ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΕΦΑΡΜΟΓΗ ΤΕΧΝΙΚΩΝ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΓΙΑ ΤΗΝ ΠΡΟΒΛΕΨΗ ΚΟΣΤΟΥΣ ΚΙΝΗΤΟΥ ΤΗΛΕΦΩΝΟΥ

Υπό

ΣΙΔΗΡΟΠΟΥΛΟΥ ΜΙΧΑΗΛ

Διπλωματική Εργασία

Υπεβλήθη για την εκπλήρωση μέρους των απαιτήσεων για
την απόκτηση του Διπλώματος Μηχανολόγου Μηχανικού

Βόλος, 2022

Εγκρίθηκε από τα Μέλη της Τριμελούς Εξεταστικής Επιτροπής:

Πρώτος Εξεταστής	Δρ. Γεώργιος Λυμπερόπουλος
(Επιβλέπων)	Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Δεύτερος Εξεταστής	Δρ. Κωσταντίνος Αμπουντώλας
	Αναπληρωτής Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Τρίτος Εξεταστής	Δρ. Γεώργιος Κοζανίδης
	Αναπληρωτής Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες και την ευγνωμοσύνη μου στον επιβλέποντα καθηγητή μου κ. Γεώργιο Λυμπερόπουλο για την ανάθεση του θέματος, για την εμπιστοσύνη που μου έδειξε και για τον χρόνο που διέθεσε, ώστε να καταστεί δυνατή η από μέρους μου διεκπεραίωση της διπλωματικής εργασίας.

Ευχαριστώ τον συνάδελφό μου Τάσο Σεϊτανίδη για την πολύτιμη βοήθειά του στο Word και στην οργάνωση της διπλωματικής εργασίας. Ευχαριστώ επίσης, τους φίλους και συναδέλφους μου Παναγιώτη Αμοργιανό, Φωτεινή Αντωνάκη, Γιώργο Τσολακίδη και Κυριάκο Χαραλαμπάκη για την υποστήριξη τους καθ' όλη την διάρκεια της φοίτησής μου.

Τέλος ευχαριστώ τους γονείς μου Δημήτρη και Μαρία και την αδερφή μου, Χρυσάνθη για την αγάπη τους όλα αυτά τα χρόνια.

ΕΦΑΡΜΟΓΗ ΤΕΧΝΙΚΩΝ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΓΙΑ ΤΗΝ ΠΡΟΒΛΕΨΗ ΚΟΣΤΟΥΣ ΚΙΝΗΤΟΥ ΤΗΛΕΦΩΝΟΥ

ΣΙΔΗΡΟΠΟΥΛΟΣ ΜΙΧΑΗΛ

Τμήμα μηχανολόγων μηχανικών, Πανεπιστήμιο Θεσσαλίας, 2022

Επιβλέπων Καθηγητής: Δρ. Λυμπερόπουλος Γεώργιος

Καθηγητής Στοχαστικών Μεθόδων στη Διοίκηση Παραγωγής

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία μελετάει την απόδοση αλγορίθμων μηχανικής μάθησης σε ένα πρόβλημα ταξινόμησης επιβλεπόμενης μάθησης. Τα δεδομένα που χρησιμοποιήθηκαν είναι χαρακτηριστικά από διάφορα μοντέλα κινητών τηλεφώνων με μεταβλητή στόχο το εύρος τιμής τους. Οι αλγόριθμοι που χρησιμοποιήθηκαν είναι το δέντρο απόφασης, το τυχαίο δάσος, η μέθοδος των κ πλησιέστερων γειτόνων, η γραμμική και η τετραγωνική διακριτική ανάλυση και τέλος, ο προσαρμοστικός και βαθμοτός ταξινομητής ενίσχυσης. Αρχικά παρουσιάζεται η βιβλιογραφική ανασκόπηση που γίνεται επεξήγηση της βασικής ιδέας της μηχανικής μάθησης και στην συνέχεια αναλύεται η μεθοδολογία των ταξινομητών που χρησιμοποιήθηκαν. Ακολουθεί, η ανάλυση του αλγόριθμου που προγραμματίστηκε και η περιγραφή των τριών συνολικά μελετών που πραγματοποιήθηκαν. Η πρώτη αφορά την μέθοδο διαχωρισμού των δεδομένων προσαρμογής – ελέγχου για διάφορα ποσοστά, η δεύτερη αναλύει τον τρόπο επίλυσης των επιμέρους δυαδικών προβλημάτων και τέλος, η τρίτη μελετάει την παραμετροποίηση κάποιων αλγορίθμων μηχανικής μάθησης και το πως επηρεάζεται η απόδοσή τους. Κλείνοντας, γίνεται η σύγκριση της ακρίβειας και του λογαριθμικού σφάλματος των τεχνικών. Ο αλγόριθμος προγραμματίστηκε σε γλώσσα προγραμματισμού «Python» όπου χρησιμοποιήθηκαν και οι κατάλληλες βιβλιοθήκες για την υλοποίηση.

Λέξεις – κλειδιά: Μηχανική μάθηση, κινητά τηλέφωνα, ταξινόμηση, δέντρο απόφασης, τυχαίο δάσος, κ πλησιέστεροι γείτονες, γραμμική και τετραγωνική διακριτική ανάλυση, προσαρμοστικός και βαθμοτός ταξινομητής ενίσχυσης, ακρίβεια, λογαριθμικό σφάλμα.

APPLICATION OF MACHINE LEARNING TECHNIQUES TO PREDICT MOBILE PHONE COSTS

SIDIROPOULOS MICHAIL

Department of Mechanical Engineering, University of Thessaly, 2022

Supervisor: Dr Lyberopoulos Georgios

Professor of Production Management

ABSTRACT

This thesis studies the performance of machine learning algorithms in a supervised learning classification problem. The data used are features from various mobile phone models with the target variable being the range of price. The algorithms used are decision tree, random forest, k nearest neighbors, linear and quadratic discriminant analysis and finally, adaptive boost and gradient boosting classifier. First, the literature review is presented which explains the basic idea of machine learning and then the methodology of the classifiers used is discussed. Next, the analysis of the algorithm that was programmed and a description of three overall studies that were analyzed. The first one, deals with the method of separating the train – test data for different percentages, the second one, analyses how to solve the individual binary problems and finally the third one studies the parameterization of some machine learning algorithms and how they are affected. Finally, the comparison of the accuracy and logarithmic loss of the techniques is made. The algorithm was programmed in Python where appropriate libraries were used for the implementation.

Keywords: Machine learning, mobile phones, classification, decision tree, random forest, k nearest neighbors, linear and quadratic discriminant analysis, adaptive boost and gradient boosting classifier, accuracy, logarithmic loss.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΠΙΝΑΚΑΣ ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγή

Στη σύγχρονη εποχή, η χρήση της τεχνολογίας έχει αυξηθεί. Ιδιαίτερα στον τομέα της κινητής τηλεφωνίας η εξέλιξη είναι θεαματική. Πιο συγκεκριμένα κάθε άνθρωπος πλέον, έχει ένα κινητό τηλέφωνο και μάλιστα είναι εξοικειωμένος με την χρήση του, αφού του είναι απαραίτητο σε πολλούς τομείς της ζωής του. Η χρησιμότητα ενός τηλεφώνου ποικίλει ανάλογα με τα χαρακτηριστικά, που ορίζουν τις δυνατότητές του. Μια από τις πιο συνηθισμένες λειτουργίες ενός κινητού είναι η φωτογραφία, επίσης μια πιο περίπλοκη λειτουργία, αφού απαιτεί περισσότερη και πιο γρήγορη επεξεργασία της πληροφορίας, είναι ένα σύστημα πλοήγησης υψηλής ακρίβειας. Επομένως τα χαρακτηριστικά ενός τηλεφώνου είναι το πρωταρχικό στοιχείο που πρέπει να παρατηρήσει κανείς για να καταλάβει αν καλύπτει τις απαιτήσεις και ανάγκες του έτσι ώστε να το αγοράσει. Ένας κατασκευαστής κινητών τηλεφώνων κοστολογεί το προϊόν του σύμφωνα με πολλούς παράγοντες. Για παράδειγμα κόστος παραγωγής, εργατοώρες που απαιτούνται, συντήρηση μηχανικού εξοπλισμού, κόστος πρώτης ύλης και άλλα. Ένα σημαντικό κριτήριο της κοστολόγησης ενός κινητού τηλεφώνου είναι και ο ανταγωνισμός [1].

1.2 Πρόβλημα

Η κοινωνική αποστολή κάθε εταιρίας θα εκπληρωθεί καλύτερα όταν η προσπάθεια που θα καταβληθεί για την παραγωγή των προϊόντων, οι υπηρεσίες καθώς και τα έργα που θα εκτελεσθούν, να είναι φθηνότερα και ποιοτικά καλύτερα. Η μηχανική μάθηση έχει εισχωρήσει τα τελευταία χρόνια σε πολλούς τομείς της επιστήμης και της ανάπτυξης με μία από τις σημαντικότερες εφαρμογές της να είναι η βιομηχανία. Η χρήση αλγορίθμων μηχανικής μάθησης παρέχει την δυνατότητα εξαγωγής πληροφοριών, από παλιότερα δεδομένα, με υψηλότερη ακρίβεια και ικανότητα, από του αλγόριθμους που μπορεί να προγραμματίσει κανείς χειροκίνητα. Αυτή η πρόοδος υπήρξε κυρίως τα τελευταία χρόνια και οφείλεται στον μεγάλο όγκο των διαθέσιμων δεδομένων. Η ικανότητα εξέτασης και κατανόησης ενός μεγάλου συνόλου δεδομένων είναι σχεδόν αδύνατη χωρίς κάποια υποβοήθηση.

Το σύνολο δεδομένων, όπου εφαρμόστηκαν διάφορες τεχνικές μηχανικής μάθησης, προσομοιώνει τη διαδικασία κοστολόγησης ενός προϊόντος και πιο συγκεκριμένα, ενός κινητού τηλεφώνου. Η κύρια ιδέα της παρούσας προπτυχιακής διπλωματική εργασίας είναι η εξαγωγή του εύρους της τιμής ενός κινητού από τα χαρακτηριστικά του. Ας υποτεθεί ότι έχει ανοίξει μια εταιρία και ο στόχος της είναι η πώληση κινητών τηλεφώνων, «χτυπώντας» τις μεγάλες βιομηχανίες («Apple», «Samsung» κλπ.). Ας υποτεθεί επίσης ότι το διευθύνων άτομο της εταιρίας δεν γνωρίζει πως κοστολογούνται τα κινητά τηλέφωνα και έχει συλλέξει δεδομένα με χαρακτηριστικά και κόστος διάφορων μοντέλων από άλλες εταιρίες. Ο στόχος είναι να

προβλέψει την τιμή του προϊόντος του, χρησιμοποιώντας τα δεδομένα και εφαρμόζοντας τεχνικές μηχανικής μάθησης.

1.3 Δομή διπλωματικής εργασίας

Στα πλαίσια της προπτυχιακής διπλωματικής εργασίας μελετήθηκε η εφαρμογή διάφορων τεχνικών μηχανικής μάθησης για την πρόβλεψη του εύρους της τιμής ενός κινητού τηλεφώνου. Αναλυτικότερα, έγινε μελέτη των αποτελεσμάτων ταξινόμησης σε τέσσερα εύρη τιμών (0,1,2,3 αύξουσα τιμή) με την χρήση αλγορίθμων μηχανικής μάθησης όπως τυχαίο δάσος («Decision tree Classifier»), τυχαίο δάσος («Random Forest Classifier»), κ – πλησιέστεροι γείτονες («K Neighbors Classifier»), γραμμική διακριτική ανάλυση(«Linear Discriminant Analysis»), τετραγωνική διακριτική ανάλυση («Quadratic Discriminant Analysis»), προσαρμοστικός ταξινομητής ενίσχυσης («Ada Boost Classifier») και βαθμοτός ταξινομητής ενίσχυσης («Gradient Boosting Classifier»).

Στο Κεφάλαιο 1 γίνεται βιβλιογραφική ανασκόπηση σχετικά με τους τρόπους λειτουργίας κάθε αλγόριθμου μηχανικής μάθησης που χρησιμοποιήθηκε. Πιο συγκεκριμένα, πραγματοποιήθηκε ανάλυση της βασικής ιδέας που ακολουθεί κάθε ταξινομητής για να εξάγει σημαντικά αποτελέσματα για το μέλλον.

Στο Κεφάλαιο 3 παρουσιάζεται, η μεθοδολογία και ο τρόπος υλοποίησης του βασικού αλγόριθμου που παρατίθεται στο παράρτημα. Αρχικά γίνεται αναφορά στο περιβάλλον υλοποίησης, δηλαδή την γλώσσα προγραμματισμού που χρησιμοποιήθηκε καθώς και οι απαραίτητες βιβλιοθήκες που καλέστηκαν. Στη συνέχεια γίνεται μια αναλυτική προεπισκόπηση του συνόλου δεδομένων που χρησιμοποιήθηκε. Παρουσιάζονται όλα τα χαρακτηριστικά σε μορφή πίνακα και γίνεται ανάλυση του κάθε ενός ξεχωριστά. Γίνεται αναφορά στις βασικές παραμέτρους των αλγόριθμων, στις προκαθορισμένες τιμές τους και παρουσιάζονται οι παράμετροι που χρησιμοποιήθηκαν για την εξαγωγή αποτελεσμάτων από παραμετροποιημένες τεχνικές. Επιπροσθέτως, εξηγείται η μελέτη της ακρίβειας που εξάγεται για διάφορους διαχωρισμούς των δεδομένων προσαρμογής – ελέγχου. Αναλύεται μια πρωτοπόρα προσέγγιση του προβλήματος ταξινόμησης πολλών κλάσεων. Τέλος παρουσιάζονται τα βήματα που πραγματοποιήθηκαν για την υλοποίηση του αλγόριθμου και την εξαγωγή των αποτελεσμάτων.

Στο Κεφάλαιο 4 παρουσιάζονται τα αποτελέσματα, από τρεις διαφορετικές μελέτες, της ακρίβειας και του λογαριθμικού σφάλματος για τις τεχνικές μηχανικής μάθησης που χρησιμοποιήθηκαν. Όλα τα αποτελέσματα εξάχθηκαν με μια επαναληπτική διαδικασία (που έγινε εκατό φορές) με αλλαγή της τυχαιότητας για καλύτερη στατιστική ακρίβεια. Αρχικά, παρατίθενται οι γραφικές παραστάσεις τις ακρίβεια και οι πίνακες αποτελεσμάτων από διάφορα ποσοστά διαχωρισμού δεδομένων προσαρμογής – ελέγχου. Στη συνέχεια παρουσιάζονται τα αποτελέσματα από την αντιμετώπιση του προβλήματος σαν τέσσερα προβλήματα δύο κλάσεων, σε γραφικές παραστάσεις της ακρίβειας κάθε τεχνικής. Τέλος εξάγονται τα αποτελέσματα από τρεις παραμετροποιημένες τεχνικές.

Στο Κεφάλαιο 5 παρουσιάζονται τα συμπεράσματα και οι μελλοντικές κατεύθυνσεις.

Στο Κεφάλαιο 6 παρουσιάζεται η βιβλιογραφία, δηλαδή οι πηγές που χρησιμοποιήθηκαν για την διεκπεραίωση της διπλωματικής εργασίας.

Τέλος στο Παράρτημα παρατίθεται ο κώδικας που χρησιμοποιήθηκε για την υλοποίηση της παρούσας διπλωματικής εργασίας.

ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΙΚΕΣ ΤΑΞΙΝΟΜΗΣΗΣ ΣΤΗΝ ΜΗΧΑΝΙΚΗ ΕΚΜΑΘΗΣΗ

Το παρόν κεφάλαιο επικεντρώνεται στο τι είναι μηχανική μάθηση, τα είδη των προβλημάτων μηχανικής μάθησης και οι βασικές εφαρμογές της. Αναλύεται επίσης, η βασική μεθοδολογία που ακολουθεί ο κάθε ταξινομητής που χρησιμοποιήθηκε για την κατάταξη των επιμέρους στοιχείων της βάσης δεδομένων.

2.1 Μηχανική μάθηση και εφαρμογές

Μηχανική μάθηση ή «Machine Learning», είναι ένα μέρος της επιστήμης των υπολογιστών το οποίο αναπτύχθηκε από την μελέτη της αναγνώρισης προτύπων και της υπολογιστικής θεωρίας μάθησης στην τεχνητή νοημοσύνη. Η μηχανική μάθηση μελετά την κατασκευή αλγορίθμων που προσαρμόζουν δεδομένα και εξάγουν πληροφορίες για καινούρια στοιχεία σχετικά με τα δεδομένα. Κατασκευάζοντας μοντέλα από πειραματικά δεδομένα, οι αλγόριθμοι εξάγουν στον προγραμματιστή προβλέψεις ή αποφάσεις σχετικές με το αρχικό σύνολο δεδομένων εκπαίδευσης. Η υπολογιστική στατιστική είναι ένας κλάδος στενά συνδεδεμένος με τεχνικές μηχανικής μάθησης αφού και αυτός επικεντρώνεται στην πρόβλεψη μέσω της χρήσης υπολογιστών. Επίσης, η μηχανική μάθηση συνδέεται άρρηκτα με την μαθηματική βελτιστοποίηση, η οποία παρέχει θεωρίες και μεθοδολογίες που διάφορες τεχνικές μηχανικής μάθησης χρησιμοποιούν για την εξαγωγή των αποτελεσμάτων ή ακόμα και για την μοντελοποίηση του προβλήματος. Μερικές από τις πιο σημαντικές εφαρμογές είναι η αναγνώριση ανεπιθύμητης αλληλογραφίας, η ιατρική διάγνωση, η αναγνώριση ομιλίας ή γραφικού χαρακτήρα, η οικονομία, η βιοπληροφορική και άλλα. [2]

Τα προβλήματα που μπορούν να επιλυθούν χωρίζονται, ανάλογα με τα δεδομένα προσαρμογής ή την «ανατροφοδότηση» σε τρεις μεγάλες κατηγορίες οι οποίες είναι:

- Επιβλεπόμενη μάθηση ή «Supervized learning» είναι τα προβλήματα των οποίων τα δεδομένα εξόδου είναι γνωστά στον αλγόριθμο και προσαρμόζονται και αυτά όπως τα χαρακτηριστικά. Η τεχνική μηχανικής μάθησης δέχεται παραδειγματικές εισόδους καθώς και επιθυμητά αποτελέσματα. Ο στόχος του αλγόριθμου είναι να βρει συσχετισμούς των χαρακτηριστικών με τις αντίστοιχες εξόδους και έτσι, όταν ένα καινούριο δεδομένο απαιτεί ταξινόμηση, να διατρέχει σε αυτούς τους συσχετισμούς και να εξάγει την αντίστοιχη πρόβλεψη.
- Μη επιβλεπόμενη μάθηση ή «Unsupervised learning» είναι τα προβλήματα των οποίων τα δεδομένα εξόδου δεν είναι γνωστά. Ο αλγόριθμος χωρίς να του παρέχεται κάποια εμπειρία, πρέπει να βρει συσχετίσεις μεταξύ των δεδομένων εισόδου. Η μη επιβλεπόμενη μάθηση, μπορεί να είναι αυτοσκοπός (ανακαλύπτοντας κρυμμένα μοτίβα στα χαρακτηριστικά) ή μέσο για ένα τέλος (χαρακτηριστικό μάθησης).

- Ενισχυτική μάθηση ή «Reinforcement learning» είναι η διαδικασία αλληλεπίδρασης ενός προγράμματος με ένα δυναμικό περιβάλλον στο οποίο ο «δάσκαλος» δίνει ελλιπές εκπαιδευτικό σήμα (κάποιες έξοδοι λείπουν).

Στην επιβλεπόμενη μάθηση που ανήκει και το πρόβλημα που απασχολεί την παρούσα διπλωματική ανάγεται σε δύο μεγάλες κατηγορίες. Το πρόβλημα της ταξινόμησης και το πρόβλημα της παλινδρόμησης [3].

- Τα προβλήματα ταξινόμησης ανήκουν στην ευρύτερη κατηγορία προβλημάτων με επίβλεψη καθώς η έξοδος είναι είδη χαρακτηρισμένη. Ο τύπος της εξόδου έχει την μορφή χαρακτηρισμένης ετικέτας, και αντιπροσωπεύει μια ευρύτερη κατηγορία με παρόμοια χαρακτηριστικά, την κλάση. Ο στόχος ενός αλγόριθμου μηχανικής μάθησης σε προβλήματα ταξινόμησης είναι η πρόβλεψη της κλάσης ενός στοιχείου που δεν έχει χαρακτηριστεί. Τα προβλήματα ταξινόμησης μπορούν να γίνουν μεταξύ δύο κλάσεων («Binary Classification problem») ή πολλών κλάσεων («Multi – Class problem»).
- Τα προβλήματα παλινδρόμησης ανήκουν και αυτά στην ευρύτερη κατηγορία προβλημάτων με επίβλεψη. Η διαφορά με τα προβλήματα ταξινόμησης είναι ότι η έξοδος είναι μία συνεχείς τιμή. Ο στόχος ενός αλγόριθμου μηχανικής μάθησης σε αυτά τα προβλήματα είναι η πρόβλεψη μιας τιμής που να προσεγγίζει καλά την πραγματική.

Για να μετρηθεί η αξιοπιστία ενός αλγόριθμου ταξινόμησης επιβλεπόμενης μάθησης υπολογίζεται η ακρίβεια σύμφωνα με τον τύπο 2.1 [4] .

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

Όπου TP και TN το σύνολο των θετικών ή αρνητικών στοιχείων που έχουν χαρακτηριστεί από το μοντέλο και οι πραγματικές τους τιμές είναι όντως θετικές ή αρνητικές αντίστοιχα. FP και FN είναι το σύνολο των θετικών ή αρνητικών στοιχείων που έχουν χαρακτηριστεί από το μοντέλο ενώ οι πραγματικές τιμές δεν είναι θετικές ή αρνητικές αντίστοιχα. Σε ένα πρόβλημα πολλών κλάσεων αντί για θετικές και αρνητικές τιμές θα υπήρχε το όνομα της κλάσης. Γενικότερα η ακρίβεια μπορεί να εκφραστεί και ως ο λόγος του συνόλου που έχει ταξινομηθεί σωστά από το μοντέλο προς το σύνολο των προβλέψεων [5].

Επίσης για την αξιοπιστία των τεχνικών χρησιμοποιείται και το λογαριθμικό σφάλμα F, όπου F_i το λογαριθμικό σφάλμα της κλάσης i, εκφράζεται από τον τύπο 2.2.

$$Log\ Loss = \sum_{j}^{M}(-\frac{1}{N}\sum_{i}^{N} y_{ij} Ln(p_{ij}))) = \sum_{j}^{M} F_i \quad (2.2)$$

Όπου N είναι ο αριθμός των περιπτώσεων, M ο αριθμός των διαφορετικών κλάσεων, y_{ij} η δυαδική μεταβλητή με τις αναμενόμενες κλάσεις και p_{ij} η πιθανότητα ταξινόμησης που εξάγεται από τον ταξινομητή για την i – περίπτωση και την j – κλάση. Γενικά, η συνάρτηση λογαριθμικού σφάλματος F μετρά την απόσταση μεταξύ δύο κατανομών πιθανοτήτων, δηλαδή

πόσο παρόμοια είναι η κατανομή των πραγματικών ετικετών και των πιθανοτήτων του ταξινομητή. Ως εκ τούτου, προτιμώνται τιμές κοντά στο μηδέν [6].

