Αυτόματη Παραγωγή Στατιστικού Προφίλ για Αλφαριθμητικά Σύνολα Δεδομένων

Βλαχόπουλος Λάμπρος

Διπλωματική Εργασία

Επιβλέπων: Π. Βασιλειάδης

Ιωάννινα, Μάρτιος 2025

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Παναγιώτη Βασιλειάδη. Είναι από τους πρώτους καθηγητές που είχα την τύχη να γνωρίσω από τα πρώτα έτη στο τμήμα Μηχανικών Η/Υ και πληροφορικής στα Ιωάννινα. Είναι αυτός στον οποίο οφείλεται πλέον η αγάπη μου για τον αντικειμενοστραφή προγραμματισμό και ειδικά η ενασχόλησή μου με την Java. Αυτός με δίδαξε τις πρώτες μου γραμμές κώδικα και φτάνω σήμερα να μπορώ να ανταποκριθώ και να συνεισφέρω σε πολύπλοκα συστήματα όπως το αντικείμενο της εν λόγω εργασίας. Τον ευχαριστώ λοιπόν θερμά για την βοήθεια, την κατανόηση και την συνεργασία καθ' όλη την διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας.

Επιπλέον θα ήθελα να ευχαριστήσω τους γονείς μου που πάντα πιστεύουν σε μένα και τις ικανότητες μου. Τους ευχαριστώ θερμά για την στήριξη και την αγάπη τους όλα αυτά τα χρόνια.

24/02/2025

Λάμπρος Βλαχόπουλος

Περίληψη στα ελληνικά

Η διπλωματική εργασία επικεντρώνεται στην επέκταση ενός ήδη υπάρχοντος συστήματος για την αυτόματη παραγωγή στατιστικού προφίλ για αλφαριθμητικά σύνολα δεδομένων, με στόχο την ημιαυτόματη εκτίμηση των τύπων των στηλών μέσω δειγματοληψίας. Το σύστημα προτείνει τους πιο πιθανούς τύπους δεδομένων για κάθε στήλη, δίνοντας στον χρήστη τη δυνατότητα να επιβεβαιώσει ή να τροποποιήσει τις εκτιμήσεις. Η επέκταση του προφίλ περιλαμβάνει βασικά στατιστικά στοιχεία, όπως αριθμό γραμμών, μέγεθος αρχείου, ημερομηνία ανάλυσης, και χαρακτηριστικά για κάθε στήλη (π.χ. αριθμός μοναδικών τιμών, πιο συχνή τιμή, κατανομή μέσω τεταρτημόριων). Αναπτύσσεται επίσης ένα γραφικό περιβάλλον (Java Swing) για την δια δραστική χρήση των λειτουργιών που προσφέρει το λογισμικό χωρίς να επηρεάζεται η υπάρχουσα αρχιτεκτονική του backend. Η εργασία περιλαμβάνει επίσης αναδιάρθρωση του κώδικα (refactoring) στο πακέτο outliers.

Λέξεις Κλειδιά: Επέκταση υπάρχοντος συστήματος, Αυτόματη εκτίμηση τύπων στηλών, Δειγματοληψία αρχείου, Προφίλ δεδομένων (γραμμές, μέγεθος, ημερομηνία ανάλυσης), Στατιστικά στήλης (μοναδικές τιμές, πιο συχνή τιμή, κατανομή τεταρτημόριων), Java Swing (γραφικό περιβάλλον), Ανίχνευση τύπων δεδομένων, Refactoring στο πακέτο outliers.

Abstract

The thesis is about expanding an existing system that automatically creates statistical profiles for alphanumeric datasets. The goal is to automatically estimate the type of each column through sampling. The system suggests the most likely data types for each column, and the user can confirm or change these estimates. The profile includes basic information like the number of rows, file size, analysis date, and details for each column (e.g., number of unique values, most frequent value, distribution using quartiles). A graphical interface (Java Swing) is also developed for interactive data type detection, dataset registration, and viewing reports without affecting the existing backend. The thesis also includes refactoring the code in the outliers package.

Keywords: System expansion, Automated column type estimation, File sampling, Data profile (rows, size, analysis date), Column statistics (unique values, most frequent value, quartile distribution), Java Swing (graphical interface), Data type detection, Refactoring in the outliers package.

Πίνακας περιεχομένων

Περιεχόμενα

Κεφάλαιο 1. Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Η διαχείριση και ανάλυση δεδομένων αποτελεί έναν από τους σημαντικότερους τομείς της σύγχρονης επιστήμης και τεχνολογίας. Η συνεχής αύξηση της διαθεσιμότητας δεδομένων, σε συνδυασμό με την ανάγκη για εξαγωγή χρήσιμων συμπερασμάτων, έχει οδηγήσει στην ανάπτυξη προηγμένων εργαλείων και τεχνικών. Ένα κρίσιμο βήμα στην ανάλυση δεδομένων είναι η δημιουργία ενός στατιστικού προφίλ, το οποίο παρέχει μια συνοπτική εικόνα των χαρακτηριστικών του συνόλου δεδομένων. Η αυτοματοποίηση αυτής της διαδικασίας είναι ζωτικής σημασίας για την αποτελεσματική επεξεργασία μεγάλου όγκου δεδομένων, ειδικά όταν πρόκειται για αλφαριθμητικά σύνολα, τα οποία είναι ευρέως διαδεδομένα σε διάφορες εφαρμογές.

