ορισμένες υπερπαράμετροι έχουν μεγαλύτερο βάρος από άλλες. Η τυχαία αναζήτηση, μπορεί να επιφέρει παρόμοια αποτελέσματα με την αναζήτηση πλέγματος, ενώ μειώνει σημαντικά τις υπολογιστικές απαιτήσεις. Πιθανό μειονέκτημα της προσέγγισης αυτής μπορεί να θεωρηθεί το γεγονός ότι δεν υπάρχει έλεγχος ως προς τον τρόπο επιλογής των διαφορετικών τιμών.

2.7.4.4. Μπεϋζιανή Ρύθμιση (Bayesian Optimization)

Χρησιμοποιώντας την μπεϋζιανή προσέγγιση, όλες οι διαφορετικές τιμές υπερπαραμέτρων μπορούν να προσδιοριστούν αυτόματα χωρίς ανθρώπινη παρέμβαση. Η βασική διαφορά της σε σχέση με την αναζήτηση πλέγματος και της τυχαίας αναζήτησης είναι ότι χρησιμοποιούνται πληροφορίες από προηγούμενα πειράματα για να κατανοήσει ποιες τιμές για τις υπερπαραμέτρους μπορεί να είναι καλύτερο να δοκιμαστούν ή να αποφευχθούν. Έτσι τείνει να βρίσκει λύση με πιο βέλτιστο τρόπο, δίνοντας ταυτόχρονα λιγότερη προσοχή σε περιοχές του χώρου αναζήτησης που δεν παρέχουν προστιθέμενη αξία [36].

2.7.5. Διασταυρωμένη Επικύρωση K-Fold (K-Fold Cross Validation)

Η διασταυρωμένη επικύρωση είναι ένας κοινός τρόπος αξιολόγησης της απόδοσης των μοντέλων μηχανικής μάθησης. Στην k-fold τα δεδομένα χωρίζονται σε k πτυχές ίσου μεγέθους. Κάθε πτυχή είναι ένα σύνολο επικύρωσης (δοκιμής) για την αξιολόγηση του μοντέλου. Οι υπόλοιπες $k - 1$ πτυχές αξιοποιούνται για την εκπαίδευση του μοντέλου. Η μέση τιμή της απόδοσης των πτυχών είναι και η τελική εκτίμηση της απόδοσης μοντέλου [37, 38].



Σχήμα 14 *Διασταυρωμένη επικύρωση k-fold.*

Πηγή: https://scikit-learn.org/stable/modules/cross_validation.html

ΚΕΦΑΛΑΙΟ 3: ΜΕΘΟΔΟΛΟΓΙΑ

3.1. ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΜΕΘΟΔΟΛΟΓΙΑΣ

Τα στάδια που ακολουθήθηκαν για την μελέτη της παρούσας έρευνας αφορούν την προ-επεξεργασία των δεδομένων, τη διερευνητική ανάλυση των δεδομένων και την παραγωγή γενικών πληροφοριών και οπτικοποιήσεων, την προετοιμασία των δεδομένων για μηχανική μάθηση και τη δημιουργία του μοντέλου και την εφαρμογή των αλγορίθμων. Το πρόβλημα μηχανικής μάθησης που επιλέχθηκε αποτελεί πρόβλημα επιβλεπόμενης μηχανικής μάθησης με σκοπό τη πρόβλεψη παραμονής ή απώλειας ενός πελάτη και ταυτόχρονα την ταξινόμησή του σε αυτές τις δύο κατηγορίες.

3.2. ΠΕΡΙΒΑΛΛΟΝ ΑΝΑΠΤΥΞΗΣ

Όλες οι διαδικασίες υλοποίησης των παραπάνω, έγιναν με τη χρήση της γλώσσας προγραμματισμού Python και την έκδοση 3.10.8, μέσα από τη χρήση του προγράμματος ανάπτυξης Visual Studio Code της εταιρείας Microsoft, καθώς και των απαραίτητων βιβλιοθηκών της Python όπως pandas, NumPy, Matplotlib, Seaborn, scikit-learn. Ο τύπος αρχείων που χρησιμοποιήθηκε είναι τα Jupyter Notebooks με την κατάληξη *.ipynb*.

3.2.1. Jupyter Notebooks

Το Jupyter Notebook είναι μια διαδικτυακή εφαρμογή που αναπτύχθηκε το Project Jupyter, το οποίο αποτελεί έναν μη-κερδοσκοπικό οργανισμό που ιδρύθηκε από τον Fernando Pérez. Οι βασικές λειτουργίες των jupyer notebooks είναι:

1. Κάθε Jupyter Notebook είναι ένα JSON αρχείο. Η JSON είναι μια μορφή δεδομένων που προέρχεται από την γλώσσα προγραμματισμού JavaScript. Χρησιμοποιεί κείμενο αναγνώσιμο από τον άνθρωπό για τη μετάδοση δεδομένων που περιέχουν πίνακες ή ζεύγη τιμών-χαρακτηριστικών.

2. Κάθε αρχείο Jupyter Notebook αποθηκεύεται με την κατάληξη *.ipynb*.
3. Είναι εύκολα στην κοινοποίηση και μεταφορά τους.
4. Είναι ένα ελεύθερο και δωρεάν λογισμικό.
5. Αποτελεί διαδραστική εφαρμογή, επιτρέποντας τους χρήστες να προσθέτουν σχόλια, και κεφαλίδες μεταξύ των γραμμών κώδικα.
6. Κάθε σημειωματάριο μπορεί να μετατραπεί σε διάφορες μορφές αρχείων όπως HTML, PDF κλπ. [39].

3.2.2. NumPy

Η βιβλιοθήκη NumPy της Python, αποτελεί το πιο χρήσιμο πακέτο για τους επιστήμονες της Python. Δημιουργήθηκε το 2005 με σκοπό τον γρήγορο υπολογισμό αριθμητικών δεδομένων [40]. Η βασική δύναμη της NumPy είναι η ικανότητά του να δημιουργεί και να χειρίζεται πίνακες n-διαστάσεων. Τα δεδομένα συχνά αναπαρίστανται σε ένα πλέγμα σειρών και στηλών που μοιάζει με μήτρα όπου κάθε γραμμή αντιπροσωπεύει μια παρατήρηση και κάθε στήλη μια μεταβλητή ή χαρακτηριστικό. Το γεγονός αυτό είναι ιδιαίτερα σημαντικό για τη δημιουργία μοντέλων μηχανικής και βαθιάς μάθησης [41].

3.2.3. pandas

Η βιβλιοθήκη pandas, είναι μια εξειδικευμένη βιβλιοθήκη της Python για ανάλυση δεδομένων, ειδικά για τεράστια σύνολα δεδομένων. Διαθέτει εύχρηστη λειτουργικότητα για ανάγνωση κι εγγραφή δεδομένων, αντιμετώπιση δεδομένων που λείπουν, μετατροπή του συνόλου δεδομένων κλπ. Ένα βασικό στοιχείο της pandas είναι η δυνατότητα ανακατάταξης και καθαρισμού των δεδομένων. Μπορεί να αποθηκεύσει και να διαχειριστεί πολυδιάστατους πίνακες όπως η NumPy με δομές όπως οι pandas series και pandas DataFrames.

DataFrame είναι μια δομή δεδομένων της pandas για την αποθήκευση και τη διαχείριση πινάκων δύο διαστάσεων. Είναι μια μορφή πίνακα που μοιάζει με τα υπολογιστικά φύλλα της Excel ή ενός πίνακα μιας σχεσιακής βάσης δεδομένων [42].

3.2.4. Matplotlib

Η Matplotlib είναι μια βιβλιοθήκη γραφικών σχεδίων που δημιουργήθηκε από τον John D. Hunter και χρησιμοποιείται για τη γραφική αναπαράσταση των δεδομένων. Μπορεί να χρησιμοποιηθεί για τη δημιουργία διαφορετικών τύπων γραφημάτων όπως γραμμικών γραφημάτων, γραφημάτων διασποράς, χαρτών θερμότητας, ραβδόγραμμα, διαγραμμάτων πίτας και τρισδιάστατων γραφημάτων. Μπορεί επίσης να υποστηρίξει κινούμενα σχέδια. Τα γραφήματα μπορούν να γίνουν εξαγωγή σε διάφορους τύπους όπως PNG, JPEG, SVG και PDF [43, 44].

3.2.5. Seaborn

Η Seaborn είναι κι αυτή μια βιβλιοθήκη γραφικών σχεδίων, σχεδιασμένη πάνω στην βιβλιοθήκη Matplotlib. Παρέχει πολλές λειτουργίες για τη σχεδίαση ελκυστικών γραφικών. Η Seaborn μπορεί να αλλάξει τις προεπιλογές της Matplotlib όπως τις χρωματικές παλέττες και να εκτελεί αυτόματα συναθροίσεις. Διαθέτει ενσωματωμένη υποστήριξη για δομές pandas series, pandas DataFrames και NumPy ndarrays [45, 46].

3.2.6. Scikit-learn

Η Scikit-learn είναι μία από τις πιο σημαντικές βιβλιοθήκες για μηχανική μάθηση. Διαθέτει πολλές λειτουργείες για διάφορους προγνωστικούς αλγορίθμους [47]. Είναι μια ελεύθερη βιβλιοθήκη επιτρέποντας την εύκολη και γρήγορη ενσωμάτωση μεθόδων μηχανικής μάθησης σε ένα κώδικα Python. Περιλαμβάνει ένα ευρύ φάσμα μεθόδων κι αλγορίθμων για ταξινόμηση, παλινδρόμηση, μείωση διαστάσεων, προ-επεξεργασία δεδομένων κλπ. Η βιβλιοθήκη χρησιμοποιείται ευρέως σε πολλές εμπορικές εφαρμογές και είναι επίσης μέρος πολλών ερευνητικών έργων και δημοσιεύσεων [47].

3.3. ΤΟ ΣΥΝΟΛΟ ΤΩΝ ΔΕΔΟΜΕΝΩΝ

3.3.1. Συλλογή των δεδομένων

Τα δεδομένα που χρησιμοποιήθηκαν κατά την ανάπτυξη της έρευνας είναι αναρτημένα στην ιστοσελίδα Kaggle (https://www.kaggle.com/datasets/datazng/telecom-company-churn-rate-call-center-data) με τον διακριτικό τίτλο «Telecom Company Churn Rate, Call Center Data». Το σύνολο των δεδομένων παρέχεται με τη μορφή αρχείου .xlsx.

3.3.2. Περιγραφή των δεδομένων

Το σύνολο των δεδομένων αφορά δεδομένα πελατών εταιρείας τηλεπικοινωνιών. Πιο συγκεκριμένα, το σύνολο των δεδομένων αποτελείται από 7043 γραμμές και 23 στήλες. Κάθε γραμμή στο σύνολο δεδομένων αντιπροσωπεύει έναν μοναδικό πελάτη ενώ, οι στήλες αντιπροσωπεύουν υπηρεσίες που αξιοποιεί ένας πελάτης, το μηνιαίο και το ετήσιο κόστος, τον διάστημα συνεργασίας του με την εταιρεία καθώς και αν εξακολουθεί να είναι πελάτης ή όχι. Στον παρακάτω πίνακα, παρουσιάζονται αναλυτικά όλα τα χαρακτηριστικά του συνόλου.

Πίνακας 5 *Χαρακτηριστικά του συνόλου δεδομένων.*

Χαρακτηριστικό	**Περιγραφή**	**Εύρος τιμών**
customerID	*Ο κωδικός του πελάτη.*	
Gender	*Το φύλλο του πλάτη.*	[Male, Female]
SeniorCitizen	*Αν ο πελάτης είναι ηλικιωμένος.*	[Yes, No]
Partner		[Yes, No]
Dependents	*Αν ο πελάτης έχει εξαρτώμενα άτομα.*	[Yes, No]
tenure	*Αριθμός μηνών που ο πελάτης έχει μείνει στην εταιρεία.*	[0, 72]
PhoneService	*Αν ο πελάτης έχει τηλεφωνική υπηρεσία.*	[Yes, No]
MultipleLines	*Αν ο πελάτης έχει πολλές γραμμές είτε όχι.*	[Yes, No, No Phone Service]
InternetService	*Τύπος παροχής διαδικτύου του πελάτη.*	[Fiber, DSL, No]

OnlineSecurity	*Αν ο πελάτης έχει ασφάλεια στο διαδίκτυο.*	[Yes, No, No internet service]
OnlineBackup	*Αν ο πελάτης έχει αντίγραφο ασφαλείας.*	[Yes, No, No internet service]
DeviceProtection	*Αν ο πελάτης έχει προστασία συσκευής.*	[Yes, No, No internet service]
TechSupport	*Είτε ο πελάτης έχει τεχνολογική υποστήρικη είτε όχι.*	[Yes, No, No internet service]
StreamingTV	*Είτε ο πελάτης έχει συνδρομητική τηλεόραση είτε όχι.*	[Yes, No, No internet service]
StremingMovies	*Είτε ο πελάτης έχει συνδρομητικές ταινίες είτε όχι.*	[Yes, No, No internet service]
Contract	*Τύπος συμβολαίου του πελάτη*	[One year, Two year, Month-to-month]
PaperlessBilling	*Αν ο πελάτης λαμβάνει έντυπο λογαριασμό.*	[Yes, No]
PaymentMethod	*Τρόπος πληρωμής του πελάτη*	[Bank transfer (automatic), Electronic check, Credit card (automatic),
MonthlyCharges	*Μηνιαίες χρεώσεις του πελάτη*	[18.3, 119]
TotalCharges	*Συνολικές χρεώσεις του πελάτη*	
numAdminTickets	*Αριθμός εισιτηρίων γενικής υποστήριξης.*	[0, 5]
numTechTickets	*Αριθμός εισιτηρίων τεχνολογικής υποστήριξης.*	[0, 9]
Churn	*Αν ο πελάτης έχει αποχωρήσει.*	[Yes, No]

3.4. ΠΡΟΕΠΕΞΕΡΓΑΣΙΑ ΤΩΝ ΔΕΔΟΜΕΝΩΝ

Στην συνέχεια έγινε η προ-επεξεργασία των δεδομένων με σκοπό την δημιουργία ενός συνόλου δεδομένων ικανό να χρησιμοποιηθεί στο επόμενο στάδιο, αυτό της διερευνητικής ανάλυσης. Κατά την διάρκεια της προ-επεξεργασίας, έγιναν οι απαραίτητοι έλεγχοι για τον εντοπισμό και την αντιμετώπιση ελλειπουσών τιμών αλλά και των διπλότυπων παρατηρήσεων. Επιπλέον, έγινε έλεγχος του τύπου δεδομένων των μεταβλητών και διαπιστώθηκε πως η μεταβλητή *TotalCharges* είχε την μορφή χρακτήρα (*object)* αντί για πραγματικό αριθμητικό αριθμό (*float)*.

3.5. ΔΙΕΡΕΥΝΗΤΙΚΗ ΑΝΑΛΥΣΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ

Στη συνέχεια έγινε η διερευνητική ανάλυση του πλέον προ-επεξεργασμένου συνόλου δεδομένων με σκοπό την κατανόηση των δεδομένων πριν από την χρήση τους για τη δημιουργία των μοντέλων μηχανικής μάθησης. Η εξερεύνηση των δεδομένων έγινε με επίκεντρο την μεταβλητή-στόχο *Churn* με σκοπό την εξαγωγή χρήσιμων περιγραφικών στατιστικών πληροφοριών και οπτικοποιήσεων κι έπειτα για την σωστή λήψη αποφάσεων κατά τη δημιουργία των μοντέλων.

Κατά τη διάρκεια της φάσης αυτής, αναλύθηκε η κατανομή των δεδομένων κάθε χαρακτηριστικού, ώστε να δούμε αν υπάρχει ισορροπία μεταξύ των κλάσεων. Έπειτα παράχθηκαν περιγραφικά στατιστικά όπως ο μέσος όρος, η ελάχιστη τιμή, η μέγιστη τιμή και η τυπική απόκλιση για τα αριθμητικά δεδομένα του συνόλου για την παρατήρηση της διακύμανσης και διασποράς των δεδομένων. Για την καλύτερη κατανόηση των αποτελεσμάτων χρησιμοποιήθηκαν διάφορα γραφήματα όπως ιστογράμματα, γραφήματα πίτας, γραφήματα ράβδων και γραφήματα κατανομής. Για να παραχθούν τα γραφήματα αυτά χρησιμοποιήθηκε η βιβλιοθήκη *matplotlib* και *seaborn.*

3.6. ΠΡΟΕΤΟΙΜΑΣΙΑ ΤΩΝ ΔΕΔΟΜΕΝΩΝ

Το στάδιο αυτό περιλαμβάνει τις διαδικασίες που ακολουθήθηκαν ώστε τα δεδομένα να λάβουν την κατάλληλη μορφή και να θεωρούνται ικανά για την χρήση τους από τα μοντέλο μηχανικής μάθησης. Το στάδιο αυτό θεωρείται ιδιαίτερα σημαντικό καθώς αποσκοπεί στη δημιουργία ενός συνόλου δεδομένων όπου τα δεδομένα μπορούν να χρησιμοποιηθούν από τα μοντέλα, αλλά και στην καλύτερη δυνατή προβλεπτική ισχύ.

3.6.1. Δημιουργία χαρακτηριστικών

Για την πληροφορία αν ένας πελάτης διαθέτει υπηρεσία διαδικτύου ή όχι, δημιουργήσαμε το γενικό χαρακτηριστικό *InternetService*. Η ενέργεια αυτή μας βοήθησε να μετατρέψουμε τα χαρακτηριστικά *OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV* και *StreamingMovies* σε δυαδικά, αντικαθιστώντας την τιμή *No internet service* που

εμπεριέχονταν μέσα σε αυτά, με την τιμή *No*, εφόσον η πληροφορία εξετάζεται πλέον από το νέο χαρακτηριστικό.

3.6.2. Κωδικοποίηση

Για όλες τις κατηγορικές δυαδικές μεταβλητές του συνόλου, έγινε η μετατροπή τους σε αριθμητικά δυαδικά χαρακτηριστικά, λαμβάνοντας τιμές 0 κι 1. Για τις μεταβλητές οι οποίες δεν είχαν δυαδική μορφή, ακολουθήθηκε η τεχνική κωδικοποίησης one-hot-encoding, για όπου κάθε τιμή της μεταβλητής δημιουργείται ένα νέο χαρακτηριστικό που λαμβάνει τιμές 0 κι 1.

3.6.3. Κλιμάκωση

Το εύρος τιμών για κάθε χαρακτηριστικό του συνόλου είναι διαφορετικό. Για τον λόγο έγινε η κλιμάκωση των αριθμητικών δεδομένων. Η μέθοδος κλιμάκωσης που ακολουθήθηκε είναι η τυποποίηση (standardization) όπου κανονικοποιεί τα δεδομένα ώστε να έχουν μέση τιμή 0 και τυπική απόκλιση 1. Για την επίτευξη της κλιμάκωσης χρησιμοποιήθηκε η κλάση *StandardScaler* της βιβλιοθήκης scikit-learn.

3.6.4. Διαχωρισμός των δεδομένων

Ως τελευταία διαδικασία της φάσης αυτής, ορίζεται ο διαχωρισμός των δεδομένων σε δεδομένα εκπαίδευσης και δοκιμής. Για τον διαχωρισμό αυτό χρησιμοποιήθηκε η συνάρτηση *train_test_spit* της βιβλιοθήκης scikit-learn. Τα δεδομένα εκπαίδευσης αντιστοιχούν στο 80% του συνόλου ενώ τα δεδομένα δοκιμής στο 20%. Το σύνολο εκπαίδευσης περιλαμβάνει όλα τα διαθέσιμα χαρακτηριστικά του συνόλου πλην του κωδικού πελάτη, ενώ τα δεδομένα δοκιμής περιλαμβάνουν την μεταβλητή-στόχο που θα προβλεφθεί κι αφορά την μεταβλητή *Churn.*

3.7. ΔΗΜΙΟΥΡΓΙΑ ΚΙ ΑΞΙΟΛΟΓΗΣΗ ΜΟΝΤΕΛΩΝ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Στο τελευταίο στάδιο της έρευνας, δημιουργήθηκε το μοντέλων μηχανικής μάθησης με σκοπό την πρόβλεψη αν ένας πελάτης θα φύγει η θα μείνει. Για την δημιουργία των μοντέλων χρησιμοποιήθηκαν διαδοχικά οι αλγόριθμοι Logistic Regression, XGBoost, Random Forest, Decision Tree και K-Nearest Neighbors (KNN) κι έπειτα έγινε η αξιολόγηση της απόδοσής τους με σκοπό την επιλογή του πιο αποδοτικού. Όλοι οι αλγόριθμοι-ταξινομητές είναι διαθέσιμοι στη βιβλιοθήκη της scikit-learn. Για την αξιολόγηση της απόδοσής τους χρησιμοποιήθηκαν οι μετρικές accuracy, f1-score, precision και recall, καθώς και διαγράμματα καμπύλης μάθησης.