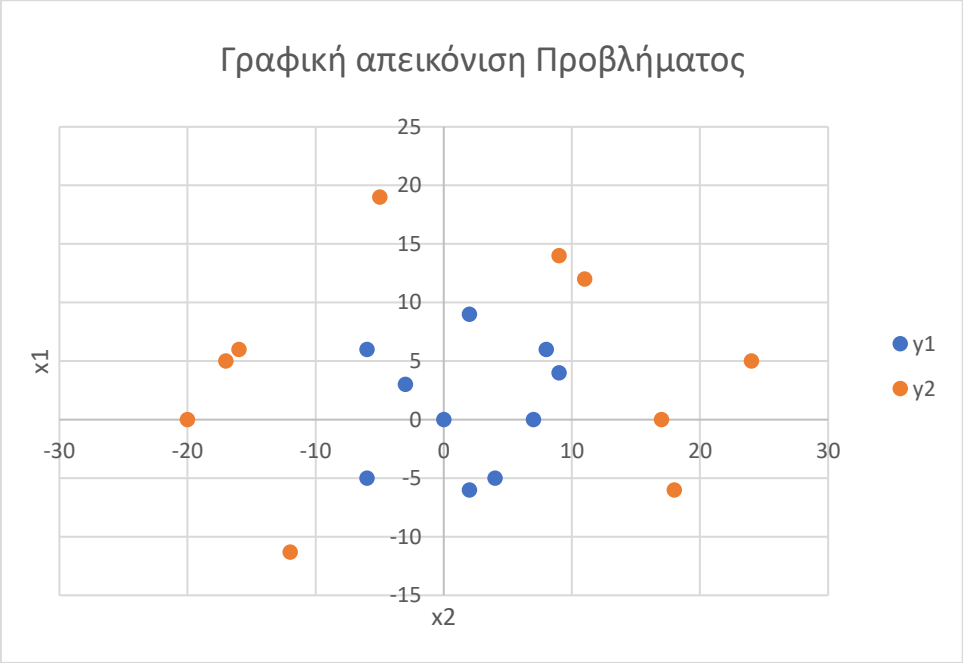
2.2 Δέντρο Απόφασης

Τα δέντρα απόφασης χρησιμοποιούνται με επιτυχία σε πολλούς τομείς. Μερικοί από αυτούς είναι η ταξινόμηση ακινήτων, η ιατρική διάγνωση, η αναγνώριση ομιλίας και χαρακτηριστικών. Το πιο σημαντικό γνώρισμα των δέντρων απόφασης είναι η ικανότητά τους να αναλύουν ένα μεγάλο σύνολο δεδομένων, απλοποιώντας μια πολύπλοκη διαδικασία λήψης αποφάσεων και αποδίδοντας μια λύση που είναι ευκολότερο να ερμηνευθεί. Γενικά ένα δυαδικό δέντρο (με δύο κλάδους) χωρίζει αναδρομικά τα δεδομένα μέχρι να καταλήξει στα φύλλα. Το κάθε φύλλο περιέχει κάποια δεδομένα και έχει αντιστοιχηθεί με μόνο μία κλάση [7].

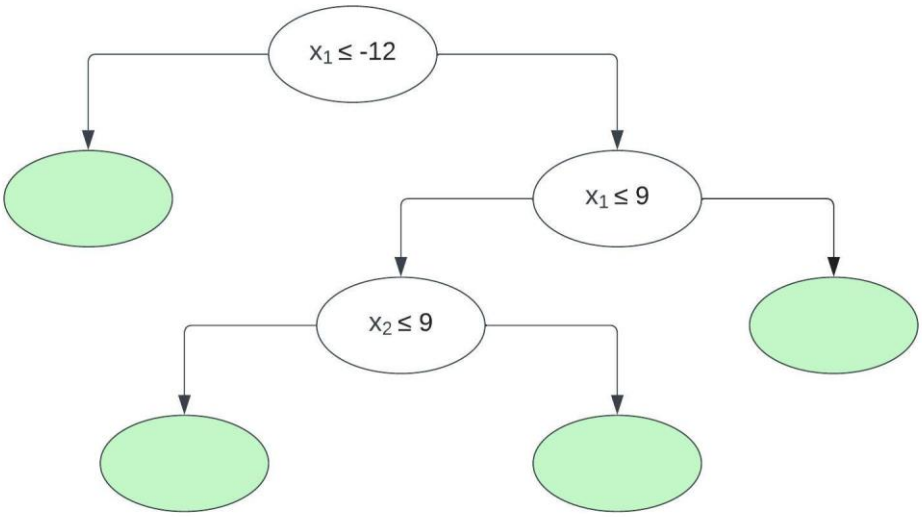
Ο πιο εύκολος τρόπος να περιγράψουμε την λειτουργία ενός δέντρου απόφασης είναι με ένα απλό παράδειγμα. Ας υποτεθεί ότι έχουμε ένα πρόβλημα με δύο χαρακτηριστικά x_1, x_2 και δύο κλάσεις y_1, y_2 όπως περιέχει ο Πίνακας 2. 1. Η Εικόνα 2. 1 είναι μια γραφική απεικόνιση του προβλήματος.

ID	x_1	x_2	y
1	0	0	1
2	-12	-11	2
3	-6	-5	1
4	2	-6	1
5	4	-5	1
6	11	12	2
7	9	14	2
8	2	9	1
9	24	5	2
10	7	0	1
11	-16	6	2
12	-20	0	2
13	-6	6	1
14	-3	3	1
15	-5	19	2
16	18	-6	2
17	9	4	1
18	17	0	2
19	8	6	1
20	-17	5	2

Πίνακας 2. 1: Πίνακας δεδομένων.



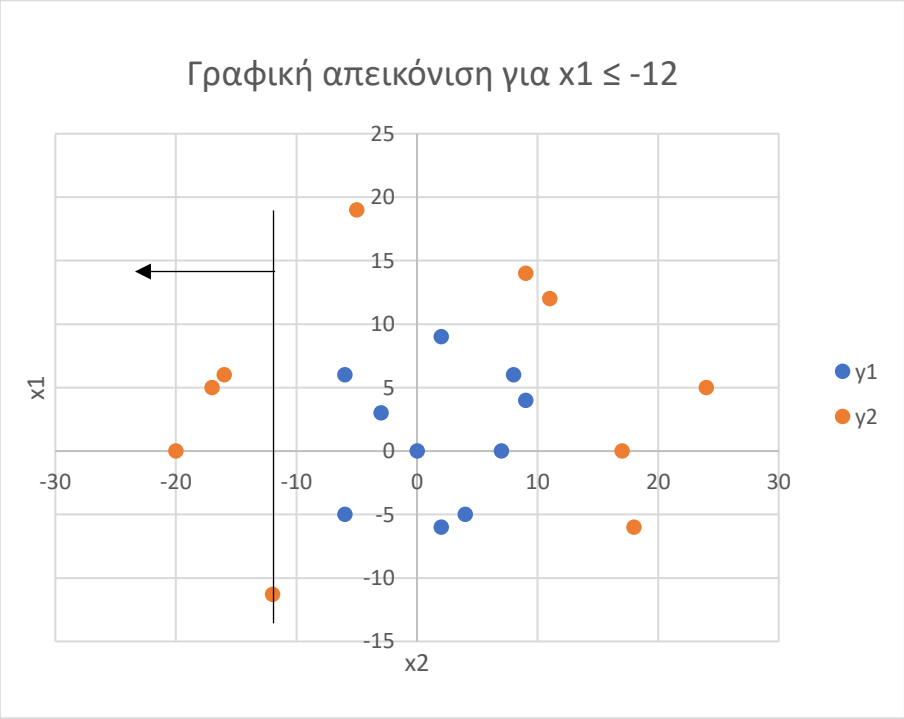
Εικόνα 2. 1: Γράφημα των δεδομένων



Εικόνα 2. 2: Εκπαιδευμένο δέντρο απόφασης

Στην Εικόνα 2. 2 φαίνεται ένα εκπαιδευμένο δέντρο απόφασης. Αρχικά, υπάρχουν δύο είδη κόμβων, οι κόμβοι απόφασης και οι κόμβοι φύλλα. Οι κόμβοι απόφασης περιέχουν έναν περιορισμό ενώ οι κόμβοι φύλλα βοηθούν στην ταξινόμηση ενός καινούριου στοιχείου από τα δεδομένα. Ξεκινώντας από τον πρώτο κόμβο απόφασης και ελέγχοντας ολόκληρο το σύνολο των δεδομένων χωρίζονται τα στοιχεία ανάλογα, αν ικανοποιούν την συνθήκη.

Όπως φαίνεται στην Εικόνα 2. 3 τα στοιχεία που βρίσκονται αριστερά από την πράσινη γραμμή και ικανοποιούν την συνθήκη, βρίσκονται πλέον στον αριστερό κόμβο στο δέντρο ενώ τα υπόλοιπα στοιχεία στον δεξιό. Με την ίδια διαδικασία ελέγχου της κάθε συνθήκης χωρίζονται τα δεδομένα προχωρώντας βαθύτερα στο δέντρο και καταλήγοντας στα φύλλα. Τελικά όλα τα στοιχεία από τα δεδομένα έχουν υποστεί τους περιορισμούς και έχουν τοποθετηθεί σε ένα κόμβο φύλλο. Ο στόχος των περιορισμών είναι να δημιουργηθούν νέοι κόμβοι οι οποίοι αποτελούνται από στοιχεία της ίδιας κλάσης. Αυτοί οι κόμβοι λέγονται καθαροί κόμβοι και δεν χωρίζονται περαιτέρω. Το ερώτημα είναι τι γίνεται με ένα στοιχείο που δεν είναι γνωστή η κλάση στην οποία ανήκει. Στην διαδικασία της ταξινόμησης το στοιχείο υπόκειται στους περιορισμούς των κόμβων απόφασης και καταλήγει σε κάποιο φύλλο. Για παράδειγμα στην Εικόνα 2. 3 φαίνεται η εφαρμογή του περιορισμού του πρώτου φύλλου. Στις περισσότερες περιπτώσεις δεν θα υπάρχουν καθαροί κόμβοι. Αν το στοιχείο καταλήξει σε καθαρό κόμβο – φύλλο τότε κατατάσσεται σε αυτήν την κλάση, αλλιώς κατατάσσεται με βάση την πλειοψηφούσα κλάση των είδη ταυτοποιημένων στοιχείων που ανήκουν στο φύλλο.



Εικόνα 2. 3: Γραφική απεικόνιση του συνόλου δεδομένων με τον περιορισμό x ≤ -12

Το μοντέλο πρέπει να μάθει, ποια χαρακτηριστικά θα λαμβάνει και τις αντίστοιχες σωστές τιμές κατωφλιού για τον βέλτιστο διαχωρισμό των δεδομένων (για αυτό λέγεται και μηχανική εκμάθηση). Για να καταφέρει να αποφασίσει ποιοι είναι οι βέλτιστοι διαχωρισμοί των δεδομένων έχει προεπιλεχθεί ως κριτήριο να μετράει το Gini Impurity προς τιμήν του Ιταλού μαθηματικού και κοινωνιολόγου Corrado Gini,

$$GINI = 1 - \sum_{i=1}^{n} (p_i)^2 \quad (2.3)$$

όπου i ο αριθμός των κλάσεων και p_i η πιθανότητα της κλάσης i.

Το Gini Impurity είναι το μέτρο συχνότητας ενός τυχαία επιλεγμένου στοιχείου από τα δεδομένα, που θα ταξινομηθεί λάθος, αν έχει κατηγοριοποιηθεί τυχαία σύμφωνα με την κατανομή των ετικετών στο υποσύνολο. Οι τιμές που παίρνει είναι από 0 έως 0.5. Στη συνέχεια υπολογίζονται τα Weighted Gini Impurities για τον κάθε διαχωρισμό.

$$WEIGHTED\ GINI = \sum_{i=1}^{n} W_i GI_i \quad (2.4)$$

Όπου i ο αριθμός των κλάσεων, W_i το βάρος της κλάσης i και GI_i το Gini Impurity του κόμβου της κλάσης i. Ο διαχωρισμός με το χαμηλότερο Weighted Gini Impurity επιλέγεται και δημιουργούνται οι επόμενοι δύο κόμβοι.

Ένα ακόμα κριτήριο που μπορεί να χρησιμοποιηθεί για την επιλογή του βέλτιστου διαχωρισμού είναι το κριτήριο «Εντροπία» και κατ' επέκταση το Information Gain.

$$Entropy = \sum_{i=1}^{n} -(p_i) \log_2(p_i) \quad (2.5)$$

Όπου i ο αριθμός των κλάσεων και p_i η πιθανότητα της κλάσης i

$$IG = E(parent) - \sum_{i=1}^{n} w_i E(child_i) \quad (2.6)$$

Όπου E(parent) η εντροπία του δημιουργού του διαχωρισμού, E($child_i$) = η εντροπία του καινούριου κόμβου i και w_i = βάρος του κόμβου i

Η εντροπία μπορεί να πάρει τιμές από 0 έως 1 και είναι το μέτρο των πληροφοριών που περιέχονται σε ένα μέλος. Γενικά το μοντέλο συγκρίνει το IG από κάθε πιθανό διαχωρισμό και επιλέγει το μεγαλύτερο.

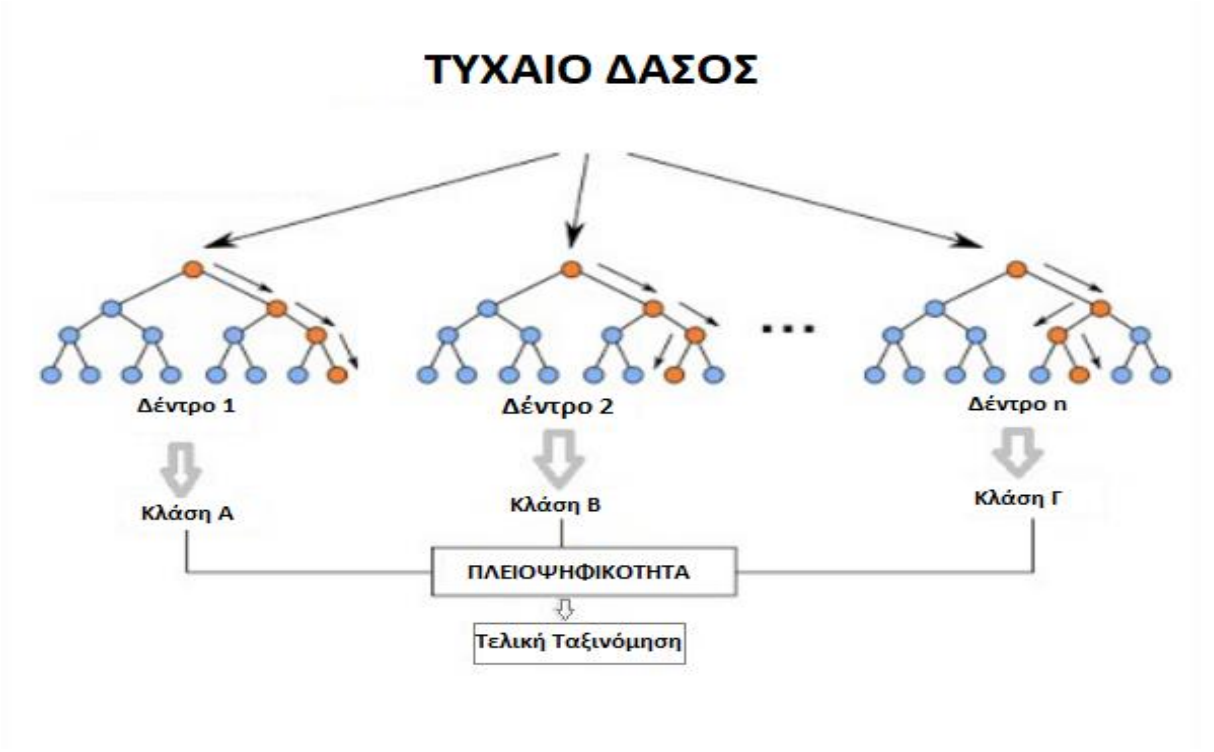
ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ:

1) Ο αλγόριθμος είναι «άπληστος» γιατί επιλέγει προσωρινά τον καλύτερο διαχωρισμό ο οποίος μεγιστοποιεί το WGI ή IG ανάλογα, χωρίς να υπάρχει η δυνατότητα να πάει πίσω και να αλλάξει τον προηγούμενο διαχωρισμό. Αυτό δεν μας εγγυάται τους ιδανικότερους διαχωρισμούς καθώς ο καθένας εξαρτάται από τους προηγούμενους.
2) Η «άπληστη» λειτουργία του ωστόσο κάνει την διαδικασία της εκπαίδευσης γρήγορη (δεν δεσμεύει πολύ μνήμη).
3) Τα αποτελέσματα του μοντέλου είναι αρκετά ακριβή συγκριτικά με την απλή λειτουργία του.
4) Γενικά τα δέντρα απόφασης είναι πολύ ευαίσθητα στα δεδομένα εκπαίδευσης άρα υπάρχει μεγάλη πιθανότητα το μοντέλο να αποτύχει να κάνει γενικοποίηση.

2.3 Τυχαία Δάση

Εν συντομία, ένα τυχαίο δάσος είναι ένας ταξινομητής που αποτελείται από δέντρα απόφασης καθένα από τα οποία παρέχουν μια ψήφο για μια συγκεκριμένη κατηγορία [8]. Ο συνδυασμός ενός μεγάλου αριθμού δέντρων σε ένα τυχαίο δάσος μπορεί να οδηγήσει σε πιο αξιόπιστες προβλέψεις, ενώ ένα μεμονωμένο δέντρο απόφασης μπορεί να προσαρμόσει υπερβολικά τα δεδομένα.

Τα τυχαία δάση χτίζουν καινούρια υποσύνολα δεδομένων από το αρχικό. Ουσιαστικά επιλέγουν τυχαία κάποια στοιχεία από τα αρχικά δεδομένα και δημιουργούν καινούρια, ίδιου μεγέθους (όσα στοιχεία είχε το αρχικό σύνολο τόσα θα έχει και το καινούριο υποσύνολο). Κατά την διαδικασία της δημιουργίας ενός καινούριου υποσυνόλου δεδομένων η επιλογή των στοιχείων είναι τυχαία και επίσης το κάθε στοιχείο μπορεί να επιλεγεί παραπάνω από μία φορά στο υπό-δημιουργία σύνολο δεδομένων. Η διαδικασία αυτή ονομάζεται bootstrapping και τα καινούρια υποσύνολα bootstrapped datasets. Στη συνέχεια το μοντέλο δημιουργεί και εκπαιδεύει δέντρα απόφασης για κάθε υποσύνολο ανεξάρτητα. Η κύρια διαφορά είναι ότι τα δέντρα απόφασης εκπαιδεύονται χρησιμοποιώντας μερικά χαρακτηριστικά από τα καινούρια υποσύνολα. Στην Εικόνα 2. 4 φαίνεται ενδεικτικά ένα εκπαιδευμένο τυχαίο δάσος.



Εικόνα 2. 4: Τυχαίο δάσος

Το ερώτημα είναι , πώς το μοντέλο κάνει μία πρόβλεψη, δηλαδή πώς κατατάσσει ένα καινούριο δεδομένο σε μία από τις υπάρχουσες κλάσεις, έχοντας όλα αυτά τα εκπαιδευμένα δέντρα απόφασης. Ο αλγόριθμος συλλέγει ψήφους για το δεδομένο προς ταξινόμηση, ελέγχοντας τα χαρακτηριστικά του από κάθε εκπαιδευμένο δέντρο. Κάθε δέντρο συμμετέχει στην ταξινόμηση

με μία ψήφο και η πλειοψηφούσα κλάση είναι αυτή που θα επικρατήσει. Η διαδικασία στην οποία συνδυάζονται αποτελέσματα από πολλαπλά μοντέλα καλείται συνάθροιση (aggregation).

Γιατί ονομάζονται τυχαία δάση;

Επειδή έχουν χρησιμοποιηθεί δύο τυχαίες διαδικασίες, το bootstrapping και η επιλογή τυχαίων χαρακτηριστικών από τα δεδομένα.

ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ:

1) Το bootstrapping δηλαδή η δημιουργία καινούριων υποσυνόλων από τα δεδομένα εξασφαλίζει ότι δεν θα χρησιμοποιηθούν τα ίδια στοιχεία για κάθε δέντρο. Αυτό καθιστά το μοντέλο πιο ευέλικτο και λιγότερα ευαίσθητο στα αρχικά δεδομένα.
2) Η επιλογή τυχαίων χαρακτηριστικών βοηθάει στην μείωση της συσχέτισης μεταξύ των δέντρων. Αν χρησιμοποιηθούν όλα τα χαρακτηριστικά, τα περισσότερα δέντρα θα έχουν τα ίδια κλαδιά απόφασης (παρόμοιους διαχωρισμούς) και θα συμπεριφέρονται το ίδιο, με αποτέλεσμα να αυξηθεί η διασπορά.
3) Επίσης εξαιτίας της επιλογής τυχαίων χαρακτηριστικών κάποια δέντρα εκπαιδεύονται με λιγότερο σημαντικά χαρακτηριστικά με αποτέλεσμα να κάνουν λάθος προβλέψεις. Ωστόσο θα υπάρχουν κάποια δέντρα τα οποία θα δίνουν λάθος προβλέψεις αλλά στην αντίθετη κατεύθυνση, έτσι θα εξισορροπούνται και δεν θα παίξουν ρόλο στην τελική ταξινόμηση.

ΠΑΡΑΤΗΡΗΣΗ:

Από έρευνες που έχουν πραγματοποιηθεί έχει διαπιστωθεί ότι χρησιμοποιώντας την ρίζα ή τον λογάριθμο του συνολικού αριθμού των χαρακτηριστικών για την εκπαίδευση των δέντρων οι προβλέψεις θα είναι πιο ακριβής.

2.4 Κ πλησιέστεροι γείτονες

Η ταξινόμηση των πλησιέστερων γειτόνων επίσης γνωστή ως Κ– Πλησιέστεροι – Γείτονες (KNN, K-nearest-neighbors), είναι βασισμένη στην ιδέα ότι τα δεδομένα με τα κοντινότερα μοτίβα σε ένα στοιχείο x, για το οποίο αναζητούμε την κλάση, παρέχουν χρήσιμες πληροφορίες για την ετικέτα. Οι KNN αποδίδουν στο στοιχείο την κλάση της πλειονότητας των Κ – κοντινότερων προτύπων στο χώρο δεδομένων [9]. Για να οριστεί ένα μέτρο ομοιότητας στον χώρο δεδομένων το μοντέλο από προεπιλογή διαλέγει να μετράει την απόσταση χρησιμοποιώντας την μετρική Minkowski (metric Minkowski).

$$\|x_i' - x_i\| = (\sum_{i=1}^{n} |x_i' - x_i|^p)^{1/p} \quad (2.7)$$

Η εξίσωση 2.5 για p=1 ανάγεται στην απόσταση Manhattan

$$\Delta x = \sum_{i=1}^{n} |x_i' - x_i| \quad (2.8)$$

Και για p = 2 ανάγεται στην Ευκλείδεια απόσταση.

$$\Delta x = \sqrt{\sum_{i=1}^{n} [(x_i' - x_i)^2} \quad (2.9)$$

Όπου i το χαρακτηριστικό i, x_i' = η τιμή του χαρακτηριστικού i της μεταβλητής και x_i = η τιμή του χαρακτηριστικού i του δεδομένου

Είναι απαραίτητο να μετρηθούν όλες οι αποστάσεις, δηλαδή της μεταβλητής με κάθε ένα δεδομένο.

Η επιλογή της τιμής k καθορίζει το μέγεθος της γειτονιάς των KNN. Για K = 1, προκύπτουν μικρές γειτονιές στον χώρο, όπου τα δεδομένα από διαφορετικές κλάσεις είναι διάσπαρτα. Για μεγαλύτερα μεγέθη γειτονιάς, π.χ. K = 20, τα πρότυπα με ετικέτες μειονότητας αγνοούνται. Για την καλύτερη κατανόηση της λειτουργίας του μοντέλου θα δοθεί ένα παράδειγμα έχοντας τα δεδομένα που περιέχει ο Πίνακας 2. 1 από το παράδειγμα που δόθηκε στο Κεφάλαιο 2.2 στα δέντρα απόφασης. Ο παρακάτω Πίνακας 2. 2 περιέχει τα δεδομένα του στοιχείου που πρέπει να ταξινομηθεί.

