ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

ΔΗΜΗΤΡΗΣ ΖΕΡΚΕΛΙΔΗΣ 03400049
ΜΑΡΙΑ ΚΑΪΚΤΖΟΓΛΟΥ 03400052
ΜΑΡΙΑ-ΦΙΛΙΠΠΑ ΤΡΙΒΥΖΑ 03400080

Μάρτιος 2020

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

Ένα γνωστό πρόβλημα στην Ελλάδα, αλλά και στον υπόλοιπο κόσμο, είναι η ύπαρξη αδέσποτων ζώων συντροφιάς. Κάθε χρόνο, εκατομμύρια αδέσποτα ζώα συντροφιάς, είτε καταλήγουν σε πρόωρο θάνατο από τις άσχημες συνθήκες διαβίωσης στο δρόμο ή από κάποια ασθένεια, είτε καταλήγουν σε καταφύγια. Τα ζώα τα οποία καταλήγουν σε καταφύγια, συνήθως έχουν εγκαταληφθεί από τους ιδιοκτήτες τους, έχουν χαθεί, ή έχουν σωθεί από περιστάσεις σκληρότητας. Πολλά από αυτά τα ζώα βρίσκουν για πάντα οικογένειες για να τα πάρουν σπίτι, αλλά δεν είναι όλα τόσο τυχερά. Υπάρχουν άραγε τάσεις στην προτίμηση των ανθρώπων για τα συγκεκριμένα ζώα με βάση τη φυλή, το χρώμα ή την ηλικία τους; Τα χαρακτηριστικά των ζώων αυτών 'προδιαγράφουν' με κάποιο τρόπο την κατάληξή τους; Ως φιλόζωους, το ερώτημα αυτό μας κέντρισε το ενδιαφέρον, καθώς η απάντησή του θα μπορούσε να βοηθήσει τα καταφύγια να εστιάσουν την ενέργειά τους σε συγκεκριμένα ζώα που χρειάζονται περισσότερη βοήθεια για την εξεύρεση νέου σπιτιού.

Μελετάμε δεδομένα που προέρχονται από τις ΗΠΑ και αφορούν ζώα συντροφιάς που βρίσκονται σε καταφύγια, θέτοντας ως στόχο την πρόβλεψη της κατάληξης ενός ζώου με βάση τα χαρακτηριστικά του. Τα δεδομένα μας προέρχονται από το kaggle και συγκεντρώνονται σε ένα CSV αρχείο, το οποίο περιέχει τις εξής δέκα στήλες: AnimalID, Name, DateTime, OutcomeType, OutcomeSubtype, AnimalType, SexuponOutcome, AgeuponOutcome, Breed και Color. Στόχος μας, είναι η πρόβλεψη του OutcomeType (ή της έκβασης), το οποίο χωρίζεται στις εξής πέντε κατηγορίες: Return_to_owner, Euthanasia, Adoption, Transfer και Died.

Έχουμε να κάνουμε, λοιπόν, με ένα πρόβλημα ταξινόμησης κατηγορικών δεδομένων με πολλαπλές κλάσεις. Αυτή η φύση του προβλήματος μάς οδήγησε στην επιλογή των δέντρων αποφάσεων και άλλων, παραγόμενων από αυτήν, μεθόδων. Συγκεκριμένα, χρησιμοποιήθηκαν οι αλγόριθμοι random forest και xgboost, αλλά δοκιμάστηκαν και μέθοδοι όπως αυτή της λογιστικής παλινδρόμησης, των διανυσμάτων υποστήριξης, των κοντινότερων γειτόνων, τα πολυστρωματικά νευρωνικά δίκτυα και άλλα.

Στο στάδιο της προεπεξεργασίας των δεδομένων έγιναν, μεταξύ άλλων, καθαρισμός, ομαδοποίηση και εξαγωγή χαρακτηριστικών. Πριν την εισαγωγή τους στους decision-tree-based αλγορίθμους, τα δεδομένα μετατράπηκαν σε κατηγορικά, κι έτσι οι αλγόριθμοι που δοκιμάσαμε δέχτηκαν δεδομένα εισόδου με τα παρακάτω χαρακτηριστικά.

NameBinary	αν το ζώο έχει όνομα ή όχι
AnimalTypeBinary	αν το ζώο είναι σκύλος ή γάτα
OutcomeTypePrec	η ηλικία του ζώου σε ημέρες
Breed	η ράτσα του ζώου
Color	το χρώμα του ζώου
NameLen	το πλήθος των γραμμάτων του ονόματος του ζώου
NameFreq	η συχνότητα εμφάνισης του ονόματος
Intact	αν το ζώο ήταν στειρωμένο ή αστείρωτο
Sex	το φύλο του ζώου
AgePrec	Λαμβάνει τιμές 1 ως 5 και κατηγοριοποιεί την ηλικία δείχνοντας αν το ζώο είναι puppy, kitten, junior, adolesence, prime, mature κλπ.
Month	ο μήνας εγγραφής του ζώου στη βάση δεδομένων
Day	η ημέρα εγγραφής του ζώου στη βάση δεδομένων
Hour	η ώρα εγγραφής του ζώου στη βάση δεδομένων
TimeOfDay	η 'περίοδος' της ημέρας (πρωί - μεσημέρι - απόγευμα - βράδυ)
YearPeriod	Η περίοδος του χρόνου
Weekday	αν ήταν Σαββατοκύριακο ή οχι
DayName	η ημέρα της εβδομάδας

Πίνακας 1.1: Χαρακτηριστικά εισόδου των αλγορίθμων.

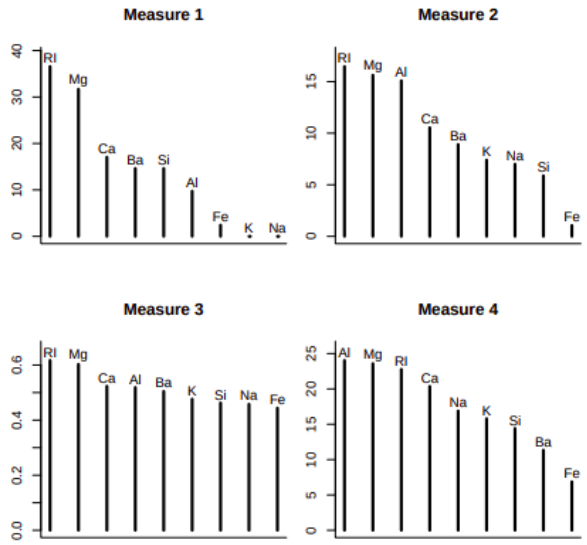
Η έξοδος ήταν μια από τις πέντε κλάσεις που προαναφέρθηκαν για την έκβαση. Σε επόμενες παραγράφους (Επεξεργασία Δεδομένων και Μέθοδοι), θα γίνει αναλυτική περιγραφή του τρόπου με τον οποίο προεπεξεργαστήκαμε τα δεδομένα και της λογικής πίσω από τις επιλογές μας. Το ίδιο και για τις μεθόδους που χρησιμοποιήσαμε.

2. ΣΧΕΤΙΚΗ ΕΡΕΥΝΑ

Εδώ θα αναφερθούμε σε υλοποιήσεις που έχουν γίνει πάνω σε αντίστοιχα προβλήματα με το Animal Shelter Outcomes.

2.1 Επιλογή των perdictors

Οι Andy Liaw και Merck & Co (2001) χρησιμοποιούν το dataset 'Forensic Glass', που εντόπισαν στο βιβλίο των Venables Ripley (2002) για να εφαρμόσουν μια λύση με τη μέθοδο random forest στην R. Ένα σημαντικό θέμα, που υπάρχει στο πρόβλημα της ταξινόμησης, είναι πόσες προβλέπουσες μεταβλητές θα χρησιμοποιηθούν. Οι Andy Liaw και Merck Co χρησιμοποιούν το variable importance που περιέχεται στο πακέτο του random forest στην R. Είναι μια έννοια δύσκολη να οριστεί και να εξερευνηθεί, γιατί η σημαντικότητα μιας μεταβλητής πολλές φορές επηρεάζεται από την αλληλεπίδραση αυτής της μεταβλητής με κάποια άλλη. Η μια μέθοδος που χρησιμοποιεί η R βασίζεται σε μετάθεση των out-of-bag (εκτός bootstrap δείγματος) δεδομένων, και η περιγραφή του βρίσκεται συνοπτικά στο επίσημο documentation της R. Η άλλη μέθοδος είναι ο υπολογισμός του Gini impurity. Οι Andy Liew και Merck & Co χρησιμοποιούν το πρώτο για να βρούνε ποιες predictors διαχωρίζουν καλύτερα τα δεδομένα. Καταλήγουν με 12 μεταβλητές από τις αρχικά χιλιάδες στο πλήθος. Η μέθοδος αυτή τους βοήθησε πολύ σε αυτό το dataset με τόσες μεταβλητές, όπου, μάλιστα, οι μεταβλητές δεν ερμηνεύονται ούτε αξιολογούνται λογικά με ευκολία (όπως για παράδειγμα στο δικό μας data set).



Εικόνα 2.1: Variable importance for the Forensic Glass data.

Το ίδιο μέτρο χρησιμοποίησαν και οι Ramon Casanova et al (2014), όταν εφάρμοσαν random forest σε ένα πρόβημα πρώιμης ανίχνευσης της ασθένειας Diabetic Retinopathy (DR) για να ανχνεύσουν τις μεταβλητές που επηρεάζουν περισσότερο την ακρίβεια ταξινόμησης.

'RF variable importance criteria revealed that microaneurysms counts in both eyes seemed to play the most important role in discrimination among the graded fundus variables, while the number of medicines and diabetes duration were the most relevant among the systemic variables.'

Χρησιμοποίησαν, εκτός από random forest, και logistic regression καταλήγοντας στο συμπέρασμα, ότι το random forest έκανε πολύ καλύτερη ταξινόμηση από τη λογιστική παλινδρόμηση. Αιτιολογούν την υπεροχή επικαλλούμενοι τη μη-γραμμικότητα του random forest και της δυνατότητάς του να εντοπίζει τις σημαντικές μεταβλητές αγνοώντας την επιρροή των μη σημαντικών.

'This advantage is very likely explained by RF non-linearity and its capability to detect important variables in the model while discarding the effects of the non-relevant ones.'

2.2 Αρχική εξερεύνηση των δεδομένων

Στη δημοσίευσή τους οι Ridhi Anand et al (2019) κάνουν μια στατιστική ανάλυση σε ένα πολύ παρόμοιο dataset με το δικό μας. Εκεί χρησιμοποιούν πολλά διαγράμματα που βοηθάνε στην εύρεση συσχετισμών μεταξύ των μεταβλητών, και της κατανομής τους. Αυτό είναι εξαιρετικά χρήσιμο για την επιλογή της μετρικής αξιολόγησης των ταξινομητών, κάτι που αξιοποιήσαμε κι εμείς στο πρόβλημά μας, όπου διαπιστώσαμε ότι οι κλάσσεις δεν είναι ισορροπημένες.

2.3 Επιλογή παραμέτρων

Σχετικά με την επιλογή παραμέτρων, οι Ridhi Anand et al (2019) αναφέρουν πως οι developers του random forest υποστηρίζουν ότι ο αλγόριθμος αυτός δε χρειάζεται ιδιαίτερη ρύθμιση (tuning) των παραμέτρων και πως οι default τιμές συχνά παράγουν καλά αποτελέσματα.

Οι Andy Liaw και Merck & Co (2001) αντίστοιχα σημειώνουν για τη ρύθμιση των παραμέτρων, μεταξύ άλλων, τα εξής:

(α) Ο απαιτούμενος αριθμός δέντρων αυξάνεται παράλληλα με τον αριθμό των predictors. Προτείνεται η σύγκριση των προβλέψεων μεταξύ ολόκληρου του forest και υποσυνόλων του. Όταν τα αποτελέσματα είναι εξίσου καλά σταματάει η πρόσθεση δέντρων.

(β) Όταν υπάρχει ανισορροπία στις κλάσεις, κρίνεται απαραίτητο να δοκιμαστεί ένας κανόνας απόφασης διαφορετικός του hard voting.

