METROPOLITAN COLLEGE

CENTRE OF EXCELLENCE

Σύγκριση κατηγοριών αλγορίθμων ML - Άσκηση λήψης αποφάσεων σε επιλογή από ορισμένα dataset

Student ID: 2018506

Μάθημα: Advanced Decision Making

MSc Data Science
Σχολή Πληροφορικής
Μητροπολιτικό Κολλέγιο
Αθήνα, Ελλάδα

Ημερομηνία:
10/06/2020

Περιεχόμενα

Περίληψη

Η παρούσα εργασία έχει σκοπό να προσεγγίσει από μια θεωρητική σκοπιά δύο βασικές και ευρείες οικογένειες αλγορίθμων που χρησιμοποιούνται στο machine learning[1], και να υλοποιήσει δύο αλγόριθμους εποπτευόμενης μάθησης πάνω σε ένα επιλεγμένο dataset. Στο θεωρητικό κομμάτι τα δύο είδη αλγορίθμων που παρουσιάζονται είναι οι tree based αλγόριθμοι και οι αλγόριθμοι πιθανοτήτων υπό συνθήκη, και ύστερα από ανάλυση του τρόπου λειτουργίας τους και των πλεονεκτημάτων και περιορισμών που έχει η κάθε κατηγορία, γίνεται μια σύγκριση μεταξύ τους με σκοπό την εύρεση ομοιοτήτων και διαφορών. Στο πρακτικό κομμάτι, επιλέχθηκαν οι αλγόριθμοι Naive Bayes και Linear Regression, οι οποίοι θα εφαρμοστούν πάνω σε δεδομένα που περιέχουν τις επιδόσεις μαθητών από σχολεία της Πορτογαλίας, και μετά την υλοποίηση θα γίνει σύγκριση και αξιολόγηση των αποτελεσμάτων.

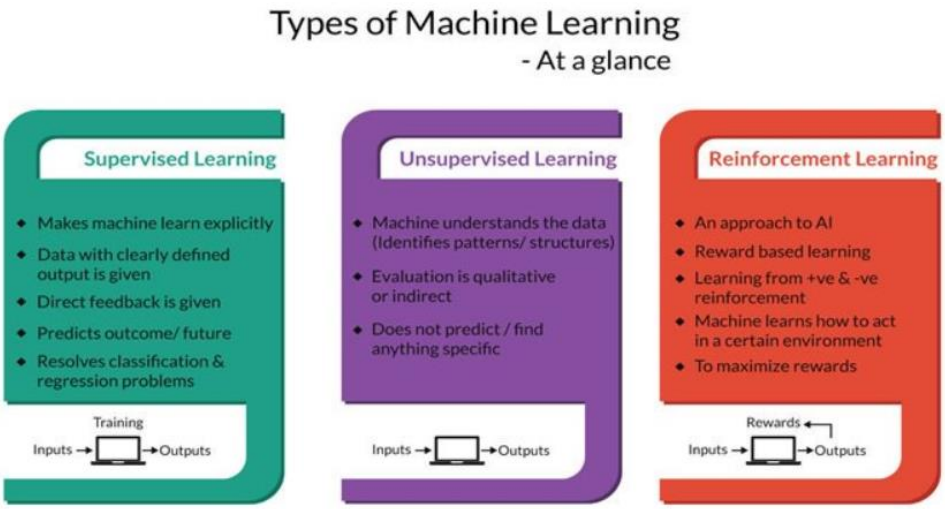
Εισαγωγή

Ιστορική αναδρομή

Σε μια σύντομη ιστορική αναδρομή, ο όρος “Machine Learning” ή αλλιώς μηχανική εκμάθηση χρησιμοποιήθηκε για πρώτη φορά το μακρινό 1952, όταν ο Arthur Samuel της IBM έφτιαξε ένα πρόγραμμα το οποίο μπορούσε να παίξει το γνωστό επιτραπέζιο παιχνίδι στρατηγικής checkers, ή αλλιώς ντάμα στα ελληνικά (Foote, 2019). 15 χρόνια αργότερα γεννήθηκε ο αλγόριθμος Nearest Neighbor, ο οποίος ήταν η αρχή του βασικού pattern recognition[2]. Μέχρι τις αρχές του 1980, το machine learning και το AI[3] είχαν ένα κοινό μονοπάτι, αλλά στη συνέχεια διαχωρίστηκαν και η επιστήμη της μηχανικής εκμάθησης διατήρησε τον προσανατολισμό που είχε στα Νευρωνικά Δίκτυα και ευδοκίμησε την δεκαετία του 1990, γεγονός που οφείλεται και στην ραγδαία αύξηση του Ίντερνετ. Το 2006 αναπτύχθηκαν οι πρώτοι αλγόριθμοι αναγνώρισης προσώπου, και το 2007 το μοντέλο νευρωνικού δικτύου LSTM[4] άρχισε να ξεπερνά σε απόδοση πιο παραδοσιακά μοντέλα αναγνώρισης φωνής. Το 2012, η ομάδα X Lab της Google ανέπτυξε έναν αλγόριθμο που μπορούσε αυτόνομα να περιηγηθεί και να βρει βίντεο που περιέχουν γάτες, και το 2014 η Facebook υλοποίησε το DeepFace, έναν προχωρημένο αλγόριθμο ο οποίος μπορούσε να αναγνωρίσει πρόσωπα σε φωτογραφίες με την ίδια ακρίβεια που θα το έκανε και ένας άνθρωπος. Για περισσότερες λεπτομέρειες αναφορικά με την ιστορία του machine learning, η Google έχει φτιάξει ένα εξαιρετικό διάγραμμα το οποίο περιέχει πληροφορίες από 60 πηγές (https://cloud.withgoogle.com/build/data-analytics/explore-history-machine-learning/).

[1] Machine learning ορίζεται ως η μελέτη και η ανάπτυξη υπολογιστικών αλγορίθμων οι οποίοι βελτιώνονται αυτόματα μέσω των επαναλήψεων και της εμπειρίας (Mitchell & Hill, 1997).
[2] Pattern recognition είναι η αυτόματη αναγνώριση μοτίβων και κανονικοτήτων σε ένα σύνολο δεδομένων (Wikipedia, 2020)
[3] AI: Τεχνητή νοημοσύνη, είναι η προσομοίωση της ανθρώπινης νοημοσύνης σε μηχανές (υπολογιστές) οι οποίες είναι προγραμματισμένες να σκέφτονται και να δρουν σαν ανθρώπινα όντα (Frankenfield, 2020)
[4] Long Short-Term Memory, αρχιτεκτονική επαναλαμβανόμενου νευρωνικού δικτύου (recurrent neural network), που χρησιμοποιείται στον τομέα του deep learning (Hochreiter & Schmidhuber, 1997)

Κατηγορίες αλγορίθμων μηχανικής εκμάθησης



Εικόνα 1 - Βασικές κατηγορίες αλγορίθμων

Κάτω από την ομπρέλα της μηχανικής εκμάθησης (machine learning), υπάρχουν αρκετοί αλγόριθμοι οι οποίοι έχουν σκοπό να προβλέψουν, να αναλύσουν, να ταξινομήσουν και να κατηγοριοποιήσουν δεδομένα. Αυτοί οι αλγόριθμοι χωρίζονται σε τρεις βασικές κατηγορίες, τις οποίες αναλύει σε ηλεκτρονικό επιστημονικό άρθρο ο Data Scientist David Fumo (Fumo, 2017):

- **Supervised:** Οι αλγόριθμοι εποπτευόμενης μάθησης χρησιμοποιούν δεδομένα με ετικέτα (labeled data[5]) και προσπαθούν να βρουν τη συσχέτιση ανάμεσα στα δεδομένα εισόδου και σε ένα ή περισσότερα δεδομένα εξόδου.
- **Unsupervised:** Οι αλγόριθμοι μη εποπτευόμενης μάθησης χρησιμοποιούν δεδομένα χωρίς ετικέτα, και "χρησιμοποιούνται σε περιπτώσεις όπου ο αναλυτής δε γνωρίζει ακριβώς τη δομή των δεδομένων και δε ξέρει τι ακριβώς πρέπει να κοιτάξει μέσα στο dataset" (Fumo, 2017)
- **Reinforcement:** Οι αλγόριθμοι ενισχυόμενης μάθησης εκπαιδεύουν συνεχώς τον εαυτό τους μέσω της trial and error διαδικασίας, και δρουν με σκοπό την μεγιστοποίηση της ανταμοιβής (πχ high score σε ένα video game) σε κάθε επανάληψη.

Υπάρχει και η μικρότερη κατηγορία των **Semi-supervised** αλγορίθμων, οι οποίοι είναι ένα κράμα εποπτευόμενης και μη εποπτευόμενης μάθησης. Επειδή το κόστος του να υπάρχει ετικέτα σε όλα τα δεδομένα είναι υψηλό, πολλές φορές χρησιμοποιείται ο ανθρώπινος παράγοντας για να μπουν ετικέτες μόνο σε κάποια από τα δεδομένα, δημιουργώντας έτσι ένα ανάμεικτο dataset από labeled και unlabeled data.

Μεθοδολογία

Η μεθοδολογία που χρησιμοποιήθηκε για την συγγραφή του θεωρητικού μέρους της παρούσας εργασίας ήταν ενδελεχής έρευνα βασισμένη σε έμπιστες πηγές του διαδικτύου, άρθρα και δημοσιεύσεις σχετικές με αλγορίθμους και machine learning, η οποία σε συνδυασμό με τις ήδη υπάρχουσες γνώσεις του συγγραφέα από τον χώρο της πληροφορικής διαμόρφωσαν το τελικό αποτέλεσμα. Οι συγγραφείς των άρθρων που χρησιμοποιήθηκαν ως πηγές είναι στην πλειοψηφία επαγγελματίες data scientists οι οποίοι έχουν εμπειρία στην ανάλυση δεδομένων και στην υλοποίηση των αλγορίθμων που αναλύονται παρακάτω. Υποστηρίζεται ακράδαντα η άποψη που λέει ότι στην σύγχρονη εποχή, οποιοσδήποτε έχει πρόσβαση στο διαδίκτυο και γνωρίζει πώς να κάνει σωστή έρευνα μπορεί να βρει πληροφορίες για οποιοδήποτε θέμα, επιστημονικό ή μη, αλλά και να διασταυρώσει αυτές τις πληροφορίες βασιζόμενος σε επιστημονικές μελέτες και ακαδημαϊκές δημοσιεύσεις.

[5] Ομάδα δειγμάτων που έχουν χαρακτηρισθεί με μια ετικέτα. Με πιο απλά λόγια, τα ονόματα των στηλών σε ένα δισδιάστατο πίνακα δεδομένων.

Αναφορικά με το πρακτικό κομμάτι της εργασίας, που περιλαμβάνει την υλοποίηση δύο αλγορίθμων μηχανικής εκμάθησης πάνω σε ένα σύνολο δεδομένων της επιλογής μας, ακολουθήθηκαν τα παραδείγματα που διδάχτηκαν στο μάθημα Advanced Decision Making, αλλά στο μεγαλύτερο μέρος έγινε μελέτη στο επίσημο documentation[6] της python, καθώς και στις κύριες βιβλιοθήκες[7] της python που χρησιμοποιήθηκαν για την ανάλυση του dataset. Στο προγραμματιστικό σκέλος, χρησιμοποιήθηκαν τεχνικές debugging (με χρήση του ενσωματωμένου debugger που παρέχει το Visual Studio Code για την python) και Jupyter notebooks όπως ακριβώς υποδείχθηκε και στις παραδόσεις του μαθήματος. Όλες οι υλοποιήσεις/δοκιμές πραγματοποιήθηκαν σε υπολογιστικό σύστημα με τις εξής προδιαγραφές:

- Λειτουργικό σύστημα: Windows 10 64-bit
- Επεξεργαστής: Intel Core i7-8550U @ 1.80Ghz
- RAM: 8,00 GB

Κάθε Jupyter notebook που υλοποιήθηκε, μετατράπηκε στη συνέχεια σε Python script και χρονομετρήθηκε, προκειμένου να αξιολογηθεί και χρονικά ο κάθε αλγόριθμος.

Αλγόριθμοι

Tree based algorithms

Εικόνα 2 - Απλό Δέντρο Αποφάσεων που καθορίζει αν θα πρέπει να δοθεί δάνειο σε κάποιον βάσει σχετικών μεταβλητών.

Ορισμός και βασικές πληροφορίες

Οι αλγόριθμοι που βασίζονται σε δέντρα αποφάσεων αποτελούν ένα από τα μοντέλα πρόβλεψης που χρησιμοποιούνται στη στατιστική, το data mining[8] και το machine learning. Τα δέντρα αποφάσεων χωρίζονται σε 2 κύριες κατηγορίες, αναλόγως του τύπου της προβλεπόμενης μεταβλητής:

[6] Python 3 official documentation: https://docs.python.org/3/
[7] Pandas official documentation: https://pandas.pydata.org/docs/
Scikit-learn official documentation: https://scikit-learn.org/stable/
Matplotlib official documentation: https://matplotlib.org/3.2.1/index.html
NumPy official documentation: https://numpy.org/doc/1.18/user/index.html
[8] "Data mining είναι η διαδικασία εύρεσης ανωμαλιών, συσχετίσεων και μοτίβων σε μεγάλα σύνολα δεδομένων με σκοπό την πρόβλεψη αποτελεσμάτων" (SAS, 2020)

- Δέντρα ταξινόμησης, τα οποία καλούνται να προβλέψουν μια διακριτή μεταβλητή ως αποτέλεσμα.
- Δέντρα παλινδρόμησης, όπου η πρόβλεψη πρέπει να γίνει σε συνεχή μεταβλητή η οποία μπορεί να πάρει πραγματικές τιμές (πχ χρονική διάρκεια, τιμή, θερμοκρασία κλπ.)