Σκοπός της παρούσας εργασίας είναι η επέκταση ενός υπάρχοντος συστήματος για την αυτόματη παραγωγή στατιστικού προφίλ αλφαριθμητικών συνόλων δεδομένων. Το νέο σύστημα θα μπορεί να αναγνωρίζει ημιαυτόματα τον τύπο των στηλών και να υπολογίζει βασικές στατιστικές πληροφορίες, όπως τον αριθμό των null τιμών, των διακριτών τιμών, καθώς και τις τιμές με τη μεγαλύτερη συχνότητα εμφάνισης (mode). Επιπλέον, θα υπολογίζει τεταρτημόρια (Q1, Q2, Q3) και θα δημιουργεί ισοϋψή διαγράμματα (quartile histograms) για οπτικοποίηση κατανομής των δεδομένων. Το σύστημα θα παρέχει συνοπτικές αναφορές ενημερωμένες με τις παραπάνω προσθήκες , ενώ θα υποστηρίζει δια δραστικές αιτήσεις profiling, επιτρέποντας στον χρήστη να χρησιμοποίει τον λογισμικό πιο εύκολα και γρήγορα χωρίς να απαιτούνται προγραμματιστικές ικανότητες.

Η δημιουργία στατιστικού προφίλ είναι μια χρονοβόρα και επίπονη διαδικασία, ειδικά για μεγάλα και σύνθετα σύνολα δεδομένων. Η χειροκίνητη ανάλυση απαιτεί εξειδικευμένες γνώσεις και εργαλεία και είναι επιρρεπής σε ανθρώπινα λάθη. Η αυτοματοποίηση αυτής της διαδικασίας είναι σημαντική για την επιτάχυνση της ανάλυσης δεδομένων και τη βελτίωση της ακρίβειας των αποτελεσμάτων. Το πρόβλημα

είναι δύσκολο λόγω της ποικιλομορφίας των αλφαριθμητικών δεδομένων, καθώς οι τιμές μπορεί να είναι σε διάφορες μορφές και να περιέχουν σφάλματα ή ασυνέπειες.

1.2 Οργάνωση του τόμου

Αυτό το κείμενο περιγράφει αναλυτικά τη διπλωματική εργασία, από την αρχή μέχρι τα αποτελέσματά της.

Στο δεύτερο κεφάλαιο περιγράφεται αναλυτικά το θέμα της εργασίας. Πιο συγκεκριμένα, αναφέρεται ο στόχος της εργασίας (2.1), οι σχετικές έρευνες και τεχνολογίες που έχουν χρησιμοποιηθεί (2.2), οι οποίες χωρίζονται σε δύο υποκατηγορίες: το θεωρητικό υπόβαθρο (2.2.1) και το τεχνολογικό υπόβαθρο (2.2.2). Επιπλέον, γίνεται ανάλυση των απαιτήσεων του συστήματος (2.3), καθορίζοντας τα κριτήρια που πρέπει να πληροί η υλοποίηση.

Το τρίτο κεφάλαιο περιγράφει τη σχεδίαση και την υλοποίηση του συστήματος. Ξεκινά με τον ορισμό του προβλήματος και την περιγραφή των αλγορίθμων επίλυσης (3.1), όπου αναλύεται το πρόβλημα (3.1.1), η υπάρχουσα υλοποίηση και οι δομές του συστήματος (3.1.2), καθώς και η διαδικασία επίλυσης των προβλημάτων (3.1.3). Στη συνέχεια, παρουσιάζεται η σχεδίαση και η αρχιτεκτονική του λογισμικού (3.2), η οποία αναλύεται μέσα από τα επιμέρους πακέτα του συστήματος. Ακολουθεί η σχεδίαση και τα αποτελέσματα ελέγχου του λογισμικού (3.3), όπου περιγράφεται η μεθοδολογία ελέγχου (3.3.1) και γίνεται μια αναλυτική παρουσίαση των ελέγχων (3.3.2). Το κεφάλαιο κλείνει με την περιγραφή της διαδικασίας εγκατάστασης και υλοποίησης (3.4), καθώς και της επεκτασιμότητας του λογισμικού (3.5).

Το τέταρτο κεφάλαιο αφορά την αξιολόγηση της απόδοσης του συστήματος. Εξηγείται η μεθοδολογία πειραματισμού (4.1) και παρουσιάζονται αναλυτικά τα αποτελέσματα (4.2) μέσω αριθμητικών δεδομένων και γραφημάτων, εξετάζοντας αν το σύστημα λειτουργεί σωστά και πόσο αποτελεσματικό είναι.

Το τελευταίο κεφάλαιο συνοψίζει τα βασικά συμπεράσματα της εργασίας (5.1) και περιγράφει προτάσεις για μελλοντικές επεκτάσεις (5.2), ώστε το σύστημα να βελτιωθεί περαιτέρω και να γίνει πιο χρήσιμο.