Οι Libera και Pereirab (2018) στη δημοσίευσή τους ασχολούνται με προβλήματα ταξινόμησης σε περιπτώσεις ύπαρξης πολλών κλάσεων και με κατηγορικά δεδομένα πολλών τιμών. Τονίζει τη σημασία συσσώρευσης χαρακτηριστικών για τη βελτίωση της απόδοσης σε αλγορίθμους, όπως τα δέντρα αποφάσεων ή τα τυχαία δάση και τη σύλληψη συσχετίσεων. Στην έρευνα αυτή, γίνεται χρήση κάποιων κριτηρίων διαχωρισμού εκ των οποίων εμείς χρησιμοποιήσαμε το Gini impurity. Το παρόν paper έχει χρησιμοποιήσει και το ίδιο dataset με εμάς και έχει κάνει κάποιες αλλαγές παρόμοιες με τις δικές μας. Για παράδειγμα, μετατρέπει την ηλικία σε ακέραιο αριθμό, που συμβολίζει τις μέρες ζωής, ενώ έχει κάνει ομαδοποίηση στο χαρακτηριστικό breed, ώστε να μειωθούν σε αριθμό οι κατηγορικές τιμές, καθώς οι σκύλοι είχαν 1300 ξεχωριστές breed-τιμές. Όπως και εμείς, αφαίρεσαν κάποια χαρακτηριστικά από την είσοδο, γιατί δε συνέβαλλαν στην αύξηση της αποδοτικότητας τού αλγορίθμου.

Ο F. Sirgist (2019) η ενισχυτική κλίση και η ενίσχυση του Νεύτωνα (Newton boosting), καθώς και μια υβριδική τεχνική αυτών των δύο, σε ένα ενοποιημένο πλαίσιο. Συγκρίνονται οι τρόποι ενίσχυσης των αλγορίθμων με τα δέντρα ως μαθητές βάσης σε ένα μεγάλο σύνολο δεδομένων παλινδρόμησης και ταξινόμησης χρησιμοποιώντας διάφορες συναρτήσεις απώλειας. Τα πειράματά τους δείχνουν ότι η ενίσχυση του Newton ξεπερνά την κλίση και την υβριδική κλίση-Newton που ενισχύει την ακρίβεια της πρόβλεψης στην πλειονότητα των συνόλων δεδομένων.

3. DATASET ΚΑΙ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Τα dataset μας προέρχεται από το Austin Animal Center, με δεδομένα από την 1η Οκτωβρίου του 2013 έως και το Μάρτιο του 2016, και περιλαμβάνει πληροφορίες σχετικά με τη φυλή, το χρώμα, το φύλο και την ηλικία ζώων συντροφιάς, που καταλήγουν σε καταφύγια στις ΗΠΑ. Σκοπός μας είναι να προβλέψουμε την έκβαση για κάθε ζώο. Οι εκβάσεις αυτές αντιπροσωπεύουν την κατάσταση των ζώων τη στιγμή που φεύγουν από το Animal Center, και περιλαμβάνουν τα εξής: Adoption, Died, Euthanasia, Return to owner, και Transfer. Όλα τα ζώα λαμβάνουν ένα μοναδικό ID κατά την εισαγωγή τους. Τα train και test δεδομένα έχουν χωριστεί τυχαία, και περιλαμβάνουν 38185 και 11456 IDs ζώων αντίστοιχα.

	AnimalID	Name	DateTime	AnimalType	SexuponOutcome	AgeuponOutcome	Breed	Color	OutcomeType
0	A671945	Hambone	2014-02-12 18:22:00	Dog	Neutered Male	1 year	Shetland Sheepdog Mix	Brown/White	Return_to_owner
1	A656520	Emily	2013-10-13 12:44:00	Cat	Spayed Female	1 year	Domestic Shorthair Mix	Cream Tabby	Euthanasia
2	A686464	Pearce	2015-01-31 12:28:00	Dog	Neutered Male	2 years	Pit Bull Mix	Blue/White	Adoption
3	A683430	NaN	2014-07-11 19:09:00	Cat	Intact Male	3 weeks	Domestic Shorthair Mix	Blue Cream	Transfer
4	A667013	NaN	2013-11-15 12:52:00	Dog	Neutered Male	2 years	Lhasa Apso/Miniature Poodle	Tan	Transfer

Εικόνα 3.1: Εικόνα αρχικού dataset.

3.1 Προεπεξεργασία Δεδομένων

Παρατηρούμε, αρχικά, ότι η στήλη 'OutcomeSubtype' δεν υπάρχει στα test δεδομένα, οπότε αφαιρέσαμε το χαρακτηριστικό αυτό και από τα train δεδομένα. Επίσης, οι καταγραφές περιέχουν ορισμένα ελλιπή στοιχεία στις στήλες 'SexuponOutcome' και 'AgeuponOutcome'. Το NaN value της πρώτης στήλης το συμπληρώσαμε με την πιο δημοφιλή καταγραφή της στήλης, και τα NaN values της δεύτερης στήλης τα αντικαταστήσαμε με τον αριθμό 0.

Στη συνέχεια, παρατηρούμε ότι τα δεδομένα μας περιλαμβάνουν πολλές κατηγορικές τιμές, επομένως θα δημιουργήσουμε νέες στήλες, οι οποίες χαρτογραφούν τις κατηγορικές αυτές τιμές σε ακέραιους αριθμούς. Συγκεκριμένα, δημιουργούμε μια νέα στήλη 'AnimalTypeBinary', η οποία περιέχει τις τιμές 0 και 1, αναλόγως της τιμής της στήλης 'AnimalType'. Δηλαδή, ο τύπος 'Dog' αντιστοιχεί στην τιμή 0 και ο τύπος 'Cat' στην τιμή 1. Επίσης, δημιουργούμε τη στήλη 'NameBinary', η οποία αποτελείται από τιμές 1 και 0, αναλόγως του αν η στήλη 'Name' περιέχει καταγραφή ή όχι, αντίστοιχα. Τη στήλη 'Name' τη χρησιμοποιήσαμε και για τη δημιουργία δύο νέων στηλών, τις 'NameLen' και 'NameFreq'. Η πρώτη περιλαμβάνει το μήκος του ονόματος της αντίστοιχης καταγραφής, ενώ η δεύτερη στήλη τη συχνότητα εμφάνισης αυτού του ονόματος.

Συνεχίζουμε την προεπεξεργασία των δεδομένων μας, μετατρέποντας τις τιμές της στήλης 'AgeuponOutcome' σε ημέρες, και αποθηκεύοντάς τες στη στήλη 'AgeuponOutcomeProc'. Τις ηλικίες αυτές μπορούμε να τις αντιστοιχίσουμε σε ορισμένες κατηγορίες, αναλόγως τον τύπο του ζώου. Συγκεκριμένα, οι γάτες μπορούν να χωριστούν στις εξής κατηγορίες, αναλόγως την ηλικίας τους: Kittens (0-6 μήνες), Junior (6 μήνες-2 χρόνια), Prime (3-6 χρόνια), Mature (7-10 χρόνια), Senior (11-14 χρόνια) και Geriatric (15 χρόνια και πάνω), ενώ οι σκύλοι στις: Puppyhood (τελειώνει μεταξύ 6 και 18 μήνες), Adolescence (ξεκινάει μεταξύ 6 και 18 μήνες), Adulthood (ξεκινάει μεταξύ 12 μήνες και 3 χρόνια) και senior years (ξεκινάνε μεταξύ 6 και 10 χρόνια). Δημιουργούμε τη στήλη 'AgePrec', η οποία περιλαμβάνει τις τιμές αυτές, αφότου τις έχουμε μετατρέψει σε ακέραιους αριθμούς, αναλόγως τα χρόνια ζωής του ζώου.

Χρησιμοποιούμε τη στήλη 'DateTime' για να εξάγουμε τις εξής πληροφορίες: Year, Month, Day και Hour. Την πληροφορία 'Hour' την χρησιμοποιούμε, ώστε να δημιουργήσουμε το χαρακτηριστικό 'TimeOfDayPrec', το οποίο περιλαμβάνει τις τιμές 'morning', 'midday', 'lateday' και 'night', αντιστοιχισμένες στους ακέραιους 1, 2, 3 και 4. Επίσης, χρησιμοποιούμε την πληροφορία 'Month', ώστε να δημιουργήσουμε τη στήλη 'YearPeriodPrec', η οποία περιλαμβάνει τις τιμές 'Winter', 'Autumn', 'Spring', 'Summer' και 'Christmas', αντιστοιχισμένες στους ακέραιους 1, 2, 3, 4 και 5. Τέλος, χρησιμοποιούμε την πληροφορία 'Day', ώστε να δημιουργήσουμε δύο νέα χαρακτηριστικά: 'Weekday', το οποίο αποτελείται από τιμές 1 και 0, αναλόγως του αν η στήλη 'Day' αντιστοιχεί σε Σαββατοκύριακο ή όχι, αντίστοιχα, και 'DayName', το οποίο αποτελείται από ακέραιους αριθμούς που αντιστοιχούν στις ημέρες της εβδομάδας.

Έπειτα, χωρίζουμε τη στήλη 'SexuponOutcome' στις στήλες 'Intact' και 'Sex'. Η πρώτη στήλη αποτελείται από τις τιμές 0, 1 και 2, αναλόγως του αν το ζώο έχει στειρωθεί, αν δεν έχει στειρωθεί και αν δεν ξέρουμε την κατάστασή του, αντίστοιχα. Η στήλη 'Sex' αποτελείται από τις τιμές 0, 1 και 2, οι οποίες αντιστοιχούν στα φύλα 'Female', 'Male' και 'Unknown'.

Παρατηρούμε ότι η στήλη 'Breed' περιέχει 1678 μοναδικές τιμές. Για να μειώσουμε τον αριθμό αυτό, μετατρέπουμε τις φυλές σκυλιών σε ομάδες σκυλιών, ακολουθώντας ένα συγκεκριμένο mapping, με αποτέλεσμα να δημιουργηθούν 132 ομάδες, αριθμός σαφώς μικρότερος του 1678. Έπειτα, δημιουργούμε τη στήλη 'BreedPrec', αντιστοιχίζοντας όλες τις μοναδικές τιμές της στήλης 'Breed' σε ακέραιους αριθμούς. Επιπλέον, ακολουθώντας αντίστοιχη λογική, δημιουργούμε ομάδες χρωμάτων σύμφωνα με τη στήλη 'Color', η οποία περιέχει 411 διαφορετικά χρώματα ζώων. Καταλήγουμε με 29 μοναδικά χρώματα, τα οποία αντιστοιχίζουμε σε ακέραιους αριθμούς και δημιουργούμε τη στήλη 'ColorPrec'.

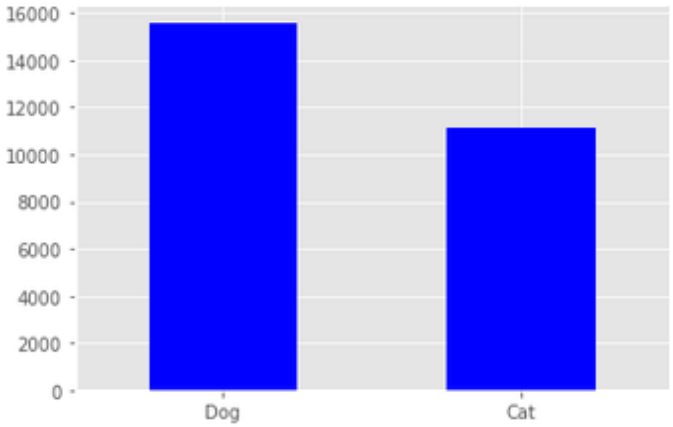
Τέλος, δημιουργούμε τη στήλη 'OutcomeTypePrec', η οποία αποτελείται από τις τιμές 1 έως 5, και αντιστοιχούν στις εκβάσεις 'Adoption', 'Died', 'Euthanasia', 'Return_to_owner' και 'Transfer', αντίστοιχα.

Έχοντας, πλέον, δημιουργήσει τα χαρακτηριστικά τα οποία θα χρησιμοποιήσουμε μετέπειτα στους αλγόριθμους που θα δοκιμάσουμε, διαγράφουμε τις στήλες: 'AnimalID', 'Name', 'DateTime', 'OutcomeType', 'AnimalType', 'SexuponOutcome', 'AgeuponOutcome', 'Breed', 'Color', 'TimeOfDay', 'YearPeriod', 'Age' και 'Year'. Το dataset μας τώρα αποτελείται από 18 στήλες. Πριν προχωρήσουμε στις μεθόδους ταξινόμησης, αποφασίζουμε να κανονικοποιήσουμε τις στήλες 'AgeuponOutcomeProc', 'NameLen' και 'NameFreq', και εφαρμόζουμε one hot encoding στις στήλες 'Hour', 'BreedPrec' και 'ColorPrec'.