Αρχικά, τα μοντέλα εκπαιδεύτηκαν χωρίς κάποια αλλαγή στις τιμές των υπεραπαραμέτρων κι έγινε σύγκριση της απόδοσής τους μεταξύ των αποτελεσμάτων στα δεδομένα εκπαίδευσης και στα δεδομένα δοκιμής.
Στη συνέχεια, τα μοντέλα εκπαιδεύτηκαν ξανά, αυτή τη φορά χρησιμοποιώντας τεχνικές όπως διασταυρωμένη επικύρωση αλλά και υπεραπαραμετροποίηση. Για την τεχνική της διασταυρωμένης επικύρωσης χρησιμοποιήσαμε την *StratiedKFold,* με τιμή *k=5 (k υποσύνολα)*.

Η συγκεκριμένη τεχνική επιλέχθηκε ώστε να διατηρήσει την αναλογία στις κλάσεις για κάθε υποσύνολο. Ταυτόχρονα, η τεχνική επιλογής υπεραπαραμέτρων που επιλέχθηκε είναι η *GridSearchCV*, ώστε να επιλέξει τον καλύτερο συνδυασμό τιμών των υπερπαραμέτρων από το πλέγμα. Παρακάτω, παρουσιάζεται το πλέγμα των υπερπαραμέτρων και των τιμών τους για κάθε αλγόριθμο:

Πλέγμα υπερπαραμέτρων Logistic Regression

```
{'C': [0.001, 0.01, 0.1, 1.0],
'class_weight': [None, 'balanced'],
'dual': [False],
'fit_intercept': [True, False],
'intercept_scaling': [1],
'max_iter': [100, 200, 300],
```

```
'multi_class': ['auto', 'ovr', 'multinomial'],
'penalty': ['l1', 'l2'],
'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
'tol': [0.0001, 0.001, 0.01],
'warm_start': [False, True]}
```

Πλέγμα υπερπαραμέτρων XGBoost

```
{'learning_rate': [0.01, 0.1, 0.2],
'n_estimators': [100, 200, 300],
'max_depth': [3, 4, 5],
'min_child_weight': [1, 3, 5],
'subsample': [0.8, 0.9, 1.0],
'colsample_bytree': [0.8, 0.9, 1.0]}
```

Πλέγμα υπερπαραμέτρων Random Forest

```
{'n_estimators': [100, 200, 300],
'max_depth': [None, 10, 20, 30],
'min_samples_split': [2, 5, 10],
'max_features': ['sqrt', 'log2', None],
'bootstrap': [True, False],
'class_weight': [None, 'balanced', 'balanced_subsample'],
'min_impurity_decrease': [0.0, 0.1, 0.2]}
```

Πλέγμα υπερπαραμέτρων Decision Tree

```
{'criterion': ['gini', 'entropy'],
'splitter': ['best', 'random'],
'max_depth': [None, 10, 20, 30],
'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4],
'max_features': ['sqrt', 'log2', None],
'min_impurity_decrease': [0.0, 0.1, 0.2]}
```

Πλέγμα υπερπαραμέτρων K-Nearest Neighbors

```
{'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
'leaf_size': [10, 20, 30, 40, 50],
'metric': ['minkowski', 'euclidean', 'manhattan', 'chebyshev',
'hamming', 'canberra', 'braycurtis'],
'n_neighbors': [1, 3, 5],
'p': [1, 2, 3],
'weights': ['uniform', 'distance']}
```

Εφόσον τα μοντέλα εκπαιδεύτηκαν, έγινε εξαγωγή των αποτελέσματα των μετρικών αξιολόγησης και για τα δεδομένα εκπαίδευσης και για τα δεδομένα δοκιμής. Επιπλέον, για τις μετρικές *accuracy* κι *f1-core* δημιουργήσαμε και γραφήματα καμπύλης μάθησης.

Τέλος, αφού τα αποτελέσματα αξιολογήθηκαν, έγινε η επιλογή του καλύτερου αλγορίθμου και παρήχθησαν το γράφημα σημαντικότητας χαρακτηριστικών, ο πίνακας σύγχυσης, το γράφημα χαρακτηριστικών λειτουργίας δέκτη, καθώς και το γράφημα SHAP (SHapley Additive exPlanations).

ΚΕΦΑΛΑΙΟ 4: ΥΛΟΠΟΙΗΣΗ & ΑΠΟΤΕΛΕΣΜΑΤΑ

4.1. ΠΡΟΕΞΕΡΓΑΣΙΑ ΤΩΝ ΔΕΔΟΜΕΝΩΝ

Στο πρώτο στάδιο της μεθοδολογίας, ελέγχουμε αν υπάρχουν πρώτα ελλείπουσες τιμές.

```
data.isnull().sum()
```

```
Output:
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
numAdminTickets     0
numTechTickets      0
```

```
Churn                0
```

Έπειτα ελέγχουμε για διπλότυπες τιμές.

```
data.duplicated().sum()
```

```
Output:
0
```

Έπειτα, χρειάστηκε να αλλάξουμε τον τύπο του χαρακτηριστικού *TotalCharges* σε αριθμητικό, καθώς εμφανιζόταν ως *object.* Η εντολή αυτή, μας έδειξε πως τελικά υπήρχαν ελλείπουσες τιμές με τη μορφή κενής σειράς.

```
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'])
```

```
Output:
ValueError: Unable to parse string " " at position 488
```

Εντοπίσαμε τις γραμμές στις οποίες παρουσιαζόταν το παρακάτω πρόβλημα.

```
data[data['TotalCharges'] == ' ']
```

Αντικαταστήσαμε τις κενές σειρές με την τιμή 0.

```
data['TotalCharges'] = data['TotalCharges'].replace(' ', 0)
```

Μετά μπορέσαμε και μετατρέψαμε τον τύπο του χαρακτηριστικού *TotalCharges* σε αριθμητικό τύπο και μετατρέψαμε επίσης τον τύπο της χαρακτηριστικού *SeniorCitizen* σε *object*.

```
data['SeniorCitizen'] = data['SeniorCitizen'].astype(str)
```

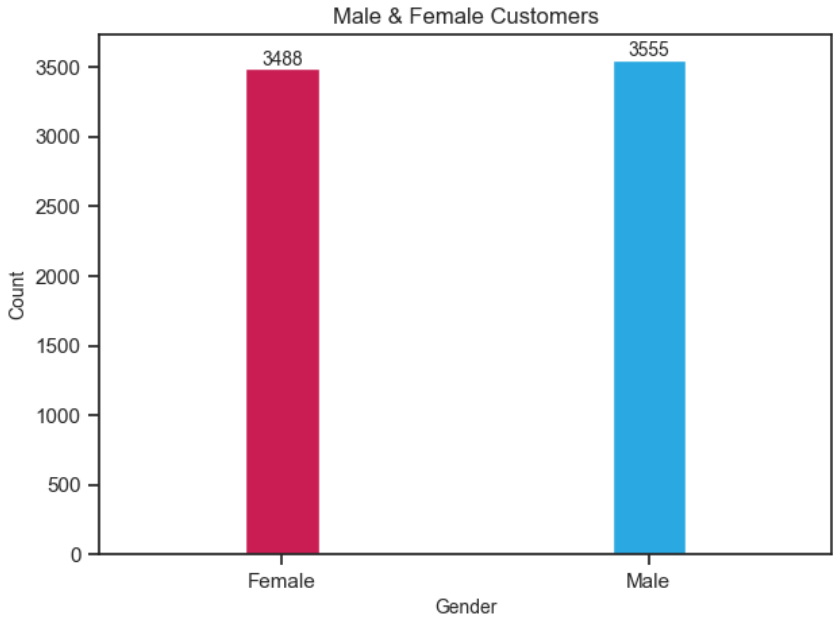
```
data['SeniorCitizen'] = data['SeniorCitizen']\
                            .replace({'1':'Yes', '0':'No'})
```

Τέλος, δημιουργήσαμε ένα καινούργιο χαρακτηριστικό *year* το οποίο περιέχει τα χρόνια τα οποία ο πελάτης είναι στην εταιρεία.

```
data['year'] = data['tenure'] / 12 + 1
data['year'] = data['year'].astype(int)
```

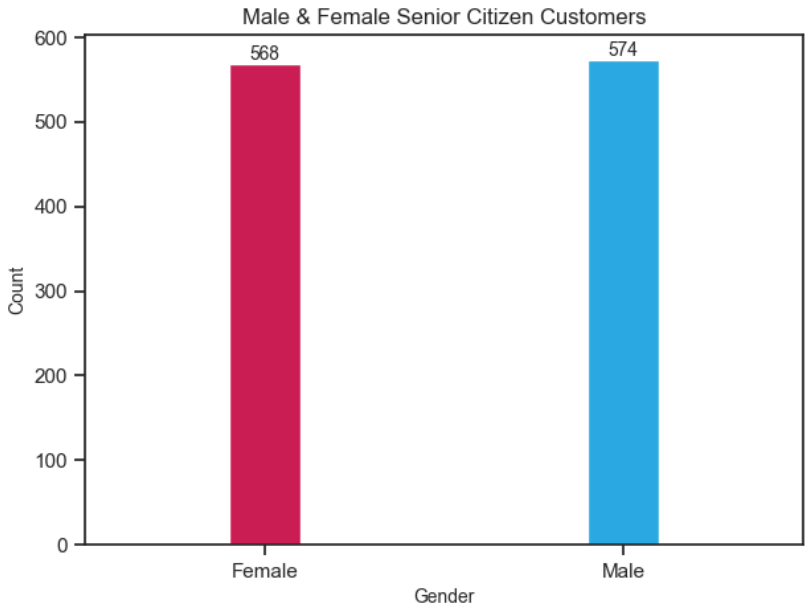
4.2. ΔΙΕΡΕΥΝΗΤΙΚΗ ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ

Στο επόμενο σκέλος της μεθοδολογίας έγινε μια διερευνητική ανάλυση των δεδομένων με σκοπό την εξαγωγή χρήσιμων πληροφοριών για το σύνολο των δεδομένων και κάποιων περιγραφικών και αθροιστικών αποτελεσμάτων. Για την παροχή αυτών, εισαγάγαμε τις βιλιοθήκες pandas, NumPy, Matplotlib και Seaborn.



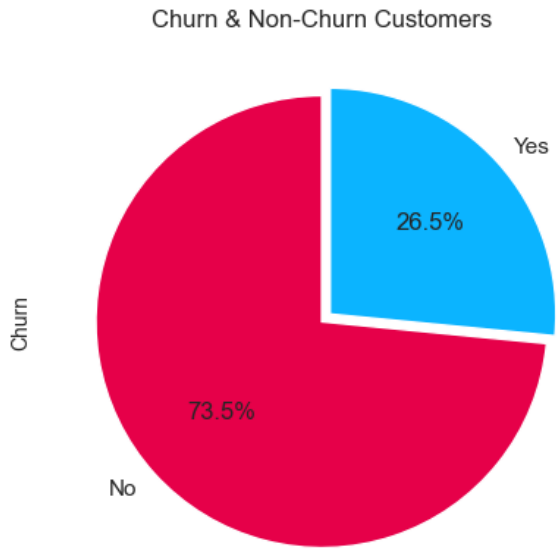
Σχήμα 15 *Πελάτες ανά φύλο.*

Το σύνολο των δεδομένων αποτελείται από 3488 γυναίκες και 3555 άνδρες.



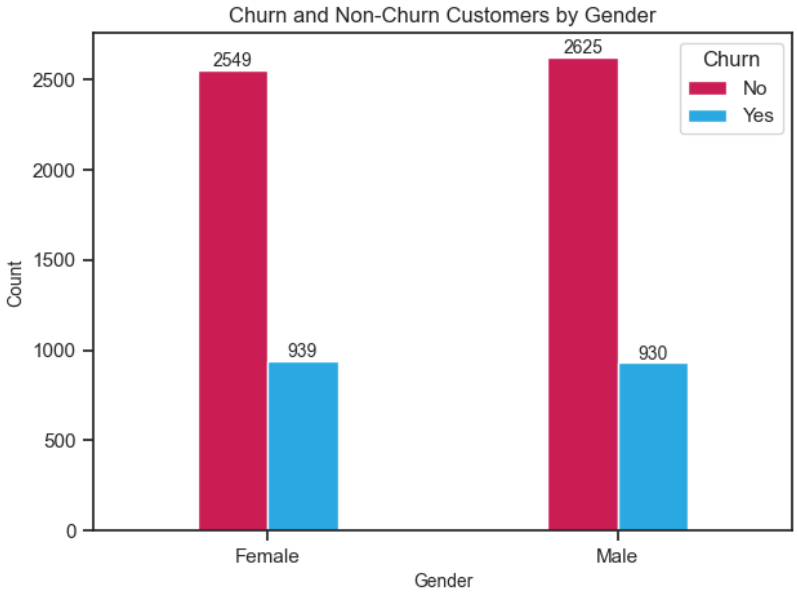
Σχήμα 16 *Ηλικιωμένοι πελάτες ανά φύλο.*

Από τα σύνολα αυτά, οι ηλικιωμένοι άνδρες είναι 574 και οι ηλικιωμένες γυναίκες είναι 568.



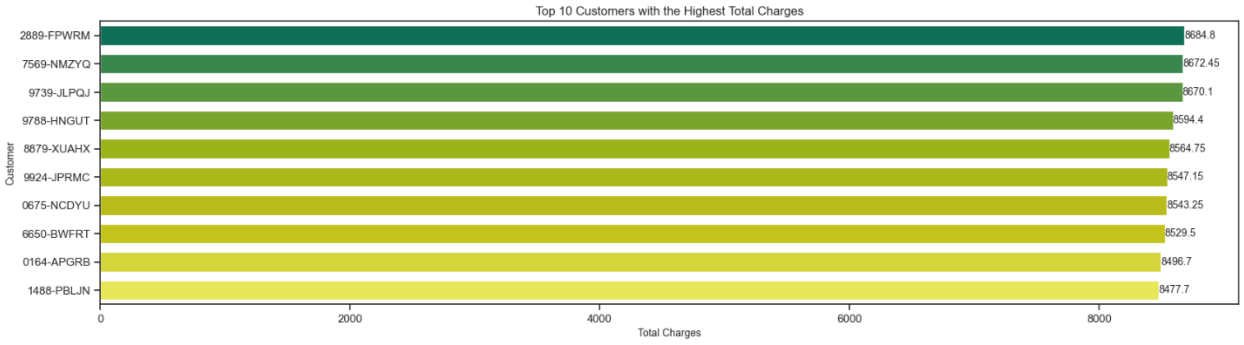
Σχήμα 17 *Πελάτες και πελάτες που αποχώρησαν.*

Το 73,5% του συνόλου αποτελούν ακόμα πελάτες της εταιρείας, ενώ 26,5% δεν αποτελούν πλέον πελάτες τις εταιρείας.



Σχήμα 18 *Πελάτες και πελάτες που αποχώρησαν ανά φύλο.*

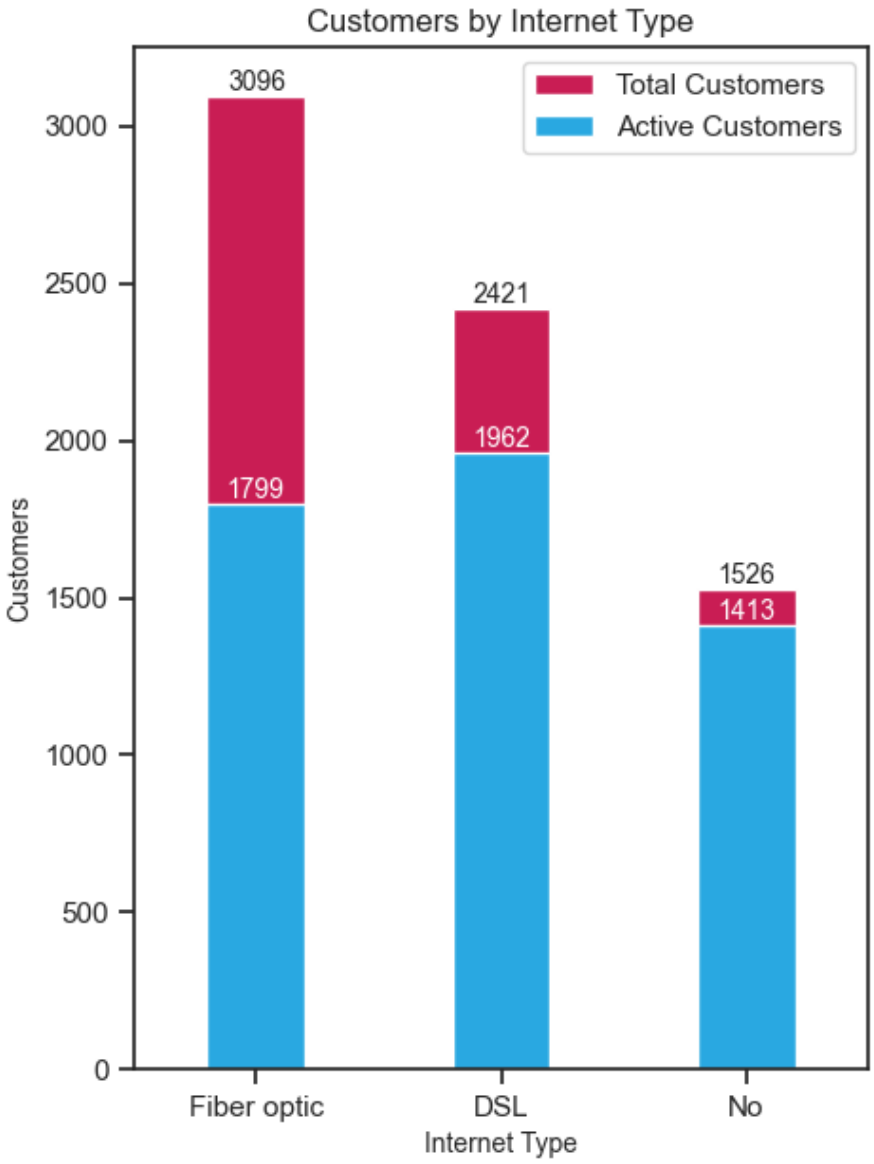
Από τους πελάτες που έχουν αποχωρήσει, οι 939 είναι γυναίκες και οι 930 άντρες.



Σχήμα 19 *Οι δέκα πελάτες με τις μεγαλύτερες συνολικές χρεώσεις.*

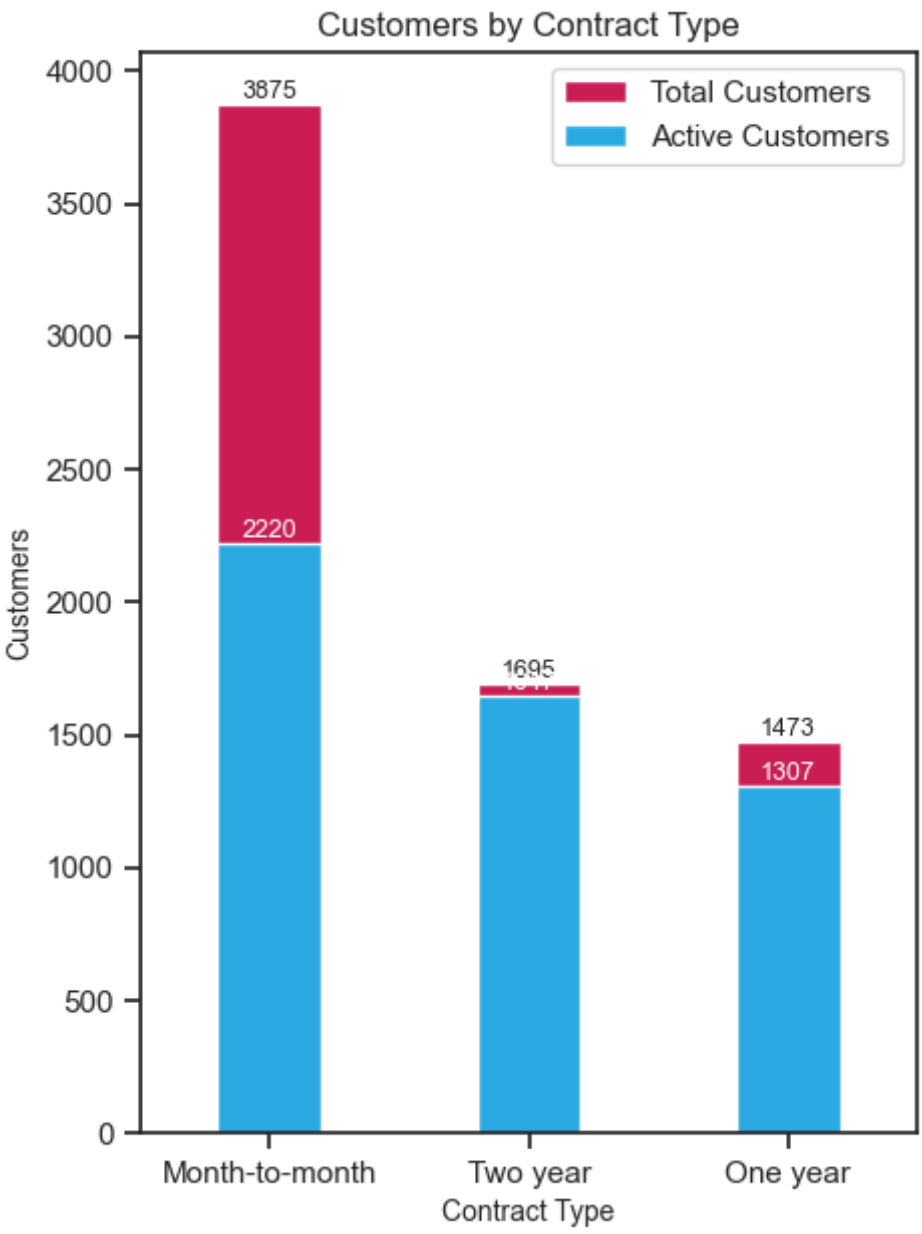
Στο παραπάνω διάγραμμα παρουσιάζονται οι 10 πελάτες με τις μεγαλύτερες συνολικές χρεώσεις. Οι 4 από τους 10 αποτελούν και πελάτες με τις μεγαλύτερες μηνιαίες χρεώσεις, ενώ οι

9 στους 10 αποτελούν και πελάτες με τη μεγαλύτερη παραμονή στην εταιρεία που είναι 72 μήνες.