Κεφάλαιο 2. Περιγραφή Θέματος

2.1 Στόχος της εργασίας

Η παρούσα διπλωματική εργασία επιδιώκει την επέκταση μιας πλατφόρμας για την αυτόματη παραγωγή στατιστικών προφίλ για αλφαριθμητικά σύνολα δεδομένων και τη διευκόλυνση δια δραστικών αιτημάτων profiling από τον αναλυτή. Το κύριο πρόβλημα που επιχειρεί να λύσει είναι η ανάγκη για ημιαυτόματη εκτίμηση του τύπου των στηλών ενός αρχείου δεδομένων και η παραγωγή στατιστικών προφίλ με βάση αυτή την εκτίμηση. Αυτό επιτυγχάνεται μέσω μιας σειράς εργαλείων και αλγορίθμων που αναλύουν τα δεδομένα και προσφέρουν στατιστικά χαρακτηριστικά, όπως ο αριθμός των null τιμών, ο αριθμός των διακριτών τιμών, η κατανομή των τιμών και άλλα. Ένα επιπλέον κύριο πρόβλημα που αντιμετωπίζεται είναι η ανάπτυξη μιας εύχρηστης γραφικής δια προσωπείας, η οποία θα διευκολύνει τη διαδικασία ανάλυσης των δεδομένων και την προβολή των αποτελεσμάτων.

Τα ζητούμενα της εργασίας περιλαμβάνουν τα ακόλουθα:

1. Ημιαυτόματη εκτίμηση από το σύστημα για τον τύπο των στηλών.
2. Επέκταση του προφίλ αρχείου με χαρακτηριστικά που αφορούν ολόκληρο το dataset, όπως ο αριθμός των γραμμών, η διαδρομή του αρχείου(file path) αλλά και ολόκληρη ημερομηνία και ώρα έναρξης της ανάλυσης των δεδομένων.
3. Επέκταση του προφίλ των στηλών με χαρακτηριστικά που αφορούν τις null, διακριτές τιμές, mode(τιμές με την μεγαλύτερη συχνότητα εμφάνισης σε μια στήλη) αλλά και ισοϋψές ιστόγραμμα με τη μορφή quartiles.
4. Τροποποίηση των υπαρχόντων αναφορών ώστε να περιλαμβάνουν τα παραπάνω χαρακτηριστικά(2,3).
5. Ανακατασκευή του πακέτου outlier.
6. Υλοποίηση δια δραστικής διεπαφής χρήστη για την εκτέλεση διαφορετικών αναλύσεων και την προβολή των αποτελεσμάτων.

Συνολικά, η εργασία αποσκοπεί στη δημιουργία ενός εργαλείου που είναι ικανό να αντιμετωπίζει ποικιλία δεδομένων και να παρέχει ένα συνεκτικό στατιστικό προφίλ για τα αλφαριθμητικά datasets, προσφέροντας παράλληλα μια φιλική προς τον χρήστη διεπαφή που διευκολύνει την εκτέλεση των αναλύσεων και την προβολή των αποτελεσμάτων.

2.2 Σχετικές εργασίες και τεχνολογίες

2.2.1 Θεωρητικό Υπόβαθρο

Data Profiling(Δημιουργία στατικού προφίλ δεδομένων)

Η δημιουργία ενός στατικού προφίλ δεδομένων είναι το σύνολο των δραστηριοτήτων και διαδικασιών που σχεδιάζονται για τον προσδιορισμό των μεταδεδομένων ενός συνόλου δεδομένων [AGNP18]. Αυτά τα μεταδεδομένα, όπως στατιστικά στοιχεία για τα δεδομένα ή οι εξαρτήσεις μεταξύ στηλών, μπορούν να βοηθήσουν στην κατανόηση και διαχείριση νέων συνόλων δεδομένων. Η διαδικασία δημιουργίας του προφίλ μας επιτρέπει να ανακαλύψουμε πόσα κενά υπάρχουν, πόσες διαφορετικές τιμές υπάρχουν σε κάθε στήλη και ποια είναι τα συνηθέστερα πρότυπα μεταξύ των τιμών στη στήλη [ABGN15]. Έχοντας αυτές τις πληροφορίες, μπορούμε να είμαστε πιο προετοιμασμένοι για να καθαρίσουμε τα δεδομένα μας, να τα ενσωματώσουμε σε άλλα συστήματα ή ακόμα και να τα αναλύσουμε περαιτέρω για να αντλήσουμε σημαντικές πληροφορίες.

Data Profiling Tasks

Η διαδικασία δημιουργίας προφίλ δεδομένων αποτελεί κρίσιμο βήμα στην ανάλυση και επεξεργασία των πληροφοριών. Αν και είναι μια σημαντική διαδικασία, δεν είναι κάτι που μπορεί να ολοκληρωθεί εύκολα. Αντίθετα, απαιτεί περαιτέρω ανάλυση και χωρισμό σε διάφορες λειτουργίες και καθήκοντα.

Η πολυπλοκότητα αυτής της διαδικασίας προκύπτει από την ανάγκη να διαχειριστούμε διάφορους τύπους δεδομένων, οι οποίοι μπορεί να προέρχονται από διαφορετικές πηγές και να έχουν διαφορετικές δομές και μορφές. Επιπλέον, η ποικιλία των λειτουργιών περιλαμβάνει την ανάλυση κάθε στήλης ξεχωριστά, τον εντοπισμό εξαρτήσεων μεταξύ στηλών και ακόμη και την εξέταση μη σχετικών δεδομένων, όπως τα δέντρα, τα γραφήματα και τα κείμενα.

Τόσο στο βιβλίο [AGNP18] όσο και [ABGN15] οι Abedjan, Z., Golab, L., Naumann, F., & Papenbrock, T., αναδεικνύουν ότι η διαδικασία δημιουργίας ενός προφίλ δεδομένων αντιπροσωπεύει ένα ευρύ φάσμα εργασιών και καθηκόντων, προσαρμοσμένο στον τύπο των δεδομένων που αντιμετωπίζει.