ID	x_1	x_2	y
21	8	4	;

Πίνακας 2. 2: Καινούριο στοιχείο προς ταξινόμηση.

Το id = 1 απέχει:

$$\Delta x = \sqrt{(8-0)^2 + (4-0)^2} = \sqrt{80} = 8{,}94$$

Το id = 2 απέχει:

$$\Delta x = \sqrt{(8-(-12))^2 + (4-(-11))^2} = \sqrt{625} = 25$$

Το id = 3 απέχει:

$$\Delta x = \sqrt{(8-(-6))^2 + (4-(-5))^2} = \sqrt{277} = 16{,}64$$

Το id = 4 απέχει:

$$\Delta x = \sqrt{(8-(2))^2 + (4-(-6))^2} = \sqrt{136} = 11{,}66$$

Ο Πίνακας 2. 3 περιέχει όλες τις αποστάσεις αντίστοιχα.

ID	x_1	x_2	Distance	y
1	0	0	8,944272	1
2	-12	-11	25	2
3	-6	-5	16,64332	1
4	2	-6	11,6619	1
5	4	-5	9,848858	1
6	11	12	8,544004	2
7	9	14	10,04988	2
8	2	9	7,81025	1
9	24	5	16,03122	2
10	7	0	4,123106	1
11	-16	6	24,08319	2
12	-20	0	28,28427	2
13	-6	6	14,14214	1
14	-3	3	11,04536	1
15	-5	19	19,84943	2
16	18	-6	14,14214	2
17	9	4	1	1
18	17	0	9,848858	2
19	8	6	2	1
20	-17	5	25,01999	2

Πίνακας 2. 3: Πίνακας δεδομένων με μέτρηση της απόστασης των χαρακτηριστικών.

Αν ορίσουμε το k = 5 τότε οι πέντε πλησιέστεροι γείτονες είναι τα στοιχεία με id 17, 19, 10, 8, 6 σε αύξουσα σειρά. Αυτά είναι ταξινομημένα με την κλάση 1, 1, 1, 1, 2 αντίστοιχα. Η πλειοψηφούσα κλάση των 5 πλησιέστερων γειτόνων είναι η y1 άρα η νέα ετικέτα που προσάπτεται στο στοιχείο με id 21 είναι η y1. Τέλος αν εισαχθεί καινούρια μεταβλητή για ταξινόμηση τότε στην ψηφοφορία συμμετέχει ενεργά και το στοιχείο 21.

ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ:

1) Ο K-NN είναι αρκετά διαισθητικός και απλός: Ο αλγόριθμος K-NN είναι πολύ απλός στην κατανόηση και εξίσου εύκολος στην εφαρμογή. Για την ταξινόμηση του νέου σημείου δεδομένων ο αλγόριθμος K-NN διαβάζει ολόκληρο το σύνολο δεδομένων για να βρει τους K πλησιέστερους γείτονες.
2) Ο K-NN δεν έχει καμία υπόθεση: Αυτό σημαίνει ότι υπάρχουν προϋποθέσεις που πρέπει να πληρούνται για την εφαρμογή του. Τα παραμετρικά μοντέλα, όπως η γραμμική παλινδρόμηση, έχουν πολλές υποθέσεις που πρέπει να πληρούνται από τα δεδομένα προτού εφαρμοστούν, πράγμα που δεν ισχύει για τον K-NN.
3) Έχει πολύ εύκολη εφαρμογή σε προβλήματα πολλαπλών κλάσεων. Οι περισσότεροι αλγόριθμοι ταξινόμησης είναι εύκολο να εφαρμοστούν για δυαδικά προβλήματα και χρειάζεται προσπάθεια η εφαρμογή τους για πολλαπλές κλάσεις, ενώ ο K-NN προσαρμόζεται σε πολλαπλές κλάσεις χωρίς καμία επιπλέον προσπάθεια.

4) Ο αλγόριθμος K-NN λειτουργεί καλά με μικρό αριθμό μεταβλητών εισόδου, αλλά καθώς ο αριθμός των μεταβλητών αυξάνεται, ο αλγόριθμος δυσκολεύεται να προβλέψει την έξοδο του νέου σημείου δεδομένων.
5) Ο K-NN δεν αποδίδει καλά σε ανισόρροπα δεδομένα. Αν για παράδειγμα θεωρήσουμε δύο κλάσεις, Α και Β, και η πλειοψηφία των δεδομένων εκπαίδευσης έχει επισημανθεί ως Α, τότε το μοντέλο θα δώσει τελικά μεγάλη προτίμηση στην Α. Αυτό μπορεί να έχει ως αποτέλεσμα να ταξινομηθεί λανθασμένα η λιγότερο συχνή κλάση Β.

2.5 Προσαρμοστικός Ταξινομητής Ενίσχυσης (AdaBoost Classifier)

Η μέθοδος Adaptive boost classifier ή αλλιώς προσαρμοστικός ταξινομητής ενίσχυσης είναι ένας μεταεκτιμητής που προσαρμόζει, στο αρχικό σύνολο δεδομένων, διαδοχικά έναν ταξινομητή, και αποσκοπεί στην μείωση της μεροληψίας και της διακύμανσης στην εποπτευόμενη μάθηση. Πιο συγκεκριμένα οι ταξινομητές που προσαρμόζονται κατά προεπιλογή είναι δέντρα απόφασης με έναν μόνο κόμβο – έναν μόνο κορμό (weak learners) και ο κάθε ένας επηρεάζει την προσαρμογή του επόμενου. Για την ευκολότερη κατανόηση του μοντέλου θα δοθεί το παρακάτω γενικό παράδειγμα.

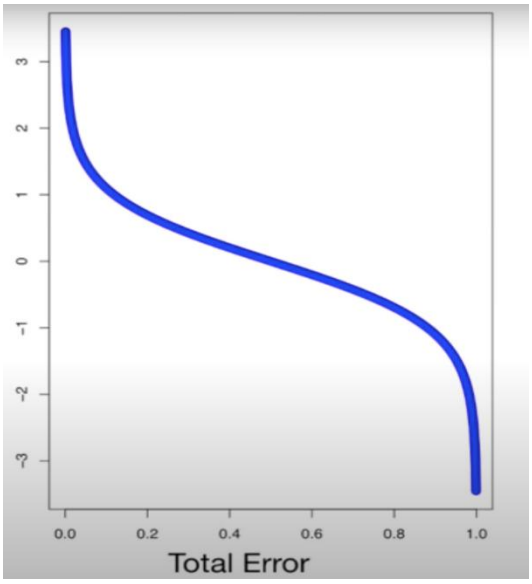
Έστω ότι έχουμε $x_1 \ldots x_n$ χαρακτηριστικά και $y_1 \ldots y_n$ εξόδους. Επίσης θεωρούνται $w_{1,1} \ldots w_{n,1}$, τα βάρη των χαρακτηριστικών. Κατά την διαδικασία της εκπαίδευσης το πρώτο βήμα που κάνει το μοντέλο είναι να δίνει σε κάθε χαρακτηριστικό ένα βάρος. Όλα τα βάρη αρχικά παίρνουν την τιμή 1/n , γεγονός που καθιστά τα χαρακτηριστικά το ίδιο σημαντικά. Έτσι ο αλγόριθμος ξεκινάει να φτιάξει το πρώτο δέντρο απόφασης χτίζοντας σιγά σιγά ένα τυχαίο δάσος. Όπως και στο Κεφάλαιο 2.1 δοκιμάζοντας τυχαίους διαχωρισμούς σε όλα τα χαρακτηριστικά x_n και μετρώντας το συνολικό τους gini (gini index) επιλέγει το καλύτερο δέντρο (το δέντρο με την χαμηλότερη τιμή gini) και το εισάγει στο τυχαίο δάσος. Στη συνέχεια το δεύτερο βήμα είναι να υπολογιστεί η «βαρύτητα του λόγου» του δέντρου (amount of say) χρησιμοποιώντας την εξίσωση 2.8 και 2.9 .

$$E_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^{n} w_{i,t} \quad (2.10)$$

$$\alpha_t = \frac{1}{2} \log\left(\frac{1 - E_t}{E_t}\right) \quad (2.11)$$

Όπου t ο μετρητής της υποθετικής συνάρτησης h_t που ορίζει ένας weak learner, E_t το σταθμισμένο άθροισμα σφάλματος για τα εσφαλμένα ταξινομημένα στοιχεία, h_t το δέντρο (weak learner) που επιλέχθηκε από το μοντέλο και ελαχιστοποιεί την E_t .

Στην Εικόνα 2. 5 φαίνεται η καμπύλη της «βαρύτητας του λόγου» του δέντρου συναρτήσει του λόγου των συνολικών λάθος ταξινομημένων χαρακτηριστικών προς το πλήθος τους. Παρατηρείται ότι όταν ένα δέντρο κάνει «καλή δουλειά» το συνολικό σφάλμα είναι μικρό άρα η τιμή του amount of say είναι σχετικά μεγάλη και θετική [10].



Εικόνα 2. 5: Γραφική απεικόνιση της α_t .

Το τρίτο και επαναλαμβανόμενο βήμα της εκπαίδευσης της μεθόδου AdaBoost είναι η τροποποίηση των βαρών έτσι ώστε το επόμενο δέντρο που θα δημιουργηθεί να λάβει υπόψιν του τα λάθη που έγιναν από το προηγούμενο. Όταν δημιουργήθηκε το πρώτο δέντρο όλα τα βάρη των χαρακτηριστικών ήταν ίσα. Μετά την δημιουργία του πρώτου δέντρου η βαρύτητα αλλάζει με στόχο το καινούριο δέντρο να δώσει έμφαση στα χαρακτηριστικά που ταξινομήθηκαν εσφαλμένα. Τα νέα βάρη αλλάζουν σύμφωνα με την εξίσωση 2.10 .

$$w_{i,t+1} = w_{i,t} e^{-y_i \alpha_t h_t(x_i)} \quad (2.12)$$

Όπου $i = 1 \dots n$

$y \in \{-1,1\}$

$h: x \rightarrow \{-1,1\}$

Παρατηρείται ότι, αν θέλουμε να υπολογίσουμε το καινούριο βάρος ενός χαρακτηριστικού, που θα έχει ταξινομηθεί σωστά από το δέντρο, οι τιμές που θα λαμβάνουν τα y και h θα είναι ομόσημες, άρα τελικώς η συνάρτηση θα είναι φθίνουσα. Αυτό υποδεικνύει ότι όσο αυξάνεται το amount of say το καινούριο βάρος θα έχει μικρότερη τιμή. Λογικό, αφού το μοντέλο θέλει να δώσει περισσότερη «βαρύτητα» στα χαρακτηριστικά που ταξινομήθηκαν εσφαλμένα. Αντίστοιχα, για ένα χαρακτηριστικό που έχει ταξινομηθεί με λανθασμένη ετικέτα, θα είναι αύξουσα και κατ' επέκταση όσο αυξάνεται το amount of say τόσο θα αυξάνεται και το νέο βάρος.

Στο σημείο αυτό αξίζει να σημειωθεί ότι το άθροισμα όλων των βαρών πρέπει να ισούται με την μονάδα άρα τα καινούρια βάρη πρέπει να κανονικοποιηθούν έτσι ώστε να ισχύει η εξίσωση 2.11 .

$$\sum_i w_{i,t+1} = 1 \quad (2.13)$$

Με βάση τα νέα βάρη που αποδίδονται στα δεδομένα εκπαίδευσης, δημιουργείται το νέο δείγμα δεδομένων κοκ. Είναι πιθανό τα δεδομένα εκπαίδευσης που έχουν υψηλό βάρος (κακώς ταξινομημένα) να ληφθούν πολλές φορές και, ως εκ τούτου, το νέο δείγμα δεδομένων θα έχει διπλά αντίγραφα των κακώς ταξινομημένων δεδομένων. Τελικά, όταν ένα μη επισημειωμένο στοιχείο απαιτεί ταξινόμηση, το μοντέλο αθροίζει τα «βάρη του λόγου» των δέντρων που αντιπροσωπεύουν μια ετικέτα και τα συγκρίνει. Το μεγαλύτερο άθροισμα ταξινομεί το νέο στοιχείο με την κλάση του.

ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ:

1) Ο AdaBoost μπορεί να χρησιμοποιηθεί για τη βελτίωση της ακρίβειας των αδύναμων ταξινομητών, καθιστώντας το έτσι ευέλικτο. Έχει πλέον επεκταθεί πέρα από τη δυαδική ταξινόμηση και έχει εφαρμοστεί και στην ταξινόμηση κειμένου και εικόνας.
2) Θεωρητικά, ο AdaBoost δεν είναι επιρρεπής στην υπερπροσαρμογή αν και δεν υπάρχει συγκεκριμένη απόδειξη γι' αυτό. Θα μπορούσε να οφείλεται στο γεγονός ότι οι παράμετροι δεν βελτιστοποιούνται από κοινού
3) Η τεχνικές ενίσχυσης μαθαίνουν προοδευτικά άρα είναι σημαντικό να διασφαλιστεί ότι τα δεδομένα είναι ποιοτικά. Το AdaBoost είναι εξαιρετικά ευαίσθητο στα θορυβώδη δεδομένα και στις ακραίες τιμές.

2.6 Βαθμοτός Ταξινομητής Ενίσχυσης (Gradient Boosting Classifier)

Μία ακόμα τεχνική ενίσχυσης είναι ο ταξινομητής ενίσχυσης βαθμίδας ή αλλιώς Gradient Boosting Classifier. Αυτός ο ταξινομητής, όπως και άλλοι στην ίδια κατηγορία, με επαναληπτικό τρόπο χρησιμοποιεί έναν συνδυασμό από «weak learners» με στόχο την δημιουργία ενός «strong learner». Έστω ότι έχει δοθεί ένα σετ δεδομένων εκπαίδευσης $\{(x_i, y_i)\}_{i=1}^{n}$, x τα χαρακτηριστικά και y οι έξοδοι τους. Ο στόχος του μοντέλου είναι να προσδιορίσει μία συνάρτηση $\hat{F}$ η οποία προσεγγίζει καλά τη μεταβλητή εξόδου από τις τιμές των μεταβλητών εισόδου. Αυτό τυποποιείται με την εισαγωγή κάποιας συνάρτησης απωλειών L(y, F(x)) και ελαχιστοποιώντας την 2.14 [11].

$$\hat{F} = arg_F \, min \, E_{x,y} \, [L(y, F(x))] \quad (2.14)$$

Πιο συγκεκριμένα η μέθοδος GBC υποθέτει μια πραγματική τιμή y και αναζητά μια προσέγγιση $\hat{F}$ της μορφής ενός σταθμισμένου αθροίσματος συναρτήσεων $h_i(x)$ από κάποια κλάση H , που ονομάζονται «base» ή «weak learners».

$$\hat{F} = \sum_{i=1}^{n} \gamma_i \, h_i(x) + ct \quad (2.15)$$

Όπου $h_i(x) \in H$ ο i weak learner, ct μία σταθερά (ένας βαθμός ελευθερίας) και γ_i το μέγεθος του βήματος i και προσδιορίζεται από την εξίσωση 2.14

$$\gamma_i = \frac{|(x_i - x_{i-1})^T [\nabla F(x_i) - \nabla F(x_{i-1})|}{\|\nabla F(x_i) - \nabla F(x_{i-1})\|} \quad (2.16)$$

Σύμφωνα με την αρχή της ελαχιστοποίησης του εμπειρικού κινδύνου (empirical risk minimization), η μέθοδος αναζητεί μία προσεγγιστική $\hat{F}$ που ελαχιστοποιεί τον εμπειρικό κίνδυνο. Αυτό επιτυγχάνεται ξεκινώντας με ένα μοντέλο, που αποτελείται από μια σταθερή συνάρτηση F_0 , και σταδιακά επεκτείνεται με «άπληστο» τρόπο.

$$F_0 = arg\ min \sum_{i=1}^{n} L(y_i, \gamma) \quad (2.17)$$

Κατά προεπιλογή ο αλγόριθμος για να μετρήσει το σφάλμα χρησιμοποιεί την συνάρτηση λογαριθμικής απώλειας, η οποία είναι ενδεικτική του πόσο κοντά είναι η πιθανότητα πρόβλεψης στην αντίστοιχη actual/true τιμή. Όσο περισσότερο αποκλίνει η προβλεπόμενη πιθανότητα από την πραγματική τιμή, τόσο υψηλότερη είναι η τιμή της λογαριθμικής απώλειας.

Η επέκταση του αλγόριθμου δηλαδή η «άπληστη» επαναλαμβανόμενη συνάρτηση που χρησιμοποιείται είναι η εξίσωση 2.16 .

$$F_m(x) = F_{m-1}(x) + arg\ min[\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h_m(x_i))] \quad (2.18)$$

Δυστυχώς, η επιλογή της καλύτερης συνάρτησης h σε κάθε βήμα για μια αυθαίρετη συνάρτηση απώλειας L είναι γενικά ένα υπολογιστικά ανέφικτο πρόβλημα βελτιστοποίησης. Ως εκ τούτου, περιορίζουμε την προσέγγισή μας σε μια απλουστευμένη εκδοχή του προβλήματος. Η βασική ιδέα του αλγόριθμου είναι να εφαρμόσει ένα βήμα απότομης καθόδου (steepest descent), δηλαδή η μετακίνηση ενός μικρού βήματος *γ* έτσι ώστε η γραμμική προσέγγιση να παραμείνει έγκυρη. Στα μαθηματικά η μέθοδος steepest descent είναι ένας επαναληπτικός αλγόριθμος βελτιστοποίησης πρώτης τάξης για την εύρεση ενός τοπικού ελαχίστου μιας διαφορίσιμης συνάρτησης. Έτσι εισάγοντας το βήμα *γ* η εξίσωση 2.16 ανάγεται στην 2.17 .

$$F_m = F_{M-1}(x) - \gamma \sum_{i=1}^{n} \nabla F_{m-1} L(y_i, F_{m-1}(x_i)) \quad (2.19)$$

Όπου m = 1,2,…,M είναι η επανάληψη m και M η τελευταία επανάληψη άρα και $F_M(x)$ είναι η τελική έξοδος.

ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ:

1) Συχνά παρέχει ακρίβεια πρόβλεψης που δεν μπορεί να ξεπεραστεί από άλλες τεχνικές ταξινόμησης
2) Ο GBC μπορεί να βελτιστοποιήσει διαφορετικές συναρτήσεις απώλειας και παρέχει αρκετές επιλογές ρύθμισης υπερπαραμέτρων που καθιστούν την προσαρμογή της συνάρτησης πολύ ευέλικτη
3) Δεν είναι απαραίτητη η προεπεξεργασία των δεδομένων. Συχνά λειτουργεί εξαιρετικά με κατηγορικές και αριθμητικές τιμές ως έχει.

4) Ο GBC βελτιώνεται συνεχώς για την ελαχιστοποίηση της συνάρτησης σφάλματος. Αυτό μπορεί να υπερτονίσει τις ακραίες τιμές και να προκαλέσει υπερπροσαρμογή.
5) Ο GBC είναι αρκετά δαπανηρός. Συχνά απαιτούνται πολλά δέντρα (>1000) τα οποία μπορεί να είναι εξαντλητικά σε χρόνο και μνήμη.

2.7 Γραμμική και Τετραγωνική Διακριτική ανάλυση (Linear and Quadratic Discriminant Analysis)

Η γραμμική διακριτική ανάλυση(Linear Discriminant Analysis) ή κανονική διακριτική ανάλυση (Normal Discriminant Analysis) είναι μια τεχνική μείωσης διαστάσεων που χρησιμοποιείται συνήθως για προβλήματα ταξινόμησης με επίβλεψη (Supervised Classification). Η τεχνική αυτή μετασχηματίζει τους δύο άξονες (X και Y) σε έναν νέο, προβάλλοντας τα δεδομένα σε αυτόν, με στόχο να μεγιστοποιηθεί ο διαχωρισμός των δύο κατηγοριών και συνεπώς να μειωθούν οι διαστάσεις. Για να πραγματοποιηθεί αυτό πρέπει να ικανοποιούνται δύο κριτήρια.

1) Μεγιστοποίηση της συνολικής διακύμανσης.
2) Ελαχιστοποίηση της διακύμανσης κάθε κλάσης.

Τα δύο κριτήρια μετασχηματίζονται σε ένα ενιαίο ως η μεγιστοποίηση της συνολικής διακύμανσης των δύο κλάσεων προς την διακύμανση της κάθε κλάσης.

Για παράδειγμα, ας υποτεθεί ότι υπάρχουν δύο σύνολα δεδομένων που ανήκουν σε δύο διαφορετικές κλάσεις. Για ευκολία κατανόησης, τα σύνολα δεδομένων των δύο κλάσεων αναπαρίστανται ως πίνακες που αποτελούνται από χαρακτηριστικά της μορφής που δίνεται παρακάτω:

$$set1 = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \cdots & \cdots \\ \cdots & \cdots \\ a_{m1} & a_{m2} \end{bmatrix} \quad (2.20)$$

$$set2 = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ \cdots & \cdots \\ \cdots & \cdots \\ b_{m1} & b_{m2} \end{bmatrix} \quad (2.21)$$

Στη συνέχει υπολογίζεται το μέσο όρο κάθε συνόλου δεδομένων και το μέσο όρο ολόκληρου του συνόλου δεδομένων. Έστω μ_1 και μ_2 οι μέσοι όροι των set1 και set2 αντίστοιχα και μ_3 ο μέσος όρος του συνολικού σετ ο οποίος υπολογίζεται σύμφωνα με την εξίσωση 2.20 .

$$\mu_3 = p_1 \times \mu_1 + p_2 \times \mu_2 \quad (2.22)$$

Όπου p_1 και p_2 οι εκ των προτέρων πιθανότητες των κλάσεων. Στην συγκεκριμένη περίπτωση που το πρόβλημα είναι ταξινόμηση δύο κλάσεων οι πιθανότητες ισούνται με 0,5 .

Ακόμα, ο LDA χρησιμοποιεί την διασπορά εντός της κάθε κλάσης και την μεταξύ των κλάσεων, για την ικανοποίηση των κριτηρίων και κατ' επέκταση την αύξηση της διαχωρισιμότητας.

$$S_w = \sum_j p_j \times (cov_j) \quad (2.23)$$

Η εξίσωση 2.21 υπολογίζει την διασπορά εντός των κλάσεων για j κλάσεις και η εξίσωση 2.22 υπολογίζει την διασπορά εντός των δύο κλάσεων για j=2 (δύο κλάσεις) .

$$S_w = 0.5 \times cov_1 + 0.5 \times cov_2 \quad (2.24)$$

Ας υποτεθεί ότι cov_1 και cov_2 είναι συμμετρικοί πίνακες συνδιακύμανση για το σετ 1 και 2 αντίστοιχα και υπολογίζονται χρησιμοποιώντας την εξίσωση 2.23 .

$$cov_j = (x_j - \mu_j)(\mu_j - \mu_3)^T \quad (2.25)$$

Η εξίσωση 2.24 υπολογίζει την διασπορά μεταξύ των κλάσεων.

$$S_b = \sum_j (\mu_j - \mu_3)(\mu_j - \mu_3)^T \quad (2.26)$$

Το S_b μπορεί να θεωρηθεί ως η συνδιακύμανση του συνόλου δεδομένων του οποίου τα μέλη είναι το μέσο διάνυσμα κάθε κλάσης. Όπως αναφέρθηκε στην αρχή του κεφαλαίου 2.6 ο στόχος του LDA είναι να μεγιστοποιήσει το κριτήριο βελτιστοποίησης.

$$κριτήριο = inv(S_w) \times S_b \quad (2.27)$$

«Εξ ορισμού, ένα ιδιοδιάνυσμα ενός μετασχηματισμού αντιπροσωπεύει έναν, μιας διάστασης, αναλλοίωτο υποχώρο του διανυσματικού χώρου στον οποίο εφαρμόζεται ο μετασχηματισμός. Τα ιδιοδιανύσματα των οποίων αντίστοιχες ιδιοτιμές είναι μη μηδενικές, είναι όλα γραμμικά ανεξάρτητα και αναλλοίωτα υπό του μετασχηματισμού. Έτσι, οποιοσδήποτε διανυσματικός χώρος μπορεί να αναπαρασταθεί με όρους γραμμικών συνδυασμών των ιδιοδιανυσμάτων. Μια γραμμική εξάρτηση μεταξύ χαρακτηριστικών υποδεικνύεται από μια μηδενική ιδιοτιμή. Για να προκύψει ένα μη πλεονασματικό σύνολο χαρακτηριστικών, όλα τα ιδιοδιανύσματα που αντιστοιχούν σε μη μηδενικές ιδιοτιμές λαμβάνονται υπόψη και αυτά που αντιστοιχούν σε μηδενικές ιδιοτιμές αγνοούνται», [12]. Στην συγκεκριμένη περίπτωση, ο μετασχηματισμός βρίσκεται ως το ιδιοδιάνυσμα του πίνακα του κριτηρίου που ορίζεται στην εξίσωση 2.25 .