Σχήμα 20 *Πελάτες ανά κατηγορία διαδικτύου.*

Στο **Σχήμα 20**, φαίνεται πως ο τύπος διαδικτύου «DSL» είναι αυτός που λαμβάνουν ως υπηρεσία οι περισσότεροι ενεργοί πελάτες της εταιρείας. Στο σύνολο των πελατών, δηλαδή των ενεργών και των ανενεργών πελατών, ο πιο δημοφιλής τύπος διαδικτύου είναι η οπτική ίνα. Σχεδόν το 20% του συνόλου δεν λαμβάνει διαδίκτυο ως υπηρεσία.



Σχήμα 21 *Πελάτες ανά τύπο συμβολαίου.*

Οι περισσότεροι πελάτες έχουν συμβόλαιο μήνα-μήνα ενώ λιγότεροι πελάτες προτιμούν ετήσια συμβόλαια.

Στον παρακάτω πίνακα συγκεντρώνονται όλες οι πελάτες βάση των υπηρεσιών διαδιτκύου που παρέχει η εταιρεία. Μερικές από τις υπηρεσίες αυτές είναι η προστασία συσκευής, ασφάλεια διαδικτύου, συνδρομητική τηλεόραση, τεχνική υποστήριξη κλπ.

Πίνακας 6 *Πελάτες ανά υπηρεσία διαδικτύου.*

index	InternetServices	No	No internet service	Yes
0	DeviceProtection	3095	1526	2422
1	OnlineBackup	3088	1526	2429
2	OnlineSecurity	3498	1526	2019
3	StreamingMovies	2785	1526	2732
4	StreamingTV	2810	1526	2707
5	TechSupport	3473	1526	2044

Βάση του **Πίνακας 6**, το σύνολο των πελατών που δεν χρησιμοποιούν το διαδίκτυο ως υπηρεσία είναι 1526. Η πιο δημοφιλής υπηρεσία είναι η «StreamingMovies» με 2732 πελάτες ενώ δεύτερη είναι η «StreamingTV» με 2707 πελάτες. Η υπηρεσία που δεν χρησιμοποιούν πολλοί είναι η ασφάλεια διαδικτύου.

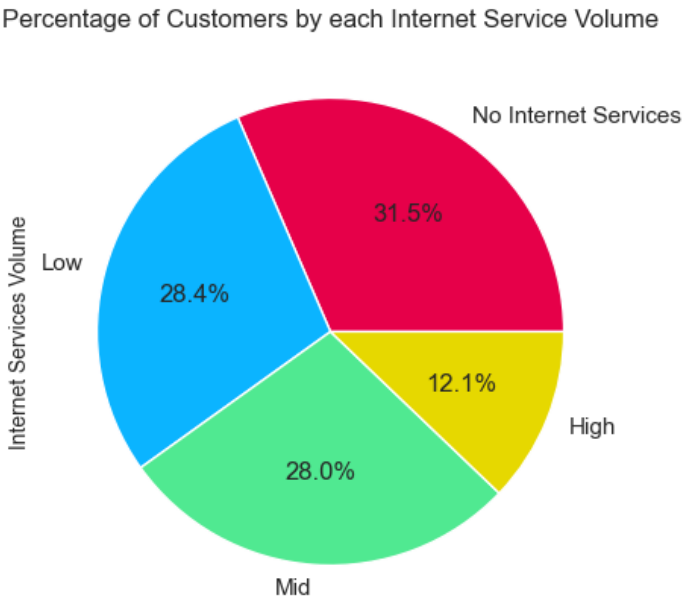
Αντίστοιχος πίνακας εμφανίζεται παρακάτω, αλλά μόνο για τους ενεργούς πελάτες της εταιρείας.

Πίνακας 7 *Ενεργοί πελάτες ανά υπηρεσία διαδικτύου.*

index	InternetServices	No	No internet service	Yes
0	DeviceProtection	1884	1413	1877
1	OnlineBackup	1855	1413	1906
2	OnlineSecurity	2037	1413	1724

3	StreamingMovies	1847	1413	1914
4	StreamingTV	1868	1413	1893
5	TechSupport	2027	1413	1734

Στην περίπτωση αυτή, η υπηρεσία «StreamingMovies» παραμένει η πιο δημοφιλής με 1914 πελάτες, ενώ δεύτερη είναι η «OnlineBackup» με 1906 συνδρομητές. Το σύνολο των πελατών που έχει το internet ως υπηρεσία αλλά δεν αξιοποιεί καμία υπηρεσία διαδικτύου είναι 693 από τους οποίους ενεργοί πελάτες τις εταιρείας είναι οι 331.



Σχήμα 22 *Ποσοστό πελατών βάση της έντασης υπηρεσιών διαδικτύου.*

Στην συνέχεια, έγινε κατηγοριοποίηση των πελατών βάση της έντασης των υπηρεσιών διαδικτύου που λαμβάνουν ως υπηρεσία. Οι πελάτες με 5 έως 6 υπηρεσίες διαδικτύου κατατάσσονται στην κατηγορία «High», οι πελάτες με 3 έως 4 υπηρεσίες διαδικτύου κατατάσσονται στην κατηγορία «Mid», οι πελάτες με 1 έως 2 υπηρεσίες διαδικτύου κατατάσσονται στην κατηγορία «Low» και οι πελάτες που δεν λαμβάνουν υπηρεσίες διαδικτύου κατατάσσονται στην κατηγορία «No Internet Services». Στο παραπάνω διάγραμμα πίτας

φαίνεται πως σχεδόν το 57% των πελατών ανήκουν στις κατηγορίες «Low» και «Mid» ενώ μόνο το 12,1% λαμβάνει παραπάνω από 4 υπηρεσίες διαδικτύου.

Πίνακας 8 *Περιγραφικά στατιστικά μηνιαίων χρεώσεων ανά κατηγορία έντασης υπηρεσιών διαδικτύου.*

InternetServicesVolume	**MonthlyCharges**			
	min	**max**	**std**	**mean**
High	52.50	118.75	17.80	94.56
Mid	37.70	97.65	19.46	83.42
Low	28.45	108.80	19.12	69.11
No Internet Services	18.25	77.90	19.90	32.79

Στον παραπάνω πίνακα, παρουσιάζεται ο μέσος όρος χρεώσεων για κάθε κατηγορία. Οι μηνιαίες χρεώσεις των πελατών από 5 υπηρεσίες διαδικτύου και πάνω κυμαίνονται από 52,5 έως 118,75 με μέσο όρο 94,56 και τυπική απόκλιση 17,8 μονάδες.

Πίνακας 9 *Ποσοστό πελατών που αποχώρησαν ανά κατηγορία έντασης υπηρεσιών διαδικτύου.*

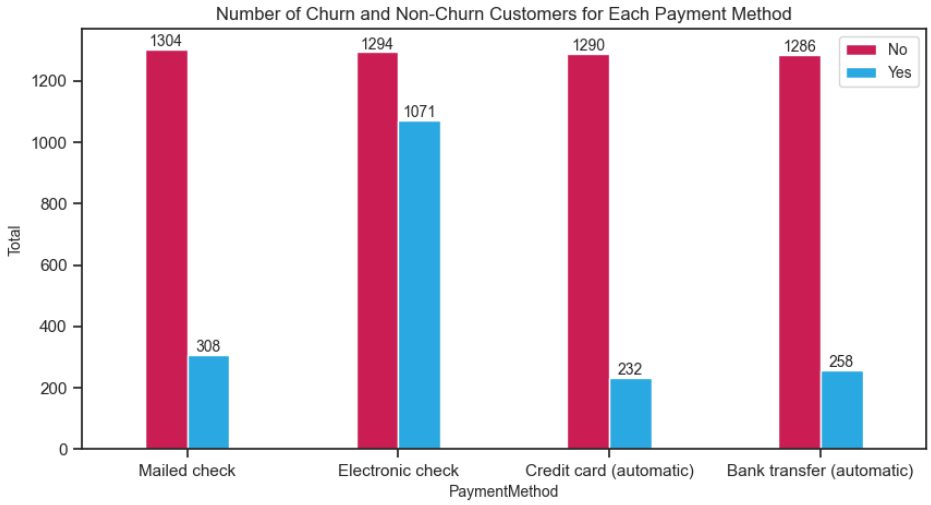
	InternetServicesVolume	**Churn**	**Percentage**
0	High	Yes	10.06
1	High	No	89.94
2	Mid	Yes	25.18
3	Mid	No	74.82
4	Low	Yes	40.62
5	Low	No	59.38
6	No Internet Services	Yes	21.41
7	No Internet Services	No	78.59

Στον πίνακα, βλέπουμε πως όσο λιγότερες είναι οι υπηρεσίες που λαμβάνει ο πελάτης, τόσο περισσότερο είναι και το ποσοστό αποχώρησης του. Πελάτες στην κατηγορία «High» φαίνεται να είναι πιο ικανοποιημένοι από τους πελάτες στις υπόλοιπες κατηγορίες με το ποσοστό των πελατών που έχουν διακόψει την συνεργασία τους με την εταιρεία να είναι σχεδόν το 10% και με το ποσοστό αυτό να αυξάνεται στις χαμηλότερες κατηγορίες.

Πίνακας 10 *Πελάτες ανά μέθοδο πληρωμής.*

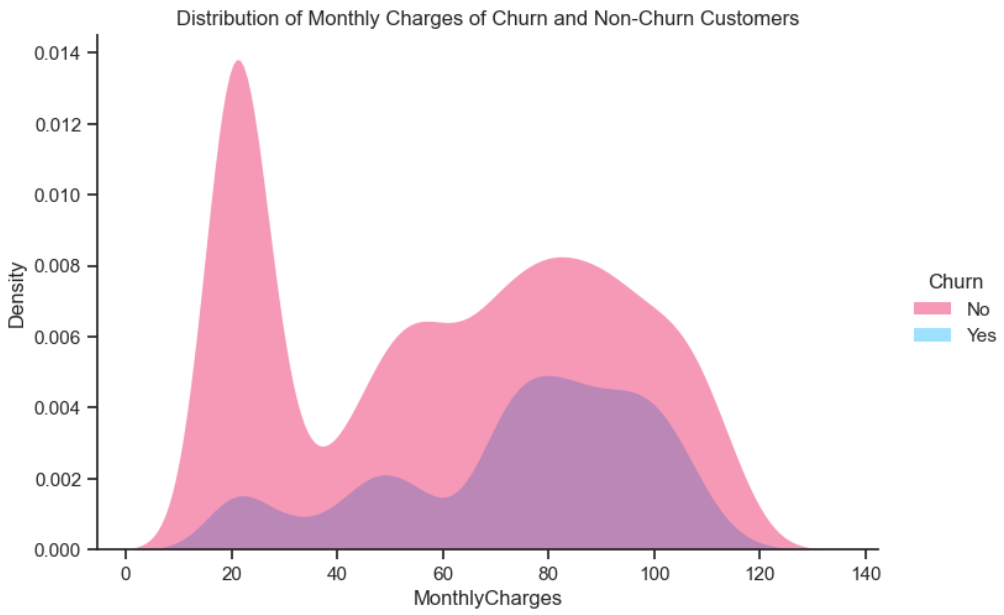
	PaymentMethod	Total
0	Electronic check	2365
1	Mailed check	1612
2	Bank Transfer (automatic)	1544
3	Credit card (automatic)	1522

Στο διάγραμμα, παρουσιάζονται πιο αναλυτικά κατηγοριοποιημένοι με βάση τον τρόπο πληρωμής αλλά και με το αν είναι πελάτες της εταιρείας ή όχι.



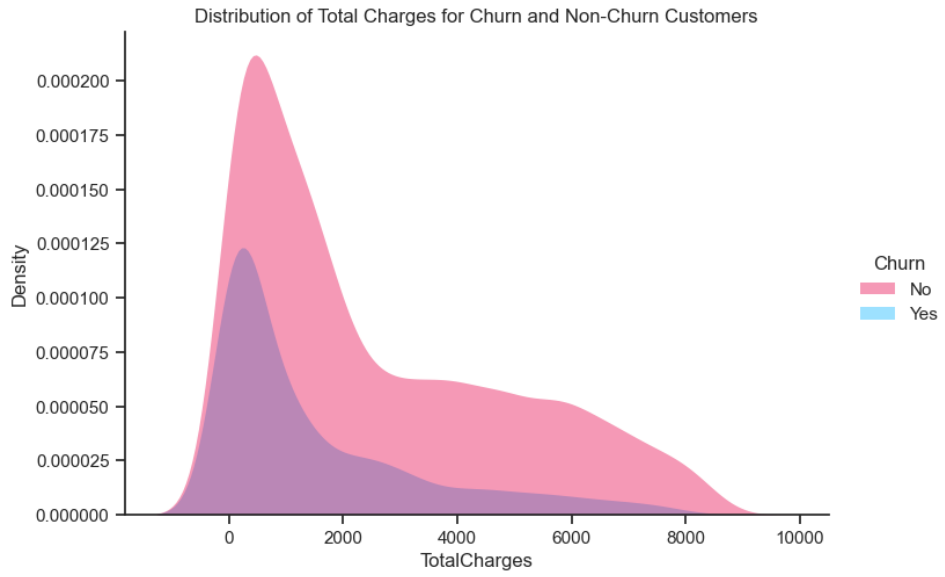
Σχήμα 23 *Αριθμός πελατών για κάθε μέθοδο πληρωμής.*

Στο διάγραμμα, παρουσιάζεται η κατανομή των μηνιαίων χρεώσεων για τους πελάτες της εταιρείες αλλά και για προηγούμενους πελάτες. Φαίνεται πως οι περισσότεροι από τους πελάτες που αποχώρησαν από την εταιρεία να συσσωρεύονται στο εύρος χρεώσεων 60-120 ενώ αντίθετα, η διασπορά των πελατών της εταιρείας είναι μεγάλη, με ένα μεγάλο μέρος αυτών να ανήκουν στο εύρος χρεώσεων 19-25.



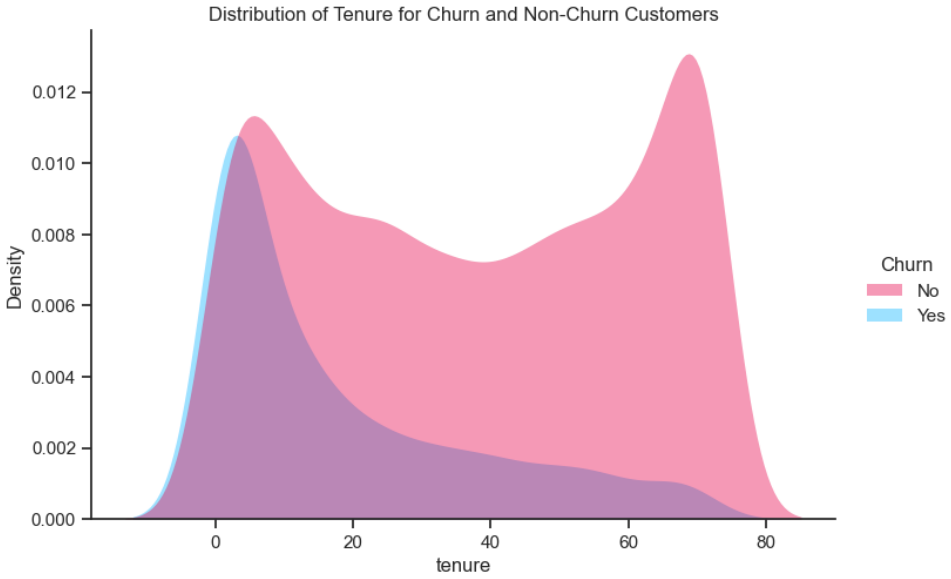
Σχήμα 24 *Κατανομή μηνιαίων χρεώσεων πελατών.*

Η κατανομή των ετησίων χρεώσεων, όπως φαίνεται στο διάγραμμα πως οι περισσότεροι πελάτες της εταιρείας εμφανίζονται στις χαμηλές ετήσιες χρεώσεις, κάτι που επίσης παρουσιάζεται και στους πελάτες που έχουν αποχωρήσει από την εταιρεία.



Σχήμα 25 *Κατανομή ετήσιων χρεώσεων πελατών.*

Τέλος, η κατανομή της διάρκειας όπου κάποιος είναι πελάτης της εταιρείας φαίνεται στο **Σχήμα 26**. Παρατηρείται πως οι πελάτες της εταιρείας διαμοιράζονται σε όλο το εύρος βάση της διάρκειας που είναι πελάτες της εταιρείας. Για τους πελάτες που έχουν αποχωρήσει από την εταιρεία, φαίνεται πως οι περισσότεροι αποχωρούν σύντομα, εντός του διαστήματος 10-20 μηνών.



Σχήμα 26 *Κατανομή διάρκειας παραμονής των πελατών.*

4.3. ΠΡΟΕΤΟΙΜΑΣΙΑ ΤΩΝ ΔΕΔΟΜΕΝΩΝ

Στο σημείο της μεθοδολογίας αυτής, ακολούθησαν διεργασίες κι ενέργειες προκειμένου το σύνολο τον δεδομένων να έχει την κατάλληλη μορφή πριν την χρήση του από τα μοντέλα μηχανικής μάθησης. Αυτές οι διεργασίες αφορούν την δημιουργία χαρακτηριστικών, την κωδικοποίηση και την κλιμάκωση των τιμών, καθώς και τον διαχωρισμό των δεδομένων σε δεδομένα εκπαίδευσης και δοκιμής.

