Αυτόματος υπολογισμός δέντρων απόφασης σε σύστημα δημιουργίας προφίλ δεδομένων

Χαρίσης Αλέξανδρος

Διπλωματική Εργασία

Επιβλέπων: Π. Βασιλειάδης

Ιωάννινα, Μάρτιος 2023

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Ευχαριστίες

Θέλω να ευχαριστήσω κυρίως τους γονείς μου, για τη στήριξη και τη βοήθεια που μου έδειξαν καθ΄ όλη τη διάρκεια της ζωής μου. Δίχως την καθοδήγησή τους δεν θα μπορούσα να είχα φτάσει στο σημείο που βρίσκομαι σήμερα και οτιδήποτε έχω καταφέρει είναι παράγωγο των προσπαθειών και της φροντίδας τους.

Επίσης, θέλω να ευχαριστήσω θερμά τον επιβλέπων καθηγητή μου, τον κύριο Παναγιώτη Βασιλειάδη, που ήταν πάντα ανοιχτός στο να με βοηθήσει και να συζητήσουμε για την ανάπτυξη της διπλωματικής μου, δείχνοντάς μου το μονοπάτι ή δίνοντάς μου το κίνητρο για να τα καταφέρω. Αν και καλός καθηγητής, αποδείχθηκε ακόμα καλύτερος άνθρωπος, με μερικά από τα χαρακτηριστικά του η ανοιχτόμυαλη προσέγγιση επί του κάθε θέματος, η φιλική προσωπικότητά του και η αντιμετώπισή του με σεβασμό προς όλους τους προπτυχιακούς και μεταπτυχιακούς του. Χαίρομαι πολύ που ήμουν κάτω από τη καθοδήγησή του κατά την εμπειρία αυτή.

24 Φεβρουαρίου 2023

Χαρίσης Αλέξανδρος

Περίληψη στα ελληνικά

Την σήμερον ημέρα όλο και περισσότεροι πολίτες ασχολούνται με τον τομέα της στατιστικής ανάλυσης. Η αυτοματοποίησή της καθίσταται όλο και πιο σημαντική καθημερινά. Στα πλαίσια της αυτοματοποίησης της στατιστικής ανάλυσης, βρίσκεται η τρέχουσα διπλωματική εργασία. Χρησιμοποιήθηκε το υπάρχον λογισμικό Pythia, το οποίο προσφέρει μερικές από τις ζητούμενες λειτουργίες, όπως το διάβασμα και την αποθήκευση δεδομένων, τον υπολογισμό διαφόρων απλών στατιστικών στοιχείων για κατηγορικά και αριθμητικά δεδομένα και την αναπαράστασή τους σε μια τελική αναφορά απλού κειμένου. Στα υπάρχοντα στατιστικά στοιχεία, προστέθηκαν ιστογράμματα και δέντρα αποφάσεων. Τα ιστογράμματα μελετούν την κατανομή και τη συχνότητα των αριθμητικών δεδομένων, παράγοντας κάδους με ισομερής εύρη τιμών. Τα δέντρα αποφάσεων προβλέπουν μία τιμή και ταξινομούν τα δεδομένα, προσφέροντας διάφορες πληροφορίες, όπως την ακρίβειά τους, τα στοιχεία κάθε κόμβου, και την αναπαράστασή τους ως εικόνα. Τα στατιστικά τους στοιχεία συμπεριλαμβάνονται στη τελική αναφορά. Επίσης, εμπλουτίστηκε η τελική αναφορά και βελτιώθηκε με την ένταξη νέων τύπων αρχείων εξόδου, όπως markdown, και τη γραφική αναπαράσταση ορισμένων αποτελεσμάτων. Η αρχιτεκτονική του προγράμματος ανοικοδομήθηκε με σκοπό την διευκόλυνση της μελλοντικής συντήρησης και της επεκτασιμότητας. Το λογισμικό γράφτηκε σε Java και για τον υπολογισμό των στατιστικών στοιχείων και την επεξεργασία των συνόλων δεδομένων χρησιμοποιήθηκε το Apache Spark.

Λέξεις Κλειδιά: Επιστήμη δεδομένων, Ανάλυση δεδομένων, Δέντρα αποφάσεων, Java, Apache Spark

Abstract

Nowadays more and more citizens are engaged in the field of statistical analysis. This thesis aims to develop a tool that automates some of its procedures. The existing Pythia software was used, which offers some of the requested functionality, such as reading and storing data, calculating various simple statistics on the given dataset, and representing it in a plain text report. To complement the existing statistical evidence for the data set, histograms and decision trees were added as parts of the data set profile. Histograms study the distribution and frequency of numerical data, producing bins with equal ranges of values. Decision trees predict a value and classify the data, offering various information such as the accuracy of the tree, information about each node, and its representation as an image. Their statistics are included in the final report. Also, the final report was enriched and improved by including new output file types, such as markdown, and graphical representation of some of the results. The architecture of the program has been refactored to facilitate future maintenance and extensibility. The software was written in Java and the computation of the statistics and the manipulation of the datasets was done through the use of Apache Spark.

Keywords: Data science, Statistical analysis, Decision trees, Java, Apache Spark

Πίνακας περιεχομένων

Κεφάλαιο 1. Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Ο σκοπός της στατιστικής ανάλυσης είναι η ανάλυση και η ερμηνεία των δεδομένων, μέσω της χρήσης μαθηματικών μεθόδων, για την απόκτηση γνώσεων και την εξαγωγή συμπερασμάτων σχετικά με τον πληθυσμό από τον οποίο ελήφθησαν τα δεδομένα. Η στατιστική ανάλυση μπορεί να χρησιμοποιηθεί για τον έλεγχο υποθέσεων, τον προσδιορισμό της ισχύος των σχέσεων μεταξύ των μεταβλητών, την πραγματοποίηση προβλέψεων και την αξιολόγηση της αποτελεσματικότητας των παρεμβάσεων ή της επεξεργασίας των δεδομένων. Η στατιστική ανάλυση είναι ένα ισχυρό εργαλείο για τη λήψη αποφάσεων σε ένα ευρύ φάσμα τομέων, συμπεριλαμβανόμενης της ιατρικής, της ψυχολογίας, της οικονομίας και των επιχειρήσεων.

Η αυτοματοποίηση της στατιστικής ανάλυσης περιλαμβάνει τη χρήση προγραμμάτων ηλεκτρονικών υπολογιστών και αλγορίθμων για την εκτέλεση εργασιών ανάλυσης δεδομένων που συνήθως απαιτούσαν ανθρώπινη τεχνογνωσία. Ο σκοπός της αυτοματοποίησης της στατιστικής ανάλυσης είναι να αυξήσει την αποτελεσματικότητα, να μειώσει το ανθρώπινο λάθος και να κάνει την ανάλυση δεδομένων πιο προσιτή σε ένα ευρύτερο φάσμα χρηστών. Η αυτοματοποίηση της στατιστικής ανάλυσης μπορεί να είναι ιδιαίτερα χρήσιμη σε κλάδους όπως η χρηματοδότηση, η υγειονομική περίθαλψη και η μεταποίηση, όπου παράγονται και αναλύονται μεγάλοι όγκοι δεδομένων σε τακτική βάση. Μπορεί επίσης να είναι επωφελές για ερευνητές που πρέπει να αναλύσουν τα δεδομένα γρήγορα και με ακρίβεια.

Η αυτοματοποίηση της στατιστικής ανάλυσης έχει σημειώσει σημαντική πρόοδο τα τελευταία χρόνια, χάρη στη πρόοδο στη μηχανική μάθηση, τη τεχνητή νοημοσύνη και την επιστήμη δεδομένων. Σήμερα, υπάρχουν πολλά διαθέσιμα εργαλεία και πλατφόρμες που επιτρέπουν στους χρήστες να αυτοματοποιήσουν διάφορες πτυχές της στατιστικής

ανάλυσης, συμπεριλαμβανομένου του καθαρισμού δεδομένων, της οπτικοποίησης δεδομένων, του ελέγχου υποθέσεων και της προγνωστικής μοντελοποίησης. Πολλά από αυτά τα εργαλεία και πλατφόρμες έχουν σχεδιαστεί για να είναι φιλικά προς τον χρήστη, με εύχρηστες διεπαφές και προκαθορισμένα πρότυπα που διευκολύνουν τους χρήστες να ξεκινήσουν την αυτοματοποιημένη ανάλυση.

Ωστόσο, τα εργαλεία αυτά συνήθως είναι διαδραστικά, δηλαδή απαιτούν την παραμετροποίηση και την δημιουργία προτύπων από τον χρήστη. Το πρόγραμμα με το οποίο θα ασχοληθούμε στην παρούσα διπλωματική εργασία, το Pythia, στοχεύει στην πλήρως αυτοματοποιημένη παραγωγή ενός στατιστικού προφίλ, με την ελάχιστη δυνατή παρέμβαση και παραμετροποίηση από τον χρήστη, και την παραγωγή μίας εκτενούς, αναλυτικής και ευπαρουσίαστης αναφοράς πάνω στα παραγόμενα στατιστικά μοντέλα. Η κύρια και βασική ιδεολογία που ακολουθεί το πρόγραμμα Pythia είναι η ευχρηστία του και η προσαρμοστικότητά του, καθώς παρέχει λίγες και απλές λειτουργείες για τον τελικό χρήστη και μπορεί να χρησιμοποιηθεί σαν εξωτερική βιβλιοθήκη σε άλλα προγράμματα.

Το Pythia είναι ένα ήδη υπάρχον και αναπτυσσόμενο λογισμικό που προσφέρει στον χρήστη τη δυνατότητα εγγραφής ενός συνόλου δεδομένων, την παραγωγή ενός στατιστικού προφίλ και την παρουσίασή του σε μια αναφορά απλού κειμένου. Οι παραγόμενες μετρικές αποτελούν στατιστικά όπως η διάμεσος, η ελάχιστη και η μέγιστη τιμή, ο μέσος όρος, το πλήθος των στοιχείων και η τυπική απόκλιση για κάθε αριθμητική ή κατηγορική μεταβλητή, ενώ παράλληλα υπολογίζονται δέντρα αποφάσεων και συσχετίσεις μόνο για τις αριθμητικές μεταβλητές.

Η παρούσα διπλωματική εργασία ασχολείται με τη προσθήκη ιστογραμμάτων, την ανακατασκευή του ελλαττωματικού υπολογισμού των δέντρων αποφάσεων, την προσθήκη μίας ευανάγνωστης αναφοράς και την αναδιοργάνωση της αρχιτεκτονικής δομής του.

Στο Pythia πλέον παράγονται αυτόματα ιστογράμματα για κάθε αριθμητική μεταβλητή, υπολογίζοντας το πλήθος των τιμών σε ορισμένα εύρη τιμών, καθώς και το πλήθος των μη έγκυρων τιμών που εμφανίστηκαν για τη συγκεκριμένη μεταβλητή. Τα δέντρα αποφάσεων δημιουργούνται μόνο για τις μεταβλητές πρόβλεψης που έχει ορίσει ο χρήστης. Αποδεκτές μεταβλητές πρόβλεψης για ένα δέντρο αποφάσεων είναι αυτές που έχουν ένα πολύ συγκεκριμένο και πεπερασμένο σύνολο τιμών, όπως οι μήνες ή οι μέρες της εβδομάδας. Το Pythia παρέχει στον χρήστη τη δυνατότητα να δημιουργήσει ο ίδιος μία νέα μεταβλητή, η οποία βασίζεται στις τιμές των υπόλοιπων μεταβλητών, με τη

χρήση προσαρμοσμένων κανόνων. Στη διπλωματική αυτή εργασία θα αναφερόμαστε σε αυτές τις μεταβλητές ως "*labeled*". Σε αντίθεση, τα χαρακτηριστικά που λαμβάνουν μέρος στον υπολογισμού του δέντρου αποφάσεων, μπορεί να είναι οποιαδήποτε αριθμητική ή κατηγορική μεταβλητή ενός συνόλου δεδομένων, με τη δυνατότητα να τα ορίσει ο χρήστης.

Για τη προσφορά μιας πιο ευπαρουσίαστης και ευανάγνωστης τελικής αναφοράς προς τον τελικό χρήστη, το εργαλείο Pythia έχει πλέον τη δυνατότητα παραγωγής της τελικής αναφοράς σε αρχείου τύπου markdown. Η επέκταση αυτή αποτελεί σημαντική βελτίωση στην παρουσίαση της τελικής αναφοράς, αφού το markdown είναι μία ευρέως χρησιμοποιούμενη και απλή γλώσσα κειμένου που μπορεί να υποστηρίξει από λίστες και πίνακες έως και γραφικές αναπαραστάσεις.

Η αναδιοργάνωση της δομής της αρχιτεκτονικής του λογισμικού έγινε μέσω της χρήσης προγραμματιστικών σχεδιαστικών προτύπων, καθώς και με την τροποποίηση και απλοποίηση κομματιών του υπάρχοντος πηγαίου κώδικα. Δίχως να προστεθούν καινούργιες λειτουργίες για τον τελικό χρήστη, αυξήθηκε η επεκτασιμότητα και διευκολύνεται η μελλοντική συντήρηση και ανάπτυξή του.

Στο υπόλοιπο σκέλος της διπλωματικής αυτής εργασίας εξετάζουμε το αντίκτυπο των αλλαγών αυτών στο εργαλείο Pythia. Εκτελώντας απομονωμένες χρονομετρήσεις στις διάφορες λειτουργίες που εισάγαμε, αναλύουμε τον χρόνο που χρειάζεται για την παραγωγή τους, κάτω από διαφορετικές συνθήκες. Παράλληλα, εξετάζουμε τον χρόνο υπολογισμού κάθε στατιστικού μοντέλου ξεχωριστά, καθώς και τον χρόνο κατασκευής του στατιστικού προφίλ, κάτω από τις ίδιες συνθήκες, και συγκρίνουμε τα αποτελέσματα.

1.2 Οργάνωση του τόμου

Η παρούσα διπλωματική εργασία αναλύεται εκτενέστερα στα επόμενα κεφάλαια, με αυτό το κεφάλαιο να προσφέρει μία προεπισκόπηση του θέματος της καθεμίας.

Στο δεύτερο κεφάλαιο γίνεται η περιγραφή των λειτουργιών που θα προσφέρει το σύστημα που αναπτύχθηκε, η ανάλυση των απαιτήσεων χρήστη, καθώς και το θεωρητικό και τεχνολογικό υπόβαθρο που απαιτείται για τη κατανόηση των υποκείμενων δομών της εργασίας.

Ο ορισμός των προβλημάτων του συστήματος και ο τρόπος αντιμετώπισής τους, η σχεδίαση και η αρχιτεκτονική του λογισμικού, η επεκτασιμότητά του, καθώς και η

διαδικασία εγκατάστασης και ελέγχου του λογισμικού, περιγράφονται στο τρίτο κεφάλαιο.

Η απόδοση του συστήματος μέσω πειραμάτων και μετρήσεων των υποσυστημάτων βρίσκεται στο τέταρτο κεφάλαιο, παρέχοντας γραφικές παραστάσεις για τη σύγκριση των αποτελεσμάτων.

Η αξιολόγηση της διπλωματικής εργασίας και των αποτελεσμάτων των πειραμάτων λαμβάνουν χώρο στο πέμπτο κεφάλαιο, όπου και παραθέτουμε τυχόν μελλοντικές επεκτάσεις.

Τέλος, γίνεται αναφορά στη βιβλιογραφία που χρησιμοποιήθηκε για την διεκπεραίωση της διπλωματικής αυτής εργασίας.

Κεφάλαιο 2. Περιγραφή Θέματος

2.1 Στόχος της εργασίας

Στη παρούσα διπλωματική εργασία, σκοπός είναι η πλήρως αυτόματη παραγωγή ενός στατιστικού προφίλ ενός συνόλου δεδομένων, που περιλαμβάνει περιγραφικά στοιχεία για κάθε πεδίο και τις συσχετίσεις μεταξύ τους, δέντρα αποφάσεων, καθώς και την αναπαράσταση των δεδομένων κάθε πεδίου σε ιστόγραμμα. Ακολουθούμενη από τη δημιουργία της τελικής αναφοράς, η οποία είναι ευπαρουσίαστη και εύκολα κατανοητή από τον τελικό χρήστη. Για την επιτυχία αυτού του στόχου, αναλαμβάνουμε την συντήρηση και την επέκταση ενός υπάρχοντος λογισμικού (Pythia, Available at: https://github.com/DAINTINESS-Group/Pythia) που προσφέρει μερικές από τις βασικές λειτουργίες που θέλουμε να παρέχονται.

Οι απαιτήσεις του συστήματος που θα αναπτύξουμε είναι:

- Η απλοποίηση του κώδικα με την μείωση των εξαρτήσεων από εξωτερικές βιβλιοθήκες, δηλαδή η αφαίρεση της βιβλιοθήκης Lombok.
- Η αυτόματη εξαγωγή ιστογραμμάτων για τις κολόνες του συνόλου δεδομένων.
- Η αναπαράσταση των αποτελεσμάτων και της δομής των δέντρων αποφάσεων (κόμβοι, διασυνδέσεις) σε αντίστοιχες κλάσεις, για να διευκολυνθεί η μελλοντική τους αξιοποίηση.
- Κατάργηση της υπάρχουσας και δημιουργία εκ νέου της διαδικασίας υπολογισμού των δέντρων αποφάσεων. Ο χρήστης θα μπορεί να επιλέξει ποια πεδία θέλει να εμπλακούν στη δημιουργία του ενός δέντρου, ενώ τα πεδία που έχουν χαρακτηριστεί με κανόνες δε θα λαμβάνουν μέρος στον υπολογισμό.
- Η βελτίωση της τελικής αναφοράς, ώστε ο τελικός χρήστης να μπορεί εύκολα να διευκρινίσει τα κύρια στοιχεία που τον ενδιαφέρουν. Η παρούσα έκδοση αναφοράς γίνεται σε μορφή απλού κειμένου (txt ή Json) και δεν είναι ευανάγνωστη.

2.2 Σχετικές εργασίες και τεχνολογίες

2.2.1 Τεχνολογικό Υπόβαθρο

Σε αυτό το κεφάλαιο θα αναλύσουμε το τεχνολογικό υπόβαθρο αυτής της εργασίας, δηλαδή το λογισμικό και τις βιβλιοθήκες που χρησιμοποιούνται, καθώς και τι προσφέρουν.

2.2.1.1 Pythia - Το Υπάρχον Λογισμικό

Όπως έχει προαναφερθεί, η παρούσα εργασία αναπτύσσεται πάνω σε ήδη υπάρχον λογισμικό, το Pythia (Available at: https://github.com/DAINTINESS-Group/Pythia), το οποίο παρέχει ορισμένες από τις λειτουργίες που επιθυμούμε.

Αυτές είναι:

- Ο χρήστης μπορεί να δώσει ως είσοδο (input) το σύνολο δεδομένων που επιθυμεί. Είναι δυνατή η εγγραφή οποιουδήποτε συνόλου δεδομένων, δηλώνοντας όμως προγραμματιστικά τα ονόματα κάθε πεδίου καθώς και τον αντίστοιχο τύπο τους (int, double, dateTime, Boolean, enum of class labels, κλπ.). Σε αυτή την έκδοση υποστηρίζονται μόνο τριών ειδών αρχεία, ".csv", ".tsv" και ". json").
- Για κάθε πεδίο υπολογίζεται αυτόματα η αντίστοιχη ελάχιστη, μέγιστη, μέση και διάμεση τιμή, το πλήθος κάθε μοναδικής τιμής, η τυπική απόκλιση, καθώς και η συσχέτιση του με τα υπόλοιπα πεδία.
- Κατά την επιθυμία του χρήστη, είναι δυνατή η δήλωση κανόνων labeling για μια στήλη, δηλαδή ο χαρακτηρισμός των τιμών αυτής της στήλης ως προς τους δηλωμένους κανόνες, για τη δημιουργία μιας καινούργιας labeled στήλης.
- Με βάση τη νέα δηλωμένη στήλη, μπορεί να υπολογιστεί ένα απλοϊκό δένδρο απόφασης. Ο χρήστης μπορεί να λάβει πληροφορίες όπως την ακρίβεια του δέντρου, τα χαρακτηριστικά του, καθώς και την απεικόνισή του σε μορφή κειμένου.
- Μετά τον υπολογισμό των παραπάνω δεδομένων, παράγεται μία απλοϊκή αναφορά αυτών σε αρχείο τύπο ".txt" ή ".json".

2.2.1.2 Project Lombok

Το Project Lombok [Lomb22] είναι μια βιβλιοθήκη για Java η οποία προσφέρει annotations που σκοπό έχουν να αυτοματοποιήσουν και να ελαχιστοποιήσουν τις γραμμές κώδικα που χρειάζεται να γράψει και να βλέπει ένας προγραμματιστής. Αντικαθιστούν κομμάτια κώδικα που είναι απλά, όπως constructors, getters και setters, με annotations (όπως @Getter) πάνω από κάθε πεδίο ή κλάση, και κρατάει καθαρή την οθόνη από κώδικα που είναι συνηθισμένος, επαναλαμβανόμενος ανάμεσα σε κλάσεις με μικρές διαφορές ή αποτελεί «θόρυβο».

2.2.1.3 Apache Spark

Το Apache Spark είναι μια μηχανή επεξεργασίας δεδομένων που μπορεί να εκτελέσει γρήγορα εργασίες επεξεργασίας σε πολύ μεγάλα σύνολα δεδομένων, και μπορεί επίσης να διανέμει εργασίες σε πολλαπλούς υπολογιστές, είτε από μόνο του είτε σε συνδυασμό με άλλα κατανεμημένα υπολογιστικά συστήματα. Αυτές οι δύο ιδιότητες είναι βασικές για τον κόσμο των μεγάλων δεδομένων και της μηχανικής μάθησης, οι οποίοι απαιτούν τη συγκέντρωση τεράστιας υπολογιστικής ισχύος για να επεξεργαστούν μεγάλα σύνολα δεδομένων. Επίσης, το Apache Spark αφαιρεί ένα βάρος από τις υποχρεώσεις του προγραμματιστή, εφόσον προσφέρει ένα εύχρηστο API που αφαιρεί ένα μεγάλο μέρος της δουλειάς που χρειάζονται τα κατανεμημένα συστήματα και η επεξεργασία μεγάλων συνόλων δεδομένων [Apac22].

Το Apache Spark υποστηρίζει πολλαπλές γλώσσες, όπως Java, Python, Scala και R, και υποστηρίζει και SQL, ροή δεδομένων, μηχανική μάθηση και επεξεργασία γραφημάτων. Διάφορες επιχειρήσεις όπως τράπεζες, εταιρείες τηλεπικοινωνιών, κυβερνήσεις και πολλοί από τους μεγάλους τεχνολογικούς γίγαντες όπως η Apple, το Facebook, η IBM και η Microsoft, χρησιμοποιούν το Apache Spark.

Στην καρδιά του Apache Spark βρίσκονται τα ανθεκτικά κατανεμημένα σύνολα δεδομένων (ή αλλιώς, Resilient Distributed Datasets, RDD). Ένα RDD είναι μια συλλογή στοιχείων, με ανοχή σε σφάλματα, που μπορούν να λειτουργήσουν παράλληλα. Αποτελεί τη θεμελιώδης δομή δεδομένων που χρησιμοποιεί η Spark και οι εγγραφές του μπορούν να βρίσκονται σε πολλαπλούς κόμβους. Μερικά από τα χαρακτηριστικά που κάνουν τα RDD να ξεχωρίζουν είναι:

- Η ετεροχρονισμένη αξιολόγηση, δηλαδή ο υπολογισμός των μετασχηματισμών που τους αναθέτεται να γίνεται σε μεταγενέστερο χρόνο, όταν χρειάζεται να γίνει

μια ενέργεια, ώστε να εκμεταλλευτούμε τη πιθανή απόρριψη κάποιων δεδομένων σε έναν διασωληνωμένο υπολογισμό.

- Ο υπολογισμός στη μνήμη του υπολογιστή (RAM). Τα δεδομένα δε χρειάζονται εγγραφή και διάβασμα από τον σκληρό δίσκο, και αναλύονται πιο αποτελεσματικά.
- Η ανοχή σε σφάλματα. Τα RDD μπορούν να ανοικοδομήσουν χαμένα δεδομένα ακολουθώντας τα ίχνη της γενεαλογίας των δεδομένων που δημιουργούν οι μεταμορφώσεις. Για να επιτευχθεί ανοχή σφαλμάτων για τα δημιουργούμενα RDD, τα παραγόμενα δεδομένα αναπαράγονται μεταξύ διαφόρων κόμβων εργασίας του Apache Spark.
- Τα αμετάβλητα δεδομένα. Τα δεδομένα ενός RDD είναι αμετάβλητα, δηλαδή δεν αλλάζουν λόγω ενημερώσεων. Αυτό σημαίνει πως πολλαπλά νήματα μπορούν να έχουν πρόσβαση και να επεξεργάζονται τα ίδια δεδομένα, δίχως την ανάγκη συγχρονισμού μεταξύ των κόμβων του κατανεμημένου περιβάλλοντος του Apache Spark.
- Η διαμέριση. Υπάρχουν συλλογές από διάφορα στοιχεία δεδομένων με τεράστιους όγκους που δε μπορούν να χωρέσουν σε έναν μόνο κόμβο και πρέπει να κατανεμηθούν σε πολλούς. Το Spark κάνει αυτόματα αυτή τη κατάτμηση των RDD και τα διανέμει.

Το Apache Spark, όμως, προσφέρει και άλλα δύο API για την διαχείριση μεγάλων δεδομένων, τα Datasets και τα DataFrames. Είναι χτισμένα πάνω στα RDD και για αυτό τον λόγο καθίσταται εύκολη η εναλλαγή μεταξύ των δομών δεδομένων RDD, DataFrames και Datasets.

Ένα DataFrame είναι παρόμοιο με έναν πίνακα μιας βάσης δεδομένων, όπου κάθε στήλη έχει έναν συγκεκριμένο τύπο δεδομένων (ακέραιο, συμβολοσειρά, ημερομηνία, κλπ.) [DWDL20]. Κάθε DataFrame αποτελείται από ένα Dataset με Row αντικείμενα, αλλά και από πρόσθετες πληροφορίες για τον τύπο κάθε στήλης. Ένα αντικείμενο Row είναι ένα generic untyped JVM αντικείμενο που περιέχει διαφορετικούς τύπους πεδίων και αποτελεί περιτύλιγμα γύρω από βασικούς τύπους δεδομένων, όπως ακέραιους ή συμβολοσειρές [KKWZ15]. Αντιθέτως, ένα Dataset είναι δομή δεδομένων που αποτελεί μια επέκταση του DataFrame API. Ένα Dataset είναι μια strongly typed, αμετάβλητη συλλογή αντικειμένων (όπως μια κλάση στη Java) που αντιστοιχίζονται σε σχεσιακό σχήμα [DWDL20].

Τα DataFrames βγάζει νόημα να χρησιμοποιηθούν στις γλώσσες Python και R, γιατί οι τύποι συνάγονται δυναμικά ή εκχωρούνται κατά την εκτέλεση και όχι κατά τη διάρκεια του χρόνου μεταγλώττισης. Αντίθετα, τα Datasets βγάζει νόημα να χρησιμοποιηθούν στην Scala και στην Java, γιατί οι τύποι συνδέονται με μεταβλητές και αντικείμενα κατά τον χρόνο της μεταγλώττισης [DWDL20]. Οι καινούργιες αυτές δομές χρησιμοποιούνται κυρίως στο πακέτο Spark MLlib, όπου περιέχει δημοφιλείς αλγορίθμους μηχανικής μάθησης, και στο Spark SQL, όπου μπορούν να υποβληθούν ερωτήματα τύπου SQL στα δεδομένα.

Καθένα από τα RDD, τα DataFrames και τα Datasets χρησιμοποιούνται σε διαφορετικές περιπτώσεις, παρόλο που τα DataFrames και τα Datasets έχουν σημαντικές ομοιότητες. Κύρια χαρακτηριστικά που πρέπει να τονίσουμε είναι:

- Ο προγραμματιστής χρειάζεται να βελτιστοποιήσει τις πράξεις στα RDD με δικό του κώδικα, ενώ στα DataFrames και στα Datasets γίνεται αυτόματα από τον βελτιστοποιητή του Apache Spark. Επίσης τα Datasets και τα DataFrames προσφέρουν, εύκολα και γρήγορα, απόδοση μνήμης και ταχύτητα, σε σχέση με τα RDD.
- Τα RDD χρειάζονται χειροκίνητη προσθήκη του σχήματος των δεδομένων, ενώ τα DataFrames και τα Datasets το αναγνωρίζουν αυτόματα.
- Τα DataFrames και τα Datasets παρέχουν μια πιο δομημένη αναπαράσταση των δεδομένων, που διευκολύνει την επεξεργασία των δεδομένων κατά όνομα ή στήλη.
- Σε απλές πράξεις, όπως η ομαδοποίηση των δεδομένων, τα DataFrames είναι πιο γρήγορα, ακολουθούμενα από τα Datasets και έπειτα από τα RDD.
- Τα Datasets είναι πιο παραγωγικά για έναν προγραμματιστή, γιατί γλυτώνουν χρόνο εντοπίζοντας σφάλματα τύπου κατά τη μεταγλώττιση και όχι κατά τον χρόνο εκτέλεσης.
- Τέλος, άμα η επεξεργασία των δεδομένων μας απαιτεί εκφράσεις υψηλού επιπέδου, φίλτρα, χάρτες, συνάθροιση, μέσους όρους, άθροισμα, ερωτήματα SQL και χρήση συναρτήσεων λάμδα, τότε χρησιμοποιούμε DataFrames ή Datasets.

2.2.1.4 Java & JVM

Η Java είναι μία από τις πιο γνωστές προγραμματιστικές γλώσσες που υπάρχουν. Χρησιμοποιείται παγκοσμίως σε διάφορους τομείς όπως αυτόνομες εφαρμογές επιφάνειας εργασίας, σε εφαρμογές ιστού, σε εταιρικές εφαρμογές, σε επιστημονικές

εφαρμογές, ως διακομιστής ιστού, σε ενσωματωμένα συστήματα, στον χρηματοπιστωτικό κλάδο, σε εργαλεία λογισμικού, σε τεχνολογίες μεγάλων δεδομένων, καθώς και σε κινητές συσκευές, αφού είναι η επίσημη γλώσσα ανάπτυξης σε Android.

Αυτό που καθιστά τη Java τόσο δημοφιλής από τη στιγμή που δημιουργήθηκε έως και σήμερα, είναι η αντικειμενοστράφειά της. Τα αντικείμενα, ή αλλιώς κλάσεις, ωθούν και επιτρέπουν στον προγραμματιστή να σχεδιάσει και να υλοποιήσει μεγάλα και περίπλοκα συστήματα, τα οποία έχουν μια εύκολα κατανοητή δομή, είναι επεκτάσιμα και συντηρήσιμα. Σε αντίθεση με άλλες γλώσσες όπως η C και η C++, η Java προσφέρει διάφορα εργαλεία αποτροπής προγραμματικών λαθών, όπως η αυτοματοποιημένη συλλογή και διαγραφή των μεταβλητών που δεν χρησιμοποιούνται περαιτέρω (Garbage Collector). Κατά αυτόν τον τρόπο, ένας προγραμματιστής μπορεί να αφιερώσει τον χρόνο του σε άλλες διαδικασίες, και είναι πιο δύσκολο να εντάσσει προβλήματα μνήμης. Επίσης, η Java αναπτύσσεται και συντηρείται από την ιδιωτική εταιρία Oracle, η οποία είναι η τρίτη μεγαλύτερη εταιρεία λογισμικού στον κόσμο ως προς τα έσοδα και την κεφαλαιοποίηση της αγορά. Μια επιχείρηση που θέλει να ξεκινήσει την ανάπτυξη ενός νέου λογισμικού, είναι αρκετά πιθανό να συνυπολογίσει τη Java στις επιλογές της, γιατί η Oracle προσφέρει την πολυχρόνια υποστήριξή της ως μεγάλη εταιρία σε ορισμένες από τις εκδόσεις της Java. Επιπρόσθετα, οι καινούργιες εκδόσεις της Java είναι συμβατές με προηγούμενες εκδόσεις της, επιτρέποντας εύκολα την αναβάθμισή της σε υπάρχουσες εφαρμογές. Λόγω αυτών και της μεγάλης απήχησής της, η Java αποτελεί μία αξιόπιστη, ασφαλής και ισχυρή γλώσσα προγραμματισμού την σήμερον ημέρα, και για το σύντομο μέλλον.