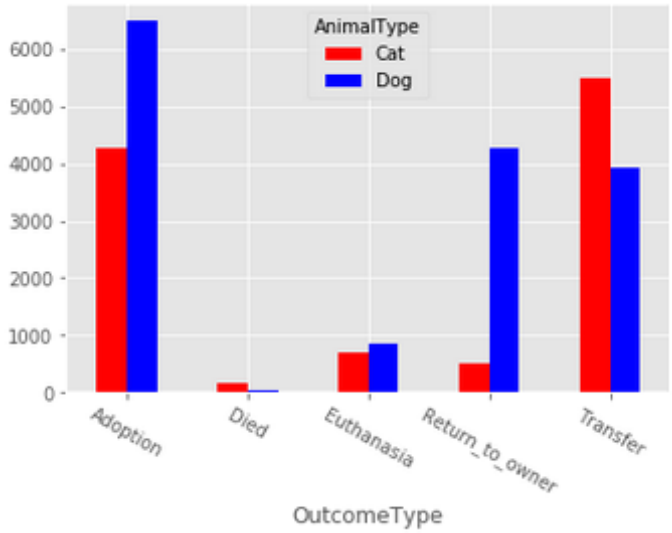
Σημειώνουμε πως, στην προσπάθειά μας να βρούμε τον καλύτερο αλγόριθμο που ταξινομεί τα δεδομένα μας, χρησιμοποιήσαμε το train set, το οποίο χωρίσαμε περαιτέρω σε train και test subset. Το νέο αυτό test set αποτελείται από το 20% του αρχικού train dataset.

3.2 Γραφικές Αναπαραστάσεις

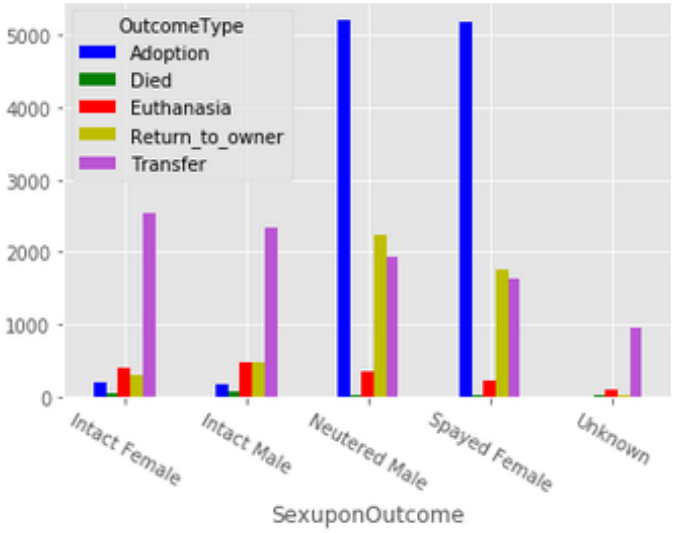
Στη συγκεκριμένη ενότητα δημιουργούμε ορισμένες οπτικές αναπαραστάσεις των αρχικών δεδομένων, ώστε να κατανοήσουμε την κατανομή τους, τις συχνότητες και τους συσχετισμούς τους.



Εικόνα 3.2: Animal Type.



Εικόνα 3.3: Outcome Type grouped by Animal Type.



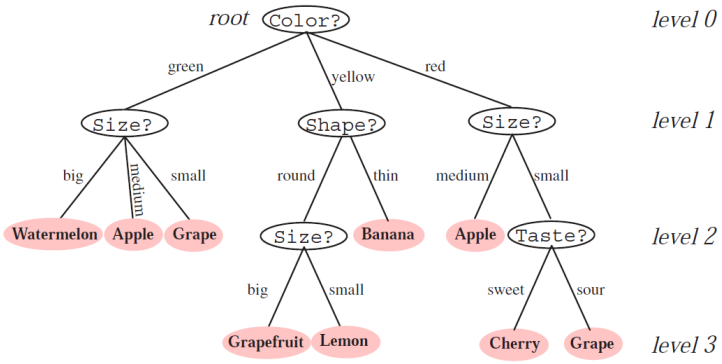
Εικόνα 3.4: Sex Upon Outcome grouped by Outcome Type.

Στην εικόνα 3.2, βλέπουμε τη συχνότητα της κατηγορίας των κατοικιδίων και παρατηρούμε πως οι σκύλοι είναι αρκετά περισσότεροι από τις γάτες. Στην εικόνα 3.3, βλέπουμε για κάθε κατηγορία κατοικιδίου το OutcomeType για κάθε κλάση. Παρατηρούμε πως έχουμε imbalanced classes γενικά, καθώς η ευθανασία και ο θάνατος έχουν αρκετά μικρή συχνότητα. Στην εικόνα 3.4, βλέπουμε για κάθε OutcomeType το SexuponOutcome για κάθε κλάση. Παρατηρούμε πως, τα στειρωμένα ζώα συντροφιάς έχουν περισσότερες πιθανότητες να υιοθετηούν, ενώ φαίνεται και πάλι πόσο unbalanced είναι το dataset που μελετάμε.

4. ΜΕΘΟΔΟΙ

4.1 Δέντρα Αποφάσεων (Decision Trees)

Η φύση του προβλήματος μάς έκανε να δοκιμάσουμε τα δέντρα αποφάσεων. Τα δέντρα αποφάσεων είναι μη-γραμμικός αλγόριθμος παλινδρόμησης / ταξινόμησης, που λειτουργούν ως πολυεπίπεδα συστήματα αποφάσεων, στα οποία οι κλάσσεις απορρίπτονται ακολουθιακά (Theodoridis & Koutroubas, 2003, ch. 4). Πώς κατασκευάζεται, όμως, το δέντρο αποφάσεων για την εκπαίδευση ενός συνόλου δεδομένων; Έστω ότι έχουμε ένα σύνολο δεδομένων με ετικέτες και ένα σύνολο 'ερωτήσεων' που θα διαχωρίσουν τα δεδομένα. Ως παράδειγμα, μπορούμε να φέρουμε μια απλή περίπτωση, που ετικέτες είναι 'ακέραιος μικρότερος του 10', 'ακέραιος μεγαλύτερος του 10', 'μη ακέραιος μικρότερος του 10', 'μη ακέραιος μεγαλύτερος του 10', και ένα αντίστοιχο σύνολο δειγμάτων. Θα μπορούσαμε να χρησιμοποιήσουμε τις ερωτήσεις 'είναι το δείγμα ακέραιος;' και 'είναι το δείγμα μεγαλύτερο του 10;', οι οποίες είναι επαρκείς μαζί για να κατηγοριοποιηθεί οποιοδήποτε δείγμα. Με ποια σειρά ωστόσο πρέπει να τις βάλουμε, έτσι ώστε να έχουμε το μικρότερο σφάλμα και το λιγότερο κόστος εκπαίδευσης; Κάθε φορά που γίνεται μια ερώτηση, το δέντρο διαχωρίζει τα δεδομένα σε μικρότερα και μικρότερα υποσύνολα. Η ιδανική περίπτωση είναι μια ερώτηση να διαχωρίσει τα δεδομένα σε υποσύνολα, τα οποία έχουν όλα δείγματα της ίδιας κλάσης. Τότε, το υποσύνολο λέγεται 'καθαρό' (pure), ωστόσο στην πράξη αυτό είναι σπάνιο, και αυτό που συμβαίνει είναι ότι το δέντρο κάθε φορά έχει να αποφασίσει αν θα συνεχίσει να ρωτάει ή αν θα σταματήσει σε μια ατελή ταξινόμηση (Duda, Hart and Stock, 2000, ch. 8). Η παρακάτω εικόνα από το βιβλιο Pattern Classification (Duda, Hart and Stock, 2000, ch. 8) οπτικοποιεί ένα δέντρο απόφασης όπου τα δεδομένα είναι είδη φρούτων. Τα κλαδιά υποδηλώνουν τις ερωτήσεις, δηλαδή τα κριτήρια διαχωρισμου που χρησιμοποιεί το δέντρο.



Εικόνα 4.1: Δέντρο απόφασης με δεδομένα είδη φρούτων.

Οι Duda, Hart και Stock ορίζουν έξι βασικά ερωτήματα που πρέπει να απαντηθούν όταν κατασκευάζεται το δέντρο απόφασης, τα οποία συνοπτικά έχουν να κάνουν με το αν τα κριτήρια διαχωρισμού θα είναι binary, ποια ερώτηση θα ρωτάται σε κάθε κόμβο, πότε θα σταματάει να μεγαλώνει το δέντρο και αν θα χρειαστεί 'κούρεμα', δηλαδή μικρότερο και απλότερο, και πως θα διαχειριστεί ελλιπή δεδομένα. Θα αναπτύξουμε λίγο παραπάνω το δεύτερο από αυτά, καθώς είναι μεγάλης σημασίας.

Έχει ήδη αναφερθεί η έννοια της 'καθαρότητας' και ξανατονίζουμε τη σημασία της, πως θέλουμε δέντρα με λίγους κόμβους που να επιτυγχάνουν όσο το δυνατόν 'καθαρότερο' διαχωρισμό. Για αυτό υπάρχει η έννοια της μη-καθαρότητας (impurity). Έχουν προταθεί αρκετά μαθηματικά μέτρα για την έννοια του impurity, τα οποία βασίζονται στην ίδια κεντρική ιδέα: Αν συμβολίσουμε με $i(N)$ το μέγεθος του impurity σε έναν κόμβο, τότε το $i(N)$->0 όσο τα δεδομένα στον κόμβο αυτόν ανήκουν στην ίδια κλάση, και η τιμή του $i(N)$ αυξάνεται όσο αυξάνονται οι κλάσσεις από όπου προέρχονται τα δεδομένα στον κόμβο (Duda, Hart and Stock, 2000, ch. 8). Τα πιο γνωστά και χρησιμοποιούμενα μέτρα είναι:
Εντροπία: $i(N) = -\sum P(\omega_j) log_2 P(\omega_j)$

Gini Inpurity: $i(N) = \sum_{i=j} P(\omega_i) P(\omega_j)$

Missclassification: $i(N) = 1 - max_j P(\omega_j)$
αλλά, φυσικά, υπάρχουν περισσότερα. Εμείς χρησιμοποιήσαμε το Gini impurity. Γενικά το ζητούμενο είναι το impurity να μειώνεται όσο το δέντρο επεκτείνεται.

4.2 Τυχαίο Δάσος (Random Forest)

Χρησιμοποιήσουμε τον αλγόριθμο random forest, ο οποίος είναι μια ensemble μέθοδος που βασίζεται στα δέντρα αποφάσεων. Συγκεκριμένα, πρόκειται για μια συλλογή από δέντρα αποφάσεων, των οποίων οι αποφάσεις συγκεντρώνονται για να βγει η τελική απόφαση. Ensembler είναι η μέθοδος, η οποία παράγει διάφορους ταξινομητές και αποφασίζει έχοντας όλα τα αποτελέσματα όλων των ταξινομητών. Το bagging είναι η τεχνική ensembling, κατά την οποία οι ταξινομητές δεν αλληλεπιδρούν στη διαδικασία μάθησης (σε αντίθεση με την τεχνική boosting). Στα δέντρα αποφάσεων, λοιπόν, κάθε δέντρο κατασκευάζεται με βάση μόνο το bootstrap δείγμα που έχει επιλεχθεί. Ο αλγόριθμος random forest διαχωρίζει έναν κόμβο του δέντρου λαμβάνοντας υπόψιν μόνο ένα υποσύνολο των μεταβλητών που προβλέπουν (predictors) (Andy Liaw, Merck Co, (2002)). Οι παράμετροί του, λοιπόν, είναι δύο, το πλήθος των μεταβλητών στο τυχαίο δείγμα σε κάθε κόμβο και το συνολικό αριθμό των δέντρων. Παρακάτω φαίνεται ο αλγόριθμος τυχαίων δασών.