Αρχικά, αφαιρέσαμε το χαρακτηριστικό *customerID* καθώς δεν προσδίδει κάποια προβλεπτική αξιά.

```
data.drop('customerID', axis=1, inplace=True)
```

Έπειτα αντικαταστήσαμε τις τιμές *No internet service* και *No phone service* με την τιμή *No.*

```
data.replace({"No internet service":"No"}, inplace=True)
data.replace({"No phone service":"No"}, inplace=True)
```

Επιλέξαμε τα χαρακτηριστικά στο οποία θα γίνει κωδικοποίηση one-hot-encoding.

```
columns_to_encode = [
    'InternetService',
    'Contract',
    'PaymentMethod']

data_encoded = pd.get_dummies(data, columns=columns_to_encode)
```

Μετατρέψαμε τα δυαδικά κατηγορικά χαρακτηριστικά σε δυαδικά αριθμητικά.

```
data_encoded["gender"] = data_encoded['gender']\
                         .replace({'Female':0, 'Male':1})
```

```
binary_cols = [
               'SeniorCitizen', 'Partner', 'Dependents',
               'PhoneService', "MultipleLines",
               "OnlineSecurity", "OnlineBackup",
               "DeviceProtection", "TechSupport", "StreamingTV",
               "StreamingMovies", 'PaperlessBilling', 'Churn'
               ]

for col in binary_cols:
    data_encoded[col] = data_encoded[col]\
                        .replace({'Yes':1, 'No':0})
```

Αλλάξαμε τον τύπο των χαρακτηριστικών σε πραγματικό (float).

```
for col in data_encoded.columns:
    if data_encoded[col].dtype != 'float':
        data_encoded[col] = data_encoded[col].astype('float')
```

Αντιστρέψαμε τις τιμές του χαρακτηριστικού *InternetService_No* που προέξυψε από την κωδικοποίηση one-hot-encoding του αρχικού χαρακτηριστικού *InternetService* και το μετονομάσαμε σε *InternetService*.

```
data_encoded["InternetService_No"] =
          data_encoded["InternetService_No"].replace({0:1, 1:0})

data_encoded.rename(columns={
                        'InternetService_No':'InternetService'}
                        ,inplace=True)
```

Χωρίσαμε τα δεδομένα σε δεδομένα εκπαίδευσης και δεδομένα δοκιμής

```
X = data_encoded.drop(columns='Churn')
y = data_encoded['Churn']

X_train, X_test, y_train, y_test = train_test_split(X, y,
               test_size=.2, random_state=123, stratify=y)
```

Κάναμε την κλιμάκωση στις τιμές των αριθμητικών χαρακτηριστικών χρησιμοποιώντας την μέθοδο standardization και την κλάση *StandardScaler* από την βιβλιοθήκη scikit-learn.

```
stc = StandardScaler()
X_train_scaled = stc.fit_transform(X_train[numeric_columns])
X_test_scaled = stc.fit_transform(X_test[numeric_columns])
X_train_scaled = pd.DataFrame(X_train_scaled,
columns=numeric_columns)
X_test_scaled = pd.DataFrame(X_test_scaled,
columns=numeric_columns)
```

Και τέλος ενώσαμε τα κλιμακούμενα αριθμητικά δεδομένα με τα αριθμητικά δυαδικά δεδομένα.

```
X_train = pd.concat([X_train, X_train_scaled], axis=1)
X_test = pd.concat([X_test, X_test_scaled], axis=1).
```

4.4. ΔΗΜΙΟΥΡΓΙΑ ΜΟΝΤΕΛΩΝ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Το τελικό στάδιο της μεθοδολογίας αφορά τη δημιουργία και την αξιολόγηση της απόδοσης του μοντέλου. Για την αξιολόγηση του μοντέλου εξήχθησαν οι μετρικές *f1-score, ακρίβεια, recall* και *precision.* Για κάθε αλγόριθμο, παράχθηκε και το γράφημα καμπύλης μάθησης ενώ για τον αλγόριθμο με την καλύτερη απόδοση, παράχθηκαν και τα γραφήματα καμπύλης χαρακτηριστικών δέκτη (ROC-AUC Curve) και shap.

Αρχικά θα δημιουργήσουμε την μεταβλητή *cv* όπου θα περιλαμβάνει την τεχνική *StratifiedKFold* για την διασταυρωμένη επικύρωση. Κατά την εκπαίδευση των μοντέλων το σύνολο των δεδομένων δοκιμής θα χωριστεί σε 5 υποσύνολα όπου κάθε φορά θα χρησιμοποιείται ένα υποσύνολο ως δεδομένα εκπαίδευσης και ένα υποσύνολο ως δεδομένα δοκιμής.

```
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

4.4.1. Χρήση του Αλγορίθμου Λογιστικής Παλινδρόμησης (Logistic Regression)

Για να μπορέσουμε να χρησιμοποιήσουμε τον αλγόριθμο Logistic Regression, εισάγουμε την κλάση *LogisticRegression* από τη βιβλιοθήκη *sklearn.linear_model* και δημιουργούμε ένα αντικείμενο της κλάσης αυτής. Στη συνέχεια εφαρμόζουμε τον αλγόριθμο στα δεδομένα εκπαίδευσης κι αφού εκπαιδευτεί το μοντέλο, τότε κάνουμε την πρόβλεψη στα δεδομένα δοκιμής. Τα αποτελέσματα από τη χρήση του αλγορίθμου λογιστικής παλινδρόμησης παρουσιάζονται στον **Πίνακας 11**.

```
log_regression = LogisticRegression()

log_regression.fit(X_train, y_train)

y_train_pred = log_regression.predict(X_train)

y_pred = log_regression.predict(X_test)
```

Πίνακας 11 *Αποτελέσματα αλγορίθμου λογιστικής παλινδρόμησης.*

Model	Accuracy	F1-Score	Recall	Precision
Logistic Regression	0.853797	0.713091	0.684491	0.744186

4.4.2. Χρήση του αλγορίθμου XGBoost.

Για την χρήση του αλγορίθμου XGBoost εισάγουμε από την βιβλιοθήκη *xgboost* την κλάση *xgb.* Έπειτα δημιουργούμε ένα αντικείμενο της κλάσης xgb.XGBClassifier() και με τον ίδιο τρόπο όπως και στη λογιστική παλινδρόμηση, εφαρμόζουμε τον αλγόριθμο στα δεδομένα εκπαίδευσης κι έπειτα κάνουμε την πρόβλεψη στα δεδομένα δοκιμής. Τα αποτελέσματα παρουσιάζονται στον **Πίνακας 12**.

```
xgb = xgb.XGBClassifier()

xgb.fit(X_train, y_train)

y_train_pred = xgb.predict(X_train)

y_pred = xgb.predict(X_test)
```

Πίνακας 12 *Αποτελέσματα αλγορίθμου XGBoost.*

Model	Accuracy	F1-Score	Recall	Precision
XGBoost	0.859474	0.728767	0.711229	0.747191

4.4.3. Χρήση του αλγορίθμου Τυχαίου Δάσους (Random Forest).

Για τη χρήση του αλγορίθμου τυχαίου δάσους εισάγουμε από την βιβλιοθήκη *sklearn.ensemble* την κλάση *RandomForestClassifier*. Έπειτα, δημιουργούμε ένα αντικείμενο της κλάσης. Στον αλγόριθμο τυχαίου δάσους η βασική υπερπαράμετρος είναι οι *n_estimators*. Με αυτή καθορίζουμε τον αριθμό των δέντρων στο τυχαίο δάσος. Εφαρμόζουμε τον αλγόριθμο στα δεδομένα εκπαίδευσης κι τέλος, κάνουμε την πρόβλεψη στα δεδομένα δοκιμής. Ο **Πίνακας 13**. παρουσιάζει τα αποτελέσματα του αλγορίθμου τυχαίου δάσους.

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)

rf.fit(X_train, y_train)

y_train_pred = rf.predict(X_train)

y_pred = rf.predict(X_test)
```

Πίνακας 13 *Αποτελέσματα αλγορίθμου Τυχαίου Δάσους (Random Forest).*

Model	**Accuracy**	**F1-Score**	**Recall**	**Precision**
Random Forest	0.853797	0.703170	0.652406	0.7625

4.4.4. Χρήση του αλγορίθμου Δέντρων Απόφασης (Decision Tree).

Για τη χρήση του αλγορίθμου δέντρων απόφασης χρειάζεται να εισάγουμε την κλάση *DecisionTreeClassifier* από την βιβλιοθήκη *sklearn.tree.* Με τον ίδιο τρόπο όπως και προηγουμένως, ο αλγόριθμος εφαρμόζεται στα δεδομένα εκπαίδευσης κι αξιολογείται στα δεδομένα δοκιμής. Στον **Πίνακας 14**, εμπεριέχονται τα αποτελέσματα του αλγορίθμου δέντρων απόφασης.

```
dt = DecisionTreeClassifier(random_state=42)

dt.fit(X_train, y_train)

y_train_pred = dt.predict(X_train)

y_pred = dt.predict(X_test)
```

Πίνακας 14 *Αποτελέσματα αλγορίθμου Δέντρων Απόφασης (Decision Tree).*

Model	Accuracy	F1-Score	Recall	Precision
Decision Trees	0.802696	0.638961	0.657754	0.621212

4.4.5. Χρήση του αλγορίθμου Κ-Κοντινότεροι Γείτονες (K-Nearest Neighbors).

Για την χρήση του αλγορίθμου Κ-Κοντινότεροι Γείτονες, εισάγουμε τον ταξινομητή KNeighborsClassifier από τη βιβλιοθήκη sklearn.neighbors. Δημιουργώντας ένα αντικείμενο της κλάσης, ακολουθούμε την ίδια διαδικασία όπως και στους προηγούμενους αλγορίθμους. Τα αποτελέσματα παρουσιάζονται στον **Πίνακας 15**, όπου εμπεριέχονται κι όλα τα αποτελέσματα των προηγούμενων αλγορίθμων.

```
# Set model to an instance
knn = KNeighborsClassifier(n_neighbors=3)

# Fit the model to the training data
knn.fit(X_train, y_train)

y_train_pred = knn.predict(X_train)
y_pred = knn.predict(X_test)
```

Πίνακας 15 *Αποτελέσματα αλγορίθμου Κ-Κοντινότεροι Γείτονες (K-Nearest Neighbors).*

Model	Accuracy	F1-Score	Recall	Precision
K-Nearest Neighbors (KNN)	0.822569	0.663072	0.657754	0.668478

4.4.6. Χρήση Διασταυρωμένης Επικύρωσης και Επιλογή Υπερπαραμέτρων.

Προηγουμένως, το μοντέλο μας αξιολογήθηκε χωρίς την εφαρμογή της διασταυρωμένης επικύρωσης, όπως αυτή αναλύεται στην ενότητα 4.5. Για την μελέτη αυτή, αξιοποιήσαμε την τεχνική GridSearchCV. Με την τεχνική αυτή, δημιουργούμε ένα πλέγμα από διάφορες τιμές που μπορούν να λάβουν οι υπερπαράμετροι, κι αυτή αναζητά τον καλύτερο συνδυασμό. Κάθε συνδυασμός αξιολογείται μέσω των επαναλήψεων της διασταυρωμένης επικύρωσης.
Η διαδικασία που εκτελείται είναι η ίδια για κάθε αλγόριθμο. Ο παρακάτω κώδικας χρησιμοποιεί την διαδικασία μαζί με τον αλγόριθμο Logistic Regression.

Δημιουργούμε το πλέγμα με τις διάφορες υπερπαραμέτρους και τις διάφορες τιμές που μπορούν να λάβουν.

```
log_param_grid = {
    'C': [0.001, 0.01, 0.1, 1.0],
    'class_weight': [None, 'balanced'],
    'dual': [False],
    'fit_intercept': [True, False],
    'intercept_scaling': [1],
    'max_iter': [100, 200, 300],
    'multi_class': ['auto', 'ovr', 'multinomial'],
    'penalty': ['l1', 'l2'],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag',
'saga'],
    'tol': [0.0001, 0.001, 0.01],
    'warm_start': [False, True]
}
```

Δημιουργούμε ένα αντικείμενο της κλάσης *GridSearchCV*. Έπειτα δηλώνουμε το μοντέλο *(log_regression)* το πλέγμα *(param_grid)* που θα ληφθεί υπόψη για την αναζήτηση του καλύτερου συνδυασμού, και τον αριθμό των υποσυνόλων όπου θα χωριστούν τα δεδομένα

εκπαίδευσης. Στην δική μας περίπτωση έχουμε ήδη ορίσει την μέθοδο διασταυρωμένης επικύρωσης οπότε θέτουμε ως *cv=cv*. Οι παράμετροι *verbose* και *n_jobs* δεν επηρεάζουν την απόδοση του μοντέλου. Με την *verbose* ίση με 0 δηλώνουμε πως δεν θέλουμε να εκτυπωθούν πληροφορίες κατά την εκτέλεση της αναζήτησης. Με *n_jobs* ίση με -1 σημαίνει πως θα χρησιμοποιηθούν όλοι οι διαθέσιμοι πυρήνες του επεξεργαστή. Αυτό μπορεί να επιταχύνει σημαντικά την αναζήτηση, ειδικά όταν υπάρχει ένα μεγάλο πλέγμα υπερπαραμέτρων ή όταν τα δεδομένα είναι πολύ μεγάλα.

```
log_grid_search = GridSearchCV(log_regression, param_grid=log_param_grid,
cv=cv, n_jobs=-1, verbose=0)
```

Έπειτα εφαρμόζουμε τη τεχνική στα δεδομένα εκπαίδευσης κι εκτυπώνουμε τον καλύτερο συνδυασμό υπερπαραμέτρων που προκύπτει από τη διαδικασία.

```
log_grid_search.fit(X_train, y_train)

print('\nBest parameters:')
print(log_grid_search.best_params_)
```

Παρακάτω εμφανίζονται όλες οι τιμές των υπερπαραμέτρων που επέλεξε το πλέγμα για κάθε αλγόριθμο

Logistic Regression

```
{'C': 0.1, 'class_weight': None, 'dual': False, 'fit_intercept':
True, 'intercept_scaling': 1, 'max_iter': 200, 'multi_class':
'auto', 'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.001,
'warm_start': True}
```

XGBoost

```
{'colsample_bytree': 1.0, 'learning_rate': 0.1, 'max_depth': 3,
'min_child_weight': 5, 'n_estimators': 200, 'subsample': 1.0}
```

Τυχαίο Δάσος (Random Forest)

```
{'bootstrap': True, 'class_weight': None, 'max_depth': 10,
'max_features': 'sqrt', 'min_impurity_decrease': 0.0,
'min_samples_split': 10, 'n_estimators': 100}
```

Δέντρα Αποφάσεων (Decision Trees)

```
{'criterion': 'entropy', 'max_depth': 10, 'max_features':
'sqrt', 'min_impurity_decrease': 0.0, 'min_samples_leaf': 4,
'min_samples_split': 10, 'splitter': 'best'}
```

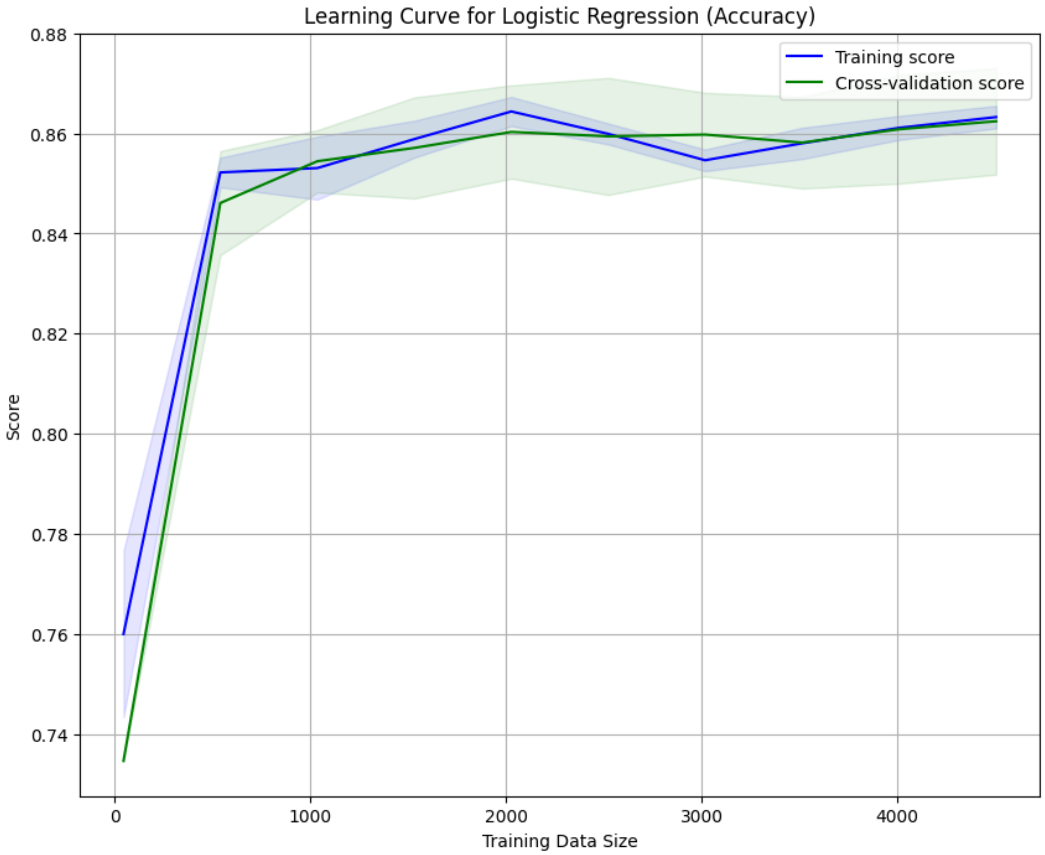
Κ-Κοντινότεροι Γείτονες (K-Nearest Neighbors)

```
{'algorithm': 'auto', 'leaf_size': 10, 'metric': 'minkowski',
'n_neighbors': 5, 'p': 3, 'weights': 'uniform'}
```

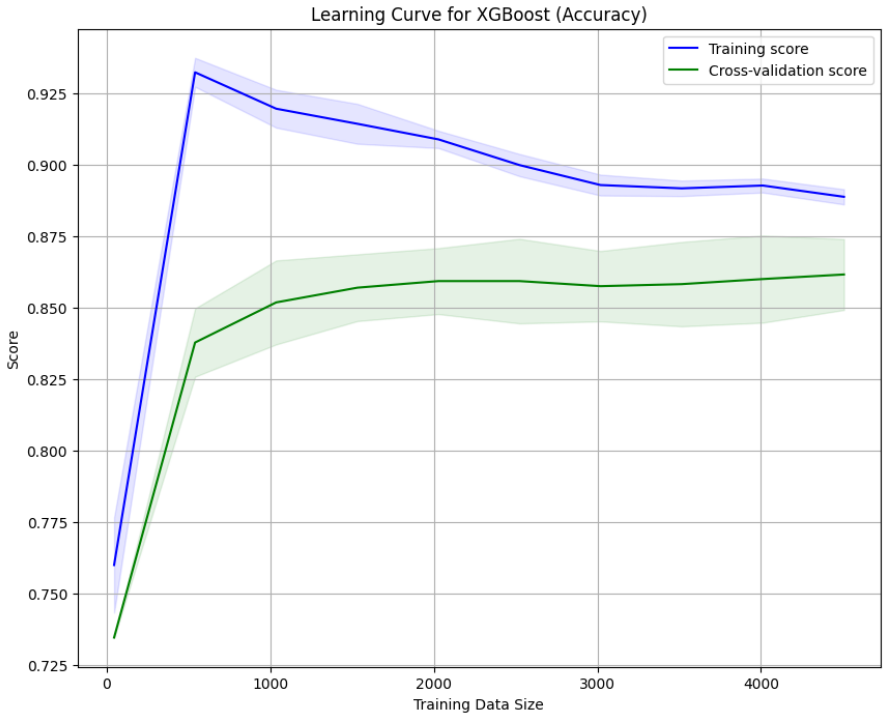
Τέλος, κάνουμε την πρόβλεψη στα δεδομένα δοκιμής και εξάγουμε τα αποτελέσματα. Στον **Πίνακας 16**, περιλαμβάνονται τα αποτελέσματα για όλους τους αλγορίθμους με τη χρήση της τεχνικής GridSearch.