Επιπλέον, η Java εκμεταλλεύεται τα χαρακτηριστικά του JVM (Java Virtual Machine), δηλαδή το περιβάλλον μεταγλώττισής και εκτέλεσής της. Το JVM είναι ένα ισχυρό εργαλείο ανοιχτού κώδικα, το οποίο μπορεί να εγκατασταθεί και να τρέξει σε πολλά από τα πιο γνωστά λειτουργικά συστήματα, όπως Windows, Linux, MacOS και Android, παρά την διαφορετική αρχιτεκτονική του καθενός. Αυτό επιτρέπει τη Java να έχει ένα μεγάλο εύρος χρήσης, επειδή ο ίδιος κώδικας μπορεί να τρέξει σε όλα τα συστήματα που είναι συμβατά με το JVM, δίχως να χρειάζονται διαφορετικές υλοποιήσεις και παραμετροποιήσεις. Το JVM είναι υπεύθυνο για βελτιστοποιήσεις του γραμμένου κώδικα κατά τη διάρκεια της μεταγλώττισης, ενώ το ίδιο παράλληλα γίνεται όλο και πιο εκλεπτυσμένο στις ενέργειές του με κάθε καινούργια έκδοση.

2.2.1.5 Maven

Κατά τη διάρκεια ανάπτυξης μιας εφαρμογής, παρουσιάζονται πολλές χρονοβόρες διαδικασίες, όπως η κατασκευή και η μεταγλώττιση του πηγαίου κώδικα και η προσθήκη καινούργιων εξαρτήσεων χειροκίνητα. Το Maven είναι ένα ισχυρό εργαλείο διαχείρισης εφαρμογών Java που αυτοματοποιεί αυτές και άλλες διαδικασίες. Μέσω ενός αρχείου pom.xml ο προγραμματιστής μπορεί να προσδιορίσει τα χαρακτηριστικά υλοποίησης μιας εφαρμογής, όπως την έκδοση της προγραμματιστικής γλώσσας, τον φάκελο του πηγαίου κώδικα, τις εξαρτήσεις, διαφορετικά προφίλ κατασκευής, τον στόχο κ.α.

Το κύριο χαρακτηριστικό του Maven είναι η εύκολη προσθήκη καινούργιων εξαρτήσεων (εξωτερικών βιβλιοθηκών). Ο προγραμματιστής χρειάζεται μόνο να προσθέσει το αναγνωστικό της καινούργιας εξάρτησης στο αρχείο pom.xml και αυτή θα προστεθεί αυτόματα στην εφαρμογή, δίχως περαιτέρω βήματα. Τα αναγνωριστικά αυτά μπορούν να βρεθούν στο κεντρικό διαδικτυακό αποθετήριο του Maven, που είναι ουσιαστικά μια τεράστια και συνεχώς αυξανόμενη συλλογή εξωτερικών βιβλιοθηκών.

Επιπλέον, το Maven επιβάλλει σε κάθε εφαρμογή που το χρησιμοποιεί να έχει μια κοινή δομή φακέλων, όπως φαίνεται στον *Πίνακα 1*.

Βασικά μέρη της τυπικής δομής φακέλων του Maven	
pom.xml	Αρχείο περιγραφής της εφαρμογής
/src/main/java	Φάκελος του πηγαίου κώδικα
/src/main/resources	Φάκελος των πόρων της εφαρμογής
/src/test/java	Φάκελος του κώδικα εξέτασης της εφαρμογής
/target	Φάκελος κατασκευής της εφαρμογής (στόχος)

Πίνακας 1

Βλέπουμε πως κάθε κατηγορία όπως ο πηγαίος κώδικας και ο στόχος, βρίσκονται σε διαφορετικά φάκελο, και άρα είναι πιο οργανωμένα.

Το Maven προσφέρει, επίσης, τη δυνατότητα εύκολης εργασίας σε πολλά έργα ταυτόχρονα, ενώ παράλληλα είναι σε θέση να δημιουργήσει οποιονδήποτε αριθμό εφαρμογών σε προκαθορισμένους τύπους αρχείων, όπως JAR και WAR.

Τέλος, το Maven μπορεί να εγκαταστήσει και να δημοσιεύσει τον πακεταρισμένο κώδικα σε ένα τοπικό ή απομακρυσμένο αποθετήριο, με αυτοματοποιημένες αναφορές που περιέχουν πληροφορίες όπως την λίστα των εξαρτήσεων, προβλήματα που περιμένουν αντιμετώπιση, την άδεια του λογισμικού, την ομάδα πίσω από το έργο, κ.α.

2.2.2 Θεωρητικό Υπόβαθρο

2.2.2.1 Σύνολα Δεδομένων

Ένα σύνολο δεδομένων δεν είναι τίποτε άλλο από πληροφορίες σε μορφή εγγραφών. Το σύνολο δεδομένων δεν αποτελεί μια καινούργια έννοια (παρόλο που ακούγεται συχνά στον μοντέρνο κόσμο της τεχνολογίας), αλλά υπήρχε από τα αρχαία χρόνια σε διάφορες μορφές. Για παράδειγμα οι Ρωμαίοι υποχρεούνταν κάθε πέντε χρόνια να γυρίσουν στη πόλη τους, ώστε να μετρηθούν στη καταγραφή της περιοχής τους. Η καταγραφή αυτή περιείχε την εγγραφή κάθε ατόμου, καθώς και την περιουσία του, και βοηθούσε με την διαχείριση του λαού και τον καθορισμό των φόρων. Συνεχείς τέτοιες καταγραφές αποτελούν ένα σύνολο δεδομένων. Στη σήμερον εποχή, που τα σύνολα δεδομένων αναπαρίστανται συνήθως στη μορφή πίνακα, κάθε στήλη αντιπροσωπεύει μια συγκεκριμένη μεταβλητή και κάθε σειρά αντιστοιχεί σε μια δεδομένη εγγραφή του εν λόγω συνόλου δεδομένων. Το σύνολο δεδομένων παραθέτει τιμές για καθεμία από τις μεταβλητές, όπως για παράδειγμα το ύψος και το βάρος ενός αντικειμένου, για κάθε μέλος του συνόλου δεδομένων [SnMR12].

Ένα σύνολο δεδομένων μπορεί να αποτελείται από λίγες εγγραφές μετρημένες στα δάχτυλα, έως και τεράστια σύνολα δισεκατομμυρίων και τρισεκατομμυρίων εγγραφών. Αυτά τα δεδομένα μπορεί να περιγράφουν δημοσιεύσεις διαφόρων ανθρώπων, τάσεις, αλλαγές καιρού, ακόμα και εμπορικές και τραπεζικές συναλλαγές. Πολλές από τις ενέργειες που λαμβάνουν χώρο στον κόσμο (είτε στο διαδίκτυο, είτε σε συσκευές τεχνολογίας, όπως η καταγραφή των στοιχείων ενός πελάτη ενός λογιστή σε μια βάση δεδομένων), καταγράφονται και συλλέγονται σε σύνολα δεδομένων με σκοπό την αξιοποίησή τους. Κάθε σύνολο δεδομένων μπορεί να προσφέρει κάτι διαφορετικό, εφόσον περιέχει διαφορετικές πληροφορίες που μπορούν να αξιοποιηθούν ανάλογα τις ανάγκες του χρήστη.

Με την άνοδο της εξόρυξης δεδομένων και της τεχνητής νοημοσύνης, γίνεται όλο και πιο εύκολο να εξάγουμε σημαντικά δεδομένα και παρατηρήσεις που θα συμβάλλουν στην ανάπτυξη μιας συγκεκριμένης πτυχής του κόσμου. Ένας αναλυτής μπορεί να συσχετίσει μερικά δεδομένα μεταξύ τους ή να κατηγοριοποιήσει τύπους δεδομένων για να δημιουργήσει καινούργια και σχετικά σύνολα δεδομένων που θα υποστηρίξουν σημαντικές επιχειρηματικές διαδικασίες, όπως οικονομικές μετρήσεις, συναλλαγές πωλήσεων ή την προώθηση αγαθών. Στα επιστημονικά και στατιστικά επαγγέλματα, τα σύνολα δεδομένων μπορούν να βοηθήσουν επαγγελματίες, όπως βιολόγους να αναλύσουν πληροφορίες σχετικά με το περιβάλλον ή το κλίμα μιας περιοχής. Επιπλέον,

καινούργιοι αλγόριθμοι στη τεχνητή νοημοσύνη απαιτούν μεγάλα σύνολα για να εκπαιδευτούν και να παρουσιάσουν αποτελέσματα.

Επομένως, τα σύνολα δεδομένων χρησιμοποιούνται για την κατασκευή μοντέλων. Συνήθως, χρησιμοποιούνται διαφορετικά σύνολα δεδομένων για την εκπαίδευση και την επικύρωση και δοκιμή ενός μοντέλου, ενώ μπορούν να υπάρξουν και σύνολα δεδομένων για την επιλογή του μοντέλου με την καλύτερη απόδοση, και το πόσο καλά το επιλεγμένο μοντέλο γενικεύεται σε άγνωστα (μη εκπαιδευμένα) σύνολα.

2.2.2.2 Περιγραφικά Στατιστικά

Τα περιγραφικά στατιστικά είναι ένα μέσο περιγραφής των χαρακτηριστικών ενός συνόλου δεδομένων και συχνά απεικονίζονται ως μια σύνοψη που εξηγεί το περιεχόμενο των δεδομένων [Bosl12]. Για παράδειγμα, μια απογραφή πληθυσμού μπορεί να περιλαμβάνει περιγραφικά στοιχεία σχετικά με την αναλογία ανδρών και γυναικών σε μια συγκεκριμένη περιοχή.

Τα περιγραφικά στατιστικά αναλύονται σε *μέτρα κεντρικής τάσης* και σε *μέτρα μεταβλητότητας*. Τα μέτρα κεντρικής τάσης περιλαμβάνουν τον μέσο όρο, τη διάμεσο και την επικρατούσα τιμή, ενώ τα μέτρα μεταβλητότητας περιλαμβάνουν την τυπική απόκλιση, τη διακύμανση, τις ελάχιστες και μέγιστες μεταβλητές, την κύρτωση και τη λοξότητα.

Ο *μέσος όρος* είναι ίσως η πιο συχνά χρησιμοποιούμενη μέθοδος για την περιγραφή της κεντρικής τάσης. Ο μέσος όρος ενός πληθυσμού συμβολίζεται με το ελληνικό γράμμα μ, ενώ ο μέσος όρος ενός δείγματος τυπικά συμβολίζεται με μια γραμμή πάνω από το σύμβολο της μεταβλητής: για παράδειγμα, ο μέσος όρος του x θα γραφόταν ως $\bar{x}$. Για να υπολογιστεί ο μέσος όρος αρκεί να αθροίσουμε όλες τις τιμές και να διαιρέσουμε με το πλήθος των τιμών. Οπότε άμα έχουμε n τιμές για ένα σύνολο δεδομένων, με τις τιμές να χαρακτηρίζονται ως $x_1, x_2, \ldots, x_n$ τότε η φόρμουλα για να βρεθεί ο μέσος όρος μ είναι:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}$$

Μία από τις σημαντικές ιδιότητες του μέσου όρου είναι πως ελαχιστοποιεί το σφάλμα κατά την πρόβλεψη μιας τιμής του συνόλου δεδομένων, δηλαδή είναι η τιμή που παράγει το χαμηλότερο ποσοστό σφάλματος από όλες τις άλλες. Παρόλα αυτά, αξίζει να τονιστεί πως έχει ένα βασικό μειονέκτημα, γιατί είναι ιδιαίτερα ευαίσθητος από την επιρροή ακραίων τιμών. Στην περίπτωση που επηρεαστεί πολύ, χάνει την ιδιότητα του να αντιπροσωπεύσει την πιο τυπική τιμή του συνόλου δεδομένων, επειδή οι ακραίες τιμές την απομακρύνουν από αυτήν.

Η *διάμεσος* αποτελεί τη μεσαία τιμή ενός συνόλου δεδομένων. Η διάμεσος επηρεάζεται λιγότερο από ακραίες τιμές και λοξά δεδομένα, και για αυτό προσφέρει καλύτερη αναπαράσταση της τυπικής τιμής. Ο τρόπος εύρεσής της εξαρτάται από τον αριθμό των τιμών που υπάρχουν. Παρακάτω βρίσκεται η φόρμουλα υπολογισμού της διαμέσου:

- Για *n* περιττό:

$$διάμεσος(x) = x_{(\frac{n+1}{2})}$$

- Για *n* άρτιο:

$$διάμεσος(x) = \frac{x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)}}{2}$$

Η *επικρατούσα τιμή* είναι η πιο συχνή τιμή στο σύνολο των δεδομένων μας. Συνήθως, η επικρατούσα τιμή χρησιμοποιείται για κατηγορικά δεδομένα, όπως όταν θέλουμε να μάθουμε ποια κατηγορία είναι η πιο κοινή. Ωστόσο, ένα πρόβλημα είναι ότι μπορεί να υπάρχουν πολλές τιμές που έχουν τον ίδιο αριθμό εμφανίσεων. Επιπλέον, δεν μας παρέχει ένα πολύ καλό μέτρο κεντρικής τάσης, όταν η πιο κοινή τιμή απέχει πολύ από τα υπόλοιπα δεδομένα, και μπορεί να προσφέρει παραπλανητικά αποτελέσματα.

Τα *μέτρα μεταβλητότητας* βοηθούν στην ανάλυση του πόσο διασκορπισμένη είναι η κατανομή σε ένα σύνολο δεδομένων. Για παράδειγμα, ενώ οι μετρήσεις της κεντρικής τάσης μπορεί να δίνουν σε ένα άτομο τον μέσο όρο ενός συνόλου δεδομένων, δεν περιγράφουν πώς κατανέμονται τα δεδομένα εντός του συνόλου.

Η *διασπορά* (ονομάζεται επίσης μεταβλητότητα ή εξάπλωση) είναι ο βαθμός κατά τον οποίο μια κατανομή τεντώνεται ή συμπιέζεται, δηλαδή η εξάπλωση των τιμών γύρω από την κεντρική τάση. Υπάρχουν δύο κοινά μέτρα διασποράς, το εύρος και η τυπική απόκλιση. Το *εύρος* υπολογίζεται αφαιρώντας την χαμηλότερη από την υψηλότερη τιμή. Η *τυπική απόκλιση* είναι μια πιο ακριβής και λεπτομερής εκτίμηση της διασποράς, επειδή μια ακραία τιμή μπορεί να υπερβάλει πολύ το εύρος. Δηλώνει τη σχέση των τιμών ενός δείγματος ως προς το μέσο όρο του δείγματος των δεδομένων. Υπολογίζεται ως εξής:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

Όπου $\bar{x}$ είναι ο μέσος όρο του δείγματος.

Η τυπική απόκλιση μας επιτρέπει να καταλήξουμε σε μερικά συμπεράσματα για το δείγμα μας. Όταν η τυπική απόκλιση είναι μεγάλη, σημαίνει πως υπάρχει μεγάλη απόκλιση των δεδομένων σε σχέση με τον μέσο όρο. Αντιθέτως μια μικρή ή χαμηλή

απόκλιση δείχνει ότι πολλά από αυτά τα δεδομένα συγκεντρώνονται στενά γύρω από τον μέσο όρο. Αυτό το είδος πληροφορίας είναι χρήσιμο για έναν αναλυτή και επιτρέπει τη περαιτέρω σύγκριση μεταξύ δεδομένων, ακόμη και όταν οι μεταβλητές μετρούνται σε εντελώς διαφορετικές κλίμακες [Bosl12].

2.2.2.3 Συσχέτιση Δεδομένων

Η συσχέτιση είναι ένα από τα πιο κοινά και χρήσιμα στατιστικά στοιχεία και εκφράζεται από έναν αριθμό που περιγράφει τον βαθμό σχέσης μεταξύ δύο μεταβλητών [Bosl12]. Η λέξη συσχέτιση υποδηλώνει κάποιο είδος σχέσης μεταξύ των δύο μεταβλητών, ωστόσο πολλές σχέσεις μπορεί να είναι αποτέλεσμα τύχης ή άλλων μεταβλητών και να μην υπάρχει εξάρτηση μεταξύ τους. Δεν αρκεί, λοιπόν, για να συναχθεί η ύπαρξη αιτιώδους σχέσης, δηλαδή η συσχέτιση δε συνεπάγεται αιτιότητα.

Υπάρχουν διάφορα μέτρα συσχέτισης με έναν από τους πιο γνωστούς να είναι ο συντελεστής συσχέτισης Pearson. Ο αριθμός *r* του Pearson έχει εύρος από το μείον ένα έως το ένα, με το μηδέν να μην δείχνει καμία σχέση μεταξύ των μεταβλητών, ενώ οι μεγαλύτερες απόλυτες τιμές να υποδεικνύουν ισχυρότερη σχέση μεταξύ τους. Η φόρμουλα εύρεσής του είναι:

$$r = \frac{SS_{xy}}{\sqrt{SS_x SS_y}}$$

Όπου:

SS_x είναι το άθροισμα των τετραγωνισμένων τιμών *x*,

SS_y είναι το άθροισμα των τετραγωνισμένων τιμών *y* και

SS_{xy} είναι το άθροισμα των τετραγωνισμένων τιμών *x* και *y*.

Το SS_x (και αντίστοιχα το SS_y) υπολογίζεται ως:

$$SS_x = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Όπου $\bar{x}$ είναι ο μέσος όρος του δείγματος για το *x*.

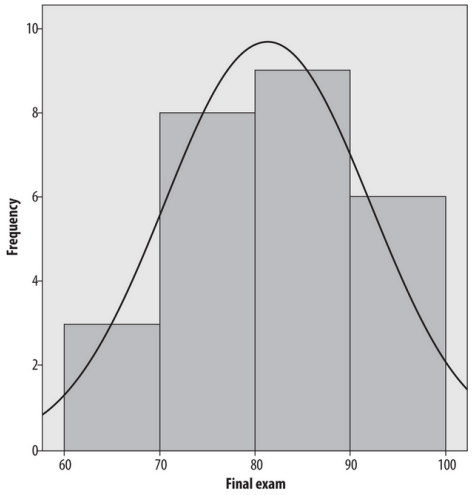
Ενώ το SS_{xy} υπολογίζεται ως:

$$SS_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

Σε ένα παράδειγμα σύγκρισης δύο μεταβλητών με *r = 0.98*, μπορούμε να πούμε πως συσχετίζονται μεταξύ τους. Ύστερα της εύρεσης του *r*, συνήθως ελέγχουμε την στατιστική σημασία για να προσδιορίσουμε εάν η συσχέτιση είναι σημαντική, το οποίο όμως δε θα αναλύσουμε περαιτέρω σε αυτή την εργασία [Bosl12].

2.2.2.4 Ιστόγραμμα

Υπάρχουν αμέτρητοι τρόποι να παρουσιάσουμε δεδομένα και καθένας από αυτούς προσφέρει έναν διαφορετικό τρόπο αναπαράστασης και μελέτης των δεδομένων αυτών. Ένας από τους πιο διάσημους είναι το *ιστόγραμμα*, γιατί προσφέρει εύκολη ομαδοποίηση συνεχείς δεδομένων σε προκαθορισμένο αριθμό κάδων [Bosl12]. Ένας κάδος δεν είναι παρά μια ορισμένη περιοχή τιμών, έστω παραδείγματος χάριν από το μηδέν έως το χίλια, που περιέχει όλες τις τιμές του συνόλου που ανήκουν σε αυτή τη περιοχή. Κάθε κάδος συνήθως έχει το ίδιο μέγεθος περιοχής, ενώ το πλήθος των κάδων που ορίζουμε μπορεί να διαφέρει. Χρησιμοποιείται συνήθως για να απεικονίσουμε την συχνότητα και την κατανομή των δεδομένων.



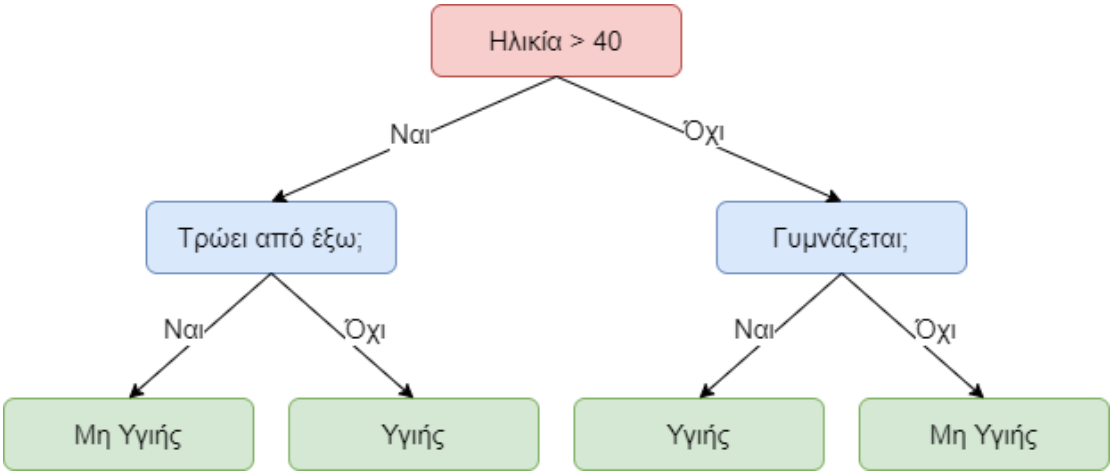
Εικόνα 1: Ιστόγραμμα βαθμολογιών τελικής εξέτασης [Bosl12].

Στο ιστόγραμμα της *Εικόνα 1* βλέπουμε το πλήθος των παιδιών (Frequency) που έγραψαν από 60 έως 100 βαθμούς σε ένα τελικό διαγώνισμα (Final exam), χωρισμένα σε 4 κάδους πλάτους 10. Παρατηρούμε εύκολα πως τα περισσότερα παιδιά έχουν γράψει από 70 έως 90 μονάδες, ενώ λιγότερα βρίσκονται στα άκρα έχοντας γράψει 60 με 70 ή 90 με 100 μονάδες. Στο ιστόγραμμα επίσης υπερτίθεται η κανονική κατανομή ως οπτική αναφορά, ώστε να μπορούμε να κρίνουμε πόσο παρόμοιες είναι οι τιμές σε σχέση με τη κατανομή.

2.2.2.5 Δέντρο Αποφάσεων

Ένα δέντρο αποφάσεων είναι ένα μοντέλο πρόβλεψης που έχει εφαρμογές που καλύπτουν έναν αριθμό διαφορετικών περιοχών. Γενικά, τα δέντρα αποφάσεων κατασκευάζονται μέσω μιας αλγοριθμικής προσέγγισης που προσδιορίζει τρόπους διαχωρισμού ενός συνόλου δεδομένων με βάση διαφορετικές συνθήκες. Τα δέντρα

αποφάσεων είναι μία από τις πιο ευρέως χρησιμοποιούμενες και πρακτικές μεθόδους για εποπτευόμενη μάθηση. Ένα δέντρο αποφάσεων αποτελεί μία μη παραμετρική εποπτευόμενη μέθοδο μάθησης που χρησιμοποιείται τόσο για ταξινόμηση (για διακριτές τιμές) όσο και για παλινδρόμηση (για συνεχείς τιμές). Ο στόχος είναι να δημιουργηθεί ένα μοντέλο που προβλέπει τη τιμή μιας μεταβλητής στόχου, μαθαίνοντας απλούς κανόνες απόφασης που συνάγονται τα χαρακτηριστικά των δεδομένων.



Εικόνα 2: Δέντρο απόφασης που εκτιμάει άμα κάποιος είναι υγιής ή όχι.

Ένα δέντρο αποφάσεων είναι ένα δενδροειδές γράφημα με κόμβους που αντιπροσωπεύουν το μέρος όπου επιλέγουμε ένα χαρακτηριστικό και κάνουμε μια ερώτηση. Οι διασυνδέσεις αντιπροσωπεύουν τις απαντήσεις στην ερώτηση και τα φύλλα τη έξοδο. Το δέντρο αναπτύσσεται κατά σύμβαση από πάνω προς τα κάτω και ξεκινάει από τον κόμβο «ρίζα», ο οποίος βρίσκεται πάντα στη κορυφή και υποδηλώνει πως περιέχει ολόκληρο το σύνολο των δεδομένων [LaLa14]. Στο παράδειγμα της *Εικόνας 2* είναι το (κόκκινο) τετράγωνο στη κορυφή που χαρακτηρίζει την ηλικία. Οι κόμβοι διασυνδέονται μεταξύ τους μέσω διακλαδώσεων που αντιπροσωπεύουν την απάντηση στη ερώτηση που έχει τεθεί στον κόμβο προέλευσής τους και συνήθως είναι απαντήσεις του τύπου ναι ή όχι, ή μια τιμή του χαρακτηριστικού έναντι ενός ορίου (Π.χ. >=40). Αλλιώς, οι ερωτήσεις στους κόμβους, απαντώνται κατηγορικά, δηλαδή με μία από τις διακριτές επιλογές εκείνου του χαρακτηριστικού. Στην *Εικόνα 2* οι διακλαδώσεις αποτελούν τα βελάκια που ενώνουν τους κόμβους. Με τον διαχωρισμό των απαντήσεων, και επομένως των δεδομένων σε μικρότερα υποσύνολα, συνήθως οι διακλαδώσεις μας οδηγούν σε ενδιάμεσους κόμβους και άρα σε περαιτέρω ερωτήσεις και απαντήσεις για άλλα χαρακτηριστικά. Στην *Εικόνα 2* οι ενδιάμεσοι κόμβοι είναι τα (μπλε) τετράγωνα στη μέση που αναλύουν άμα τρώει από έξω ή άμα γυμνάζεται το δείγμα. Κάθε ενδιάμεσο χαρακτηριστικό μπορεί να εμφανίζεται πολλαπλές φορές και με διαφορετικές τιμές. Η

ρίζα και κάθε ενδιάμεσος κόμβος μπορούν να καταλήξουν σε μια τελική απόφαση που ονομάζεται τελικός ή τερματικός κόμβος, το «φύλλο». Τα φύλλα περιέχουν τις τελικές αποφάσεις, δηλαδή τις τιμές της μεταβλητής στόχου που θέλουμε να προβλέψουμε χρησιμοποιώντας τις υπόλοιπες μεταβλητές. Στην *Εικόνα 2* οι τελικοί κόμβοι («φύλλα») είναι τα (πράσινα) τετράγωνα στο κάτω μέρος που δηλώνουν άμα είναι υγιής ή όχι. Τα φύλλα είναι τερματικά και δεν διακλαδώνονται περαιτέρω. Το δέντρο αποφάσεων, λοιπόν, αποτελεί μια σειρά ερωτήσεων που πρέπει να απαντηθούν για να πάρουμε το τελικό αποτέλεσμα, διαβαίνοντας από πάνω προς τα κάτω, από τον κόμβο ρίζα μέχρι έναν τερματικό κόμβο [Agga15]. Βάση του παραδείγματος της *Εικόνας 2*, αν έχουμε ένα άτομο 65 ετών που τρώει από έξω τότε θεωρείται μη υγιής, ενώ ένα άτομο 35 ετών που γυμνάζεται θεωρείται υγιής.

Ο στόχος του κριτηρίου διαχωρισμού του συνόλου δεδομένων (της ερώτησης) σε κάθε κόμβο είναι να μεγιστοποιήσει τον διαχωρισμό των διαφορετικών χαρακτηριστικών μεταξύ των κόμβων παιδιών του. Στη συνέχεια θα συζητήσουμε μόνο για χαρακτηριστικά μίας μεταβλητής. Ο σχεδιασμός του κριτηρίου διαχωρισμού εξαρτάται από τη φύση του υποκείμενου χαρακτηριστικού [Agga15].

Τα κριτήρια επιλογής χαρακτηριστικών αναλόγως τη φύση τους είναι:

- *Δυαδικό χαρακτηριστικό*: Είναι δυνατός μόνο ένας τύπος διαχωρισμού σε δύο διακλαδώσεις, όπου ο κάθε διακλάδωση αντιστοιχεί σε μία από τις δύο τιμές.
- *Κατηγορικό χαρακτηριστικό*: Άμα ένα κατηγορικό χαρακτηριστικό έχει x τιμές, τότε θα υπάρχουν πολλοί τρόποι να διαχωριστεί. Μια επιλογή είναι να χωριστεί σε x διακλαδώσεις, όπου κάθε διακλάδωση αντιστοιχεί σε μία από τις τιμές. Ένας άλλος τρόπος διαχωρισμού είναι να χρησιμοποιηθεί δυαδικός διαχωρισμός, υπολογίζοντας τους $2^x - 1$ συνδυασμούς των τιμών του κατηγορικού χαρακτηριστικού και επιλέγοντας τον καλύτερο. Αυτό αποτελεί, όμως, δαπανηρή διαδικασία και δεν είναι πάντα εφικτό. Μία απλή προσέγγιση που χρησιμοποιείται συνήθως είναι η μετατροπή των κατηγορικών σε δυαδικά δεδομένα, όπου η κάθε τιμή μετατρέπεται σε 0 ή 1, και όλες οι τιμές έχουν την τιμή 0, εκτός της επιλεγμένης που έχει τη τιμή 1. Σε αυτή τη περίπτωση, ακολουθούμε τον διαχωρισμό ενός δυαδικού χαρακτηριστικού.
- *Αριθμητικό χαρακτηριστικό*: Άμα ένα αριθμητικό χαρακτηριστικό περιέχει έναν μικρό αριθμό x τιμών σε σειρά (Π.χ. ακέραιοι εύρους $[1, x]$), είναι δυνατό να δημιουργηθούν x διακλαδώσεις, μία για κάθε τιμή, όπως και στα κατηγορικά δεδομένων. Ωστόσο, συνήθως, για συνεχείς αριθμητικά χαρακτηριστικά, ο

διαχωρισμός γίνεται χρησιμοποιώντας αριθμητικούς τελεστές σύγκρισης, όπως $r \leq \alpha$, όπου r είναι η τιμή του χαρακτηριστικού και α μια σταθερή τιμή.

Για να προσδιορίσουμε το κατάλληλο κριτήριο διαχωρισμού, δηλαδή το χαρακτηριστικό και τον τρόπο διαχωρισμού του, απαιτούνται μέθοδοι ή αλγόριθμοι που υπολογίζουν τη ποιότητα του κάθε διαχωρισμού. Στόχος των μεθόδων και αλγορίθμων αυτών είναι η μεγιστοποίηση του διαχωρισμού των διαφορετικών χαρακτηριστικών μεταξύ των κόμβων παιδιών του, όπως και στα κριτήρια επιλογής χαρακτηριστικών [Agga15].

Μερικοί από αυτούς είναι:

- *Δείκτης Gini (Gini index)*: Ο δείκτης Gini $G(S)$ υποδηλώνει τον βαθμό πρόσμειξης των δεδομένων. Ο δείκτης του έχει εύρος $[0, 1]$. Το μηδέν αναπαριστά την έλλειψη προσμείξεων χαρακτηριστικών σε αυτό το φύλλο-κόμβο, δηλαδή υπάρχει μόνο ένα συγκεκριμένο χαρακτηριστικό από όλα. Μεγαλύτερες τιμές δηλώνουν πως υπάρχει ανάμειξη χαρακτηριστικών. Μεταξύ δύο χαρακτηριστικών, πάντα επιλέγουμε αυτό με τη μικρότερη τιμή πρόσμειξης [Agga15].