Training Phase

Given

- X: the objects in the training data set (an $N \times n$ matrix)
- Y: the labels of the training set (an $N \times 1$ matrix)
- L: the number of classifiers in the ensemble
- K: the number of subsets
- $\{\omega_1, \ldots, \omega_c\}$: the set of class labels

For $i = 1 \ldots L$

- Prepare the rotation matrix R_i^a:
  - Split $\mathbf{F}$ (the feature set) into K subsets: $\mathbf{F}_{i,j}$ (for $j = 1 \ldots K$)
  - For $j = 1 \ldots K$
    - Let $X_{i,j}$ be the data set X for the features in $\mathbf{F}_{i,j}$
    - Eliminate from $X_{i,j}$ a random subset of classes
    - Select a bootstrap sample from $X_{i,j}$ of size 75% of the number of objects in $X_{i,j}$. Denote the new set by $X'_{i,j}$
    - Apply PCA on $X'_{i,j}$ to obtain the coefficients in a matrix $C_{i,j}$
  - Arrange the $C_{i,j}$, for $j = 1 \ldots K$ in a rotation matrix R_i as in equation (1)
  - Construct R_i^a by rearranging the the columns of R_i so as to match the order of features in $\mathbf{F}$.
- Build classifier D_i using (XR_i^a, Y) as the training set

Classification Phase

- For a given $\mathbf{x}$, let $d_{i,j}(\mathbf{x}R_i^a)$ be the probability assigned by the classifier D_i to the hypothesis that $\mathbf{x}$ comes from class ω_j. Calculate the confidence for each class, ω_j, by the average combination method:

$$\mu_j(\mathbf{x}) = \frac{1}{L}\sum_{i=1}^{L} d_{i,j}(\mathbf{x}R_i^a), \quad j = 1, \ldots, c.$$

- Assign $\mathbf{x}$ to the class with the largest confidence.

Εικόνα 4.2: Αλγόριθμος random forest.

4.3 SVM (Support Vectors Machine)

Δοκιμάσαμε και τον ταξινομητή SVM, ο οποίος πρόκειται για αλγόριθμο ταξινόμησης / παλινδρόμησης επιβλεπόμενης μάθησης. Μμέσω μετασχηματισμού των δεδομένων από το χώρο που βρίσκονται σε έναν χώρο μεγαλύτερης διάστασης, αυξάνει τις πιθανότητες να παράξει ένα ισοδύναμο πρόβλημα, που θα είναι όμως πλέον γραμμικά διαχωρίσιμο. Η διαδικασία αυτή γίνεται μέσω των συναρτήσεων πυρήνα (kernel functions)

$K : R^d R^{d\prime}$, όπου $d' > d$

$K(x_i, x_j) = \Phi(x_i) * \Phi(x_j)$

Όταν τα δεδομένα είναι γραμμικά διαχωρίσιμα, ο στόχος είναι το υπερεπίπεδο απόφασης να έχει ένα περιθώριο χώρο ανάμεσα από τα δεδομένα των κλάσεων, έτσι ώστε να μειώνεται η πιθανότητα σφάλματος. Ο SVM λειτουργεί πολύ καλά όταν το πλήθος των χαρακτηριστικών είναι μεγαλύτερο από τα δείγματα, κάτι που είναι πολύ μακριά από τη δική μας περίπτωση. Επίσης απαιτεί πολύ χρόνο για την ολοκλήρωση της εκπαίδευσης.

4.4 Λογιστική Παλινδρόμηση (Logistic Regression)

Μια ακόμα μέθοδος επιβλεπόμενης μάθησης που δοκιμάσαμε είναι η λογιστική παλινδρόμηση. Όταν υπάρχουν πολλές ετικέτες, χρησιμοποιεί τη συνάρτηση softmax, που ορίζεται ως

$softmax(z) = \frac{\exp(z)}{\sum_{i=1}^{k} \exp(z_i)}$

όπου z είναι μια γραμμική έκφραση του διανύσματος x (δηλ. $z = wx + B$). Η softmax δίνει στο output ουσιαστικά τις πιθανότητες το x να ανήκει σε μια κλάση και οι πιθανότητες αυτές συγκρίνονται μεταξύ τους ώστε να βρεθεί η μέγιστη. Η συνάρτηση κόστους, που χρησιμοποιείται για να ανανεώνονται τα βάρη, είναι η cross-entropy.

4.5 Κοντινότεροι Γείτονες (Nearest Neighbors)

Ο '$k-$πλησιέστερος γείτονας' αλγόριθμος ($k - NN$) είναι μια μη παραμετρική μέθοδος, που χρησιμοποιείται για ταξινόμηση και παλινδρόμηση. Η είσοδος αποτελείται από τα πλησιέστερα παραδείγματα εκπαίδευσης στο χώρο των χαρακτηριστικών. Η έξοδος εξαρτάται από το εάν το $k - NN$ χρησιμοποιείται για ταξινόμηση ή regression. Στη δική μας περίπτωση, έχουμε ταξινόμηση, οπότε η κλάση είναι αυτή που έχει την πλειοψηφία τους $k-$πλησιέστερους γείτονες.

4.6 LDA - Fisher's linear discriminant

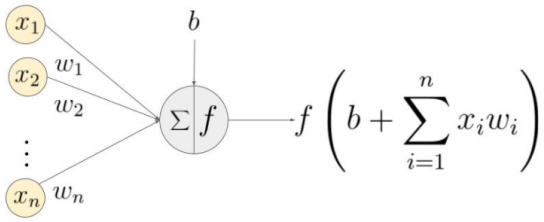
Η περιγραφή του αλγορίθμου βασίζεται στο βιβλίο Pattern Classification Duda Hart, Stock. Ο γραμμικός διαχωρισμός του Fisher είναι μια μέθοδος ταξινόμησης που προβάλλει δεδομένα μεγάλης διαστάσεως σε μια γραμμή και εκτελεί ταξινόμηση σε αυτόν τον μονοδιάστατο χώρο. Η προβολή μεγιστοποιεί την απόσταση μεταξύ των μέσων των δύο κλάσεων ενώ ελαχιστοποιεί τη διακύμανση εντός κάθε κατηγορίας. Αυτό ορίζει το κριτήριο Fisher, το οποίο μεγιστοποιείται σε όλες τις γραμμικές προβολές, w:

$J(\omega) = \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2}$

όπου το m αντιπροσωπεύει ένα μέσο, το s^2 αντιπροσωπεύει μια διακύμανση και οι δείκτες υποδηλώνουν τις δύο κατηγορίες.

4.7 Πολυστρωματικό Πέρσεπτρον (MLP)

Ένα MLP είναι μια κλάση του feedforward τεχνητού νευρωνικού δικτύου (ANN). Αποτελείται από τουλάχιστον τρία στρώματα κόμβων: ένα στρώμα εισόδου, ένα κρυμμένο στρώμα και ένα στρώμα εξόδου. Εκτός από τους κόμβους εισόδου, κάθε κόμβος είναι ένας νευρώνας που χρησιμοποιεί μια μη γραμμική λειτουργία ενεργοποίησης. Το MLP χρησιμοποιεί μια εποπτευόμενη τεχνική εκμάθησης που ονομάζεται back propagation για την εκπαίδευση. Τα πολλαπλά στρώματα και η μη γραμμική ενεργοποίηση διακρίνουν το MLP από ένα γραμμικό perceptron. Μπορεί να διακρίνει δεδομένα που δεν είναι γραμμικά διαχωρίσιμα.



Εικόνα 4.3: Perceptron.

Παραπάνω, φαίνεται η δομή ενός Perceptron-νευρώνα αναλυτικά. Παρατηρούμε την είσοδο, η οποία έχει διάφορα βάρη, τα οποία πολλαπλασιάζονται και προστίθενται. Στη συνέχεια, αυτή η τιμή είναι είσοδος στη συνάρτηση ενεργοποίησης του νευρώνα, ο οποίος παράγει το output. Σημαντικές συναρτήσεις ενεργοποίησης είναι οι εξής:

Υπερβολικής εφαπτομένης: $y(v_i) = tanh(v_i)$

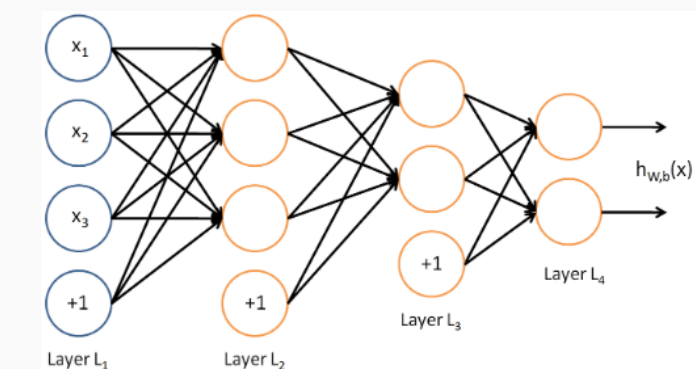
Σιγμοειδής: $y(v_i) = (1 + \exp(-v_i))^{-1}$

Υπάρχουν, φυσικά, και άλλες, όπως η ReLu. Περισσότερες συναρτήσεις ενεργοποίησης παρουσιάζονται εδώ.

Ο τρόπος που μαθαίνει ένα νευρωνικό δίκτυο είναι, όπως αναφέραμε, ο αλγόριθμος back propagation, ο οποίος χρησιμοποιεί μεθόδους gradient descent και chain rule.

Error output: $E(n) = \frac{1}{2} \sum_j e_j^2(n)$, το οποίο υπολογίζεται μετά από την εκτέλεση του forward αλγορίθμου.

Gradient Descent: $\Delta\omega_{ji}(x) = -\eta \frac{\partial E(n)}{\partial v_j(n)} y_i(n)$, όπου η ο ρυθμός μάθησης που ρυθμίζει την ταχύτητα σύγκλισης του αλγορίθμου.



Εικόνα 4.4: MLP με 4 εισόδους (με το bias) 2 κρυφά επίπεδα μεγέθους 4 και 3 και έξοδο με 2 νευρώνες.

4.8 Adaboost με Decision Trees

Το AdaBoost (Adaptive Boosting) είναι ένας από τους ensemble boosting classifier που προτάθηκε από τους Yoav Freund και Robert Schapire το 1996. Συνδυάζει πολλαπλούς ταξινομητές για να αυξήσει την ακρίβεια των ταξινομητών. Το AdaBoost είναι μια επαναληπτική μέθοδος συνόλου. Ο ταξινομητής AdaBoost δημιουργεί έναν ισχυρό ταξινομητή, συνδυάζοντας πολλούς ταξινομητές με χαμηλή απόδοση, έτσι ώστε να δημιουργηθεί ταξινομητής υψηλής ακρίβειας. Η βασική ιδέα πίσω από το Adaboost είναι η ρύθμιση των βαρών των ταξινομητών και η εκπαίδευση των training δεδομένων σε κάθε επανάληψη, έτσι ώστε να διασφαλίζεται ότι θα πραγματοποιηθούν ακριβείς προβλέψεις των ασυνήθιστων παρατηρήσεων. Οποιοσδήποτε αλγόριθμος μηχανικής μάθησης μπορεί να χρησιμοποιηθεί ως ταξινομητής βάσης, εάν δέχεται βάρη στο σετ εκπαίδευσης. Το AdaBoost λειτουργεί ιδιαίτερα καλά με το δέντρο αποφάσεων.

Το Adaboost θα πρέπει να πληροί δύο προϋποθέσεις. Η πρώτη είναι ο ταξινομητής να εκπαιδευτεί σε διάφορα διαβαθμισμένα παραδείγματα εκπαίδευσης και η δεύτερη να παρέχει μια καλή προσαρμογή σε αυτά τα παραδείγματα, ελαχιστοποιώντας το σφάλμα που παρουσιάστηκε κατά της εκπαίδευσης. Παρακάτω φαίνεται ο αλγόριθμος AdaBoost.

Algorithm AdaBoost

1: Init data weights $\{w_n\}$ to $1/N$
2: **for** $m = 1$ to M **do**
3: fit a classifier $y_m(x)$ by minimizing weighted error function J_m:
4: $J_m = \sum_{n=1}^{N} w_n^{(m)} 1[y_m(x_n) \neq t_n]$
5: compute $\epsilon_m = \sum_{n=1}^{N} w_n^{(m)} 1[y_m(x_n) \neq t_n] / \sum_{n=1}^{N} w_n^{(m)}$
6: evaluate $\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$
7: update the data weights: $w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m 1[y_m(x_n) \neq t_n]\}$
8: **end for**
9: Make predictions using the final model: $Y_M(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(x)\right)$

Εικόνα 4.5: Αλγόριθμος Adaboost με τύπους.