Για κάθε πρόβλημα ν – κλάσεων θα υπάρχουν πάντα ν-1 μη μηδενικές ιδιοτιμές. Αυτό αποδίδεται στους περιορισμούς των μέσων διανυσμάτων των κλάσεων εξίσωση 2.20 . Για το συγκεκριμένο παράδειγμα των δύο κλάσεων, έχοντας λάβει τους πίνακες μετασχηματισμού, μετασχηματίζουμε τα σύνολα δεδομένων χρησιμοποιώντας τον απλό μετασχηματισμό LDA. Η περιοχή απόφασης στο μετασχηματισμένο χώρο είναι μια γραμμή που διαχωρίζει τα αλλαγμένα σύνολα δεδομένων.

$$μετασχηματισμένο_{σύνολο} = μετασχηματισμένος_{χώρος}{}^T \times σύνολο_{δεδομένων}{}^T \quad (2.28)$$

Όταν ολοκληρωθούν οι μετασχηματισμοί, η ευκλείδεια απόσταση (εξίσωση 2.29) χρησιμοποιείται για την ταξινόμηση των καινούριων στοιχείων.

$$απόσταση_n = μετασχηματισμένος_{χώρος_n} \times x - \mu_{μετασχ_n} \quad (2.30)$$

Όπου $\mu_{μετασχ_n}$ ο μέσος όρος του μετασχηματισμένου σετ δεδομένων, n η κλάση και χ το διάνυσμα

Η μικρότερη Ευκλείδεια απόσταση μεταξύ όλων των n κλάσεων ταξινομεί το νέο στοιχείο.

Η LDA είναι μια ειδική περίπτωση της τεχνικής «Τετραγωνική διακριτική ανάλυση» [13] . Η τετραγωνική διακριτική ανάλυση (ή Quadratic Discriminant Analysis) είναι μια τεχνική παρόμοια με την LDA, με βασική διαφορά ότι η υπόθεση ισότητας της μέσης τιμής και της συνδιακύμανσης όλων των κλάσεων να είναι πιο «χαλαρή» [14]. Στην περίπτωση της τεχνικής QDA για κάθε μία από τις κλάσεις j ο πίνακας της συνδιακύμανσης υπολογίζεται από την εξίσωση 2.28.

$$cov_j = \frac{1}{N_j - 1} \sum_{j_i=j} \left(x_i - \mu_j\right) \left(x_i - \mu_j\right)^T \quad (2.31)$$

Προσθέτοντας τον ακόλουθο όρο και επιλύοντας η τετραγωνική συνάρτηση διάκρισης δίνεται από την σχέση 2.29 .

$$\delta_k(x) = \log \pi_k - \frac{1}{2}\mu_k^T inv(cov_k) + x^T inv(cov_k)\mu_k - \frac{1}{2}x^T inv(cov_k)x - \frac{1}{2}\log|cov_k| \quad (2.32)$$

ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ:

1) Οι τεχνικές LDA και QDA είναι απλές και πρωτότυπες για ταξινόμηση. Χρησιμοποιούνται αποστάσεις από τον μέσο όρο της κάθε κλάσης.
2) Η LDA έχει γραμμικό όριο απόφασης. Είναι απλή τεχνική και η ταξινόμηση είναι ισχυρή
3) Η μείωση διαστάσεων παρέχει μια πληροφοριακή οπτική χαμηλής διάστασης για τα δεδομένα, η οποία είναι χρήσιμη τόσο για την οπτικοποίηση όσο και για την μηχανική των χαρακτηριστικών.
4) Κάποιες φορές δεν είναι καλή τεχνική όταν υπάρχουν πολλά κατηγορικά χαρακτηριστικά.
5) Τα γραμμικά όρια απόφασης μπορεί να μην διαχωρίζουν επαρκώς τις κλάσεις. Είναι επιθυμητή η υποστήριξη πιο γενικών ορίων όπως παρέχει η QDA.
6) Είναι απαραίτητη η υποστήριξη για πιο σύνθετη ταξινόμηση πρωτοτύπων.

ΚΕΦΑΛΑΙΟ 3. ΜΕΘΟΔΟΛΟΓΙΑ ΚΑΙ ΑΝΑΛΥΣΗ

Το παρόν κεφάλαιο επικεντρώνεται στην βασική μεθοδολογία που επικράτησε για την υλοποίηση της διπλωματικής εργασίας. Πιο συγκεκριμένα γίνεται εκτενής αναφορά στο περιβάλλον υλοποίησης, και αναλύεται το σύνολο δεδομένων που χρησιμοποιήθηκε. Στη συνέχεια παρουσιάζονται οι αλγόριθμοι που επιλέχθηκαν και οι παράμετροι τους, η μεθοδολογία διαχωρισμού των δεδομένων προσαρμογής – ελέγχου και ο τρόπος επίλυσης του δυαδικού προβλήματος. Τέλος, παρουσιάζονται αναλυτικά τα βήματα υλοποίησης που έγιναν για την εξαγωγή των αποτελεσμάτων.

3.1 Περιβάλλον υλοποίησης

Η υλοποίηση της παρούσας διπλωματικής εργασίας και κατ' επέκταση η πραγματοποίηση των αλγορίθμων μηχανικής μάθησης, έγιναν με την γλώσσα προγραμματισμού Python στην έκδοση 3.8.6 . Παράλληλα χρησιμοποιήθηκαν οι βιβλιοθήκες Pandas, NumPy, csv, Matplotlib, Scikit Learn, Statistics, Math. Το περιβάλλον προγραμματισμού το οποίο χρησιμοποιήθηκε είναι το Visual Studio Code, ένας επεξεργαστής πηγαίου κώδικα της Microsoft για Windows, Linux και MacOS που παρέχεται δωρεάν. Οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι οι ακόλουθες:

1) Pandas: Η βιβλιοθήκη αυτή χρησιμοποιείται για επεξεργασία και αναδιαμόρφωση μεγάλου συνόλου δεδομένων σε ακαδημαϊκούς και εμπορικούς τομείς, όπως τα χρηματοοικονομικά, ανάλυση ιστού, στατιστική και άλλα. Η βιβλιοθήκη pandas είναι ανοιχτή και έχει στόχο να αποτελέσει το θεμελιώδες δομικό στοιχείο για την ανάλυση δεδομένων [15].
2) NumPy: Η βιβλιοθήκη NumPy χρησιμοποιεί αυθαίρετους τύπους δεδομένων επιτρέποντας της να ενσωματώνεται χωρίς προβλήματα και με ταχύτητα σε αρκετές βάσεις δεδομένων. Παρέχει, μεθόδους για πίνακες πολλών διαστάσεων, μαθηματικές συναρτήσεις και έχει την δυνατότητα παραγωγής ψευδοτυχαίων αριθμών. Όπως και η Pandas, η NumPy είναι ανοιχτή βιβλιοθήκη για την Python.
3) CSV: Η βιβλιοθήκη CSV (Comma Separated Values) χρησιμοποιείται στην Python για εισαγωγή, εξαγωγή και επεξεργασία λογιστικών φύλλων και βάσεις δεδομένων.
4) Matplotlib: Η βιβλιοθήκη Matplotlib χρησιμοποιείται για σχεδιασμό στην Python. Μέσο αυτής μπορεί κανείς να κατασκευάσει γραφήματα, γραμμικά διαγράμματα, ιστογράμματα και άλλα. Στον προγραμματισμό μηχανικής μάθησης η συγκεκριμένη βιβλιοθήκη χρησιμοποιείται για αναπαράσταση της βάσης δεδομένων με στόχο την ευκολότερη κατανόηση της και την εύρεση συσχετίσεων μεταξύ των χαρακτηριστικών.
5) Scikit Learn: Η βασική βιβλιοθήκη για το κάλεσμα διάφορων τεχνικών μηχανικής μάθησης. Η Scikit Learn καθιστά τον προγραμματισμό αλγορίθμων μηχανικής μάθησης εύκολη σε χρήστες μη ειδικούς χρησιμοποιώντας μια γλώσσα υψηλού επιπέδου γενικού σκοπού. Δίνεται έμφαση στην απόδοση, την τεκμηρίωση και την συνοχή του API [16].

6) Statistics: Η βιβλιοθήκη Statistics παρέχει συναρτήσεις για τον υπολογισμό μαθηματικών, στατιστικών και αριθμητικών δεδομένων.
7) Math: Η βιβλιοθήκη Math παρέχει βασικές συναρτήσεις μαθηματικών όπως είναι οι τριγωνομετρικές συναρτήσεις, η ρίζα, το λογάριθμο, αλλά και τη σταθερά του Euler, το π και άλλα.

3.2 Σύνολο δεδομένων

Στη σύγχρονη εποχή, η χρήση της τεχνολογίας έχει αυξηθεί ραγδαία, καθώς η εξέλιξή της αυξάνεται εκθετικά. Πιο συγκεκριμένα κάθε άνθρωπος πλέον, είναι φυσικό να έχει ένα κινητό τηλέφωνο και μάλιστα να είναι εξοικειωμένος με την χρήση του αφού του είναι απαραίτητο σε πολλούς τομείς της ζωής του. Η χρησιμότητα ενός κινητού τηλεφώνου ποικίλει, από μία απλή εφαρμογή αριθμητικού υπολογιστή, μέχρι και εφαρμογές πλοήγησης υψηλής ακρίβειας.

Το σύνολο δεδομένων που χρησιμοποιήθηκε για την εκπόνηση αυτής της διπλωματικής εργασία είναι χαρακτηριστικά από κινητά τηλέφωνα και ονομάζεται «Mobile Price Classification» . Η βάση δεδομένων ανήκει στην ιστοσελίδα Kaggle, όπου παλιότερα είχε διεξαχθεί διαγωνισμός μηχανικής μάθησης (Machine Learning Competition) με την συγκεκριμένη βάση δεδομένων [17] . Όπως φαίνεται στην ιστοσελίδα υπάρχουν δύο σύνολα δεδομένων το «Train» και το «Test» . Πιο συγκεκριμένα χρησιμοποιήθηκε μόνο το «Train» αφού το «Test» είναι για να εφαρμόσει ο προγραμματιστής τις τεχνικές της επιλογής του και να ανεβάσει τα αποτελέσματά του στον διαγωνισμό. Γενικά τα χαρακτηριστικά μιας βάσης δεδομένων χωρίζονται σε δύο κατηγορίες. Τα αριθμητικά χαρακτηριστικά τα οποία αντιπροσωπεύουν δεδομένα της μορφής βαθμοτών τιμών, που απεικονίζουν παρατηρήσεις, καταγραφές ή μετρήσεις. Τα κατηγορικά χαρακτηριστικά τα οποία λαμβάνουν τιμές από ένα πεπερασμένο σύνολο, συνήθως είναι δεδομένο σε μορφή κειμένου, το οποίο ανάλογα με το σύνολο τον μοναδικών χαρακτηρισμών μετατρέπεται σε αριθμητική μεταβλητή, που γίνεται κατανοητή από τον ηλεκτρονικό υπολογιστή. Ο Πίνακας 3. 1 και Πίνακας 3. 2 περιέχουν το σύνολο δεδομένων «Train» που αποτελείται από 2000 διαφορετικά κινητά τηλέφωνα με 20 χαρακτηριστικά και μία έξοδο («Price Range») . Τα χαρακτηριστικά που καταγράφηκαν με την σειρά που εμφανίζονται στους πίνακες είναι:

- Battery power: Η συνολική ενέργεια που μπορεί να αποθηκεύσει μια μπαταρία μετρημένη σε mAh.
- Blue: Αν έχει ή όχι «Bluetooth».
- Clock speed: Η ταχύτητα η οποία ο μικροεπεξεργαστής εκτελεί εντολές.
- Dual sim: Αν υποστηρίζει ή όχι διπλή κάρτα «SIM».
- FC: Η ανάλυση την εμπρόσθιας κάμερας σε «Mega Pixels» (Αν είναι μηδέν δεν έχει).
- Four g: Αν υποστηρίζει ή όχι «4G».
- Int memory: Η εσωτερική μνήμη σε «Gigabytes».
- M dep: Το πάχος της συσκευής σε εκατοστά.
- Mobile wt: Το βάρος της συσκευής.
- N cores: Ο αριθμός των πυρήνων του επεξεργαστή.

- Pc: Η ανάλυση της βασικής κάμερας σε «Mega Pixels».
- Px height: Το ύψος ανάλυσης εικονοστοιχείων («Pixel Resolution Height»).
- Px width: Το πλάτος ανάλυσης εικονοστοιχείων («Pixel Resolution Width»).
- Ram: Η χωρητικότητα της μνήμης τυχαίας προσπέλασης σε «Megabyte».
- Sc h: Το ύψος της οθόνης σε εκατοστά.
- Sc w: Το πλάτος της οθόνης σε εκατοστά.
- Talk time: Ο μέγιστος χρόνος ομιλίας που μπορεί να υποστηρίξει 100% φορτισμένη η μπαταρία.
- Three g: Αν υποστηρίζει ή όχι «3G».
- Touch screen: Αν έχει ή όχι οθόνη αφής.
- Wi-Fi: Αν έχει ή όχι κεραία ασύρματης σύνδεσης στο διαδίκτυο.

Battery power	blue	Clock speed	Dual sim	fc	Four g	int memory	M dep	Mobile wt	N cores
842	0	2.2	0	1	0	7	0.6	188	2
1021	1	0.5	1	0	1	53	0.7	136	3
563	1	0.5	1	2	1	41	0.9	145	5
615	1	2.5	0	0	0	10	0.8	131	6
1821	1	1.2	0	13	1	44	0.6	141	2

Πίνακας 3. 1: Πίνακας δεδομένων Μέρος 1ο.

pc	Px height	Px width	ram	Sc h	Sc w	Talk time	Three g	Touch screen	Wi-Fi
2	20	756	2549	9	7	19	0	0	1
6	905	1988	2631	17	3	7	1	1	0
6	1263	1716	2603	11	2	9	1	1	0
9	1216	1786	2769	16	8	11	1	0	0
14	1208	1212	1411	8	2	15	1	1	0

Πίνακας 3. 2: Πίνακας δεδομένων Μέρος 2ο.

Price range
1
2
2
2
1

Πίνακας 3. 3: Πίνακας εξόδου.

Ο Πίνακας 3. 3 περιέχει τις τιμές εξόδου για κάθε σειρά (κινητό τηλέφωνο) αντίστοιχα. Στο σημείο αυτό αξίζει να σημειωθεί ότι η έξοδος δηλαδή η τιμή που καλείται κάθε ταξινομητής να προβλέψει είναι το εύρος της αξίας του κινητού. Υπάρχουν δηλαδή τέσσερις κλάσεις οι οποίες είναι 0, 1, 2, 3 από το πιο φθηνό στο πιο ακριβό αντίστοιχα.

	Battery power	blue	Clock speed	Dual sim	fc	Four g	Int memory
count	2000	2000	2000	2000	2000	2000	2000
mean	1238.51	0.495	1.522	0.509	4.309	0.5215	32.0465
std	439.4182	0.5001	0.816004	0.500035	4.341444	0.499662	18.14572
min	501	0	0.5	0	0	0	2
25%	851.75	0	0.7	0	1	0	16
50%	1226	0	1.5	1	3	1	32
75%	1615.25	1	2.2	1	7	1	48
max	1998	1	3	1	19	1	64

Πίνακας 3. 4: Πίνακας περιγραφής δεδομένων Μέρος 1ο .

	M dep	Mobile wt	N cores	pc	Px height	Px width	ram
count	2000	2000	2000	2000	2000	2000	2000
mean	0.50175	140.249	4.5205	9.9165	645.108	1251.516	2124.213
std	0.288416	35.39966	2.287837	6.064315	443.7808	432.1994	1084.732
min	0.1	80	1	0	0	500	256
25%	0.2	109	3	5	282.75	874.75	1207.5
50%	0.5	141	4	10	564	1247	2146
75%	0.8	170	7	15	947.25	1633	3064
max	1	200	8	20	1960	1998	3998

Πίνακας 3. 5: Πίνακας περιγραφής δεδομένων Μέρος 2ο .

	Sc h	Sc w	Talk time	Three g	Touch screen	Wi-Fi	Price range
count	2000	2000	2000	2000	2000	2000	2000
mean	12.3065	5.767	11.011	0.7615	0.503	0.507	1.5
std	4.213245	4.356398	5.463955	0.426273	0.500116	0.500076	1.118314
min	5	0	2	0	0	0	0
25%	9	2	6	1	0	0	0.75
50%	12	5	11	1	1	1	1.5
75%	16	9	16	1	1	1	2.25
max	19	18	20	1	1	1	3

Πίνακας 3. 6: Πίνακας περιγραφής δεδομένων Μέρος 3ο .

Οι παραπάνω πίνακες παράχθηκαν εκτελώντας την εντολή «describe()» και παρέχουν κάποιες πληροφορίες για ολόκληρο το σύνολο δεδομένων. Πιο συγκεκριμένα για κάθε χαρακτηριστικό, το «count» μετράει πόσα κελιά έχουν τιμές, το «mean» βγάζει τον μέσο όρο από όλα τα κελιά του κάθε χαρακτηριστικού, το «std» μετράει την τυπική απόκλιση, το «min» εμφανίζει την χαμηλότερη τιμή, το «25%» εμφανίζει το ποσοστό 25%, το «50%» εμφανίζει το ποσοστό 50%, το «75%» εμφανίζει το ποσοστό 75% και το «max» εμφανίζει την υψηλότερη τιμή. Οι πίνακες εμφανίζονται για την καλύτερη κατανόηση του συνόλου δεδομένων από τον χρήστη καθώς μπορούν να εξαχθούν κάποιες παρατηρήσεις.

- Από κανένα χαρακτηριστικό δεν λείπει κάποια τιμή.
- Φαίνεται ξεκάθαρα ποια χαρακτηριστικά είναι κατηγορικά καθώς και πόσοι είναι οι μοναδικοί χαρακτηρισμοί τους.

- Τα χαρακτηριστικά «Battery power», «Px height», «Px width» και «ram» αναμένεται να είναι θορυβώδη καθώς η τυπική τους απόκλιση είναι μεγάλη συγκριτικά με τα υπόλοιπα.

Ωστόσο αυτές οι παρατηρήσεις δεν είναι απόλυτες και θα πρέπει να γίνει περαιτέρω ανάλυση για την εξακρίβωση τους.

Στο σημείο αυτό αξίζει να σημειωθεί ότι, το σύνολο δεδομένων είναι απόλυτα ισορροπημένο καθώς τα σύνολα των μοναδικών χαρακτηρισμών της εξόδου («price range») είναι ίσα και συγκεκριμένα αποτελούνται από 500 τιμές το καθένα.

3.3 Επιλογή αλγόριθμων

Για την υλοποίηση της προπτυχιακής διπλωματικής εργασίας οι αλγόριθμοι που χρησιμοποιήθηκαν βρίσκονται στην βιβλιοθήκη της Python, την Scikit Learn. Η επιλογή των αλγόριθμων έγινε βάση πολλών κριτηρίων, διαφορετικών σε κάθε περίπτωση. Οι λόγοι που οδήγησαν στην επιλογή των αλγόριθμων είναι οι εξής:

- Δέντρο απόφασης: Η λειτουργία των δέντρων απόφασης δεν διαφέρει πολύ από τη διαδικασία λήψης αποφάσεων από τον άνθρωπο. Σε συνδυασμό με την απλή τους λειτουργία και την μικρή επιβάρυνση τους σε ένα υπολογιστικό σύστημα, αποτελούν μια καλή πρώτη επιλογή για την επίλυση του συγκεκριμένου προβλήματος ταξινόμησης. Αξίζει να σημειωθεί ότι, τα δέντρα απόφασης χρησιμοποιούνται και για προβλήματα παλινδρόμησης. Έτσι ο αλγόριθμος που επιλέχθηκε είναι ο «DecisionTreeClassifier()» για δέντρο απόφασης ταξινόμησης.
- Τυχαίο Δάσος: Η βασική ιδέα ενός τυχαίου δάσους είναι η εκπαίδευση πολλών δέντρων απόφασης με παράγωγα σύνολα δεδομένων από το αρχικό. Το κάθε εκπαιδευμένο δέντρο παρέχει μία ψήφο για την τελική ταξινόμηση και σε αυτό το χαρακτηριστικό έγκειται και ότι το τυχαίο δάσος δεν υπερποσαρμόζει τα δεδομένα. Έτσι ο αλγόριθμος που επιλέχθηκε για τυχαίο δάσος είναι ο «RandomForestClassifier()»
- Κ – Πλησιέστεροι Γείτονες: Ο αλγόριθμος Κ – Πλησιέστεροι Γείτονες είναι ένας από τους πιο απλούς αλγόριθμους τόσο στην κατανόηση όσο και στην εκτέλεσή του. Πολλές εταιρίες μάλιστα χρησιμοποιούν τον συγκεκριμένο αλγόριθμο για προβλήματα ταξινόμησης. Το βασικό χαρακτηριστικό του είναι ότι μετράει τις αποστάσεις των χαρακτηριστικών και ταξινομεί ένα στοιχείο με βάση την ψήφο των κοντινότερων είδη κατανεμημένων δεδομένων. Ιδανικός για το παρόν σύνολο δεδομένων καθώς υπάρχουν πολλά αριθμητικά χαρακτηριστικά. Όπως είναι προφανές σε περίπτωση εκτέλεσης του KNN σε ένα σύνολο δεδομένων, που υπάρχουν μόνο κατηγορικά χαρακτηριστικά, θα ήταν ανακριβείς. Έτσι επιλέχθηκε ο «KNeighborsClassifier()» για την μέθοδο των Κ πλησιέστερων γειτόνων.
- Τεχνικές Ενίσχυσης: Οι τεχνικές ενίσχυσης εκπαιδεύουν πολλούς αδύναμους ταξινομητές με στόχο την δημιουργία ενός δυνατού. Συνήθως οι αδύναμοι ταξινομητές είναι δέντρα απόφασης ενός κόμβου έχοντας κάποιον συντελεστή βαρύτητας.

Αλγόριθμοι κατάλληλοι για προσαρμογή περίπλοκων δεδομένων με πολλές εξαιρέσεις («outliers») . Έτσι οι τεχνικές που επιλέχθηκαν είναι ο «AdaboostClassifier()» και ο «GradientBoostingClassifier()».