Πίνακας 16 *Αποτελέσματα αλγορίθμων με χρήση διασταυρωμένης επικύρωσης και υπερπαραμετροποίησης.*

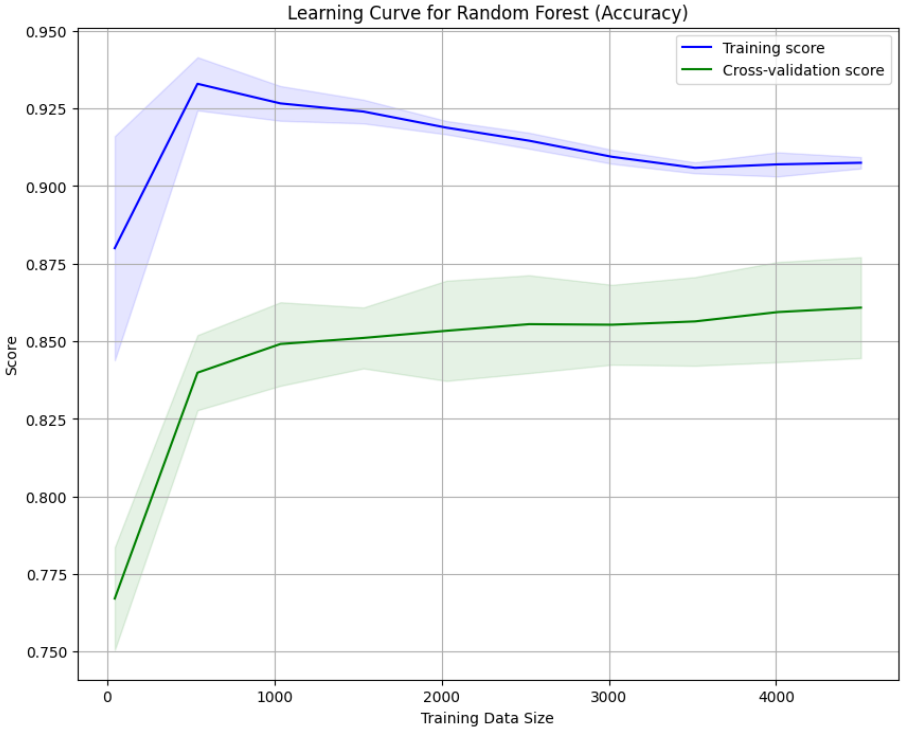
Model	Accuracy	F1-Score	Recall	Precision
Logistic Regression	0.855216	0.713483	0.679144	0.751479
XGBoost	0.862314	0.730556	0.703209	0.760116
Random Forest	0.851668	0.697540	0.644385	0.760252
Decision Trees	0.841022	0.702128	0.705882	0.698413
K-Nearest Neighbors (KNN)	0.823989	0.673684	0.684492	0.663212



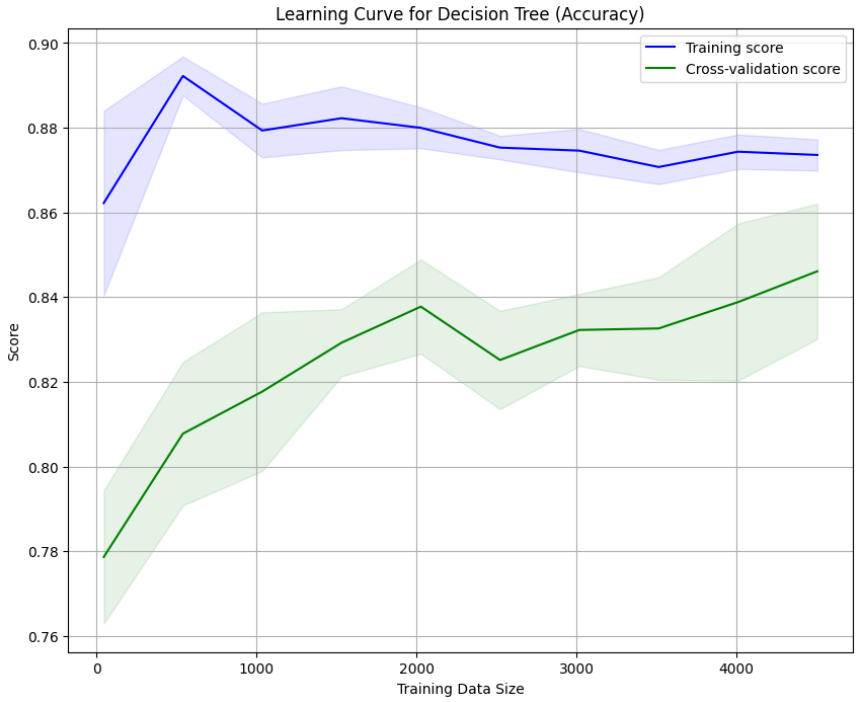
Σχήμα 27 *Καμπύλη μάθησης για τον αλγόριθμο Λογιστικής Παλινδρόμησης (Logistc Regression).*



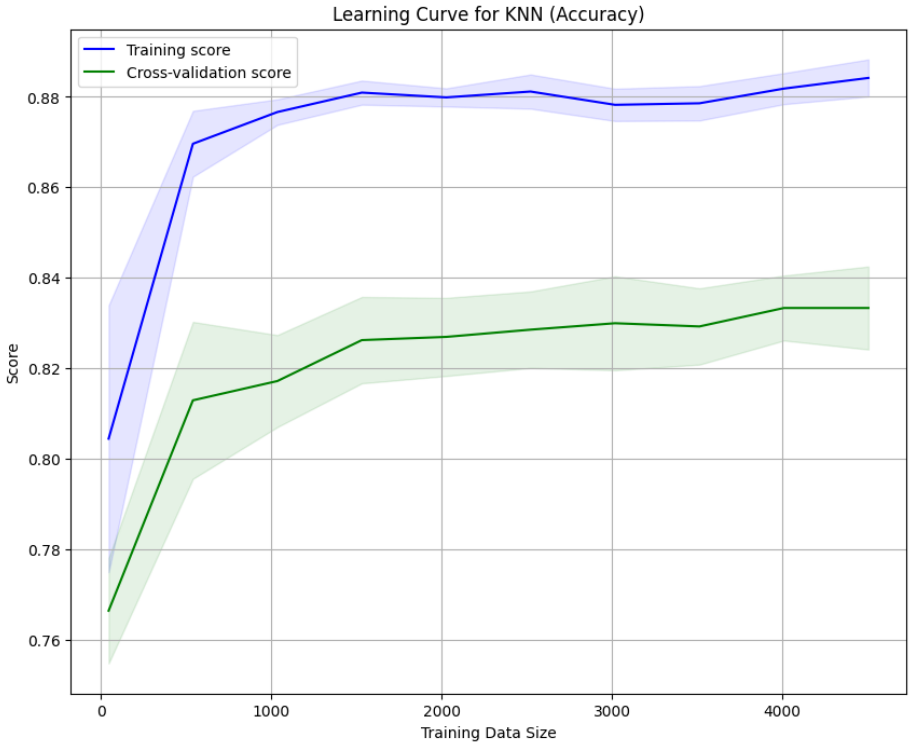
Σχήμα 28 *Καμπύλη μάθησης για τον αλγόριθμο XGboost.*



Σχήμα 29 *Καμπύλη μάθησης για τον αλγόριθμο Τυχαίου Δάσους (Random Forest).*



Σχήμα 30 *Καμπύλη μάθησης για τον αλγόριθμο Δέντρου Απόφασης (Decision Tree).*



Σχήμα 31 *Καμπύλη μάθησης για τον αλγόριθμο Κ-Κοντινότεροι Γείτονες (K-Nearest Neighbors).*

4.5. ΑΞΙΟΛΟΓΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Στο σημείο της μελέτης αυτής, γίνεται η αξιολόγηση των αποτελεσμάτων, όπως παρουσιάζονται στον πίνακα 5.11. Η ανάλυση και σύγκριση των αποτελεσμάτων από διάφορους αλγορίθμους μηχανικής μάθησης αποτελεί ένα κρίσιμο στάδιο στην επιστημονική έρευνα και ανάπτυξη. Μέσω της λεπτομερούς εξέτασης των μετρικών απόδοσης, όπως η Ακρίβεια, το F1-Score, το Recall, και η Precision, είναι δυνατή η αντικειμενική αξιολόγηση και σύγκριση των μοντέλων.

Οι αλγόριθμοι *XGBoost* και *Logistic Regression* παρουσιάζουν τα μεγαλύτερο ποσοστό ορθότητας (accuracy) συγκριτικά με τους υπόλοιπους αλγορίθμους, υποδηλώνοντας έτσι την δυνατότητα να προβλέπουν με μεγαλύτερη ακρίβεια. Ότι αφορά την μετρική F1-Score όπου περιγράφεται αναλυτικότερα στο Κεφάλαιο 3.4, πάλι οι αλγόριθμοι *XGBoost* και *Logistic Regression* παρουσιάζουν τα μεγαλύτερα ποσοστά, εξηγώντας την συνδυαστική δυνατότητα να ανιχνεύουν αποτελεσματικά τα θετικά δείγματα και παράλληλα την δυνατότητα αποφυγής ψευδών θετικών. Τα υψηλότερα αποτελέσματα της μετρικής της ανάκλησης (recall), φέρουν μαζί τους οι αλγόριθμοι *XGBoost* και *Decision Tree*, όπου αποδεικνύουν την ικανότητά τους να εντοπίζουν τα θετικά δείγματα πιο αποτελεσματικά. Στην μετρική της ακρίβειας (precision), οι αλγόριθμοι *XGBoost* και *Random Forest* κατέχουν τα υψηλότερα ποσοστά. Πιο συγκεκριμένα, οι δύο αλγόριθμοι αυτοί μπορούν να προβλέπουν με μεγαλύτερη ακρίβεια θετικά δείγματα από το σύνολο της θετικής κλάσης, όπως αυτό περιγράφεται αναλυτικότερα και στο Κεφάλαιο 3.2.

Η καμπύλη μάθησης για την Logistic Regression δείχνει ότι η καμπύλη της εκπαίδευσης ξεκινάει ψηλά και μειώνεται ελαφρώς καθώς εισάγονται περισσότερα δεδομένα, κάτι που είναι τυπικό καθώς το μοντέλο αρχικά προσαρμόζεται σε ένα μικρό σύνολο δεδομένων και στη συνέχεια γενικεύεται καθώς παρέχονται περισσότερα δεδομένα. Η καμπύλη της διασταυρωμένης επικύρωσης αυξάνεται με περισσότερα δεδομένα, υποδεικνύοντας ότι το μοντέλο μαθαίνει και γενικεύεται καλά.

Αντιθέτως, η καμπύλη μάθησης του XGBoost δείχνει ένα διαφορετικό μοτίβο. Το αποτέλεσμα της εκπαίδευσης είναι εξαιρετικά υψηλό στην αρχή, κάτι που υποδηλώνει υπερβολική προσαρμογή όταν τα δεδομένα είναι ελάχιστα. Ωστόσο, καθώς τροφοδοτούνται περισσότερα

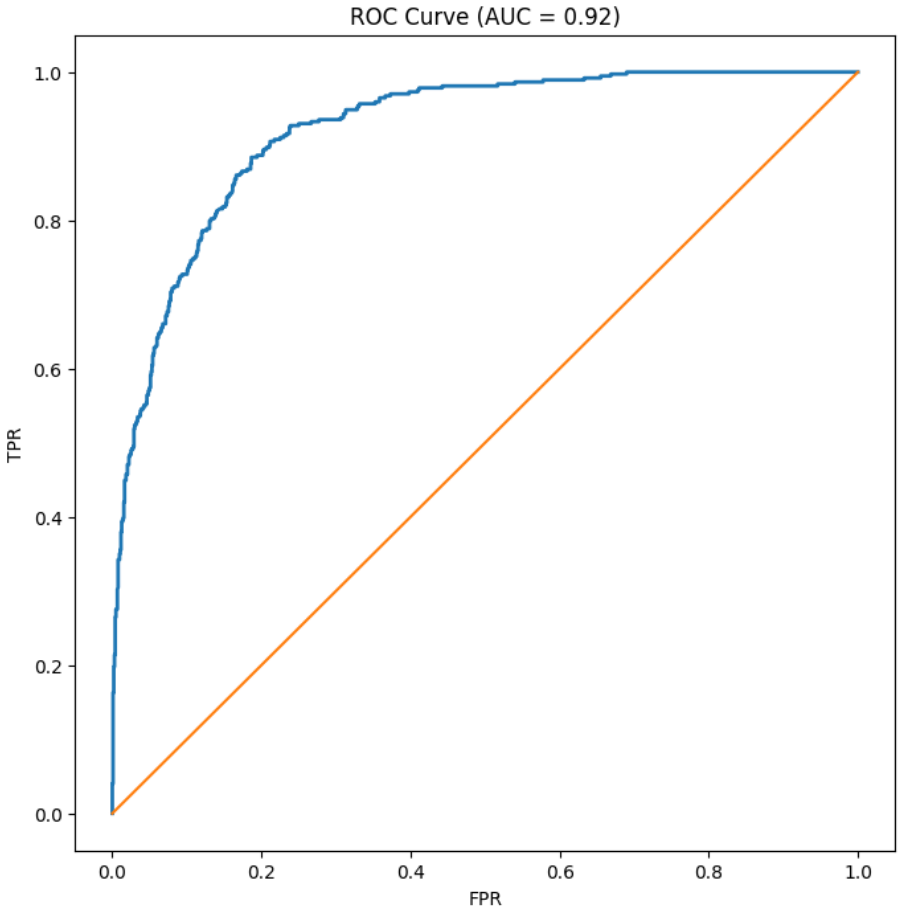
δεδομένα, η η καμπύλη εκπαίδευσης μειώνεται ενώ η καμπύλη διασταυρωμένης επικύρωσης αυξάνεται, κάτι που είναι χαρακτηριστικό ενός μοντέλου που μαθαίνει από περισσότερα δεδομένα. Επιτυγχάνει επίσης την υψηλότερη βαθμολογία διασταυρωμένης επικύρωσης μεταξύ όλων των μοντέλων, υποδεικνύοντας μια ισχυρή απόδοση σε αόρατα δεδομένα.

Η καμπύλη μάθησης Random Forest δείχνει υψηλό σκορ εκπαίδευσης που δεν μειώνεται πολύ με περισσότερα δεδομένα, υποδεικνύοντας ότι το μοντέλο είναι αρκετά σίγουρο για τις προβλέψεις του στο σετ εκπαίδευσης. Το σκορ της διασταυρωμένης επικύρωσης αυξάνεται με περισσότερα δεδομένα αλλά πλησιάζει στο 0,85. Οι καμπύλες δεν συγκλίνουν τόσο στενά όσο στην Logistic Regression, υποδηλώνοντας ότι το μοντέλο μπορεί να επωφεληθεί από περαιτέρω υπεραπαραμετροποίηση ή περισσότερα δεδομένα για τη μείωση της διακύμανσης.

Η καμπύλη μάθησης για το μοντέλο Δέντρου Αποφάσεων παρουσιάζει υψηλή διακύμανση. Το σκορ εκπαίδευσης είναι υψηλό, υποδηλώνοντας καλή απόδοση στα δεδομένα εκπαίδευσης, αλλά το σκορ της διασταυρωμένης επικύρωσης είναι αρκετά χαμηλότερο και παρουσιάζει μεγαλύτερη μεταβλητότητα σε διάφορα σύνολα δεδομένων.

Η καμπύλη μάθησης του KNN δείχνει μια σταδιακή μείωση στην καμπύλη εκπαίδευσης και μια πιο αργή αύξηση στην καμπύλη διασταυρωμένης επικύρωσης καθώς εισάγονται περισσότερα δεδομένα.

Το σύνολο τον αποτελεσμάτων δείχνει πως ο αλγόριθμος XGBoost είναι αυτός που αποδίδει καλύτερα σε όλες τις μετρικές αξιολόγησης αλλά και στην καμπύλη μάθησης συγκριτικά με τους υπόλοιπους αλγορίθμους που χρησιμοποιήθηκαν. Τόσο στην ακρίβεια όσο και στις άλλες μετρικές, ο XGBoost παρουσιάζει την καλύτερη επίδοση και καθίσταται ως ο πιο αποτελεσματικός αλγόριθμος για την εκτέλεση του μοντέλου.

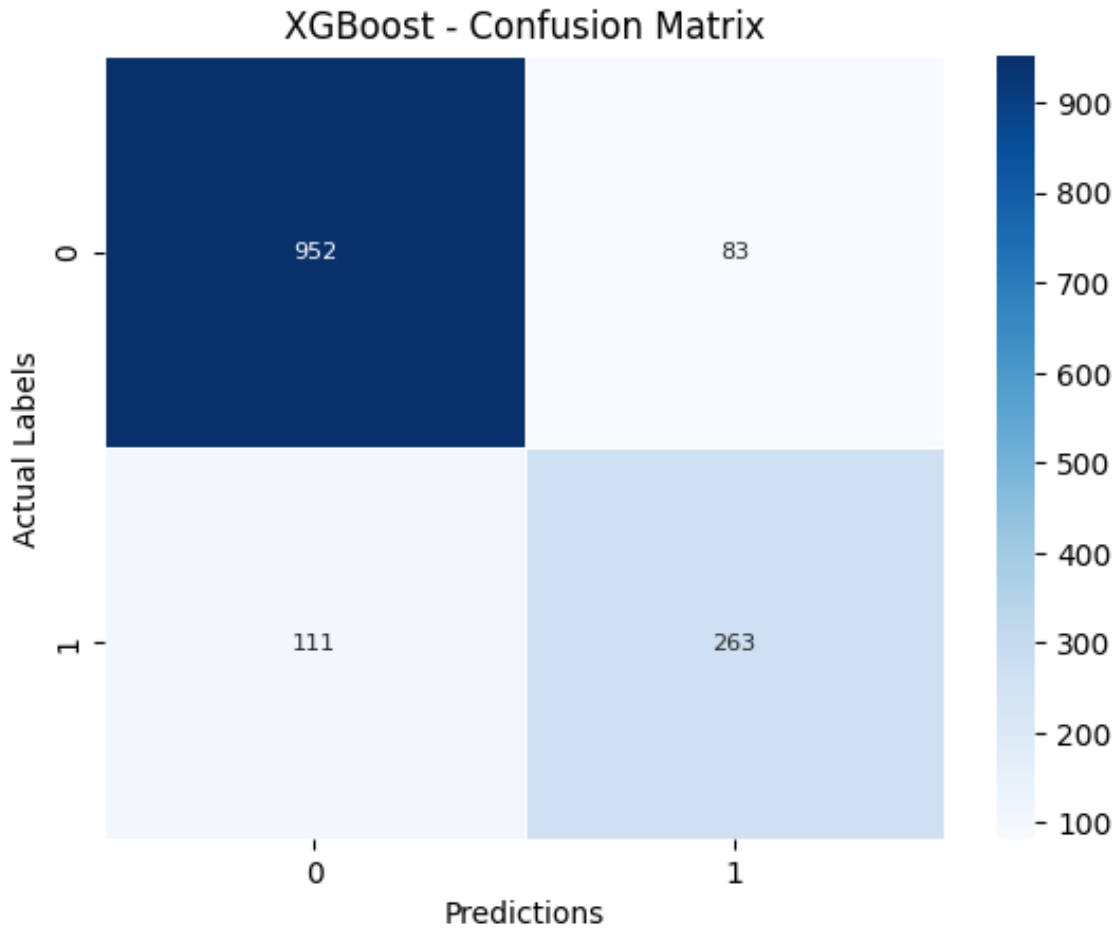


Σχήμα 32 *Καμπύλη ROC του αλγορίθμου XGBoost.*

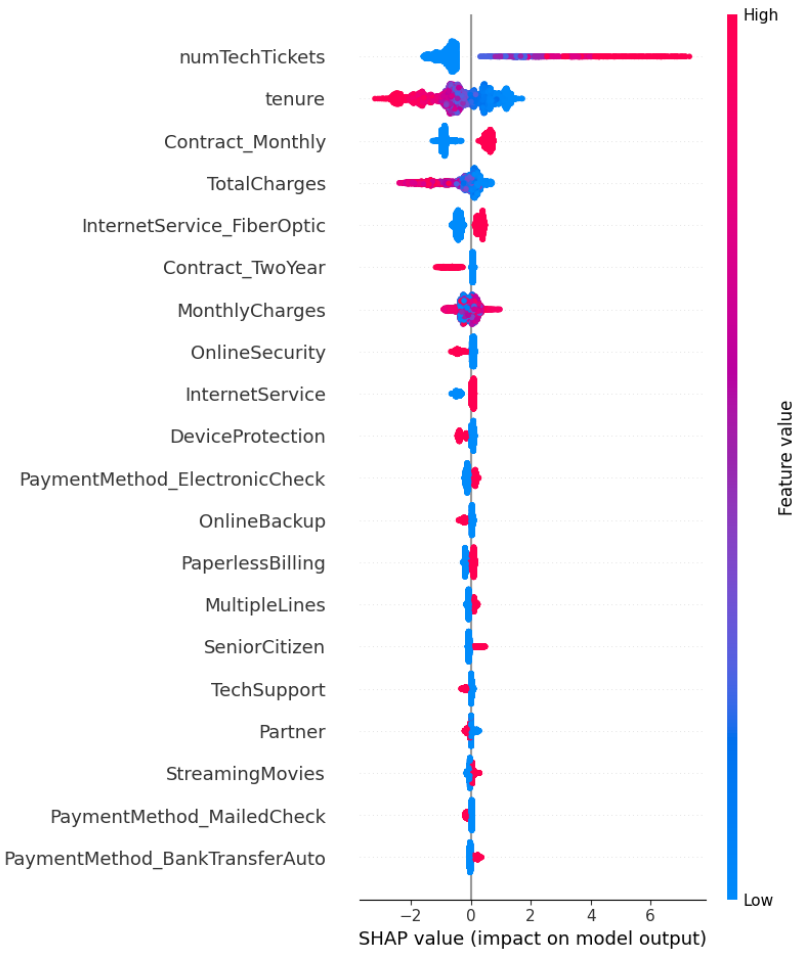
Στο παραπάνω σχήμα παρουσιάζεται η καμπύλη ROC για τον πιο αποτελεσματικό αλγόριθμο που αξιοποιήθηκε από το μοντέλο, τον *XGBoost*. Το διάγραμμα φέρει στους άξονές του τις TPR (True Positive Rate) και FPR (False Positive Rate) και με αυτόν τον τρόπο παρουσιάζεται μέσω διαγράμματος η σχέση μεταξύ του ποσοστού των αληθή θετικών και του ποσοστού των ψευδώς θετικών. Η διακριτική ικανότητα του μοντέλου με τη χρήση του *XGBoost* παρουσιάζει σημαντική απόδοση (AUC=0.92) και μπορεί να διακρίνει και να ταξινομήσει εξαιρετικά τα αληθή και ψευδώς θετικά του προβλεπτικού συνόλου δεδομένων.