Ο όρος Classification And Regression Tree (CART) χρησιμοποιείται για να αναφερθεί και στις 2 παραπάνω κατηγορίες (Breiman, Friedman, Olshen, & Stone, 1984). Και τα 2 είδη έχουν κάποιες ομοιότητες όσον αφορά στον τρόπο λειτουργίας τους, αλλά και κάποιες διαφορές, όπως για παράδειγμα την διαδικασία που χρησιμοποιείται για να καθοριστεί το που θα γίνει ο διαχωρισμός στις τιμές. "Η κύρια διαφορά ανάμεσα στα ανάμεσα στα δέντρα ταξινόμησης και στα δέντρα παλινδρόμησης, είναι ότι τα πρώτα δημιουργούνται με μη ταξινομημένες εξαρτώμενες μεταβλητές, ενώ τα δεύτερα παίρνουν ταξινομημένες μεταβλητές με συνεχείς τιμές" (Pulipaka, 2016).

Περιπτώσεις που χρησιμοποιούνται δέντρα αποφάσεων

Οι κατηγορίες προβλημάτων στα οποία χρησιμοποιούνται δέντρα αποφάσεων έχουν τα ακόλουθα κοινά χαρακτηριστικά (Teggi, 2020):

- Οι εγγραφές αναπαρίστανται από ζευγάρια χαρακτηριστικών-τιμών. Περιγράφονται από ένα συγκεκριμένο σύνολο χαρακτηριστικών (πχ Θερμοκρασία) και των τιμών τους (πχ ζεστό, χλιαρό, κρύο). Η ευκολότερη κατάσταση για ένα δέντρο αποφάσεων είναι όταν οι τιμές παίρνουν λίγες και διακριτές τιμές (όπως στο προηγούμενο παράδειγμα), αλλά με τις κατάλληλες μορφοποιήσεις ο αλγόριθμος μπορεί να διαχειριστεί και πραγματικές τιμές (στην προκειμένη περίπτωση την τιμή της θερμοκρασίας σε μία από τις γνωστές κλίμακες, πχ βαθμοί Κελσίου).
- Το χαρακτηριστικό που μελετά ο αλγόριθμος, και για το οποίο πρέπει να γίνει πρόβλεψη, έχει διακριτές τιμές. Σε ιδανικές συνθήκες παίρνει τις τιμές 0 και 1, που σημαίνει εκφράζει την πραγματοποίηση ή μη του ενδεχομένου. Και σε αυτήν την περίπτωση, με τις κατάλληλες επεκτάσεις και τροποποιήσεις ο αλγόριθμος μπορεί να υποστηρίξει και ενδεχόμενα με συνεχείς τιμές.
- Υπάρχουν διαχωρισμοί που χωρίζουν το dataset σε κατηγορίες. Τα δέντρα αποφάσεων λειτουργούν με αυτόν τον τρόπο και απεικονίζουν διαχωριστικές εκφράσεις, ακόμα και σε συνεχείς τιμές (πχ Ηλικία > 40, Ηλικία < 40).
- Τα δεδομένα εκμάθησης (training dataset) του αλγορίθμου μπορεί να περιέχουν λάθη. Οι μέθοδοι εκμάθησης των δέντρων αποφάσεων είναι ισχυρές απέναντι σε σφάλματα τόσο στις διακριτές τιμές των χαρακτηριστικών όσο και στην περιγραφή των ίδιων των attributes.
- Τα δεδομένα εκμάθησης μπορεί να περιέχουν ελλιπείς τιμές. Σε αυτήν την περίπτωση τα δέντρα αποφάσεων μπορούν να εκπαιδευτούν με μεγαλύτερη ακρίβεια και αποτελεσματικότητα σε σχέση με άλλες κατηγορίες αλγορίθμων.

Τρόπος λειτουργίας

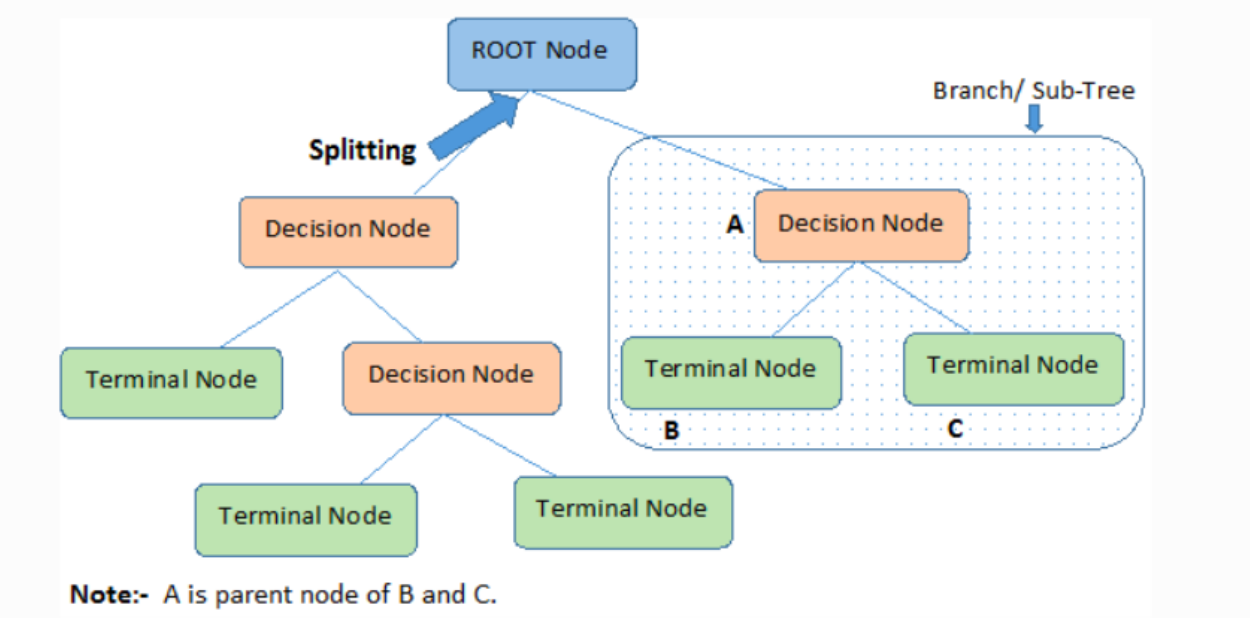
Σε επιστημονικό άρθρο (Chauhan, 2019) που έχει γραφτεί από τον Data Scientist Nagesh Singh Chauhan, επεξηγείται αναλυτικά ο τρόπος λειτουργίας των Δέντρων Αποφάσεων, και αναλύονται βασικές ορολογίες που βοηθούν στην καλύτερη κατανόηση των κανόνων βάσει των οποίων δρουν οι συγκεκριμένοι αλγόριθμοι.

Για την καλύτερη αντίληψη σχετικά με τα decision trees πρέπει να γίνουν κατανοητοί οι όροι που χρησιμοποιούνται κατά κόρον:

1. **Ριζικός κόμβος**: Ο αρχικός κόμβος από τον οποίο ξεκινάει όλο το dataset, στον οποίο προφανώς περιέχονται όλα τα δεδομένα αφού δεν έχει γίνει ακόμα κάποιος διαχωρισμός.
2. **Διαχωρισμός**: Η διαδικασία κατά την οποία ένας κόμβος διαχωρίζεται σε δύο ή περισσότερους υποκόμβους.

3. **Κόμβος απόφασης**: Κόμβος εκτός του ριζικού, ο οποίος διαχωρίζεται περαιτέρω σε επόμενα βήματα εκτέλεσης του αλγορίθμου.
4. **Φύλλο/Τερματικός κόμβος**: Τελικός κόμβος ο οποίος δεν διαχωρίζεται περαιτέρω.
5. **Κλάδεμα**: Το αντίθετο του διαχωρισμού, η ελάττωση των κόμβων που γίνεται μέσω της αφαίρεσης υποκόμβων που ανήκουν σε ένα κόμβο απόφασης. Η συγκεκριμένη ενέργεια γίνεται για να επιτευχθεί μείωση της πολυπλοκότητας του αλγορίθμου έτσι ώστε να μπορούν να εξαχθούν συμπεράσματα με μεγαλύτερη ευκολία.
6. **Κλαδί/Υποδέντρο:** Μια υποενότητα του δέντρου αποφάσεων.
7. **Γονέας/Παιδί:** Ένας κόμβος απόφασης που περιέχει υποκόμβους αποκαλείται γονέας, και οι υποκόμβοι αποκαλούνται παιδιά.

Στην παρακάτω εικόνα είναι εμφανείς όλοι οι προαναφερθέντες όροι εκτός από την ενέργεια του κλαδέματος (pruning).



Εικόνα 3 - Παράδειγμα δέντρου αποφάσεων

"Οι αποφάσεις που καθορίζουν τα σημεία στα οποία θα γίνει διαχωρισμός επηρεάζουν σε μεγάλο βαθμό την ακρίβεια και την απόδοση ενός δέντρου αποφάσεων" (Chauhan, 2019). Χρησιμοποιούνται διάφοροι αλγόριθμοι οι οποίοι αποφασίζουν πότε χρειάζεται διαχωρισμός και εάν ένας κόμβος πρέπει να διαχωριστεί σε δύο ή περισσότερους υποκόμβους. Οι κυριότεροι εξ' αυτών είναι οι παρακάτω:

- **ID3**: επέκταση του D3
- **C4.5**: διάδοχος του ID3
- **CART**: δέντρο ταξινόμησης και παλινδρόμησης
- **CHAID**: κάνει αυτόματη ανίχνευση της τιμής chi-square[9] και πραγματοποιεί διαχωρισμούς πολλών επιπέδων κατά τον υπολογισμός δέντρων ταξινόμησης
- **MARS**: multivariate adaptive regression splines. Μέθοδος παλινδρομικής ανάλυσης που εφευρέθη από τον Jerome H. Friedman το 1991 (Friedman, 1991)

Ένας εκ των παραπάνω αλγορίθμων, ο ID3 είναι ένας άπληστος αλγόριθμος που δημιουργεί δέντρα αποφάσεων χρησιμοποιώντας την καλύτερη λύση για κάθε χρονική στιγμή χωρίς να ανατρέχει προς τα πίσω. Τα βήματα που ακολουθούνται είναι τα εξής:

1. Επιλογή αρχικού κόμβου S που περιλαμβάνει το σύνολο δεδομένων

[9] Chi-square: $x^2 = \sum \frac{(O-E)^2}{E}$, όπου O είναι τα observed frequencies (οι φορές που ένα ενδεχόμενο πραγματικά συνέβη) και E είναι τα expected frequencies (οι φορές που ένα ενδεχόμενο αναμένεται να συμβεί)

2. Σε κάθε επανάληψη του αλγορίθμου, παρατηρείται το πιο άσημο attribute του dataset και υπολογίζονται οι τιμές Entropy (H)[10] και Information Gain (IG)[11].
3. Στη συνέχεια επιλέγεται το χαρακτηριστικό που έχει το μικρότερο Entropy ή το μεγαλύτερο Information Gain.
4. Ο αρχικός κόμβος S διαχωρίζεται βάσει του επιλεγμένου χαρακτηριστικού και παράγει δύο ή περισσότερους υποκόμβους.
5. Ο αλγόριθμος συνεχίζει με αναδρομή[12] σε κάθε υποκόμβο, λαμβάνοντας υπόψιν μόνο attributes που δεν έχουν χρησιμοποιηθεί σε προηγούμενες επαναλήψεις.

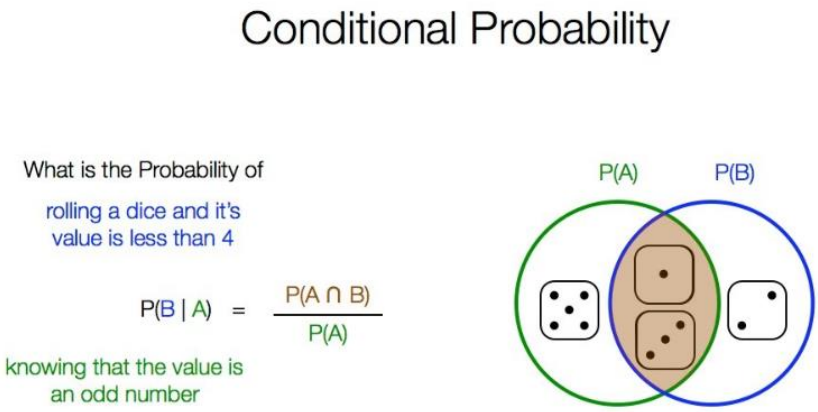
Πλεονεκτήματα και περιορισμοί

Σύμφωνα με τον Data Scientist Dhiraj K., τα Δέντρα Αποφάσεων έχουν ορισμένα πλεονεκτήματα που τα καθιστούν κατάλληλα σε ορισμένους τύπους προβλημάτων. Έχουν όμως, όπως και όλοι οι αλγόριθμοι άλλωστε, και συγκεκριμένους περιορισμούς, οι οποίοι πρέπει να λαμβάνονται σοβαρά υπόψιν προκειμένου να μην χρησιμοποιούνται σε λανθασμένα use cases[13] (Dhiraj, 2019).