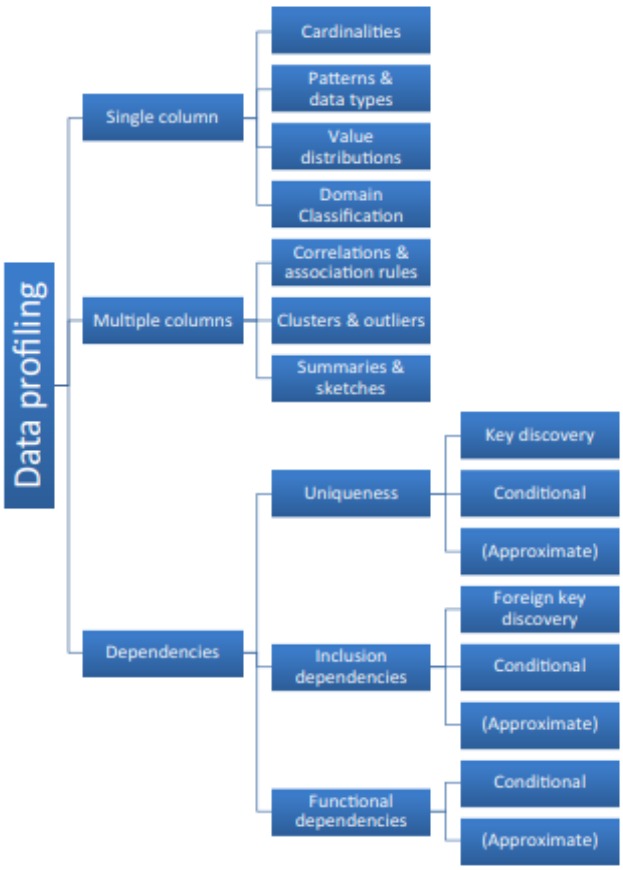


Fig. 1 A classification of traditional data profiling tasks

Εικόνα 1 Ταξινόμηση εργασιών [AGNP18],[ABGN15]

Στην Εικόνα 1 παρουσιάζεται η διαδικασία δημιουργίας ενός στατικού προφίλ δεδομένων. Γίνεται εύκολα αντιληπτό, ότι αυτή η διαδικασία αποτελείται από αρκετές επιμέρους λειτουργίες που εστιάζουν είτε σε ανάλυση που αφορά μια στήλη είτε σε λειτουργίες που αφορούν πολλές στήλες αλλά και των εξαρτήσεων μεταξύ τους.

Δημιουργία στατικού προφίλ μεμονωμένων στηλών

Στο βιβλίο [AGNP18] και [ABGN15] από τους Abedjan, Z., Golab, L., Naumann, F., & Papenbrock, T., παρουσιάζεται η σημασία της διαδικασίας ανάλυσης μιας μεμονωμένης στήλης ως κρίσιμο στάδιο στη διαδικασία δημιουργίας του προφίλ δεδομένων. Κατά τη διάρκεια αυτής της διαδικασίας, πραγματοποιείται λεπτομερής ανάλυση και συλλογή μεταδεδομένων για κάθε στήλη ενός πίνακα δεδομένων, προσφέροντας μια ολοκληρωμένη εικόνα των χαρακτηριστικών των δεδομένων.

Συγκεκριμένα, κατά τη διαδικασία αυτή, αναλύεται το μέγεθος της στήλης, δηλαδή ο αριθμός των τιμών που περιλαμβάνει, εξετάζεται ο αριθμός των διακριτών τιμών που παρατηρούνται στη στήλη, και εξετάζεται η πληρότητα, που αναφέρεται στον αριθμό των μη μηδενικών τιμών σε μια στήλη, παρέχοντας πληροφορίες για την πυκνότητα των δεδομένων.

Μέσω της ανάλυσης μιας μεμονωμένης στήλης επιτρέπεται η δημιουργία ιστογραμμάτων και κατανομών των τιμών, καθώς και η αναγνώριση τυπικών προτύπων στα δεδομένα. Αυτό επιτρέπει στους επιστήμονες δεδομένων να αποκτήσουν μια πιο εμπεριστατωμένη κατανόηση των δεδομένων τους, αναδεικνύοντας τάσεις, παραλλαγές και ανωμαλίες στα δεδομένα. Επιπλέον, μπορούν να προβλέψουν μελλοντικές τάσεις και να λάβουν αποφάσεις βασισμένες σε αυτές τις πληροφορίες.

Πιο συγκεκριμένα η single-column analysis(Δημιουργία στατικού προφίλ μεμονωμένων στηλών) περιλαμβάνει τις παρακάτω εργασίες υπολογισμού όπως φαίνονται στην Εικόνα 2:

Table 3.1: Overview of single-column profiling

Category	Task	Task Description
Cardinalities	num-rows	Number of rows
	null values	Number or percentage of null values
	distinct	Number of distinct values
	uniqueness	Number of distinct values divided by number of rows
Value Distributions	histogram	Frequency histograms (equi-width, equi-depth, etc.)
	extremes	Minimum and maximum values in a numeric column
	constancy	Frequency of most frequent value divided by number of rows
	quartiles	Three points that divide (numeric) values into four equal groups
	first digit	Distribution of first digit in numeric values; to check Benford's law
Data Types, Patterns, and Domains	basic type	Numeric, alphanumeric, date, time, etc.
	data type	DBMS-specific data type (varchar, timestamp, etc.)
	lengths	Minimum, maximum, median, and average lengths of values within a column
	size	Maximum number of digits in numeric values
	decimals	Maximum number of decimals in numeric values
	patterns	Histogram of value patterns (Aa9. . .)
	data class	Generic semantic data type, such as code, indicator, text, date/time, quantity, identier
	domain	Semantic domain, such as credit card, first name, city, phenotype