  Ο δείκτης Gini $G(S)$ υπολογίζεται ως εξής:

$$G(S) = 1 - \sum_{j=1}^{k} p_j^2$$

  Όπου σε ένα σύνολο δεδομένων S:

  - k είναι ο αριθμός των μοναδικών τιμών ενός χαρακτηριστικού και
  - p_j η πιθανότητα των δειγμάτων να ανήκουν στη τιμή j ενός χαρακτηριστικού σε έναν κόμβο.

  Ο συνολικός δείκτης Gini ενός συνόλου S, χωρισμένο σε x υποσύνολα $S_1 \dots S_x$ (όπου x είναι το πλήθος των τιμών του χαρακτηριστικού που έγινε ο διαχωρισμός), υπολογίζεται ως ο σταθμισμένος μέσος όρος των δεικτών Gini $G(S_i)$ του κάθε S_i, όπου το βάρος του S_i είναι το $|S_i|$:

$$Gini\text{-}Split(S \Rightarrow S_1 \dots S_x) = \sum_{i=1}^{x} \frac{|S_i|}{|S|} G(S_i)$$

- *Εντροπία (Entropy)*: Η εντροπία χρησιμοποιείται σε έναν από τους πρώτους αλγόριθμους κατηγοριοποίησης-ταξινόμησης, τον *ID3*, καθώς και τον *C4.5* [Agga15].

  Η εντροπία $E(S)$ υπολογίζεται ως εξής:

$$G(S) = -\sum_{j=1}^{k} p_j \log_2(p_j)$$

Όπου σε ένα σύνολο δεδομένων εκπαίδευσης S:

- k είναι ο αριθμός των μοναδικών τιμών ενός χαρακτηριστικού και
- p_j η πιθανότητα των δειγμάτων να ανήκουν στη τιμή j ενός χαρακτηριστικού σε έναν κόμβο.

Όπως και στον συνολικό δείκτη Gini, η συνολική εντροπία ενός συνόλου S, χωρισμένο σε x υποσύνολα $S_1 \dots S_x$ (όπου x είναι το πλήθος των τιμών του χαρακτηριστικού που έγινε ο διαχωρισμός), υπολογίζεται ως ο σταθμισμένος μέσος όρος των δεικτών Gini $G(S_i)$ του κάθε S_i, όπου το βάρος του S_i είναι το $|S_i|$:

$$Entropy\text{-}Split(S \Rightarrow S_1 \dots S_x) = \sum_{i=1}^{x} \frac{|S_i|}{|S|} G(S_i)$$

Προτιμάμε πάντα την εντροπία με τη χαμηλότερη τιμή.

Η βιβλιοθήκη υπολογισμού δέντρου αποφάσεων που χρησιμοποιούμε στην εφαρμογή μας υποστηρίζει μόνο αυτές τις δύο μετρικές ποιότητας διαχωρισμού χαρακτηριστικών. Ωστόσο, αξίζει να δηλωθεί πως υπάρχουν κι άλλες, όπως το Error rate και το Information Gain, που έχει κοντινή συσχέτιση με την εντροπία [Agga15].

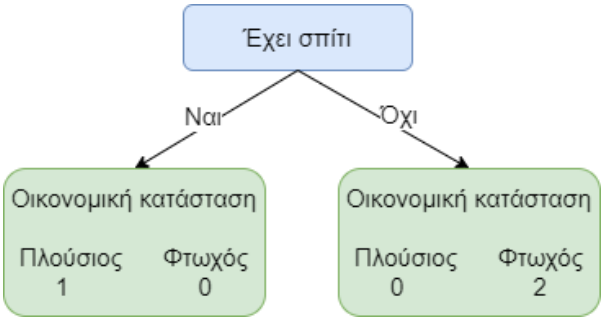
Συνεχίζοντας, θα καταφύγουμε σε ένα παράδειγμα υπολογισμού ενός μικρού δέντρου απόφασης, για να καταλάβουμε καλύτερα τον υπολογισμό του δέντρου μέσω του δείκτη Gini.

Έστω το σύνολο δεδομένων του *Πίνακα 2*, με χαρακτηριστικά τις 3 πρώτες στήλες και αναζητώντας την εύρεση της «Οικονομική κατάσταση» της κάθε εγγραφής (σειράς) μέσω αυτών.

Έχει σπίτι	Ηλικία	Εισόδημα	Οικονομική κατάσταση
Ναι	30	23000	Πλούσιος
Όχι	50	17000	Φτωχός
Όχι	19	10000	Φτωχός

Πίνακας 2: Σύνολο δεδομένων S με τυχαία στοιχεία.

Για χαρακτηριστικά με διακριτές μεταβλητές όπως η στήλη «Έχει σπίτι», θα αναλύσουμε κάθε μοναδική ετικέτα της (σε αυτή τη περίπτωση «Ναι» ή «Όχι») ως προς το πλήθος των ετικετών της μεταβλητής στόχου που αναλογεί στη καθεμία [Star22].



Εικόνα 3: Αναλογία των τιμών του χαρακτηριστικού «Έχει σπίτι» ως προς τις τιμές της στήλης «Οικονομική κατάσταση».

Όπως παρατηρούμε στην *Εικόνα 3*, η απάντηση «Ναι» στο «Έχει σπίτι» αναλογεί σε ένα «Πλούσιος» και μηδέν «Φτωχός», ενώ στην απάντηση «Όχι» αναλογεί σε μηδέν «Πλούσιος» και δύο «Φτωχός». Λαμβάνοντας υπόψιν μας αυτά τα δεδομένα, για το «Έχει σπίτι» θα πάμε να υπολογίσουμε τον δείκτη Gini του «Ναι»:

$$G(S_1) = 1 - \sum_{j=1}^{k} p_j^2$$

$$= 1 - (\text{πιθανότητα του «Πλούσιος»})^2 - (\text{πιθανότητα του «Φτωχός»})^2$$

$$= 1 - (\tfrac{1}{1+0})^2 - (\tfrac{0}{1+0})^2 = 0$$

και τον δείκτη Gini του «Όχι»:

$$G(S_2) = 1 - \sum_{j=1}^{k} p_j^2$$

$$= 1 - (\text{πιθανότητα του «Πλούσιος»})^2 - (\text{πιθανότητα του «Φτωχός»})^2$$

$$= 1 - (\tfrac{0}{0+2})^2 - (\tfrac{2}{0+2})^2 = 0$$

Και τα δύο έχουνε την τιμή μηδέν, γεγονός που είναι λογικό, γιατί σε κανένα φύλλο δεν υπάρχει πρόσμειξη των τιμών του χαρακτηριστικού «Οικονομική Κατάσταση».

Έπειτα, υπολογίζουμε τον συνολικό δείκτη Gini ως εξής:

$$Gini\text{-}Split(S \Rightarrow S_1 \dots S_x) = \sum_{i=1}^{x} \frac{|S_i|}{|S|} G(S_i)$$

$$= \left(\frac{1}{1+2}\right) \cdot 0 + \left(\frac{2}{1+2}\right) \cdot 0$$

$$= 0$$

Επομένως, ο συνολικός δείκτης Gini για το χαρακτηριστικό «Έχει σπίτι» ισούται με μηδέν.

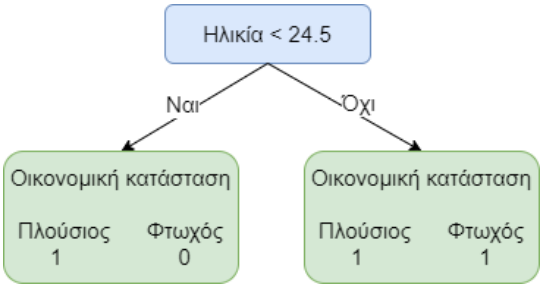
Για χαρακτηριστικά με συνεχείς μεταβλητές όπως η στήλη «Ηλικία», πρώτα θα ταξινομήσουμε τις σειρές του πίνακα με αύξουσα σειρά ως προς την αυτήν (*Πίνακας 3*) [Star22].

Έχει σπίτι	Ηλικία	Εισόδημα	Οικονομική κατάσταση
Όχι	19	10000	Φτωχός
Ναι	30	23000	Πλούσιος
Όχι	50	17000	Φτωχός

Πίνακας 3: Ταξινομημένος Πίνακας 2 με αύξουσα σειρά ως προς την «Ηλικία».

Έπειτα θα υπολογίσουμε τις μέσες τιμές όλων των γειτονικών τιμών της στήλης «Ηλικία», δηλαδή στη περίπτωσή μας παίρνουμε την τιμή $x = 24.5$ για τις ηλικίες *19* και *30* και $y = 40$ για τις ηλικίες *30* και *50*.

Ύστερα, υπολογίζουμε τον δείκτη Gini για κάθε τιμή όπως και πριν. Αυτή τη φορά η ερώτηση που θα κάνουμε στον κόμβο θα είναι άμα η ηλικία είναι μικρότερη των τιμών που βρήκαμε. Για παράδειγμα για την τιμή $x = 24.5$ ισχύει:



Εικόνα 4: Αναλογία των τιμών του χαρακτηριστικού «Ηλικία» μικρότερη του 24.5 ως προς τις τιμές της στήλης «Οικονομική κατάσταση».

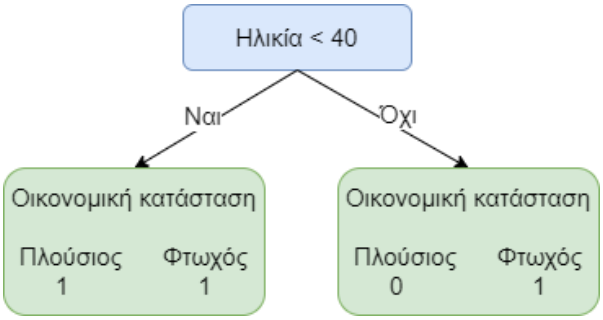
Με τη βοήθεια της *Εικόνας 4*, βρίσκουμε τον δείκτη Gini του πρώτου φύλλου που είναι *0* και του δεύτερου που είναι *0.5,* ακολουθώντας τα ίδια βήματα με τον υπολογισμό του χαρακτηριστικού «Έχει σπίτι». Έπειτα βρίσκουμε τον συνολικό δείκτη Gini ως εξής:

$$Gini\text{-}Split(S \Rightarrow S_1 \dots S_x) = \sum_{i=1}^{x} \frac{|S_i|}{|S|} G(S_i)$$

$$= \left(\frac{1}{1+2}\right) \cdot 0 + \left(\frac{2}{1+2}\right) \cdot 0.5$$

$$= 0.33\tilde{3}$$

Με τον ίδιο τρόπο υπολογίζουμε τον συνολικό δείκτη Gini για το $y = 40$ ο οποίος επίσης ανέρχεται στη τιμή $0.33\tilde{3}$.



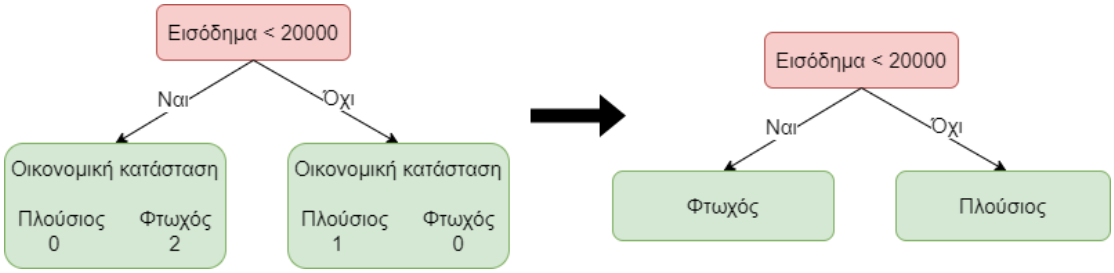
Εικόνα 5: Αναλογία των τιμών του χαρακτηριστικού «Ηλικία» μικρότερη του 40 ως προς τις τιμές της στήλης «Οικονομική κατάσταση».

Αφού βρούμε όλους τους δείκτες Gini για κάθε μέση τιμή, τους συγκρίνουμε και επιλέγουμε τον μικρότερο. Όταν υπάρχει ισοπαλία μεταξύ των δεικτών κάθε τιμής, δεν υπάρχει πρόβλημα και επιλέγουμε οποιονδήποτε επιθυμούμε από αυτούς, έστω στο παράδειγμά μας την $y = 40$.

Άρα, ο συνολικός δείκτης Gini για το χαρακτηριστικό «Ηλικία» ισούται με $0.33\tilde{3}$ για $Ηλικία < 40$.

Ακολουθούμε την ίδια διαδικασία για τη στήλη «Εισόδημα» και βρίσκουμε τον συνολικό δείκτη Gini να ισούται με 0 για $Εισόδημα < 20000$.

Αφού υπολογίσουμε τους δείκτες Gini για κάθε χαρακτηριστικό, τότε επιλέγουμε αυτόν με τη μικρότερη τιμή και τον ορίζουμε ως ρίζα του δέντρου μας. Επειδή έχουμε δύο επιλογές στη τιμή του μηδενός, θα επιλέξουμε τυχαία το $Εισόδημα < 20000$. Κατά αυτόν τον τρόπο χωρίζουμε τα δεδομένα σε δύο υποσύνολα, αυτούς που έχουνε εισόδημα άνω των 20000 και αυτούς που δεν το έχουνε. Ύστερα, για τον υπολογισμό ολόκληρου του δέντρου, εφαρμόζουμε την ίδια μεθοδολογία σε κάθε υποσύνολο μέχρι να καταλήξουμε να μην χρειάζεται περαιτέρω διαχωρισμός, όπου και μετατρέπουμε τους κόμβους σε τερματικούς (φύλλα). Συνήθως, το αποτέλεσμα που αναγράφεται στο φύλλο είναι η κατηγορία που έχει τις περισσότερες ψήφους (*Εικόνα 6*).



Εικόνα 6: Μετατροπή σε τελικό δέντρο

Στη περίπτωσή μας (*Εικόνα 6*), κάθε κόμβος εκτός της ρίζας έχει δείκτη Gini μηδέν, οπότε είμαστε ευχαριστημένοι με το πως έχουν χωριστεί τα δεδομένα μας και δεν εξετάζουμε περαιτέρω χαρακτηριστικά.

Αξίζει να τονίσουμε πως ένα δέντρο αποφάσεων κατά τη διαδικασία της δημιουργίας του μπορεί να αποβεί υπερβολικά προσαρμοσμένο (overfit) στα δεδομένα που του δόθηκαν και να αποτύχει να διακρίνει σημαντικά μοτίβα. Αυτό σημαίνει πως αδυνατεί να ανταπεξέλθει σε διαφορετικά σύνολα δεδομένων, λόγω της εξειδίκευσής του στο δοθέν σύνολο. Υπάρχουν δύο κύριοι τρόποι αντιμετώπισης αυτού του προβλήματος. Ο πρώτος τρόπος αντιμετώπισης του overfitting απαιτεί τη θέση κανόνων (stopping criteria), όπως ένας ελάχιστος αριθμός ατόμων σε έναν κόμβο για να μπορεί να γίνει διαχωρισμός, ενώ ο δεύτερος τρόπος εξετάζει το κλάδεμα ορισμένων ή όλων των κόμβων και τη σύγκριση μεταξύ των συνδυασμών των κλαδεμένων δέντρων, για να βρεθεί το κατάλληλο βάθος ή συνδυασμός [RoOd14]. Γενικά, απλούστερα δέντρα αποφάσεων προτιμώνται σε σχέση με πιο περίπλοκα, άμα παρουσιάζουν τον ίδιο βαθμό σφάλματος στα δεδομένα εκπαίδευσης [Agga15].

2.3 Ανάλυση απαιτήσεων

Σε αυτήν την υποενότητα θα εξετάσουμε τα User Stories που παρατίθενται από τον χρήστη στη καινούργια έκδοση του λογισμικού. Η φύση της εργασίας είναι η αυτοματοποίηση των ενεργειών που απαιτούνται από τον χρήστη και αυτό συμβάλει στη ελαχιστοποίηση της αλληλεπίδρασής του με το σύστημα, άρα και σε λιγότερα User Stories από ότι θα ανέμενε κανείς από ένα λογισμικό ανάλογου μεγέθους. Η καινούργια έκδοση, επίσης, επικεντρώνεται στη αναδιοργάνωση, ανασυγκρότηση, συντήρηση και επέκταση της δομής και κάποιων λειτουργιών του συστήματος, οπότε δεν επιφέρει σημαντικές αλλαγές στις ενέργειες του χρήστη.

Παρακάτω παρατίθενται τα User stories:

- [US1] Ως χρήστης, θέλω να αναλύω τα δεδομένα όλων των πεδίων σε μορφή ιστογράμματος.
- [US2] Ως χρήστης, θέλω να μπορώ να εξάγω ένα δέντρο αποφάσεων για κάθε labeled στήλη, όπου θα εμπλέκονται αυτόματα όλα τα πεδία της επιλογής μου ως χαρακτηριστικά, εκτός και άμα έχω επιλέξει μόνο συγκεκριμένα πεδία.
- [US3] Ως χρήστης, θέλω να λαμβάνω μια αναφορά με τα δεδομένα που έχουν υπολογισθεί, ώστε να τα αναλύσω.

Κεφάλαιο 3. Σχεδίαση & Υλοποίηση

3.1 Ορισμός προβλήματος και αλγόριθμοι επίλυσης

3.1.1 Ορισμός προβλήματος

Η αξιοποίηση μεγάλων συνόλων δεδομένων, και καθεξής, η ανάλυση τους και η εξαγωγή χρήσιμων πληροφοριών και στατιστικών συμπερασμάτων μέσα από αυτά, έχουν οδηγήσει τον κόσμο στην δημιουργία πολλών εργαλείων που θα εξυπηρετήσουν τον σκοπό αυτό. Τα εργαλεία αυτά ξεκίνησαν από προγραμματιστικές γλώσσες και εντολές που μπορούσαν να χρησιμοποιήσουν μόνο όσοι είχαν μια ευπρεπές βάση στον προγραμματισμό, ενώ τα τελευταία χρόνια έχουν εξελιχθεί σε εργαλεία, ακόμα και διαδικτυακά, που προσφέρουν μια αφαίρεση των δυσκολότερων προγραμματιστικών εντολών και την ένταξη φιλικών προς τον χρήστη διεπαφών. Ωστόσο, ακόμα και για τις πιο απλές εργασίες, χρειάζεται σημαντική συμβολή του χρήστη για να παραχθούν αποτελέσματα, είτε προγραμματιστικά όπως στη Python και το Jupyter Notebook, είτε με την εισαγωγή των αντίστοιχων εντολών σε εργαλεία όπως το Orange και το Tableau.

Η εφαρμογή που αναπτύσσουμε (Pythia) προσπαθεί να καταπολεμήσει αυτό το πρόβλημα, προσφέροντας τη δυνατότητα αυτόματης παραγωγής ενός στατιστικού προφίλ, επικεντρώνοντας στα βασικότερα στατιστικά μοντέλα και αναδεικνύοντας σημεία στα δεδομένα που αξίζει να δοθεί έμφαση. Όλα αυτά, με σχεδόν μηδαμινή συμβολή του χρήστη και παράγοντας συγκεντρωτικά όλα τα στατιστικά της σε μία οπτικά ευανάγνωστη τελική αναφορά.

Ωστόσο, η τρέχουσα εφαρμογή παρουσιάζει μερικές ελλείψεις και προβλήματα όπως:

- Υπάρχουν πολλές εξαρτήσεις σε εξωτερικές βιβλιοθήκες, όπως η Lombok, και χρειάζεται απλοποίηση του ολικού κώδικα.
- Τα τωρινά παραγόμενα δέντρα αποφάσεων συμπεριλαμβάνουν μόνο αριθμητικές κολόνες στην παραγωγή τους.
- Τα τωρινά παραγόμενα δέντρα αποφάσεων συμπεριλαμβάνουν όλες τις δυνατές κολόνες στην παραγωγή τους, ακόμα και αυτές που δημιούργησαν την κολόνα-στόχο.
- Τα δέντρα αποφάσεων είναι ουσιαστικά ένα απλό κείμενο και δεν περιέχουν περαιτέρω πληροφορίες, όπως την ακρίβεια του δέντρου.

- Στη τρέχουσα εφαρμογή, η τελική αναφορά είναι δυσανάγνωστη και όλη σε κείμενο.
- Επίσης, το τρέχον πρόγραμμα θα μπορούσε να επωφεληθεί από καινούργιες εντάξεις στατιστικών στοιχείων.
- Κατά το τρέξιμο του προγράμματος, η κονσόλα είναι γεμάτη καταγραφές (log) του Spark και καθιστάτε δύσκολη η αποσφαλμάτωση του κώδικα.
- Η κεντρική μηχανή που αλληλοεπιδρά με τον τελικό χρήστη, κάνει πολλές δουλειές και εξυπηρετεί λειτουργίες που δε θα έπρεπε να λαμβάνουν μέρος στο πακέτο αυτό.

3.1.2 Περιγραφή των χρησιμοποιούμενων δομών δεδομένων.

Για να επιτευχθεί η διαδικασία εγγραφής ενός συνόλου δεδομένων, η αυτόματη συγκεντρωτική παραγωγή των στατιστικών του, καθώς και όλες οι λειτουργίες του λογισμικού Pythia, χρησιμοποιήθηκαν οι παρακάτω δομές δεδομένων:

3.1.2.1 Dataset<Row>

Η βιβλιοθήκη Apache Spark παρέχει τη δομή δεδομένων Dataset που περιέχει στοιχεία Row, που αντιπροσωπεύουν το σύνολο δεδομένων και τις εγγραφές αντίστοιχα. Χρησιμοποιείται για να εγγράψουμε το εξωτερικό σύνολο δεδομένων που δίνει ο χρήστης στη μηχανή και να το αναπαραστήσουμε ως μια δομή που μπορεί να χρησιμοποιηθεί περαιτέρω, μέσω πράξεων, σε στατιστικά μοντέλα ή για την συλλογή συγκεκριμένων πληροφοριών.

3.1.2.2 Labeled Columns (Labeling System)

Ένα Labeled Column είναι μια νέα κολόνα στο σύνολο δεδομένων που παράγεται από τα δεδομένων άλλων κολόνων, βάση των κανόνων που τους έχουν δοθεί από τον χρήστη. Το Pythia παρέχει τη δυνατότητα στον χρήστη να ορίσει τους δικούς του κανόνες και να δημιουργήσει μια καινούργια κολόνα βάση αυτών, με την βοήθεια του Apache Spark. Αργότερα οι κολόνες αυτές χρησιμοποιούνται στη κατασκευή δέντρων αποφάσεων, ενώ έχουν κι άλλες εφαρμογές όπως για ελέγχους υπόθεσης.

3.1.2.3 Περιγραφικά Στατιστικά (Descriptive Statistics)

Τα περιγραφικά στατιστικά, όπως περιγράφηκαν και στο Κεφάλαιο 2.2.2.2, είναι από τα κύρια και πιο απλά στατιστικά που μπορεί κάποιος να υπολογίσει για ένα σύνολο δεδομένων. Στο Pythia είναι τα πρώτα που υπολογίζονται, βρίσκοντας αυτόματα τη μέση

τιμή, τη διάμεσο, την τυπική απόκλιση, τη μέγιστη και ελάχιστη τιμή και το πλήθος των τιμών για κάθε κολόνα. Ο υπολογισμός τους γίνεται μέσω του Apache Spark και τα αποτελέσματα διαφέρουν, ανάλογα με την φύση των δεδομένων της κολόνας, δηλαδή αν είναι αριθμητικά ή κάτι άλλο.

3.1.2.4 Συσχετίσεις Δεδομένων (All Pairs Correlations)

Εξίσου σημαντικές είναι και οι συσχετίσεις δεδομένων μεταξύ των διάφορων κολόνων του συνόλου δεδομένων. Περιγράφηκαν περαιτέρω στο Κεφάλαιο 2.2.2.3. Το Pythia υπολογίζει όλους τους πιθανούς συνδυασμούς των στηλών με τη μέθοδο συσχέτισης Pearson.

3.1.2.5 Ιστογράμματα (Histogram Plots)

Ένα ιστόγραμμα είναι απλό και επιτρέπει στον χρήστη να ομαδοποιήσει εύκολα τα δεδομένα μιας κολόνας ενός συνόλου δεδομένων, σε προκαθορισμένους κάδους (Δείτε Κεφάλαιο 2.2.2.4). Το Pythia παράγει ένα ιστόγραμμα για κάθε αριθμητική στήλη, με τον αριθμό κάδων που έχει προσδιορίσει ο χρήστης. Κάθε κάδος περιέχει το εύρος (άνω και κάτω όριο) των τιμών που απευθύνεται και έναν αριθμό που δηλώνει το πλήθος των τιμών της στήλης που βρίσκονται σε αυτό το εύρος.

3.1.2.6 Δέντρα Αποφάσεων (Decision Trees)

Το δέντρο αποφάσεων είναι ένα μοντέλο που προβλέπει τη τιμή μιας μεταβλητής στόχου, μαθαίνοντας απλούς κανόνες απόφασης που συνάγονται τα χαρακτηριστικά των δεδομένων (Δείτε Κεφάλαιο 2.2.2.5). Το Pythia παράγει ένα δέντρο αποφάσεων για κάθε Labeled κολόνα, με τις προκαθορισμένες ή τις δοθέντες από τον χρήστη ρυθμίσεις. Κάθε δέντρο αποφάσεων περιέχει πληροφορίες όπως τις στήλες που έδρασαν ως χαρακτηριστικά και μη, την ακρίβειά του και κάθε κόμβο του, με τις αντίστοιχες πληροφορίες για τον καθένα (Π.χ. επίπεδο πρόσμειξης, κριτήριο διαχωρισμού, αν είναι φύλλο ή όχι). Επίσης, ένα δέντρο απόφασης αναπαρίσταται οπτικά σε μορφή εικόνας PNG για την ένταξή του στη τελική αναφορά.

3.1.2.7 Εξαγωγή ευρημάτων σε αρχεία στον δίσκο (Report System)

Αφού ολοκληρωθεί η διαδικασία του υπολογισμού του προφίλ του δοθέντος συνόλου δεδομένων, το Pythia εξάγει τα ευρήματά του σε μορφή απλού κειμένου, JSON ή markdown αρχείου μέσω των αντίστοιχων Report κλάσεων. Ο προορισμός δημιουργίας και ο τύπος του αρχείου αυτού, καθορίζονται από τον χρήστη. Αναλόγως τον τύπο της

τελικής αναφοράς, χρησιμοποιούνται διαφορετικοί τρόποι αναπαράστασης των δομών δεδομένων του στατιστικού προφίλ. Κύριο παράδειγμα αποτελεί η αναπαράσταση των δέντρων αποφάσεων με εικόνες σε αρχείο markdown, σε αντίθεση με το αρχείο απλού κειμένου που εμφανίζονται οι πληροφορίες του, όπως τα μονοπάτια, έως κείμενο.

3.1.2.8 Εξαγωγή συνόλου δεδομένων στον δίσκο (Write System)

Τέλος, είναι δυνατό, άμα ο χρήστης έχει προσθέσει πολλές καινούργιες Labeled κολόνες, να εξάγει το τροποποιημένο σύνολο δεδομένων στη μορφή ενός CSV αρχείου.

3.1.3 Επίλυση των Προβλημάτων

Σε αυτό το κεφάλαιο αναλύονται οι τεχνικές που χρησιμοποιήθηκαν για την επίλυση των προβλημάτων που αναφέρθηκαν στο Κεφάλαιο 3.1.1.

3.1.3.1 Αφαίρεση εξάρτησης Lombok

Όπως αναφέρθηκε και στο Κεφάλαιο 2.2.1.2 η βιβλιοθήκη Lombok προσφέρει annotations που σκοπό έχουν να αυτοματοποιήσουν και να ελαχιστοποιήσουν τις γραμμές κώδικα που χρειάζεται να γράψει και να βλέπει ένας προγραμματιστής. Ωστόσο, θέλαμε να μειώσουμε την εξάρτησή μας σε εξωτερικές βιβλιοθήκες και ήταν περιττή, οπότε αποφασίσαμε να την αφαιρέσουμε τελείως. Για να την αφαιρέσουμε άρκεσε η διαγραφή των annotation αυτών και η εισαγωγή του αντίστοιχου κώδικα.

Στον *Πίνακα 4* παρουσιάζονται τα annotation που υπήρχαν στο υπάρχον λογισμικό και τις ιδιότητες του καθενός:

Annotation	Ιδιότητα
@Getter	Δημιουργεί μεθόδους πρόσβασης για τα αντίστοιχα πεδία.
@Setter	Δημιουργεί μεθόδους θέσης για τα αντίστοιχα πεδία.
@AllArgsConstructor	Δημιουργεί μια μέθοδο δημιουργίας του αντικειμένου που έχει ως παραμέτρους όλα τα πεδία της κλάσης.
@NoArgsConstructor	Δημιουργεί μια μέθοδο δημιουργίας του αντικειμένου με καμία παράμετρο, ακόμα κι αν υπάρχουν πεδία στη κλάση.

Πίνακας 4: Τα annotation της βιβλιοθήκης Lombok και οι ιδιότητές τους.

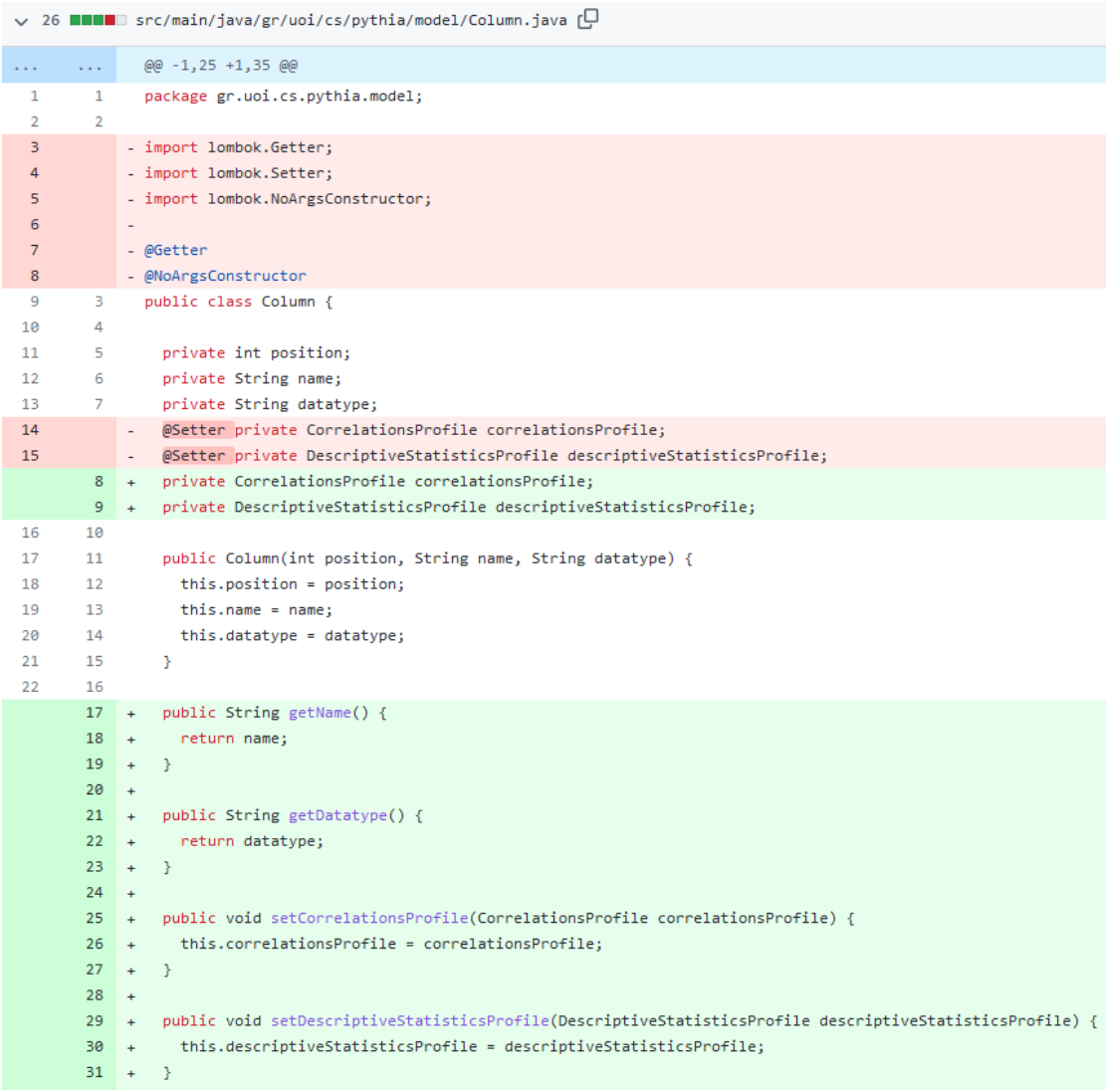
Ύστερα, ακολουθούν μερικά δειγματικά παραδείγματα για την διευκόλυνση της κατανόησης της διαδικασίας αφαίρεσης των annotations αυτών:

14 src/main/java/gr/uoi/cs/pythia/config/SparkConfig.java

```
@@ -3,9 +3,7 @@
3  3  import java.io.IOException;
4  4  import java.io.InputStream;
5  5  import java.util.Properties;
6     - import lombok.Getter;
7  6
8     - @Getter
9  7  public class SparkConfig {
10 8    private String master;
11 9    private String appName;
@@ -25,4 +23,16 @@ public SparkConfig() {
25 23         ex.printStackTrace();
26 24       }
27 25     }
   26 +
   27 +   public String getMaster() {
   28 +     return master;
   29 +   }
   30 +
   31 +   public String getAppName() {
   32 +     return appName;
   33 +   }
   34 +
   35 +   public String getSparkWarehouse() {
   36 +     return sparkWarehouse;
   37 +   }
28 38  }
```

Εικόνα 7: @Getter πάνω από τη κλάση

Στην *Εικόνα 7* παρατηρούμε πως το annotation *@Getter* βρισκόταν πάνω από τη δήλωση της κλάσης. Αυτό υποδηλώνει πως δημιουργούνται μέθοδοι πρόσβασης για όλα τα πεδία της κλάσης. Επομένως, διαγράψαμε το annotation αυτό και το import του ακριβώς από πάνω και εισήγαμε τη μέθοδο πρόσβασης για κάθε πεδίο, που φαίνονται στις σειρές από 26 έως 37.