Τα κυριότερα πλεονεκτήματα των tree based αλγορίθμων έχουν να κάνουν με το γεγονός ότι δεν χρειάζεται μεγάλη προσπάθεια και χρόνος στην προ-επεξεργασία των δεδομένων. Πιο συγκεκριμένα, δεν είναι απαραίτητα βήματα ούτε η κανονικοποίηση (normalization) ούτε η προτυποποίηση (standardization/scaling) των δεδομένων. Επιπροσθέτως, ένα ακόμη σημαντικό στοιχείο είναι ότι οι ελλιπείς τιμές στα δεδομένα (missing data) δεν επηρεάζουν σε μεγάλο βαθμό την τελική έκβαση και τον τρόπο με τον οποίο θα δημιουργηθεί το δέντρο. Τέλος, μεγάλη αξία (business value) τους προσδίδει το γεγονός ότι είναι το ίδιο εύκολο να επεξηγηθούν τόσο σε μια τεχνική ομάδα όσο και σε ένα διοικητικό board από stakeholders που ενδιαφέρονται μόνο για οπτικοποιημένα αποτελέσματα.

Στην αντίπερα όχθη, τα Δέντρα Αποφάσεων παρουσιάζουν και κάποιους περιορισμούς. Μια μικρή αλλαγή στο dataset στο οποίο εφαρμόζεται ένας tree based algorithm, μπορεί να προκαλέσει μεγάλη αλλαγή στη δομή του Δέντρου προκαλώντας αστάθεια. Αυτό κατά την προσωπική άποψη του συγγραφέα είναι ένα αρκετά σημαντικό γεγονός, καθώς στην καθημερινή εργασία ενός data scientist τα δεδομένα τα οποία μελετά μπορούν να υποστούν πολλές αλλαγές στο πέρασμα του χρόνου. Επίσης, οι υπολογισμοί που κάνει ένα δέντρο αποφάσεων πολλές φορές μπορεί να αποδειχθούν ιδιαίτερα πολύπλοκοι σε σχέση με άλλους τύπους αλγορίθμων, αυξάνοντας έτσι τον χρόνο εκτέλεσης της ανάλυσης. Άλλο ένα βήμα που αυξάνει τον συνολικό χρόνο εκτέλεσης είναι η εκμάθηση (training) του αλγορίθμου, η οποία είναι και αυτή σχετικά "ακριβή" (expensive). Τέλος, κάποιοι από τους tree based αλγορίθμους κρίνονται ακατάλληλοι για να εφαρμόσουν παλινδρόμηση και να προβλέψουν αποτελέσματα που περιλαμβάνουν συνεχείς και όχι διακριτές τιμές.

[10] Entropy είναι μια μεταβλητή που δείχνει το ποσοστό τυχαιότητας στα δεδομένα που είναι υπό επεξεργασία. Όσο μεγαλύτερη η τιμή της τόσο δυσκολότερο είναι να βγουν συμπεράσματα για αυτά τα δεδομένα. (Sujan, 2018)
[11] Information gain είναι μια στατιστική ιδιότητα που υπολογίζει πόσο αποτελεσματικά ένα δεδομένο χαρακτηριστικό (attribute) διαχωρίζει το training dataset ανάλογα με τον στόχο που έχει καθοριστεί. (Sujan, 2018)
[12] Αναδρομή στην Πληροφορική είναι η διαδικασία κατά την οποία μια μέθοδος καλεί τον εαυτό της συνεχώς και σπάει ένα αρχικό πρόβλημα σε πολλά μικρότερα προβλήματα βρίσκοντας την λύση με μια σχετική ευκολία.
[13] "Ένα use case είναι μια λίστα βημάτων που καθορίζουν την αλληλεπίδραση ανάμεσα σε ένα ρόλο και σε ένα σύστημα με σκοπό την επίτευξη ενός στόχου" (Wikipedia, 2020). Με πιο απλά λόγια είναι ή περίπτωση στην οποία χρησιμοποιείται ένα εργαλείο για έναν συγκεκριμένο σκοπό.



Εικόνα 4 - Πιθανότητα υπό συνθήκη

Ορισμός και βασικές πληροφορίες

Για να μελετηθούν οι αλγόριθμοι που έχουν ως βάση τις πιθανότητες υπό συνθήκη, θα πρέπει να γίνει κατανοητός ο μαθηματικός ορισμός που δίνεται για τον συγκεκριμένο όρο. "Στην θεωρία των πιθανοτήτων, η πιθανότητα υπό συνθήκη είναι ο υπολογισμός της πιθανότητας πραγματοποίησης ενός συμβάντος με δεδομένο ότι ένα άλλο συμβάν έχει πραγματοποιηθεί (είτε από υπόθεση, είτε αποδεδειγμένα) " (Barone, 2020). Με άλλα λόγια, για δύο ενδεχόμενα Α και Β, η πιθανότητα υπό συνθήκη για το Β ορίζεται ως η πιθανότητα της τομής των δύο ενδεχομένων διά την πιθανότητα πραγματοποίησης του Α[14].

Απλό παράδειγμα υπολογισμού πιθανοτήτων υπό συνθήκη: Έστω ένας μαθητής που κάνει αίτηση για να σπουδάσει σε ένα πανεπιστήμιο, και ελπίζει να πάρει και υποτροφία. Το συγκεκριμένο πανεπιστήμιο έχει ως πολιτική να δέχεται 100 υποψηφίους ανά 1000 αιτήσεις, δηλαδή ένα ποσοστό 10% (P(A)). Ανά 500 υποψηφίους που δέχεται, παρέχει δωρεάν υποτροφία στους 10 εξ' αυτών (δηλαδή σε ένα ποσοστό 2% (P(B))). Η γενναιοδωρία του πανεπιστημίου δε σταματά εκεί, καθώς παρέχει δωρεάν στέγαση, διατροφή και βιβλία στο 50% (P(C)) των υποτρόφων. Σύμφωνα με τα παραπάνω δεδομένα λοιπόν, η πιθανότητα για ένα φοιτητή να γίνει δεκτός στο πανεπιστήμιο και στη συνέχεια να λάβει υποτροφία είναι η εξής:
P(B|A) = 0.1*0.02 = 0.002 = 0.2%
Με το ίδιο σκεπτικό, η πιθανότητα ο φοιτητής να γίνει αποδέκτης και των δωρεάν παροχών του πανεπιστημίου είναι 0.1*0.02*0.5 = 0.1%, δηλαδή μόλις 1 στους 1000 υποψηφίους γίνεται δεκτός και απολαμβάνει όλα τα προνόμια.

Οι αλγόριθμοι πιθανοτήτων υπό συνθήκη βασίζονται πάνω σε μια θεμελιώδη μαθηματική αρχή των πιθανοτήτων: το θεώρημα του Bayes. Ονομάστηκε έτσι από τον Βρετανό μαθηματικό του 18ου αιώνα Thomas Bayes, και είναι μια μαθηματική φόρμουλα που χρησιμοποιείται για τον υπολογισμό πιθανοτήτων υπό συνθήκη. Ο τύπος της είναι ο εξής:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Εικόνα 5 - Θεώρημα Bayes

Με λόγια περιγράφεται ως η υπό συνθήκη πιθανότητα ενός ενδεχομένου Α δεδομένου ότι έχει συμβεί το Β, η οποία ισούται με το γινόμενο της υπό συνθήκη πιθανότητας του Β δεδομένου ότι έχει συμβεί το Α επί την πιθανότητα του Α, διαιρεμένο με την πιθανότητα του Β. Οι αλγόριθμοι που βασίζονται στο θεώρημα του Bayes γνωρίζουν πολλές εφαρμογές, με μια εξ' αυτών να είναι

[14] P(B|A) = P(A∩B) / P(A)

στον χώρο των οικονομικών και στον υπολογισμό του ρίσκου που υπάρχει από τις τράπεζες όταν καλούνται να δώσουν δάνειο σε έναν πιθανό δανειολήπτη (Hayes, 2020). Στην μηχανική εκμάθηση, χρησιμοποιούνται κατά κόρον Naive Bayes αλγόριθμοι που έχουν ως βασική αρχή, όπως λέει και το όνομα τους, το προαναφερθέν θεώρημα.

Τύποι ταξινομητών Naive Bayes

Σε ηλεκτρονικό επιστημονικό άρθρο που έχει γνωρίσει μεγάλη απήχηση λόγω της καλής επεξήγησης που παρέχει (Gandhi, 2018), αναλύονται με σαφήνεια οι 3 τύποι ταξινομητών Naive Bayes που χρησιμοποιούνται στο Machine Learning:

- **Gaussian Naive Bayes:** Χρησιμοποιείται για attributes των οποίων οι τιμές είναι συνεχείς και πραγματικές. Λαμβάνεται η υπόθεση ότι τα δείγματα ακολουθούν κανονική κατανομή[15].
- **Multinomial Naive Bayes:** Η κύρια χρήση του είναι για προβλήματα ταξινόμησης εγγράφων, όπως για παράδειγμα η απόδοση του είδους σε ένα έγγραφο ανάλογα με τις λέξεις τις οποίες περιέχει. Τα attributes πάνω στα οποία γίνεται η ανάλυση/πρόβλεψη είναι οι ίδιες οι λέξεις του κειμένου.
- **Bernoulli Naive Bayes:** Ο αλγόριθμος αυτός είναι όμοιος με τον προηγούμενο, αλλά οι παράμετροι που χρησιμοποιούνται για την πρόβλεψη παίρνουν μόνο δύο τιμές, που αναφέρονται σε πραγματοποίηση ή μη (true/false) ενός χαρακτηριστικού, ή μιας λέξης στο ανωτέρω παράδειγμα.

Περιπτώσεις που χρησιμοποιούνται Naive Bayes Αλγόριθμοι

Σύμφωνα με το ίδιο άρθρο, οι αλγόριθμοι Naive Bayes χρησιμοποιούνται κατά κύριο λόγο σε 4 είδη εφαρμογών:

- **Πρόβλεψη σε πραγματικό χρόνο.** Οι Naive Bayes εκπαιδεύονται γρήγορα και παράγουν αποτελέσματα σε μικρό χρόνο, και για αυτόν τον λόγο χρησιμοποιούνται σε εφαρμογές όπου είναι αναγκαία η άμεση ανατροφοδότηση και αναπροσαρμογή σε νέα δεδομένα.
- **Πρόβλεψη σε πολλαπλά χαρακτηριστικά.**
- **Ταξινόμηση κειμένου**. Οι Naive Bayes αλγόριθμοι λόγω της αφέλειας τους τα καταφέρνουν περίφημα σε επεξεργασία κειμένου, και γι' αυτό χρησιμοποιούνται κατά κόρον για spam filtering[16] καθώς και για sentiment analysis[17].
- **Συστήματα προτάσεων (recommendation systems).** Οι ταξινομητές Naive Bayes σε συνδυασμό με την τεχνική του collaborative filtering[18] χτίζουν ένα δυνατό recommendation system το οποίο μέσω του machine learning και του data mining φιλτράρει πληροφορίες και προβλέπει την πιθανότητα μια συγκεκριμένη πηγή να αρέσει σε έναν χρήστη.

Πλεονεκτήματα και περιορισμοί

Στην συγκεκριμένη ενότητα θα παρουσιαστούν πλεονεκτήματα και περιορισμοί για τους αλγορίθμους πιθανοτήτων υπό συνθήκη. Σε γενικές γραμμές η υλοποίηση αυτών των αλγορίθμων και πιο συγκεκριμένα των Naive Bayes, στην μηχανική εκμάθηση, είναι αρκετά απλή και γι' αυτό το λόγο έχουν γνωρίσει μεγάλη απήχηση και υιοθετούνται αρκετά συχνά από data scientists.

---

[15] Κανονική κατανομή είναι βασική αρχή της στατιστικής και αναφέρεται σε δείγματα πραγματικών τιμών τα οποία τείνουν να συγκεντρώνονται γύρω από μια μέση τιμή.

[16] Τεχνική που χρησιμοποιείται από email providers η οποία αναγνωρίζει την ανεπιθύμητη αλληλογραφία βάσει του περιεχομένου της.

[17] Τεχνική που χρησιμοποιείται σε κοινωνικά δίκτυα και αναγνωρίζει το συναίσθημα (θετικό, αρνητικό) που κρύβεται πίσω από μια πρόταση/παράγραφο ενός χρήστη.

[18] Μέθοδος που χρησιμοποιείται από συστήματα προτάσεων για να προβλέψει τις προτιμήσεις ενός χρήστη.

Άλλα πλεονεκτήματα των ταξινομητών που βασίζονται σε πιθανολογικούς αλγορίθμους έχουν να κάνουν με τον μικρό χρόνο εκπαίδευσης του μοντέλου, όπως και την καλή προσαρμογή που παρουσιάζουν σε αλλαγές του dataset (όταν για παράδειγμα εμφανίζονται νέα data points στα δεδομένα) (Catanzarite, 2018). Καλές επιδόσεις παρατηρούνται επίσης και στους υπολογιστικούς πόρους που καταναλώνουν (RAM, CPU), καθώς σε κάθε βήμα δε φορτώνεται όλο το dataset στη μνήμη του συστήματος προκειμένου να υπολογιστούν οι πιθανότητες και να βγουν προβλέψεις. Ένα ακόμη στοιχείο που δείχνει την καλή προσαρμοστικότητα τους είναι ότι η πολυπλοκότητα τους αυξάνεται γραμμικά σε αναλογία με το μέγεθος των δεδομένων (αριθμός εγγραφών και χαρακτηριστικών, ή αλλιώς γραμμών και στηλών), γεγονός που τους καθιστά κατάλληλους σε πολύπλοκα datasets. Τέλος, όπως τα Δέντρα Αποφάσεων έτσι και αυτοί οι αλγόριθμοι αποδίδουν καλά με ελλιπή δεδομένα υπολογίζοντας μέσους όρους ή επιλέγοντας να αγνοήσουν εντελώς τα attributes που παρουσιάζουν κενά.