- Βήμα 1: Ο Adaboost επιλέγει τυχαία ένα υποσύνολο εκπαίδευσης και εκπαιδεύει ένα ταξινομητή βάσης.
- Βήμα 2: Υπολογίζει το σταθμισμένο ποσοστό σφάλματος (e) του δέντρου αποφάσης. Το σταθμισμένο ποσοστό σφάλματος (e) δείχνει το πλήθος των λανθασμένων συνολικά προβλέψεων στο σύνολο των ταξινομητών βάσης και χειρίζεται τις λανθασμένες προβλέψεις διαφορετικά, σύμφωνα με το βάρος των δεδομένων του. Όσο μεγαλύτερο είναι το βάρος, τόσο περισσότερο σταθμίζεται το αντίστοιχο σφάλμα κατά τον υπολογισμό του e.
- Βήμα 3: Υπολογίζει το βάρος του δέντρου απόφασης στο σύνολο των ταξινομητών βάσης.

```
Βάρος του δέντρου= learning rate * log( (1 – e) / e)
```

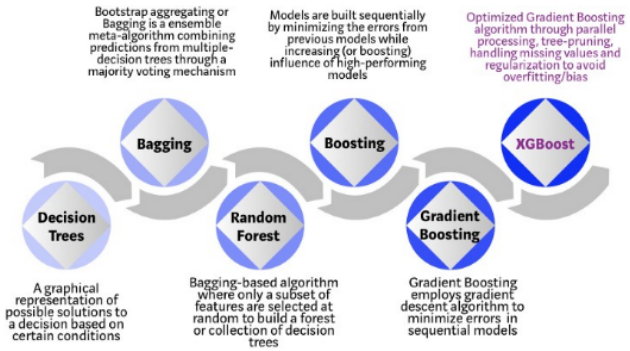
Όσο υψηλότερο είναι το σταθμισμένο ποσοστό σφάλματος ενός δέντρου, τόσο λιγότερο θα συμμετέχει κατά τη διάρκεια της ψηφοφορίας ενώ αντίθετα όσο χαμηλότερο είναι το σταθμισμένο ποσοστό σφάλματος ενός δέντρου, τόσο περισσότερο θα συμμετέχει στη μεταγενέστερης ψηφοφορίας.

- Βήμα 4: Ενημερώνει τα βάρη των εσφαλμένα ταξινομημένων δεδομένων. Το βάρος κάθε δεδομένου θα αλλάξει μόνο αν το μοντέλο πήρε αυτό το δεδομένο λάνθασμένων (το νέο βάρος του δεδομένου= παλαιό βάρος * np.exp (βάρος αυτού του δέντρου)). Όσο μεγαλύτερο είναι το βάρος του δέντρου, τόσο μεγαλύτερη βαρύτητα θα έχει το λανθασμένο δεδομένο από αυτό το δέντρο. Τα βάρη των δεδομένων κανονικοποιούνται αφού ενημερωθούν όλα τα εσφαλμένα ταξινομημένα δεδομένα.
- Βήμα 5: Επαναλαμβάνει το Βήμα 1 (έως ότου επιτευχθεί ο αριθμός των δένδρων που θέτουμε για εκπαίδευση)

Εικόνα 4.6: Αλγόριθμος Adaboost, περιγραφή από σημειώσεις μαθήματος.

4.9 XGBoost - Gradient Boosting

Το XGBoost είναι ένας αλγόριθμος Machine Learning, που βασίζεται σε δέντρα και χρησιμοποιεί μια μέθοδο ενίσχυσης κλίσης. Όταν πρόκειται για μικρά έως μεσαία δομημένα / πινακοειδή δεδομένα, οι αλγόριθμοι που βασίζονται σε δέντρο αποφάσεων θεωρούνται βέλτιστες στην τάξη αυτή τη στιγμή.



Εικόνα 4.7: Από τα Decision trees στο XGBoost.

Η παραπάνω εικόνα μάς δείχνει διαισθητικά πώς δημιουργήθηκαν τα XGBoost δέντρα. Βλέπουμε, πως από τα δέντρα αποφάσεων με τη μέθοδο bagging ή bootstrap, υλοποιούμε τα random forest. Στη συνέχεια, χρησιμοποιούμε τη μέθοδο Boosting, στην οποία δίνουμε παραπάνω βάρος στις λάθος ταξινομήσεις και επανεκπαιδεύουμε. Τέλος, υλοποιείται το gradient boosting και έχουμε τη μορφή δέντρων γνωστά ως XGBoost.

Ακολουθεί μια σύντομη περιγραφή για το gradient και Newton boosting, η οποία χρησιμοποιεί αναφορές από το paper 'Tree Boosting With XGBoost Why Does XGBoost Win "Every" Machine Learning Competition? Didrik Nielsen'.

Η ενίσχυση (boosting) αναφέρεται σε μια κλάση αλγορίθμων μάθησης, που ταιριάζουν μοντέλα συνδυάζοντας πολλά απλούστερα μοντέλα. Αυτά τα απλούστερα μοντέλα αναφέρονται συνήθως ως μοντέλα βάσης και μαθαίνονται χρησιμοποιώντας ένα βασικό εκπαιδευόμενο ή έναν ασθενή μαθητή. Αυτά τα απλούστερα μοντέλα τείνουν να έχουν περιορισμένη προγνωστική ικανότητα, αλλά όταν επιλέγονται προσεκτικά χρησιμοποιώντας έναν αλγόριθμο ενίσχυσης (boosting), σχηματίζουν ένα σχετικά πιο ακριβές μοντέλο. Αυτό μερικές φορές αναφέρεται ως μοντέλο ενός συνόλου, καθώς μπορεί να θεωρηθεί ως σύνολο βασικών μοντέλων. Το gradient boosting αναφέρεται πως μπορεί να ερμηνευθεί σαν ένας αλγόριθμος gradient descent για ένα χώρο συναρτήσεων (function space). Το Newton boosting από την άλλη πλευρά είναι ένας αλγόριθμος ενίσχυσης που υλοποιεί το xgboost.

Στο paper που αναφέραμε, δείχνεται ότι αυτό μπορεί να ερμηνευτεί ως μια μέθοδος του Νεύτωνα στο χώρο λειτουργίας και, επομένως, να το ονομάσουμε 'Newton boosting'. Οι δύο αυτοί αλγόριθμοι έχουν έτσι την ερμηνεία ότι είναι αριθμητικοί αλγόριθμοι βελτιστοποίησης στο χώρο συναρτήσεων. Αυτοί οι αλγόριθμοι ενίσχυσης είναι αρκετά γενικοί, καθώς είναι εφαρμόσιμοι για ένα ευρύ φάσμα λειτουργιών απώλειας και βασικών μαθητών.

Algorithm 2: Newton boosting

Input : Data set $\mathcal{D}$.
A loss function L.
A base learner $\mathcal{L}_\Phi$.
The number of iterations M.
The learning rate η.

Initialize $\hat{f}^{(0)}(x) = \hat{f}_0(x) = \hat{\theta}_0 = \arg\min_\theta \sum_{i=1}^{n} L(y_i, \theta)$;

for $m = 1,2,..,M$ **do**

$\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}^{(m-1)}(x)}$;

$\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=\hat{f}^{(m-1)}(x)}$;

$\hat{\phi}_m = \arg\min_{\phi\in\Phi} \sum_{i=1}^{n} \frac{1}{2}\hat{h}_m(x_i)\left[\left(-\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right) - \phi(x_i)\right]^2$;

$\hat{f}_m(x) = \eta\hat{\phi}_m(x)$;

$\hat{f}^{(m)}(x) = \hat{f}^{(m-1)}(x) + \hat{f}_m(x)$;

end

Output: $\hat{f}(x) \equiv \hat{f}^{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x)$

Εικόνα 4.8: Αλγόριθμος Newton Boosting.

5. ΠΕΙΡΑΜΑΤΑ / ΑΠΟΤΕΛΕΣΜΑΤΑ / ΣΧΟΛΙΑΣΜΟΣ

Σε αυτήν την ενότητα, παρουσιάζουμε τα πειράματα και τα αποτελέσματα του κάθε αλγορίθμου.

5.1 Μέθοδοι Εκτίμησης Αλγορίθμων

Έγινε χρήση μεθόδων διασταυρωμένης επικύρωσης για εύρεση αλγορίθμου με την υψηλότερη απόδοση (accuracy / f1 score) και Grid Search για hyperparameter tuning.

5.1.1 Διασταυρωμένη Επικύρωση (Cross Validation)

Αρχικά, χρησιμοποιούμε την μέθοδο διασταυρωμένης επικύρωσης 5-fold για να λάβουμε αποτελέσματα απόδοσης. Η μέθοδος αυτή είναι μια μέθοδος επαναδειγματοληψίας, η οποία χωρίζει το σύνολο δεδομένων μας σε k group και τα $k-1$ χρησιμοποιούνται για εκπαίδευση (train set), ενώ το ένα για πρόβλεψη (test set). Η συγκεκριμένη μέθοδος μειώνει τη διακύμανση της εκτίμησης. Η διασταυρούμενη επικύρωση χρησιμοποιείται, κυρίως, στην εφαρμοσμένη μηχανική μάθηση, για την εκτίμηση της προγνωστικότητας ενός μοντέλου σε δεδομένα, τα οποία δεν έχει ξαναδεί. Δηλαδή, θα χρησιμοποιήσουμε ένα περιορισμένο δείγμα για να εκτιμήσουμε τον τρόπο με τον οποίο το μοντέλο το προβλέπει. Γενικά, χρησιμοποιείται για να κάνει προβλέψεις για δεδομένα που δε χρησιμοποιήθηκαν κατά την εκπαίδευση του μοντέλου.

Στη δική μας περίπτωση, χωρίζουμε το dataset σε 5 ομάδες από τις οποίες οι 4 είναι το training set και η άλλη το test set.

5.1.2 Μετρικές Απόδοσης

Παρακάτω παρουσιάζουμε, συνοπτικά, μετρικές απόδοσης, που μελετήθηκαν και χρησιμοποιήθηκκαν για την εύρεση του καλύτερου αλγορίθμου.

ACCURACY: (Αριθμός σωστών προβλέψεων) / (Συνολικός αριθμός προβλέψεων).

PRECISION: Το μέτρο των σωστά προσδιορισμένων θετικών περιπτώσεων από όλες τις προβλεπόμενες θετικές περιπτώσεις. Είναι χρήσιμο όταν το κόστος των False Positives είναι υψηλό.

RECALL: Το μέτρο των σωστά προσδιορισμένων θετικών περιπτώσεων από όλες τις πραγματικές θετικές περιπτώσεις. Είναι σημαντικό όταν το κόστος των False Negatives είναι υψηλό.

F1-SCORE: Σταθμισμένος μέσος όρος του precision και recall. Η βαθμολογία F1 φτάνει την καλύτερη τιμή της στο 1 και τη χειρότερη στο 0.

Το accuracy μπορεί να χρησιμοποιηθεί όταν η κατανομή της τάξης είναι παρόμοια, ενώ η βαθμολογία F1 είναι καλύτερη μετρική όταν υπάρχουν imbalanced classes. Στα περισσότερα προβλήματα ταξινόμησης της πραγματικής ζωής, η ισορροπημένη κατανομή τάξης δεν υπάρχει, και έτσι η βαθμολογία F1 είναι μια καλύτερη μετρική για την αξιολόγηση του μοντέλου.

5.1.3 Αναζήτηση στο Πλέγμα (Grid Search)

Αρχικά, θα αναλύσουμε τη μέθοδο Grid Search, η οποία πραγματοποιήθηκε πάνω στα μοντέλα που τρέξαμε, ώστε να δούμε με ποιες υπερπαραμέτρους έχουμε καλύτερη απόδοση. Στη συγκεκριμένη στρατηγική αξιολογούνται όλοι οι πιθανοί συνδυασμοί δεδομένων διακριτών παραμέτρων. Αν υπάρχουν συνεχείς παράμετροι, τότε πρέπει να διακριτοποιηθούν εκ των προτέρων.

5.2 Δοκιμές Αλγορίθμων και Αποτελέσματα

Παρουσιάζουμε τους αλγορίθμους που δοκιμάσαμε και τα αντίστοιχα αποτελέσματά τους βάση των μεθόδων που παρουσιάστηκαν προηγουμένως.