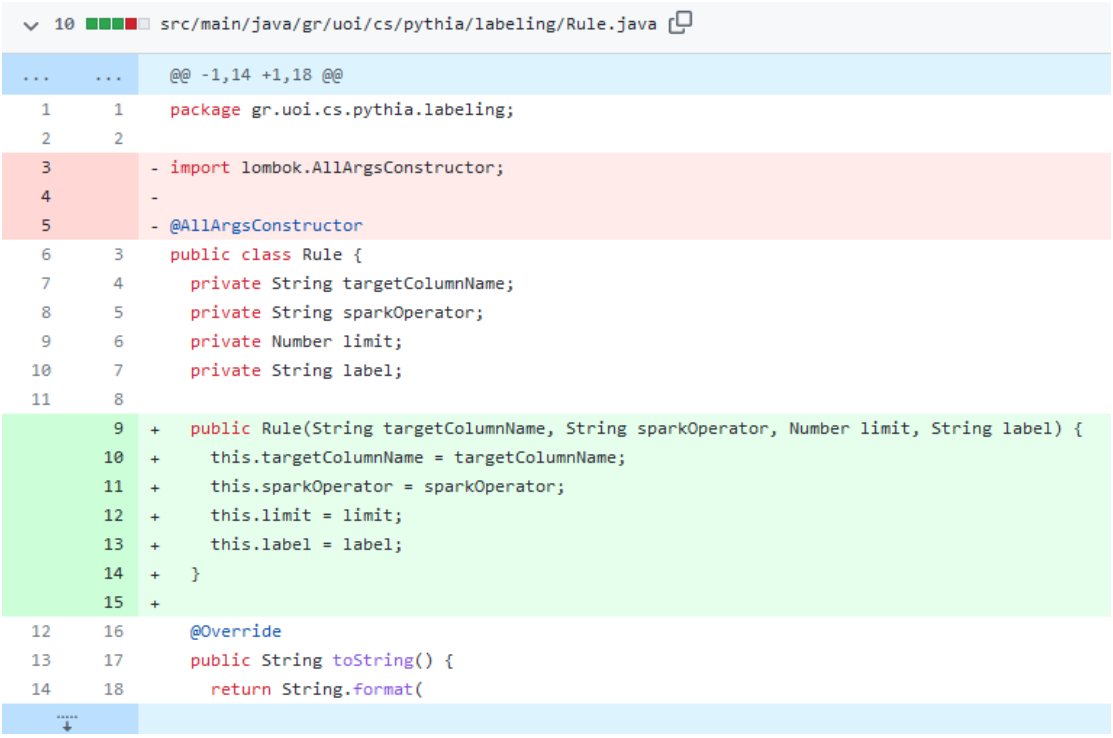


```
26 src/main/java/gr/uoi/cs/pythia/model/Column.java
@@ -1,25 +1,35 @@
package gr.uoi.cs.pythia.model;

- import lombok.Getter;
- import lombok.Setter;
- import lombok.NoArgsConstructor;
-
- @Getter
- @NoArgsConstructor
public class Column {

  private int position;
  private String name;
  private String datatype;
-   @Setter private CorrelationsProfile correlationsProfile;
-   @Setter private DescriptiveStatisticsProfile descriptiveStatisticsProfile;
+   private CorrelationsProfile correlationsProfile;
+   private DescriptiveStatisticsProfile descriptiveStatisticsProfile;

  public Column(int position, String name, String datatype) {
    this.position = position;
    this.name = name;
    this.datatype = datatype;
  }

+   public String getName() {
+     return name;
+   }
+
+   public String getDatatype() {
+     return datatype;
+   }
+
+   public void setCorrelationsProfile(CorrelationsProfile correlationsProfile) {
+     this.correlationsProfile = correlationsProfile;
+   }
+
+   public void setDescriptiveStatisticsProfile(DescriptiveStatisticsProfile descriptiveStatisticsProfile) {
+     this.descriptiveStatisticsProfile = descriptiveStatisticsProfile;
+   }
```

Εικόνα 8: @Getter, @Setter & @NoArgsConstructor

Στην *Εικόνα 8* παρατηρούμε τρία διαφορετικά annotations. Το *@Getter* το χειριζόμαστε όπως και πριν. Το *@Setter* εφαρμόζεται μόνο σε δύο από τα πεδία, κι όχι σε ολόκληρη τη κλάση όπως το *@Getter*, επομένως αρκεί να δημιουργήσουμε δύο μεθόδους θέσης, μόνο για τα πεδία αυτά, όπως φαίνεται στις σειρές 25 μέχρι 31. Το *@NoArgsConstructor* θα αναλογούσε σε μια μέθοδο δόμησης του αντικειμένου δίχως παραμέτρους, αλλά είναι περιττό και δεν χρησιμοποιούταν πουθενά τέτοιος δημιουργός (constructor method), οπότε και το διαγράψαμε τελείως, δίχως αντικατάσταση με κώδικα.

Επομένως, βλέπουμε πως ορισμένα από αυτά τα annotations δεν είχαν χρησιμότητα κάπου και ήταν δύσκολο να καταλάβει ο παρατηρητής άμα χρησιμοποιούταν ή όχι. Στη πράξη, υπήρχαν κι άλλα πεδία με μεθόδους πρόσβασης ή θέσης που δεν χρησιμοποιούνταν.



Εικόνα 9: @AllArgsConstructor

Τέλος, στη *Εικόνα 9* παρατηρούμε τη μετατροπή του *@AllArgsConstructor* annotation στην αντίστοιχη μέθοδο δημιουργίας αντικειμένου με όλες τις παραμέτρους.

3.1.3.2 Ανακατασκευή των Δέντρων Αποφάσεων

Για να καταπολεμήσουμε τα πολλαπλά προβλήματα που απασχολούν την παρούσα κλάση παραγωγής δέντρων αποφάσεων, αποφασίσαμε να τη καταργήσουμε και να δημιουργήσουμε ένα καινούργιο *decisiontree* πακέτο που μοναδικό σκοπό θα έχει την πρέπουσα δημιουργία ενός δέντρου αποφάσεων, την σωστή αναπαράσταση του και όλων των δεδομένων του και τη απεικόνισή του γραφικά για τη τελική αναφορά.

Δημιουργήσαμε για αυτό το σκοπό μία κλάση που βρίσκει τα χαρακτηριστικά και τις κολόνες που παράγουν την κολόνα-στόχο, ενώ μία άλλη επεξεργάζεται τα δεδομένα και τα προετοιμάζει για την ένταξή τους στη διαδικασία δημιουργίας του δέντρου.

Οι πληροφορίες που αναζητούμε από ένα δέντρο απόφασης έχουν μετακινηθεί στις αντίστοιχες κλάσεις δεδομένων, όπως για παράδειγμα μία κλάση που αντιπροσωπεύει το δέντρο και περιέχει την ακρίβεια, τα χαρακτηριστικά και τους κόμβους του. Κάθε κόμβος αποτελεί μια καινούργια κλάση και περιέχει περαιτέρω κλάσεις δεδομένων που περιέχουν τις δικές τους αντίστοιχες πληροφορίες. Όλα τα δεδομένα προέρχονται από τις αρχικές κλάσεις του Apache Spark.

Επιπλέον, εισάγαμε μια κλάση που τα συνδυάζει όλα αυτά και ευθύνεται για τη κατασκευή του δέντρου, επιστρέφοντας το τελικό δέντρο, ενώ μία άλλη δημιουργεί ένα δέντρο για κάθε Labeled Column που έχει ορίσει ο χρήστης.

Κάθε δέντρο, έπειτα, μπορεί να απεικονιστεί σε μορφή εικόνας PNG με τις αντίστοιχες καινούργιες κλάσεις, ώστε μετά να προστεθεί στη τελική αναφορά.

Τελευταίο αλλά εξίσου σημαντικό, ο χρήστης μπορεί να ρυθμίσει τις παραμέτρους που λαμβάνουν χώρο στη δημιουργία του δέντρου απόφασης, όπως το επιθυμητό βάθος του δέντρου.

Επομένως, τώρα τα δέντρα παράγονται σωστά, περιέχουν (οργανωμένα) όλες τις πληροφορίες τους (και μπορούν να αξιοποιηθούν περαιτέρω) και μετατρέπονται εύκολα σε γραφικά που διευκολύνουν τον χρήστη να το καταλάβει.

3.1.3.3 Εμπλουτισμός στατιστικών στοιχείων

Για να αυξήσουμε τη ποιότητα της αναφοράς και τον όγκο στατιστικών στοιχείων που παράγει η υπάρχουσα εφαρμογή, αποφασίσαμε να εντάξουμε διαγράμματα, ξεκινώντας με ένα ιστόγραμμα για κάθε κολόνα. Θα προσφέρουν επιπλέον πληροφορίες και θα είναι εύκολα κατανοητά κατά την ανάγνωση. Αυτή τη στιγμή δε παράγονται σε μορφή γραφικής αναπαράστασης, αλλά εμφανίζονται ως κείμενο, όπου κάθε ιστόγραμμα περιέχει τους κάδους του, και κάθε κάδος το εύρος και το πλήθος των τιμών στο αντίστοιχο εύρος.

3.1.3.4 Καθαρισμός της κονσόλας

Για να καθαρίσουμε τα μηνύματα που δημιουργούνταν στη κονσόλα λόγω του Spark, βρήκαμε το αρχείο *"log4j2.properties"* από το *"core"* πακέτο του Spark και το αντιγράψαμε στον φάκελο *"/src/main/resources/"*. Έπειτα, θέσαμε όλα τα παραγόμενα μηνύματα να γράφονται σε ένα συγκεκριμένο αρχείο, ώστε να μη τα χάσουμε, και ορίσαμε στην κονσόλα να εμφανίζονται μόνο τα μηνύματα του Pythia.

3.1.3.5 Μεταφορά ευθυνών της κεντρικής μηχανής

Η κεντρική μηχανή του Pythia (DatasetProfiler) είχε πολλές ευθύνες για τις οποίες δεν ήταν προορισμένη να κάνει, με κύριο υπαίτιο τον υπολογισμό των περιγραφικών στατιστικών. Δημιουργήσαμε ένα καινούργιο πακέτο *"descriptivestats"* (περιγραφικά στατιστικά) και μεταφέραμε τη μέθοδο που τα υπολόγιζε σε μια αυτοτελής κλάση, μαζί με τη κλάση *"DatasetProfilerConstants"*, η οποία μετονομάστηκε σε *"DescriptiveStatisticsConstants"* και διαγράφηκε ο κώδικάς της που δε χρησιμοποιούταν

πουθενά. Πλέον, η κεντρική μηχανή καλεί αυτό το πακέτο για να υπολογίσει τα περιγραφικά στατιστικά. Επίσης, απλοποιήθηκε μέρος του κώδικα στη κεντρική μηχανή, που ήταν υπεύθυνο για την εγγραφή ενός συνόλου δεδομένων, δίχως να αλλάξει το τελικό αποτέλεσμα. Επίσης, τροποποιήθηκε ένα ακόμα μέρος του πηγαίου κώδικα ώστε να φαίνονται πιο ξεκάθαρα οι εξαρτήσεις στη κεντρική μηχανή.

3.1.3.6 Βελτίωση της τελικής αναφοράς

Τέλος, αξίζει να σημειωθεί η βελτίωση της τελικής αναφοράς. Προστέθηκε η επιλογή παραγωγής τύπου αρχείου markdown που εκτός από απλό κείμενο, υποστηρίζει τίτλους, λίστες, πίνακες, συνδέσμους, εικόνες και μερικές τροποποιήσεις στο κείμενο, όπως υπογράμμιση, πλάγια και έντονη γραφή. Αξιοποιώντας τις δομές που προσφέρει η γλώσσα markdown, καταφέραμε να δημιουργήσουμε ένα κείμενο ευανάγνωστο, ακόμα και όταν ο χρήστης διαβάζει τον πηγαίο αρχείου, δίχως να χρησιμοποιεί ένα πρόγραμμα προβολής σήμανσης.

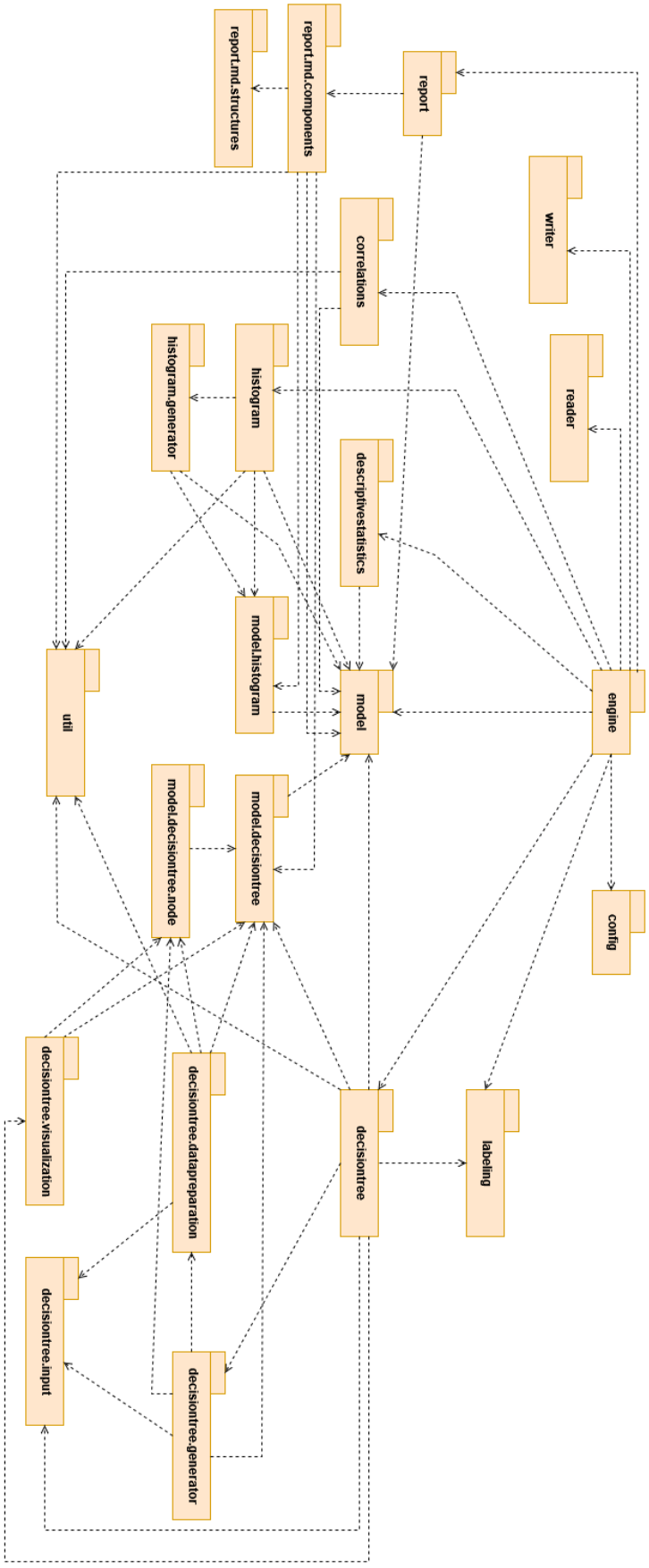
3.2 Σχεδίαση και αρχιτεκτονική λογισμικού

Στο Pythia κάθε λειτουργία υλοποιείται σε διαφορετικό πακέτο, με εξαίρεση τη δημιουργία του στατιστικού προφίλ, όπου υπάρχουν διαφορετικά πακέτα για κάθε παραγόμενη στατιστική μέτρηση ή μοντέλο. Στη συνέχεια, θα μελετήσουμε τη συνολική δομή του εργαλείου, καθώς και κάθε πακέτο ξεχωριστά, αναλύοντας τις συσχετίσεις και τις εξαρτήσεις μεταξύ των κλάσεών του.

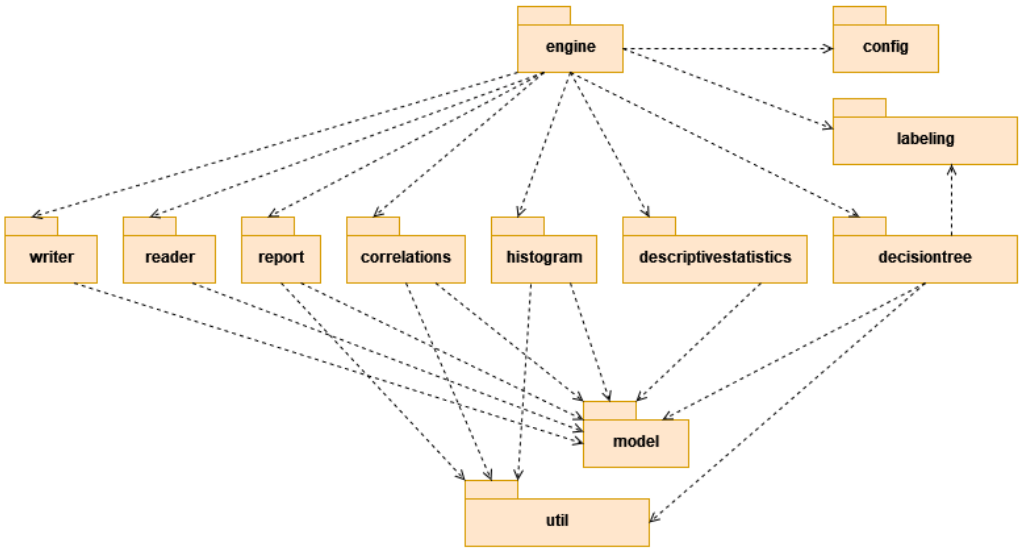
3.2.1 Διάγραμμα πακέτων

Στην επόμενη σελίδα φαίνεται το αναλυτικό διάγραμμα πακέτων του Pythia.

Κάθε λειτουργία έχει το δικό της πακέτο και το πακέτο "*engine*" καλεί αυτά τα κομμάτια. Όπως βλέπουμε, όλα τα πακέτα εξαρτούνται από το πακέτο "*model*" και τα υποπακέτα του, εφόσον το πρόγραμμα ακολουθάει μια δομή όπου όλες οι κλάσεις αναπαράστασης δεδομένων βρίσκονται εκεί. Άμα παρατηρήσουμε καλύτερα, δεν υπάρχουν κύκλοι, και άρα όχι αλληλεξαρτήσεις μεταξύ των κλάσεων.


report.md.structures
report.md.components
report
writer
correlations
histogram.generator
histogram
reader
descriptivestatistics
model.histogram
util
model
engine
model.decisiontree.node
model.decisiontree
config
decisiontree.visualization
decisiontree.dataprepparation
decisiontree
labeling
decisiontree.input
decisiontree.generator

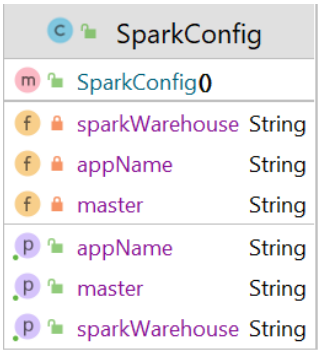
Για τη διευκόλυνσή μας, παρουσιάζεται και ένα πιο απλοϊκό διάγραμμα πακέτων ολόκληρου του συστήματος:



Εικόνα 10: Απλουστευμένο διάγραμμα πακέτων

Στην *Εικόνα 10* φαίνεται καλύτερα η δομή του προγράμματος. Παρατηρούμε την ύπαρξη ενός μοτίβου, όπου κάθε λειτουργία έχει το δικό της πακέτο. Ένας μελλοντικός προγραμματιστής αυτού του εργαλείου, θα κλίνει προς τη δημιουργία ενός νέου πακέτου για την ένταξη νέων στατιστικών ή και λειτουργιών.

3.2.2 Πακέτο config

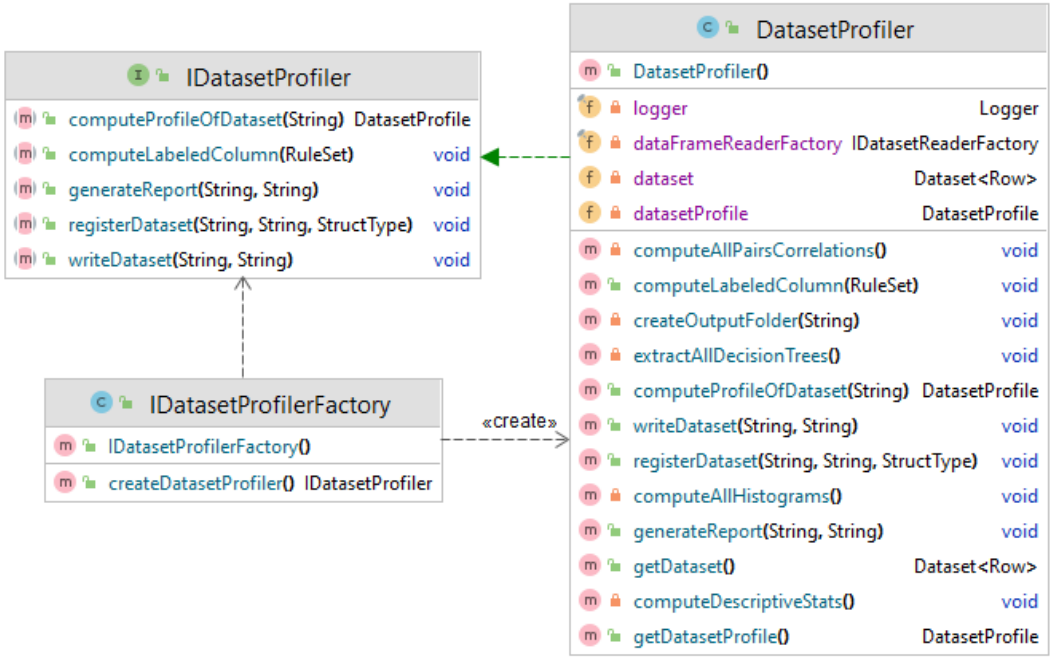


Εικόνα 11: Διάγραμμα UML για το πακέτο config

Το πακέτο *config* περιέχει μία κλάση η οποία είναι υπεύθυνη για την ρύθμιση των παραμέτρων εκκίνησης του Spark. Οι παράμετροι αυτοί βρίσκονται στο αρχείο "*spark.properties*" στον φάκελο "*src/main/resources/*", όπου και κάποιος μπορεί να τις

αλλάξει. Στην τρέχουσα υλοποίηση, το Spark έχει ρυθμιστεί να τρέχει τοπικά και να χρησιμοποιεί όλους τους πόρους του επεξεργαστή.

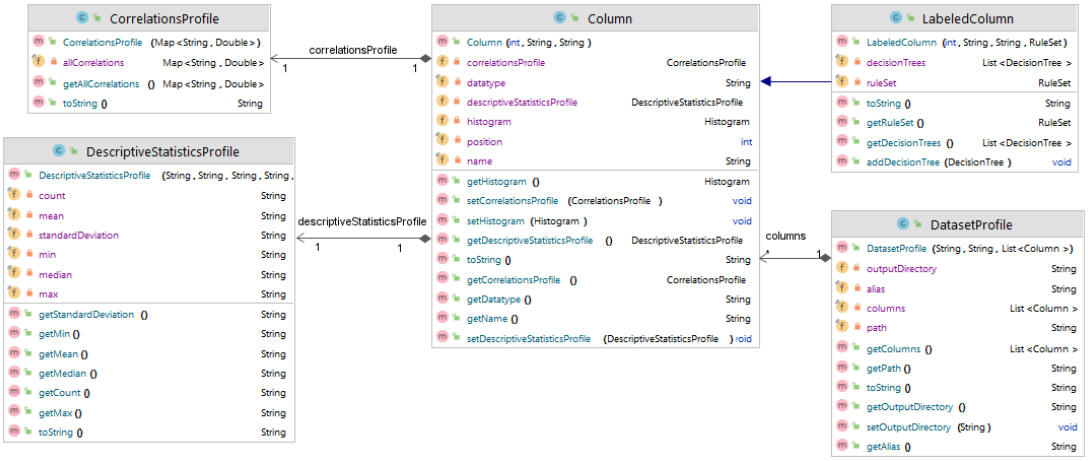
3.2.3 Πακέτο engine



Εικόνα 12: Διάγραμμα UML για το πακέτο engine

Το πακέτο *engine* αποτελεί τη κεντρική μηχανή του συστήματος και τη διεπαφή με την οποία αλληλοεπιδρά ο χρήστης. Ακολουθεί τον συνδυασμό Factory Pattern και διεπαφής (Interface), δηλαδή κάθε υλοποίησή του πρέπει να συμβιβάζεται με τη διεπαφή *IDatasetProfiler* και παράγεται από την κλάση *IDatasetProfilerFactory*. Ο χρήστης έρχεται αντιμέτωπος μόνο με τη διεπαφή και τις λειτουργίες της, και δεν χρειάζεται να ασχοληθεί με τον τρόπο με τον οποίο έχει φτιαχτεί. Η κλάση *DatasetProfiler*, η οποία είναι και η μοναδική υλοποίησή μας, είναι υπεύθυνη για τη εγγραφή και την αποθήκευση ενός συνόλου δεδομένων, τον υπολογισμό του στατιστικού του προφίλ, καθώς και την παραγωγή της τελικής αναφοράς του. Επίσης, παρέχει τη δυνατότητα να προστεθεί μία κολόνα βάση κανόνων που θα θέσει ο χρήστης, μέσω της μεθόδου *computeLabeledColumn()*. Στο παρασκήνιο, καλεί τις ανάλογες κλάσεις από τα ξεχωριστά πακέτα του προγράμματος Pythia, το οποίο και ξεχωρίζει στην *Εικόνα 12* από τα ονόματα των ιδιωτικών της μεθόδων, όπως παραδείγματος χάριν *computeDescriptiveStats()*, *computeAllHistograms()* και *extractAllDecisionTrees()*.

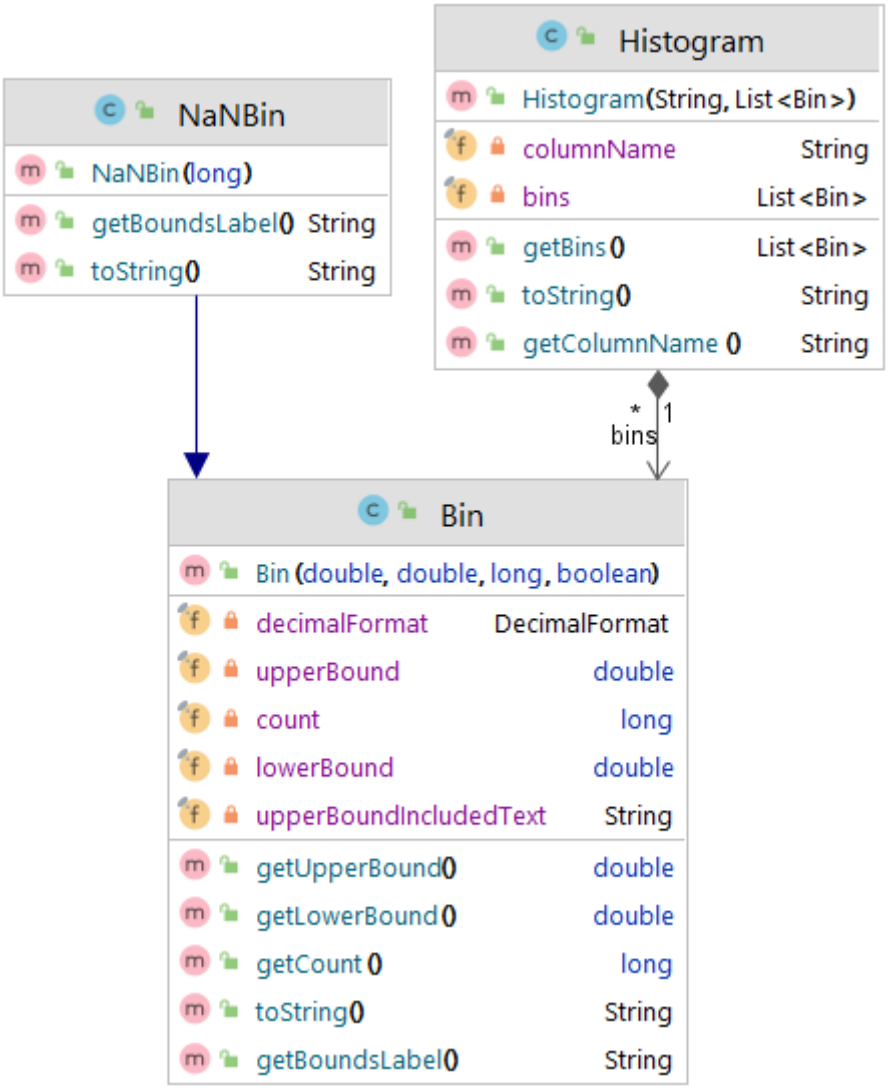
3.2.4 Πακέτο model



Εικόνα 13: Διάγραμμα UML για το πακέτο model

Οι κλάσεις στο πακέτο *model* και τα υποπακέτα του, αποτελούν την αναπαράσταση των στατιστικών αποτελεσμάτων που έχουμε δημιουργήσει για ένα σύνολο δεδομένων. Κάθε στατιστικό στοιχείο στο Pythia έχει την αναπαράστασή του ως μία κλάση. Τα περιγραφικά στατιστικά έχουν τη κλάση *DescriptiveStatisticsProfile*, οι συσχετίσεις τη κλάση *CorrelationsProfile*, τα ιστογράμματα τη κλάση *Histogram*, κ.τ.λ. Συγκεκριμένα στατιστικά στοιχεία έχουνε και το δικό τους υποπακέτο για την αναπαράστασή τους μέσα στο πακέτο *model*, επειδή η δομή τους είναι πιο περίπλοκη. Ύστερα της στατιστικής ανάλυση του Pythia, δημιουργούνται αντικείμενα κλάσης *Column* για κάθε ένα από τα πεδία του συνόλου δεδομένων και μέσα σε αυτή τη κλάση αποθηκεύουμε τις σχετικές αναπαραστάσεις των στατιστικών αποτελεσμάτων. Επομένως, όλα τα δεδομένα για ένα πεδίο βρίσκονται στη κλάση *Column*. Η κλάση *DatasetProfiler* είναι η κύρια δομή του προγράμματος Pythia, εφόσον εμπεριέχει μία λίστα με όλα τα αντικείμενα *Column* που έχουν δημιουργηθεί, και άρα όλα τα δεδομένα. Επιπλέον, συγκρατεί πληροφορίες όπως το όνομα του συνόλου των δεδομένων και τη διεύθυνσή του στον υπολογιστή. Επίσης, υπάρχει και μια παραλλαγή της κλάσης *Column* που απευθύνεται στα *labeled* πεδία, δηλαδή αυτά που δημιούργησε ο χρήστης μέσω κανόνων. Κάθε *labeled* πεδίο αναπαρίσταται με τη κλάση *LabeledColumn*, η οποία κληρονομεί τη κλάση *Column*, δηλαδή έχει όλα τα πεδία της και επιπρόσθετα περιέχει πληροφορίες όπως τους κανόνες *(RuleSet)* που το δημιούργησαν και τα δέντρα αποφάσεων που παράχθηκαν για αυτό.