Οι περιορισμοί που παρουσιάζει η ανωτέρω οικογένεια πιθανολογικών αλγορίθμων υπό συνθήκη είναι απόρροια του τρόπου λειτουργίας τους και σχετίζεται άμεσα και με κάποια από τα πλεονεκτήματα τους. Για παράδειγμα, το γεγονός ότι για κάθε πρόβλεψη δεν λαμβάνεται υπόψιν ολόκληρο το dataset (για να υπάρχει καλύτερη απόδοση σε πόρους και χρόνο), πολλές φορές έχει ως αποτέλεσμα η ακρίβεια του μοντέλου να είναι χαμηλότερη σε σχέση με άλλους αλγορίθμους. Η ακρίβεια επηρεάζεται επίσης και από τις υποθέσεις που γίνονται για τα δεδομένα που περνάνε από ανάλυση - πιο συγκεκριμένα η υπόθεση ανεξαρτησίας για τα attributes και η υπόθεση ότι τα δεδομένα με συνεχείς τιμές ακολουθούν κανονική κατανομή.

Σύγκριση των 2 κατηγοριών

Μετά την ανάλυση των δύο κατηγοριών, παρατηρείται ότι παρουσιάζουν κάποιες ομοιότητες και ορισμένες διαφορές, που καθιστούν την κάθε κατηγορία κατάλληλη για συγκεκριμένο σκοπό. Παρ’ όλα αυτά, και οι δύο οικογένειες αλγορίθμων αποτελούν θεμέλια της μηχανικής εκμάθησης πάνω στα οποία μπορεί να χτιστεί ένα πολύπλοκο μοντέλο το οποίο να αναλύει δεδομένα και να προβλέπει ενδεχόμενα με μεγάλη ακρίβεια και αποτελεσματικότητα.

Μια κύρια ομοιότητα που παρουσιάζεται ανάμεσα στα δύο είδη, είναι η ευκολία υλοποίησης τους, καθώς υπάρχουν έτοιμες βιβλιοθήκες σε διάσημες γλώσσες προγραμματισμού (Python, R) οι οποίες κρύβουν τις πολύπλοκες μαθηματικές συναρτήσεις των αλγορίθμων και προσφέρουν βιβλιοθήκες που υλοποιούν Δέντρα Αποφάσεων και Πιθανολογικούς αλγορίθμους υπό συνθήκη, σε λίγες μόλις γραμμές κώδικα. Ένας ακόμη τομέας στον οποίο μοιάζουν, είναι η ικανότητα των αλγορίθμων να προσαρμόζονται στα ελλιπή δεδομένα και να μην επηρεάζεται σε μεγάλο βαθμό η απόδοση τους από αυτά. Τέλος, και οι δύο κατηγορίες αλγορίθμων τα καταφέρνουν καλύτερα σε δεδομένα με διακριτές τιμές, καθώς στις συνεχείς χρειάζονται περαιτέρω βήματα βελτιώσεων και τροποποιήσεων προκειμένου να επιτευχθεί το καλύτερο δυνατό αποτέλεσμα.

Οι διαφορές που παρουσιάζονται παρ’ όλα αυτά, είναι περισσότερες. Για αρχή, οι αλγόριθμοι πιθανοτήτων υπό συνθήκη, όπως αναφέρθηκε και παραπάνω, χρειάζονται πολύ λίγο χρόνο να εκπαιδευτούν και προτιμώνται για προβλέψεις πραγματικού χρόνου. Τα Δέντρα Αποφάσεων από την άλλη, απαιτούν περισσότερους πόρους και δεδομένα εκμάθησης (training data) προκειμένου να μπορέσουν να αποδώσουν σε ικανοποιητικό βαθμό. Επιπροσθέτως, αναφέρθηκε ότι τα Δέντρα Αποφάσεων δεν απαιτούν κάποια ιδιαίτερη προ-επεξεργασία των δεδομένων (κανονικοποίηση, προτυποποίηση), γεγονός το οποίο δεν ισχύει για όλους τους υπόλοιπους αλγορίθμους (και κατ’ επέκταση τους πιθανολογικούς). Επίσης, από την οπτική γωνία του Business, τα Δέντρα αποφάσεων παρουσιάζουν μεγαλύτερη ευκολία στην επεξήγηση και στην κατανόηση τους, καθώς η οπτικοποίηση τους είναι αρκετά επεξηγηματική. Τα μαθηματικά και οι πιθανότητες που υπάρχουν στους conditional probability αλγορίθμους, δε βοηθούν στον συγκεκριμένο τομέα. Μια ακόμη διαφορά που παρατηρείται, είναι ότι στους αλγορίθμους πιθανοτήτων υπό συνθήκη υπάρχει καλύτερη απόδοση σε μεγάλο αριθμό χαρακτηριστικών, ενώ οι tree based αλγορίθμους μπορεί να μην αποδώσουν το ίδιο καλά καθώς

θα είναι αναγκαία η δημιουργία πολλών κόμβων και υποκόμβων (δυσκολεύοντας έτσι και την οπτικοποίηση τους).

Συμπεράσματα

Συμπερασματικά, σε αυτήν την ενότητα "Αλγόριθμοι", παρουσιάστηκαν δύο κύριες οικογένειες αλγορίθμων που χρησιμοποιούνται στην μηχανική εκμάθηση. Η πρώτη κατηγορία είναι τα Δέντρα Αποφάσεων, τα οποία χωρίζονται σε δέντρα ταξινόμησης (για διακριτές τιμές) και δέντρα παλινδρόμησης (για συνεχείς τιμές, με αυξημένη πολυπλοκότητα). Η δεύτερη κατηγορία είναι οι αλγόριθμοι πιθανοτήτων υπό συνθήκη, από τους οποίους χρησιμοποιούνται κυρίως οι Naive Bayes για την επίλυση προβλημάτων μηχανικής εκμάθησης.

Αναλύθηκαν επίσης τα use cases στα οποία χρησιμοποιούνται οι δύο παραπάνω κατηγορίες, και συμπεραίνεται πως απαιτείται μελέτη του dataset (τύπος χαρακτηριστικών (διακριτές/συνεχείς τιμές), αριθμός χαρακτηριστικών, αριθμός εγγραφών) πριν ληφθεί η απόφαση για το ποιος αλγόριθμος θα χρησιμοποιηθεί για την ανάλυση και την πρόβλεψη του. Τέλος, καταγράφηκαν τα πλεονεκτήματα και οι περιορισμοί της κάθε κατηγορίας, και παρατηρήθηκε ότι οι διαφορές που έχουν μεταξύ τους είναι περισσότερες και πιο σημαντικές από τις ομοιότητες τους.

Υλοποίηση Machine learning αλγορίθμων

Διερευνητική ανάλυση δεδομένων

Το dataset που επιλέχθηκε, πάνω στο οποίο θα εφαρμοστούν οι 2 machine learning αλγόριθμοι, είναι το **Student Performance Data Set**[19] (Cortez & Silva, 2008). Το συγκεκριμένο σύνολο δεδομένων περιέχει 2 CSV[20] αρχεία, στα οποία περιλαμβάνονται βαθμοί μαθητών, δημογραφικά και κοινωνικά στοιχεία, από μαθητές δύο πορτογαλικών σχολείων δευτεροβάθμιας εκπαίδευσης. Σημαντικό είναι το γεγονός πως οι βαθμοί αφορούν αποκλειστικά 2 μαθήματα, τα Μαθηματικά και τα Πορτογαλικά, και ο διαχωρισμός στα 2 CSV αρχεία έχει γίνει με βάση τα μαθήματα (1 αρχείο περιέχει δεδομένα για τα Μαθηματικά και 1 αρχείο δεδομένα για τα Πορτογαλικά). Συμπεριλαμβάνεται επίσης ένα αρχείο R[21] το οποίο περιέχει ενδεικτικό κώδικα που συγχωνεύει τα 2 αρχεία και δείχνει τους κοινούς μαθητές που συμμετείχαν και στα 2 μαθήματα (382 στον αριθμό). Το προαναφερθέν αρχείο δε θα χρησιμοποιηθεί στην παρούσα έρευνα.

Το σύνολο των δεδομένων απαρτίζεται από συνολικά 1044 εγγραφές, εκ των οποίων οι 649 είναι για το μάθημα των Πορτογαλικών και οι 395 για το μάθημα των Μαθηματικών. Βέβαια όπως αναφέρθηκε και στην προηγούμενη παράγραφο, 382 από τους μαθητές είναι κοινοί στα 2 μαθήματα, οπότε στην πραγματικότητα οι μοναδικές εγγραφές του dataset μας είναι 662. Αυτή η πληροφορία μπορεί να αποδειχθεί ιδιαίτερα χρήσιμη σε περίπτωση που θέλουμε να εξάγουμε συμπεράσματα για τον κάθε μαθητή, ανεξαρτήτως του τι μάθημα παρακολούθησε (για αυτούς που παρακολούθησαν και τα 2 μπορούμε να πάρουμε τον μέσο όρο τους).

Στον παρακάτω πίνακα, μπορούμε να δούμε αναλυτικά τα χαρακτηριστικά (attributes) της κάθε εγγραφής του dataset:

[19] Το dataset είναι διαθέσιμο στο URL: https://archive.ics.uci.edu/ml/datasets/Student+Performance
[20] CSV = Comma Separated Value, είδος αρχείου που χρησιμοποιείται για αποθήκευση δεδομένων.
[21] R = είδος αρχείου και είδος γλώσσας προγραμματισμού, κατάλληλης για machine learning.

Πίνακας 1 - Dataset Attributes

Attribute	Values	Description
School	GP/MS	Το σχολείο του μαθητή
Sex	M/F	Φύλο
Age	15-22	Ηλικία
Address	R/U	Δυαδική τιμή που δείχνει αν ο μαθητής έμενε σε εξοχική κατοικία
Famsize	LE3/GT3	Μέλη οικογένειας, μικρότερο/ίσο του 3 ή μεγαλύτερο του 3
Pstatus	T/A	Οι γονείς ζουν μαζί ή όχι
Medu	0-4	Μορφωτικό επίπεδο μητέρας
Fedu	0-4	Μορφωτικό επίπεδο πατέρα
Mjob	Teacher/health/services/at_home/other	Επάγγελμα μητέρας
Fjob	Teacher/health/services/at_home/other	Επάγγελμα πατέρα
Reason	Home/reputation/course/other	Λόγος επιλογής του σχολείου
Guardian	Mother/father/other	Ο κηδεμόνας του μαθητή
Traveltime	1-4	Χρόνος που χρειάζεται ο μαθητής για να φτάσει στο σχολείο, 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour
Studytime	1-4	Εβδομαδιαίος χρόνος μελέτης στο σπίτι, 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours
Failures	1-4	Πόσες φορές έχει κοπεί ο μαθητής στο παρελθόν
Schoolsup	Yes/no	Ενισχυτική διδασκαλία από το σχολείο
Famsup	Yes/no	Ενισχυτική διδασκαλία από την οικογένεια
Paid	Yes/no	Ενισχυτική διδασκαλία επί πληρωμή
Activities	Yes/no	Εξωσχολικές δραστηριότητες
Nursery	Yes/no	Παρουσία μαθητή σε παιδικό σταθμό
Higher	Yes/no	Επιθυμία μαθητή να παρακολουθήσει ανώτατη εκπαίδευση
Internet	Yes/no	Πρόσβαση στο διαδίκτυο από το σπίτι
Romantic	Yes/no	Ο μαθητής βρίσκεται σε σχέση ή όχι
Famrel	1-5	Ποιότητα ενδοοικογενειακών σχέσεων
Freetime	1-5	Ελεύθερος χρόνος μετά το σχολείο
Goout	1-5	Διασκέδαση έξω με φίλους
Dalc	1-5	Κατανάλωση αλκοόλ καθημερινές
Walc	1-5	Κατανάλωση αλκοόλ σαββατοκύριακα
Health	1-5	Κατάσταση υγείας
Absences	0-93	Απουσίες κατά τη διάρκεια της σχολικής χρονιάς
G1	0-20	Βαθμός πρώτης περιόδου
G2	0-20	Βαθμός δεύτερης περιόδου
G3	0-20	Τελικός βαθμός

Παρατηρείται ότι υπάρχει πληθώρα χαρακτηριστικών για τον κάθε μαθητή, με πολύ χρήσιμες πληροφορίες στις οποίες αν γίνει μεθοδική ανάλυση, πιθανότατα να εξαχθούν κάποια ιδιαίτερα ενδιαφέροντα συμπεράσματα σχετικά με το πως μπορούν κάποιοι παράγοντες να επηρεάσουν τις επιδόσεις ενός μαθητή στο σχολείο.
Οι λόγοι για τους οποίους επιλέχθηκε το παραπάνω dataset είναι οι εξής:

- Ο αριθμός των attributes είναι αρκετά μεγάλος, έτσι ώστε τα δεδομένα να αναλυθούν από διαφορετικές οπτικές γωνίες.
- Τα στοιχεία που περιέχονται στο σύνολο δεδομένων προέρχονται από ποικίλες κατηγορίες, και περιέχουν πληροφορίες από όλες τις πτυχές στη ζωή των μαθητών. Από επιδόσεις στο σχολείο, μέχρι συνήθειες σχετικά με το αλκοόλ, όπως και στοιχεία σχετικά με τους γονείς, όλα αυτά προσδίδουν ιδιαίτερη αξία στο να αναλυθεί ενδελεχώς το dataset.
- Είναι ένα κατανοητό dataset, του οποίου τα attributes είναι ξεκάθαρα και δεν αφορούν εξειδικευμένα επιστημονικά θέματα. Αυτό αναφέρεται γιατί σε ένα αρκετά μεγάλο

ποσοστό από τα διαθέσιμα datasets, τα attributes παρουσίαζαν μεγάλη ιδιαιτερότητα και απαιτούσαν εξειδικευμένες γνώσεις για να γίνει αντιληπτή η αξία τους και τι πραγματικά υποδηλώνουν. Χαρακτηριστικά παραδείγματα τέτοιων dataset είναι τα Wine[22], Breast Cancer Wisconsin[23] και Heart Disease[24].