5.2.1 SVM Ταξινομητές

Για τον αλγόριθμο SVM, δοκιμάσαμε να υλοποιηθεί με βάση τα 3 διαφορετικά kernels που δίνονται: Radial Basis Function, Linear και Polynomial. Καλύτερο μοντέλο με κριτήριο τη διασταυρωμένη επικύρωση για 5-fold αναδείχθηκε το SVM με τη

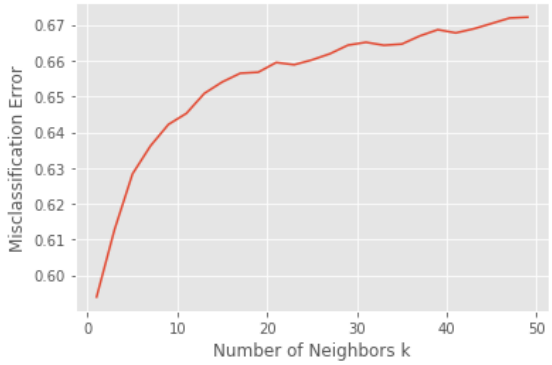
χρήση Polynomial kernel, αν και ήταν πολύ κοντά με τα άλλα δύο στα αποτελέσματα. Γενικά, είναι γνωστό από το θεώρημα του Cover πως όσο αυξάνεται η διαστατικότητα, τόσο αυξάνεται η πιθανότητα το πρόβλημα να είναι γραμμικά διαχωρίσιμο.

Kernel	Accuracy(%)	Macro Avg F1	Weighted Avg F1
RBF	62	0.36	0.60
Linear	63	0.38	0.61
Polynomial	64	0.36	0.61

Πίνακας 5.1: SVM Classifiers - Accuracy & F1 score.

5.2.2 K-NN Ταξινομητής

Για τον αλγόριθμο των k–κοντινότερων γειτόνων, η αναζήτηση πλέγματος πραγματοποιήθηκε χρησιμοποιώντας 5-fold cross validation και μετρική F1–score. Στην αναζήτηση αυτήν, ουσιαστικά τρέξαμε τον αλγόριθμο K-NN για 1-50 γείτονες και καταλήξαμε στο καλύτερο μοντέλο βάση της μετρικής που αναφέραμε. Η αναζήτηση αυτή, η οποία κατέληξε στην επιλογή $k = 1$ (1 κοντινότερος γείτονα), φαίνεται στο παρακάτω διάγραμμα.



	Accuracy(%)	Macro Avg F1	Weighted Avg F1
1-nn	60	0.40	0.60

Πίνακας 5.2: 1-NN Classifier - Accuracy & F1 score.

5.2.3 Naive Bayes Ταξινομητής

Τρέξαμε ένα μοντέλο Naive Bayes και λάβαμε accuracy 3.5%. Το συγκεκριμένο κακό ποσοστό είναι αναμενόμενο από το συγκεκριμένο ταξινομητή, λόγω της ύπαρξης πολλών κατηγορικών τιμών και του ότι θεωρεί πως κάθε χαρακτηριστικό είναι ανεξάρτητο από το άλλο.

5.2.4 MLP Ταξινομητής

Δοκιμάσαμε να εφαρμόσουμε και ένα νευρωνικό δίκτυο με 2 hidden layers, τα οποία αποτελούνται από 15 και 10 και αντίστοιχα. Μετά απο αρκετές δοκιμές καταλήξαμε στην εξής δομή και λόγω απόδοσης, αλλά και λόγω χρόνου εκπαίδευσης, καθώς τα μεγαλύτερα δίκτυα απαιτούν παραπάνω χρόνο.

Για αλγόριθμο καθόδου κλίσεως χρησιμοποιήθηκε ο Adam, που είναι ένας optimized stochastic gradient descent αλγόριθμος. Ο αλγόριθμος Adam, ανάλογα με την αβεβαιότητα του dataset, μικραίνει το learning rate. Επιπλέον, σαν activation function στους κόμβους, υλοποιήθηκε η ReLu, η οποία ορίζεται ως $max(0, x)$. Το learning rate είναι μια παράμετρος, η οποία ουσιαστικά μας υποδηλώνει πόσο μεγάλο βήμα κάνουν τα βάρη σε κάθε επανάληψη, για να πάνε στο ελάχιστο της συνάρτησης κόστους. Η υπερπαράμετρος αυτή ορίστηκε 0.001 και, τελικά, ο MLP ταξινομητής έδωσε 66.82% accuracy.

	Accuracy(%)	Macro Avg F1	Weighted Avg F1
MLP	67	0.44	0.65

Πίνακας 5.3: MLP Classifier - Accuracy & F1 score.

5.2.5 LDA (Linear Discriminant Analysis)

Ένας ακόμα αλγόριθμος μηχανικής μάθησης που παρουσιάστηκε είναι ο LDA. Αρχικά, εφαρμόσαμε PCA (Principal Component Analysis) πάνω στα δεδομένα μας κρατώντας τις 10 καλύτερες συνιστώσες με τη μεγαλύτερη διασπορά. Τα καινούρια χαρακτηριστικά, εξηγούσαν το 97% του variance των αρχικών δεδομένων μας. Ο LDA εκπαιδεύτηκε πολύ γρήγορα και έδωσε accuracy 61.5%.

5.2.6 Logistic Regression Ταξινόμηση

Η λογιστική παλινδρόμηση υλοποιήθηκε με τη μέθοδο LBFGS, η οποία είναι μια μέθοδος Quasi-Newton, που προσεγγίζει τον αλγόριθμο Broyden-Fletcher-Goldfarb-Shanno, χρησιμοποιώντας περιορισμένη υπολογιστική μνήμη. Η συγκεκριμένη μέθοδος έδωσε accuracy 63.20%.

Kernel	Accuracy(%)	Macro Avg F1	Weighted Avg F1
Logistic Regression	63	0.40	0.61

Πίνακας 5.4: Logistic Regression Classification - Accuracy & F1 score.

5.2.7 Random Forest Ταξινόμηση

Η μέθοδος των 'Τυχαίων Δασών' έχει αρκετές υπερπαράμετρους, επομένως η χρήση της αναζήτησης στο πλέγμα είναι απαραίτητη. Η αναζήτηση στο πλέγμα περιελάμβανε τις συγκεκριμένες υπερπαραμέτρους (στο σχολιασμό παρακάτω γίνεται αναφορά στο paper Hyperparameters and tuning strategies for random forest από τον Philipp Probst, Marvin N. Wright, Anne Laure Boulesteix):

1. Αριθμός δέντρων.

   Ο αριθμός των δένδρων σε ένα δάσος είναι μια παράμετρος που πρέπει να ρυθμιστεί αρκετά υψηλά. Ο ρυθμός σύγκλισης και, συνεπώς, ο αριθμός των δέντρων που απαιτούνται για την επίτευξη της βέλτιστης απόδοσης, εξαρτάται από τις ιδιότητες του συνόλου δεδομένων. Χρησιμοποιώντας ένα μεγάλο αριθμό συνόλων δεδομένων, οι Oshiro et al. (2012) και Probst και Boulesteix (2017) δείχνουν εμπειρικά ότι το μεγαλύτερο κέρδος απόδοσης μπορεί συχνά να επιτευχθεί κατά την καλλιέργεια των πρώτων 100 δέντρων. Περισσότερες δομές απαιτούνται, γενικά, για σταθερές εκτιμήσεις μεταβλητής σημασίας από ό,τι για τον απλό σκοπό πρόβλεψης.

2. Μέγιστο βάθος που φτάνει το κάθε δέντρο.

3. Ελάχιστος αριθμός δειγμάτων για το διαχωρισμό ενός κόμβου.

4. Ελάχιστος αριθμός δειγμάτων στους κόμβους – φύλλα.

   Η παράμετρος αυτή καθορίζει τον ελάχιστο αριθμό παρατηρήσεων σε έναν τερματικό κόμβο. Ρυθμίζοντας το χαμηλότερο, οδηγεί σε δέντρα με μεγαλύτερο βάθος, που σημαίνει ότι εκτελούνται περισσότερες χωρίσεις μέχρι τους τερματικούς κόμβους.

5. Χρήση μεθόδου επαναδειγματοληψίας bootstrap ή όχι.

   Αποσυσχετίζει τα δέντρα με την εισαγωγή του διαχωρισμού σε ένα τυχαίο σύνολο χαρακτηριστικών. Αυτό σημαίνει ότι, σε κάθε διάσπαση του δέντρου, το μοντέλο θεωρεί μόνο ένα μικρό υποσύνολο χαρακτηριστικών και όχι όλα τα χαρακτηριστικά του μοντέλου.

6. Κριτήριο διαχωρισμού των κόμβων (Gini impurity, missclassification, Entropy).

Καταλήξαμε στα εξής αποτελέσματα, με βάση 5-fold cross validation:

- 408 δέντρα.
- 10 δείγματα το ελάχιστο για τη διάσπαση ενός κόμβου.
- 2 δείγματα το ελάχιστο για να θεωρείται ο κόμβος σα φύλλο του δέντρου.
- Bootstrap = True.

- Μέγιστο βάθος δέντρων -> None, δηλαδή μπορεί να επεκταθεί όσο θέλει, ώσπου όλοι οι τελικοί κόμβοι είναι φύλλα.
- Gini impurity.

Τα παραπάνω μπορούν να παρουσιαστούν συνοπτικά στην παρακάτω εικόνα:

TABLE 1 Overview of the different hyperparameter of random forest and typical default values. n is the number of observations and p is the number of variables in the dataset

Hyperparameter	Description	Typical default values
mtry	Number of drawn candidate variables in each split	$\sqrt{p}$, $p/3$ for regression
Sample size	Number of observations that are drawn for each tree	n
Replacement	Draw observations with or without replacement	TRUE (with replacement)
Node size	Minimum number of observations in a terminal node	1 for classification, 5 for regression
Number of trees	Number of trees in the forest	500, 1,000
Splitting rule	Splitting criteria in the nodes	Gini impurity, p value, random

Εικόνα 5.1: Hyperparameters and tuning strategies for random forest by Philipp B., Marvin N.W. , Anne-Laure B.

Η απόδοση του αλγορίθμου random forest είναι 68.57% με τη χρήση των παραπάνω υπερπαραμέτρων, μέσω της αναζήτησης πλέγματος. Είναι λογικό το random forest να έχει μεγαλύτερη απόδοση από όλους τους άλλους αλγορίθμους, καθώς τα δεδομένα μας είναι κυρίως κατηγορικά και δεν έχουν σχέση με μετρικούς χώρους, όπως ο ευκλείδιος, στον οποίο παίζει ρόλο το μέγεθος της τιμής.

	Accuracy(%)	Macro Avg F1	Weighted Avg F1
Random Forest	69	0.43	0.66

Πίνακας 5.5: Random Forest Classification - Accuracy & F1 score.

5.2.8 Decision Tree - AdaBoost - Gradient Boosting

Δοκιμάσαμε να τρέξουμε ένα μοντέλο με ένα δέντρο απόφασης, χωρίς κάποιο αλγόριθμο boosting. Αυτό που πετύχαμε ήταν ένα ποσοστό ακριβείας 60.89%.

	Accuracy(%)	Macro Avg F1	Weighted Avg F1
Decision Tree	61	0.43	0.61

Πίνακας 5.6: Decision Tree Classifier - Accuracy & F1 score.

Στη συνέχεια, με τη χρήση του αλγορίθμου ενίσχυσης AdaBoost με 500 εκτιμητές, παρατηρήσαμε αύξηση της τάξης του 8%, δηλαδή ακρίβεια 68.20%.

	Accuracy(%)	Macro Avg F1	Weighted Avg F1
ADABoost Dec.Tree	68	0.45	0.66

Πίνακας 5.7: ADABoost and Decision Tree - Accuracy & F1 score.

Τέλος, δοκιμάστηκε και ο αλγόρθμος Gradient Boosting του SKlearn ο οποίος έδωσε ποσοστό ακριβείας 68.24%.

	Accuracy(%)	Macro Avg F1	Weighted Avg F1
SKlearn Gradient Boosting	68	0.47	0.67

Πίνακας 5.8: SKlearn Gradient Boosting - Accuracy & F1 score.

Παρακάτω που θα αναφερθούμε στο XGBoost θα δούμε πως υλοποιει ταχύτερα τον αλγόριθμο και πετυχαίνει καλύτερα αποτελέσματα, χρησιμοποιώντας τις ίδιες υπερπαραμέτρους, που θα αναλυθούν παρακάτω.

5.2.9 XGBoost Ταξινομητής

Υπερπαράμετροι - Ανάλυση

Οι δύο κύριοι υπερπαράμετροι, τόσο της καθόδου κλίσεως όσο και της ενίσχυσης του Νεύτωνα, είναι ο αριθμός των επαναλήψεων M και η παράμετρος της εκμάθησης η. Αυτές οι παράμετροι δεν είναι ανεξάρτητες και πρέπει να επιλεγούν από κοινού.