3.2.5 Πακέτο model.histogram

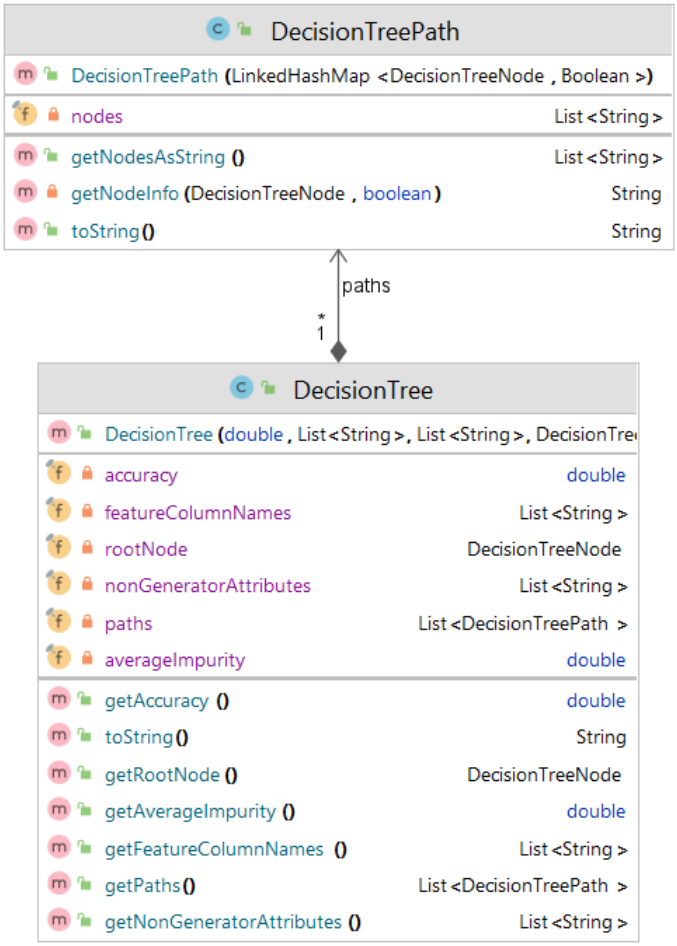


Εικόνα 14: Διάγραμμα UML για το πακέτο model.histogram

Ένα ιστόγραμμα στο Pythia αναπαρίσταται με τη κλάση *Histogram*, η οποία είναι μέρος της κλάσης *Column*. Ένα αντικείμενο *Histogram* περιέχει το όνομα του πεδίου για το οποίο δημιουργήθηκαν τα δεδομένα, καθώς και μία συλλογή από κάδους. Ένα αντικείμενο *Bin* ή *NaNBin*, χρησιμοποιείται για να αποθηκεύσουμε τα δεδομένα ενός κάδου. Ένας κάδος κλάσης *Bin* περιέχει δύο τιμές που περιγράφουν το εύρος του κάδου, δηλαδή το άνω και το κάτω όριο. Το άνω όριο δεν συμπεριλαμβάνεται πάντα στο εύρος, οπότε υπάρχει μία μεταβλητή που το ενδεικνύει και αυτό. Φυσικά, περιέχει και μία μεταβλητή που περιγράφει το πλήθος των τιμών στο εύρος αυτό. Ένας κάδος κλάσης *NaNBin* δεν είναι τίποτα άλλο από έναν κάδο ο οποίος συγκρατεί το πλήθος των τιμών που δεν είναι αριθμοί, όπως *null* ή κενές εγγραφές. Η κλάση *NaNBin* υιοθετεί από τη

κλάση *Bin* επειδή περιγράφει κι αυτή κάδο, ωστόσο δεν έχει άνω και κάτω όρια εκ της φύσεως του. Οι μη αριθμητικές τιμές δε βρίσκονται σε κάποιο εύρος.

3.2.6 Πακέτο model.decisiontree

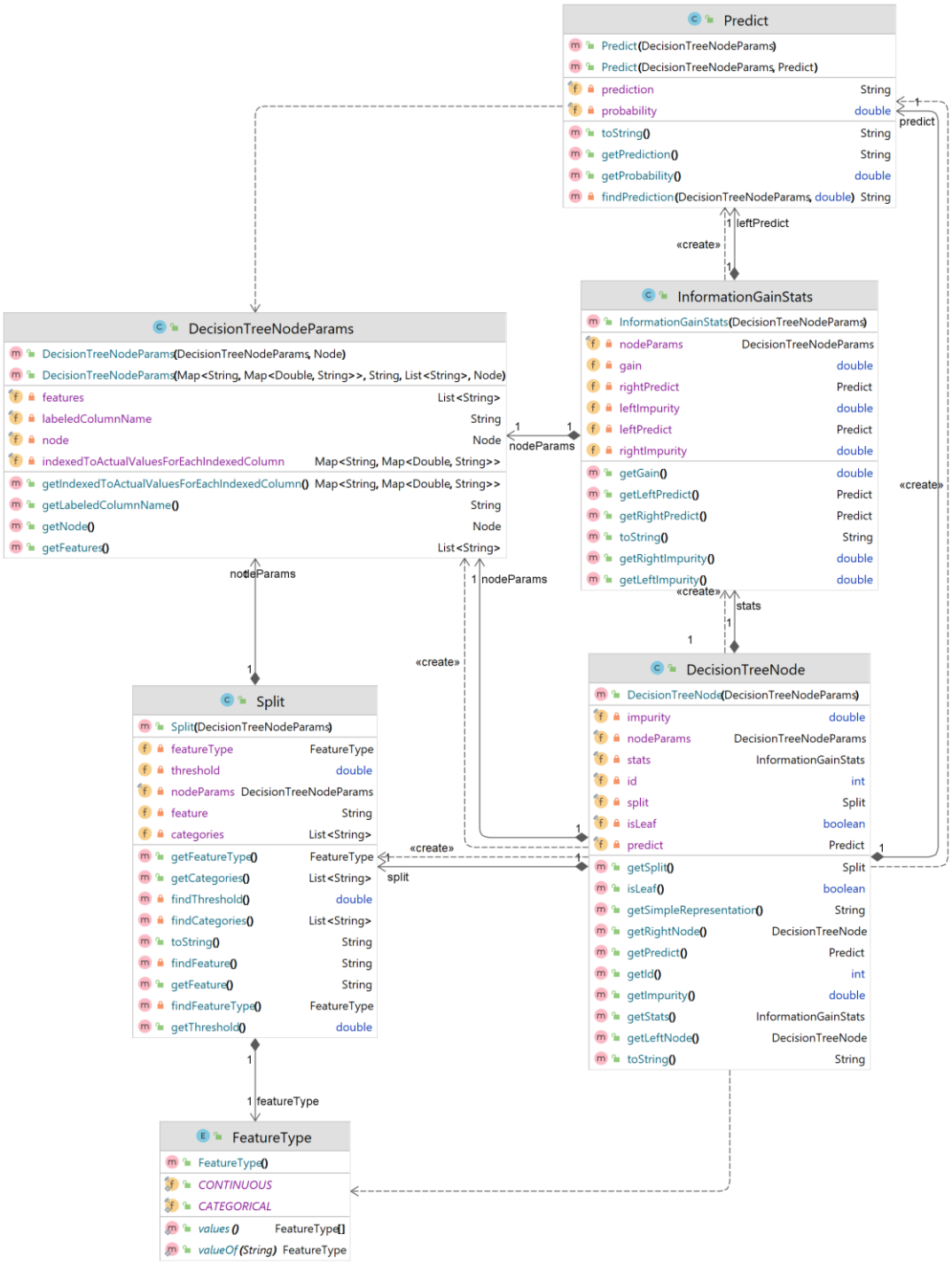


Εικόνα 15: Διάγραμμα UML για το πακέτο model.decisiontree

Το πακέτο *model.decisiontree* περιέχει τις απαραίτητες κλάσεις για την αναπαράσταση ενός δέντρου αποφάσεων. Κύρια δομή είναι η κλάση *DecisionTree,* στην οποία βρίσκονται όλα τα χαρακτηριστικά ενός δέντρου αποφάσεων, όπως η ακρίβεια του συνολικού δέντρου, τα συμπεριλαμβανόμενα και μη χαρακτηριστικά, η μέση πρόσμειξη των φύλλων του, τα μονοπάτια του δέντρου και ο ριζικός κόμβος του. Μονοπάτια σε ένα δέντρο αποφάσεων θεωρούνται οι διακριτές διαδρομές από τον ριζικό κόμβο προς κάθε φύλλο του. Ένα μονοπάτι χαρακτηρίζεται με τη κλάση *DecisionTreePath* και περιέχει μερικές βασικές πληροφορίες για κάθε κόμβο που διανύει, όπως τη πρόσμειξη, το κριτήριο διαχωρισμού, κ.α. Χρησιμοποιείται κυρίως για την τελική αναφορά. Ο ριζικός κόμβος

δίνει πρόσβαση σε όλους τους υπόλοιπους κόμβους, διανύοντας το δέντρο προς τα δεξιά ή αριστερά. Θα συζητηθεί περαιτέρω στο επόμενο πακέτο *model.decisiontree.node*.

3.2.7 Πακέτο model.decisiontree.node



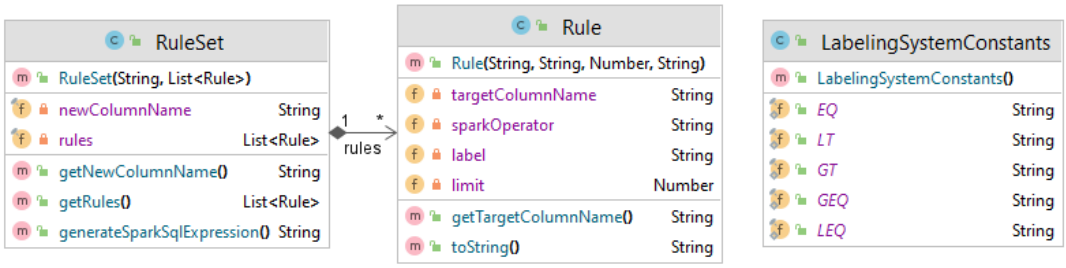
Εικόνα 16: Διάγραμμα UML για το πακέτο model.decisiontree.node

Το πακέτο *model.decisiontree.node* αποτελεί συνέχεια (ή επέκταση) του προηγούμενου πακέτου *model.decisiontree* και ασχολείται με τη περιγραφή και αναπαράσταση των δεδομένων ενός κόμβου ενός δέντρου αποφάσεων.

Όπως φαίνεται και στην *Εικόνα 16*, η βασική δομή είναι η κλάση *DecisionTreeNode*, στην οποία βρίσκονται όλα τα στοιχεία του κόμβου. Συμπεριλαμβάνονται η πρόσμειξη, οι παράμετροι για να δημιουργηθεί ο επόμενος αριστερός ή δεξής κόμβος του δέντρου και των πεδίων του, ο αναγνωριστικός αριθμός του κόμβου *(id)* και οι κλάσεις *Predict, InformationGainStats* και *Split*. Η κλάση *Split* συγκρατεί τα στοιχεία διαχωρισμού, όπως το όνομα του χαρακτηριστικού, το όριο διαχωρισμού των δεδομένων για συνεχείς πεδία ή τις συμπεριλαμβανόμενες τιμές για κατηγορικά, καθώς και τον τύπο του χαρακτηριστικού, δηλαδή άμα είναι συνεχή ή κατηγορικό. Η κλάση *InformationGainStats* περιέχει το κέρδος, καθώς και τις προβλέψεις και τους βαθμούς πρόσμειξης του κάθε διαχωριστικού του κόμβου. Στη κλάση *Predict* βρίσκεται η πρόβλεψη και η πιθανότητα της πρόβλεψης.

Για τη δημιουργία και τη θέση των δεδομένων των προαναφερθέντων κλάσεων, υπάρχει η κλάση *DecisionTreeNodeParams*. Η *DecisionTreeNodeParams* περιέχει διάφορες χρήσιμες πληροφορίες που χρειάζονται κατά τη κατασκευή των κλάσεων αυτών, όπως το όνομα της κολόνας-στόχου και την αληθινή τιμή ενός κατηγορικού πεδίου που είχε μετατραπεί σε αριθμητική αναπαράσταση (*actual to indexed value*) κατά τη διάρκεια του υπολογισμού. Τέλος, υπάρχει η κλάση *FeatureType* που χρησιμοποιείται για να δείχνουμε άμα το χαρακτηριστικό είναι διακριτό ή κατηγορικό.

3.2.8 Πακέτο labeling

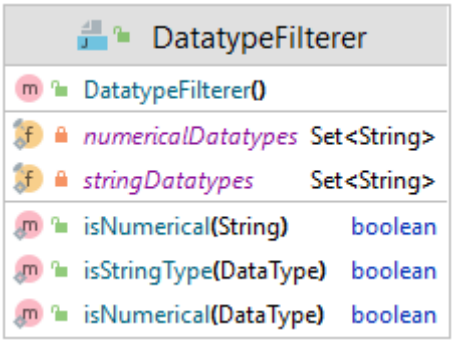


Εικόνα 17: Διάγραμμα UML του πακέτου labeling

Το συγκεκριμένο πακέτο παρέχει τις κλάσεις που αναπαριστούν και δίνουν τη δυνατότητα στον χρήστη να δημιουργήσει μία καινούργια *labeled* κολόνα στο σύνολο δεδομένων. Η κλάση *Rule* αναπαριστά έναν κανόνα που απευθύνεται σε μία από τις υπάρχουσες κολόνες του συνόλου δεδομένων και δηλώνει ένα όριο, ή αλλιώς εύρος

τιμών, για το οποίο θα αντιστοιχεί μία δηλωμένη τιμή. Για παράδειγμα για την κολόνα «Ηλικία», για τα άτομα κάτω των 40 ετών, θέτουμε τη τιμή «Νέος». Ο ορισμός του ορίου γίνεται μέσω τελεστών του Spark (<, >, <=, >=, =) και χρησιμοποιείται η κλάση *LabelingSystemConstants* για την αναπαράστασή τους, ώστε να αφαιρεθεί η πολυπλοκότητα από τον χρήστη. Όλοι οι κανόνες συλλέγονται στη κλάση *RuleSet*, η οποία είναι υπεύθυνη για τη παραγωγή της τελικής έκφρασης SQL που θα περάσουμε στο Spark για να δημιουργηθεί η νέα *labeled* κολόνα.

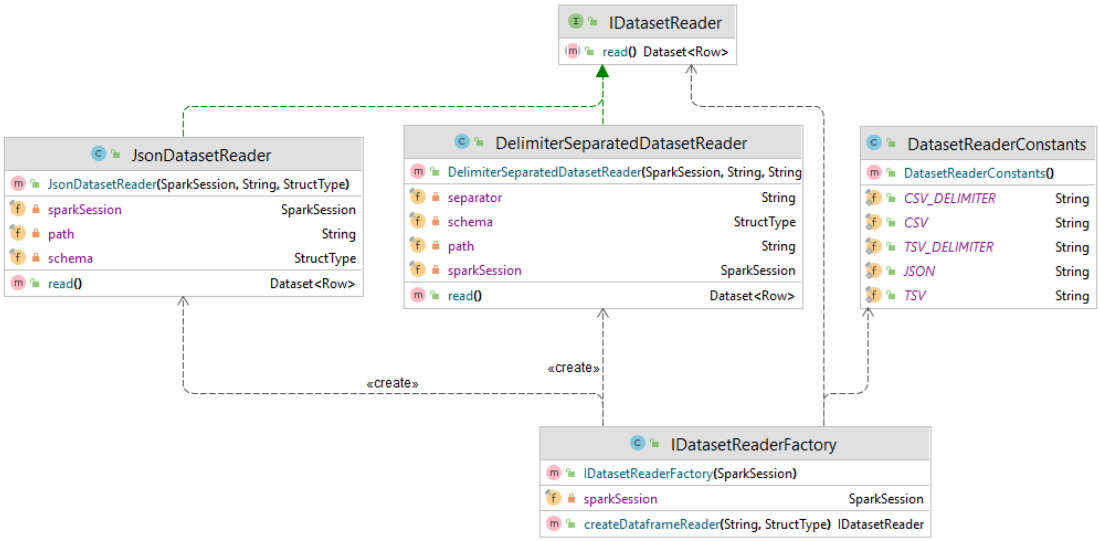
3.2.9 Πακέτο util



Εικόνα 18: Διάγραμμα UML του πακέτου util

Το πακέτο *util*, όπως υποδηλώνει και το όνομά του, περιέχει βοηθητικές λειτουργίες. Αποτελείται από συναρτήσεις και μεθόδους που λαμβάνουν μέρος σε πολλά πακέτα του εργαλείου, τα οποία όλα απαιτούν την ίδια ακριβώς λειτουργία. Η μοναδική κλάση που εμπεριέχεται αυτή τη στιγμή είναι η *DatatypeFilterer* που χρησιμοποιείται για να αναγνωριστεί ο τύπος μιας κολόνας, δηλαδή άμα είναι αριθμητική, απλό κείμενο ή κάτι άλλο.

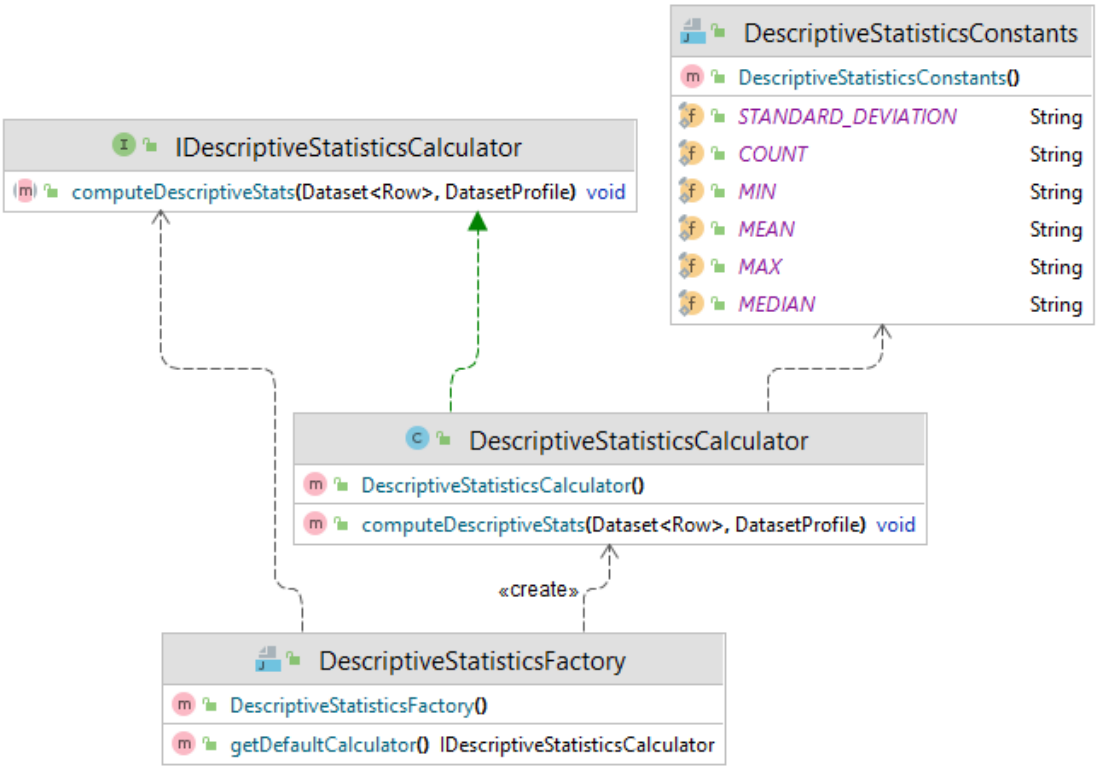
3.2.10 Πακέτο reader



Εικόνα 19: Διάγραμμα UML του πακέτου reader

Το πακέτο *reader* περιέχει τις λειτουργίες διαβάσματος συνόλων δεδομένων και υποστηρίζει πολλά είδη αρχείων, όπως *".csv", ".tsv" και ".json"*. Χρησιμοποιείται κι εδώ ο συνδυασμός *Factory Pattern* με διεπαφή, όπου η διεπαφή είναι η κλάση *IDatasetReader* και θέτει ως κανόνα μόνο μία μέθοδο, ενώ η κλάση *IDatasetReaderFactory* δημιουργεί τις διάφορες υλοποιήσεις της διεπαφής. Η επιλογή της υλοποίησης που θα χρησιμοποιηθεί γίνεται από τον χρήστη μέσω της κλάσης *DatasetReaderConstants*, η οποία προσφέρει τις διαθέσιμες επιλογές για τον τελικό χρήστη. Το πραγματικό διάβασμα γίνεται μέσω της βιβλιοθήκης Spark, ωστόσο η ρύθμιση των παραμέτρων για κάθε διαφορετικό τύπο αρχείου γίνεται μέσω των υλοποιήσεών μας. Συγκεκριμένα, υπάρχει η κλάση *JsonDatasetReader* η οποία διαβάζει αρχεία τύπου *".json"*, ενώ η κλάση *DelimiterSeparatedReader* ασχολείται με τα υπόλοιπα ήδη αρχείων που έχουν έναν διαχωριστικό χαρακτήρα, όπως η κόμμα στα αρχεία *".csv"* ή το *"tab"* στα αρχεία *".tsv"*.

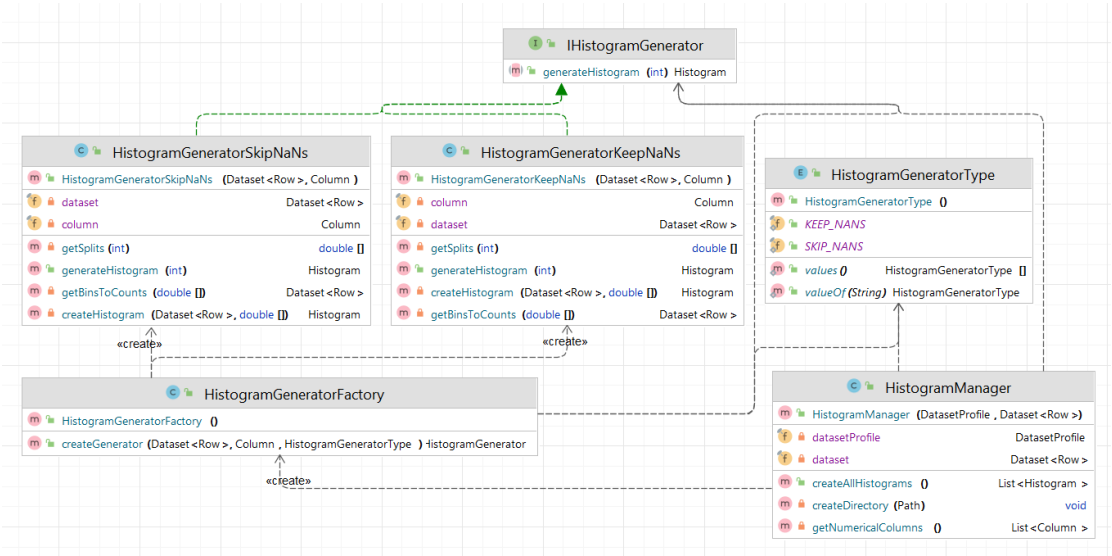
3.2.11 Πακέτο descriptivestatistics



Εικόνα 20: Διάγραμμα UML του πακέτου descriptivestatistics

Το πακέτο *descriptivestatistics* έχει αναλάβει τη δουλειά της δημιουργίας των περιγραφικών στατιστικών. Μέσα του βρίσκονται η κλασσική διεπαφή και Factory κλάση, καθώς και η μοναδική υλοποίηση της διεπαφής, η κλάση *DescriptiveStatisticsCalculator*, η οποία χρησιμοποιεί τις μεθόδους του Spark για να υπολογίσει τα στατιστικά για κάθε πεδίο του συνόλου δεδομένων. Το περιγραφικά στατιστικά που θα υπολογιστούν, ορίζονται στη κλάση DescriptiveStatisticsConstants ως σταθερές μεταβλητές.

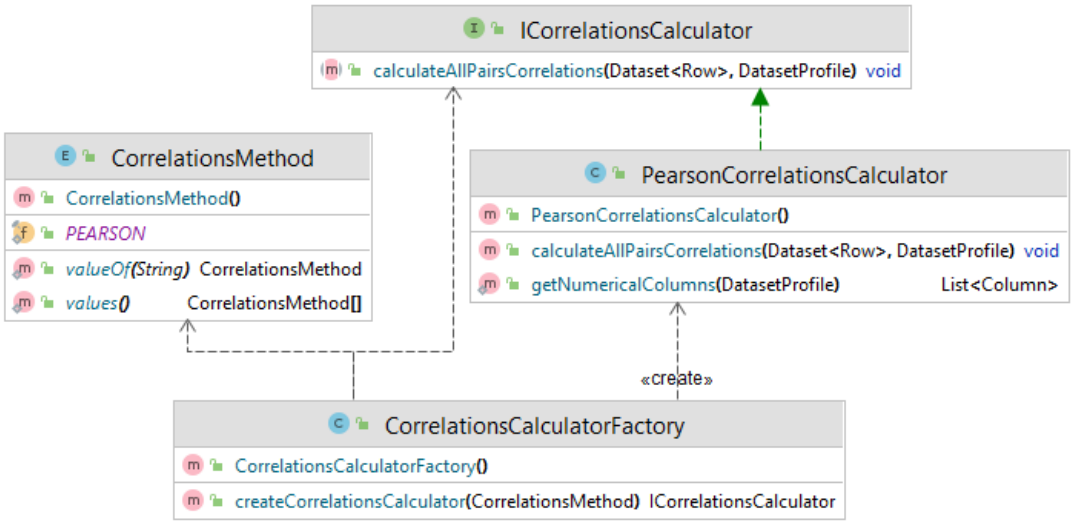
3.2.12 Πακέτο histogram



Εικόνα 21: Διάγραμμα UML του πακέτου histogram

Στο πακέτο *histogram* κατασκευάζονται όλα ιστογράμματα. Η κλάση *HistogramManager*, η οποία αποτελεί και την «βιτρίνα» του πακέτου, διακρίνει τα αριθμητικά πεδία του συνόλου δεδομένων και παράγει ιστογράμματα μόνο για αυτά. Η παραγωγή των ιστογραμμάτων γίνεται μέσω των κλάσεων του πακέτου *histogram.generator*, δηλαδή ένα υποπακέτο του *histogram*. Το πακέτο *histogram.generator* ακολουθεί τη δομή *Factory Pattern* με διεπαφή (*Interface*) και έχει δύο βασικές υλοποιήσεις, την *HistogramGeneratorSkipNaNs* και την *HistogramGeneratorKeepNaNs.* Η *HistogramGeneratorSkipNaNs* δε προσμετράει τις μη αριθμητικές τιμές (κενές, null, NaN) στους κάδους της, ενώ η *HistogramGeneratorKeepNaNs* δημιουργεί έναν έξτρα κάδο με αυτές. Η αναπαράστασή τους γίνεται μέσω των κλάσεων του πακέτου *model.histogram*.

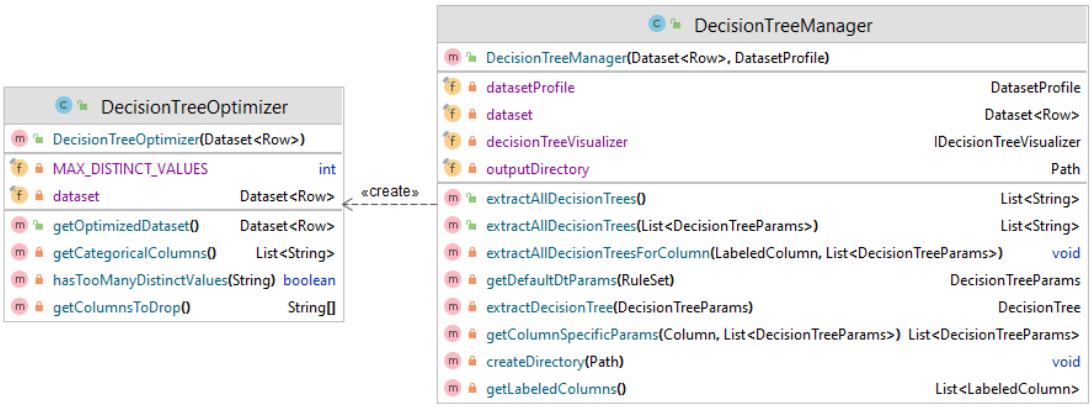
3.2.13 Πακέτο correlations



Εικόνα 22: Διάγραμμα UML του πακέτου correlations

Το πακέτο *correlations* περιέχει τις κλάσεις υπεύθυνες για τον υπολογισμό των συσχετίσεων μεταξύ των πεδίων του δοθέντα συνόλου δεδομένων. Έχει δημιουργηθεί ένα *Factory Pattern* σε συνδυασμό με μια διεπαφή, όπου κάθε υλοποίηση ενός αλγορίθμου υπολογισμού συσχετίσεων ακολουθεί τη διεπαφή *ICorrelationsCalculator* και η κλάση *CorrelationsCalculatorFactory* είναι υπεύθυνη για την δημιουργία επιλεχθέντας υλοποίησης. Η επιλογή γίνεται μέσω της κλάσης *CorrelationsMethod* τύπου *enum*. Αυτή τη στιγμή ωστόσο, υπάρχει μόνο ένας υλοποιημένος αλγόριθμος, του Pearson, στην αντίστοιχη κλάση *PearsonCorrelationsCalculator*.