Ένα σαφές μειονέκτημα το οποίο πρέπει να αναφερθεί είναι ο μικρός αριθμός των εγγραφών.

Διαμόρφωση/προ-επεξεργασία δεδομένων

Κατά την προσωπική άποψη του συγγραφέα, η μεγαλύτερη δυσκολία στην επιστήμη του Machine Learning και του Data Science γενικότερα, είναι η σωστή μελέτη των δεδομένων από τον data scientist. Για τα καλύτερα δυνατά αποτελέσματα απαιτείται η κατανόηση των δεδομένων και της σημασίας των attributes, τι αξία προσδίδει κάθε χαρακτηριστικό στο dataset και πώς η αφαίρεση ή η τροποποίηση του μπορεί να επηρεάσει το αποτέλεσμα της έρευνας. Σίγουρα υπάρχουν και αυτόματα εργαλεία που προσφέρουν dimensionality reduction[25], αλλά ως πρώτο βήμα επιλέχθηκε η χειροκίνητη μείωση του αριθμού των attributes, που προέκυψε μετά από μελέτη των δεδομένων και εύρεση συσχετισμών ανάμεσα στα χαρακτηριστικά τους.

Αρχικά, στην παρούσα έρευνα δε θα γίνει διαχωρισμός στα δεδομένα με βάση τα μαθήματα. Όπως αναφέρθηκε προηγουμένως, τα δεδομένα συνιστώνται από δύο διαφορετικά αρχεία (1 για κάθε μάθημα), τα οποία φορτώθηκαν σε ένα python script (*scripts/mergeDatasets.py*) και συγχωνεύθηκαν σε ένα κοινό csv αρχείο. Στη συνέχεια, αφαιρέθηκαν οι μαθητές που είναι κοινοί και στα δύο μαθήματα με βάση το παρακάτω υποσύνολο στηλών:

```
subset=["school","sex","age","address","famsize","Pstatus","Medu","Fedu","Mjob","Fjob","reason
","nursery","internet","romantic","health","famrel","freetime","goout","traveltime"]
```

Επιλέχθηκε υποσύνολο στηλών (που αφορούν στοιχεία μαθητών) και όχι όλες οι στήλες, γιατί κάποιες στήλες (πχ απουσίες, βαθμός μαθήματος) αναφέρονται στο κάθε μάθημα αυτό καθαυτό και όχι σε χαρακτηριστικά μαθητή, επομένως δεν υπήρχε νόημα να συμπεριληφθούν στην ανωτέρω διαδικασία, η οποία σκοπό είχε να δημιουργήσει ένα dataset και με τα δύο μαθήματα, αλλά με μοναδικούς μαθητές. Το αποτέλεσμα ήταν να δημιουργηθεί ένα νέο csv αρχείο με 672 συνολικές εγγραφές, το οποίο ονομάστηκε *student-merged.csv*.

Ύστερα από μελέτη των attributes του συνόλου δεδομένων παρατηρήθηκε ότι στο κομμάτι της προ-επεξεργασίας, μπορούν τρεις δυάδες χαρακτηριστικών να συγχωνευθούν:

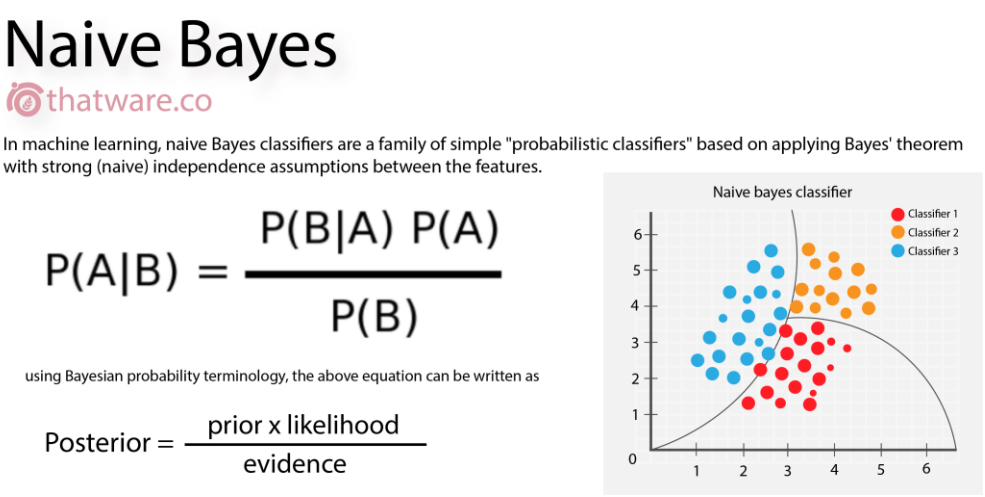
- Fedu + Medu: Παίρνουν τιμές από 0-4 και εκφράζουν το μορφωτικό επίπεδο του πατέρα και της μητέρας αντίστοιχα. Θα συγχωνευθούν σε ένα attribute “Pedu” (parent education) το οποίο θα περιέχει τους μέσους όρους από τις 2 αυτές στήλες, προφανώς στρογγυλοποιημένους σε ακέραια τιμή.
- Freetime + Goout: Παίρνουν τιμές από 1-5 και δείχνουν πόσο ελεύθερο χρόνο έχει ο μαθητής και πόσο βγαίνει έξω. Θα συγχωνευθούν στο υφιστάμενο attribute “Freetime”.
- Dalc + Walc: Παίρνουν τιμές από 1-5 και δείχνουν την κατανάλωση αλκοόλ καθημερινές και σαββατοκύριακα. Θα συγχωνευθούν σε ένα attribute “Alc” (alcohol).

Με τις παραπάνω συγχωνεύσεις ο αριθμός των attributes έπεσε από το 33 στο 30, που σημαίνει μια μείωση κοντά στο 9%. Το script που εκτελεί την παραπάνω διαδικασία είναι το *scripts/reduceFeatures.py*.

[22] https://archive.ics.uci.edu/ml/datasets/Wine
[23] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29
[24] https://archive.ics.uci.edu/ml/datasets/Heart+Disease
[25] Διαδικασία μείωσης των χαρακτηριστικών ενός dataset, με σκοπό την καλύτερη ανάλυση δεδομένων.

Naive Bayes Algorithm

Παρουσίαση Naive Bayes classifier



Εικόνα 6 - Naive Bayes Classifier & Bayes Theorem

Η πρώτη μέθοδος που επιλέχθηκε για να αναλυθεί το ανωτέρω σύνολο δεδομένων, είναι ο αλγόριθμος εποπτευόμενης μάθησης Naive Bayes. Στην επιστήμη της μηχανικής εκμάθησης, οι ταξινομητές (classifiers) Naive Bayes ανήκουν στην οικογένεια πιθανολογικών αλγορίθμων οι οποίοι βασίζονται στο θεώρημα του Bayes, υποθέτοντας ότι υπάρχει ανεξαρτησία ανάμεσα στα χαρακτηριστικά των δεδομένων. Λόγω της παραπάνω υπόθεσης οι αλγόριθμοι αυτοί ονομάζονται "αφελείς" (Wikipedia, 2020). Οι ταξινομητές Naive Bayes έχουν αναλυθεί εκτενώς στην ενότητα "Conditional probability algorithms".

Ένας από τους λόγους που επιλέχθηκε ο συγκεκριμένος αλγόριθμος είναι οι εξαιρετικές επιδόσεις του και στους 4 επιστημονικούς τομείς[26] *accuracy, prediction, recall, F1*. Σύμφωνα με μια δημοσίευση του 2009 τριών εισηγητών του Loughborough University, στην οποία συγκρίνεται ο Naive Bayes (NB) με τα Δέντρα Αποφάσεων και τα Νευρωνικά Δίκτυα, αποδεικνύεται ότι τα καταφέρνει πολύ καλά τόσο σε επιδόσεις όσο και σε χρόνους εκτέλεσης (Xhemali, Hinde, & Stone, 2009). Ένα επιπλέον πλεονέκτημα του NB είναι ότι μπορεί να χρησιμοποιηθεί τόσο για δυαδικές τιμές όσο και για προβλήματα ταξινόμησης με πολλές κλάσεις (Oracle, 2020). Στην προκειμένη περίπτωση το dataset μας ικανοποιεί πλήρως τις προαναφερθείσες προϋποθέσεις. Ένα σαφές μειονέκτημα του αλγορίθμου είναι η "αφέλεια" του, όπως προαναφέρθηκε, καθώς κάθε χαρακτηριστικό (κλάση) θεωρείται ανεξάρτητο από τα υπόλοιπα και σε πολλά προβλήματα της καθημερινότητας συμβαίνει το ακριβώς αντίθετο, δηλαδή να υπάρχουν αρκετές εξαρτήσεις ανάμεσα στα attributes ενός συνόλου δεδομένων.

Εφαρμογή Naive Bayes

Από την ανάλυση που έχει γίνει στην ενότητα "Conditional probability algorithms", είναι γνωστό πως υπάρχουν 3 κατηγορίες αλγορίθμων Naive Bayes, και η κάθε κατηγορία χρησιμοποιείται σε συγκεκριμένες περιπτώσεις. Με αυτό το δεδομένο αποκλείονται οι παρακάτω κατηγορίες:

- "Multinomial Naive Bayes": Χρησιμοποιούνται αποκλειστικά για προβλήματα ταξινόμησης και στο dataset που έχει επιλεχθεί, αρκετά από τα attributes περιέχουν ακεραίους.

[26] Οι 4 όροι (accuracy, prediction, recall, F1) καθορίζουν πόσο αποδοτικός είναι ένας αλγόριθμος (Riggio, 2019)

$$\textbf{Accuracy} = \frac{\text{αριθμός σωστών προβλέψεων}}{\text{συνολικός αριθμός προβλέψεων}} \qquad \textbf{Precision} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\textbf{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative} \qquad \textbf{F1} = 2 * \frac{precision * recall}{precision + recall}$$

- “Bernoulli Naive Bayes”: Χρησιμοποιούνται αποκλειστικά σε προβλήματα με δυαδικές τιμές, κάτι το οποίο δεν ισχύει στην προκειμένη περίπτωση.

Συνεπώς, ο αλγόριθμος που χρησιμοποιήθηκε πάνω στο dataset είναι ο **Gaussian Naive Bayes**.

Ο συγκεκριμένος αλγόριθμος λειτουργεί αποκλειστικά με αριθμητικές τιμές, οπότε έπρεπε να γίνει μεγάλη προ-επεξεργασία στα δεδομένα προκειμένου να μετατραπούν όλα τα χαρακτηριστικά σε αριθμητικές τιμές, χωρίς να χάνεται το νόημα. Η μετατροπή έγινε σύμφωνα με τον παρακάτω πίνακα, και μπορεί να χρησιμοποιηθεί για αναφορά:

Πίνακας 2 - Μετατροπή χαρακτηριστικών σε αριθμητικές τιμές

attribute	value before	value after
school	GP	0
	MS	1
sex	F	0
	M	1
address	U	0
	R	1
famsize	GT3	0
	LE3	1
Pstatus	A	0
	T	1
Mjob	at_home	0
	health	1
	services	2
	teacher	3
	other	4
Fjob	at_home	0
	health	1
	services	2
	teacher	3
	other	4
reason	home	0
	reputation	1
	course	2
	other	3
guardian	mother	0
	father	1
	other	2
schoolsup	no	0
	yes	1
famsup	no	0
	yes	1
paid	no	0
	yes	1
activities	no	0
	yes	1

nursery	no	0
	yes	1
higher	no	0
	yes	1
internet	no	0
	yes	1
romantic	no	0
	yes	1

Μετά την επιτυχή μετατροπή, το dataset χωρίστηκε σε train και test subsets με αναλογία 80/20, και εφαρμόστηκε ο Gaussian Naive Bayes. Εξετάστηκαν τα παρακάτω χαρακτηριστικά:

- Sex: 2 πιθανές τιμές
- School: 2 πιθανές τιμές
- Higher: 2 πιθανές τιμές
- Studytime: 4 πιθανές τιμές
- G3: 21 πιθανές τιμές
- Absences: 94 πιθανές τιμές

Είναι προφανές πως όσο περισσότερες είναι οι πιθανές τιμές ενός χαρακτηριστικού, τόσο μικρότερη θα είναι και η ακρίβεια του αλγορίθμου, κάτι το οποίο επιβεβαιώνεται και από τις δοκιμές που έγιναν στην παρούσα έρευνα. Εκτός από δύο ενδεικτικά Jupyter notebooks (*notebooks/GaussianNBSex.ipynb, notebooks/GaussianNBG3.ipynb*) που φτιάχτηκαν για σκοπούς παρουσίασης (demonstration purposes), δημιουργήθηκε και ένα interactive python script (*gaussianNB.py*) στο οποίο ο χρήστης μπορεί να επιλέξει κάποιο από τα παραπάνω χαρακτηριστικά, και να επιλέξει και τον αριθμό των επαναλήψεων που θέλει για να τρέξει ο αλγόριθμος (με σκοπό να εξαχθεί πιο ασφαλές συμπέρασμα αναφορικά με την ακρίβεια του αλγορίθμου). Στον παρακάτω πίνακα απεικονίζονται τα αποτελέσματα για κάθε ένα από τα χαρακτηριστικά που εξετάστηκαν, για **10000 επαναλήψεις**:

Πίνακας 3 - Αποτελέσματα αλγορίθμου Gaussian NB (10000 επαναλήψεις)

attribute	time (in seconds)	average score
sex	33.37	69.16%
school	27.66	76.45%
higher	28.78	79.88%
studytime	30.56	42.14%
G3	45.10	22.12%
absences	55.63	2.78%

Αυτό που παρατηρείται είναι πως στα 3 πρώτα χαρακτηριστικά, που παίρνουν μόνο 2 πιθανές τιμές, η ακρίβεια του αλγορίθμου κυμαίνεται στα επίπεδα 70% - 80%, με το φύλο να παίρνει τη χαμηλότερη βαθμολογία και το higher (επιθυμία μαθητή να παρακολουθήσει ανώτατη εκπαίδευση) να παίρνει την υψηλότερη. Μια πιθανή εξήγηση για αυτό μπορεί να είναι το γεγονός πως το attribute higher επηρεάζεται σε μεγαλύτερο βαθμό από τα υπόλοιπα χαρακτηριστικά, σε σχέση με το attribute sex, το οποίο είναι ένα δημογραφικό στοιχείο.