Εικόνα 2 Επισκόπηση εργασιών single-column profiling[AGNP18]

Cardinalities Tasks

Οι πληθικότητες ή οι μετρήσεις των τιμών σε μια στήλη αποτελούν βασική μορφή μεταδεδομένων στον τομέα της διαχείρισης δεδομένων. Σύμφωνα με το άρθρο των Abedjan, Z., Golab, L., & Naumann, F. [ABGN15], αυτές οι μετρήσεις είναι κρίσιμες για την κατανόηση της δομής των δεδομένων και για την αξιολόγηση της αποδοτικότητας διάφορων αλγορίθμων ανάλυσης. Μια από τις βασικές εργασίες σε αυτό το πλαίσιο είναι η αξιολόγηση του πλήθους των σειρών σε έναν πίνακα δεδομένων (numrows), η οποία αντανακλά τον αριθμό των εννοιών που αναπαρίστανται στα δεδομένα.

Η πληροφορία σχετικά με το μήκος των τιμών σε χαρακτήρες (value length) είναι επίσης σημαντική και μπορεί να χρησιμοποιηθεί για ανάκτηση του σχήματος των δεδομένων (schema reverse engineering), ανίχνευση ακραίων τιμών (outliers) και μορφοποίηση. Η

παραπάνω πληροφορία περιλαμβάνει το μέγεθος της μεγαλύτερης και της μικρότερης τιμής, καθώς και το μέσο μήκος των τιμών.

Η καταμέτρηση των κενών κελιών (null values) είναι σημαντική για τον προσδιορισμό της πληρότητας μιας στήλης. Το πλήθος των διακριτών τιμών (distinct) μπορεί να χρησιμοποιηθεί για την εκτίμηση της επιλεκτικότητας επιλογής ή ένωσης ερωτημάτων, με περισσότερες διακριτές τιμές υποδεικνύοντας μεγαλύτερη επιλεκτικότητα. Επιπλέον, η μοναδικότητα των τιμών μιας στήλης αναφέρεται στην κατανομή των τιμών και μπορεί να χρησιμοποιηθεί για τον προσδιορισμό υποψήφιων κλειδιών σε μια βάση δεδομένων.

Το άρθρο[ABGN15] αναφέρει ότι η καταμέτρηση των πλυθικοτήτων μπορεί να γίνει αποτελεσματικά σε ένα μόνο πέρασμα πάνω από τα δεδομένα, καθιστώντας τη διαδικασία αποδοτική και χρήσιμη για την ανάλυση δεδομένων.

Value distributions Tasks

Η ανάλυση των κατανομών τιμών αποτελεί ένα σημαντικό κομμάτι της διαδικασίας ανάλυσης δεδομένων, και το άρθρο [ABGN15] από τους Abedjan, Golab και Naumann αναδεικνύει τη σημασία αυτής της διαδικασίας. Η ανάλυση ξεκινά με τη συλλογή δεδομένων από διάφορες πηγές, όπως βάσεις δεδομένων ή αρχεία κειμένου, και στη συνέχεια ακολουθεί η κατασκευή ιστογράμματος για την αναπαράσταση της συχνότητας των τιμών.

Μια σημαντική πτυχή που αναδεικνύεται είναι η αξιολόγηση της σταθερότητας μιας στήλης, η οποία μετρά το ποσοστό μιας συγκεκριμένης τιμής σε σχέση με το σύνολο των τιμών της στήλης, δείχνοντας αν περιέχει επαναλαμβανόμενα ή μη πληροφοριακά δεδομένα. Αν, για παράδειγμα, το 90% των τιμών μιας στήλης είναι το ίδιο, τότε αυτή η στήλη θεωρείται σταθερή. Αυτό μπορεί να έχει σημασία στην ανάλυση δεδομένων, γιατί μια υπερβολικά σταθερή στήλη ίσως δεν παρέχει χρήσιμες πληροφορίες ή μπορεί να κρύβει προβλήματα στα δεδομένα.

Εκτός από αυτά, υπάρχουν και άλλα στατιστικά μέτρα και αναλύσεις που μπορούν να εφαρμοστούν, όπως η κατανομή κωδικών Soundex, τα n-grams, η αντίστροφη κατανομή συχνότητας και η εντροπία της κατανομής συχνότητας. Αυτά τα εργαλεία και μέθοδοι προσφέρουν πλήρη κατανόηση της φύσης των δεδομένων και οδηγούν σε ενδιαφέροντα ευρήματα και συμπεράσματα.