3.2.14 Πακέτο decisiontree



Εικόνα 23: Διάγραμμα UML του πακέτου decisiontree

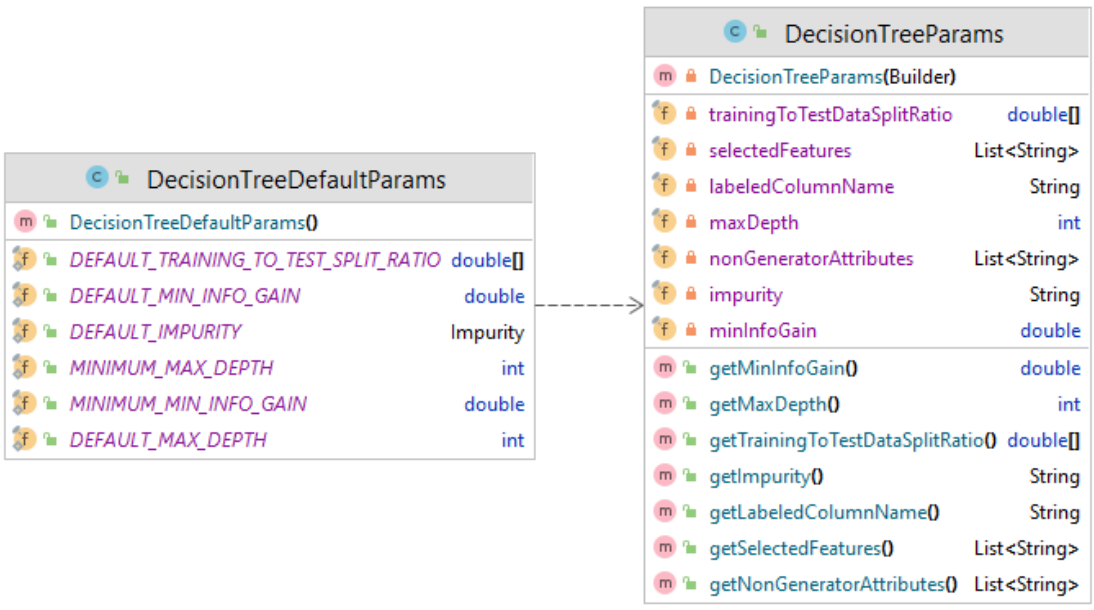
Το πακέτο *decisiontree* περιέχει όλες τις κλάσεις και τα υποπακέτα που ασχολούνται με τον υπολογισμό, την δημιουργία και την γραφική αναπαράσταση των δέντρων

αποφάσεων. Στις επόμενες υποενότητες, θα αναλύσουμε τις ευθύνες του κάθε πακέτου ξεχωριστά. Στο συγκεκριμένο πακέτο βρίσκονται δύο κλάσεις, η *DecisionTreeManager* και η *DecisionTreeOptimizer*.

Η *DecisionTreeOptimizer* βελτιστοποιεί τα δεδομένα που λαμβάνουν μέρος στον υπολογισμό του δέντρου και αφαιρεί τα χαρακτηριστικά που είναι κατηγορικά και έχουν παραπάνω από 32 διαφορετικές τιμές. Σκοπός είναι η μείωση του κόστους υπολογισμού ανούσιων δεδομένων, γιατί τα πεδία που αφαιρούμε συνήθως έχουνε χιλιάδες ή παραπάνω διαφορετικές τιμές.

Η κλάση *DecisionTreeManager* είναι υπεύθυνη για τη παραγωγή όλων των δέντρων αποφάσεων, είτε έχουν δοθεί παράμετροι για τα δέντρα, είτε όχι. Καλείται από την κεντρική μηχανή και παρέχει δύο μεθόδους, η μία εκ των οποίων επιτρέπει στον χρήστη να εισάγει μία λίστα με τις παραμέτρους που θέλει για κάθε δέντρο αποφάσεων. Στη περίπτωση που δε δοθεί η λίστα, είναι κενή ή χρησιμοποιηθεί η άλλη μέθοδος που δε δέχεται παραμέτρους, τότε η κλάση *DecisionTreeManager* παράγει μόνη της τις προκαθορισμένες παραμέτρους για κάθε δέντρο. Επίσης, θέτει σε κάθε *LabeledColumn* κλάση τα δέντρα αποφάσεων και τα εξάγει ως εικόνα *".png"* στον φάκελο που έχει οριστεί η παραγωγή τους, ώστε να χρησιμοποιηθούν αργότερα στην τελική αναφορά.

3.2.15 Πακέτο decisiontree.input

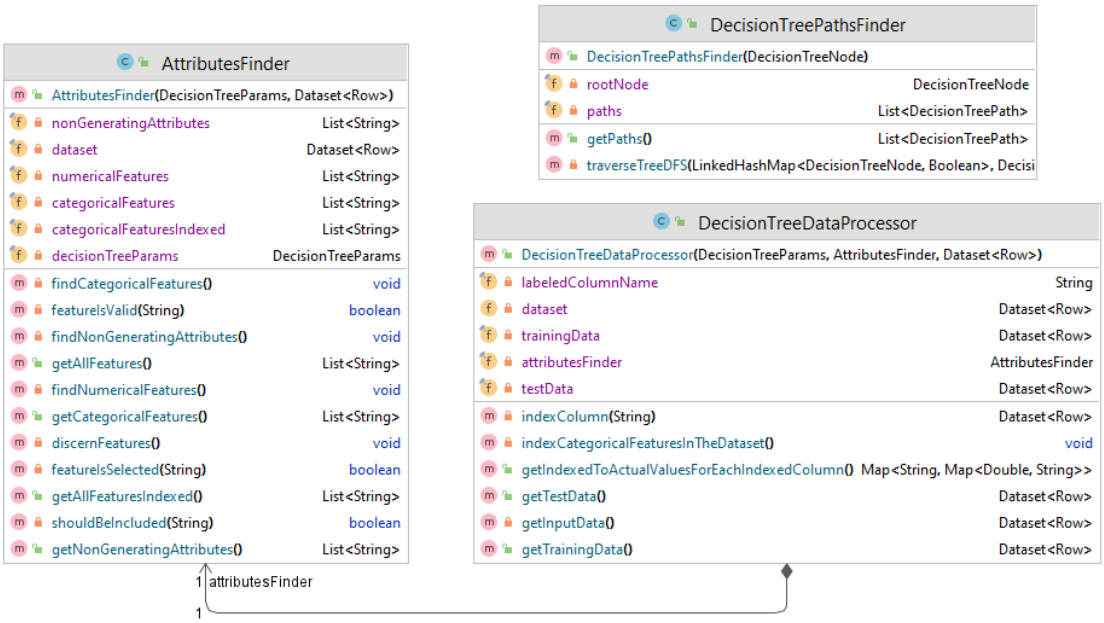


Εικόνα 24: Διάγραμμα UML του πακέτου decisiontree.input

Το πακέτο *decisiontree.input* είναι ένα υποπακέτο του *decisiontree* και περιέχει τις κλάσεις παραμετροποίησης των δέντρων αποφάσεων. Ο χρήστης μπορεί να θέσει το

ελάχιστο κέρδος, τον αλγόριθμο επιλογής χαρακτηριστικών (*Gini ή Entropy*), το βάθος, τα χαρακτηριστικά που θα συμβάλουν στη δημιουργία του δέντρου και τη ποσότητα των δεδομένων που θα χρησιμοποιηθούν ως δεδομένα εκπαίδευσης και δοκιμής. Όλα αυτά θέτονται μέσω της κλάσης *DecisionTreeParams*, η οποία έχει υλοποιηθεί με *Builder Pattern*, δηλαδή έχει προκαθορισμένες τιμές για κάθε παράμετρο που αναφέρθηκε και ο χρήστης μπορεί να την αλλάξει. Οι προκαθορισμένες τιμές των παραμέτρων ορίζονται στη κλάση *DecisionTreeDefaultParams* και ως βάση έχουν τις ήδη προκαθορισμένες τιμές από το Spark.

3.2.16 Πακέτο decisiontree.datapreparation



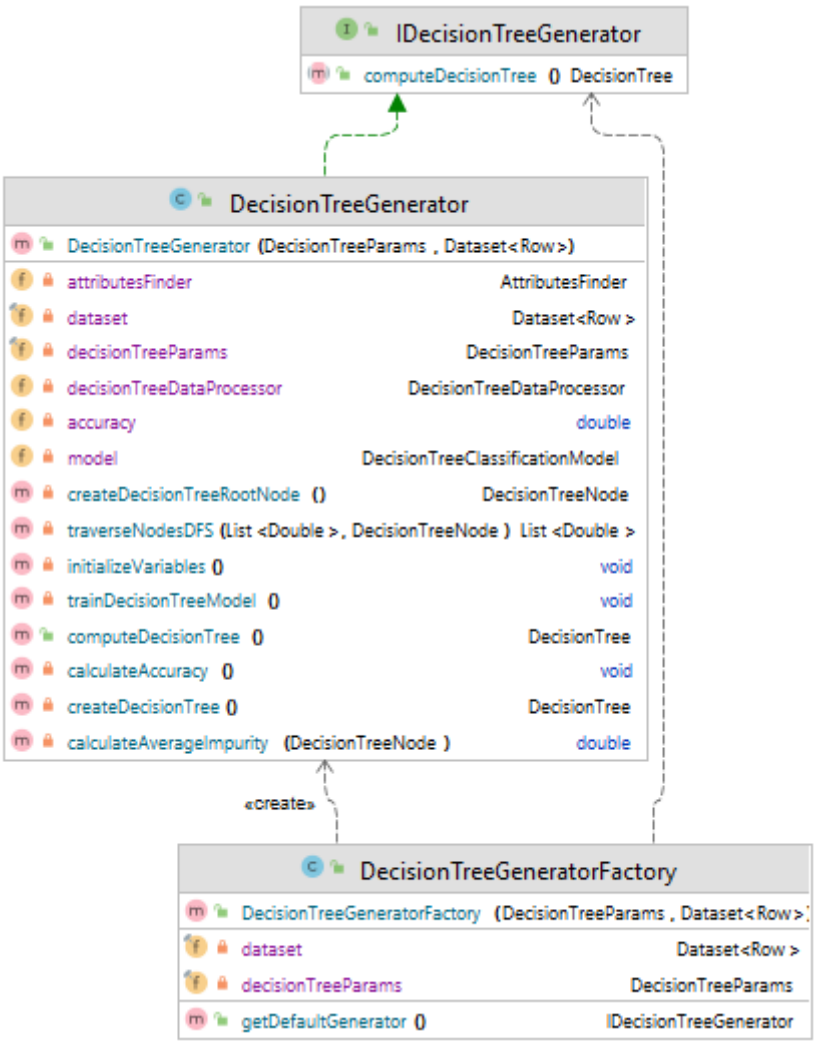
Εικόνα 25: Διάγραμμα UML του πακέτου decisiontree.datapreparation

Οι κλάσεις του πακέτου *decisiontree.datapreparation* είναι υπεύθυνες για την επεξεργασία των δεδομένων, είτε πριν τον υπολογισμό, είτε κατά την εύρεση των μονοπατιών του δέντρου.

Συγκεκριμένα, οι κλάσεις *AttributesFinder* και *DecisionTreeDataProcessor* προετοιμάζουν τα δεδομένα για τον υπολογισμό του δέντρου. Σκοπός της κλάσης *AttributesFinder* είναι να διαλέξει τα έγκυρα χαρακτηριστικά και ύστερα να διακρίνει τα συνεχείς και τα κατηγορικά. Έπειτα, η κλάση *DecisionTreeDataProcessor* λαμβάνει το σύνολο των δεδομένων και τα χαρακτηριστικά που συμμετέχουν στον υπολογισμό από τη κλάση *AttributesFinder* και προετοιμάζει τα δεδομένα. Αναλυτικότερα, μετατρέπει σε αριθμούς τα κατηγορικά δεδομένα, όπου κάθε ξεχωριστή τιμή αναθέτεται σε έναν αριθμό και έπειτα διαχωρίζει τα δεδομένα σε ένα σύνολο δεδομένων εκπαίδευσης και ένα δοκιμής.

Η κλάση *DecisionTreePathsFinder* χρησιμοποιείται για να εντοπισθούν τα μονοπάτια ενός δέντρου. Στο παρασκήνιο, το όριο διαχωρισμού σε κάθε κόμβο του δέντρου αναφέρεται πάντα ως προς «μικρότερο ή ίσο από μια τιμή» στα συνεχείς και «ανήκει στις τάδε κατηγορίες» στα κατηγορικά. Η κλάση *DecisionTreePathsFinder* εντοπίζοντας τα μονοπάτια, σημειώνοντας για ποιους κόμβους χρειάζεται αντιστροφή των συνθηκών διαχωρισμού, ώστε να γνωρίζουμε την κατεύθυνσή μας στον τάδε κόμβο.

3.2.17 Πακέτο decisiontree.generator



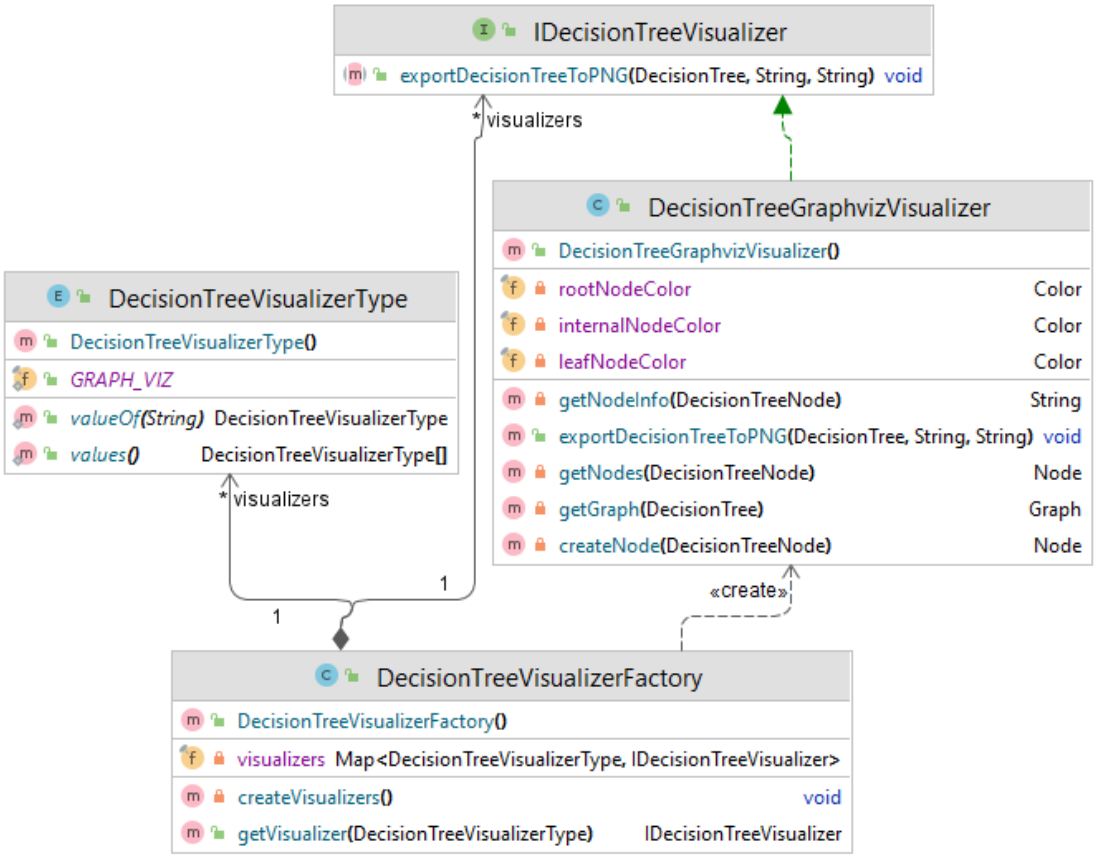
Εικόνα 26: Διάγραμμα UML του πακέτου decisiontree.generator

Ο σκοπός του πακέτου *decisiontree.generator* είναι η υλοποίηση του μηχανισμού δημιουργίας των δέντρων αποφάσεων. Ακολουθείται ο συνδυασμός *Factory Pattern* με διεπαφή (*Interface*), όπου η διεπαφή *IDecisionTreeGenerator* θέτει τους κανόνες για κάθε υλοποίηση, η κλάση *DecisionTreeGenerator* την υλοποιεί και η κλάση *DecisionTreeGeneratorFactory* χρησιμοποιείται για τη κατασκευή της μοναδικής αυτής

υλοποίησης. Με αυτόν τον τρόπο, μπορεί εύκολα να εισαχθούν διαφορετικές υλοποιήσεις στο μέλλον.

Το πραγματικό μας ενδιαφέρον βρίσκεται στη κλάση *DecisionTreeGenerator*, η οποία αξιοποιεί πολλές από τις κλάσεις των πακέτων *decisiontree* και *model.decisiontree* για να δημιουργήσει το τελικό δέντρο αποφάσεων. Πρώτα, υπολογίζεται το δέντρο αποφάσεων μέσω του Spark και των κλάσεων *AttributesFinder* και *DecisionTreeDataProcessor*. Έπειτα, συναρμολογούνται τα διάφορα κομμάτια της κλάσης *DecisionTree* από το πακέτο *model.decisiontree*, όπως τα μονοπάτια και οι κόμβοι, για να παραχθεί η τελική μορφή του δέντρου. Ολόκληρη η διαδικασία κατασκευής λαμβάνει μέρος σε αυτή τη κλάση.

3.2.18 Πακέτο decisiontree.visualization

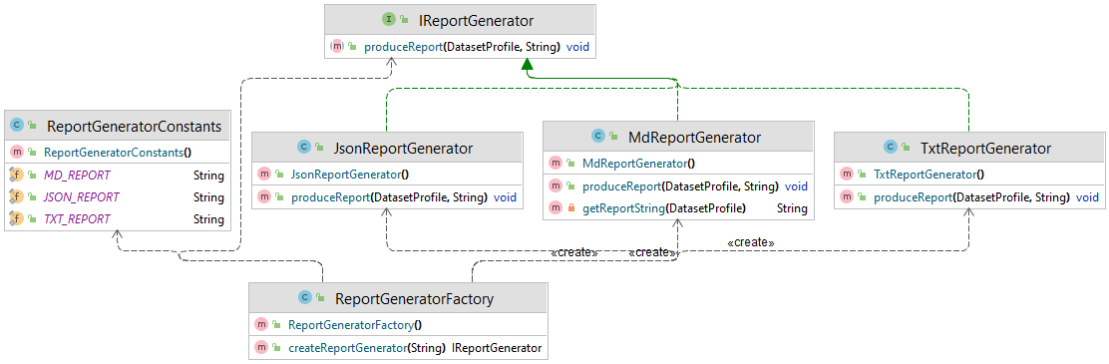


Εικόνα 27: Διάγραμμα UML του πακέτου decisiontree.visualization

Το πακέτο *decisiontree.visualization* είναι υπεύθυνο για τη δημιουργία της γραφικής αναπαράστασης ενός δέντρου αποφάσεων και την εξαγωγή της σε εικόνα τύπου "*.png*". Ομοίως με τα περισσότερα πακέτα του Pythia, χρησιμοποιείται η αρχιτεκτονική *Factory Pattern* σε συνδυασμό με διεπαφή. Η διεπαφή *IDecisionTreeVisualizer* επιτρέπει ένα είδος

απλούστευσης της λειτουργικότητας ως προς τον χρήστη ή το πακέτο που τη χρησιμοποιεί, προσφέροντας μία μοναδική μέθοδο εξαγωγής του δέντρου αποφάσεων ως εικόνα. Την υλοποίηση αναλαμβάνει η κλάση *DecisionTreeGraphvizVisualizer*, η οποία χρησιμοποιεί την εξωτερική βιβλιοθήκη *guru.nidi.graphviz* για να κατασκευάσει και αποθηκεύσει τη γραφική αναπαράσταση του δέντρου. Η βιβλιοθήκη αυτή χρησιμοποιεί μια μηχανή γλώσσας *Javascript* μέσω *Java*, ώστε να μπορέσει να αξιοποιήσει τα χαρακτηριστικά της γλώσσας *Graphviz*, η οποία δουλεύει σε *Javascript* και είναι ιδανική για τη κατασκευή δυσδιάστατων γραφικών. Η κλάση *DecisionTreeGraphvizVisualizer* αποτελεί τη μοναδική υλοποίηση μιας κλάσης δημιουργίας γραφικών αναπαραστάσεων για τα δέντρα αποφάσεων. Ωστόσο είναι εύκολη η αλλαγή της ή η προσθήκη μίας νέας υλοποίησης, μέσω της κλάσης που τη δημιουργεί *DecisionTreeVisualizerFactory*. Ο τύπος της υλοποίησης που θα δημιουργηθεί καθορίζεται από τις μεταβλητές της κλάσης *DecisionTreeVisualizerType*.

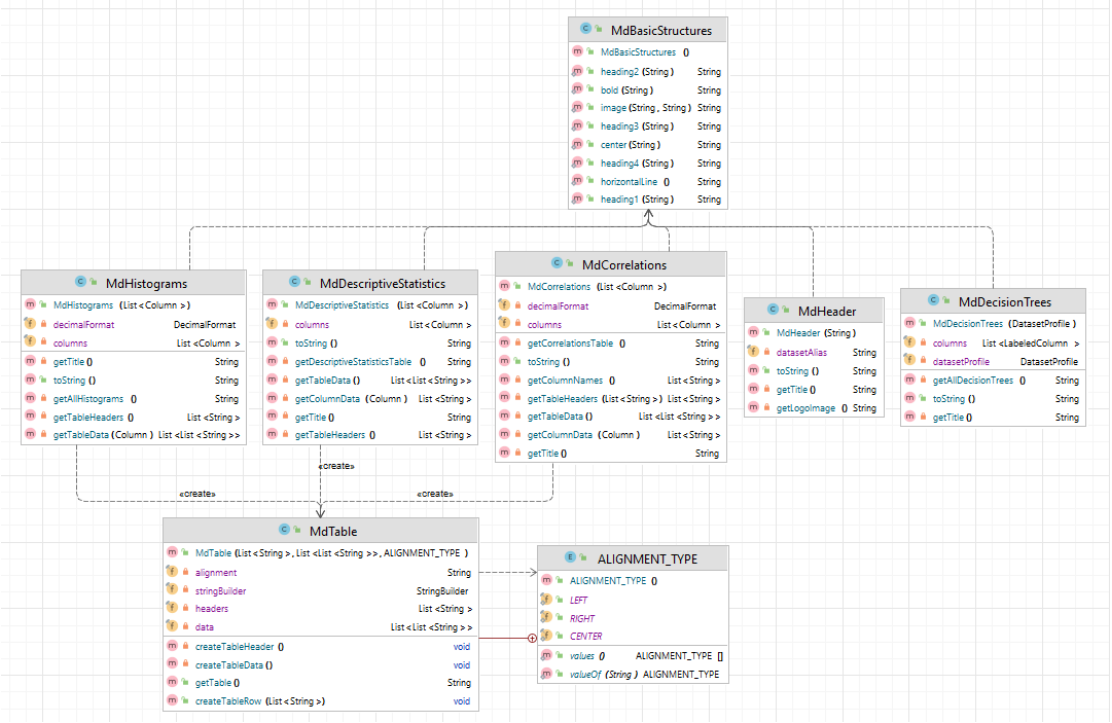
3.2.19 Πακέτο report



Εικόνα 28: Διάγραμμα UML του πακέτου report

Το πακέτο *report* είναι υπεύθυνο για τη δημιουργία της τελικής αναφοράς, συμπεριλαμβάνοντας όλα τα στατιστικά στοιχεία που έχουν υπολογιστεί. Δίνει τη δυνατότητα παραγωγής τριών διαφορετικών τύπων αρχείου *".json"*, *".txt"* και *".md"*, όπου η επιλογή τους γίνεται μέσω της κλάσης *ReportGeneratorConstants,* η οποία παρέχει όλες τις διαθέσιμες επιλογές. Μαζί με την κλάση *ReportGeneratorFactory*, μπορούμε να δημιουργήσουμε την υλοποιήση που θέλουμε, δηλαδή τις κλάσεις *JsonReportGenerator, TxtReportGenerator* και *MdReportGenerator* αντίστοιχα, για κάθε τύπο αρχείου. Όλες οι υλοποιήσεις ακολουθούνε τις συμβάσεις τις διεπαφής *IReportGenerator*, δηλαδή υλοποιούν τη μέθοδο *produceReport(…)*. Ακολουθείται, λοιπόν, το μοτίβο *Factory Pattern* με διεπαφή. Με αυτόν τον τρόπο, καθίσταται εύκολη η ανάπτυξη επιπλέον υλοποιήσεων.

3.2.20 Πακέτο report.md



Εικόνα 29: Διάγραμμα UML του πακέτου report.md

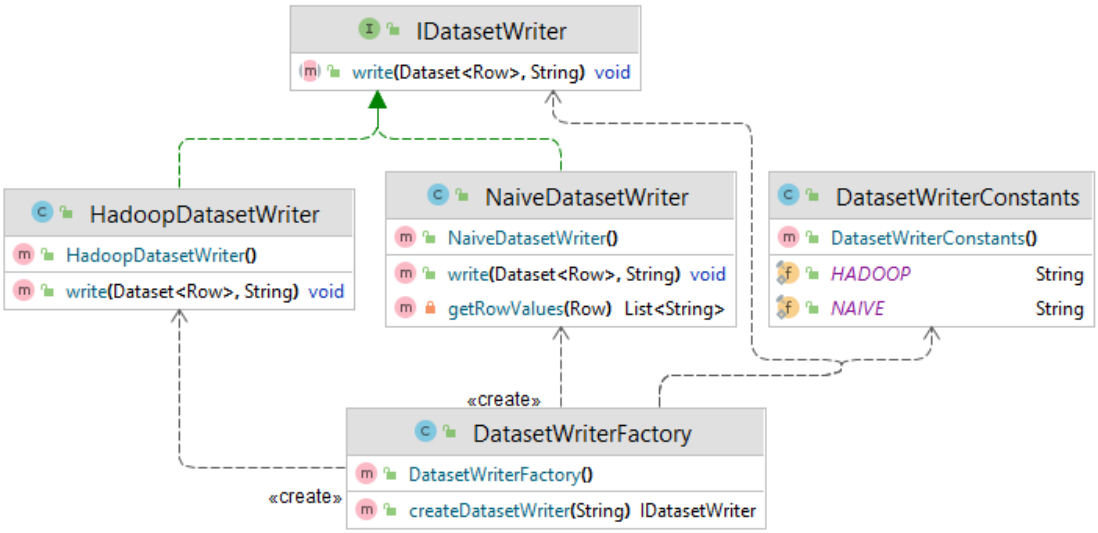
Το πακέτο report.md περιέχει δύο υποπακέτα, το report.md.structures και το report.md.components. Ωστόσο αξίζει να τα εξετάσουμε μαζί γιατί αλληλεξαρτούνται σε μεγάλο βαθμό.

Το πακέτο report.md.structures περιέχει τις κλάσεις που αναπαριστούν τις βασικές δομές της γλώσσας markdown, όπως πλάγια και έντονη γραφή, κεφαλίδες εικόνες και πίνακες. Όλα, εκτός από τους πίνακες, έχουν συμπιεστεί στη κλάση MdBasicStructures, ενώ οι πίνακες έχουν την δική τους υλοποίηση μέσω της κλάσης MdTable. Επίσης, υπάρχει μία βοηθητική μεταβλητή στους πίνακες, η ALIGNMENT_TYPE, με την οποία θέτουμε τη στοίχιση των δεδομένων στον πίνακα. Στόχος των συγκεκριμένων κλάσεων είναι να μετατρέψουν το δοθέντα κείμενο ή δεδομένα στην θεμιτή μορφή. Αυτές οι κλάσεις είναι τα «οικοδομικά υλικά», ή αλλιώς οι βασικές δομές markdown, που χρησιμοποιούνται από τις υπόλοιπες κλάσεις για την αναπαράσταση των στατιστικών στοιχείων. Στην Εικόνα 29 μπορούμε εύκολα να διακρίνουμε ποιες κλάσεις χρησιμοποιούν πίνακες και ποιες όχι.

Η ολική δομή της αναφοράς, ωστόσο, κατασκευάζεται βήμα προς βήμα, με τις κλάσεις του πακέτου report.md.components. Η τελική αναφορά markdown χωρίζεται σε ξεχωριστά κεφάλαια, όπου το κάθε κεφάλαιο περιγράφει ένα διαφορετικό στατιστικό στοιχείο. Κάθε κεφάλαιο παράγεται από μία διαφορετική κλάση, όπως τα περιγραφικά

στατιστικά από τη κλάση MdDescriptiveStatistics και οι συσχετίσεις από τη MdCorrelations. Εξαίρεση αποτελεί η κλάση MdHeader η οποία δεν περιέχει κάποιο στατιστικό στοιχείο, αλλά μερικά γενικά δεδομένα για την αναφορά, όπως τον τίτλο και το λογότυπο του Pythia.

3.2.21 Πακέτο writer



Εικόνα 30: Διάγραμμα UML του πακέτου writer

Τέλος, το πακέτο *writer* περιέχει τις κλάσεις απαραίτητες για την αποθήκευση ενός συνόλου δεδομένων στον δίσκο. Υποστηρίζει δύο τρόπους, έναν απλό και έναν γρήγορο, με τις κλάσεις που το υλοποιούν να είναι το *NaiveDatasetWriter* και το *HadoopDatasetWriter* αντίστοιχα. Ο πρώτος διαβάζει ολόκληρο το σύνολο δεδομένων γραμμή προς γραμμή και το αποθηκεύει σταδιακά στο καινούργιο αρχείο. Η διαδικασία αυτή είναι μη αποδοτική όταν υπάρχει μεγάλος όγκος δεδομένων, γιατί τα δεδομένα μπορεί να είναι διάσπαρτα σε διάφορους κόμβους του Spark. Σε αντίθεση, ο δεύτερος τρόπος χρησιμοποιεί το *Hadoop HDFS* σύστημα, το οποίο είναι πολύ γρήγορο στην εξαγωγή τεράστιων όγκων δεδομένων, ωστόσο απαιτεί τα εκτελέσιμα αρχεία του *Hadoop* στο σύστημα του υπολογιστή. Και οι δύο κλάσεις υλοποιούν τη μέθοδο που έχει οριστεί από τη διεπαφή *IDatasetWriter*, ενώ η κλάση *DatasetWriterFactory* τα παράγει. Όπως και στα υπόλοιπα πακέτα, υπάρχει μια βοηθητική κλάση ονόματι *DatasetWriterConstants,* με σταθερές που επιτρέπουν στον χρήστη να διαλέξει τον τύπο υλοποίησης που θέλει.

3.3 Σχεδίαση και αποτελέσματα ελέγχου του λογισμικού

Η ενότητα αυτή έχει ως θέμα την πραγματοποίηση ελέγχου στο σύστημα, τη μέθοδο με την οποία αυτός πραγματοποιήθηκε και μία αναλυτική παρουσίασή του.

3.3.1 Μεθοδολογία ελέγχου

Παράλληλα με την ανάπτυξη των λειτουργιών του εργαλείου, δημιουργούνταν και οι αντίστοιχοι έλεγχοι. Στόχος ήταν η διασταύρωση των παραγόμενων δεδομένων του στατιστικού προφίλ του Pythia, με αναμενόμενες εξόδους. Η μέθοδος αυτή είναι συχνά χρησιμοποιούμενη σε ελέγχους και ονομάζεται μαύρο κουτί. Υποδηλώνει πως δοθείσας μία συγκεκριμένης εισόδου στο πρόγραμμά μας, αναμένουμε μία συγκεκριμένη έξοδο, δίχως να εμπλακούμε στο πως τρέχει ο κώδικας ή να επέμβουμε σε αυτόν. Άμα το τελικό αποτέλεσμα διαφέρει, σημαίνει πως το εργαλείο λειτουργεί εσφαλμένα.

3.3.2 Αναλυτική παρουσίαση ελέγχου

Οι έλεγχοι που αναπτύξαμε εξετάζουν το σύστημα σε ατομικό επίπεδο λειτουργιών και κλάσεων, αλλά και ως σύνολο, χρησιμοποιώντας διάφορους εισόδους.