Ο αλγόριθμος δεν τα πήγε καλά στο attribute studytime, αφού πέτυχε ακρίβεια 42.14% για 4 πιθανές τιμές. Στον τελικό βαθμό (G3), τα πήγε σχετικά καλύτερα, αφού με ακρίβεια 22.12% πετυχαίνει με απόλυτη τιμή το βαθμό του μαθητή, από 21 πιθανές τιμές. Όπως ήταν φυσικό, η ακρίβεια πέφτει κατακόρυφα (περίπου 2 στις 100 σωστές προσπάθειες) για την πρόβλεψη των απουσιών του μαθητή (absences), αφού οι πιθανές τιμές του συγκεκριμένου attribute είναι 94.

Σχετικά με τον βαθμό (G3) του μαθητή, όπως αναφέρθηκε και παραπάνω η ακρίβεια (22.12%) αναφέρεται στην απόλυτη πρόβλεψη. Στον παρακάτω ενδεικτικό πίνακα απεικονίζονται οι τιμές των πραγματικών βαθμών και των βαθμών που προέβλεψε ο αλγόριθμος:

Πίνακας 4 - Τελικοί βαθμοί 30 μαθητών (Naive Bayes)

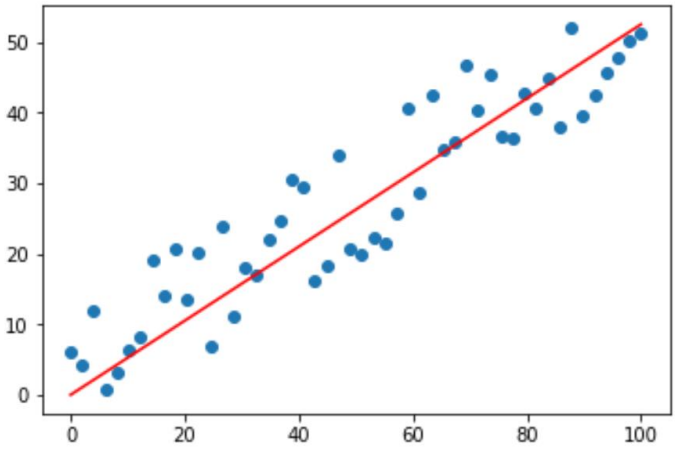
actual	8	9	6	11	14	11	17	10	11	0	12	15	14	6	9	15	11	10	14	13	10	10	12	16	11	13	19	8	0	8
predicted	9	6	6	0	16	12	19	6	16	0	16	16	16	8	11	16	13	11	13	16	6	0	16	17	0	16	17	6	0	5

Στις 22 από τις 30 ενδεικτικές εγγραφές (ποσοστό 73%) ο αλγόριθμος έχει προβλέψει τον τελικό βαθμό με μια απόκλιση ± 3 μονάδων. Αυτό στην ουσία μπορεί να μεταφραστεί πως για τους 7 από τους 10 μαθητές ο αλγόριθμος γνωρίζει πόσο θα γράψουν στις τελικές εξετάσεις με μια μικρή (σχετικά) απόκλιση.

Τέλος, ένα ακόμη στοιχείο που πρέπει να αναφερθεί είναι πως ο χρόνος εκτέλεσης του αλγορίθμου φαίνεται να αυξάνεται για τα χαρακτηριστικά που παίρνουν περισσότερες πιθανές τιμές.

Linear Regression Algorithm

Παρουσίαση Linear Regression



Εικόνα 7 - Linear Regression Visualized

Η δεύτερη μέθοδος που επιλέχθηκε να υλοποιηθεί με σκοπό την ανάλυση/πρόβλεψη του dataset, είναι ο αλγόριθμος εποπτευόμενης μάθησης Linear Regression. Ο συγκεκριμένος αλγόριθμος είναι από τους πιο γνωστούς στους τομείς της στατιστικής και της μηχανικής εκμάθησης, και ορίζεται ως "ένα γραμμικό μοντέλο το οποίο βρίσκει την συσχέτιση ανάμεσα σε δεδομένα εισόδου x και σε αυστηρά ένα δεδομένο εξόδου y" (Brownlee, 2016). Πιο συγκεκριμένα, το y μπορεί να υπολογιστεί από έναν γραμμικό συνδυασμό των x. Χρησιμοποιείται σε προβλήματα όπου τα δεδομένα αποτελούνται από συνεχείς τιμές.

Ο Linear Regression χωρίζεται στις εξής παρακάτω κατηγορίες, αναλόγως του αριθμού των δεδομένων εισόδου:

- **Simple Linear Regression**: Όταν υπάρχει μόνο ένα δεδομένο εισόδου.
- **Multiple Linear Regression**: Όταν τα δεδομένα εισόδου είναι περισσότερα από ένα.

Σχετικά με τις τεχνικές που χρησιμοποιούνται στην προετοιμασία των δεδομένων και στην εκπαίδευση του αλγορίθμου, η πιο γνωστή είναι η μέθοδος των ελαχίστων τετραγώνων[27]. Σε ένα απλό πρόβλημα παλινδρόμησης, όπου υπάρχει μια μεταβλητή εισόδου και μια μεταβλητή εξόδου, η σχέση ανάμεσα στο x και στο y είναι η παρακάτω:

$y = B0 + B1*x$

[27] Σε αυτή τη μέθοδο παράγεται μια γραμμή η οποία είναι όσο το δυνατόν πιο κοντά σε όλα τα data points. Αυτό επιτυγχάνεται ελαχιστοποιώντας το άθροισμα των τετραγώνων των αποστάσεων των data points.

Σε περισσότερες διαστάσεις, όπου τα δεδομένα εισόδου είναι περισσότερα, αντί για γραμμή (line) δημιουργείται ένα αεροπλάνο (plane) ή ένα hyperplane, και η πολυπλοκότητα σαφώς αυξάνεται.

Ο κύριος λόγος για τον οποίο επιλέχθηκε ο συγκεκριμένος αλγόριθμος ως δεύτερη μέθοδος για την ανάλυση των δεδομένων, είναι γιατί πρόκειται και αυτός για έναν αλγόριθμο εποπτευόμενης μάθησης, που σημαίνει ότι μπορεί να χρησιμοποιηθεί για πρόβλεψη πάνω σε μια μεταβλητή. Χρειάζεται και αυτός, όπως και ο Naive Bayes, να εκπαιδευτεί προτού μπορέσει να παράγει κάποια αποτελέσματα, και ο σκοπός της παρούσας έρευνας είναι να συγκρίνει δύο αλγορίθμους υπό τις ίδιες συνθήκες (ίδιο dataset, ίδια προ-επεξεργασία δεδομένων, ίδια χαρακτηριστικά υπό μελέτη) με σκοπό την σύγκριση των αποτελεσμάτων και των χρόνων υλοποίησης.

Εφαρμογή Linear Regression

Για την συγκεκριμένη έρευνα χρησιμοποιήθηκε ο πιο πολύπλοκος Multiple Linear Regression αλγόριθμος. Αυτό σημαίνει στην ουσία ότι για κάθε χαρακτηριστικό που μελετάται, σαν δεδομένα εισόδου δίνονται όλα τα υπόλοιπα, και αυτή η προσέγγιση χρησιμοποιήθηκε γιατί με τον ίδιο τρόπο έγινε η ανάλυση και στον Naive Bayes αλγόριθμο. Και εδώ έγινε η ίδια προ-επεξεργασία δεδομένων, και μετατράπηκαν όλα τα χαρακτηριστικά σε αριθμητικές τιμές σύμφωνα με τον Πίνακα 2. Τα χαρακτηριστικά στα οποία έγινε η πρόβλεψη είναι τα ίδια με του Naive Bayes:

sex, school, higher, studytime, G3, absences

Το dataset χωρίστηκε επίσης με το ίδιο ποσοστό 80/20 σε train και test subsets, αλλά στην προκειμένη περίπτωση πρέπει να αναφερθεί ένα ακόμη βήμα που υλοποιήθηκε προκειμένου τα αποτελέσματα να είναι συγκρίσιμα. Ο Linear Regression είναι φύσει αλγόριθμος που χρησιμοποιείται για πραγματικές συνεχείς τιμές, οπότε το επιπλέον βήμα που έγινε ήταν η στρογγυλοποίηση των αποτελεσμάτων και η μετατροπή τους σε ακεραίους, για να έχουν την ίδια μορφή με τα αποτελέσματα του Naive Bayes. Φτιάχτηκαν και εδώ 2 ενδεικτικά Jupyter notebooks (*notebooks/LinearRegressionSex.ipynb, notebooks/LinearRegressionG3.ipynb*) που δείχνουν την υλοποίηση του αλγορίθμου, καθώς και ένα interactive python script (*linearRegression.py*) το οποίο προσφέρει τις ίδιες δυνατότητες με το αντίστοιχο του Naive Bayes. Στον παρακάτω πίνακα απεικονίζονται τα αποτελέσματα για κάθε ένα από τα χαρακτηριστικά που εξετάστηκαν, για **10000 επαναλήψεις**:

Πίνακας 5 - Αποτελέσματα αλγορίθμου Linear Regression (10000 επαναλήψεις)

attribute	time (in seconds)	average score
sex	39.03	70.37%
school	40.57	80.00%
higher	39.88	91.85%
studytime	39.01	55.55%
G3	36.71	39.25%
absences	36.94	11.11%

Σύμφωνα με τον πίνακα φαίνεται ότι για τα 3 πρώτα χαρακτηριστικά (δυαδικές τιμές) η ακρίβεια του αλγορίθμου κυμαίνεται σε επίπεδα 70% - 90%. Αναφορικά με το studytime, τις μισές και λίγο παραπάνω φορές ο αλγόριθμος προβλέπει σωστά τον χρόνο που ξοδεύει ένας μαθητής για μελέτη στο σπίτι, ενώ με ποσοστό κοντά στο 40% προβλέπει επιτυχώς και τον τελικό βαθμό του (21 πιθανές τιμές). Για το συγκεκριμένο χαρακτηριστικό (G3) η επίδοση του συγκεκριμένου αλγορίθμου μπορεί να χαρακτηριστεί εξαιρετική. Για περίπου 1 στους 10 μαθητές προβλέπει με ακρίβεια και τις απουσίες του μέσα στη χρονιά, κάτι το οποίο πάλι είναι

εντυπωσιακό καθώς οι πιθανές τιμές του ανωτέρω χαρακτηριστικού είναι 91. Πρέπει να αναφερθεί ότι ο παραπάνω πίνακας αναφέρεται σε απόλυτες προβλέψεις και όχι προσεγγιστικές.

Αναφορικά με τον τελικό βαθμό G3 των μαθητών, στον παρακάτω πίνακα φαίνονται οι προβλέψεις του αλγορίθμου σε 30 τυχαίους μαθητές:

Πίνακας 6 - Τελικοί βαθμοί 30 μαθητών (Linear Regression)

actual	8	0	16	8	10	12	11	16	18	12	17	18	0	10	8	10	12	13	12	10	9	11	11	8	13	18	0	13	11	14
predicted	8	8	16	7	11	12	9	16	19	11	17	18	6	10	7	8	12	13	12	12	8	10	11	7	14	17	0	13	11	14

Σύμφωνα με το παραπάνω δείγμα επιβεβαιώνονται οι εξαιρετικές επιδόσεις του αλγορίθμου, αφού σε ποσοστό 50% έχει προβλέψει **ακριβώς** τον βαθμό, και στους 28/30 (ποσοστό 93%) έχει κάνει σωστή πρόβλεψη με απόκλιση ± 3 μονάδων.

Σχετικά με τους χρόνους, στον συγκεκριμένο αλγόριθμο φαίνεται πως δεν σχετίζεται ο αριθμός των πιθανών τιμών ενός attribute με τον χρόνο που χρειάζεται για να ολοκληρωθεί η ανάλυση. Αντιθέτως, στα χαρακτηριστικά με πολλές πιθανές τιμές ολοκληρώθηκε 3 δευτερόλεπτα νωρίτερα. Στα πρώτα 4 χαρακτηριστικά πάντως, οι επιδόσεις είναι αρκετά χειρότερες σε σχέση με τον Naive Bayes, αφού ο χρόνος ολοκλήρωσης είναι μεγαλύτερος κατά περίπου 10 δευτερόλεπτα (αύξηση χρόνου κατά 33%).