Types and patterns, and Domains

Η διαδικασία ανάλυσης τύπων δεδομένων, προτύπων και σημασιολογικών πεδίων αποτελεί θεμελιώδη διαδικασία στον τομέα της διαχείρισης και ανάλυσης δεδομένων

συμφωνα με τους Abedjan, Z., Golab, L., & Naumann [ABGN15]. Σε αυτό το πλαίσιο, η ανάλυση αυτή περιλαμβάνει τρία βασικά βήματα:

- **Ανάλυση Τύπων Δεδομένων:** Αρχικά, γίνεται ανάλυση των δεδομένων για να προσδιοριστεί τι είδους πληροφορίες περιέχει κάθε στήλη. Εξετάζεται αν οι τιμές είναι αριθμοί, κείμενο, ημερομηνίες ή χρόνοι. Αυτό γίνεται ελέγχοντας αν οι τιμές περιέχουν αριθμούς, γράμματα ή ειδικούς χαρακτήρες που υποδηλώνουν την ύπαρξη των παραπάνω.
- **Ανάλυση Προτύπων:** Αφού αναλυθούν οι τύποι δεδομένων, το επόμενο βήμα είναι η ανίχνευση συχνά εμφανιζόμενων προτύπων στις τιμές των στηλών. Αυτό περιλαμβάνει τον εντοπισμό μοτίβων όπως τηλεφωνικοί αριθμοί ή ημερομηνίες, μέσω της χρήσης κανονικών εκφράσεων όπως +dd (ddd) ddd dddd ή \d3-\d3-\d4 ή άλλων μεθόδων που μπορούν να ανιχνευθούν συγκεκριμένα πρότυπα. Στο άρθρο[ABGN15] και στο βιβλίο [AGNP18] προτείνονται τρεις κύριες προσεγγίσεις για τον εντοπισμό του σημασιολογικού πεδίου μιας στήλης:
    - Η μέθοδος MDL (Minimal Description Length), που προτείνουν οι Raman και Hellerstein [RAHE01], εφαρμόζεται, για παράδειγμα, στο εργαλείο Potter's Wheel. Αυτή η μέθοδος μοντελοποιεί το μήκος περιγραφής ως συνδυασμό ακρίβειας, ανάκλησης και συνοπτικότητας.
    - Το σύστημα RelIE[LKRV08], το οποίο σχεδιάστηκε για την εξαγωγή πληροφοριών από κείμενα, δημιουργεί κανονικές εκφράσεις βασιζόμενο σε εκπαιδευτικά δεδομένα με θετικά και αρνητικά παραδείγματα.
    - Η προσέγγιση που παρουσιάζει ο Fernau[Fern09] για τη μάθηση κανονικών εκφράσεων από δεδομένα και παρέχει έναν αλγόριθμο για αυτή την εργασία.
- **Ανάλυση Σημασιολογικών Πεδίων:** Σε αυτό το βήμα, επιχειρείται η σημασιολογική ερμηνεία των δεδομένων, πέραν της συντακτικής τους μορφής. Αυτό είναι το πιο προβληματικό βήμα καθώς αφορά το να κατανοήσουμε το νόημα πίσω από τα δεδομένα. Για παράδειγμα, μια στήλη που περιέχει αριθμούς μπορεί να αντιπροσωπεύει είτε ποσά χρημάτων είτε μετρήσεις θερμοκρασίας. Συνήθως απαιτείται ανθρώπινη επέμβαση ή εξειδικευμένες αναλυτικές μεθόδους, καθώς η αυτοματοποίηση είναι πιο περίπλοκη σε αυτό το επίπεδο. Στο άρθρο [ABGN15] και στο βιβλίο [AGNP18] προτείνονται προσεγγίσεις για τον εντοπισμό του σημασιολογικού πεδίου μιας στήλης:
    - Ο Zhang και οι συνεργάτες του [ZHO+11] προτείνουν την ομαδοποίηση στηλών που έχουν την ίδια σημασία σε διαφορετικούς πίνακες μιας

βάσης δεδομένων, διευκολύνοντας έτσι τη διαδικασία αντιστοίχισης σχημάτων (schema matching).

- Στο έργο των Vogel και Naumann [VoNa11], επιχειρείται η αντιστοίχιση στηλών σε προκαθορισμένα σημασιολογικά πεδία, όπως δεδομένα που σχετίζονται με άτομα (π.χ., ονόματα, διευθύνσεις).
- Οι Zhang και Chakrabarti [ZhCh13] επεκτείνουν την ανάλυση σημασιολογικών πεδίων και σε αριθμητικά δεδομένα, τα οποία είναι ιδιαίτερα δύσκολο να ταξινομηθούν σημασιολογικά.

Δημιουργία προφίλ πολλαπλών στηλών

Η ανάλυση πολλαπλών στηλών αποτελεί μια σημαντική προσέγγιση στον τομέα της ανάλυσης δεδομένων, που έχει εξελιχθεί και αναλυθεί από διάφορους ερευνητές. Σύμφωνα με τη μελέτη των Abedjan, Golab, & Naumann [ABGN15], η ανάλυση αυτή επιτρέπει την ανακάλυψη σημαντικών πληροφοριών από πολλές στήλες ταυτόχρονα σε ένα σύνολο δεδομένων. Αυτή η προσέγγιση επικεντρώνεται στην ανακάλυψη συσχετίσεων μεταξύ διαφορετικών χαρακτηριστικών του συνόλου δεδομένων, καθώς και στην εξαγωγή συνεκτικών υποσυνόλων εγγραφών δεδομένων και τον εντοπισμό εκκεντρικών τιμών. Επιπλέον, η δημιουργία περιλήψεων και sketches μεγάλων συνόλων δεδομένων συμβάλλει στην κατανόηση των παρατηρούμενων προτύπων και τάσεων στα δεδομένα.