1. Ο αριθμός των επαναλήψεων M:

   Καθώς ο αριθμός των επαναλήψεων M αυξάνεται, η πολυπλοκότητα του μοντέλου θα τείνει να αυξηθεί. Αυτό δεν προκαλεί έκπληξη, καθώς η επέκταση της λειτουργίας βάσης τείνει να έχει μεγαλύτερη ικανότητα αναπαραγωγής, όταν ο αριθμός των βασικών λειτουργιών αυξάνεται. Επομένως, σε κάποιο σημείο, η αύξηση του αριθμού των επαναλήψεων θα οδηγήσει σε overfit, δηλαδή υπερβολική προσαρμογή. Κατά συνέπεια, το regularization μπορεί να επιτευχθεί με την έγκαιρη διακοπή. Ένας κατάλληλος αριθμός επαναλήψεων M καθορίζεται, συνήθως, με την παρακολούθηση της ακρίβειας της πρόβλεψης σε ένα σετ επικύρωσης ή μέσω της διασταυρωμένης επικύρωσης.

2. Ρυθμός μάθησης η:

   Ο Friedman (2001) βρήκε εμπειρικά ότι οι μικρότερες τιμές της η τείνουν να βελτιώνουν την απόδοση γενίκευσης. Με τη μείωση του, όμως, ο αριθμός των απαιτούμενων επαναλήψεων M τυπικά αυξάνεται. Έτσι, η μείωσή του η έρχεται με το κόστος της μεγαλύτερης υπολογιστικής ζήτησης. Θα πρέπει να ρυθμιστεί ο ρυθμός εκμάθησης η, όσο χαμηλότερα αντέχουμε υπολογιστικά και, έπειτα, να καθορίστει ο βέλτιστος αριθμός επαναλήψεων γι'αυτό το η.

3. Μέγιστο βάθος δένδρων:

   Το μήκος της μεγαλύτερης διαδρομής από τη ρίζα του δέντρου σε ένα φύλλο.

4. Colsample_bytree:

   Είναι η αναλογία υπο-δειγμάτων των στηλών κατά την κατασκευή κάθε δέντρου. Η υποδειγματοληψία εμφανίζεται μία φορά για κάθε δομημένο δέντρο.

5. Subsample:

   Ο ορισμός του σε 0,5 σημαίνει ότι το XGBoost θα δειχτεί τυχαία το μισό από τα δεδομένα εκπαίδευσης πριν από την δημιουργία δέντρων και αυτό θα αποτρέψει την υπερφόρτωση. Η υποδειγματοληψία θα εμφανιστεί μία φορά σε κάθε επαναλαμβανόμενη επανάληψη. Λαμβάνει τιμές από 0 έως 1.

6. n_estimators:

   Ο αριθμός των δένδρων που θα δημιουργηθούν. Πρέπει να γίνει tune παράλληλα με το learning rate για να αποφύγουμε το overfitting. Η χρήση της επικυρωμένης διασταύρωσης συνίσταται για τη μείωση της διασποράς της εκτίμησης.

7. Reg_alpha:

   L1 regularization στους όρους των βαρών. Η αύξηση αυτής της τιμής θα κάνει το μοντέλο πιο απλό για την αποφυγή του overfitting.

Κάνοντας αναζήτηση στο πλέγμα, λάβαμε τις παρακάτω υπερπαραμέτρους, και accuracy 69.72%:

reg_alpha	1.77827941
colsample_bytree	0.5722532094026564
objective	'multi:softprob'
no_estimators	115
max_depth	22
learning_rate	0.7578955028741008
subsample	0.848348985132586

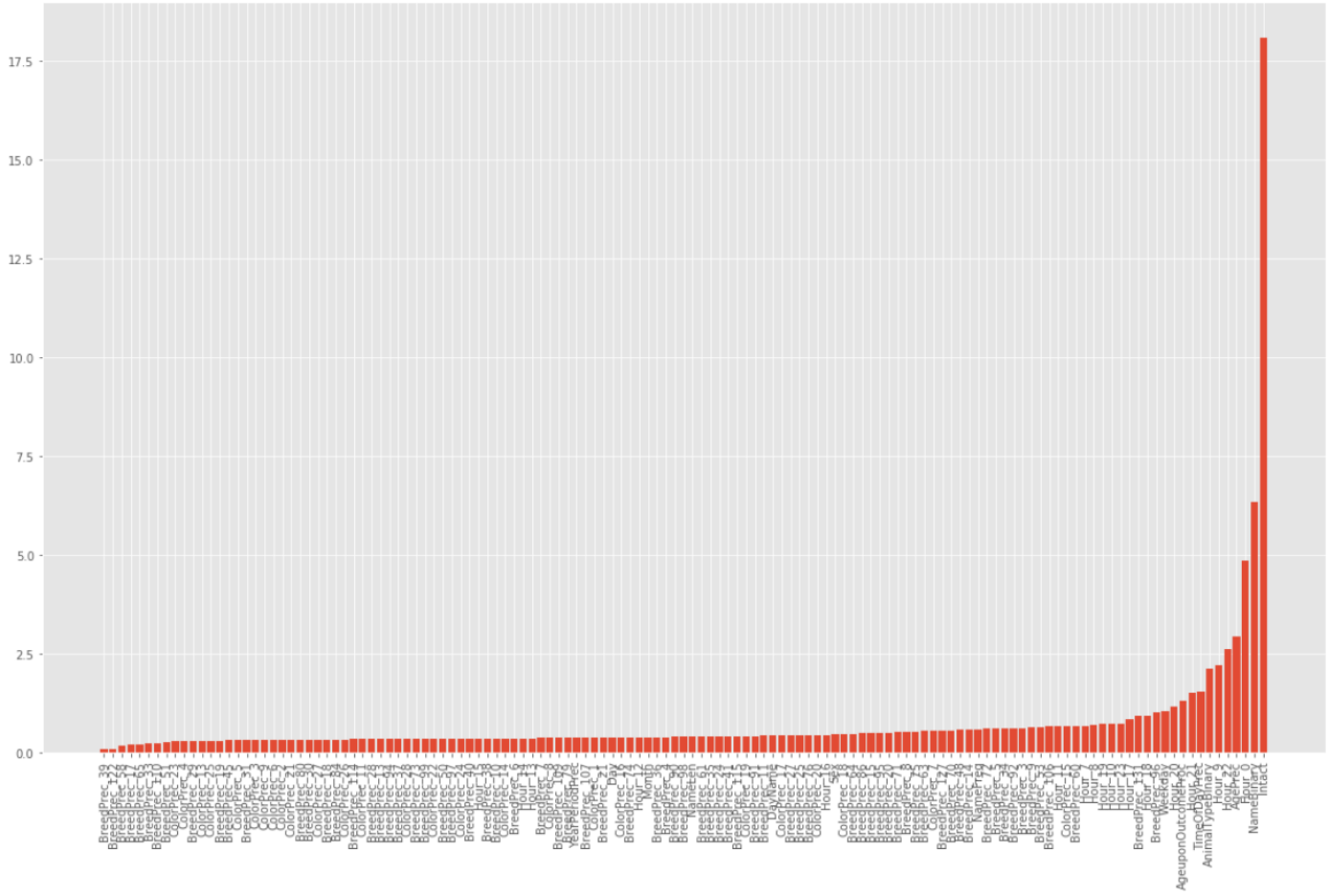
Πίνακας 5.9: Υπερπαράμετροι XGBoost.

	Accuracy(%)	Macro Avg F1	Weighted Avg F1
XGBoost	70	0.49	0.69

Πίνακας 5.10: XGBoost Classifier - Accuracy & F1 score.

Feature Importance

Παρακάτω, παρουσιάζεται η σημαντικότητα του κάθε χαρακτηριστικού στο μοντέλο, που κάναμε, τελικά, την υποβολή στο kaggle (XGBoost). Είναι εμφανές πως το πιο σημαντικό χαρακτηριστικό στην προγνωστικότητα του μοντέλου μας είναι το Intact, ενώ οι ράτσες φαίνεται ότι δεν επηρεάζουν την έκβαση ιδιαίτερα.



Εικόνα 5.2: Feature Importance.

Accuracy σε 5-fold cross validation

Decision Tree	59.89%
1-NN	58.54%
SVM RBF kernel	61.92%
SVM Linear kernel	62.83%
SVM Polynomial kernel	63.37%
MLP	65.83%
Logistic Regression	62.48%
LDA	61.52%
Random Forest	68.10%
Adaboost and Dec.Tree	65.96%
Sklearn Grad.Boosting	68.33%
XGBoost	69.18%

Πίνακας 5.11: Accuracy σε 5-fold cross validation επί τοις εκατό για τον κάθε αλγόριθμο.

5.2.10 Voting Ταξινομητές

Σε αυτήν την υπο-ενότητα θα αναφέρουμε voting ταξινομητές, με Soft Voting, που χρησιμοποιήσαμε για ένα καλύτερο και πιο σταθερό αποτέλεσμα.

Αρχικά κάναμε χρήση random forest, XGBoost και K-NN αλγορίθμου. Το accuracy του συγκεκριμένου μοντέλου δεν ήταν αρκετά υψηλό και δεν ξεπέρασε το XGBoost, που μέχρι τώρα ήταν το πρώτο σε απόδοση. Με αυτόν τον ταξινομητή πετύχαμε 64.89% accuracy. Ο λόγος που συνδύασαμε τους συγκεκριμένους ταξινομητές είναι μήπως βελτιωνόταν η απόδοση, λόγω του ότι οι δύο θα διόρθωναν τον ένα, μιας που παρατηρήσαμε ότι βρίσκανε διαφορετικές κλάσεις από τα confusion matrices και είχαν κάποιες διαφορές στις μετρικές απόδοσης τους, που φαίνονται από ένα classification report.

Έπειτα, δοκιμάσαμε και τους ταξινομητές random forest, XGBoost και SVM με soft voting. Το accuracy σε αυτή την περίπτωση έφτασε το 68.67%, ωστόσο και πάλι δεν ξεπέρασε το accuracy του XGBoost.

Ο τρίτος voting ταξινομητής αποτελείται από τα μοντέλα XGBoost, AdaBoost με δέντρα αποφάσεων και τυχαία δάση. Το αποτέλεσμα που έδωσε accuracy 69.62% και σε 5-Fold διασταυρωμένης επικύρωσης ήταν 68.98% ακρίβεια.

Ο τέταρτος voting ταξινομητής αποτελείται από το XGBoost και το μοντέλο AdaBoost με δέντρα αποφάσεων και έδωσε accuracy 69.30% και ακρίβεια σε 5-fold διασταυρωμένης επικύρωσης 68.84%.

Οι παραπάνω δύο ταξινομητές παρά τα πολύ κοντά τους αποτελέσματα με το XGBoost, δεν το ξεπέρασαν και, επομένως, παραμένει το καλύτερό μας μοντέλο.

5.3 Παρατηρήσεις

Βλέπουμε πως το XGBoost υπερίσχυσε των άλλων ταξινομητών κάτι το οποίο μας παραπέμπτει στο paper 'WHY XGBOOST WINS EVERY MACHINE LEARNING COMPETITION' από τον Didrik Nielsen. Στη συγκεκριμένη έρευνα αναφέρει πως, η δενδρική ενίσχυση (Tree boosting) είναι τόσο αποτελεσματική, επειδή ταιριάζει με μοντέλα προσθέτων δέντρων, τα οποία έχουν πλούσια αναπαραστατική ικανότητα, χρησιμοποιώντας προσαρμοστικά καθορισμένες γειτονιές. Επίσης, χρησιμοποιείται έξυπνο penalization των μεμονωμένων δένδρων, με αποτέλεσμα να έχουν ποικίλο αριθμό τερματικών κόμβων. Τέλος, το XGBoost κάνει χρήση του newton boosting με σκοπό να μάθει καλύτερα τις δομές δένδρων.

Classification Report των 2 καλύτερων αλγορίθμων

	precision	recall	f1-score	support
Adoption	0.72	0.82	0.77	2159
Died	0.50	0.03	0.06	31
Euthanasia	0.71	0.24	0.36	322
Return_to_owner	0.51	0.51	0.51	950
Transfer	0.76	0.75	0.75	1884
accuracy			0.70	5346
macro avg	0.64	0.47	0.49	5346
weighted avg	0.70	0.70	0.69	5346

Εικόνα 5.3: Classification Report του XGBoost.

	precision	recall	f1-score	support
Adoption	0.69	0.85	0.76	2159
Died	0.00	0.00	0.00	31
Euthanasia	0.96	0.08	0.15	322
Return_to_owner	0.51	0.47	0.49	950
Transfer	0.76	0.72	0.74	1884
accuracy			0.69	5346
macro avg	0.58	0.42	0.43	5346
weighted avg	0.70	0.69	0.66	5346

Εικόνα 5.4: Classification Report του Random Forest.