Σύγκριση αποτελεσμάτων - Συμπεράσματα

Στην ενότητα “Υλοποίηση Machine learning αλγορίθμων”, υλοποιήθηκαν οι αλγόριθμοι εποπτευόμενης μάθησης Gaussian Naive Bayes και Multiple Linear Regression, πάνω σε ένα dataset που περιέχει δεδομένα μαθητών από δύο σχολεία, με σκοπό την ανάλυση και πρόβλεψη τιμών σε 6 εκ των 33 χαρακτηριστικών του. Οι συνθήκες υλοποίησης ήταν οι ίδιες (με τις μόνες διαφορές να σχετίζονται με τα ιδιαίτερα χαρακτηριστικά του κάθε αλγορίθμου), προκειμένου η σύγκριση να είναι όσο το δυνατόν πιο δίκαιη. Παρακάτω υπάρχει ένας συγκεντρωτικός πίνακας και από τις δύο υλοποιήσεις, που δείχνει τις επιδόσεις των αλγορίθμων σε ακρίβεια και σε χρόνους:

Naive Bayes			**Linear Regression**		
attribute	**time (in seconds)**	**average score**	**attribute**	**time (in seconds)**	**average score**
sex	33.37	69.16%	sex	39.03	70.37%
school	27.66	76.45%	school	40.57	80.00%
higher	28.78	79.88%	higher	39.88	91.85%
studytime	30.56	42.14%	studytime	39.01	55.55%
G3	45.10	22.12%	G3	36.71	39.25%
absences	55.63	2.78%	absences	36.94	11.11%

Από τα αποτελέσματα φαίνεται η ξεκάθαρη υπεροχή του Linear Regression στην ακρίβεια, αφού και στα 6 χαρακτηριστικά που αναλύθηκαν έχει καλύτερες επιδόσεις. Οι μεγαλύτερες διαφορές παρατηρούνται στα attributes higher, studytime και G3 (σχεδόν διπλάσιο ποσοστό ακρίβειας). Με αυτόν τον τρόπο αποδεικνύεται ότι ένας αλγόριθμος παλινδρόμησης (που εφαρμόζεται πάνω σε συνεχείς τιμές) μπορεί να τροποποιηθεί κατάλληλα έτσι ώστε να εφαρμοστεί και σε προβλήματα ταξινόμησης, και μάλιστα με αρκετά μεγάλη αποτελεσματικότητα. Στους χρόνους, όπως προαναφέρθηκε, ο Naive Bayes είναι καλύτερος και επιβεβαιώνει ένα από τα πλεονεκτήματα που έχουν αναφερθεί στην παρουσίαση του, που είναι ο μικρός χρόνος εκπαίδευσης και πρόβλεψης.

Σε κάθε περίπτωση, και τα 2 πειράματα μπορούν να θεωρηθούν επιτυχή αφού δεν οδήγησαν σε κάποιο αδιέξοδο (πχ αποτυχία πρόβλεψης, υπερβολικά χαμηλή ακρίβεια κλπ.) και θα μπορούσαν να χρησιμοποιηθούν σε πραγματικές συνθήκες με το εν λόγω dataset.

Σύνοψη

Ανακεφαλαίωση

Συνοψίζοντας, η παρούσα εργασία είχε δύο κύρια μέρη. Στην πρώτη ενότητα "Αλγόριθμοι" αναλύθηκαν από θεωρητική σκοπιά τα δέντρα αποφάσεων και οι αλγόριθμοι πιθανοτήτων υπό συνθήκη, και παρουσιάστηκαν τα δυνατά και αδύναμα σημεία της κάθε κατηγορίας. Στη συνέχεια, από την σύγκριση που έγινε μεταξύ τους παρουσιάστηκαν οι ομοιότητες και οι (περισσότερες) διαφορές τους.

Στην δεύτερη ενότητα που περιείχε την υλοποίηση των δύο αλγορίθμων, έγινε μια προσπάθεια δημιουργίας ενός συστήματος λήψης αποφάσεων με σκοπό την πρόβλεψη συγκεκριμένων χαρακτηριστικών του dataset. Υλοποιήθηκαν οι αλγόριθμοι Naive Bayes και Linear Regression, και εκ του αποτελέσματος ο δεύτερος αποδείχθηκε κατά πολύ καλύτερος, αφού οι προβλέψεις που έκανε ήταν εντυπωσιακές. Τα χαρακτηριστικά που επιλέχθηκαν να αναλυθούν ήταν 6, και είχαν διαφορετικό αριθμό πιθανών τιμών για την εξαγωγή καλύτερων συμπερασμάτων.

Επόμενα βήματα

Σαν επόμενα βήματα, θα μπορούσε ο κώδικας Python που έχει γραφτεί για την ανάλυση και πρόβλεψη του παραπάνω dataset να εμπλουτιστεί περαιτέρω ώστε να βελτιωθεί η απόδοση και η ακρίβεια των αλγορίθμων. Μια ακόμη ιδέα είναι να δοκιμαστούν ακόμη περισσότεροι αλγόριθμοι όπως CART, Apriori, Logistic regression έτσι ώστε να εξαχθεί ένα γενικότερο συμπέρασμα σχετικά με το πως συμπεριφέρεται κάθε τύπος αλγορίθμου απέναντι στο συγκεκριμένο dataset. Επίσης, θα μπορούσαν να δοκιμαστούν και κάποια επιπλέον σύνολα δεδομένων πάνω στον υπάρχοντα κώδικα (με τις κατάλληλες τροποποιήσεις) για να αποδειχθεί κατά πόσο ήταν προσαρμοσμένος στο συγκεκριμένο dataset (overfitting) ή αν παράγει ικανοποιητικά αποτελέσματα για οποιοδήποτε dataset. Ο όρος overfitting αναφέρεται σε κακή απόδοση αλγορίθμου και "συμβαίνει όταν ο αλγόριθμος μαθαίνει τις λεπτομέρειες και τον θόρυβο ενός dataset σε τέτοιο βαθμό, ώστε να επηρεάζει αρνητικά την απόδοση του σε νέα δεδομένα" (Brownlee, 2016).

Ακόμη, θα μπορούσε να γίνει μελέτη όλων των χαρακτηριστικών του dataset για τους 2 αλγορίθμους, έτσι ώστε να μπορέσει να γίνει καλύτερη σύγκριση μεταξύ τους. Τέλος, θα μπορούσε να γραφτεί ένα επιπλέον script το οποίο να γεννάει ψευδοτυχαίους μαθητές και να τους προσθέτει στο υφιστάμενο dataset, αυξάνοντας έτσι τον συνολικό αριθμό των εγγραφών, με σκοπό να εξεταστούν οι 2 αλγόριθμοι και σε μεγαλύτερα μεγέθη για να αποδειχθεί αν έχουν την ίδια συμπεριφορά.

Οδηγίες εκτέλεσης αλγορίθμων

Για την εκτέλεση των βημάτων και του κώδικα που συμπεριλαμβάνεται στην παρούσα εργασία, είναι απαραίτητο να υπάρχει εγκατεστημένη η python 3 καθώς και (ιδανικά) jupyter server. Περισσότερες οδηγίες σχετικά με τα παραπάνω:
https://phoenixnap.com/kb/how-to-install-python-3-windows
https://jupyter.org/install

Εφόσον πληρούνται οι ανωτέρω προδιαγραφές, ο υπολογιστής θα μπορεί να εκτελέσει τα βασικά python scripts. Όλος ο κώδικας βρίσκεται στον φάκελο “code”. Για την ομαλή διεξαγωγή του πειράματος, συνίσταται να τρέξουν τα python scripts με αυτή τη σειρά:

scripts/mergeDatasets.py: Χρησιμοποιείται για την συγχώνευση των 2 csv αρχείων και την παραγωγή ενός νέου csv που περιέχει μαθητές που έχουν παρακολουθήσει και τα 2 μαθήματα.

scripts/reduceFeatures.py: Χρησιμοποιείται για την μείωση των attributes στο dataset από 33 σε 30.

gaussianNB.py: Interactive shell μέσω του οποίου μπορεί να εκτελεστεί ο Naive Bayes αλγόριθμος σε attribute επιλογής του χρήστη, για όσες επαναλήψεις θέλουμε.

linearRegression.py: Interactive shell μέσω του οποίου μπορεί να εκτελεστεί ο Linear Regression αλγόριθμος σε attribute επιλογής του χρήστη, για όσες επαναλήψεις θέλουμε.

Για την εκτέλεση των notebooks (στον φάκελο notebooks), θα πρέπει να τρέχει ένας jupyter server και να φορτωθούν εκεί.
Σημαντική σημείωση: Για όλα τα αρχεία χρησιμοποιούνται relative και όχι absolute paths, που σημαίνει ότι δεν πρέπει να αλλάξει το structure του project.

Αναφορές

Anne Marie Helmenstine, P. (2019, August 12). Retrieved from Thoughtco: https://www.thoughtco.com/bayes-theorem-4155845

Barone, A. (2020). *Investopedia*. Retrieved from https://www.investopedia.com/terms/c/conditional_probability.asp

Botha, M. (2019, January). Retrieved from https://towardsdatascience.com/the-15-most-important-ai-companies-in-the-world-79567c594a11

Breiman, Friedman, Olshen, & Stone. (1984). *Classification and Regression Trees.* Wadsworth Int. Group.

Brownlee, J. (2016, March). Retrieved from https://machinelearningmastery.com/linear-regression-for-machine-learning/

Brownlee, J. (2016, March). Retrieved from https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

Catanzarite, J. (2018, December). *towardsdatascience*. Retrieved from https://towardsdatascience.com/the-naive-bayes-classifier-e92ea9f47523

Chauhan, N. S. (2019, December). *towardsdatascience*. Retrieved from https://towardsdatascience.com/decision-tree-algorithm-explained-83beb6e78ef4

Cortez, P., & Silva, A. (2008). *Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference.* Porto: EUROSIS.

Dabbura, I. (2018, September). *towardsdatascience*. Retrieved from towardsdatascience: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

Das, S., & Cakmak, U. M. (2018). *Hands-On Automated Machine Learning.* April.

Dhiraj, K. (2019, May). *Medium*. Retrieved from https://medium.com/@dhiraj8899/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a

Foote, K. D. (2019, March). *dataversity*. Retrieved from https://www.dataversity.net/a-brief-history-of-machine-learning/#

Frankenfield, J. (2020, March). Retrieved from https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp

Friedman, J. H. (1991). *Multivariate adaptive regression splines.* Retrieved from https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.382.970

Fumo, D. (2017, June). *towardsdatascience*. Retrieved from https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

Gandhi, R. (2018, May). *towardsdatascience*. Retrieved from https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

GeeksForGeeks. (2020). *GeeksForGeeks*. Retrieved from https://www.geeksforgeeks.org/python-pandas-dataframe/

Google. (2020, February). *Google*. Retrieved from Google: https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages

Hayes, A. (2020). *Investopedia*. Retrieved from https://www.investopedia.com/terms/b/bayes-theorem.asp

Hochreiter, S., & Schmidhuber, J. (1997, December). *Long Short0term Memory. Neural computation*. Retrieved from https://www.researchgate.net/publication/13853244_Long_Short-term_Memory

Mitchell, T., & Hill, M. (1997). *Machine Learning*. Retrieved from http://www.cs.cmu.edu/~tom/mlbook.html

Naeem, S., & Wumaier, A. (2018, December). *Study and Implementing K-mean Clustering Algorithm on English Text and Techniques to Find the Optimal Value of K*. Retrieved from https://www.researchgate.net/publication/331045887_Study_and_Implementing_K-mean_Clustering_Algorithm_on_English_Text_and_Techniques_to_Find_the_Optimal_Value_of_K

Oracle. (2020). *Oracle*. Retrieved from Oracle: https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/algo_nb.htm#DMCON018

Pulipaka, D. (2016). *Medium*. Retrieved from https://medium.com/@gp_pulipaka/an-essential-guide-to-classification-and-regression-trees-in-r-language-4ced657d176b

Riggio, C. (2019, November). *towardsdatascience*. Retrieved from https://towardsdatascience.com/whats-the-deal-with-accuracy-precision-recall-and-f1-f5d8b4db1021

SAS. (2020). *SAS*. Retrieved from https://www.sas.com/en_us/insights/analytics/data-mining.html

Sujan, N. I. (2018, June). *Medium*. Retrieved from Medium: https://medium.com/coinmonks/what-is-entropy-and-why-information-gain-is-matter-4e85d46d2f01

Teggi, P. (2020, February). *Medium*. Retrieved from https://medium.com/@pralhad2481/chapter-3-decision-tree-learning-part-1-d0ca2365bb22

Wikipedia. (2020). Retrieved from https://en.wikipedia.org/wiki/Use_case

Wikipedia. (2020). Retrieved from https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Wikipedia. (2020). Retrieved from https://en.wikipedia.org/wiki/Pattern_recognition

Xhemali, D., Hinde, C. J., & Stone, R. G. (2009). Naïve Bayes vs. Decision Trees vs. Neural Networks in the Classification of Training Web Pages . Leicestershire, United Kingdom.