3.3.2.1 Έλεγχος δέντρων αποφάσεων

Κύριο και βασικότερο στοιχείο που εξετάζουμε σε ένα δέντρο αποφάσεων είναι τα δομικά του στοιχεία, δηλαδή οι κόμβοι, και η ακρίβειά του. Για να το επιτύχουμε αυτό, δώσαμε σαν είσοδο ένα συγκεκριμένο σύνολο δεδομένων, το "*carseats.csv*". Έπειτα, μέσω της κλάσης *"DecisionTreeParams"*, που χρησιμοποιείται για το πέρασμα παραμέτρων σε ένα δέντρο αποφάσεων, επιλέγουμε ορισμένες στήλες ως χαρακτηριστικά και βάθος δέντρου ίσο με δύο, ώστε να μην είναι πολύ μεγάλο. Τα δέντρα αποφάσεων που έχουμε υλοποιήσει, χρησιμοποιούν το τριάντα τις εκατό του συνόλου των δεδομένων ως δεδομένα εκπαίδευσης και το υπόλοιπο ως δεδομένα δοκιμής. Ωστόσο κάθε φορά διαλέγονται, τυχαία και διαφορετικά δεδομένα από όλο το σύνολο δεδομένων για την εκπαίδευσή του, με αποτέλεσμα το παραγόμενο δέντρο να είναι πάντα διαφορετικό. Για να κρατήσουμε τα αποτελέσματα πάντα ίδια, θέτουμε μέσω του *"DecisionTreeParams"* όλο το σύνολο των δεδομένων να λαμβάνει μέρος ως δεδομένα εκπαίδευσης. Επομένως, το τελικό δέντρο θα είναι πάντα το ίδιο, όπου και ελέγχουμε άμα τα στοιχεία του κάθε κόμβου και η συνολική ακρίβεια είναι τα ίδια με αυτά που περιμέναμε.

Επιπλέον, εξετάζουμε και την ακεραιότητα της κλάσης περάσματος παραμέτρων *"DecisionTreeParams"*. Ορίζοντας μόνο συγκεκριμένα χαρακτηριστικά για τον υπολογισμό ενός δέντρου αποφάσεων, εξετάζουμε άμα το τελικό δέντρο πράγματι υπολογίστηκε μόνο από αυτά. Οι είσοδοι των ελέγχων αυτών αποτελούνται από όλα ή μερικά χαρακτηριστικά, ή και λανθασμένες μεταβλητές όπως *null*.

Στη συνέχεια, εξετάζουμε άμα έχουν καταγραφεί τα σωστά μονοπάτια σε ένα σταθερό δέντρο αποφάσεων και άμα έχουν δημιουργηθεί οι εικόνες στις ανάλογες τοποθεσίες για κάθε δέντρο αποφάσεων.

Τέλος, ελέγχουμε άμα δουλεύει σωστά ο βελτιστοποιητής του δέντρου αποφάσεων, που ουσιαστικά αφαιρεί από τον υπολογισμό κατηγορικές κολόνες με περισσότερες από 32 διακριτές τιμές. Αυτό καταφέρεται περνώντας ως είσοδο το σύνολο δεδομένων "*tweets.csv*" και παρακολουθώντας ποιες στήλες έχουν μείνει μετά την επεξεργασία τους από τη "*DecisionTreeOptimizer"* κλάση.

3.3.2.2 Έλεγχος ιστογραμμάτων

Στα ιστογράμματα ελέγχουμε τα δεδομένα όλων των κάδων του κάθε ιστογράμματος, όπως το εύρος και το πλήθος των τιμών στο τάδε εύρους. Για την παραγωγή τους χρησιμοποιούμε το λογισμικό Pythia και υπολογίζουμε όλο το στατιστικό προφίλ με τη χρήση του συνόλου δεδομένων "*breasts-w.csv*".

3.3.2.3 Έλεγχος συσχετίσεων

Για να εξετάσουμε τα αποτελέσματα των συσχετίσεων των μεταβλητών, φορτώνουμε ένα μικρό σύνολο δεδομένων, το "*people.json*" και ελέγχουμε άμα οι αριθμητικές στήλες έχουν συσχετίσεις με άλλες αριθμητικές και μη, και άμα οι μη αριθμητικές δεν έχουν καθόλου.

3.3.2.4 Έλεγχος συστήματος αναφοράς

Οι έλεγχοι συστήματος αναφοράς ουσιαστικά ελέγχουν τη λειτουργία ολόκληρου του εργαλείου, εφόσον για την παραγωγή τους χρειάζεται ο υπολογισμός του στατιστικού προφίλ και η έκδοσή του σε μία αναφορά. Φορτώνουμε, λοιπόν, και τρέχουμε στο Pythia ένα μικρό σύνολο δεδομένων, το "*people.json"*, ώστε να μπορούμε να διακρίνουμε εύκολα με το μάτι το αποτέλεσμα και τυχόν σφάλματα. Έπειτα, οι παραγόμενες αναφορές συγκρίνονται με ήδη υπάρχουσες, μία για τον κάθε τύπο αρχείου.

3.3.2.5 Έλεγχος συστήματος αποθήκευσης συνόλου δεδομένων

Για να ελέγξουμε τη δυνατότητα αποθήκευσης ενός ήδη φορτωμένου ή και αλλοιωμένου συνόλου δεδομένων, φορτώνουμε ένα συγκεκριμένο σύνολο δεδομένων στην εφαρμογή, το καταγράφουμε και έπειτα φορτώνουμε το αποθηκευμένο αρχείο και ελέγχουμε άμα πράγματι τα δεδομένα είναι τα αναμενόμενα, δηλαδή υπάρχουν οι ίδιες εγγραφές και στήλες στο αποθηκευμένο σύνολο δεδομένων.

3.3.2.6 Λοιποί έλεγχοι

Τέλος, δεν αναπτύχθηκαν περαιτέρω, αλλά υπάρχουν από πριν, μερικοί απλοί έλεγχοι για τον έλεγχο της ακεραιότητας του συστήματος δημιουργίας labeled κολόνας.

3.4 Λεπτομέρειες εγκατάστασης και υλοποίησης

Το Pythia είναι ένα εργαλείο που οποιοσδήποτε μπορεί να έχει πρόσβαση στο αποθετήριό του στο GitHub. Παρακάτω, σε αυτή την ενότητα, θα αναλύσουμε τα χαρακτηριστικά της υλοποίησης του, όπως η πλατφόρμα ανάπτυξης και εκτέλεσης, τα προγραμματιστικά εργαλεία, οι απαιτήσεις της εφαρμογής σε hardware και άλλα.

3.4.1 Χαρακτηριστικά Υλοποίησης

3.4.1.1 Γλώσσα προγραμματισμού

Το Pythia γράφτηκε στη προγραμματιστική γλώσσα Java και εκμεταλλεύεται τα χαρακτηριστικά που τη περιβάλλουν (Κεφάλαιο 2.2.1.4). Επομένως, το τρέχον λογισμικό μπορεί να εκτελεστεί σε οποιοδήποτε υπολογιστικό σύστημα που υποστηρίζει την εγκατάσταση του JVM (Κεφάλαιο 2.2.1.4), όπως Windows, Linux, MacOS και Android. Επίσης, η αντικειμενοστραφής φύση της Java μας επέτρεψε να αναπτύξουμε το Pythia με δομές, κλάσεις και κώδικα που είναι επαναχρησιμοποιήσιμος και εύκολος στο να επεκταθεί και συντηρηθεί.

3.4.1.2 Διαχείριση χαρακτηριστικών υλοποίησης

Για τη διαχείριση των χαρακτηριστικών υλοποίησης, όπως τη γλώσσα προγραμματισμού και την έκδοσή της, τα εξωτερικά πακέτα (βιβλιοθήκες), τη διαδικασία κατασκευής (build) του συστήματος και την εξαγωγή του σε ένα αρχείο JAR (που περιέχει όλον τον κώδικα με τις εξαρτήσεις του), χρησιμοποιήθηκε το εξωτερικό εργαλείο Maven (Κεφάλαιο 2.2.1.5). Βρίσκεται ένα αναλυτικό αρχείο pom.xml στον ριζικό φάκελο του προγράμματος, ο οποίος περιέχει όλες τις πληροφορίες της υλοποίησης.

3.4.1.3 Διαχείριση συνόλων δεδομένων και στατιστικών

Όσον αφορά την διαχείριση και την επεξεργασία μεγάλων συνόλων δεδομένων, καθώς και τον υπολογισμό διάφορων μετρικών και στατιστικών μοντέλων, χρησιμοποιήθηκε η βιβλιοθήκη Apache Spark (Κεφάλαιο 2.2.1.3). Το Apache Spark είναι υπεύθυνο για το διάβασμα των μεγάλων συνόλων δεδομένων και την αναπαράσταση τους ως δεδομένα που μπορούνε να χρησιμοποιηθούνε για την άντληση πληροφοριών. Απλές μετρικές όπως τα περιγραφικά στατιστικά υπολογίζονται μέσω έτοιμων εντολών του Spark, ενώ πιο περίπλοκα στατιστικά μοντέλα όπως τα δέντρα αποφάσεων, υπολογίζονται και αναπαρίστανται με τις αντίστοιχες παρεχόμενες κλάσεις. Στο πρόγραμμά μας το Apache Spark είναι η κύρια μηχανή κοστοβόρων υπολογισμών και για αυτό έχει ρυθμιστεί κατάλληλα ώστε να χρησιμοποιεί όλους τους πόρους του επεξεργαστή του συστήματος. Ο προγραμματιστής, όμως, έχει την ευχέρεια να το ρυθμίσει εύκολα μέσω του αρχείου ρυθμίσεων "*spark.properties*".

3.4.1.4 Περιβάλλον ανάπτυξης

Το πρόγραμμα αναπτύχθηκε στο λογισμικό σύστημα Windows 10. Η συγγραφή του πηγαίου κώδικα έγινε στο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) IntelliJ IDEA, το οποίο προσφέρει πολύ καλή υποστήριξη στην ανάπτυξη εφαρμογών Java κ.α. Ωστόσο, η ανάπτυξη του Pythia μπορεί να συνεχιστεί και σε άλλα περιβάλλοντα ανάπτυξης, όπως το Eclipse.

3.4.2 Διαδικασία Εγκατάστασης

Για την επιτυχής έναρξη και λειτουργία του προγράμματος Pythia σε έναν άλλον υπολογιστή, απαιτείται:

- Να εγκαταστήσουμε τη Java 1.8 στο λογισμικό σύστημα του υπολογιστή. Έπειτα χρειάζεται να θέσουμε τον φάκελο εγκατάστασης ως μεταβλητή περιβάλλοντος με όνομα "JAVA_HOME" και μετά να προσθέσουμε στο "PATH" το "%JAVA_HOME%\bin". Στο σύστημα που αναπτύχθηκε η εφαρμογή, το "JAVA_HOME" ισούται με τον φάκελο *"C:\Program Files\Java\jdk1.8.0_202".*
- Να εγκαταστήσουμε το IDE της επιλογής μας, όπως το IntelliJ IDEA ή Eclipse.
- Να εγκαταστήσουμε την έκδοση 3.2.2 του Hadoop στον υπολογιστή μας, δηλαδή να εξάγουμε το αρχείο "*hadoop-3.2.2.tar.gz"* σε έναν προορισμό της επιλογής μας. Έπειτα όπως και για τη Java, θα θέσουμε τη μεταβλητή περιβάλλοντος "HADOOP_HOME" που θα δείχνει στο αρχείο που μόλις εξάγαμε και θα προσθέσουμε στο "PATH" το "%HADOOP_HOME%\bin". Στο σύστημα που

αναπτύχθηκε η εφαρμογή, το "HADOOP_HOME" ισούται με τον φάκελο *"C:\Hadoop\hadoop-3.2.2"*.

- Προαιρετικά: Σε συστήματα Windows χρειάζεται το WinUtils, το οποίο είναι δυαδικά αρχεία για να τρέξει το Hadoop σε Windows. Κατεβάζετε την έκδοση 3.2.2 του WinUtils και τοποθετείτε τα αρχεία του στον φάκελο "*bin"* του φακέλου εγκατάστασης του Hadoop, αντικαθιστώντας τα αρχεία που έχουν το ίδιο όνομα. Στο σύστημα που αναπτύχθηκε η εφαρμογή, είναι ο φάκελος *"C:\Hadoop\hadoop-3.2.2\bin"*.

Αξίζει να σημειωθεί πως δεν υπάρχει ανάγκη εγκατάστασης του Maven, εφόσον ήδη έχει συμπεριληφθεί ένα Maven Wrapper, το οποίο είναι μια ενσωματωμένη εγκατάσταση του Maven. Για να λειτουργήσει όμως, πρέπει να έχει δημιουργηθεί η μεταβλητή περιβάλλοντος "JAVA_HOME", που αναφέρθηκε παραπάνω.

Ωστόσο, άμα κάποιος επιθυμεί να εγκαταστήσει το Maven στον υπολογιστή του, αρκεί να κατεβάσει το αρχείο "*Binary zip archive*" από την επίσημα σελίδα του Maven και να το εγκαταστήσει. Πρέπει να το εξάγει, και όπως και στο Hadoop να θέσει τη μεταβλητή περιβάλλοντος "MAVEN_HOME" και "%MAVEN_HOME%\bin" στο "PATH". Στο σύστημα που αναπτύχθηκε η εφαρμογή, το "MAVEN_HOME" ισούται με τον φάκελο *"C:\Program Files\Maven\apache-maven-3.8.7"*.

3.4.3 Διαδικασία κατασκευής και ελέγχου

Η διαδικασία της κατασκευής περιλαμβάνει την εγκατάσταση των απαραίτητων εξαρτήσεων, τη μεταγλώττιση του πηγαίου κώδικα και έλεγχο της αναμενόμενης λειτουργίας των αντίστοιχων κλάσεων.

Για να εκκινήσουμε τη διαδικασία κατασκευής χρειάζεται μέσω της τερματικής κονσόλας να μεταβούμε στον ριζικό φάκελο του Pythia και για Windows να εκτελέσουμε την εντολή *"mvnw.cmd clean install"*, ενώ για συστήματα Unix την εντολή *"./mvnw clean install"*.

Ο έλεγχος γίνεται κατά τη διάρκεια της κατασκευής και τυχόν αποτυχίες ή σφάλματα θα εμφανιστούν στο παράθυρο της τερματικής κονσόλας.

Επίσης, ο έλεγχος μπορεί και να γίνει μέσω των εντολών *"mvnw.cmd test"* και *"./mvnw test"*, για συστήματα Windows και Unix αντίστοιχα.

Το αποτέλεσμα της διαδικασίας κατασκευής είναι η παραγωγή δύο αρχείων JAR, ονόματι *Pythia-x.y.z-all-deps.jar* και *Pythia-x.y.z.jar*. Το πρώτο αρχείο περιέχει μεταγλωττισμένες όλες τις εξωτερικές βιβλιοθήκες που απαιτούνται από το Pythia, ενώ το δεύτερο καμία.

Αυτό σημαίνει πως για να χρησιμοποιηθεί το δεύτερο αρχείο σε ένα άλλο πρόγραμμα, απαιτείται η μέριμνα της εγκατάστασης των βιβλιοθηκών αυτών. Καθένα από τα δύο αρχεία όμως, περιέχουν τον πηγαίο κώδικα του Pythia.

3.5 Επεκτασιμότητα του λογισμικού

Κάθε πρόγραμμα έχει την ανάγκη να συντηρηθεί και να επεκταθεί με τα χρόνια, για να προσφέρει βελτιωμένες και καινούργιες λειτουργικότητες. Το Pythia έχει εφαρμόσει τις καλύτερες προγραμματιστικές τεχνικές, ώστε οι λειτουργίες του να είναι ευέλικτες και φιλόξενες σε αλλαγές.

Κάθε πακέτο της είναι υπεύθυνο για την υλοποίηση μιας συγκεκριμένης λειτουργίας, και κάθε διαφορετική υλοποίηση της τάδε λειτουργίας ακολουθεί μια κοινή λίστα συμβάσεων, μέσω της υλοποίησης μιας διεπαφής (*Interface)*. Επομένως, για την επέκταση μίας λειτουργίας, αρκεί να δημιουργηθεί μία καινούργια κλάση η οποία συμβαδίζει με τους κανόνες που θέτει η διεπαφή. Η δημιουργία των διαφορετικών υλοποιήσεων της τάδε λειτουργίας, γίνεται μέσω της αντίστοιχης κλάσης παραγωγής αντικειμένων (*Factory*) και αφαιρεί τα καθήκοντα αρχικοποίησης από άλλα πακέτα, ενώ παράλληλα επιστρέφει μόνο αφηρημένες κλάσεις με τις δυνατότητες της διεπαφής. Αυτό σημαίνει πως για να ενταχθεί μια καινούργια υλοποίηση, αρκεί να προστεθεί μια μέθοδος αρχικοποίησής της μέσα στη κλάση παραγωγής, δίχως να χρειαστεί να αλλάξει κάτι σε εξωτερικά πακέτα που τη χρησιμοποιούν.

Παραδείγματος χάριν, το πακέτο "*report*" προσφέρει διαφορετικά είδη αναφορών, όπως αρχεία txt, json και markdown, με τις αντίστοιχες κλάσεις να είναι τα *TxtReportGenerator*, *JsonReportGenerator* και *MdReportGenerator*. Όλες αυτές οι κλάσεις υλοποιούν τη διεπαφή *IReportGenerator*, δηλαδή τις μεθόδους της με τον δικό τους τρόπο. Έπειτα, αυτά μπορούμε να τα χρησιμοποιήσουμε σε ένα άλλο πακέτο φτιάχνοντας ένα αντικείμενο *IReportGeneratorFactory* που τα κατασκευάζει εσωτερικά και με την σωστή είσοδο μπορούμε να πάρουμε την ανάλογη υλοποίηση. Άμα θελήσουμε να εισάγουμε μια καινούργια υλοποίηση, δημιουργούμε τη κλάση, ακολουθούμε τη διεπαφή και τη προσθέτουμε σαν επιλογή στην κλάση παραγωγής μας.

Επομένως, τα πακέτα που ακολουθούν αυτό το μοτίβο προτύπων (*Factory pattern με Interface*) είναι ιδιαίτερα ανεκτικά σε επεκτάσεις και αλλαγές των υπάρχων υλοποιήσεων.

Εξαίρεση αποτελούν τα πακέτα *config*, *labeling* και *util*, τα οποία έχουνε πολύ συγκεκριμένες αρμοδιότητες. Το πρώτο χρησιμοποιείται για την αναγνώριση των

παραμέτρων λειτουργίας της Spark, ενώ το δεύτερο για τη δημιουργία καινούργιων κολόνων στο σύνολο δεδομένων, βάση δοσμένων κανόνων και χρησιμοποιώντας το Spark. Το τρίτο εμπεριέχει μία κλάση η οποία είναι υπεύθυνη για να φιλτράρει τις κολόνες, ανάλογα τον τύπο (αριθμητικές ή κατηγορικές).

Κεφάλαιο 4. Πειραματική Αξιολόγηση

4.1 Μεθοδολογία πειραματισμού

Είναι δύσκολο να καθοριστεί το σκέλος της μακροχρόνιας επιρροής που φέρνουν οι αλλαγές οι οποίες επισυνάπτονται τη συντήρηση και την επέκταση ενός προγράμματος. Στο Pythia εξετάζουμε μόνο μία πτυχή των πολλών πειραμάτων που μπορούν να λάβουν μέρος, χρονομετρώντας τις λειτουργίες που προσθέσαμε, σε σχέση με τον συνολικό χρόνο εκτέλεσης μιας ολοκληρωμένης ακολουθίας λειτουργιών και τον χρόνο εκτέλεσης κάθε λειτουργίας ξεχωριστά. Επιπλέον, εξετάζουμε τον χρόνο κατασκευής και υπολογισμού ενός δέντρου αποφάσεων, για διάφορες τιμές των παραμέτρων του, όπως τον αριθμό των εγγραφών, τον αριθμό των συμπεριλαμβανομένων χαρακτηριστικών, τον αριθμό των συνολικών στηλών και τον τύπο των δεδομένων.

Συγκεκριμένα, για την εξέταση του συνολικού χρόνου παραγωγής μιας αναλυτικής αναφοράς, αλλά και των επιμέρους λειτουργιών του λογισμικού, χρησιμοποιήθηκαν δύο σύνολα δεδομένων, ένα με δεδομένα ασθενών “Covid-19” και ένα με χρήστες της επιχείρησης “Yelp”.

Το πρώτο σύνολο δεδομένων ονομάζεται “COVID-19 Case Surveillance Public Use Data with Geography”, έχει δημιουργεί από το CDC (Centers for Disease Control and Prevention, ή αλλιώς Κέντρα Ελέγχου και Πρόληψης Ασθενειών), το οποίο είναι ένας εθνικός οργανισμός των Ηνωμένων Πολιτειών και παρέχει δεδομένα σχετικά με άτομα που έχουν ασθενήσει με τον ιό “Covid-19”. Περιέχει 19 μεταβλητές, κυρίως τύπου κειμένου, και 95 εκατομμύρια εγγραφές, οπότε αποτελεί ιδανικό σύνολο δεδομένων για να ελέγξουμε πως ανταπεξήλθε το Pythia σε κατηγορικά δεδομένα. Μπορεί να βρεθεί εδώ: https://data.cdc.gov/Case-Surveillance/COVID-19-Case-Surveillance-Public-Use-Data-with-Ge/n8mc-b4w4.

Το δεύτερο σύνολο δεδομένων ονομάζεται “user.json” και αποτελεί ένα υποσύνολο των δεδομένων της επιχείρησης Yelp, που προορίζεται για εκπαιδευτικούς και ακαδημαϊκούς σκοπούς. Αναφέρεται στους πελάτες του Yelp, όπου κάθε εγγραφή αντιπροσωπεύει έναν πελάτη της εταιρείας και τα πεδία του είναι κυρίως αριθμητικά. Έχει περίπου δύο εκατομμύρια διαφορετικές εγγραφές και το χρησιμοποιούμε για να ελέγξουμε την απόδοση του συστήματος σε κυρίως αριθμητικά δεδομένα. Το Yelp είναι μια αμερικανική

εταιρεία, που παρέχει υπηρεσίες κράτησης θέσεων και αξιολόγησης των επιχειρήσεων από τους πελάτες της. Το σύνολο δεδομένων μπορεί να βρεθεί στην ιστοσελίδα της: https://www.yelp.com/dataset.

Στο κάθε σύνολο δεδομένων, διαλέγουμε υποσύνολα των 500.000, 1.000.000, 1.500.000 και 2.000.000 εγγραφών και καταγράφουμε τον συνολικό χρόνο υπολογισμού και επεξεργασίας των δεδομένων, για κάθε λειτουργία του Pythia. Κατά τη διάρκεια της διαδικασίας αυτής, μελετάμε παράλληλα και τον χρόνο της δημιουργίας των δέντρων αποφάσεων και των ιστογραμμάτων, δηλαδή των λειτουργιών που προσθέσαμε. Για να συγκρίνουμε σωστά τα αποτελέσματά μας, χρησιμοποιήσαμε τον ίδιο αριθμό μεταβλητών (στηλών) από το κάθε σύνολο δεδομένων.

Για τη χρονομέτρηση της επιρροής των υπόλοιπων παραμέτρων των δέντρων αποφάσεων, χρησιμοποιήσαμε ως σύνολο δεδομένων το "NMDC_with_indicators.csv", το οποίο αποτελείται από δεδομένα μετοχών και δείκτες που διαπραγματεύονται στο χρηματιστήριο της Ινδίας. Καθίσταται ιδανικό για τα πειράματα διαφορετικού αριθμού χαρακτηριστικών ή στηλών, εφόσον έχει 58 μεταβλητές, κυρίως αριθμητικού τύπου. Βρίσκεται στην ιστοσελίδα: https://www.kaggle.com/datasets/debashis74017/stock-market-data-nifty-100-stocks-5-min-data.

Κάθε σύνολο δεδομένων έχει τον εξής αριθμό αριθμητικών και κατηγορικών στηλών:

ΣΥΝΟΛΟ ΔΕΔΟΜΕΝΩΝ	**ΑΡΙΘΜΟΣ ΑΡΙΘΜΗΤΙΚΩΝ ΣΤΗΛΩΝ**	**ΑΡΙΘΜΟΣ ΚΑΤΗΓΟΡΙΚΩΝ ΣΤΗΛΩΝ**
Covid	2	17
Yelp	17	4
NMDC	57	0

Πίνακας 5: Αριθμός αριθμητικός και κατηγορικών δεδομένων για κάθε σύνολο δεδομένων.

Για τις παραμέτρους του κάθε πειράματος χρησιμοποιήσαμε συνδυασμούς των παρακάτω τιμών:

ΑΡΙΘΜΟΣ ΕΓΓΡΑΦΩΝ	ΑΡΙΘΜΟΣ ΣΤΗΛΩΝ	ΑΡΙΘΜΟΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ
500.000	15	15
1.000.000	30	30
1.500.000	45	45
2.000.000		

Πίνακας 6: Οι τιμές που είχε η κάθε παράμετρος κατά τα πειράματα.

Η καταγραφή των μετρήσεων έλαβε μέρος σε υπολογιστή με τα ακόλουθα χαρακτηριστικά υλικού και λογισμικού:

OS	**Windows 10 Pro**
CPU	Intel(R) Core(TM) i5-4670 CPU @3.40GHz, 4 Cores
RAM	8GB DDR3
DISK	Samsung 840 Pro 256 GB, έως 540MB/s sequential Read, έως 520 MB/s sequential Write

Πίνακας 7: Χαρακτηριστικά του υπολογιστικού συστήματος που έτρεξαν τα πειράματα.

4.2 Αναλυτική παρουσίαση αποτελεσμάτων

4.2.1 Πειράματα δέντρων αποφάσεων

Αρχικά, μελετάμε το χρόνο υπολογισμού ενός δέντρου αποφάσεων, κάτω από διαφορετικές συνθήκες. Οι βασικοί παράμετροι που εξετάζουμε είναι ο αριθμός των εγγραφών, ο αριθμός των συνολικών στηλών και ο αριθμός των χαρακτηριστικών, δηλαδή τα δεδομένα από τα οποία δημιουργείται ένα δέντρο αποφάσεων. Σε κάθε πείραμα αλλάζουμε μόνο μία από τις παραμέτρους, ενώ κρατάμε τις υπόλοιπες σταθερές. Με αυτόν τον τρόπο, συγκρίνουμε τι επίπτωση έχει η καθεμία παράμετρος στον συνολικό χρόνο κατασκευής ενός δέντρου αποφάσεων.

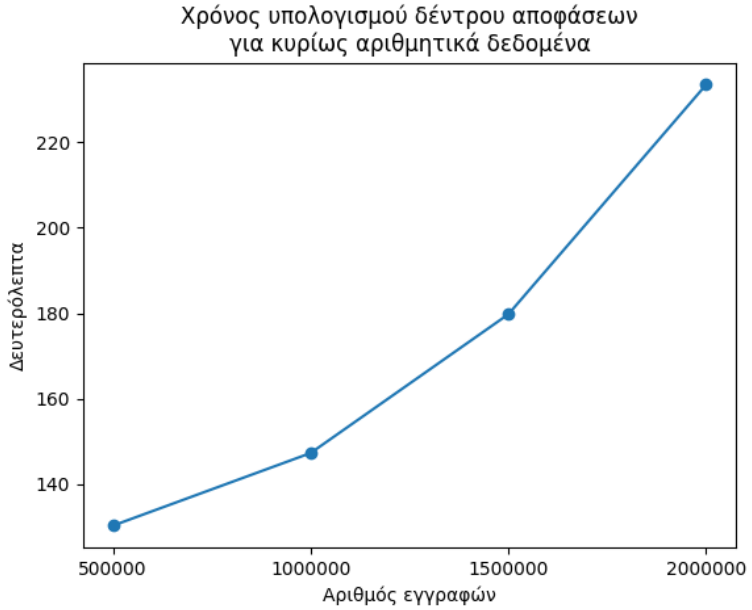
4.2.1.1 Διαφορετικός αριθμός εγγραφών

Πρωτίστως, θα εξετάσουμε τη διάρκεια δημιουργίας του δέντρου αποφάσεων για διαφορετικές ποσότητες εγγραφών. Εφόσον τα δέντρα αποφάσεων απευθύνονται και σε αριθμητικά και σε κατηγορικά δεδομένα, το πρώτο πείραμα λαμβάνει μέρος δύο φορές, με τις ίδιες παραμετρικές τιμές, μία για κυρίως αριθμητικές μεταβλητές και μία

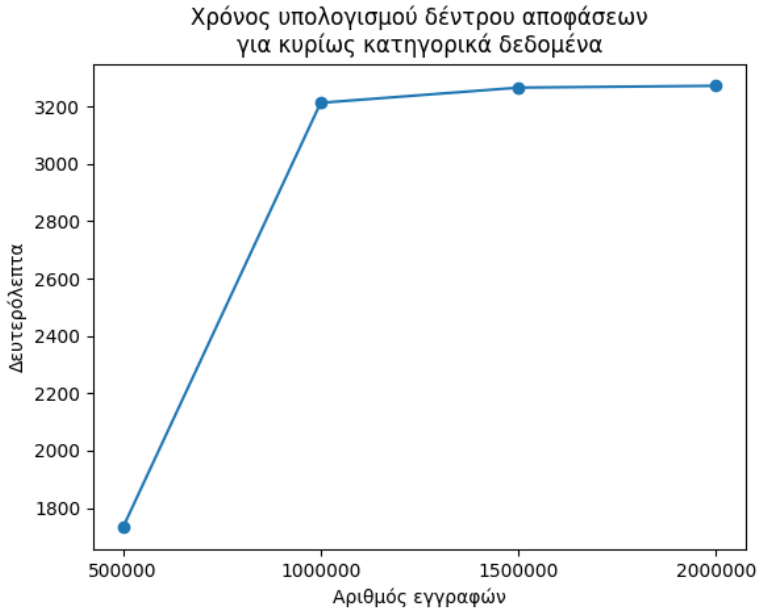
για κυρίως κατηγορικές μεταβλητές. Χρησιμοποιούνται τα σύνολα δεδομένων "*user.json*" και "*covid-19*" αντίστοιχα. Ο συνολικός αριθμός στηλών και χαρακτηριστικών περιορίζονται στα δεκαοχτώ, για να είναι ισομεγέθη τα υποσύνολα κάθε συνόλου δεδομένων. Ακολουθούν οι μετρήσεις και οι γραφικές αναπαραστάσεις των μετρήσεων αυτών:

ΑΡΙΘΜΟΣ ΕΓΓΡΑΦΩΝ	**ΚΥΡΙΩΣ ΑΡΙΘΜΗΤΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ (YELP DATA SET)**	**ΚΥΡΙΩΣ ΚΑΤΗΓΟΡΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ (COVID DATA SET)**
500.000	130.25	1731.67
1.000.000	147.25	3213.33
1.500.000	179.75	3266.33
2.000.000	233.50	3272.67

Πίνακας 8: Χρόνος υπολογισμού δέντρου αποφάσεων (σε δευτερόλεπτα) για αριθμητικά και κατηγορικά δεδομένα



Εικόνα 31: Χρόνος υπολογισμού δέντρου αποφάσεων (σε δευτερόλεπτα) για κυρίως αριθμητικά δεδομένα στο σύνολο δεδομένων Yelp.



Εικόνα 32: Χρόνος υπολογισμού δέντρου αποφάσεων (σε δευτερόλεπτα) για κυρίως κατηγορικά δεδομένα στο σύνολο δεδομένων Covid.