- Τεχνικές μείωσης διαστάσεων: Οι τεχνικές μείωσης διαστάσεων και συγκεκριμένα η μείωση κύριων συνιστωσών καταφέρνει να προβάλει ένα σύνολο δεδομένων από διανύσματα υψηλής διάστασης σε έναν χώρο χαμηλότερης. Αναζητώντας δομές χαμηλής διάστασης στα πολυδιάστατα δεδομένα δίνει μία λύση στο πρόβλημα διαχείρισης δεδομένων. Επίσης οι τεχνικές μείωσης διαστάσεων έχοντας ως σκοπό να αυξήσουν την διαχωρισημότητα επιτυγχάνουν να ξεχωρίσουν τις εξαιρέσεις με αποτέλεσμα να αυξάνεται η ακρίβεια της ταξινόμησης. Έτσι επιλέχθηκαν οι αλγόριθμοι «LinearDiscriminantAnalysis()» και «QuadraticDiscriminantAnalysis()».

3.4 Παράμετροι Αλγορίθμων

Η λειτουργία κάθε τεχνικής είναι ένας περίπλοκος συνδυασμός από σύνθετες μαθηματικές πράξεις και συνθήκες. Έτσι κάθε αλγόριθμος έχει την δυνατότητα ορισμού ή αλλαγής κάποιον από τους υπολογισμούς που πραγματοποιούνται κατά την προσαρμογή – εκπαίδευση των δεδομένων αλλά και κατά την διαδικασία εξαγωγής των αποτελεσμάτων. Η παραμετροποίηση είναι μια διαδικασία που γίνεται από τον προγραμματιστή και χρήζει απόλυτη γνώση τόσο της λειτουργίας όσο και των παραμέτρων. Αλλάζοντας κάποιες παραμέτρους επηρεάζεται η ποιότητα της τεχνικής δηλαδή η ακρίβειά του. Ο στόχος της παραμετροποίησης λοιπόν, είναι η βελτίωση της τεχνικής, έτσι ώστε να παράγει πιο ακριβή αποτελέσματα και να ξεχωρίζει κάποιες εξαιρέσεις («outliers») που με τις προκαθορισμένες παραμέτρους θα ταξινομούσε λάθος. Εκτός από γνώσεις, κάθε πρόβλημα σε συνδυασμό με μία τεχνική απαιτεί μία επαναληπτική διαδικασία με στόχο την εύρεση των βέλτιστων παραμέτρων καθώς κάθε πρόβλημα είναι διαφορετικό και χρειάζεται διαφορετική αντιμετώπιση. Στην παρούσα προπτυχιακή διπλωματική εργασία πραγματοποιήθηκε παραμετροποίηση σε τρείς ταξινομητές που θεωρήθηκαν σημαντικότεροι και είναι οι εξής:

RandomForestClassifier:

Στο τυχαίο δάσος υπάρχουν πολλές παράμετροι πράγμα που δίνει πολλές ελευθερίες στον προγραμματιστή για να προσαρμόσει την τεχνική στο εκάστοτε πρόβλημα [18]. Οι πιο σημαντικές είναι :

- N estimators: Ο απαιτούμενος αριθμός των δέντρων που συμπληρώνουν το δάσος. Ο προκαθορισμένος αριθμός των εκτιμητών (δέντρων) είναι 100.
- Criterion: Η συνάρτηση που μετράει την ποιότητα κάθε διαχωρισμού των δέντρων. Η προκαθορισμένη μεταβλητή είναι το «Gini».
- Max features: Ο μέγιστος αριθμός των χαρακτηριστικών που εξετάζονται για την αναζήτηση του καλύτερου διαχωρισμού. Η προκαθορισμένη μεταβλητή είναι «sqrt».

- Bootstrap: Η παράμετρος αυτή καθορίζει, τα δεδομένα στα οποία εκπαιδεύεται κάθε δέντρο. Αν είναι «True» τα δεδομένα που χρησιμοποιούνται ένας καινούριος συνδυασμός ίδιου μεγέθους με το αρχικό σύνολο. Αν είναι «False» τότε τα δέντρα εκπαιδεύονται με το αρχικό σύνολο δεδομένων.
- N jobs: Ο αριθμός των διεργασιών που τρέχουν παράλληλα κατά τη διαδικασία της προσαρμογής των δεδομένων και της εξαγωγής των αποτελεσμάτων. Αν ισούται με -1 τότε το πρόγραμμα χρησιμοποιεί όλους τους επεξεργαστές.

Μετά από την διαδικασία εύρεσης των βέλτιστων παραμέτρων επιλέχθηκαν:

- ✓ N estimators: 300
- ✓ Criterion: «Entropy»
- ✓ Max features: «log2»
- ✓ Bootstrap: «True»
- ✓ N jobs: -1

KNeighborsClassifier:

Η τεχνική των πλησιέστερων γειτόνων έχει λιγότερες παραμέτρους από αυτή του τυχαίου δάσους λόγο της απλούστερης διαδικασίας που ακολουθεί [19]. Οι παράμετροι που αλλάχθηκαν στην συγκεκριμένη τεχνική είναι:

- N neighbors: Ο αριθμός των γειτόνων που ψηφίζουν για την ταξινόμηση ενός καινούριου στοιχείου. Προτείνεται να επιλέγονται μονοί αριθμοί για να μην υπάρχει περίπτωση ισοψηφίας. Η προκαθορισμένη τιμή είναι 5.
- Weights: Η μεταβλητή που ορίζει την βαρύτητα ψήφου κάθε γείτονα. Ορίζοντάς την «Uniform» η βαρύτητα είναι ίδια για όλους τους γείτονες. Ορίζοντάς την «distance» η ψήφος του κοντινότερου γείτονα έχει μεγαλύτερη βαρύτητα. Η προκαθορισμένη μεταβλητή είναι «distance».
- Metric: Η συνάρτηση που μετράει την απόσταση. Εδώ ο προγραμματιστής μπορεί να καλέσει και εξωτερικές συναρτήσεις που μετράνε απόσταση. Η προκαθορισμένη μεταβλητή είναι «Minkowski».
- P: Αν επιλεχθεί «Minkowski» τότε είναι αναγκαίο να οριστεί και μια τιμή στο p που ορίζει την συνάρτηση.

Μετά από την διαδικασία εύρεσης των βέλτιστων παραμέτρων επιλέχθηκαν:

- ✓ N neighbors: 11
- ✓ Weights: «distance»
- ✓ Metric: «Minkowski»
- ✓ P: 2

AdaBoostClassifier:

Η τεχνική ενίσχυσης χρησιμοποιεί λίγες αλλά αποτελεσματικές παραμέτρους για την προσαρμογή της στο σύνολο δεδομένων [20]. Οι παράμετροι που ρυθμίστηκαν στην συγκεκριμένη τεχνική είναι:

- N estimators: Ο μέγιστος αριθμός των εκτιμητών που η τεχνική ενίσχυσης τερματίζεται. Αν γίνει τέλεια προσαρμογή των δεδομένων η διαδικασία εκπαίδευσης τερματίζεται νωρίτερα. Η προκαθορισμένη τιμή είναι 50 εκτιμητές.
- Learning rate: Η βαρύτητα που δίνεται σε κάθε ταξινομητή ανά επανάληψη. Όσο μεγαλύτερος είναι ο ρυθμός εκμάθησης τόσο αυξάνεται η συνεισφορά κάθε ταξινομητή.

Στην συγκεκριμένη τεχνική πρέπει να βρεθεί ο καλύτερος συνδυασμός αυτών τον παραμέτρων αφού η μια επηρεάζει την άλλη άμεσα. Μετά από την διαδικασία εύρεσης των βέλτιστων παραμέτρων επιλέχθηκαν:

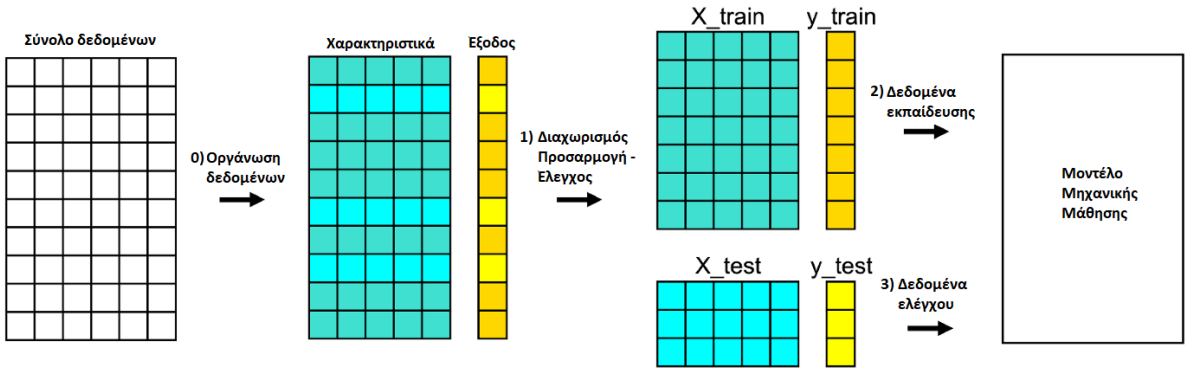
- ✓ N estimators: 387
- ✓ Learning rate: 0.001

3.5 Διαχωρισμός Προσαρμογή – Έλεγχος

Η σημαντικότερη διεργασία για τον προγραμματισμό ενός μοντέλου μηχανικής είναι η διαδικασία διαχωρισμού των δεδομένων προσαρμογής και των δεδομένων ελέγχου. Χρησιμοποιώντας την βιβλιοθήκη «sklearn.model_selection.train_test_split» επιτυγχάνεται αυτόματα η διαδικασία διαχωρισμού των δεδομένων. Χρησιμοποιώντας αυτή τη λειτουργία ουσιαστικά μπορούν να εξαχθούν αποτελέσματα για την ακρίβεια και το σφάλμα των τεχνικών μηχανικής μάθησης, πράγμα που παρέχει στον προγραμματιστή μία πλήρη εικόνα για το επίτευγμά του. Κάποιες σημαντικές παράμετροι της λειτουργίας είναι:

- Train size: Το ποσοστό του συνόλου δεδομένων που παρέχεται στο μοντέλο μηχανικής μάθησης για την εκπαίδευση του.
- Test size: Το ποσοστό του συνόλου δεδομένων που εφαρμόζεται στο μοντέλο μηχανικής μάθησης για ταξινόμηση χωρίς να γνωρίζει τα δεδομένα εξόδου [21].

Όπως φαίνεται στην Εικόνα 3. 1 [22] ο προγραμματιστής αρχικά διαχωρίζει το σύνολο δεδομένων εισάγοντας σε δύο μεταβλητές, τα δεδομένα εισαγωγής, δηλαδή όλα τα χαρακτηριστικά των στοιχείων, και τα δεδομένα εξαγωγής, δηλαδή την κλάση που ανήκει το κάθε στοιχείο σε μορφή «Data Frame» από την βιβλιοθήκη «Pandas». Στην συνέχεια ορίζει το ποσοστό των δύο παραμέτρων που περιεγράφηκαν παραπάνω. Ο αλγόριθμος δημιουργεί δύο τυχαία διαφορετικά υποσύνολα, σύμφωνα με τα ποσοστά που έχουν οριστεί, και τα εισάγει σε τέσσερις μεταβλητές. Τις μεταβλητές «X train», «y train», «X test» και «y test» που περιέχουν τα χαρακτηριστικά και τις εξόδους των δύο καινούριων υποσυνόλων αντίστοιχα. Τέλος τα δύο υποσύνολα εισάγονται στο επιλεγμένο μοντέλο μηχανικής μάθηση για να γίνει η εκπαίδευση και η δοκιμασία.



Εικόνα 3. 1: Αναπαράσταση της λειτουργίας train – test split.

3.6 Τεχνική επίλυσης αντίστοιχων δυαδικών προβλημάτων

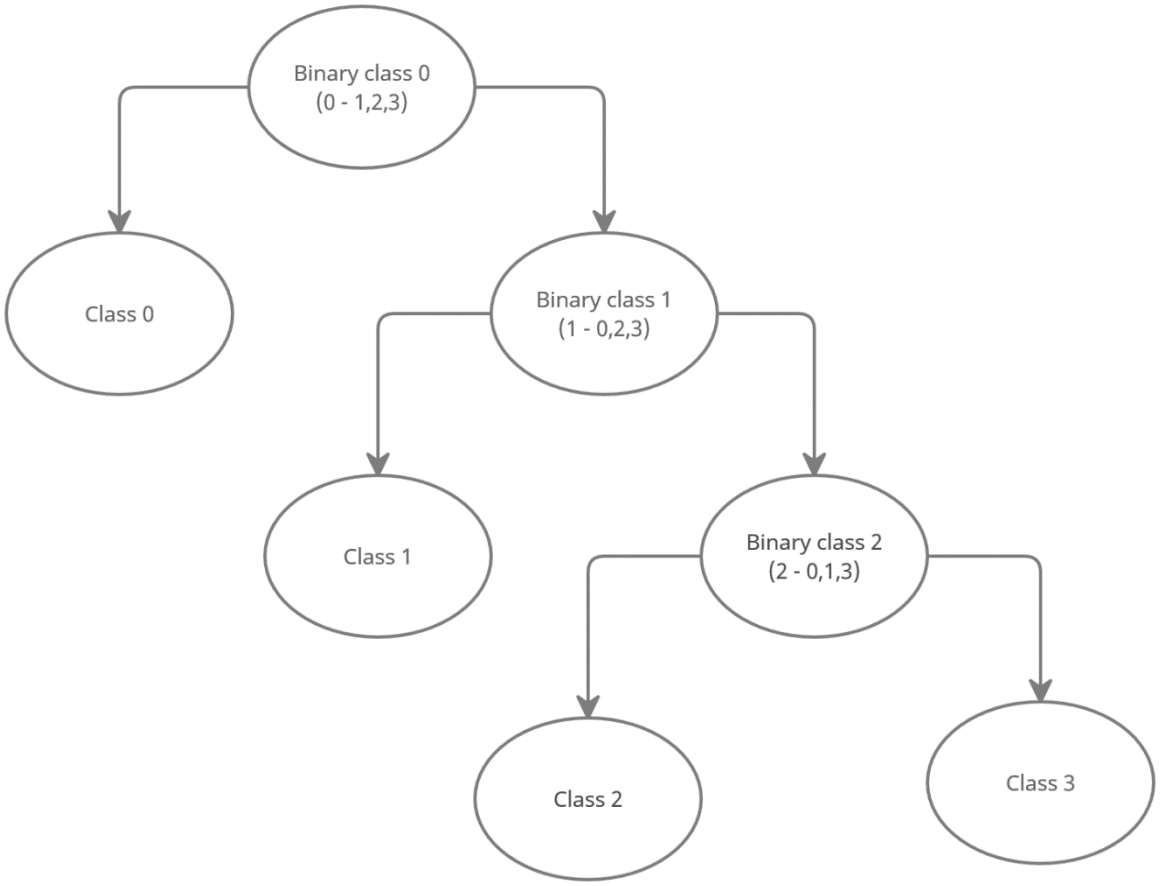
Στην παρούσα διπλωματική εργασία μελετάται και η διαδικασία επίλυσης τεσσάρων επιμέρους προβλημάτων δύο κλάσεων αντί για την επίλυση ενός προβλήματος τεσσάρων [23]. Η γενική ιδέα αυτής της προσέγγισης του προβλήματος είναι ότι, αν υποτεθεί ότι υπάρχει ένα δεδομένο προς ταξινόμηση, τότε η ταξινόμηση θα έχει ως εξής:

1) Ταξινόμηση του στοιχείου στο «Binary class 0». Αν το στοιχείο ταξινομηθεί στην κλάση 0 τότε η διαδικασία τερματίζεται. Αν το στοιχείο ταξινομηθεί στις υπόλοιπες κλάσεις τότε η διαδικασία προχωράει στο βήμα 2.

2) Ταξινόμηση του στοιχείου στο «Binary class 1». Αν το στοιχείο ταξινομηθεί στην κλάση 1 τότε η διαδικασία τερματίζεται. Αν το στοιχείο ταξινομηθεί στις υπόλοιπες κλάσεις τότε η διαδικασία προχωράει στο βήμα 2 και επίσης ο προγραμματιστής γνωρίζει ότι το στοιχείο δεν ανήκει ούτε στην κλάση 0, άρα είναι στην 2 ή 3.

3) Ταξινόμηση του στοιχείου στο «Binary class 2». Αν το στοιχείο ταξινομηθεί στην κλάση 2 τότε η διαδικασία τερματίζεται. Αν το στοιχείο ταξινομηθεί στην κλάση 3 τότε η διαδικασία τερματίζεται. Για επαλήθευση ο προγραμματιστής μπορεί να τρέξει και το «Binary class 3»

Με αυτόν τον τρόπο επιτυγχάνεται υψηλότερη ακρίβεια απ' ότι αυτή του αντίστοιχου προβλήματος με τέσσερις κλάσει. Τα βήματα παρουσιάζονται και στην Εικόνα 3. 2.



Εικόνα 3. 2: Δέντρο μεθοδολογίας επίλυσης του αντίστοιχου δυαδικού προβλήματος.

3.7 Βήματα υλοποίησης

Για την υλοποίηση της προπτυχιακής διπλωματική εργασίας χρησιμοποιήθηκαν οι παραπάνω τεχνικές. Πιο συγκεκριμένα πραγματοποιήθηκαν τα εξής βήματα:

1) Ανάγνωση του αρχικού συνόλου δεδομένων με την χρήση της βιβλιοθήκης «Pandas» και εισαγωγή των τιμών σε «Data Frame».
2) Φόρτωση και προγραμματισμός των μοντέλων μηχανικής μάθησης που θα χρησιμοποιηθούν για την κατηγοριοποίηση σε υπορουτίνες.
3) Διάβασμα των «csv» που θα αποθηκευτούν τα αποτελέσματα.
4) Δημιουργία μιας επανάληψης που βγάζει αποτελέσματα από όλους τους ταξινομητές και αποθηκεύει τις τιμές των «accuracy» και «loss» στα csv, εκατό φορές.

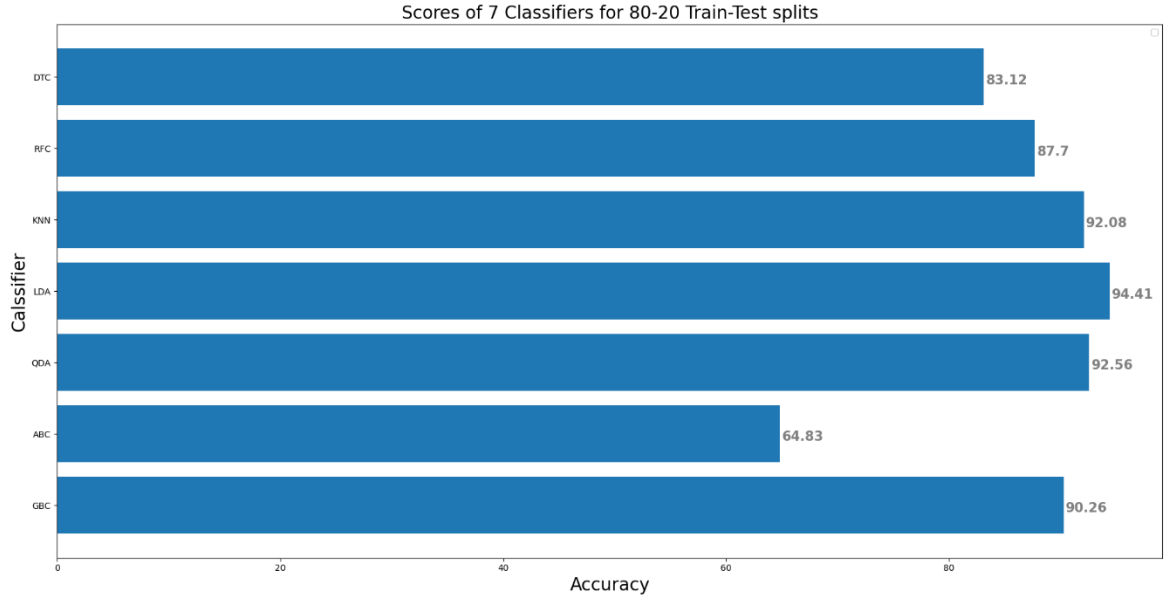
5) Διάβασμα των αποτελεσμάτων και εξαγωγή του μέσου όρου, της διασποράς, της τυπικής απόκλισης και του συντελεστή διακύμανσης σε ένα νέο «csv».
6) Ορισμός του ποσοστού «Train – Test Split» σε 70 – 30, 75 – 25, 80 – 20, 85 – 15 και 90 – 10, εκτέλεση του προγράμματος και εφαρμογή του βήματος 5.
7) Μετατροπή του προβλήματος από τέσσερις κλάσεις σε 2, εκτέλεση του προγράμματος για τις τέσσερις διαφορετικές περιπτώσεις και εφαρμογή του βήματος 5.
8) Εφαρμογή των παραμέτρων που περιγράφηκαν στο Κεφάλαιο 3.4 και εφαρμογή του βήματος 5, καλώντας μόνο τις υπορουτίνες των αντίστοιχων ταξινομητών που παραμετροποιήθηκαν.

ΚΕΦΑΛΑΙΟ 4. ΑΠΟΤΕΛΕΣΜΑΤΑ

Στο παρόν κεφάλαιο της διπλωματικής εργασίας θα γίνει αναφορά και σχολιασμός των αποτελεσμάτων από τις τεχνικές και την μεθοδολογία που προαναφέρθηκαν στο Κεφάλαιο 2 και 3. Τα δεδομένα προσαρμόστηκαν στις τεχνικές μηχανικής μάθησης χωρίς κάποια προεργασία. Αρχικά, θα παρουσιαστούν τα αποτελέσματα από τους διαφορετικούς διαχωρισμούς προσαρμογής – ελέγχου που εφαρμόστηκαν. Στην συνέχεια ελέγχεται η ακρίβεια των αλγορίθμων για το αντίστοιχο δυαδικό πρόβλημα, δηλαδή συμπτύσσοντας τα δεδομένα εξόδου μεταξύ τους έτσι ώστε η ταξινόμηση να γίνει ανάμεσα σε δύο κλάσεις. Τέλος παρουσιάζονται τα αποτελέσματα των παραμετροποιημένων αλγορίθμων και η σύγκρισή τους με το αρχικό πρόβλημα.

4.1 Διαχωρισμός δεδομένων προσαρμογής - ελέγχου

Οι γραφικές παραστάσεις που ακολουθούν παρουσιάζουν όλα τα αποτελέσματα που εξάχθηκαν χρησιμοποιώντας πέντε διαφορετικά ποσοστά προσαρμογής – ελέγχου. Όλοι οι αλγόριθμοι μηχανικής μάθησης, που χρησιμοποιήθηκαν, εφαρμόστηκαν με την χρήση των προκαθορισμένων τιμών των παραμέτρων και προσαρμόζοντας όλα τα χαρακτηριστικά του συνόλου δεδομένων «mobile database». Στην Εικόνα 4. 1 εμφανίζονται στον y άξονα οι όροι «DTC», «RFC», «KNN», «LDA», «QDA», «ABC» και «GBC» που είναι αντίστοιχα το δέντρο απόφασης, το τυχαίο δάσος, ο K πλησιέστερος γείτονας, η γραμμική διακριτική ανάλυση, η τετραγωνική διακριτική ανάλυση, ο προσαρμοστικός ταξινομητής ενίσχυσης, ο βαθμοτός ταξινομητής ενίσχυσης και στον χ άξονα η ακρίβεια κάθε ταξινομητή για διαχωρισμό δεδομένων ελέγχου 80 – 20.



Εικόνα 4. 1: Διάγραμμα της ακρίβειας όλων των τεχνικών για προσαρμογή – έλεγχο 80 – 20.