Ο πίνακας σύγχυσης του ***Σχήμα 33***, δείχνει πως ο ταξινομητής *XGBoost* κατατάσσει τα σημεία στις ακόλουθες κατηγορίες των αληθή θετικών (true positive), ψευδών θετικών (false positive), αληθή αρνητικών (true negative) και ψευδών αρνητικών (false negative). Συνολικά το μοντέλο,

έκανε 1215 σωστές προβλέψεις και 194 λανθασμένες, διαμορφώνοντας την προβλεπτική ακρίβεια του στο 86,82%.



Σχήμα 33 *Πίνακας σύγχυσης του μοντέλου με τη χρήση του XGBoost.*



Σχήμα 34 *Διάγραμμα SHAP (SHapley Additive exPlanations).*

Το διάγραμμα SHAP (Shapley Additive exPlanations) αποτυπώνει την επίδραση του κάθε χαρακτηριστικού στο αποτέλεσμα του μοντέλου. Τα χαρακτηριστικά που βρίσκονται στο επάνω μέρος του μοντέλου είναι κι αυτά που έχουν την μεγαλύτερη επίδραση, ενώ το χρώμα (μπλε ή κόκκινο) υποδηλώνει την κατεύθυνση της επίδρασης ενός χαρακτηριστικού (θετική ή αρνητική). Πιο συγκεκριμένα, στο ***Σχήμα 33***, φαίνεται πως τα χαρακτηριστικά με τη μεγαλύτερη επίδραση στην απόδοση του μοντέλου είναι ο αριθμός των εισητηρίων τεχνικής υποστήριξης (numTechTickets), διάρκεια παραμονής του πελάτη (tenure), πελάτες με μηνιαίο συμβόλαιο (Contract_Monthly) και οι συνολικές χρεώσεις. Πιο συγκεκριμένα, οι πελάτες με μεγάλο αριθμό εισιτηρίων τεχνικής υποστήριξης έχουν θετική επίδραση στην απόδοση του μοντέλου. Το αντίθετο συμβαίνει με την διάρκεια της σχέσης του πελάτη με την εταιρεία. Φαίνεται πως πελάτες με μακροχρόνια σχέση με την εταιρεία έχουν αρνητική επίδραση στην απόδοση ενώ

νέοι πελάτες έχουν θετική επίδραση. Επίσης, οι πελάτες με υψηλές μηνιαίες χρεώσεις επιδρούν αρνητικά στην απόδοση του μοντέλου ενώ ταυτόχρονα, πελάτες με μηνιαίο συμβόλαιο έχουν θετική επίδραση.

ΚΕΦΑΛΑΙΟ 5: ΣΥΜΠΕΡΑΣΜΑΤΑ

Η αξιολόγηση των αποτελεσμάτων έδειξε πως ο αλγόριθμος *XGBoost* είχε την καλύτερη επίδοση σε σύγκριση με τους υπόλοιπους αλγορίθμους. Ωστόσο, ο αλγόριθμος *Logistic Regression* ήταν επίσης αποτελεσματικός με επίδοση κοντά σε αυτή του αλγορίθμου *XGBoost*. Η επιλογή του *XGBoost* ως τον πιο αποτελεσματικό αλγόριθμο για το μοντέλο της μελέτης αυτής έγινε από την συνολική αξιολόγηση των αποτελεσμάτων που έλαβε το μοντέλο για κάθε μετρική αξιολόγησης του μοντέλου.

Ωστόσο, είναι φανερό πως με την κατάλληλη παραμετροποίηση των υπερπαραμέτρων μπορούμε να επιτευχθούν καλύτερα αποτελέσματα και να γίνει το μοντέλο πιο αποδοτικό και με μεγαλύτερη προβλεπτική ισχύ. Πριν την χρήση της μεθόδου διασταυρωμένης επικύρωσης, το μοντέλο έλαβε χαμηλότερα αποτελέσματα για κάθε αλγόριθμο που αξιοποίησε. Με την χρήση του πλέγματος αναζήτησης, το μοντέλο απέδωσε αποτελεσματικότερα για κάθε αλγόριθμο που χρησιμοποιήθηκε, έχοντας αναζητήσει τον καλύτερο δυνατό συνδυασμό τιμών για τις υπερπαραμέτρους.

Καθ’ όλη την διάρκεια εφαρμογής των αλγορίθμων, δεν επιτεύχθηκε κάποια περίπλοκη παραμετροποίηση των υπερπαραμέτρων. Η επιλογή αυτή κατέστησε το μοντέλο απλό και κατανοητό και για το σύστημα λειτουργίας, αλλά και για τον άνθρωπο. Με αυτό τον τρόπο επιτεύχθηκαν αποτελέσματα αρκετά υψηλά αποφεύγοντας ταυτόχρονα και την πιθανότητα εμφάνισης υπερπροσαρμογής (overfitting).

Το μοντέλο μπορεί να βελτιωθεί και να αποκτήσει καλύτερα αποτελέσματα. Υπάρχουν δυνατότητες βελτίωσης τις απόδοσης κι αυτό μπορεί να συμβεί με διάφορους τρόπους. Ένας από αυτούς είναι η εφαρμογή τεχνικών υψηλής παραμετροποίησης των υπερπαραμέτρων, που με τις κατάλληλες τιμές, μπορούν να επιτευχθούν αυξημένα αποτελέσματα και ταυτόχρονη αποφυγή της γενίκευσης. Ένας άλλος τρόπος για την βελτίωση της απόδοσης είναι η αξιοποίηση των τεχνικών βαθιάς μάθησης και των νευρωνικών δικτύων. Τα νευρωνικά δίκτυα μπορούν να βοηθήσουν σημαντικά στην επίλυση ενός προβλήματος ταξινόμησης όπως της μελέτης αυτής και να διαμορφώσουν ένα μοντέλο μάθησης πιο βελτιωμένο και αποδοτικό.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Agrawal, T. (2021). Introduction to Hyperparameters. In: *Hyperparameter Optimization in Machine Learning*. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6579-6_1

2. Paper, D. (2020). Introduction to Scikit-Learn. In: *Hands-on Scikit-Learn for Machine Learning Applications.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5373-1_1

3. El Morr, C., Jammal, M., Ali-Hassan, H., El-Hallak, W. (2022). Introduction to Machine Learning. In: *Machine Learning for Practical Decision Making. International Series in Operations Research & Management Science, vol 334.* Springer, Cham. https://doi.org/10.1007/978-3-031-16990-8_1

4. Sarkar, D., Bali, R., Sharma, T. (2018). Machine Learning Basics. In: *Practical Machine Learning with Python.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-3207-1_1

5. Singh, P. (2022). Introduction to Machine Learning. In: *Machine Learning with PySpark.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-7777-5_3

6. Panesar, A. (2019). What Is Machine Learning?. In: *Machine Learning and AI for Healthcare.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-3799-1_3

7. El Morr, C., Jammal, M., Ali-Hassan, H., El-Hallak, W. (2022). Overview of Machine Learning Algorithms. In: *Machine Learning for Practical Decision Making. International Series in Operations Research & Management Science, vol 334.* Springer, Cham. https://doi.org/10.1007/978-3-031-16990-8_3

8. Huawei Technologies Co., Ltd.. (2023). Machine Learning. In: *Artificial Intelligence Technology.* Springer, Singapore. https://doi.org/10.1007/978-981-19-2879-6_2

9. Jo, T. (2021). Introduction. In: *Machine Learning Foundations.* Springer, Cham. https://doi.org/10.1007/978-3-030-65900-4_1

10. Moolayil, J. (2019). An Introduction to Deep Learning and Keras. In: *Learn Keras for Deep Neural Networks.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4240-7_1

11. Goyal, P., Pandey, S., Jain, K. (2018). Introduction to Natural Language Processing and Deep Learning. In: *Deep Learning for Natural Language Processing.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-3685-7_1

12. Ghatak, A. (2017). Classification. In: *Machine Learning with R.* Springer, Singapore. https://doi.org/10.1007/978-981-10-6808-9_5

13. McCarthy, R.V., McCarthy, M.M., Ceccucci, W., Halawi, L. (2019). Predictive Models Using Regression. In: *Applying Predictive Analytics.* Springer, Cham. https://doi.org/10.1007/978-3-030-14038-0_4

14. Schonlau, M. (2023). Logistic Regression. In: *Applied Statistical Learning. Statistics and Computing.* Springer, Cham. https://doi.org/10.1007/978-3-031-33390-3_4

15. Hull, I. (2021). Regression. In: *Machine Learning for Economics and Finance in TensorFlow 2.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6373-0_3

16. Hull, I. (2021). Trees. In: *Machine Learning for Economics and Finance in TensorFlow 2.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6373-0_4

17. Quinto, B. (2020). Introduction to Machine Learning. In: *Next-Generation Machine Learning with Spark.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5669-5_1

18. Ramasubramanian, K., Singh, A. (2019). Machine Learning Theory and Practice. In: *Machine Learning Using R.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4215-5_6

19. Rosett, C.M., Hagerty, A. (2021). Common Machine Learning Techniques. In: *Introducing HR Analytics with Machine Learning.* Springer, Cham. https://doi.org/10.1007/978-3-030-67626-1_9

20. Panesar, A. (2019). Machine Learning Algorithms. In: *Machine Learning and AI for Healthcare.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-3799-1_4

21. Soujanya, K.V., Tembhurne, O. (2023). Analysis of Machine Learning Algorithms for Detection of Cyberbullying on Social Networks. In: Misra, R., Omer, R., Rajarajan, M., Veeravalli, B., Kesswani, N., Mishra, P. (eds) *Machine Learning and Big Data Analytics. ICMLBDA 2022.* Springer Proceedings in Mathematics & Statistics, vol 401. Springer, Cham. https://doi.org/10.1007/978-Mα3-031-15175-0_14

22. Sarkar, D., Bali, R., Sharma, T. (2018). Building, Tuning, and Deploying Models. In: *Practical Machine Learning with Python.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-3207-1_5

23. Gupta, P., Sehgal, N.K. (2021). Machine Learning Algorithms. In: *Introduction to Machine Learning in the Cloud with Python.* Springer, Cham. https://doi.org/10.1007/978-3-030-71270-9_2

24. Panesar, A. (2019). Evaluating Learning for Intelligence. In: *Machine Learning and AI for Healthcare.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-3799-1_5

25. Gupta, A., Thustu, S.R., Thakor, R.R., Patil, S.A., Joshi, R., Laban, R.M. (2023). Prediction of Maneuvering Status for Aerial Vehicles Using Supervised Learning Methods. In: Misra, R., Omer, R., Rajarajan, M., Veeravalli, B., Kesswani, N., Mishra, P. (eds) *Machine Learning*

and Big Data Analytics. ICMLBDA 2022. Springer Proceedings in Mathematics & Statistics, vol 401. Springer, Cham. https://doi.org/10.1007/978-3-031-15175-0_22

26. El Morr, C., Jammal, M., Ali-Hassan, H., El-Hallak, W. (2022). Data Preprocessing. In: *Machine Learning for Practical Decision Making.* International Series in Operations Research & Management Science, vol 334. Springer, Cham. https://doi.org/10.1007/978-3-031-16990-8_4

27. Singh, P. (2019). Data Processing. In: *Machine Learning with PySpark.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4131-8_3

28. Panesar, A. (2021). Preparing Data. In: *Machine Learning and AI for Healthcare.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6537-6_6

29. Bermbach, D., Wittern, E., Tai, S. (2017). Data Preprocessing. In: *Cloud Service Benchmarking.* Springer, Cham. https://doi.org/10.1007/978-3-319-55483-9_11

30. Batista, N.A., Brandão, M.A., Pinheiro, M.B., Dalip, D.H., Moro, M.M. (2020). Data from Multiple Web Sources: Crawling, Integrating, Preprocessing, and Designing Applications. In: Roesler, V., Barrére, E., Willrich, R. (eds) *Special Topics in Multimedia, IoT and Web Technologies.* Springer, Cham. https://doi.org/10.1007/978-3-030-35102-1_8

31. Aggarwal, C.C. (2017). An Introduction to Outlier Analysis. In: *Outlier Analysis.* Springer, Cham. https://doi.org/10.1007/978-3-319-47578-3_1

32. Ramasubramanian, K., Singh, A. (2019). Introduction to Machine Learning and R. In: *Machine Learning Using R.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4215-5_1

33. Silaparasetty, N. (2020). Machine Learning Programming with Tensorflow 2.0. In: *Machine Learning Concepts with Python and the Jupyter Notebook Environment.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5967-2_11

34. Ball, R., Rague, B. (2022). Machine Learning. In: *The Beginner's Guide to Data Science.* Springer, Cham. https://doi.org/10.1007/978-3-031-07865-1_8

35. Ippolito, P.P. (2022). Hyperparameter Tuning. In: *Egger, R. (eds) Applied Data Science in Tourism. Tourism on the Verge.* Springer, Cham. https://doi.org/10.1007/978-3-030-88389-8_12

36. Kovalerchuk, B. (2020). Enhancement of Cross Validation Using Hybrid Visual and Analytical Means with Shannon Function. In: *Kosheleva, O., Shary, S., Xiang, G., Zapatrin, R. (eds) Beyond Traditional Probabilistic Data Processing Techniques: Interval, Fuzzy etc. Methods and Their Applications. Studies in Computational Intelligence, vol 835.* Springer, Cham. https://doi.org/10.1007/978-3-030-31041-7_29

37. Fontanari, T., Fróes, T.C., Recamonde-Mendoza, M. (2022). Cross-validation Strategies for Balanced and Imbalanced Datasets. In: *Xavier-Junior, J.C., Rios, R.A. (eds) Intelligent Systems. BRACIS 2022. Lecture Notes in Computer Science(), vol 13653.* Springer, Cham. https://doi.org/10.1007/978-3-031-21686-2_43

38. Silaparasetty, N. (2020). Introduction to Jupyter Notebook. In: *Machine Learning Concepts with Python and the Jupyter Notebook Environment.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5967-2_6

39. Schäfer, C. (2021). Extensions for Scientists: NumPy, SciPy, Matplotlib, Pandas. In: *Quickstart Python. essentials().* Springer, Wiesbaden. https://doi.org/10.1007/978-3-658-33552-6_9

40. Bisong, E. (2019). NumPy. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4470-8_10

41. Bisong, E. (2019). Pandas. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4470-8_11

42. Hunt, J. (2023). Introduction to Matplotlib. In: *Advanced Guide to Python 3 Programming. Undergraduate Topics in Computer Science.* Springer, Cham. https://doi.org/10.1007/978-3-031-40336-1_14

43. Pajankar, A. (2022). NumPy Routines and Getting Started with Matplotlib. In: *Hands-on Matplotlib.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-7410-1_3

44. Rajagopalan, G. (2021). Data Visualization with Python Libraries. In: *A Python Data Analyst's Toolkit.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6399-0_7

45. Pajankar, A. (2022). Introduction to Data Visualization with Seaborn. In: *Hands-on Matplotlib.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-7410-1_17

46. Porcu, V. (2018). Scikit-learn. In: *Python for Data Mining Quick Syntax Reference.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4113-4_11

47. Kramer, O. (2016). Scikit-Learn. In: *Machine Learning for Evolution Strategies. Studies in Big Data, vol 20.* Springer, Cham. https://doi.org/10.1007/978-3-319-33383-0_5

48. Amin, A., Shehzad, S., Khan, C., Ali, I., & Anwar, S. (2015). *Churn Prediction in Telecommunication Industry Using Rough Set Approach.* , 83-95. https://doi.org/10.1007/978-3-319-10774-5_8.

49. De, S., & Prabu, P. (2022). *Predicting customer churn: A systematic literature review.* Journal of Discrete Mathematical Sciences and Cryptography, 25, 1965 - 1985. https://doi.org/10.1080/09720529.2022.2133238.

50. Huang, B., Kechadi, M., & Buckley, B. (2012). *Customer churn prediction in telecommunications.* Expert Syst. Appl., 39, 1414-1425. https://doi.org/10.1016/j.eswa.2011.08.024.

51. Jadhav, R., & Pawar, U. (2011). *Churn Prediction in Telecommunication Using Data Mining Technology.* International Journal of Advanced Computer Science and Applications, 2. https://doi.org/10.14569/IJACSA.2011.020204.

52. Joolfoo, M., Jugumauth, R., & Joolfoo, K. (2020). *A Systematic Review of Algorithms applied for Telecom Churn Prediction.* 2020 3rd International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM), 136-140. https://doi.org/10.1109/ELECOM49001.2020.9296999.

53. Keramati, A., Jafari-Marandi, R., Aliannejadi, M., Ahmadian, I., Mozaffari, M., & Abbasi, U. (2014). *Improved churn prediction in telecommunication industry using data mining techniques.* Appl. Soft Comput., 24, 994-1012. https://doi.org/10.1016/j.asoc.2014.08.041.

54. Kim, K., Jun, C., & Lee, J. (2014*). Improved churn prediction in telecommunication industry by analyzing a large network.* Expert Syst. Appl., 41, 6575-6584. https://doi.org/10.1016/j.eswa.2014.05.014.

55. Lin, W., Tsai, C., & Ke, S. (2014). *Dimensionality and data reduction in telecom churn prediction.* Kybernetes, 43, 737-749. https://doi.org/10.1108/K-03-2013-0045.

56. Patil, A., & Wankhade, N. (2023). *Telecom Churn Prediction Using Machine Learning.* International Journal of Advanced Research in Science, Communication and Technology. https://doi.org/10.48175/ijarsct-13886.

57. Xu, S., Lai, S., & Qiu, M. (2009). *Privacy preserving churn prediction.* , 1610-1614. https://doi.org/10.1145/1529282.1529643.