Η ανάλυση πολλαπλών στηλών έχει εφαρμογές σε πολλούς τομείς, συμπεριλαμβανομένων της εξερεύνησης δεδομένων και της ανάλυσης δεδομένων, καθώς και στην εφαρμογή διαδικασιών καθαρισμού δεδομένων για τη διασφάλιση της ποιότητας και της αξιοπιστίας των δεδομένων. Αναλυτικότερα οι εργασίες της πολλαπλής ανάλυσης στηλών όπως αναφέρονται [ABGN15] είναι:

- **Correlations and Association rules**

  Η ανάλυση συσχέτισης αποκαλύπτει σχέσεις μεταξύ αριθμητικών στηλών σε ένα σύνολο δεδομένων. Ένας απλός τρόπος για να επιτευχθεί αυτό είναι να υπολογιστούν οι συνδυαστικοί συσχετισμοί μεταξύ όλων των ζευγών στηλών. Εκτός από τους συσχετισμούς σε επίπεδο στήλης, οι συσχετίσεις σε επίπεδο τιμής μπορεί να παρέχουν χρήσιμες πληροφορίες στο προφίλ των δεδομένων. Οι Joseph Lee Rodgers & W. Alan Nicewander, στο άρθρο τους [RoNi88], παρουσιάζουν κάποιους από τους τρόπους εξαγωγής των συντελεστών συσχέτισης, με έναν από τους πιο γνωστούς να είναι ο Pearson, ο οποίος χρησιμοποιείται ευρέως στη στατιστική και την ανάλυση δεδομένων. Ενώ στο άρθρο των Matthias, Amer, Peter και Michael

[HDSM05], που αναφέρονται σχετικά με το Statistical Correlation, εκτός από τον Pearson, εξετάζουν και τον Spearman's Rank Correlation Coefficient, αλλά και άλλες μεθόδους που έχουν να κάνουν με την απόσταση μεταξύ των αντίστοιχων τιμών των ακολουθιών για τον προσδιορισμό της συσχέτισης ή μη.

Οι αλγόριθμοι για τη δημιουργία κανόνων συσχέτισης από δεδομένα, όπως παρουσιάζονται στην μελέτη τους [AgSr94] οι Rakesh Agrawal και Ramakrishnan Srikant, διαχωρίζουν το πρόβλημα σε δύο βήματα, όπως αναφέρουν και οι Ziawasch Abedjan, Lukasz Golab και Felix Naumann [ABGN15]:

- **Βήμα 1:** Ανακάλυψη Συχνών Συνόλων Στοιχείων. Κατά το πρώτο βήμα, εκτελούμε την ανίχνευση όλων των συχνών συνόλων στοιχείων στα δεδομένα μας. Αυτά τα συχνά σύνολα στοιχείων αναφέρονται σε συνδυασμούς διαφόρων γνωρισμάτων που εμφανίζονται συχνά μαζί. Για παράδειγμα, αν έχουμε ένα σύνολο στοιχείων που περιλαμβάνει τα προϊόντα "ψωμί" και "βούτυρο", θέλουμε να εντοπίσουμε πόσο συχνά εμφανίζεται αυτό το σύνολο σε σχέση με το σύνολο των συναλλαγών. Αυτό επιτυγχάνεται με τον υπολογισμό της συχνότητας εμφάνισης του συνόλου αυτού, γνωστής και ως υποστήριξη. Στόχος μας είναι να εντοπίσουμε τα συχνά σύνολα στοιχείων που υπερβαίνουν ένα προκαθορισμένο όριο υποστήριξης.
- **Βήμα 2:** Στο δεύτερο βήμα, δημιουργούμε κανόνες που δείχνουν τη σχέση μεταξύ διαφορετικών χαρακτηριστικών. Αν, για παράδειγμα, σε πολλές περιπτώσεις το "τμήμα = οικονομικά" και "θέση = βοηθός διευθυντή" οδηγούν σε επίδομα $1000, δημιουργούμε έναν κανόνα που λέει: "Όταν κάποιος είναι στο οικονομικό τμήμα και έχει θέση βοηθού διευθυντή, τότε το επίδομα είναι πιθανό να είναι $1000". Αυτός ο κανόνας βασίζεται στην εμπιστοσύνη, δηλαδή το ποσοστό των περιπτώσεων που ισχύει το συμπέρασμα (το επίδομα $1000).

Διάφοροι αλγόριθμοι έχουν προταθεί για τη χρήση σε συγκεκριμένα προβλήματα [ABGN15]. Μερικοί από αυτούς όπως:

- Ο αλγόριθμος Apriori, που εκμεταλλεύεται την ιδέα ότι όλα τα υποσύνολα ενός συχνού συνόλου πρέπει επίσης να είναι συχνά.
- Ο αλγόριθμος FP-Growth, που ανακαλύπτει συχνά σύνολα δεδομένων χωρίς το βήμα παραγωγής υποψήφιων συνόλων, παρέχοντας έτσι μια αποτελεσματική εναλλακτική λύση.

- Ο αλγόριθμος Eclat, που βασίζεται στη διασταύρωση των συνόλων ταυτότητας συναλλαγών (TID) και είναι καλύτερα κατάλληλος για την αντιμετώπιση μεγάλων συχνών συνόλων δεδομένων.