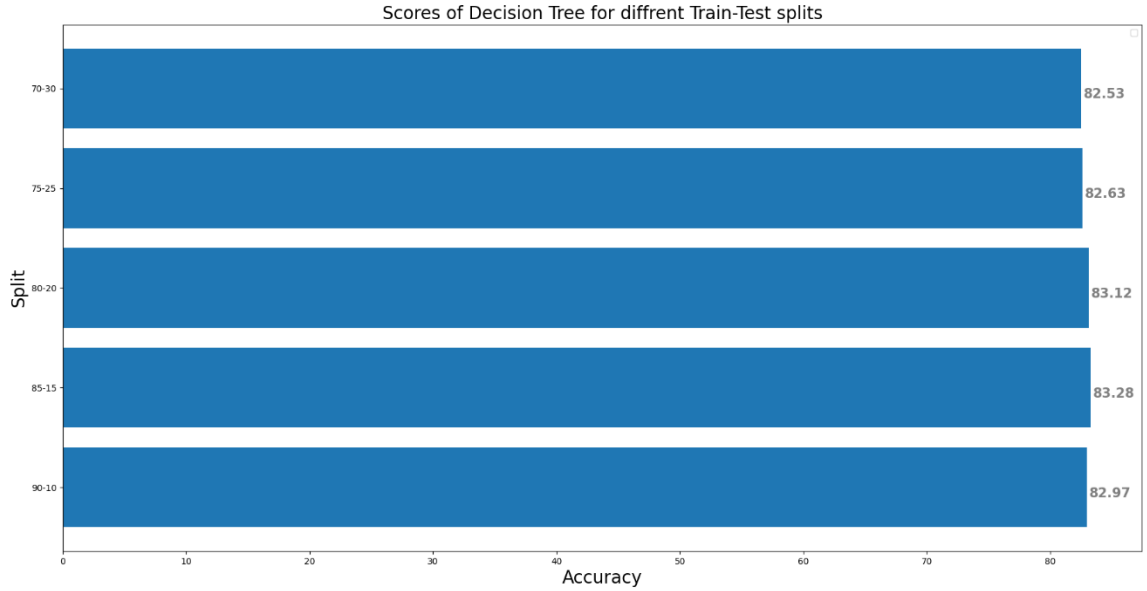
	Decision Tree Classifier	Random Forest Classifier	K Neighbors Classifier	Linear Discriminant Analysis	Quadratic Discriminant Analysis	Ada Boost Classifier	Gradient Boosting Classifier
Mean	83.1175	87.705	92.0775	94.4075	92.565	64.835	90.2625
Variance	3.441231	3.128005	1.489514	1.190221	1.466692	71.76543	1.982797
Standard Deviation	1.855056	1.768617	1.220456	1.090972	1.211071	8.471448	1.408118
Coefficient of Variation	0.022318	0.020166	0.013255	0.011556	0.013083	0.130662	0.0156

Πίνακας 4. 1: Πίνακας μέσου όρου, διασποράς, τυπικής απόκλισης και συντελεστή διακύμανσης της ακρίβειας κάθε ταξινομητή.

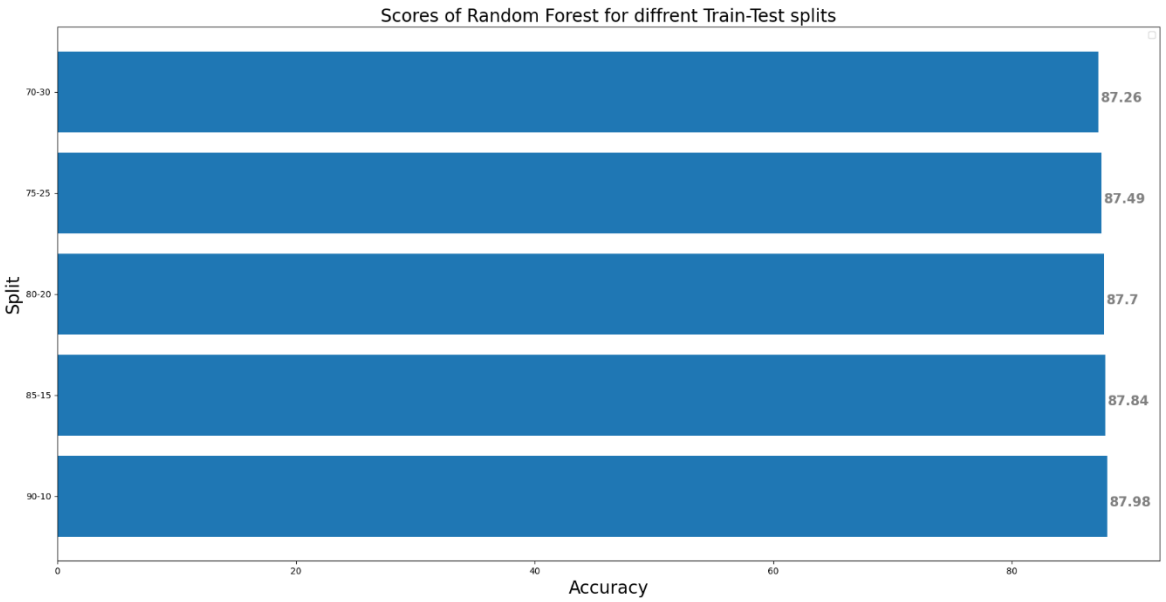
	Decision Tree Classifier	Random Forest Classifier	K Neighbors Classifier	Linear Discriminant Analysis	Quadratic Discriminant Analysis	Ada Boost Classifier	Gradient Boosting Classifier
Mean	5.831009	0.550773	0.359484	0.172375	0.176904	0.831283	0.251939
Variance	0.410514	0.000261	0.012327	0.000184	0.000276	0.006238	0.000733
Standard Deviation	0.640713	0.016151	0.111028	0.013574	0.016622	0.078983	0.027071
Coefficient of Variation	0.10988	0.029325	0.308855	0.078746	0.093961	0.095014	0.107451

Πίνακας 4. 2: Πίνακας μέσου όρου, διασποράς, τυπικής απόκλισης και συντελεστή διακύμανσης του λογαριθμικού σφάλματος κάθε ταξινομητή.

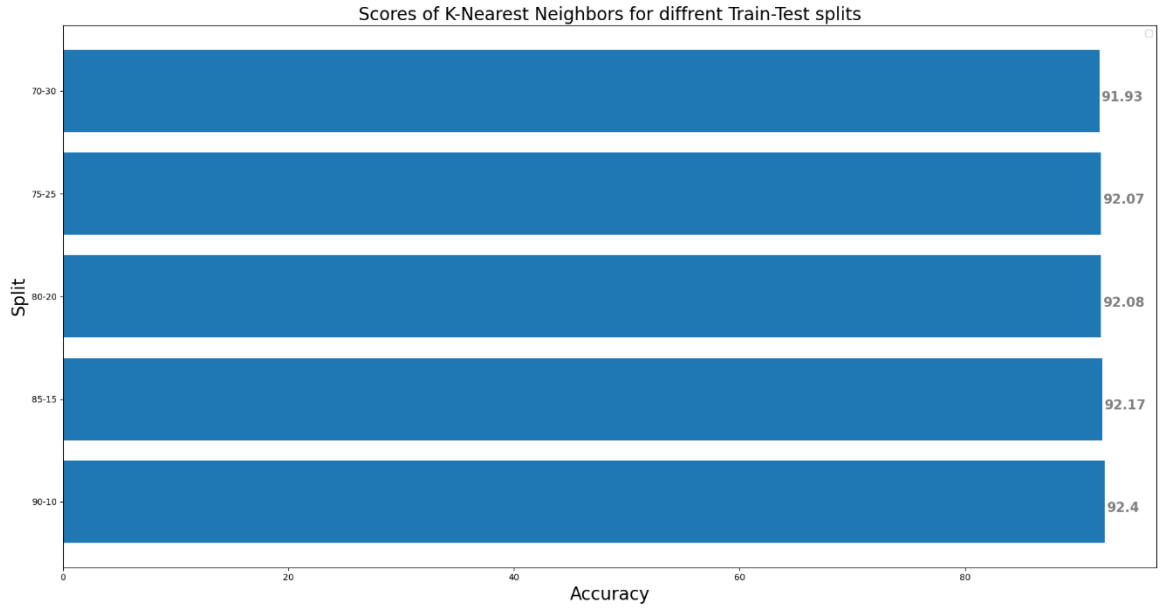
Από την Εικόνα 4. 1 παρατηρείται ότι χωρίς προεπεξεργασία των δεδομένων, παραμετροποίησης και χρησιμοποιώντας όλα τα χαρακτηριστικά, καλύτερη ακρίβεια έχει η γραμμική διακριτική ανάλυση με ποσοστό 94,41%. Ο Πίνακας 4. 1 και Πίνακας 4. 2 περιέχουν τον μέσος όρος, την διασπορά, την τυπική απόκλιση και τον συντελεστής διακύμανσης της ακρίβειας και του λογαριθμικού σφάλματος αντίστοιχα, από την επαναληπτική διαδικασία που πραγματοποιήθηκε. Στη συνέχεια φαίνεται το διάγραμμα κάθε ταξινομητή για όλες τις περιπτώσεις διαχωρισμού.



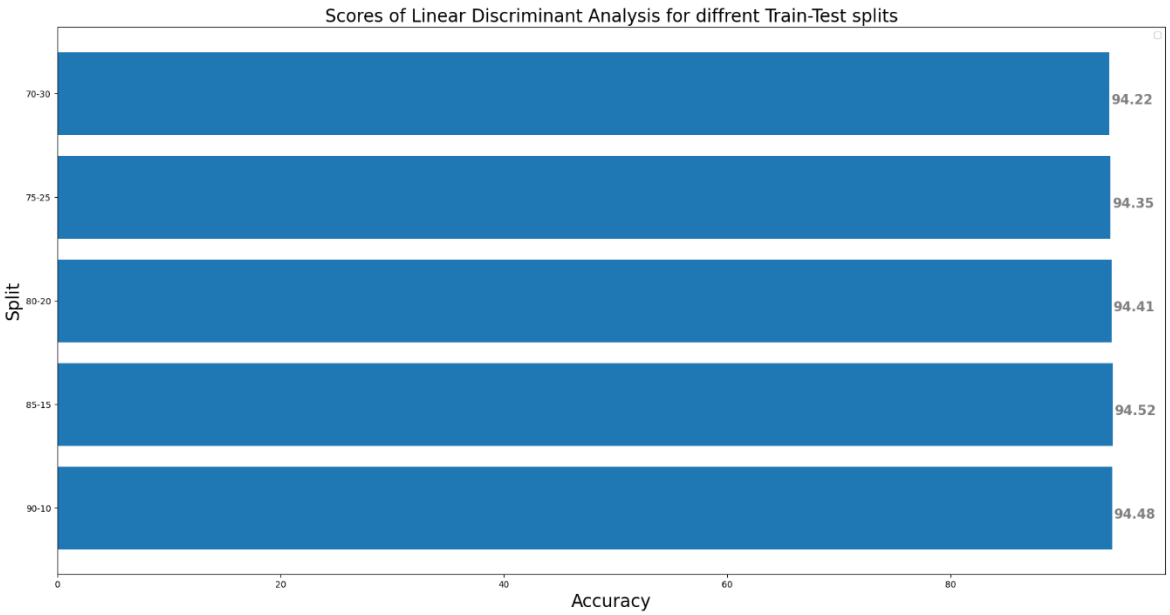
Εικόνα 4. 2: Διάγραμμα ακρίβειας του δέντρου απόφασης για διάφορα δεδομένα προσαρμογής – ελέγχου.



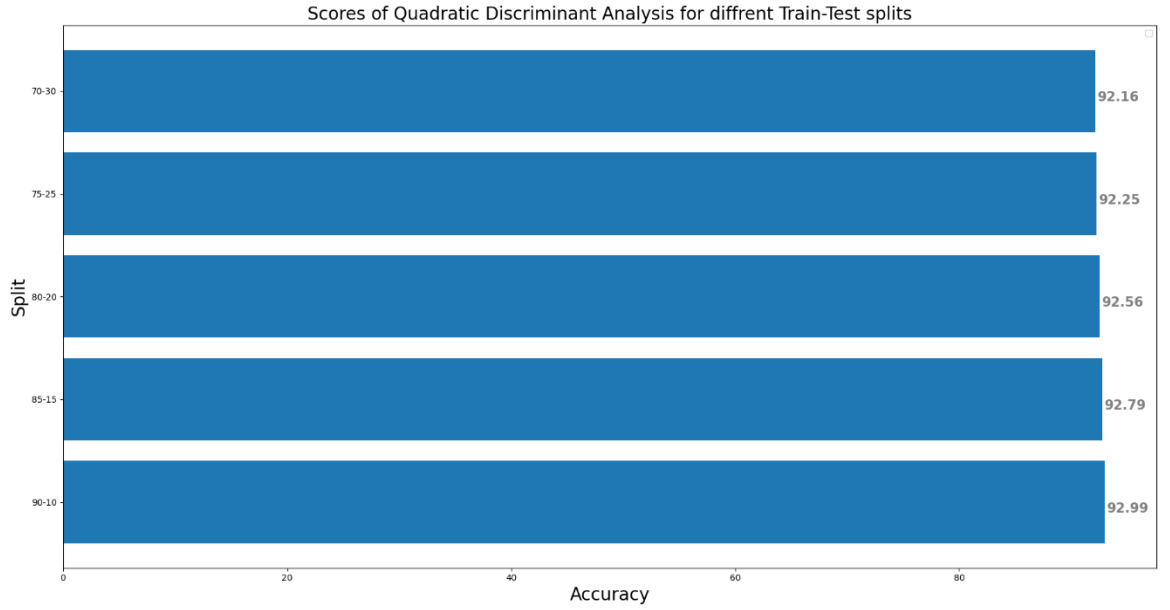
Εικόνα 4. 3: Διάγραμμα ακρίβειας του τυχαίου δάσους για διάφορα δεδομένα προσαρμογής – ελέγχου.



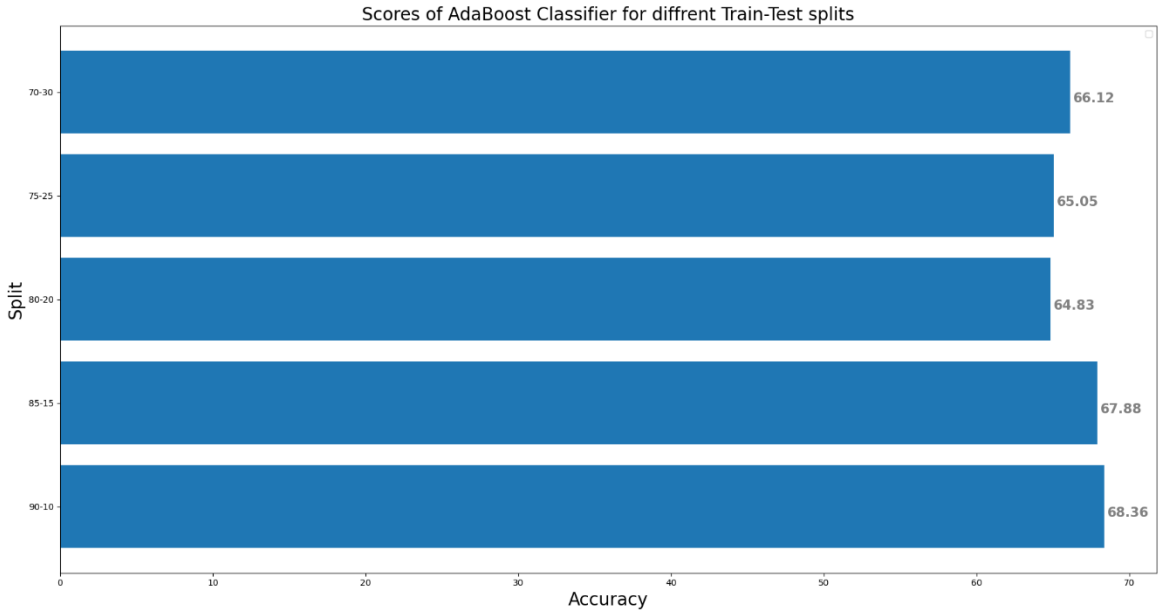
Εικόνα 4. 4: Διάγραμμα ακρίβειας των κ πλησιέστερων γειτόνων για διάφορα δεδομένα προσαρμογής – ελέγχου.



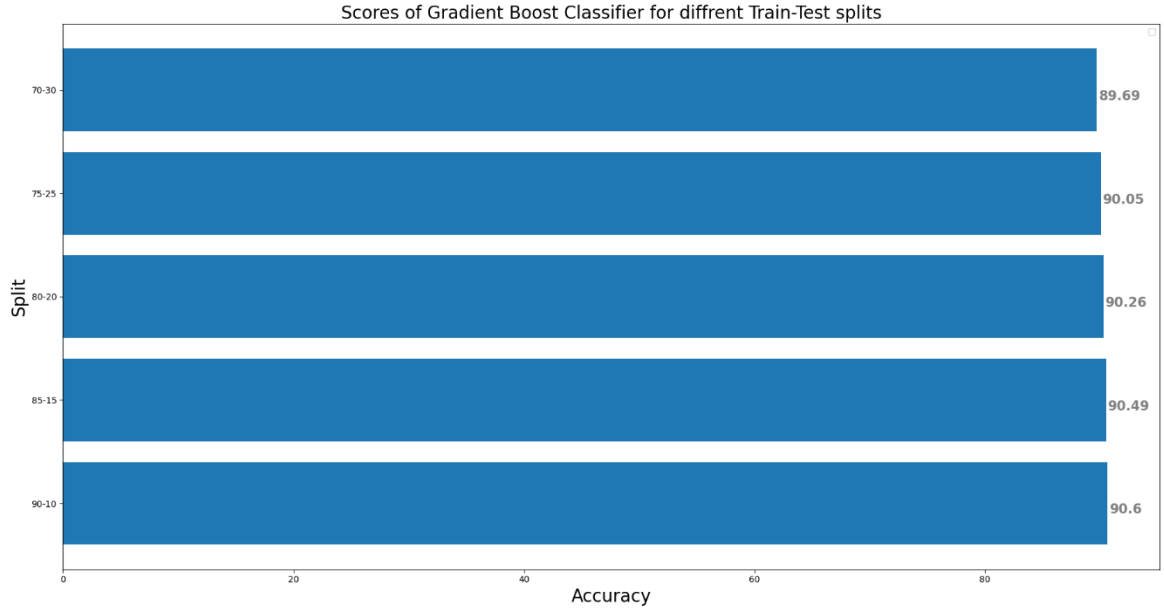
Εικόνα 4. 5: Διάγραμμα ακρίβειας της γραμμικής διακριτικής ανάλυσης για διάφορα δεδομένα προσαρμογής – ελέγχου.



Εικόνα 4. 6: Διάγραμμα ακρίβειας της τετραγωνικής διακριτικής ανάλυσης για διάφορα δεδομένα προσαρμογής – ελέγχου.



Εικόνα 4. 7: Διάγραμμα ακρίβειας του προσαρμοστικού ταξινομητή ενίσχυσης για διάφορα δεδομένα προσαρμογής – ελέγχου.



Εικόνα 4. 8: Διάγραμμα ακρίβειας του βαθμοτού ταξινομητή ενίσχυσης για διάφορα δεδομένα προσαρμογής – ελέγχου.

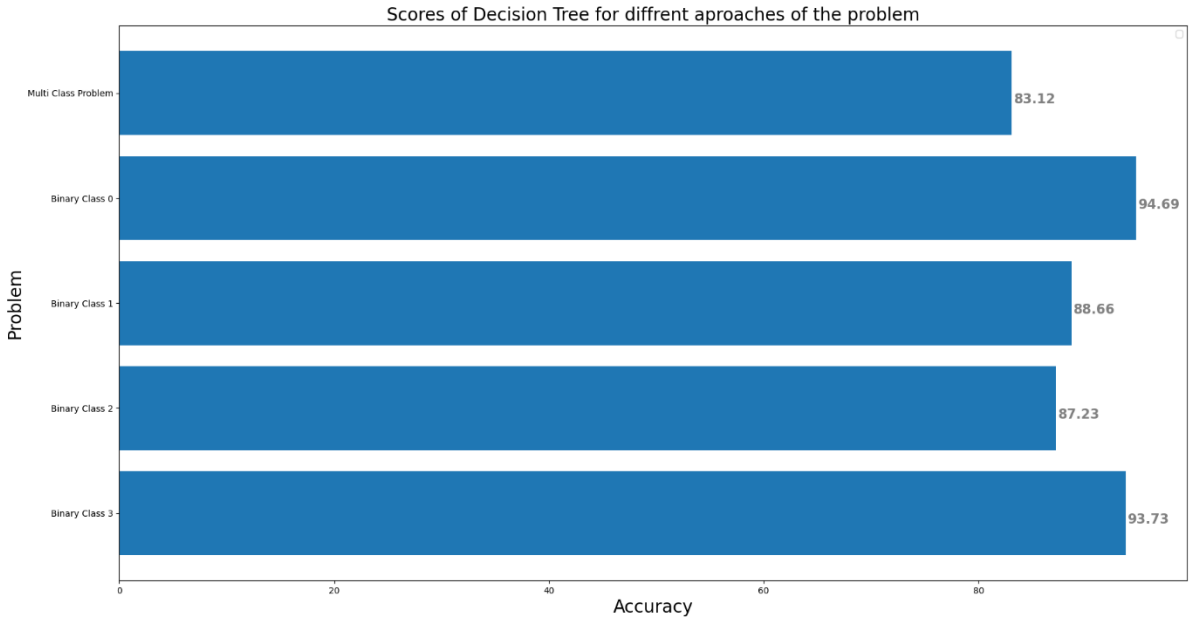
Από τα παραπάνω διαγράμματα είναι ξεκάθαρο ότι όσο αυξάνεται το ποσοστό προσαρμογής των δεδομένων υπάρχουν μικρές μεταβολές αύξησης της ακρίβειας. Καθώς το ποσοστό του ελέγχου μικραίνει κάθε τεχνική έχει λιγότερα δεδομένα να εφαρμοστεί, άρα μία λάθος ταξινόμηση επιφέρει μεγαλύτερη αλλαγή στη ακρίβεια.

Οι καλύτεροι συνδυασμοί για τους αντίστοιχους ταξινομητές είναι:

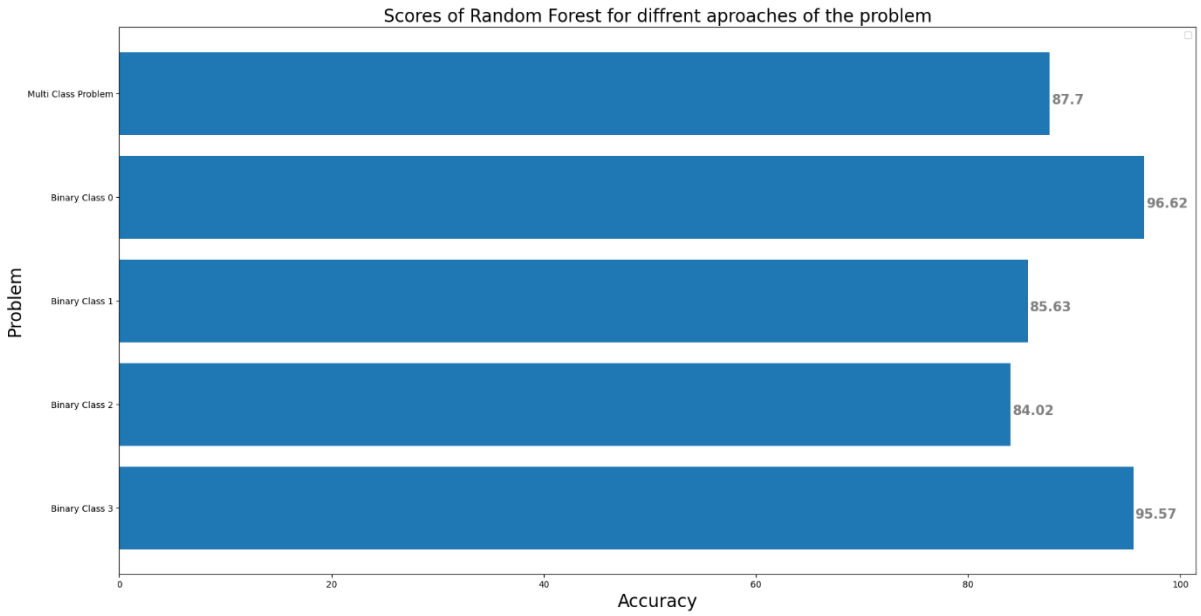
- Δέντρο απόφασης 83,28 % με διαχωρισμό 85 – 15.
- Τυχαίο δάσος 87,98 % με διαχωρισμό 90 – 10.
- Κ πλησιέστεροι γείτονες 92,4 % με διαχωρισμό 90 – 10.
- Γραμμική διακριτική ανάλυση 94,52 % με διαχωρισμό 85 – 15.
- Τετραγωνική διακριτική ανάλυση 92,99 % με διαχωρισμό 90 – 10.
- Προσαρμοστικός ταξινομητής ενίσχυσης 68,36 % με διαχωρισμό 90 – 10.
- Βαθμοτός ταξινομητής ενίσχυσης 90,6 % με διαχωρισμό 90 – 10.