58. Bartz-Beielstein, T., Zaefferer, M. (2023). Models. In: Bartz, E., Bartz-Beielstein, T., Zaefferer, M., Mersmann, O. (eds) *Hyperparameter Tuning for Machine and Deep Learning with R.* Springer, Singapore. https://doi.org/10.1007/978-981-19-5170-1_3

59. Lin, S., Zhang, K., Geng, R., Ma, L. (2022). Using an Ensembled Boosted Model for IoT Time Series Regression. In: Wang, S., Zhang, Z., Xu, Y. (eds) *IoT and Big Data Technologies for Health Care. IoTCare 2021.* Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 415. Springer, Cham. https://doi.org/10.1007/978-3-030-94182-6_28

60. Kounev, S., Lange, KD., Kistowski, J.v. (2020). Experimental Design. In: Systems Benchmarking. Springer, Cham. https://doi.org/10.1007/978-3-030-41705-5_5

61. Ahmed Arafa, A., Radad, M., Badawy, M., & El-Fishawy, N. (2022). Logistic Regression Hyperparameter Optimization for Cancer Classification. Menoufia Journal of Electronic Engineering Research, 31(1), 1-8. doi: 10.21608/mjeer.2021.70512.1034

62. scikit-learn developers. (n.d.). LogisticRegression. scikit-learn. Retrieved March 5, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ 1

```
#Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


#Setting pandas options
pd.set_option('display.float_format', '{:.2f}'.format)


#Loading the dataset
data =
pd.read_excel('C:/Users/smosc/Desktop/Thesis/data/Telecom_Churn_
Rate_Dataset.xlsx')


#Information of the dataset
data.info()


#Showing the first 5 rows of the dataset.
data.head()


#Finding if there are any missing/null values.
data.isnull().sum()


#Searching for duplicates.
```

```
data.duplicated().sum()


#Finding if a customer appeared more than once in the dataset.
data['customerID'].unique().shape

#Every customer appeared in the dataset is unique.


#Changing the "TotalCharges" type to numeric.
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'])


#Exploring the error in the row 488.
data.iloc[488]

#Searching if there are any other missing cells.
data[data['TotalCharges'] == ' ']

#Probably they could be new customers. They are also have 0 in
tenure column.
#We will fill the missing values with 0.
data['TotalCharges'] = data['TotalCharges'].replace(' ', 0)

#Testing if it worked.
data[data['TotalCharges'] == ' ']

#Change the type of the column to string.
data['SeniorCitizen'] = data['SeniorCitizen'].astype(str)

#Replace Senior Citizen values (1-Yes, 0-No).
```

```
data['SeniorCitizen'] =
data['SeniorCitizen'].replace({'1':'Yes', '0':'No'})

#Showing results.
data['SeniorCitizen'].value_counts()

#Creating a new column that calculates the year.
data['year'] = data['tenure'] / 12 + 1
data['year'] = data['year'].astype(int)
```

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ 2

```
#Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns



#Setting seaborn settings for the plots.
sns.set_theme(style='ticks')

palette = ['#e60049', '#0bb4ff', '#50e991', '#e6d800',
'#9b19f5', '#ffa300', '#dc0ab4', '#b3d4ff', '#00bfa0']
palette_2 = ['#007f5f', '#2b9348', '#55a630', '#80b918',
'#aacc00', '#bfd200', '#d4d700', '#dddf00', '#eeef20',
'#ffff3f']
sns.set_palette(palette)

#Loading the clean dataset
data =
pd.read_csv('C:/Users/smosc/Desktop/Thesis/data/Telecom_Churn_Ra
te_Dataset_clean.csv')

#First 5 rows of the clean dataset
data.head()

data.info()

#Male and female customers.
fig, ax = plt.subplots(figsize=(7,5))
```

```
sns.countplot(data=data, x='gender', width=0.2)
plt.xlabel('Gender', size=10)
plt.ylabel('Count', size=10)
plt.xticks(rotation=0)
ax.bar_label(ax.containers[0], size=10)
plt.title('Male & Female Customers', size=12)

plt.show()

#Male and female senior citizen customers
senior_citizens =
data.groupby('gender')['SeniorCitizen'].value_counts().to_frame(
)
senior_citizens.rename(columns = {'count':'total'},
inplace=True)
senior_citizens.reset_index(inplace=True)
senior_citizens =
senior_citizens[senior_citizens['SeniorCitizen'] == 'Yes']
senior_citizens

fig, ax = plt.subplots(figsize=(7,5))
sns.barplot(data=senior_citizens, x='gender', y='total',
width=0.2)
plt.xlabel('Gender', size=10)
plt.ylabel('Count', size=10)
plt.xticks(rotation=0)
ax.bar_label(ax.containers[0], size=10)
plt.title('Male & Female Senior Citizen Customers', size=12)

plt.show()
```

```
#Male and female churn and non-churn customers.
churn_customers =
data.groupby('gender')['Churn'].value_counts().to_frame()
churn_customers.rename(columns={'count':'total'}, inplace=True)
churn_customers.reset_index(inplace=True)

churn_customers

fig, ax = plt.subplots(figsize=(7,5))

sns.barplot(x='gender', y='total', hue='Churn',
data=churn_customers, width=0.4)
plt.xlabel('Gender', size=10)
plt.ylabel('Count', size=10)
plt.xticks(rotation=0)
ax.bar_label(ax.containers[0], size=10)
ax.bar_label(ax.containers[1], size=10)
plt.title('Churn and Non-Churn Customers by Gender', size=12)

plt.show()

#Male and female customer total charges.
gender_total_charges =
data.groupby('gender')['TotalCharges'].sum().to_frame().reset_in
dex()

gender_total_charges

fig, ax = plt.subplots(figsize=(20,5))

sns.barplot(x='TotalCharges',
y='gender',data=gender_total_charges, width=0.3)
```

```
plt.xlabel('Total Charges', size=10)
plt.ylabel('Gender', size=10)
plt.xticks(rotation=0)
ax.bar_label(ax.containers[0], size=10)
plt.title('Total Charges by Gender', size=12)

plt.show()

#Total churn and non-churn customers.
data['Churn'].value_counts()

fig, ax = plt.subplots(figsize=(5,5))

data['Churn'].value_counts().plot(kind='pie', autopct='%1.1f%%',
startangle=90, explode=[0, 0.05])
plt.ylabel('Churn', size=10)
plt.title('Churn & Non-Churn Customers', size=12)

plt.show()


#Total charges of churn and non-churn customers.
churn_total_charges =
data.groupby('Churn')['TotalCharges'].sum().to_frame().reset_ind
ex()
churn_total_charges.replace({'No':'No Churn', 'Yes':'Churn'},
inplace=True)

fig, ax = plt.subplots(figsize=(7,5))

sns.barplot(x='Churn',
y='TotalCharges',data=churn_total_charges, width=0.2)
```

```
plt.xlabel('Churn', size=10)
plt.ylabel('Total Charges', size=10)
plt.xticks(rotation=0)
ax.bar_label(ax.containers[0], size=10)
plt.title('Churn & Non-Churn Total Charges', size=12)
plt.show()

#The top 10 customers by total charges.
top_customers_total_charges = data[['customerID',
'TotalCharges', 'tenure']].sort_values(by='TotalCharges',
ascending=False).head(10)

top_customers_total_charges

fig, ax = plt.subplots(figsize=(20,5))

sns.barplot(x='TotalCharges',
y='customerID',data=top_customers_total_charges, width=0.7,
palette=palette_2)
plt.xlabel('Total Charges', size=10)
plt.ylabel('Customer', size=10)
plt.xticks(rotation=0)
ax.bar_label(ax.containers[0], size=10)
plt.title('Top 10 Customers with the Highest Total Charges',
size=12)

plt.show()

#The customers with the highest monthly charges.
top_customers_monthly_charges = data[['customerID',
'MonthlyCharges', 'tenure']].sort_values(by='MonthlyCharges',
ascending=False).head(10)
```

```
top_customers_monthly_charges

#Finding if the customers with the highest monthly charges
included to the customers with the highest total charges.
top_customers_monthly_charges[top_customers_monthly_charges['cus
tomerID'].isin(top_customers_total_charges['customerID'])]

#The customers with the longest stay.
top_customers_longest_stay = data[data['tenure'] ==
data['tenure'].max()]
top_customers_longest_stay

#Finding if the customers with the longest stay included to the
customers with the highest total charges.
top_customers_longest_stay[top_customers_longest_stay['customerI
D'].isin(top_customers_total_charges['customerID'])]

#Total charges of customers with the longest stay.
top_customers_longest_stay['TotalCharges'].sum()

#Total customers for each internet type.
customers_internet_type =
data['InternetService'].value_counts().to_frame().reset_index()
customers_internet_type.rename(columns={'InternetService':'Inter
netType', 'count':'TotalCustomers'}, inplace=True)

customers_internet_type

#Total active customers for each internet type.
active_customers =data[data['Churn'] == 'No']
```

```
active_customers_internet_type =
active_customers['InternetService'].value_counts().to_frame().re
set_index()
active_customers_internet_type.rename(columns={'InternetService'
:'InternetType', 'count':'ActiveCustomers'}, inplace=True)

active_customers_internet_type

#Merged table with total customers and active customers for each
internet type.
customers_internet_type_merged =
customers_internet_type.merge(active_customers_internet_type,
on='InternetType')

customers_internet_type_merged

fig, ax = plt.subplots(figsize=(5,7))

sns.barplot(data=customers_internet_type_merged,
x='InternetType', y='TotalCustomers', label='Total Customers',
color='#e60049', width=0.4)
sns.barplot(data=customers_internet_type_merged,
x='InternetType', y='ActiveCustomers', label='Active Customers',
color='#0bb4ff', width=0.4)
plt.xlabel('Internet Type', size=10)
plt.ylabel('Customers', size=10)
plt.bar_label(ax.containers[0], size=10)
plt.bar_label(ax.containers[1], size=10, color='white')
plt.title('Customers by Internet Type', size=12)

plt.legend()
plt.show()
```

```
#Customers by contract type.
customers_contract =
data['Contract'].value_counts().to_frame().reset_index()
customers_contract.rename(columns={'count':'TotalCustomers'},
inplace=True)

customers_contract

#Active customers by contract type.
active_customers_contract =
active_customers['Contract'].value_counts().to_frame().reset_ind
ex()
active_customers_contract.rename(columns={'count':'ActiveCustome
rs'}, inplace=True)

active_customers_contract

#Merged table of total and active customers for each contract
type.
customers_contract_merged =
customers_contract.merge(active_customers_contract,
on='Contract')

customers_contract_merged

fig, ax = plt.subplots(figsize=(5,7))

sns.barplot(data=customers_contract_merged, x='Contract',
y='TotalCustomers', label='Total Customers', color='#e60049',
width=0.4)
```

```
sns.barplot(data=customers_contract_merged, x='Contract',
y='ActiveCustomers', label='Active Customers', color='#0bb4ff',
width=0.4)
plt.xlabel('Contract Type', size=10)
plt.ylabel('Customers', size=10)
plt.bar_label(ax.containers[0], size=9)
plt.bar_label(ax.containers[1], size=9, color='white')
plt.title('Customers by Contract Type', size=12)

plt.legend()
plt.show()

#Creating a table only with internet services.
internet_services = data.iloc[:, 9:15]

#The size of each internet service.
internet_services_melted = pd.melt(internet_services)
internet_services_size =
internet_services_melted.pivot_table(index='variable',
columns='value', aggfunc='size', fill_value=0).reset_index()
internet_services_size.rename(columns={'value' : 'index',
                                       'variable' :
'InternetServices'}, inplace=True)

internet_services_size

#The size of each internet service for active customers.
active_internet_services_size = active_customers.iloc[:,9:15]
active_internet_services_melted =
pd.melt(active_internet_services_size)
```

```
active_internet_services_size =
active_internet_services_melted.pivot_table(index='variable',
columns='value', aggfunc='size', fill_value=0).reset_index()
active_internet_services_size.rename(columns={'value' : 'index',
'variable': 'InternetServices'}, inplace=True)

active_internet_services_size

fig, ax= plt.subplots(1,2, figsize=(20,5))

plt.subplot(1,2,1)
internet_services_size.plot(x='InternetServices', kind='bar',
ax=ax[0])
plt.xlabel('Internet Services')
plt.ylabel('Customers')
plt.title('Customers by Internet Services', size=12)
plt.xticks(rotation=0, size=10)
plt.yticks(size=10)
plt.legend(fontsize='small')

plt.subplot(1,2,2)
active_internet_services_size.plot(x='InternetServices',
kind='bar', ax=ax[1])
plt.xlabel('Internet Services')
plt.ylabel('Customers')
plt.title('Active Customers by Internet Services', size=12)
plt.xticks(rotation=0, size=10)
plt.yticks(size=10)
plt.legend(fontsize='small')

plt.tight_layout()
plt.show()
```

```
#Customers with internet but without any further internet
service.
customers_internet = data[data['InternetService'] != 'No']

customers_internet.iloc[:,9:15].value_counts().head(1)

#Active customers with internet but without any further internet
service.
active_customers_internet =
active_customers[active_customers['InternetService'] != 'No']

active_customers_internet.iloc[:,9:15].value_counts().head(1)

#Categorize the total internet services in a volume column.
#0 Internet Services -- No Internet Services | 1-2 Internet
Services -- Low | 3-4 Internet Services -- Mid | 5-6 Internet
Services -- High

def categorize_row(row, value):
    count = (row == value).sum()
    if 5 <= count <= 6:
        return 'High'
    elif 3 <= count <= 4:
        return 'Mid'
    elif 1 <= count <= 2:
        return 'Low'
    else:
        return 'No Internet Services'
```

```
internet_services['InternetServicesVolume'] =
internet_services.apply(lambda row: categorize_row(row, 'Yes'),
axis=1)

internet_services.head()

#Percentage of customers for each internet services volume.
data['InternetServicesVolume'] =
internet_services['InternetServicesVolume']

percentage_isv = data['InternetServicesVolume'].value_counts() /
data['InternetServicesVolume'].count() * 100
percentage_isv = pd.DataFrame(percentage_isv)
percentage_isv.reset_index(inplace=True)
percentage_isv.rename(columns={'count':'Percentage'},
inplace=True)

percentage_isv

#Percentage of customers for each internet services volume
(plot).
fig, ax = plt.subplots(figsize=(5,5))

(data['InternetServicesVolume'].value_counts() /
data['InternetServicesVolume'].count() * 100).plot(kind='pie',
autopct='%1.1f%%')
plt.ylabel('Internet Services Volume', size=10)
plt.title('Percentage of Customers by each Internet Service
Volume', size=12)

plt.show()
```

```
#Average, min and max of monthly and total charges for each
internet services volume.
isv_stats =
data.groupby('InternetServicesVolume')[['MonthlyCharges','TotalC
harges']].agg({'mean', 'median', 'min', 'max',
'std'}).round(2).reset_index()

isv_stats

#Average, min and max of monthly and total charges for each
internet services volume (plot).
fig, ax = plt.subplots(1, 2, figsize=(25,8))

plt.subplot(1, 2, 1)
isv_stats.plot(x='InternetServicesVolume', y='MonthlyCharges',
kind='bar', ax=ax[0])
plt.xlabel('Internet Services Volume')
plt.ylabel('Monthly Charges')
plt.title('Monthly Charges Descriptive Statistcs for Internet
Services Volume', size=12)
plt.xticks(rotation=0, size=10)
plt.bar_label(ax[0].containers[0], size=10, rotation=90)
plt.bar_label(ax[0].containers[1], size=10, rotation=90)
plt.bar_label(ax[0].containers[2], size=10, rotation=90)
plt.bar_label(ax[0].containers[3], size=10, rotation=90)
plt.yticks(size=10)
plt.legend(fontsize='small')

plt.subplot(1, 2, 2)
isv_stats.plot(x='InternetServicesVolume', y='TotalCharges',
kind='bar', ax=ax[1])
plt.xlabel('Internet Services Volume')
```

```
plt.ylabel('Total Charges')
plt.title('Total Charges Descriptive Statistcs for Internet
Services Volume', size=12)
plt.xticks(rotation=0, size=10)
plt.bar_label(ax[1].containers[0], size=10, rotation=90)
plt.bar_label(ax[1].containers[1], size=10, rotation=90)
plt.bar_label(ax[1].containers[2], size=10, rotation=90)
plt.bar_label(ax[1].containers[3], size=10, rotation=90)
plt.yticks(size=10)
plt.legend(fontsize='small')

plt.tight_layout()

plt.show()

#Percentage of churned and non-churned customers for each
internet services volume.
isv_pct_churn =
(data.groupby('InternetServicesVolume')['Churn'].value_counts()
/ data.groupby('InternetServicesVolume')['Churn'].count() *
100).to_frame().rename(columns={0:'Percentage'}).reset_index()

isv_pct_churn

#Total tickets.
data['numTickets'] = data['numAdminTickets'] +
data['numTechTickets']
data['numTickets'].agg({'sum', 'mean'}).to_frame()


#Avg, median, std, min and max stay duration for churn
customers.
```

```
churn_customers = data[data['Churn'] == 'Yes']
churn_customers[['tenure', 'year']].agg({'mean', 'median',
'std', 'min', 'max'}).transpose()

#Number of customers for each payment method.
cust_payment =
data['PaymentMethod'].value_counts().to_frame().rename(columns={
'count':'Total'})

cust_payment

#Total senior citizen customers for each payment method.
senior_cit_payment = data[data['SeniorCitizen'] ==
'Yes'].groupby('PaymentMethod')['PaymentMethod'].count().to_fram
e().rename(columns={'PaymentMethod':'Total'})

senior_cit_payment

fig, ax = plt.subplots(1, 2, figsize=(25,7))

plt.subplot(1, 2, 1)
cust_payment.plot(kind='pie', autopct='%1.1f%%', y='Total',
startangle=90, legend=False, ax=ax[0])
plt.title('Number of Customers for Each Payment Method',
size=12)

plt.subplot(1, 2, 2)
senior_cit_payment.plot(kind='pie', autopct='%1.1f%%',
y='Total', startangle=90, legend=False, ax=ax[1])
plt.title('Total Senior Citizen Customers for Each Payment
Method', size=12)
```

```
plt.show()

#Number of churn and non churn customers for each payment
method.
churn_payment =
data.groupby('Churn')['PaymentMethod'].value_counts().to_frame()
.rename(columns={'count':'Total'}).reset_index()

churn_payment

fig, ax = plt.subplots(figsize=(10,5))

sns.barplot(data=churn_payment, x='PaymentMethod', y='Total',
hue='Churn', width=0.4)
plt.title('Number of Churn and Non-Churn Customers for Each
Payment Method', size=12)
plt.xlabel('PaymentMethod', size=10)
plt.ylabel('Total', size=10)
plt.bar_label(ax.containers[0], size=10)
plt.bar_label(ax.containers[1], size=10)
plt.legend(fontsize=10)

plt.show()

sns.displot(data, x="tenure", kind="kde", hue="Churn",
fill=True, alpha=0.4, aspect=1.5, height=5, linewidth=False)
plt.title("Distribution of Tenure for Churn and Non-Churn
Customers")
plt.show()
```

```
sns.displot(data, x="MonthlyCharges", kind="kde", hue="Churn",
fill=True, alpha=0.4, aspect=1.5, linewidth=False)
plt.title("Distribution of Monthly Charges for Churn and Non-
Churn Customers")
plt.show()


sns.displot(data, x="TotalCharges", kind="kde", hue="Churn",
fill=True, alpha=0.4, aspect=1.5, linewidth=False)
plt.title("Distribution of Total Charges for Churn and Non-Churn
Customers")
plt.show()

data["TotalCharges"].value_counts()
```

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ 3

```
#Importing libraries.
import pandas as pd
import numpy as np

#Loading dataset.
data =
pd.read_csv('C:/Users/smosc/Desktop/Thesis/data/Telecom_Churn_Ra
te_Dataset_clean.csv')

#Show first 5 rows
data.head()

# Dataset columns
data.columns

# Drop "customerID" column
data.drop('customerID', axis=1, inplace=True)

# Numeric features
numeric_columns = data[['tenure', 'numAdminTickets',
'numTechTickets', 'year', 'TotalCharges',
'MonthlyCharges']].columns.tolist()

# Categorical features
categorical_columns = data.drop(columns=numeric_columns)
categorical_columns.drop('Churn', axis=1, inplace=True)

categorical_columns = categorical_columns.columns.tolist()
```

```
# Replace "No internet service" value with "No"
data.replace({"No internet service":"No"}, inplace=True)

# Replace "No phone service" value with "No" as the information
already exists in column "Phone Service"
data.replace({"No phone service":"No"}, inplace=True)

# Encoding the dataset
columns_to_encode = [
    'InternetService',
    'Contract',
    'PaymentMethod']

data_encoded = pd.get_dummies(data, columns=columns_to_encode)

# Rename columns
data_encoded.rename(columns={
    'InternetService_Fiber optic':'InternetService_FiberOptic',
    'Contract_Month-to-month':'Contract_Monthly',
    'Contract_One year':'Contract_OneYear',
    'Contract_Two year':'Contract_TwoYear',
    'PaymentMethod_Bank transfer
(automatic)':'PaymentMethod_BankTransferAuto',
    'PaymentMethod_Credit card
(automatic)':'PaymentMethod_CreditCardAuto',
    'PaymentMethod_Mailed check':'PaymentMethod_MailedCheck',
    'PaymentMethod_Electronic
check':'PaymentMethod_ElectronicCheck'
}, inplace=True)

data_encoded.columns
```

```
# Transform binary columns to 0 & 1
data_encoded["gender"] =
data_encoded['gender'].replace({'Female':0, 'Male':1})
binary_cols = ['SeniorCitizen', 'Partner', 'Dependents',
'PhoneService', "MultipleLines", "OnlineSecurity",
"OnlineBackup", "DeviceProtection", "TechSupport",
"StreamingTV", "StreamingMovies", 'PaperlessBilling', 'Churn']

for col in binary_cols:
    data_encoded[col] = data_encoded[col].replace({'Yes':1,
'No':0})

data_encoded.info()

for col in data_encoded.columns:
    if data_encoded[col].dtype == "bool":
        data_encoded[col] = data_encoded[col].replace({True:1,
False:0})

# Changing column types to "float"
for col in data_encoded.columns:
    if data_encoded[col].dtype != 'float':
        data_encoded[col] = data_encoded[col].astype('float')

# Revise the values of "InternetService_No"
data_encoded["InternetService_No"] =
data_encoded["InternetService_No"].replace({0:1, 1:0})

# Rename the column as from now contains revised information
data_encoded.rename(columns={'InternetService_No':
'InternetService'}, inplace=True)
```

```
# Checking the dataset info
data_encoded.info()

# Split the data to X (features) and y (target variable)
X = data_encoded.drop(columns='Churn')
y = data_encoded['Churn']

# Import train_test_split from sklearn library
from sklearn.model_selection import train_test_split

# Split data to train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=.2, random_state=123, stratify=y)

# Scaling numeric columns
from sklearn.preprocessing import StandardScaler
stc = StandardScaler()
X_train_scaled = stc.fit_transform(X_train[numeric_columns])
X_test_scaled = stc.fit_transform(X_test[numeric_columns])

# Transform to dataframe
X_train_scaled = pd.DataFrame(X_train_scaled,
columns=numeric_columns)
X_test_scaled = pd.DataFrame(X_test_scaled,
columns=numeric_columns)

X_train.drop(columns=numeric_columns, inplace=True)
X_test.drop(columns=numeric_columns, inplace=True)

# Reset the index for X_train_scaled and X_test_scaled
X_train_scaled.reset_index(drop=True, inplace=True)
X_test_scaled.reset_index(drop=True, inplace=True)
```

```
# Reset the index for X_train and X_test
X_train.reset_index(drop=True, inplace=True)
X_test.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)
y_test.reset_index(drop=True, inplace=True)

# Join scaled numeric columns and categorical columns
X_train = pd.concat([X_train, X_train_scaled], axis=1)
X_test = pd.concat([X_test, X_test_scaled], axis=1)

X_train.to_csv("data_processed/X_train.csv", index=False)
X_test.to_csv("data_processed/X_test.csv", index=False)
y_train.to_csv("data_processed/y_train.csv", index=False)
y_test.to_csv("data_processed/y_test.csv", index=False)
```