5.4 Kaggle Submission Αποτελέσματα

Μετά από όλα αυτά τα πειράματα και τις επιδόσεις που λάβαμε για τον κάθε αλγόριθμο, καταλήξαμε στο μοντέλο XGBoost για να καταθέσουμε τα αποτελέσματα του test set που δίνονται από το kaggle, και να δούμε το σκορ και την κατάταξη μας σε σχέση με άλλα μοντέλα. Το kaggle υπολογίζει το multiclass logloss, το οποίο δίνεται από τον παρακάτω τύπο:

$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(p_{ij})$

όπου N είναι ο αριθμός των ζώων στο test set και M είναι ο αριθμός των κλάσεων που θέλουμε να προβλέψουμε (5 στη δική μας περίπτωση). Η τιμή y_{ij} είναι 1 αν η παρατήρηση i έχει ως σωστή πρόβλεψη την κλάση j, αλλιώς 0. Ενώ p_{ij} είναι η πιθανότητα που προέβλεψε το μοντέλο μας η παρατήρηση i να ανήκει στην κλάση j.

Τελικά, $logloss = 0.72753$, το οποίο μας κατατάσσει στη θέση 256 από τους 1599 περίπου στο τοπ 15% του διαγωνισμού, χωρίς τη χρήση leaked data.

6. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΔΟΥΛΕΙΑ

6.1 Συμπεράσματα

Στην παρούσα εργασία, ασχοληθήκαμε με το πρόβλημα ταξινόμησης κατηγορικών δεδομένων. Τα δεδομένα προέρχονται από το Austin Animal Center, αφορούν πληροφορίες για ζώα που καταλήγουν σε καταφύγια στις ΗΠΑ και είναι καταγεγραμμένα από 1 Οκτωβρίου 2013 μέχρι το Μάρτιο του 2016. Τα δείγματα αποτελούνται από δέκα χαρακτηριστικά που αφορούν πληροφορίες για τα ζώα και η πρόβλεψη γίνεται για μια από τις πέντε κλάσεις που δείχνουν την έκβαση που είχε η ζωή του ζώου. Αρχικά πραγματοποιήθηκε προεπεξεργασία των δεδομενων για να διασφαλιστεί η καθαρότητα, η συνέπεια και η πληρότητά τους, αλλά και για να χρησιμοποιήσουμε τα σημαντικά χαρακτηριστικά που θα βοηθήσουν στην επίτευξη ακρίβειας των ταξινομητών.

Στη συνέχεια κάναμε οπτικές αναπαραστάσεις στα δεδομένα για να κατανοήσουμε την κατανομή τους, τις συχνότητες και τους συσχετισμούς τους. Αυτό μας βοήθησε να διαπιστώσουμε την ύπαρξη μη ισορροπημένων κλάσεων και αντιστοίχως να επιλέξουμε το F1-score ως μέτρο αξιολόγησης των ταξινομητών.

Οι αλγόριθμοι που χρησιμοποιήσαμε ήταν οι Decision Tree, Random Forest, SVM, Logistic Regression, K-NN, MultiLayer Perceptron, LDA, Gaussian Naive Bayes και XGBoost. Από σχετική μας έρευνα πάνω σε υλοποιήσεις που έγιναν σε προβλήματα παρόμοιας φύσης δώσαμε έμφαση στους αλγόριθμους που είναι παράγωγοι των δέντρων αποφάσεων (Random Forest, XGBoost), οι οποίοι πράγματι έδωσαν τα καλύτερα αποτελέσματα. Το αποτέλεσμα αυτό δικαιολογείται από το ότι τα δεδομένα είναι κατηγορικά και αυτοί οι δυο αλγόριθμοι πλεονεκτούν έναντι των άλλων σε αυτήν την περίπτωση. Ένα ακόμα πλεονέκτημα είναι ότι οι decision tree-based αλγόριθμοι λειτουργούν πολύ καλά σε μη γραμμικά δεδομένα.

Ξεκινώντας από τις χαμηλότερες επιδόσεις, μπορούμε να πούμε ότι ο συσχετισμός των δεδομένων ήταν καταλυτικός για την επίδοση του GNB, ο οποίος βασίζεται στην υπόθεση ότι τα δεδομένα είναι ασυσχέτιστα.

Το μεγάλο πλήθος δεδομένων και η ανισορροπία των κλάσεων είναι κάτι που επέδρασε αρνητικά στον KNN, ο οποίος δε λειτουργεί καλά κάτω από αυτές τις προϋποθέσεις.

Η λογιστική παλινδρόμηση είχε μέτρια απόδοση, καθώς τη δυσκόλεψε η μη γραμμικότητα και οι συσχετίσεις μεταξύ των δεδομένων.

Ο SVM δυσκολεύτηκε λόγω του μεγάλου όγκου δεδομένων, αλλά πρέπει να τονίσουμε ότι είναι ένας αλγόριθμος που απαιτεί ενδελεχή μελέτη των παραμέτρων που θα χρησιμοποιηθούν. Έχοντας επιλέξει να επικεντρωθούμε σε άλλους αλγορίθμους δεν μπορούμε να εξάγουμε περαιτέρω συμπεράσματα για την εν δυνάμει επίδοσή του στο πρόβλημα μας.

Το ίδιο θα πρέπει να πούμε και για το MLP, που αν και είναι σε θέση να αντιμετωπίσει τη μη-γραμμικότητα (φαίνεται από το αυξημένο accuracy που πέτυχε) περιέχει πληθώρα υπερπαραμέτρων που χρειάζονται tuning. Θα μπορούσαμε να χρησιμοποιήσουμε άλλα frameworks (pytorch, tensorflow) και να χτίσουμε λεπτομερώς ένα κατάλληλο νευρωνικό δίκτυο, ωστόσο ήταν εκτός του βασικού σκοπού αυτής της εργασίας, κι έτσι αρκεστήκαμε στη χρήση του υλοποιημένου από το sklearn MLP.

Ο Random Forest είναι χτισμένος, όπως αναφέραμε, με τεχνική ensemble πάνω στα decision trees, και το μεγάλο μέγεθος του dataset ευνοεί τη συνεχόμενη δειγματοληψία και τη μείωση του σφάλματος. Επίσης, λειτουργεί πολύ καλά με μη γραμμικά δεδομένα και με μη ισορροπημένες κλάσεις, όπως είναι η περίπτωσή μας. Επίσης, μπορεί να ελέγχει τη σημαντικότητα των μεταβλητών, κάτι που, σε παραλληλία με τη δική μας χειροκίνητη επεξεργασία των χαρακτηριστικών, απέδωσε πολύ καλά αποτελέσματα.

Τέλος, ο XGBoost πέτυχε την καλύτερη επίδοση στην ταξινόμηση, καθώς έχει όλα τα πλεονεκτήματα του random forest κι επιπλέον η δενδρική ενίσχυση ευνοεί αυτό το μοντέλο εφαρμόζωντας 'τιμωρία' (penalization) σε μεμονομένα δέντρα (Nielsen, D. 2016). Επίσης, μαθαίνει καλύτερα τις δομές των δέντρων χρησιμοποιώντας τη μέθοδο Newton Boosting, για την οποία, ωστόσο, δε θα εξηγήσουμε κάτι παραπάνω σε αυτήν την εργασία, αλλά πληροφορίες μπορούν να βρεθούνε στη δημοσίευση του Nielsen (2016). Για την επιλογή των παραμέτρων βασιστήκαμε πολύ στη μέθοδο αναζήτησης πλέγματος, διασταυρώνοντας τα αποτελέσματα της με την υπάρχουσα έρευνα πάνω στον αλγόριθμο αυτόν.

6.2 Μελλοντική Έρευνα

Λαμβάνοντας υπόψιν τους περιορισμούς της εργασίας μας, επιλέξαμε να επικεντρωθούμε, όπως αναφέραμε στους, decision tree-based αλγόριθμους και βασιστήκαμε κυρίως στην αναζήτηση πλέγματος. Θα μπορούσαμε σε μελλοντικό χρόνο να μελετήσουμε περισσότερο λεπτομερώς την επιλογή των παραμέτρων και να συνδυάσουμε τον XGBoost με κάποιον άλλον αλγόριθμο. Επίσης, θα μπορούσαμε να κατασκευάσουμε με περισσότερη προσπάθεια ένα νευρωνικό δίκτυο και να συγκρίνουμε τόσο τα αποτελέσματά του σε σχέση με τον XGBoost όσο και το κόστος εκπαίδευσης.

Κλείνοντας, θα θέλαμε να δοκιμάσουμε και μια μεθοδολογία που αναφέρεται στην έρευνα των Yin Wen Jun Wang, Tianyao Chen Weinan Zhang με τίτλο CAT2VEC: Learning Distributed Representation of Multi-Field Categorical Data. Αυτή η ερεύνα παρουσιάζει μια μέθοδο εκμάθησης κατανεμημένης αναπαράστασης για κατηγορηματικά δεδομένα πολλαπλών πεδίων. Η επιτυχία μη γραμμικών μοντέλων, όπως ενισχυμένων δέντρων, έχει αποδείξει τη δυνατότητα διερεύνησης των αλληλεπιδράσεων μεταξύ κατηγορικών πεδίων. Εμπνευσμένοι από το Word2Vec, την κατανεμημένη αναπαράσταση για τη φυσική γλώσσα, πρότειναν το μοντέλο Cat2Vec (κατηγορίες σε φορείς). Στο Cat2Vec, ένας συνεχές διάνυσμα χαμηλής διάστασης αποκτάται αυτόματα για κάθε κατηγορία σε κάθε πεδίο.

ΑΝΑΦΟΡΕΣ / ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Liaw, Andy & Wiener, Matthew. (2001). Classification and Regression by RandomForest. Forest. 23.

[2] Casanova, R., Saldana, S., Chew, E. Y., Danis, R. P., Greven, C. M., & Ambrosius, W. T. (2014). Application of random forests methods to diabetic retinopathy classification analyses.PloS one, 9(6), e98587. [https://doi.org/10.1371/journal.pone.0098587]

[3] Anand, R., Cherian, S. M., Ravi, S., Agarwal, V., Sampath, A., Shreenidhi, S., ... & Srinath, K. R. Statistical Analysis of Animal Adoption using R.

[4] Laber, E. S., & Pereira, F. D. A. M. (2018). Splitting criteria for classification problems with multi-valued attributes and large number of classes. Pattern Recognition Letters, 111, 58-63.

[5] Sigrist, F. (2018). Gradient and newton boosting for classification and regression. arXiv preprint arXiv:1808.03064.

[6] Scikit Learn machine learning library in python. [https://scikit-learn.org/stable/]

[7] XGBoost Documentation, an optimized gradient boosting library [https://XGBoost.readthedocs.io/en/latest/]

[8] R documentation [https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/importance]

[9] [https://pdfs.semanticscholar.org/b48c/8741eca1197f2f472af4a51438ae2524ba57.pdf]

[10] Richard O. Duda, Peter E. Hart, Pattern Classification, 2nd Edition 2001, ch. 8.

[11] S. Theodoridis & K. Koutroubas, Pattern Recognition, 4th Edition, 2003, ch. 4.

[12] Nielsen Didrik, 2006, Tree Boosting With XGBoost - Why Does XGBoost Win 'Every' Machine Learning Competition?

[13] Wikipedia, Cover's theorem. [https://en.wikipedia.org/wiki/Cover%27s_theorem]

[14] P. Probst, M. Wright and A. Boulesteix , 2018, Hyperparameters and Tuning Strategies for Random Forest.

[15] Yin Wen Jun Wang, Tianyao Chen Weinan Zhang, 2017, CAT2VEC: Learning Distributed Representation of Multi-Field Categorical Data.

[16] Eduardo Sany Laber, Felipe de A. Mello Pereira, 2018, Splitting Criteria for classification problems with multi-valued attributed and large number of classes.

[17] Deep Learning Tutorial by Stanford university Computer Science Department. [http://deeplearning.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/]

[18] Nielsen, D. (2016).Tree boosting with XGBoost-why does XGBoost win 'every' machine learning competition? Master's thesis, NTNU).

[19] A Short Introduction to Boosting Yoav Freund Robert E. Schapire (1999).