4.2 Δυαδικό πρόβλημα

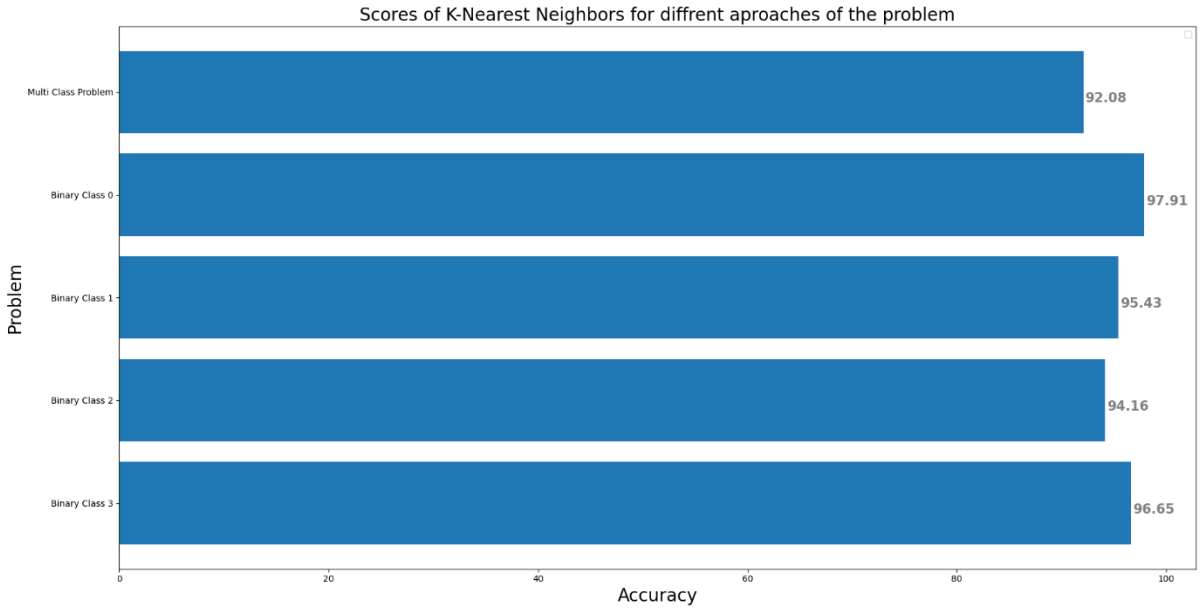
Στη συνέχεια εφαρμόστηκε η μέθοδος μείωσης των κλάσεων και υπολογίστηκαν η ακρίβεια και το λογαριθμικό σφάλμα των τεχνικών μηχανικής μάθησης. Η διαδικασία έγινε χρησιμοποιώντας όλα τα χαρακτηριστικά, για διαχωρισμό προσαρμογής – ελέγχου (80 – 20) και οι παράμετροι των τεχνικών που χρησιμοποιήθηκαν είναι οι προκαθορισμένες. Για να πραγματοποιηθεί η επίλυση του αντίστοιχου δυαδικού προβλήματος ταξινόμησης έγινε σύμπτυξη των τριών κλάσεων σε μία έναντι της τέταρτης. Αναλυτικότερα παρακάτω παρουσιάζονται τα δεδομένα με σύμπτυξη των κλάσεων 1,2 και 3 έναντι της 0 («Binary class 0»), των κλάσεων 0,2 και 3 έναντι της 1 («Binary class 1»), των κλάσεων 0,1 και 3 έναντι της 2 («Binary class 2»), των κλάσεων 0,1 και 2 έναντι της 3 («Binary class 3»).



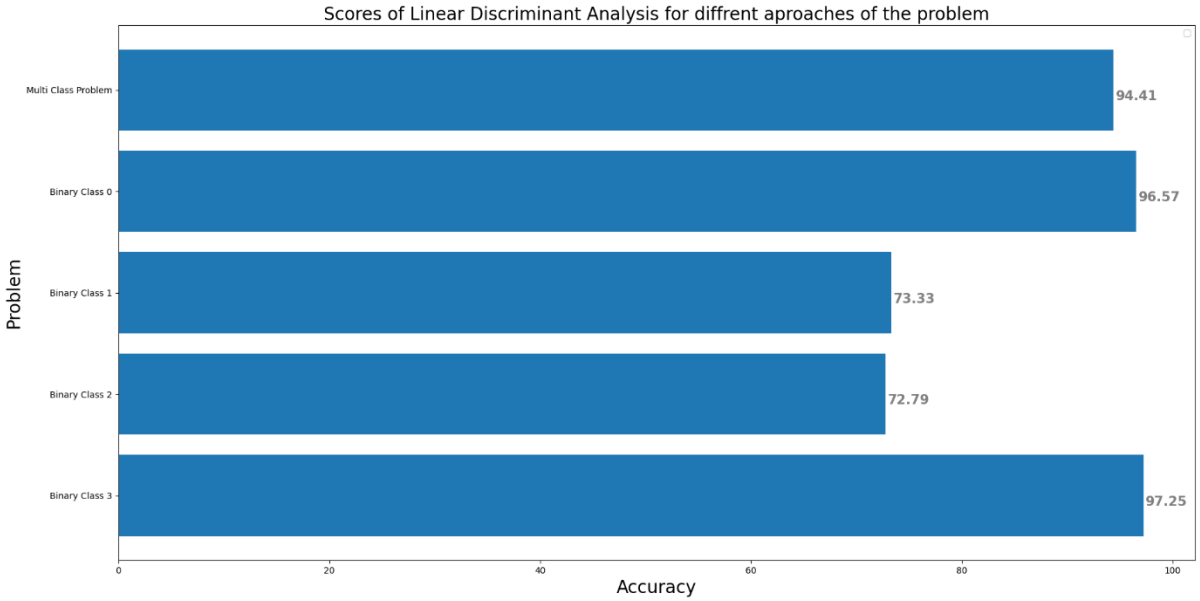
Εικόνα 4. 9: Διάγραμμα ακρίβειας του δέντρου απόφασης για τα προβλήματα δυαδικής ταξινόμησης.



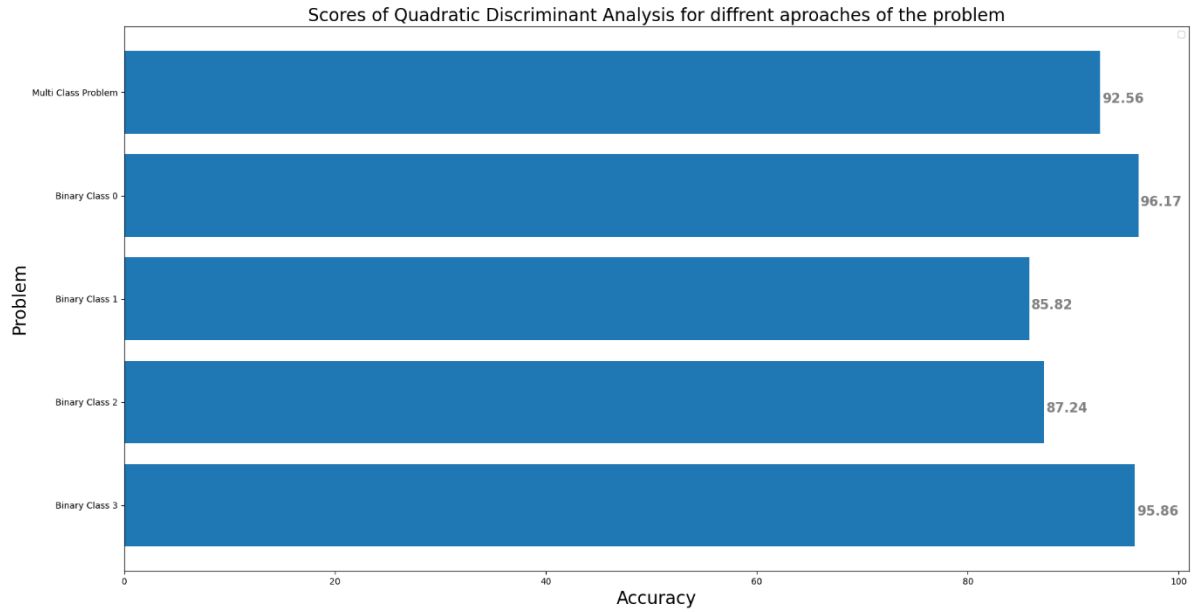
Εικόνα 4. 10: Διάγραμμα ακρίβειας του τυχαίου δάσους για τα προβλήματα δυαδικής ταξινόμησης.



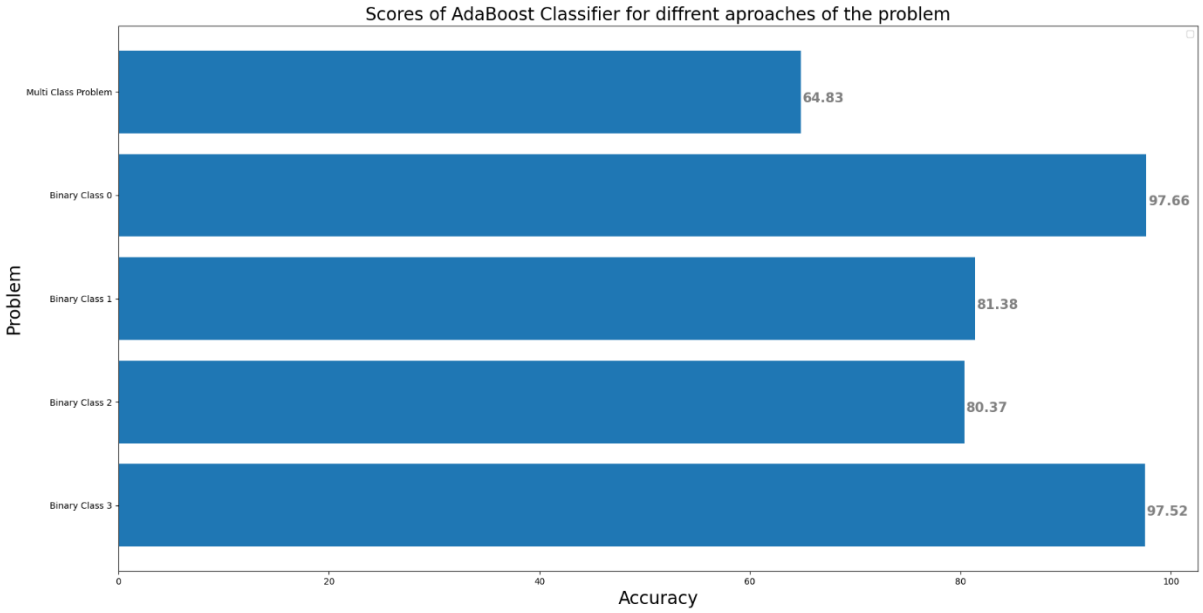
Εικόνα 4. 11: Διάγραμμα ακρίβειας των κ πλησιέστερων γειτόνων για τα προβλήματα δυαδικής ταξινόμησης.



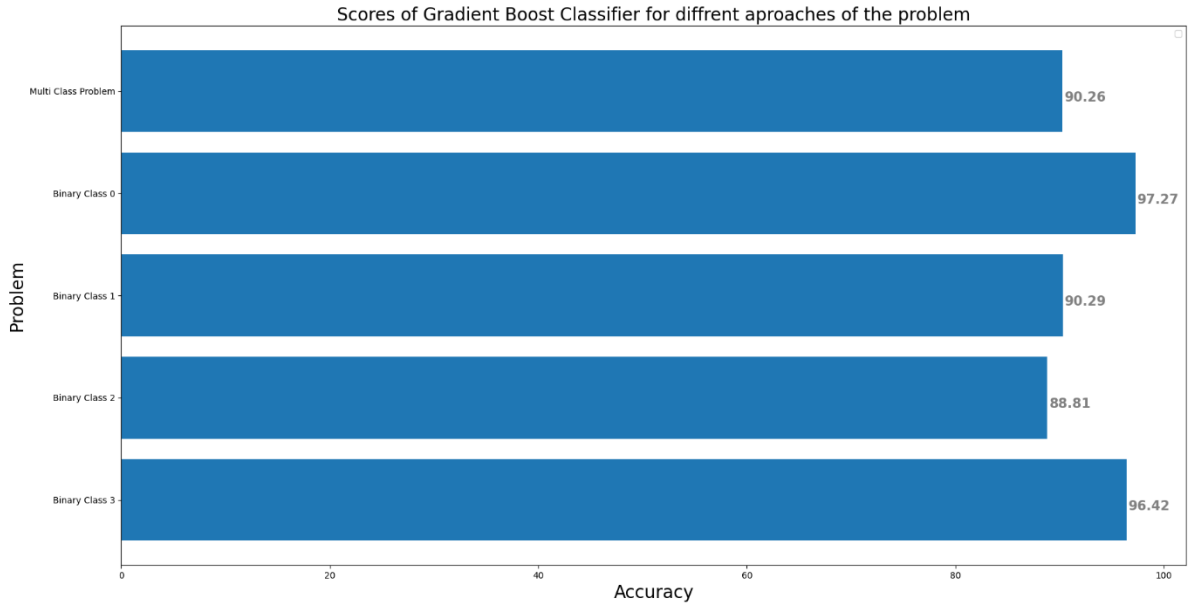
Εικόνα 4. 12: Διάγραμμα ακρίβειας της γραμμικής διακριτικής ανάλυσης για τα προβλήματα δυαδικής ταξινόμησης.



Εικόνα 4. 13: Διάγραμμα ακρίβειας της τετραγωνικής διακριτικής ανάλυσης για τα προβλήματα δυαδικής ταξινόμησης.



Εικόνα 4. 14: Διάγραμμα ακρίβειας του προσαρμοστικού ταξινομητή ενίσχυσης για τα προβλήματα δυαδικής ταξινόμησης.



Εικόνα 4. 15: Διάγραμμα ακρίβειας του βαθμοτού ταξινομητή ενίσχυσης για τα προβλήματα δυαδικής ταξινόμησης.

Από τα παραπάνω διαγράμματα εύκολα εξάγεται το συμπέρασμα ότι οι τεχνικές μηχανικής μάθησης βελτιώνονται όταν ταξινομούν δεδομένα σε δύο κλάσεις και κυρίως η καλύτερη ακρίβεια παρουσιάζεται στις ακραίες τιμές δηλαδή στην πιο χαμηλή και στην πιο υψηλή τιμή. Οι μεσαίες τιμές είναι δυσκολότερο να ταξινομηθούν καθώς η εκπαίδευση γίνεται για δύο κλάσεις και τα καινούρια δεδομένα είναι πιθανό να αντιλαμβάνονται σαν εξαιρέσεις από τις τεχνικές.

Οι καλύτεροι συνδυασμοί για τους αντίστοιχους ταξινομητές είναι:

- Δέντρο απόφασης 94,69 % στο πρόβλημα «Binary class 0»
- Τυχαίο δάσος 96,62 % στο πρόβλημα «Binary class 0».
- Κ πλησιέστεροι γείτονες 97,91 % στο πρόβλημα «Binary class 0».
- Γραμμική διακριτική ανάλυση 97,25 % στο πρόβλημα «Binary class 3».
- Τετραγωνική διακριτική ανάλυση 96,17 % στο πρόβλημα «Binary class 0».
- Προσαρμοστικός ταξινομητής ενίσχυσης 97,66 % στο πρόβλημα «Binary class 0».
- Βαθμοτός ταξινομητής ενίσχυσης 97,27 % στο πρόβλημα «Binary class 0».

Ο καλύτερος συνδυασμός από όλους είναι αυτός των πλησιέστερων γειτόνων καθώς εμφανίζει την μεγαλύτερη ακρίβεια. Μάλιστα, εμφανίζει υψηλότερη ακρίβεια από την αντίστοιχη του προβλήματος με τις τέσσερις κλάσεις σε όλα τα επιμέρους προβλήματα δηλαδή ακόμα και στις μεσαίες τιμές οι οποίες φαίνεται να «μπερδεύουν» τις τεχνικές.

4.3 Παραμετροποίηση

Τέλος, επιλέχθηκαν οι κατάλληλες παράμετροι για τρείς τεχνικές μηχανικής μάθησης, το τυχαίο δάσος, τους κ πλησιέστερους γείτονες και του προσαρμοστικού ταξινομητή ενίσχυσης όπως αναφέρθηκε στο Κεφάλαιο 3.4. Οι τεχνικές υλοποιήθηκαν με διαχωρισμό δεδομένων προσαρμογής – ελέγχου 80 – 20 και χρησιμοποιώντας όλα τα χαρακτηριστικά. Στους παρακάτω πίνακες φαίνονται ο μέσος όρος, η διασπορά, η τυπική απόκλισή και ο συντελεστής διακύμανσής για την ακρίβεια και το λογαριθμικό σφάλμα από την επαναληπτική διαδικασία.

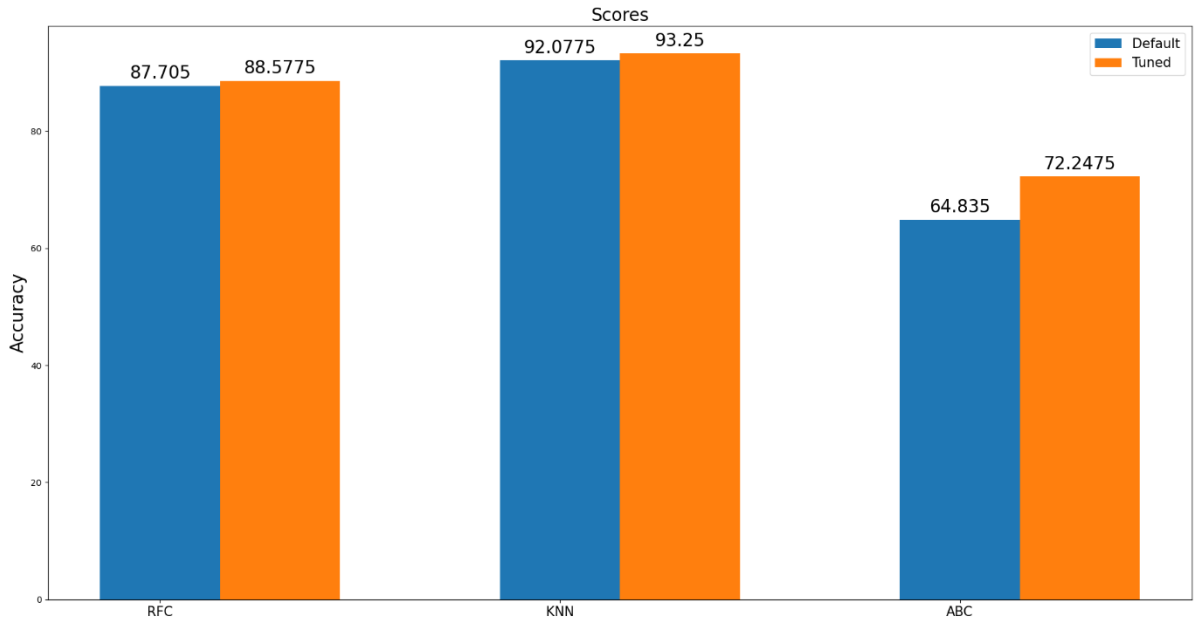
	Random Forest Classifier	K Neighbors Classifier	Ada Boost Classifier
Mean	88.5775	93.25	72.2475
Variance	3.019817	1.463384	6.063756313
Standard Deviation	1.737762	1.209704	2.462469556
Coefficient of Variation	0.019619	0.012973	0.034083803

Πίνακας 4. 3: Πίνακας μέσου όρου, διασποράς, τυπικής απόκλισης και συντελεστή διακύμανσης της ακρίβειας κάθε παραμετροποιημένου ταξινομητή.

	Random Forest Classifier	K Neighbors Classifier	Ada Boost Classifier
Mean	0.516454	0.191214	0.789987
Variance	0.000205	0.000989	0.000619
Standard Deviation	0.014305	0.031453	0.024887
Coefficient of Variation	0.027699	0.164492	0.031503

Πίνακας 4. 4: Πίνακας μέσου όρου, διασποράς, τυπικής απόκλισης και συντελεστή διακύμανσης του λογαριθμικού σφάλματος κάθε παραμετροποιημένου ταξινομητή.

Το παρακάτω διάγραμμα είναι μια συγκριτική γραφική απεικόνιση των τεχνικών χωρίς παραμετροποίηση και των τεχνικών με παραμετροποίηση.



Εικόνα 4. 16: Συγκριτικό διάγραμμα ακρίβειας των τεχνικών που παραμετροποιήθηκαν.

Όπως παρατηρείται από την Εικόνα 4. 16 η αύξηση της ακρίβειας που επιτεύχθηκε στην τεχνική τυχαίο δάσος δεν είναι τόσο σημαντική καθώς είναι λιγότερη από 1 % . Ωστόσο στην τεχνική των κ πλησιέστερων γειτόνων η αύξηση είναι περίπου ίση με 1,17 % άρα φαίνεται ότι κάποια δεδομένα εξαιρέσεις κατάφερε να τα ταξινομήσει ορθά. Ο προσαρμοστικός ταξινομητής ενίσχυσης φαίνεται γενικότερα να «δυσκολεύεται» αρκετά στην ταξινόμηση του προβλήματος των τεσσάρων κλάσεων ωστόσο με την παραμετροποίηση σημείωσε εξέλιξη της τάξης 7,41%, που σημαίνει ότι δεν προσάρμοζε καλά τα δεδομένα.

4.4 Συζήτηση αποτελεσμάτων

Στην παρούσα διπλωματική εξετάστηκε ο τρόπος που αντιμετωπίζει κάθε τεχνική μηχανικής μάθησης τα δεδομένα και πόσο ακριβείς είναι τα αποτελέσματά τους. Συνολικά εξετάστηκαν τρείς περιπτώσεις.

1) Διαφορετικοί διαχωρισμοί των δεδομένων προσαρμογής – ελέγχου.
2) Προσέγγιση των επιμέρους δυαδικών προβλημάτων
3) Παραμετροποίηση τριών σημαντικών τεχνικών μηχανικής μάθησης.