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ 4

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, f1_score,
recall_score, precision_score, confusion_matrix, roc_curve, auc
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
from xgboost import plot_importance
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import StratifiedKFold,
learning_curve, GridSearchCV
from sklearn.metrics import RocCurveDisplay
from helper_functions import learningCurvePlot
import shap

def learningCurvePlot(model, cv, X_train, y_train, metric,
title):
# Learning curve function
    train_sizes, train_scores, validation_scores =
learning_curve(estimator=model,

X=X_train,

y=y_train,

train_sizes=np.linspace(0.01, 1.0, 10),
```

```
cv=cv,

n_jobs=-1,

scoring=metric)

    # Calculate mean and standard deviation for training set
scores
    train_score_mean = np.mean(train_scores, axis=1)
    train_score_std = np.std(train_scores, axis=1)

    # Calculate mean and standard deviation for validation set
scores
    validation_score_mean = np.mean(validation_scores, axis=1)
    validation_score_std = np.std(validation_scores, axis=1)

    # Plot learning curves
    plt.figure(figsize=(10, 8))

    plt.plot(train_sizes, train_score_mean, label='Training
score', color='blue')
    plt.fill_between(train_sizes, train_score_mean -
train_score_std,
                     train_score_mean + train_score_std,
alpha=0.1, color="blue")

    plt.plot(train_sizes, validation_score_mean, label='Cross-
validation score', color='green')
    plt.fill_between(train_sizes, validation_score_mean -
validation_score_std,
```

```
                    validation_score_mean +
validation_score_std, alpha=0.1, color="green")

    plt.title(title)
    plt.xlabel('Training Data Size')
    plt.ylabel('Score')
    plt.legend(loc='best')
    plt.grid()

    # Show plot
    return plt.show()

# Import pre-processed datasets
X_train = pd.read_csv("data_processed/X_train.csv")
X_test = pd.read_csv("data_processed/X_test.csv")
y_train = pd.read_csv("data_processed/y_train.csv")
y_test = pd.read_csv("data_processed/y_test.csv")

# Specify the Cross Validation method and parameters
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Create a dataframe to store the results
results_df = pd.DataFrame(columns=["model", "accuracy", "f1-
score", "percision", "recall"])

# Set model to an instance
log_regression = LogisticRegression()

# Fit the model to the training data
log_regression.fit(X_train, y_train)

y_train_pred = log_regression.predict(X_train)
```

```
y_pred = log_regression.predict(X_test)



# Train scores
print("Train Set Scores")
print(f"Accuracy Score: {accuracy_score(y_train,
y_train_pred)}")
print(f"F1-Score: {f1_score(y_train, y_train_pred)}")
print(f"Percision Score: {precision_score(y_train,
y_train_pred)}")
print(f"Recall Score: {recall_score(y_train, y_train_pred)}")

# Test scores
print("\nTest Set Scores")
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
print(f"F1-Score: {f1_score(y_test, y_pred)}")
print(f"Percision Score: {precision_score(y_test, y_pred)}")
print(f"Recall Score: {recall_score(y_test, y_pred)}")

# Get the parameters of the model
log_regression.get_params()

# Specify grid with Hyperparameters and values
log_param_grid = {
    'C': [0.001, 0.01, 0.1, 1.0],
    'class_weight': [None, 'balanced'],
    'dual': [False],
    'fit_intercept': [True, False],
    'intercept_scaling': [1],
    'max_iter': [100, 200, 300],
    'multi_class': ['auto', 'ovr', 'multinomial'],
    'penalty': ['l1', 'l2'],
```

```
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag',
'saga'],
    'tol': [0.0001, 0.001, 0.01],
    'warm_start': [False, True]
}

# Set GridSearch
log_grid_search = GridSearchCV(log_regression,
param_grid=log_param_grid, cv=cv, n_jobs=-1, verbose=0)
log_grid_search.fit(X_train, y_train)


# Best estimator and best parameters
print('Best estimator:')
print(log_grid_search.best_estimator_)

print('\nBest parameters:')
print(log_grid_search.best_params_)

# Predictions to train set
y_train_pred = log_grid_search.predict(X_train)

# Predictions to test set
y_pred = log_grid_search.predict(X_test)

# Scores for train and test sets
print("Train Set Scores")
print(f"Accuracy Score: {accuracy_score(y_train,
y_train_pred)}")
print(f"F1-Score: {f1_score(y_train, y_train_pred)}")
print(f"Percision Score: {precision_score(y_train,
y_train_pred)}")
```

```
print(f"Recall Score: {recall_score(y_train, y_train_pred)}")

print("\nTest Set Scores")
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
print(f"F1-Score: {f1_score(y_test, y_pred)}")
print(f"Percision Score: {precision_score(y_test, y_pred)}")
print(f"Recall Score: {recall_score(y_test, y_pred)}")

print("\nConfusion Matrix")
print(confusion_matrix(y_test, y_pred))
log_confusion_matrix=confusion_matrix(y_test, y_pred)

# Store results in a dictionary
log_results = {
    "model": "Logistic Regression",
    "accuracy": accuracy_score(y_test, y_pred),
    "f1-score": f1_score(y_test, y_pred),
    "percision": precision_score(y_test, y_pred),
    "recall": recall_score(y_test, y_pred)
}

# Add them to the results dataframe
results_df.loc[0] = log_results

# Learning Curve Plot for accuracy
learningCurvePlot(model=log_grid_search.best_estimator_,
                  cv = cv,
                  X_train=X_train,
                  y_train=y_train,
                  metric="accuracy",
                  title="Learning Curve for Logistic Regression
(Accuracy)")
```

```
# Learning Curve Plot for F1-score
learningCurvePlot(model=log_grid_search.best_estimator_,
                  cv = cv,
                  X_train=X_train,
                  y_train=y_train,
                  metric="f1",
                  title="Learning Curve for Logistic Regression
(F1-Score)")

# Set model to an instance
xgb = xgb.XGBClassifier()

# # Fit the model to the training data
xgb.fit(X_train, y_train)

y_train_pred = xgb.predict(X_train)
y_pred = xgb.predict(X_test)

# Scores for train and test sets
print("Train Set Scores")
print(f"Accuracy Score: {accuracy_score(y_train,
y_train_pred)}")
print(f"F1-Score: {f1_score(y_train, y_train_pred)}")
print(f"Percision Score: {precision_score(y_train,
y_train_pred)}")
print(f"Recall Score: {recall_score(y_train, y_train_pred)}")

print("\nTest Set Scores")
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
print(f"F1-Score: {f1_score(y_test, y_pred)}")
```

```
print(f"Percision Score: {precision_score(y_test, y_pred)}")
print(f"Recall Score: {recall_score(y_test, y_pred)}")

# Get the parameters of the model
xgb.get_params()

# Specify grid with Hyperparameters and values
xgb_param_grid = {
    'learning_rate': [0.01, 0.1, 0.2],
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 4, 5],
    'min_child_weight': [1, 3, 5],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0]
}

# Set GridSearch
xgb_grid_search = GridSearchCV(xgb, param_grid=xgb_param_grid,
cv=cv, n_jobs=-1, verbose=0)
xgb_grid_search.fit(X_train, y_train)

# Best estimator and best parameters
print('Best estimator:')
print(xgb_grid_search.best_estimator_)

print('\nBest parameters:')
print(xgb_grid_search.best_params_)

# Predictions to train set
y_train_pred = xgb_grid_search.predict(X_train)

# Predictions to test set
```

```
y_pred = xgb_grid_search.predict(X_test)

# Scores for train and test sets
print("Train Set Scores")
print(f"Accuracy Score: {accuracy_score(y_train,
y_train_pred)}")
print(f"F1-Score: {f1_score(y_train, y_train_pred)}")
print(f"Percision Score: {precision_score(y_train,
y_train_pred)}")
print(f"Recall Score: {recall_score(y_train, y_train_pred)}")

print("\nTest Set Scores")
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
print(f"F1-Score: {f1_score(y_test, y_pred)}")
print(f"Percision Score: {precision_score(y_test, y_pred)}")
print(f"Recall Score: {recall_score(y_test, y_pred)}")

print("\nConfusion Matrix")
print(confusion_matrix(y_test, y_pred))
xgb_confusion_matrix=confusion_matrix(y_test, y_pred)

# Store results in a dictionary
xgb_results = {
    "model": "XGBoost",
    "accuracy": accuracy_score(y_test, y_pred),
    "f1-score": f1_score(y_test, y_pred),
    "percision": precision_score(y_test, y_pred),
    "recall": recall_score(y_test, y_pred)
}

# Add them to the results dataframe
results_df.loc[1] = xgb_results
```

```
# Learning Curve Plot for accuracy
learningCurvePlot(model=xgb_grid_search.best_estimator_,
                  X_train=X_train,
                  y_train=y_train,
                  cv=cv,
                  metric="accuracy",
                  title="Learning Curve for XGBoost (Accuracy)")


# Learning Curve Plot for F1-score
learningCurvePlot(model=xgb_grid_search.best_estimator_,
                  X_train=X_train,
                  y_train=y_train,
                  cv=cv,
                  metric="f1",
                  title="Learning Curve for XGBoost (F1-Score)")

# Set model to an instance
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Fit the model to the training data
rf.fit(X_train, y_train)

y_train_pred = rf.predict(X_train)
y_pred = rf.predict(X_test)

# Scores for train and test sets
print("Train Set Scores")
print(f"Accuracy Score: {accuracy_score(y_train,
y_train_pred)}")
```

```
print(f"F1-Score: {f1_score(y_train, y_train_pred)}")
print(f"Percision Score: {precision_score(y_train,
y_train_pred)}")
print(f"Recall Score: {recall_score(y_train, y_train_pred)}")

print("\nTest Set Scores")
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
print(f"F1-Score: {f1_score(y_test, y_pred)}")
print(f"Percision Score: {precision_score(y_test, y_pred)}")
print(f"Recall Score: {recall_score(y_test, y_pred)}")

# Get the parameters of the model
rf.get_params()

# Specify grid with Hyperparameters and values
rf_param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'max_features': ['sqrt', 'log2', None],
    'bootstrap': [True, False],
    'class_weight': [None, 'balanced', 'balanced_subsample'],
    'min_impurity_decrease': [0.0, 0.1, 0.2]
}

# Set GridSearch
rf_grid_search = GridSearchCV(rf, param_grid=rf_param_grid,
cv=cv, n_jobs=-1, verbose=0)
rf_grid_search.fit(X_train, y_train)

# Best estimator and best parameters
print('Best estimator:')
```

```
print(rf_grid_search.best_estimator_)

print('\nBest parameters:')
print(rf_grid_search.best_params_)

# Predictions to train set
y_train_pred = rf_grid_search.predict(X_train)

# Predictions to test set
y_pred = rf_grid_search.predict(X_test)

# Scores for train and test sets
print("Train Set Scores")
print(f"Accuracy Score: {accuracy_score(y_train,
y_train_pred)}")
print(f"F1-Score: {f1_score(y_train, y_train_pred)}")
print(f"Percision Score: {precision_score(y_train,
y_train_pred)}")
print(f"Recall Score: {recall_score(y_train, y_train_pred)}")

print("\nTest Set Scores")
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
print(f"F1-Score: {f1_score(y_test, y_pred)}")
print(f"Percision Score: {precision_score(y_test, y_pred)}")
print(f"Recall Score: {recall_score(y_test, y_pred)}")

print("\nConfusion Matrix")
print(confusion_matrix(y_test, y_pred))
rf_confusion_matrix=confusion_matrix(y_test, y_pred)

# Store results in a dictionary
rf_results = {
```

```
    "model": "Random Forest",
    "accuracy": accuracy_score(y_test, y_pred),
    "f1-score": f1_score(y_test, y_pred),
    "percision": precision_score(y_test, y_pred),
    "recall": recall_score(y_test, y_pred)
}

# Add them to the results dataframe
results_df.loc[2] = rf_results

# Learning Curve Plot for accuracy
learningCurvePlot(model=rf_grid_search.best_estimator_,
                  X_train=X_train,
                  y_train=y_train,
                  cv=cv,
                  metric="accuracy",
                  title="Learning Curve for Random Forest
(Accuracy)")


# Learning Curve Plot for F1-score
learningCurvePlot(model=rf_grid_search.best_estimator_,
                  X_train=X_train,
                  y_train=y_train,
                  cv=cv,
                  metric="f1",
                  title="Learning Curve for Random Forest (F1
Score)")

# Set model to an instance
dt = DecisionTreeClassifier(random_state=42)
```

```
# Fit the model to the training data
dt.fit(X_train, y_train)

y_train_pred = dt.predict(X_train)
y_pred = dt.predict(X_test)

# Scores for train and test sets
print("Train Set Scores")
print(f"Accuracy Score: {accuracy_score(y_train,
y_train_pred)}")
print(f"F1-Score: {f1_score(y_train, y_train_pred)}")
print(f"Percision Score: {precision_score(y_train,
y_train_pred)}")
print(f"Recall Score: {recall_score(y_train, y_train_pred)}")

print("\nTest Set Scores")
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
print(f"F1-Score: {f1_score(y_test, y_pred)}")
print(f"Percision Score: {precision_score(y_test, y_pred)}")
print(f"Recall Score: {recall_score(y_test, y_pred)}")

# Get the parameters of the model
dt.get_params()

# Setting a grid with parameters
dt_param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None],
```

```
    'min_impurity_decrease': [0.0, 0.1, 0.2]
}

# Set GridSearch
dt_grid_search = GridSearchCV(dt, param_grid=dt_param_grid,
cv=cv, n_jobs=-1, verbose=0)
dt_grid_search.fit(X_train, y_train)

# Best estimator and best parameters
print('Best estimator:')
print(dt_grid_search.best_estimator_)

print('\nBest parameters:')
print(dt_grid_search.best_params_)

# Predictions to train set
y_train_pred = dt_grid_search.predict(X_train)

# Predictions to test set
y_pred = dt_grid_search.predict(X_test)

# Scores for train and test sets
print("Train Set Scores")
print(f"Accuracy Score: {accuracy_score(y_train,
y_train_pred)}")
print(f"F1-Score: {f1_score(y_train, y_train_pred)}")
print(f"Percision Score: {precision_score(y_train,
y_train_pred)}")
print(f"Recall Score: {recall_score(y_train, y_train_pred)}")

print("\nTest Set Scores")
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
```

```
print(f"F1-Score: {f1_score(y_test, y_pred)}")
print(f"Percision Score: {precision_score(y_test, y_pred)}")
print(f"Recall Score: {recall_score(y_test, y_pred)}")

print("\nConfusion Matrix")
print(confusion_matrix(y_test, y_pred))
dt_confusion_matrix=confusion_matrix(y_test, y_pred)

# Store results in a dictionary
dt_results = {
    "model": "Decision Tree",
    "accuracy": accuracy_score(y_test, y_pred),
    "f1-score": f1_score(y_test, y_pred),
    "percision": precision_score(y_test, y_pred),
    "recall": recall_score(y_test, y_pred)
}
# Add them to the results dataframe
results_df.loc[3] = dt_results

# Learning Curve Plot for accuracy
learningCurvePlot(model=dt_grid_search.best_estimator_,
                  X_train=X_train,
                  y_train=y_train,
                  cv=cv,
                  metric="accuracy",
                  title="Learning Curve for Decision Tree
(Accuracy)")


# Learning Curve Plot for F1-score
learningCurvePlot(model=dt_grid_search.best_estimator_,
                  X_train=X_train,
```

```
                    y_train=y_train,
                    cv=cv,
                    metric="f1",
                    title="Learning Curve for Decision Tree (F1
Score)")

# Set model to an instance
knn = KNeighborsClassifier(n_neighbors=3)

# Fit the model to the training data
knn.fit(X_train, y_train)

y_train_pred = knn.predict(X_train)
y_pred = knn.predict(X_test)

# Scores for train and test sets
print("Train Set Scores")
print(f"Accuracy Score: {accuracy_score(y_train,
y_train_pred)}")
print(f"F1-Score: {f1_score(y_train, y_train_pred)}")
print(f"Percision Score: {precision_score(y_train,
y_train_pred)}")
print(f"Recall Score: {recall_score(y_train, y_train_pred)}")

print("\nTest Set Scores")
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
print(f"F1-Score: {f1_score(y_test, y_pred)}")
print(f"Percision Score: {precision_score(y_test, y_pred)}")
print(f"Recall Score: {recall_score(y_test, y_pred)}")

# Get the parameters of the model
knn.get_params()
```

```
# Setting a grid with parameters
knn_param_grid = {
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'leaf_size': [10, 20, 30, 40, 50],
    'metric': ['minkowski', 'euclidean', 'manhattan',
'chebyshev', 'hamming', 'canberra', 'braycurtis'],
    'n_neighbors': [1, 3, 5],
    'p': [1, 2, 3],
    'weights': ['uniform', 'distance']
}

# Set GridSearch
knn_grid_search = GridSearchCV(knn, param_grid=knn_param_grid,
cv=cv, n_jobs=-1, verbose=0)
knn_grid_search.fit(X_train, y_train)

# Best estimator and best parameters
print('Best estimator:')
print(knn_grid_search.best_estimator_)

print('\nBest parameters:')
print(knn_grid_search.best_params_)

# Predictions to train set
y_train_pred = knn_grid_search.predict(X_train)

# Predictions to test set
y_pred = knn_grid_search.predict(X_test)

# Scores for train and test sets
print("Train Set Scores")
```

```
print(f"Accuracy Score: {accuracy_score(y_train,
y_train_pred)}")
print(f"F1-Score: {f1_score(y_train, y_train_pred)}")
print(f"Percision Score: {precision_score(y_train,
y_train_pred)}")
print(f"Recall Score: {recall_score(y_train, y_train_pred)}")

print("\nTest Set Scores")
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
print(f"F1-Score: {f1_score(y_test, y_pred)}")
print(f"Percision Score: {precision_score(y_test, y_pred)}")
print(f"Recall Score: {recall_score(y_test, y_pred)}")

print("\nConfusion Matrix")
print(confusion_matrix(y_test, y_pred))
knn_confusion_matrix=confusion_matrix(y_test, y_pred)

# Store results in a dictionary
knn_results = {
    "model": "K-Nearest Neighbors",
    "accuracy": accuracy_score(y_test, y_pred),
    "f1-score": f1_score(y_test, y_pred),
    "percision": precision_score(y_test, y_pred),
    "recall": recall_score(y_test, y_pred)
}

# Add them to the dataframe
results_df.loc[4] = knn_results

# Learning Curve Plot for accuracy
learningCurvePlot(model=knn_grid_search.best_estimator_,
                  X_train=X_train,
```

```
                    y_train=y_train,
                    cv=cv,
                    metric="accuracy",
                    title="Learning Curve for KNN (Accuracy)")

# Learning Curve Plot for F1-Score
learningCurvePlot(model=knn_grid_search.best_estimator_,
                    X_train=X_train,
                    y_train=y_train,
                    cv=cv,
                    metric="f1",
                    title="Learning Curve for KNN (F1 Score)")

# Show results
results_df

# Plot feature importance
plot_importance(xgb_grid_search.best_estimator_)
plt.show()

# Confusion matrix plots for XGBoost.
sns.heatmap(xgb_confusion_matrix, annot=True, fmt='d',
cmap='Blues', linewidths=0.5, annot_kws={'size': 8})
plt.title("XGBoost - Confusion Matrix")
plt.xlabel("Predictions")
plt.ylabel("Actual Labels")
plt.show()

# Produce a shap explainer plot for the best algorithm
explainer = shap.TreeExplainer(xgb_grid_search.best_estimator_)

shap_values = explainer.shap_values(X_train)
```

```
%matplotlib inline
shap.summary_plot(shap_values, X_train, show=False)
plt.show()

# ROC-AUC Curve for the best algorithm.
y_proba = xgb_grid_search.best_estimator_.predict_proba(X_test)

fpr, tpr, thresholds = roc_curve(y_test, y_proba[:, 1])
roc_auc = auc(fpr, tpr)

#Plot ROC Curve for the best algorithm.
fig, ax = plt.subplots(figsize=(8,8))
plt.plot(fpr, tpr, lw=2)
plt.plot([0, 1], [0, 1])
plt.title(f'ROC Curve (AUC = {roc_auc:.2f})')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.show()
```