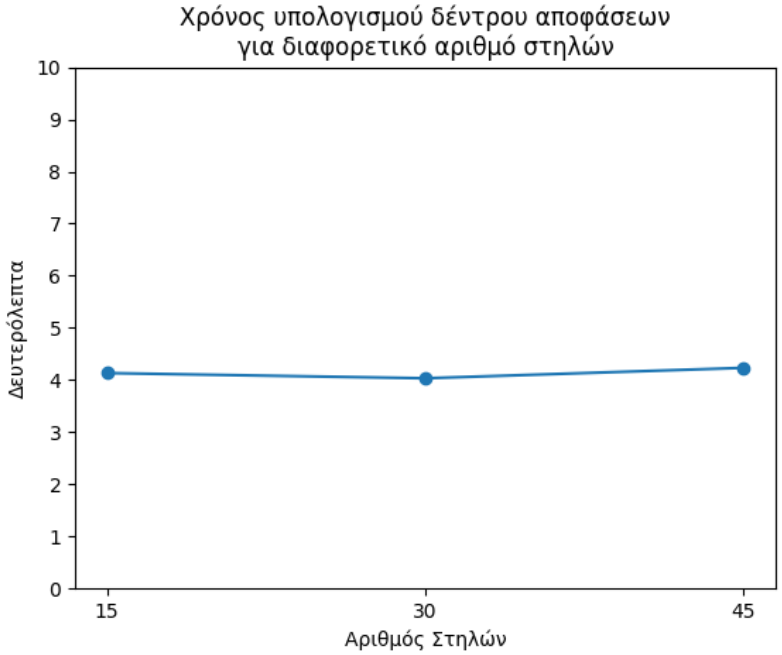
Στον *Πίνακα 8* παρατηρούμε τους καταγεγραμμένους χρόνους. Με μια πρώτη ματιά, βλέπουμε σημαντική διαφορά στον χρόνο υπολογισμού ενός δέντρου αποφάσεων, βάση τον τύπο των δεδομένων που εισήχθησαν. Μία εξήγηση του φαινομένου αυτού, είναι η επιπλέον διεργασίες που γίνονται για τις κατηγορικές μεταβλητές στο υπόβαθρο του συστήματος από το Spark. Για να επεξεργαστούμε πρώτα τα κατηγορικά δεδομένα, χρειάζεται να δημιουργήσουμε μία νέα στήλη για το καθένα, όπου κάθε διαφορετική τιμή ενός πεδίου αναπαρίσταται ως ένας συγκεκριμένος αριθμός *(string indexing)*. Αυτή η διαδικασία, καθώς και ο υπολογισμός της ταξινόμησης των κατηγορικών δεδομένων, είναι κοστοβόρα και καθυστερεί πολύ τον συνολικό υπολογισμό του δέντρου αποφάσεων. Ένας ακόμα λόγος μπορεί να είναι η μικρή μνήμη του συστήματος που διεξάχθηκαν τα πειράματα. Επίσης, παρατηρούμε μία σταθερή αύξηση του υπολογιστικού χρόνου στα κυρίως αριθμητικά δεδομένα, ανάλογη της αύξησης των δεδομένων. Αντίθετα, για τα κυρίως κατηγορικά δεδομένα ισχύει η ίδια ανάλογη αύξηση για τις 500.000 και τις 1.000.000 εγγραφές, αλλά παρατηρείται ελάχιστη (συγκριτικά) διαφορά μεταξύ των 1.000.000, 1.500.000 και 2.000.000 εγγραφών. Αυτό γίνεται πιο εμφανής στις *Εικόνα 31* και *Εικόνα 32*, όπου βλέπουμε για τα αριθμητικά δεδομένα μία γραμμική περίπου αύξηση, ενώ για τα κατηγορικά μία λογαριθμική.

4.2.1.2 Διαφορετικός αριθμός στηλών

Στη συνέχεια, θα αναλύσουμε τις επιπτώσεις του διαφορετικού αριθμού στηλών, στον συνολικό χρόνο υπολογισμού ενός δέντρου αποφάσεων. Για το συγκεκριμένο πείραμα, χρησιμοποιούμε το σύνολο δεδομένων "NMDC_with_indicators.csv" και κρατάμε σταθερό τον αριθμό των εγγραφών και τον αριθμό των χαρακτηριστικών. Στον *Πίνακα 9* παραθέτουμε τις τιμές των μεταβαλλόμενων παραμέτρων και τον χρόνο υπολογισμού του δέντρου αποφάσεων:

ΑΡΙΘΜΟΣ ΣΤΗΛΩΝ	ΧΡΟΝΟΣ ΥΠΟΛΟΓΙΣΜΟΥ
15	4.13
30	4.03
45	4.23

Πίνακας 9: Χρόνος υπολογισμού δέντρου αποφάσεων (σε δευτερόλεπτα) για διαφορετικό αριθμό στηλών στο σύνολο δεδομένων NMDC.



Εικόνα 33: Χρόνος υπολογισμού δέντρου αποφάσεων (σε δευτερόλεπτα) για διαφορετικό αριθμό στηλών στο σύνολο δεδομένων NMDC.

Αναλύοντας τα αποτελέσματα του πειράματος και μέσω της *Εικόνας 33*, παρατηρούμε πως ο χρόνος δημιουργίας ενός δέντρου αποφάσεων μένει σταθερός για διαφορετικό αριθμό στηλών. Το γεγονός αυτό αποτελεί ένα φυσιολογικό αποτέλεσμα, με τον κύριο υπαίτιο να είναι ο αριθμός των χαρακτηριστικών, ο οποίος παραμένει ίδιος σε όλα τα πειράματα. Στη πράξη, το δέντρο υπολογίζεται μόνο από τα χαρακτηριστικά, ή αλλιώς

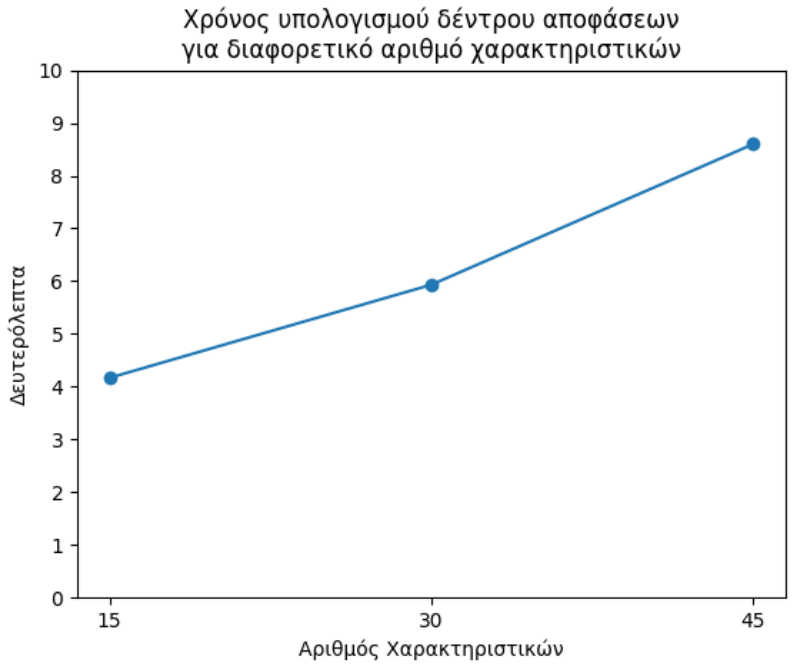
μεταβλητές, που του έχουμε ορίσει, οπότε η ένταξη περισσότερων στηλών δε συμβάλει στην αύξηση του χρόνου υπολογισμού του δέντρου αποφάσεων, εφόσον οι επιπλέον στήλες δεν εντάσσονται στη διαδικασία υπολογισμού. Η μικρή μεταβολή των τιμών, οφείλεται στο Spark και τις διαφορετικές διεργασίες του υπολογιστή εκείνη τη στιγμή.

4.2.1.3 Διαφορετικός αριθμός χαρακτηριστικών

Για να εξετάσουμε την επιρροή του αριθμού των χαρακτηριστικών που λαμβάνουν μέρος στη δημιουργία ενός δέντρου αποφάσεων, χρησιμοποιούμε το ίδιο σύνολο δεδομένων "NMDC_with_indicators.csv" και κρατώντας σταθερό τον αριθμό των εγγραφών και των στηλών. Ακολουθούν οι τιμές των μεταβαλλόμενων παραμέτρων και οι χρόνοι υπολογισμού των δέντρων αποφάσεων:

ΑΡΙΘΜΟΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ	ΧΡΟΝΟΣ ΥΠΟΛΟΓΙΣΜΟΥ
15	4.17
30	5.93
45	8.60

Πίνακας 10: Χρόνος υπολογισμού δέντρου αποφάσεων (σε δευτερόλεπτα) για διαφορετικό αριθμό χαρακτηριστικών στο σύνολο δεδομένων NMDC.



Εικόνα 34: Χρόνος υπολογισμού δέντρου αποφάσεων (σε δευτερόλεπτα) για διαφορετικό αριθμό χαρακτηριστικών στο σύνολο δεδομένων NMDC.

Όπως παρατηρούμε στον *Πίνακα 10*, ο χρόνος υπολογισμού ενός δέντρου αποφάσεων αυξάνεται ανάλογα με τον αριθμό των χαρακτηριστικών. Αποτελεί αναμενόμενη συμπεριφορά, εφόσον ένα παραπάνω χαρακτηριστικό σημαίνει μία παραπάνω μεταβλητή που συμμετέχει στη διαδικασία κατασκευής του δέντρου αποφάσεων, και άρα, επιπλέον δεδομένα. Στην *Εικόνα 34*, διακρίνουμε καλύτερα τη συσχέτιση μεταξύ χρόνου υπολογισμού και αριθμού χαρακτηριστικών, με τη γραμμή να ακολουθεί μία σχεδόν γραμμική κατεύθυνση.

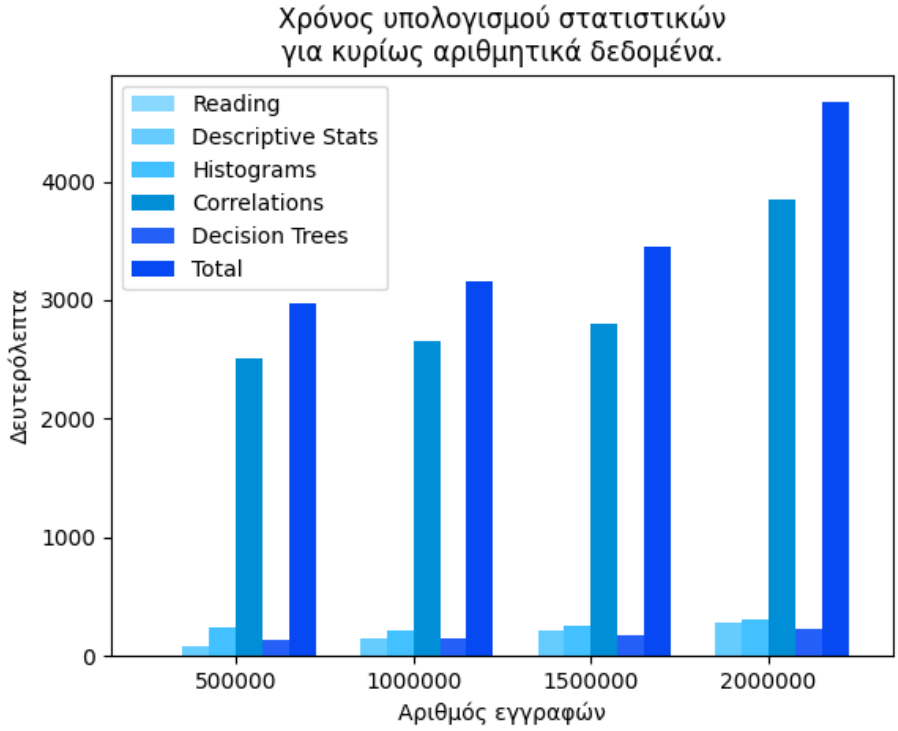
4.2.2 Χρονομέτρηση της διαδικασίας υπολογισμού των στατιστικών

Τέλος, αξίζει να διερευνήσουμε το υπολογιστικό βάρος που εισέφεραν οι νέες λειτουργίες που υλοποιήσαμε, στο ολικό σύστημα. Για να επιτύχουμε τον σκοπό μας, χρονομετρούμε για διάφορα σύνολα δεδομένων τη διαδικασία υπολογισμού των στατιστικών στοιχείων, ολικά και ξεχωριστά. Για να λάβουμε καλύτερη γνώση περί του αντίκτυπου που έφεραν οι αλλαγές μας, εξετάζουμε το σύστημα με δύο σύνολα δεδομένων, το "*users.json*" που περιέχει κυρίως αριθμητικά δεδομένα και το "*covid-19*" που περιέχει κυρίως κατηγορικά.

Για το σύνολο δεδομένων "*users.json*" έχουμε τα παρακάτω αποτελέσματα:

ΑΡΙΘΜΟΣ ΕΓΓΡΑΦΩΝ	500.000	1.000.000	1.500.000	2.000.000
ΕΓΓΡΑΦΗ ΣΥΝΟΛΟΥ	3	3	3	3
ΠΕΡΙΓΡΑΦΙΚΑ ΣΤΑΤΙΣΤΙΚΑ	82.25	147.25	215.50	280.25
ΙΣΤΟΓΡΑΜΜΑΤΑ	240.50	208.25	255.25	307.25
ΣΥΣΧΕΤΙΣΕΙΣ	2514.25	2655.50	2804.25	3845.25
ΔΕΝΤΡΑ ΑΠΟΦΑΣΕΩΝ	130.25	147.25	179.75	233.50
ΣΥΝΟΛΟ	2971.00	3160.25	3456.25	4667.25

Πίνακας 11: Χρόνος υπολογισμού όλων των στατιστικών (σε δευτερόλεπτα) για κυρίως αριθμητικά δεδομένα στο σύνολο δεδομένων Yelp.



Εικόνα 35: Χρόνος υπολογισμού όλων των στατιστικών (σε δευτερόλεπτα) για κυρίως αριθμητικά δεδομένα στο σύνολο δεδομένων Yelp.

Στον *Πίνακα 11* βλέπουμε όλες τις χρονομετρήσεις, για κάθε στατιστικό στοιχείο ξεχωριστά, καθώς και τον συνολικό χρόνο που χρειάστηκε. Οι μετρήσεις αυτές απευθύνονται σε διάφορα μεγέθη του συνόλου δεδομένων. Αρχικά, παρατηρούμε πως η εγγραφή του συνόλου είναι αμελητέα, ενώ τα περιγραφικά στατιστικά, τα ιστογράμματα και τα δέντρα αποφάσεων δεν έχουν μεγάλη διαφορά μεταξύ τους, σε σχέση με τον συνολικό χρόνο. Μέσω της *Εικόνας 34*, διακρίνουμε εύκολα τη μεγάλη συμβολή του υπολογισμού των συσχετίσεων στον συνολικό χρόνο. Αυτό οφείλεται στη φύση της

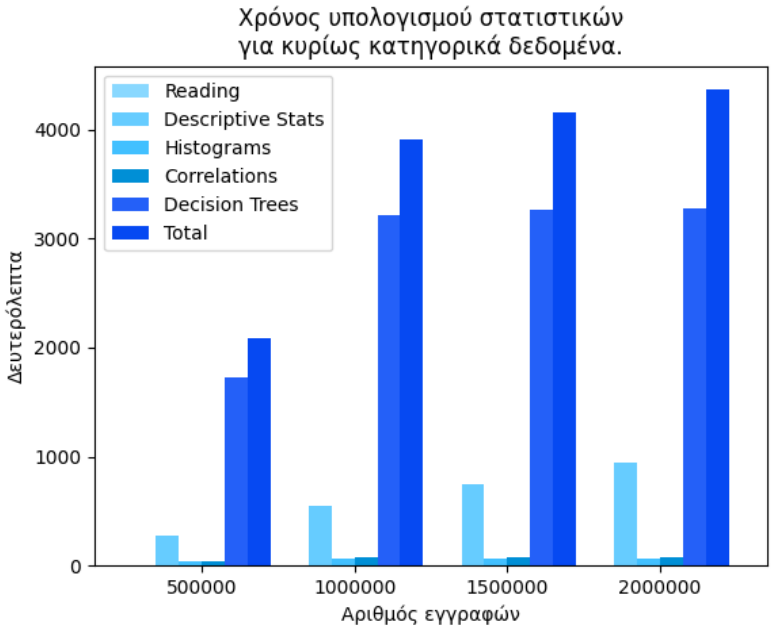
στατιστικής αυτής μετρικής, που συγκρίνει κάθε αριθμητική στήλη με όλες τις άλλες για να παράγει τη σχέση μεταξύ τους. Αποτελεί μία ακριβή υπολογιστικά διαδικασία και για αυτό καταλαμβάνει τον περισσότερο χρόνο όταν ασχολούμαστε με σύνολα δεδομένων που έχουν κυρίως αριθμητικές μεταβλητές. Στη συνέχεια, βλέπουμε πως οι χρόνοι σε όλα τα στατιστικά στοιχεία ακολουθούν ένα μοτίβο, δηλαδή αυξάνονται κατά ένα ελάχιστα μεταβαλλόμενο ποσοστό, ανάλογα με τον αριθμό των εγγραφών. Επομένως, ο συνολικός χρόνος υπολογισμού όλων των στατιστικών στοιχείων αυξάνεται με περίπου έναν προβλεπόμενο ρυθμό, με εξαίρεση τη χρονομέτρηση στις 2.000.000 εγγραφές, όπου η αύξηση ήταν ραγδαία, κατά περίπου χίλια δευτερόλεπτα. Αυτό οφείλεται αποκλειστικά και μόνο στον υπολογισμό των συσχετίσεων, ο οποίος ο καθυστέρησε ο ίδιος κατά περίπου χίλια δευτερόλεπτα, ενώ κοιτώντας τα δεδομένα των προηγούμενων μετρήσεων, κάποιος θα περίμενε μία αύξηση των 100 με 150 δευτερολέπτων.

Τέλος, αξίζει να αναλύσουμε τη δική μας συμβολή στην εργασία, μελετώντας το αντίκτυπο που έχουν τα ιστογράμματα και τα δέντρα αποφάσεων στον ολικό χρόνο υπολογισμού των στατιστικών στοιχείων. Λαμβάνοντας υπόψιν τα δεδομένα του *Πίνακα 11*, βρίσκουμε πως, για κυρίως αριθμητικά δεδομένα, τα δέντρα αποφάσεων προσφέρουν μία αύξηση χρόνου της τάξης του 5%, ενώ τα ιστογράμματα της τάξης του 7%.

Στη συνέχεια, θα εξετάσουμε τα πειράματα που απευθύνονται σε κυρίως κατηγορικά δεδομένα, δηλαδή στο σύνολο δεδομένων "*covid.csv*":

ΑΡΙΘΜΟΣ ΕΓΓΡΑΦΩΝ	**500.000**	**1.000.000**	**1.500.000**	**2.000.000**
ΕΓΓΡΑΦΗ ΣΥΝΟΛΟΥ	3	3	3	3
ΠΕΡΙΓΡΑΦΙΚΑ ΣΤΑΤΙΣΤΙΚΑ	280.00	553.67	748.00	951.33
ΙΣΤΟΓΡΑΜΜΑΤΑ	38.00	69.33	69.33	69.00
ΣΥΣΧΕΤΙΣΕΙΣ	38.33	72.00	71.33	72.67
ΔΕΝΤΡΑ ΑΠΟΦΑΣΕΩΝ	1731.67	3213.33	3266.33	3272.67
ΣΥΝΟΛΟ	2091.67	3909.33	4156.67	4367.00

Πίνακας 12: Χρόνος υπολογισμού όλων των στατιστικών (σε δευτερόλεπτα) για κυρίως κατηγορικά δεδομένα στο σύνολο δεδομένων Covid.



Εικόνα 36: Χρόνος υπολογισμού όλων των στατιστικών (σε δευτερόλεπτα) για κυρίως κατηγορικά δεδομένα στο σύνολο δεδομένων Covid.

Όπως και στις προηγούμενες μετρήσεις όλων των στατιστικών στοιχείων, η εγγραφή του συνόλου δεδομένων παραμένει αμελητέα στα 3 δευτερόλεπτα. Σύμφωνα με τον *Πίνακα 12*, ο χρόνος υπολογισμού των περιγραφικών στατιστικών αυξάνεται με περίπου σταθερό ρυθμό. Παράλληλα, παρατηρούμε πως τα περιγραφικά στατιστικά είναι τα δεύτερο πιο κοστοβόρα σε χρόνο, μετά τα δέντρα αποφάσεων. Επίσης, παρατηρούμε πως τα στατιστικά των συσχετίσεων και των ιστογραμμάτων παύουν να αυξάνονται μετά τις 1.000.000 εγγραφές. Μία πιθανή εξήγηση είναι η έλλειψη νέων αριθμητικών μεταβλητών και η βελτιστοποίηση του υπολογισμού των υπαρχόντων από το Spark. Είναι φυσιολογικό, ωστόσο, να έχουμε γενικά μικρούς χρόνους υπολογισμού για αυτά τα στατιστικά, αφού οι αριθμητικές στήλες είναι λίγες. Επίσης, παρατηρούμε την ίδια απόδοση στα δέντρα αποφάσεων με τα προηγούμενα πειράματα, όπου όλα τα δεδομένα ήταν κυρίως κατηγορικά. Αυτός αποτελεί και ο λόγος ο οποίος τα δέντρα αποφάσεων διαρκούν τόσο πολύ για να υπολογιστούν και καταλαμβάνουν το μεγαλύτερο μέρος του συνολικού χρόνου.

Στη περίπτωση αυτή, όπου τα δεδομένα είναι κυρίως κατηγορικά, τα ιστογράμματα παρέχουν μία 1% αύξηση στο συνολικό σύστημα, ενώ τα δέντρα αποφάσεων μία αύξηση της τάξης του 79%.

Κεφάλαιο 5. Επίλογος

5.1 Σύνοψη και συμπεράσματα

Η αυτοματοποίηση της στατιστικής ανάλυσης περιλαμβάνει τη χρήση προγραμμάτων ηλεκτρονικών υπολογιστών και αλγορίθμων για την εκτέλεση εργασιών ανάλυσης δεδομένων που συνήθως απαιτούσαν ανθρώπινη τεχνογνωσία. Ο σκοπός της αυτοματοποίησης της στατιστικής ανάλυσης είναι να αυξήσει την αποτελεσματικότητα, να μειώσει το ανθρώπινο λάθος και να κάνει την ανάλυση δεδομένων πιο προσιτή σε ένα ευρύτερο φάσμα χρηστών. Αυτό προσπάθησε και να καλύψει το Pythia, παρέχοντας στον χρήστη μια αναλυτική στατιστική αναφορά, με τις ελάχιστες δυνατές λειτουργίες και παραμετροποιήσεις.

Το Pythia, επέτρεψε στον χρήστη να εγγράψει ένα σύνολο δεδομένων και να παραλάβει μία αναφορά με στατιστικά στοιχεία, όπως τον μέσο όρο, την ελάχιστη και τη μέγιστη τιμή, συσχετίσεις, ιστογράμματα και δέντρα αποφάσεων, σχετικά με αυτό. Κατέστησε δυνατή τη δημιουργία ενός δέντρου αποφάσεων, προσφέροντας στον χρήστη τη λειτουργία δημιουργίας μίας καινούργιας κολόνας-στόχου, για την οποία θα παραγόταν το δέντρο. Παρείχε τη δυνατότητα αλλαγής του τύπου αρχείου της στατιστικής αναφοράς σε απλό κείμενο, *".json"* και *"markdown"*, με το τελευταίο να αποτελεί το πιο ευανάγνωστο και φιλικό προς τον χρήστη. Έδωσε τη δυνατότητα στον χρήστη να μπορεί να αποθηκεύσει ένα σύνολο δεδομένων, σε περίπτωση που είχε κάνει αλλαγές, στο σύνολο δεδομένων, τις οποίες δε ήθελε να χάσει.

Οι δικές μας συνεισφορές βοήθησαν στην περαιτέρω ανάπτυξη του εργαλείου, και το προχώρησαν ένα βήμα παραπέρα. Από τα αποτελέσματα των πειραμάτων, προέκυψαν διάφορα προβλήματα όπως η μεγάλη καθυστέρηση για τις συσχετίσεις και για τα δέντρα αποφάσεων, σε αριθμητικά και κατηγορικά δεδομένα αντίστοιχα. Ωστόσο, επιδιορθώθηκαν πολλά προβλήματα που λάμβαναν μέρος στον πηγαίο κώδικα και στην αρχιτεκτονική, και προστέθηκαν διάφορες λειτουργίες που δεν υπήρχαν. Οι αλλαγές μας κατέστησαν το εργαλείο πιο συντηρήσιμο και επεκτάσιμο από ποτέ. Πλέον αποτελεί ένα σημείο έναρξης για την ένταξη περισσότερων δυνατοτήτων και την δημιουργία κάτι σπουδαίου. Σε απλά πλαίσια, οι εντάξεις μας κατά την διπλωματική αυτή εργασία, αποδείχθηκαν συμβατές με τον σκοπό και την λειτουργία του λογισμικού.

5.2 Μελλοντικές επεκτάσεις

Οι επεκτάσεις που μπορούνε να εφαρμοστούνε στο εργαλείο είναι πολλές και είναι πάνω στην φαντασία, τη γνώση και τη δημιουργικότητα του προγραμματιστή να τις υλοποιήσει. Σε αυτήν την υποενότητα αναγράφονται μερικές από τις επεκτάσεις που μπορέσαμε να σκεφτούμε και προτείνουμε εμείς.

5.2.1 Υποστήριξη επιπλέον τύπων εισόδων

Αυτή τη στιγμή το Pythia υποστηρίζει τη φόρτωση αρχείων *Json*, *Csv* και *Tsv*. Στο μέλλον, θα μπορούσε να παρέχει υποστήριξη και για άλλους τύπους αρχείων, όπως *Psv*, *Parquet, Excel* κ.α.

5.2.2 Αυτόματη αναγνώριση του σχήματος των δεδομένων

Για να εγγραφεί ένα καινούργιο σύνολο δεδομένων στο σύστημα, απαιτείται να δοθεί σαν παράμετρος και το σχήμα των δεδομένων, δηλαδή το όνομα και ο τύπος του κάθε πεδίου. Εφόσον στο Pythia κυνηγάμε την αυτοματοποίηση και την μείωση των φορτικών διαδικασιών από τον χρήστη, αυτή η παράμετρος αξίζει να εξαλειφθεί. Το Spark ήδη παρέχει αυτόματη αναγνώριση του σχήματος με ένα παραπάνω πέρασμα των δεδομένων, ωστόσο δε δουλεύει καλά με όλους τους τύπους αρχείων και ίσως χρειάζεται μια διαφορετική προσέγγιση.

5.2.3 Επιλογή των κολόνων ενδιαφέροντος

Ο χρήστης θα μπορούσε να επιλέξει ποιες κολόνες τον ενδιαφέρουν ή μη από το σύνολο των δεδομένων, ώστε να μειωθεί ο αριθμός των στατιστικών υπολογισμών και ο όγκος των πληροφοριών στη τελική αναφορά.

5.2.4 Αυτόματη αναγνώριση labeled κολόνων

Αν και ο χρήστης μπορεί να δημιουργήσει μία δική του labeled κολόνα, αναλύουμε την προοπτική το πρόγραμμά μας να αναγνωρίζει από μόνο του ήδη υπάρχουσες κολόνες ως labeled. Υποψήφιες κολόνες μπορεί να είναι οι μη αριθμητικές, που έχουν πλήθος διαφορετικών τιμών κάτω του κατωφλίου που έχει προσδιορίσει ο χρήστης.'

5.2.5 Βελτιστοποίηση του υπολογισμού των στατιστικών στοιχείων

Το Pythia έχει φτιαχτεί με κύριο σκοπό την ευέλικτη, επεκτάσιμη και συντηρήσιμη αρχιτεκτονική, και την γρήγορη, αλλά σωστή, υλοποίηση στατιστικών στοιχείων. Ωστόσο, έχει παραλείψει την βελτιστοποίηση του υπολογισμού των στατιστικών

στοιχείων, και είναι ανοιχτό στην ελαχιστοποίηση των μετρήσεων που δεν χρειάζεται να λαμβάνουν μέρος.

5.2.6 Εμπλουτισμός των στατιστικών μετρήσεων

Το Spark παρέχει πολλούς διαφορετικούς αλγορίθμους υπολογισμών στατιστικών που δεν έχουν υλοποιηθεί ακόμη στο Pythia. Αυτοί κυμαίνονται από διαφορετικές μέθοδοι που δεν έχουμε υλοποιήσει σε ήδη υπάρχουσες μετρικές, όπως η μέθοδος *Spearman* για τις συσχετίσεις, έως και στατιστικά που δεν έχουμε υλοποιήσει καθόλου, όπως ο Chi-squared έλεγχος υπόθεσης, το clustering (η ομαδοποίηση των εγγραφών) και η αποτίμηση της ποιότητάς του.

5.2.7 Γραφικές παραστάσεις

Εκτός των ιστογραμμάτων, υπάρχουν κι άλλες γραφικές παραστάσεις που μπορούνε να συμπεριληφθούνε στο λογισμικό για να αναπαραστήσουν τα δεδομένα με διαφορετικό τρόπο και αποτελέσματα, όπως τα γραφήματα ράβδων, διασποράς και γραμμών, ανάμεσα σε πολλά άλλα.

5.2.8 Υποστήριξη επιπλέον τύπων αναφορών

Αν και προσθέσαμε έναν επιπλέον τύπο αναφοράς, το *markdown*, η εφαρμογή μπορεί να επωφεληθεί περαιτέρω και να υποστηρίξει και άλλους τύπους αρχείων αναφοράς, όπως *Pdf, PowerPoint* και *Latex*. Υπάρχουν έτοιμες βιβλιοθήκες για Java όπως η *itext5* για τη παραγωγή *pdf*, οι οποίες διευκολύνουν το έργο αυτό.

5.2.9 Δημιουργία διεπαφής χρήστη

Τέλος, το Pythia παρέχει πολλές λειτουργικότητες στον χρήστη, που δυστυχώς είναι δύσκολο να χρησιμοποιηθούν. Μπορούμε, όμως, να δημιουργήσουμε μια διεπαφή για τον τελικό χρήστη, που θα αφαιρεί την προγραμματιστική πολυπλοκότητα της χρήσης του, είτε μέσω ενός τοπικού γραφικού περιβάλλοντος (GUI), είτε μέσω εντολών στο τερματικό, είτε μέσω της μετατροπής του σε backend API και τη δημιουργία μιας διεπαφής σε περιηγητές διαδικτύου.

Βιβλιογραφία

[Agga15] Charu C. Aggarwal. Data Mining: The Textbook. Springer. April 27 2015

[Apac22] Apache Spark. Apache Software Foundation. Last visited: November 25 2022. Available at: https://spark.apache.org/

[Bosl12] Sarah Boslaugh. Statistics in a Nutshell. 2nd Edition, O'Reilly Media Inc., November 2012. Available at: https://learning.oreilly.com/library/view/statistics-in-a/9781449361129/

[DWDL20] Jules S. Damji, Brooke Wenig, Tathagata Das, Denny Lee. Learning Spark: Lightning-Fast Data Analytics. 2nd Edition. . O'Reilly Media, Inc.. July 2020. Available at: https://learning.oreilly.com/library/view/learning-spark-2nd/9781492050032/

[KKWZ15] Holden Karau, Andy Konwinski, Patrick Wendell, Matei Zahari. Learning Spark: Lightning-Fast Data Analysis. O'Reilly Media, Inc.. February 2015. Available at: https://learning.oreilly.com/library/view/learning-spark/9781449359034/

[LaLa14] Daniel T. Larose, Chantal D. Larose. Discovering Knowledge in Data: An introduction to Data Mining. 2nd Edition, Wiley, July 2014. Available at: https://learning.oreilly.com/library/view/discovering-knowledge-in/9781118873571/

[Lomb22] Project Lombok. Lasted visited: November 25 2022. Available at: https://projectlombok.org/

[RoOd14] Lior Rokach, Oded Maimon. Data Mining with Decision Trees: Theory and Applications. 2nd Edition, WSPC, September 3, 2014. Available at: https://doc.lagout.org/Others/Data%20Mining/Data%20Mining%20with%20Decision%20Trees_%20Theory%20and%20Applications%20%282nd%20ed.%29%20%5BRokach%20%26%20Maimon%202014-10-23%5D.pdf

[SnMR12] Chris Snijders, Uwe Matzat, Ulf-Dietrich Reips. 'Big Data': Big Gaps of Knowledge in the Field of Internet Science. International Journal of

Internet Science, 7 (1), 1-5, 2012. Available at: https://www.ijis.net/ijis7_1/ijis7_1_editorial.html

[Star22] Josh Starmer. The StatQuest Illustrated Guide to Machine Learning. StatQuest Publications, November 7, 2022. Available at: https://statquest.gumroad.com/l/wvtmc