Στην περίπτωση του διαχωρισμού των δεδομένων προσαρμογής – ελέγχου, οι προβλέψεις γίνονταν καλύτερες όσο αυξανόταν το ποσοστό προσαρμογής και μειωνόταν το ποσοστό ελέγχου. Η καλύτερη αναλογία βρίσκεται μεταξύ του 85 – 15 και του 90 – 10. Αν επιλεχθεί υπερβολικό ποσοστό δεδομένων προσαρμογής, είναι πιθανή η τρομακτική μείωση της ακρίβειας γιατί θα υπάρχει μεγαλύτερη πιθανότητα υπερπροσαρμογής των δεδομένων και μικρότερο σύνολο ελέγχου το οποίο δεν θα είναι ενδεικτικό για τα αποτελέσματα.

Στην περίπτωση της επίλυσης των επιμέρους δυαδικών προβλημάτων φαίνεται να υπάρχει μεγάλη αύξηση στην ακρίβεια πράγμα που δείχνει ότι αξίζει να προσεγγίσει κανείς ένα πρόβλημα ταξινόμησης πολλών κλάσεων σαν πολλά προβλήματα ταξινόμησης δύο κλάσεων. Η τεχνική μάλιστα που αξίζει να εφαρμοστεί αυτή η μεθοδολογία είναι οι κ πλησιέστεροι γείτονες καθώς σημείωναν αύξηση της ακρίβειας ακόμα και στις ενδιάμεσες τιμές που άλλοι ταξινομητές «μπερδεύονταν».

Στην περίπτωση της παραμετροποίησης των τριών τεχνικών μηχανικής μάθησης παρατηρείται ότι η βελτιστοποίηση που επιτεύχθει δεν είναι τόσο σημαντική ωστόσο βοηθάει στην διαλεύκανση κάποιων δεδομένων που αποτελούν εξαίρεση. Η παραμετροποίηση του προσαρμοστικού ταξινομητή ενίσχυσης φαίνεται να ήταν απαραίτητη για την προσαρμογή του στο πρόβλημα αλλά και πάλι δεν ήταν τόσο ανταγωνιστικός όσο οι άλλες τεχνικές.

ΚΕΦΑΛΑΙΟ 5. ΣΥΜΠΕΡΑΣΜΑΤΑ

5.1 Συμπεράσματα

Οι επιπτώσεις που υπάρχουν με την ανάπτυξη διαφόρων συστημάτων και εργαλείων μηχανικής μάθησης σε συνδυασμό με την εξόρυξη δεδομένων από διάφορους τομείς μπορούν να επιλύσουν διάφορα προβλήματα. Ένας από τους τομείς είναι ο τομέας της βιομηχανίας. Για τρεις περιπτώσεις εξετάστηκε κάθε μοντέλο μηχανικής μάθησης.

Το σύνολο δεδομένων που χρησιμοποιήθηκε όπως προαναφέρθηκε είναι το «Mobile Price Classification». Για την προσαρμογή χρησιμοποιήθηκαν όλα τα χαρακτηριστικά χωρίς προεπεξεργασία καθώς το σετ δεδομένων ήταν εξ αρχής ισορροπημένο και τα χαρακτηριστικά δεν ήταν τόσο θορυβώδη.

Αρχικά μελετήθηκε η συμπεριφορά των αλγορίθμων σε διάφορα ποσοστά δεδομένων προσαρμογής και ελέγχου. Παρατηρήθηκε ότι ο καλύτερος διαχωρισμός βρίσκεται ανάμεσα στις αναλογίες 85 – 15 και 90 – 10 καθώς οι τιμές της ακρίβειας ήταν οι μεγαλύτερες. Ωστόσο η μεταβολή τις ακρίβειας δεν είναι τόσο σημαντική για να σημειωθεί εκτός από τον αλγόριθμο «Ada Boost Classifier». Στην περίπτωση αυτού του αλγόριθμου οι τιμές ακρίβειας ακόμα και σε αναλογία διαχωρισμού 90 – 10 δεν είναι καλές και δεν μπορούν να χρησιμοποιηθούν σε προβλήματα της πραγματικής ζωής καθώς η πιθανότητα σωστής ταξινόμησης είναι πολύ κοντά στην πιθανότητα ρίψης ενός νομίσματος. Αυτό οφείλεται στην μεθοδολογία του αλγόριθμου καθώς είναι σχεδιασμένος για προβλήματα δυαδικής ταξινόμησης και αποδεικνύεται στο δεύτερο σκέλος της παρούσας διπλωματικής εργασίας.

Στη συνέχει εκτελέστηκε η μεθοδολογία μείωσης των κλάσεων. Παρουσιάστηκε ένας διαφορετικός τρόπος προσέγγισης του προβλήματος πολλών κλάσεων και πιο συγκεκριμένα έγινε η δημιουργία τεσσάρων διαφορετικών προβλημάτων με δύο κλάσεις. Σε αυτό το τμήμα της διπλωματικής εργασίας παρατηρήθηκε σημαντική αύξηση στην ακρίβεια όλων των τεχνικών και ιδιαίτερα του «Ada Boost Classifier» που ανταγωνίζεται την τεχνική των κ πλησιέστερων γειτόνων στις ακραίες τιμές.

Στην τελευταία μελέτη που διεξάχθηκε, χρησιμοποιήθηκαν τρεις τεχνικές μηχανικής μάθησης οι οποίες παραμετροποιήθηκαν με στόχο την αύξηση της ακρίβειας και μείωση του λογαριθμικού σφάλματος. Ωστόσο στις τεχνικές τυχαίο δάσος και κ πλησιέστεροι γείτονες παρατηρήθηκε μικρή αύξηση του ποσοστού ακρίβειας, που όμως ήταν είδη αρκετά υψηλό. Ο προσαρμοστικός ταξινομητής ενίσχυσης σημείωσε σημαντική αύξηση της ακρίβειάς του σε βαθμό που πλέον μπορεί να χαρακτηριστεί «καλός». Φαίνεται ότι με τις προκαθορισμένες παραμέτρους για το συγκεκριμένο σύνολο δεδομένων, συνυπολογίζοντας ότι το πρόβλημα είναι ταξινόμησης πολλών κλάσεων, έκανε κακή προσαρμογή των δεδομένων.

Με τα αποτελέσματα που παρουσιάστηκαν στην παρούσα διπλωματική εργασία γίνεται εύκολα αντιληπτό ότι, ο ορθός διαχωρισμός των δεδομένων, οι τυχόν διαφορετικές προσεγγίσεις του

προβλήματος και η παραμετροποίηση των τεχνικών συμβάλουν σημαντικά στην βελτίωση των τεχνικών μηχανικής μάθησης.

5.2 Μελλοντικές Κατευθύνσεις

Η παρούσα διπλωματική εργασία είχε σκοπό την σύγκριση των ταξινομητών για την πρόβλεψη του εύρους τιμής ενός κινητού τηλεφώνου σύμφωνα με τα χαρακτηριστικά του.

Τέλος, η μελέτη περισσότερων τεχνικών μηχανικής μάθησης καθώς και η διαδικασία προεπεξεργασία των δεδομένων θα μπορούσε να βελτιώσει ακόμα περισσότερο την ακρίβεια της πρόβλεψης του εύρους τιμής. Επίσης θα μπορούσε να μελετηθεί βαθύτερα η προσέγγιση των επιμέρους δυαδικών προβλημάτων σε ένα πρόβλημα πολλών κλάσεων. Εύκολα μπορεί να διεξαχθεί το συμπέρασμα ότι, η μηχανική μάθηση έχει πάρα πολλές εφαρμογές και θα αποτελέσει, θέμα συζητήσεις από πολλούς επιστήμονες για τα χρόνια που ακολουθούν.

ΚΕΦΑΛΑΙΟ 6. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] «Insight.Samsung,» Samsung, 22 Apr 2022. [Ηλεκτρονικό]. Available: https://insights.samsung.com/2022/04/22/how-much-should-your-business-spend-on-mobile/.

[2] «CSC,» 18 Nov 2018. [Ηλεκτρονικό]. Available: https://www.csc.com.gr/machine-learning-%CE%BC%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE-%CE%BC%CE%AC%CE%B8%CE%B7%CF%83%CE%B7-%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9/.

[3] «GeeksforGeeks,» 13 Sep 2021. [Ηλεκτρονικό]. Available: https://www.geeksforgeeks.org/ml-types-learning-supervised-learning/.

[4] E. B. G. V. Margherita Grandini, «METRICS FOR MULTI-CLASS CLASSIFICATION: AN OVERVIEW,» pp. 2-3, 13 Aug 2020.

[5] «Google developers,» Google, Feb 2020. [Ηλεκτρονικό]. Available: https://developers.google.com/machine-learning/crash-course/classification/accuracy.

[6] Tashuhka, «Stack Exchange,» Cross Validated, 26 Aug 2014. [Ηλεκτρονικό]. Available: https://stats.stackexchange.com/q/113301.

[7] N. Nerd, «Decision Tree Classification Clearly Explained!,» 13 Jan 2021. [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=ZVR2Way4nwQ.

[8] N. Normalized, «Random Forest Algorithm Clearly Explained!,» 21 Apr 2021. [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=v6VJ2RO66Ag.

[9] O. Kramer, «K-Nearest Neighbors,» σε *Dimensionality Reduction with Unsupervised Nearest Neighbors* , Oldenburg, Springer Berlin, Heidelberg, 2013, pp. 13-14.

[10] A. Kumar, «Vitalflux,» 9 September 2020. [Ηλεκτρονικό]. Available: https://vitalflux.com/adaboost-algorithm-explained-with-python-example/.

[11] J. H. Friedman, «Greedy Function Approximation: A Gradient Boosting Machine,» *The Annals of Statistics*, τόμ. 29, αρ. 5, pp. 1189-1232, 1999.

[12] S. Balakrishnama και A. Ganapathiraju, «LINEAR DISCRIMINANT ANALYSIS - A BRIEF TUTORIAL,» Mississippi, 1998.

[13] «Quadratic Discriminant Analysis,» 7 january 2022. [Ηλεκτρονικό]. Available: https://www.geeksforgeeks.org/quadratic-discriminant-analysis/.

[14] «1.2. Linear and Quadratic Discriminant Analysis,» [Ηλεκτρονικό]. Available: https://scikit-learn.org/stable/modules/lda_qda.html?highlight=linear%20quadratic%20discriminant%20analysis#linear-and-quadratic-discriminant-analysis.

[15] «About pandas,» [Ηλεκτρονικό]. Available: https://pandas.pydata.org/about/.

[16] F. P. a. G. V. a. A. G. a. V. M. a. B. Thirion, «Scikit-learn: Machine Learning in Python,» *Journal of Machine Learning Research*, τόμ. 12, pp. 2825-2830, 2011.

[17] Google LLC, April 2010. [Ηλεκτρονικό]. Available: https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification.

[18] «sklearn.ensemble.RandomForestClassifier,» 2007-2022. [Ηλεκτρονικό]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random%20forest#sklearn.ensemble.RandomForestClassifier.

[19] T. Srivastava, «Tuning the parameters of your Random Forest model,» 9 june 2015. [Ηλεκτρονικό]. Available: https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/.

[20] «scikit-learn,» 2007-2022. [Ηλεκτρονικό]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html?highlight=adaboost%20classifier#sklearn.ensemble.AdaBoostClassifier.

[21] sklearn.model_selection.train_test_split, «Scikit-Learn,» 2007-2022. [Ηλεκτρονικό]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[22] M. Galarnyk, «Understanding Train Test Split (Scikit-Learn + Python),» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/understanding-train-test-split-scikit-learn-python-ea676d5e3d1.

[23] H. Fleyeh και E. Davami, «Multiclass Adaboost Based on an Ensemble of Binary Adaboosts.,» American Journal of Intelligent Systems, 2013.

ΠΑΡΑΡΤΗΜΑ

Λογισμικό: Τα χαρακτηριστικά του υπολογιστή που χρησιμοποιήθηκε είναι τα εξής:

- Επεξεργαστής: AMD Ryzen 5 3600 6 – core Processor 3.60 GHz
- Εγκατεστημένη μνήμη: 16 GB
- Λογισμικό: Windows 10 Professional 64 – bit

Η γλώσσα προγραμματισμού είναι η Python και ο αλγόριθμος εκτελέστηκε σε text editor Visual Studio Code.

```
#                                   |===============|
#                                   |____IMPORTS____|
#                                   |===============|
import time
start_time = time.time()


import pandas as pd
import numpy as np
import csv
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis ,
QuadraticDiscriminantAnalysis
from sklearn.ensemble import GradientBoostingClassifier , AdaBoostClassifier
from sklearn.metrics import accuracy_score, log_loss
from statistics import variance          #Variance, Standard Deviation etc.
from math import sqrt
#          ===_PLOTS_===
import seaborn as sns
```

```
import matplotlib.pyplot as plt

#                                       |=====================|
#                                       |___PATH & DATA READ___|
#                                       |=====================|

test_path = (r'C:\Users\Mihalis\Desktop\py course\my thesis\mobile
database\archive(4)\test.csv')
train_path = (r'C:\...\train.csv')

test_data = pd.read_csv(test_path)
train_data = pd.read_csv(train_path)

#                                      |======================|
#                                      |___FEATURES & TARGET___|
#                                       |======================|
#                 ---_TARGET_---
#TRAIN VALS
y_train=train_data['price_range']
x_train=train_data.drop(['price_range'],axis=1)

#TEST VALS
y_test=pd.DataFrame(columns=['price_range'])
x_test=test_data.drop(['id'],axis=1)

#                                        |===============|
#                                        |__MY MODELS___|
#                                        |===============|
#                 ---_DECISION TREE CLASSIFIER_---
def dec_tree_class(split_x_train,split_y_train,split_x_val,split_y_val):
```

```
    model=DecisionTreeClassifier(random_state=1)
    model.fit(split_x_train,split_y_train)
    y_preds=model.predict(split_x_val)
    acc=accuracy_score(split_y_val,y_preds,normalize=True,sample_weight=None)
    train_prediction = model.predict_proba(split_x_val)
    ll=log_loss(split_y_val,train_prediction)
    return acc,ll

#                   ---_RANDOM FOREST CLASSIFIER_---
def ran_forest_class(split_x_train,split_y_train,split_x_val,split_y_val):
    model=RandomForestClassifier(random_state=1)
    model.fit(split_x_train,split_y_train)
    y_preds=model.predict(split_x_val)
    acc=accuracy_score(split_y_val,y_preds,normalize=True,sample_weight=None)
    train_prediction = model.predict_proba(split_x_val)
    ll=log_loss(split_y_val,train_prediction)
    return acc,ll

#                   ---_K NEIGHBORS PROCESS CLASSIFICATION_---
def k_neigh_class(split_x_train,split_y_train,split_x_val,split_y_val):
    model = KNeighborsClassifier()
    model.fit(split_x_train,split_y_train)
    y_preds = model.predict(split_x_val)
    acc=accuracy_score(split_y_val,y_preds,normalize=True,sample_weight=None)
    train_prediction = model.predict_proba(split_x_val)
    ll=log_loss(split_y_val,train_prediction)
    return acc,ll

#                   ---_Linear Discriminant Analysis_---
```

```
def linear(split_x_train,split_y_train,split_x_val,split_y_val):
    model=LinearDiscriminantAnalysis()
    model.fit(split_x_train,split_y_train)
    y_preds=model.predict(split_x_val)
    acc=accuracy_score(split_y_val,y_preds,normalize=True,sample_weight=None)
    train_prediction = model.predict_proba(split_x_val)
    ll=log_loss(split_y_val,train_prediction)
    return acc,ll

#                   ---_Quadratic Discriminant Analysis_---
def quadratic(split_x_train,split_y_train,split_x_val,split_y_val):
    model=QuadraticDiscriminantAnalysis()
    model.fit(split_x_train,split_y_train)
    y_preds=model.predict(split_x_val)
    acc=accuracy_score(split_y_val,y_preds,normalize=True,sample_weight=None)
    train_prediction = model.predict_proba(split_x_val)
    ll=log_loss(split_y_val,train_prediction)
    return acc,ll

#                   ---_Ada Boost Classifier_---
def ada(split_x_train,split_y_train,split_x_val,split_y_val):
    model=AdaBoostClassifier(random_state=1)
    model.fit(split_x_train,split_y_train)
    y_preds=model.predict(split_x_val)
    acc=accuracy_score(split_y_val,y_preds,normalize=True,sample_weight=None)
    train_prediction = model.predict_proba(split_x_val)
    ll=log_loss(split_y_val,train_prediction)
    return acc,ll
```

```
#                    ---_Gradient Boosting Classifier_---
def gradient(split_x_train,split_y_train,split_x_val,split_y_val):
    model=GradientBoostingClassifier(random_state=1)
    model.fit(split_x_train,split_y_train)
    y_preds=model.predict(split_x_val)
    acc=accuracy_score(split_y_val,y_preds,normalize=True,sample_weight=None)
    train_prediction = model.predict_proba(split_x_val)
    ll=log_loss(split_y_val,train_prediction)
    return acc,ll


#                                |================|
#                                |___MAIN CODING___|
#                                |================|


accuracy_path = (r'C:\Users\Mihalis\Desktop\py course\my thesis\mobile
database\archive(4)\train_test_split_80-20\accuracy.csv')   #EXPORTED CSV PATHS
loss_path = (r'C:\Users\Mihalis\Desktop\py course\my thesis\mobile
database\archive(4)\train_test_split_80-20\loss.csv')
final_acc_vals_path = (r'C:\Users\Mihalis\Desktop\py course\my thesis\mobile
database\archive(4)\train_test_split_80-20\final_acc_values.csv')
final_loss_vals_path = (r'C:\Users\Mihalis\Desktop\py course\my thesis\mobile
database\archive(4)\train_test_split_80-20\final_loss_values.csv')



accuracy_csv = open(accuracy_path, 'w', newline='')        # open the file in the write mode
loss_csv = open(loss_path, 'w', newline='')
final_acc_vals_csv = open(final_acc_vals_path, 'w', newline='')
final_loss_vals_csv = open(final_loss_vals_path, 'w', newline='')
```

```
writer_acc = csv.writer(accuracy_csv)                    # create the csv writer
writer_loss = csv.writer(loss_csv)
writer_final_acc = csv.writer(final_acc_vals_csv)
writer_final_loss = csv.writer(final_loss_vals_csv)



name = ['Decision Tree Classifier',
   'Random Forest Classifier',
   'K Neighbors Classifier',
   'Linear Discriminant Analysis',
   'Quadratic Discriminant Analysis',
   'Ada Boost Classifier',
   'Gradient Boosting Classifier']

modified_name=['']                              # Name variable for final vals csv

for classifiers_count in range(0,7):                # Loop for applying final vals variable
   modified_name.append(name[classifiers_count])      # into the final csv

writer_acc.writerow(name)                        # write the title of each column
writer_loss.writerow(name)                       # to the csv file
writer_final_acc.writerow(modified_name)
writer_final_loss.writerow(modified_name)



for seed in range(0,100):              #LOOP FOR EXPORTING ACCURACY AND LOG
                                        #LOSS DATA FROM 100 TIMES ITTERATION
```

```
   #SPLIT TRAIN VALS FOR MODEL ACCURACY AND LOG LOSS MESSURMENT

split_x_train,split_x_val,split_y_train,split_y_val=train_test_split(x_train,y_train,train_size=0
.80,test_size=0.20,random_state=seed)


   classifier_acc_loss=[
      dec_tree_class(split_x_train,split_y_train,split_x_val,split_y_val),
      ran_forest_class(split_x_train,split_y_train,split_x_val,split_y_val),
      k_neigh_class(split_x_train,split_y_train,split_x_val,split_y_val),
      linear(split_x_train,split_y_train,split_x_val,split_y_val),
      quadratic(split_x_train,split_y_train,split_x_val,split_y_val),
      ada(split_x_train,split_y_train,split_x_val,split_y_val),
      gradient(split_x_train,split_y_train,split_x_val,split_y_val)]


   acc = list()
   loss = list()



   for i in classifier_acc_loss:     # Loop for saving each value of the "randon_state"
      acc.append((i[0])*100)        # itteration in a variable
      loss.append((i[1]))


   writer_acc.writerow(acc)      # Saves the accuracy and loss
   writer_loss.writerow(loss)    # values to the csv for each model



accuracy_csv.close() # close the files
loss_csv.close()
```

```
                    # MAIN LOOP FOR SAVING AND EXPORTING MEAN,
VARIANCE, STANDARD DEVIATION etc.
acc_read_pd = pd.read_csv(accuracy_path)
loss_read_pd = pd.read_csv(loss_path)


mean_list = ['Mean']            #ACCURACY
variance_list = ['Variance']
standard_deviation_list = ['Standard Deviation']
coefficient_variation_list = ['Coefficient of Variation']

for title in name:              #Loop for accuracy
   mean_list.append(acc_read_pd[title].mean())
   variance_list.append(variance(acc_read_pd[title]))
   standard_deviation_list.append(sqrt(variance(acc_read_pd[title])))

coefficient_variation_list.append((sqrt(variance(acc_read_pd[title]))/(acc_read_pd[title].mean
())))

writer_final_acc.writerow(mean_list)
writer_final_acc.writerow(variance_list)
writer_final_acc.writerow(standard_deviation_list)
writer_final_acc.writerow(coefficient_variation_list)

mean_list = ['Mean']            #LOSS
variance_list = ['Variance']
standard_deviation_list = ['Standard Deviation']
coefficient_variation_list = ['Coefficient of Variation']

for title in name:              #Loop for log loss
```

```
    mean_list.append(loss_read_pd[title].mean())
    variance_list.append(variance(loss_read_pd[title]))
    standard_deviation_list.append(sqrt(variance(loss_read_pd[title])))

coefficient_variation_list.append((sqrt(variance(loss_read_pd[title]))/(loss_read_pd[title].mean())))

writer_final_loss.writerow(mean_list)
writer_final_loss.writerow(variance_list)
writer_final_loss.writerow(standard_deviation_list)
writer_final_loss.writerow(coefficient_variation_list)

mean_list = ['Mean']
variance_list = ['Variance']
standard_deviation_list = ['Standard Deviation']
coefficient_variation_list = ['Coefficient of Variation']

final_acc_vals_csv.close()    # close the files
final_loss_vals_csv.close()

print("--- %s seconds ---" % (time.time() - start_time))
```

output: -- 191.82808017730713 seconds ---

Για την εξαγωγή των αποτελεσμάτων για train – test split 70 – 30, 75 – 25, 80 – 20, 85 – 15, 90 – 10 κάθε φορά που έτρεχε ο αλγόριθμος αλλάχθηκαν τα paths των csv αρχείων και οι παράμετροι train_size και test_size του train_test_split().

Για την επίλυση των αντίστοιχων διαδίκων προβλημάτων στο τμήμα του αλγόριθμου «Features and target» έγινε εισαγωγή των παρακάτω σειρών αντίστοιχα για τα προβλήματα Binary class 0, Binary class 1, Binary class 2, Binary class 3.

Binary class 0:

```
for line in range(0,2000):
    if  y_train.iloc[line] == 2:
        y_train.iloc[line] = 1
    if y_train.iloc[line] == 3:
        y_train.iloc[line] = 1              # Making the y target binary
```

Binary class 1:

```
for line in range(0,2000):
    if  y_train.iloc[line] == 2:
        y_train.iloc[line] = 0
    if y_train.iloc[line] == 3:
        y_train.iloc[line] = 0              # Making the y target binary
```

Binary class 2:

```
for line in range(0,2000):
    if  y_train.iloc[line] == 1:
        y_train.iloc[line] = 0
    if y_train.iloc[line] == 2:
        y_train.iloc[line] = 1
    if  y_train.iloc[line] == 3:
        y_train.iloc[line] = 0              # Making the y target binary
```

Binary class 3:

```
for line in range(0,2000):
    if  y_train.iloc[line] == 1:
        y_train.iloc[line] = 0
    if y_train.iloc[line] == 2:
        y_train.iloc[line] = 0
    if  y_train.iloc[line] == 3:
        y_train.iloc[line] = 1              # Making the y target binary
```