Προσάρτημα

Στο παρόν προσάρτημα υπάρχει όλος ο κώδικας που χρησιμοποιήθηκε σε αυτήν την έρευνα. Τα jupyter notebooks δεν υπάρχουν εδώ καθώς η επικόλληση τους δεν βοηθάει καθόλου στην ανάγνωση.

mergeDatasets.py

```
import pandas as pd
import datetime

begin_time = datetime.datetime.now()

# Load datasets
df1=pd.read_csv("../data/student-mat.csv", sep=";")
df2=pd.read_csv("../data/student-por.csv", sep=";")

# Merge datasets
merged=pd.concat([df1, df2])

# Drop duplicate students
result = merged.drop_duplicates(subset=["school","sex","age","address","famsize",
"Pstatus","Medu","Fedu","Mjob","Fjob","reason","nursery","internet","romantic","h
ealth","famrel","freetime","goout","traveltime"])

# Export dataframe to csv
result.to_csv("../data/student-merged.csv", index=False)

# Print script execution time
print(f"The new merged dataset has {len(result)} rows and is saved at ./data/stud
ent-merged.csv")
print(f"Script completed successfully, time: {datetime.datetime.now() - begin_tim
e}")
```

reduceFeatures.py

```
import pandas as pd
import datetime

begin_time = datetime.datetime.now()

df=pd.read_csv('../data/student-merged.csv')

# Calculate means and create new columns
df['Pedu'] = df.apply(lambda row: int(round((row.Medu + row.Fedu)/2)), axis = 1)
df['freetime'] = df.apply(lambda row: int(round((row.freetime + row.goout)/2)), a
xis = 1)
df['alc'] = df.apply(lambda row: int(round((row.Dalc + row.Walc)/2)), axis = 1)

# Drop unneeded columns
df.drop(['Medu','Fedu','goout','Dalc','Walc'],axis='columns',inplace=True)

# Export dataframe to csv
df.to_csv("./data/student-merged.csv", index=False)
```

```
# Print script execution time
print(f"Script completed successfully, time: {datetime.datetime.now() - begin_tim
e}")
```

gaussianNB.py

```
attributes = ['sex', 'school', 'higher', 'studytime', 'G3', 'absences']

print("This python script applies Gaussian Naive Bayes to the dataset.")
print('''Please select one of the following attributes to test:
sex                     -> 0
school                  -> 1
higher education        -> 2
study time              -> 3
G3                      -> 4
absences                -> 5
'''
)

selection = input()
while(True):

        try:
                if int(selection) not in range(6):
                        selection = input("Wrong input. Please select a number fr
om 0 to 5: ")
                else:
                        print(f"Your selected attribute is {attributes[int(select
ion)]}\n")
                        selection = attributes[int(selection)]
                        break
        except:
                selection = input("Wrong input. Please select a number from 0 to
5: ")

print("Please select how many iterations you would like to execute: ")
iterations = input()
while(True):

        if(iterations==0):
                iterations = input("Wrong input. Please select a positive integer
: ")

        try:
                iterations = int(iterations)
                print(f"Iterations: {iterations}\n")
                break
        except:
                iterations = input("Wrong input. Please select a positive integer
: ")
```

```
print("Calculating average score...", end='')

# ################################################ #
# GaussianNB implementation
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
import datetime

begin_time = datetime.datetime.now()

df = pd.read_csv("./data/student-merged.csv")

# Data preprocessing - convert all classified attributes to numeric
for index_label, row_series in df.iterrows():

    if row_series['school'] == 'GP':
        df.at[index_label , 'school'] = 0
    elif row_series['school'] == 'MS':
        df.at[index_label , 'school'] = 1

    if row_series['sex'] == 'F':
        df.at[index_label , 'sex'] = 0
    elif row_series['sex'] == 'M':
        df.at[index_label , 'sex'] = 1

    if row_series['address'] == 'U':
        df.at[index_label , 'address'] = 0
    elif row_series['address'] == 'R':
        df.at[index_label , 'address'] = 1

    if row_series['famsize'] == 'GT3':
        df.at[index_label , 'famsize'] = 0
    elif row_series['famsize'] == 'LE3':
        df.at[index_label , 'famsize'] = 1

    if row_series['Pstatus'] == 'A':
        df.at[index_label , 'Pstatus'] = 0
    elif row_series['Pstatus'] == 'T':
        df.at[index_label , 'Pstatus'] = 1

    if row_series['Mjob'] == 'at_home':
        df.at[index_label , 'Mjob'] = 0
    elif row_series['Mjob'] == 'health':
        df.at[index_label , 'Mjob'] = 1
    elif row_series['Mjob'] == 'services':
        df.at[index_label , 'Mjob'] = 2
    elif row_series['Mjob'] == 'teacher':
        df.at[index_label , 'Mjob'] = 3
    elif row_series['Mjob'] == 'other':
        df.at[index_label , 'Mjob'] = 4
```

```
    if row_series['Fjob'] == 'at_home':
        df.at[index_label , 'Fjob'] = 0
    elif row_series['Fjob'] == 'health':
        df.at[index_label , 'Fjob'] = 1
    elif row_series['Fjob'] == 'services':
        df.at[index_label , 'Fjob'] = 2
    elif row_series['Fjob'] == 'teacher':
        df.at[index_label , 'Fjob'] = 3
    elif row_series['Fjob'] == 'other':
        df.at[index_label , 'Fjob'] = 4

    if row_series['reason'] == 'home':
        df.at[index_label , 'reason'] = 0
    elif row_series['reason'] == 'reputation':
        df.at[index_label , 'reason'] = 1
    elif row_series['reason'] == 'course':
        df.at[index_label , 'reason'] = 2
    elif row_series['reason'] == 'other':
        df.at[index_label , 'reason'] = 3

    if row_series['guardian'] == 'mother':
        df.at[index_label , 'guardian'] = 0
    elif row_series['guardian'] == 'father':
        df.at[index_label , 'guardian'] = 1
    elif row_series['guardian'] == 'other':
        df.at[index_label , 'guardian'] = 2

    if row_series['schoolsup'] == 'no':
        df.at[index_label , 'schoolsup'] = 0
    elif row_series['schoolsup'] == 'yes':
        df.at[index_label , 'schoolsup'] = 1

    if row_series['famsup'] == 'no':
        df.at[index_label , 'famsup'] = 0
    elif row_series['famsup'] == 'yes':
        df.at[index_label , 'famsup'] = 1

    if row_series['paid'] == 'no':
        df.at[index_label , 'paid'] = 0
    elif row_series['paid'] == 'yes':
        df.at[index_label , 'paid'] = 1

    if row_series['activities'] == 'no':
        df.at[index_label , 'activities'] = 0
    elif row_series['activities'] == 'yes':
        df.at[index_label , 'activities'] = 1

    if row_series['nursery'] == 'no':
        df.at[index_label , 'nursery'] = 0
    elif row_series['nursery'] == 'yes':
        df.at[index_label , 'nursery'] = 1
```

```
    if row_series['higher'] == 'no':
        df.at[index_label , 'higher'] = 0
    elif row_series['higher'] == 'yes':
        df.at[index_label , 'higher'] = 1

    if row_series['internet'] == 'no':
        df.at[index_label , 'internet'] = 0
    elif row_series['internet'] == 'yes':
        df.at[index_label , 'internet'] = 1

    if row_series['romantic'] == 'no':
        df.at[index_label , 'romantic'] = 0
    elif row_series['romantic'] == 'yes':
        df.at[index_label , 'romantic'] = 1

# Convert dataframe to numeric in order to be used with GaussianNB
df = df.apply(pd.to_numeric)

# Create inputs
inputs = df.drop(selection, axis='columns')

# Create target
target = df[selection]

score = count = 0
for i in range(iterations):

        X_train, X_test, y_train, y_test = train_test_split(inputs,target,test_si
ze=0.2)

        model = GaussianNB()

        model.fit(X_train, y_train)

        count += 1
        score += model.score(X_test,y_test)

        if count%100==0:
                print(".", end='')


print(f"\nThe average score for the attribute {selection} after {iterations} iter
ations is {score/count}")
print(f"Script completed successfully, time: {datetime.datetime.now() - begin_tim
e}")
```

linearRegression.py

```
attributes = ['sex', 'school', 'higher', 'studytime', 'G3', 'absences']

print("This python script applies Gaussian Naive Bayes to the dataset.")
```

```
print('''Please select one of the following attributes to test:
sex                     -> 0
school                  -> 1
higher education        -> 2
study time              -> 3
G3                      -> 4
absences                -> 5
'''
)

selection = input()
while(True):

        try:
                if int(selection) not in range(6):
                        selection = input("Wrong input. Please select a number fr
om 0 to 5: ")
                else:
                        print(f"Your selected attribute is {attributes[int(select
ion)]}\n")
                        selection = attributes[int(selection)]
                        break
        except:
                selection = input("Wrong input. Please select a number from 0 to
5: ")

print("Please select how many iterations you would like to execute: ")
iterations = input()
while(True):

        if(iterations==0):
                iterations = input("Wrong input. Please select a positive integer
: ")

        try:
                iterations = int(iterations)
                print(f"Iterations: {iterations}\n")
                break
        except:
                iterations = input("Wrong input. Please select an integer: ")

print("Calculating average score...", end='')

# ################################################ #
# LinearRegression implementation
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import datetime

begin_time = datetime.datetime.now()
```

```
df = pd.read_csv("./data/student-merged.csv")

# Data preprocessing - convert all classified attributes to numeric
for index_label, row_series in df.iterrows():

    if row_series['school'] == 'GP':
        df.at[index_label , 'school'] = 0
    elif row_series['school'] == 'MS':
        df.at[index_label , 'school'] = 1

    if row_series['sex'] == 'F':
        df.at[index_label , 'sex'] = 0
    elif row_series['sex'] == 'M':
        df.at[index_label , 'sex'] = 1

    if row_series['address'] == 'U':
        df.at[index_label , 'address'] = 0
    elif row_series['address'] == 'R':
        df.at[index_label , 'address'] = 1

    if row_series['famsize'] == 'GT3':
        df.at[index_label , 'famsize'] = 0
    elif row_series['famsize'] == 'LE3':
        df.at[index_label , 'famsize'] = 1

    if row_series['Pstatus'] == 'A':
        df.at[index_label , 'Pstatus'] = 0
    elif row_series['Pstatus'] == 'T':
        df.at[index_label , 'Pstatus'] = 1

    if row_series['Mjob'] == 'at_home':
        df.at[index_label , 'Mjob'] = 0
    elif row_series['Mjob'] == 'health':
        df.at[index_label , 'Mjob'] = 1
    elif row_series['Mjob'] == 'services':
        df.at[index_label , 'Mjob'] = 2
    elif row_series['Mjob'] == 'teacher':
        df.at[index_label , 'Mjob'] = 3
    elif row_series['Mjob'] == 'other':
        df.at[index_label , 'Mjob'] = 4

    if row_series['Fjob'] == 'at_home':
        df.at[index_label , 'Fjob'] = 0
    elif row_series['Fjob'] == 'health':
        df.at[index_label , 'Fjob'] = 1
    elif row_series['Fjob'] == 'services':
        df.at[index_label , 'Fjob'] = 2
    elif row_series['Fjob'] == 'teacher':
        df.at[index_label , 'Fjob'] = 3
    elif row_series['Fjob'] == 'other':
        df.at[index_label , 'Fjob'] = 4
```

```
    if row_series['reason'] == 'home':
        df.at[index_label , 'reason'] = 0
    elif row_series['reason'] == 'reputation':
        df.at[index_label , 'reason'] = 1
    elif row_series['reason'] == 'course':
        df.at[index_label , 'reason'] = 2
    elif row_series['reason'] == 'other':
        df.at[index_label , 'reason'] = 3

    if row_series['guardian'] == 'mother':
        df.at[index_label , 'guardian'] = 0
    elif row_series['guardian'] == 'father':
        df.at[index_label , 'guardian'] = 1
    elif row_series['guardian'] == 'other':
        df.at[index_label , 'guardian'] = 2

    if row_series['schoolsup'] == 'no':
        df.at[index_label , 'schoolsup'] = 0
    elif row_series['schoolsup'] == 'yes':
        df.at[index_label , 'schoolsup'] = 1

    if row_series['famsup'] == 'no':
        df.at[index_label , 'famsup'] = 0
    elif row_series['famsup'] == 'yes':
        df.at[index_label , 'famsup'] = 1

    if row_series['paid'] == 'no':
        df.at[index_label , 'paid'] = 0
    elif row_series['paid'] == 'yes':
        df.at[index_label , 'paid'] = 1

    if row_series['activities'] == 'no':
        df.at[index_label , 'activities'] = 0
    elif row_series['activities'] == 'yes':
        df.at[index_label , 'activities'] = 1

    if row_series['nursery'] == 'no':
        df.at[index_label , 'nursery'] = 0
    elif row_series['nursery'] == 'yes':
        df.at[index_label , 'nursery'] = 1

    if row_series['higher'] == 'no':
        df.at[index_label , 'higher'] = 0
    elif row_series['higher'] == 'yes':
        df.at[index_label , 'higher'] = 1

    if row_series['internet'] == 'no':
        df.at[index_label , 'internet'] = 0
    elif row_series['internet'] == 'yes':
        df.at[index_label , 'internet'] = 1
```

```
    if row_series['romantic'] == 'no':
        df.at[index_label , 'romantic'] = 0
    elif row_series['romantic'] == 'yes':
        df.at[index_label , 'romantic'] = 1

df = df.apply(pd.to_numeric)

# Create inputs
X = df.drop(selection, axis='columns')

# Create target
y = df[selection]

for i in range(iterations):

        #Split Data 80-20
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)

        #train the algorithm
        lnr = LinearRegression()
        lnr.fit(X_train, y_train)

        y_pred = lnr.predict(X_test)

        # First round the values of predictions, the typecast them as ints
        y_pred = y_pred.round()
        y_pred = y_pred.astype(int)

        #Compare the actual output values with the predicted values
        df = pd.DataFrame({'Actual': y_test.to_numpy().flatten(), 'Predicted': y_
pred.flatten()})

# No need to calculate average from each iteration as the Linear Regression algor
ithm makes the same predictions every time
correct_predictions = total_counts = 0
for index_label, row_series in df.iterrows():
        if(row_series['Actual']==row_series['Predicted']):
                correct_predictions += 1
        total_counts += 1

print(f"\nThe average score for the attribute {selection} after {iterations} iter
ations is {correct_predictions/total_counts}")
print(f"Script completed successfully, time: {datetime.datetime.now() - begin_tim
e}")
```