Ωστόσο, όπως αναφέρουν οι Zhicheng Liu και Aoqian Zhang στο άρθρο τους [LiZh20], όταν αντιμετωπίζουμε μεγάλα σύνολα δεδομένων (big data), είναι συχνά δύσκολο να εκτελέσουμε αλγόριθμους ανάλυσης λόγω του υψηλού χρόνου που απαιτείται για τον χειρισμό των δεδομένων. Η λύση σε αυτό το πρόβλημα συχνά προέρχεται από τη χρήση δειγματοληψίας (sampling). Μέσω της δειγματοληψίας, μπορούμε να αποκτήσουμε αντιπροσωπευτικά δείγματα των δεδομένων μας και να εφαρμόσουμε τους αλγόριθμους μας σε αυτά τα δείγματα, επιταχύνοντας τη διαδικασία ανάλυσης και μειώνοντας την απαιτούμενη μνήμη και τον χρόνο εκτέλεσης.

- **Clustering and outlier detection**

  Η μελέτη των Abedjan, Golab και Naumann [ABGN15] αναφέρει ότι στη δημιουργία προφίλ δεδομένων, είναι κρίσιμο να χωρίζουμε τα δεδομένα σε ομάδες που είναι ομοιογενείς, χρησιμοποιώντας εξειδικευμένους αλγορίθμους. Αυτή η διαδικασία βοηθά στην αναγνώριση σχέσεων και μοτίβων που δεν είναι άμεσα ορατά από την πρώτη ανάλυση. Μέσω της ομαδοποίησης, μπορούμε να κατανοήσουμε καλύτερα τα δεδομένα και να εξάγουμε χρήσιμες πληροφορίες που διαφορετικά θα μπορούσαν να παραμείνουν κρυφές. Αυτό καθιστά τη δημιουργία προφίλ δεδομένων μια πολύτιμη διαδικασία για την αποκόμιση σημαντικών συμπερασμάτων και την ανακάλυψη νέων τάσεων. Με βάση το βιβλίο [HaKP11] των Jiawei Han, Micheline Kamber και Jian Pei, ορισμένες από τις μεθόδους ομαδοποίησης παρουσιάζονται στην Εικόνα 3:

Method	General Characteristics
Partitioning methods	– Find mutually exclusive clusters of spherical shape – Distance-based – May use mean or medoid (etc.) to represent cluster center – Effective for small- to medium-size data sets
Hierarchical methods	– Clustering is a hierarchical decomposition (i.e., multiple levels) – Cannot correct erroneous merges or splits – May incorporate other techniques like microclustering or consider object "linkages"
Density-based methods	– Can find arbitrarily shaped clusters – Clusters are dense regions of objects in space that are separated by low-density regions – Cluster density: Each point must have a minimum number of points within its "neighborhood" – May filter out outliers
Grid-based methods	– Use a multiresolution grid data structure – Fast processing time (typically independent of the number of data objects, yet dependent on grid size)

Εικόνα 3 Παρουσίαση Μεθόδων Ομαδοποίησης με τα γενικά χαρακτηριστικά[HaKP11]

Και για κάθε μέθοδο στο βιβλίο [HaKP11] τους, οι Jiawei Han, Micheline Kamber και Jian Pei, μελετούν τους αλγορίθμους όπως παρουσιάζεται παρακάτω:

Μέθοδος	Αλγόριθμοι
Partitioning	k-Means k-Medoids
Hierarchical	Agglomerative versus Divisive Hierarchical Clustering Distance Measures in Algorithmic Methods BIRCH: Multiphase Hierarchical Clustering ,Using Clustering Feature Trees Chameleon: Multiphase Hierarchical ,Clustering Using Dynamic Modeling Probabilistic Hierarchical Clustering
Density-based	DBSCAN: Density-Based Clustering Based on Connected Regions with High Density OPTICS: Ordering Points to Identify the Clustering Structure DENCLUE: Clustering Based on Density Distribution Functions

Grid-Based	STING: Statistical Information Grid CLIQUE: An Apriori-like Subspace Clustering Method

Σύμφωνα με την έρευνα[KuFi16] των Kusumasari και Fitria, το OpenRefine, ένα εργαλείο δημιουργίας προφίλ δεδομένων, επικεντρώνεται κυρίως στη χρήση της μεθόδου Partitioning με τους αλγορίθμους k-Means και κλειδιού σύγκρουσης για το clustering.

Από την άλλη πλευρά, η ανίχνευση των αποκλίσεων αποτελεί ένα κρίσιμο και αναγκαίο βήμα, όπως αναφέρεται στο άρθρο[NYKK18] από τους Zahra Nazari, Seong-Mi Yu, Dongshik Kang και Yousuke Kawachi. Η αναγνώριση αποκλίσεων αποτελεί κλειδί για την απόκτηση μιας συνεκτικής και αξιόπιστης ανάλυσης δεδομένων, καθώς οι αποκλίσεις μπορούν να επηρεάσουν σοβαρά το συνολικό προφίλ των δεδομένων και να οδηγήσουν σε εσφαλμένα συμπεράσματα.

Στο άρθρο [FaNK17] των Anosh Fatima, Nosheen Nazir και Muhammad Gufran Khan, η εύρεση αποκλίσεων και η αφαίρεσή τους κατά τη φάση καθαρισμού των δεδομένων προς επεξεργασία είναι σημαντική, καθώς οι τιμές των αποκλίσεων μπορούν να επηρεάσουν σημαντικά τις τιμές άλλων μετρήσεων, όπως για παράδειγμα το μέσο όρο και την τυπική απόκλιση. Στην ανάλυσή τους, οι συγγραφείς του άρθρου [FaNk17] χρησιμοποιούν μεθόδους ομαδοποίησης, όπως τον αλγόριθμο k-Means, για την εύρεση αποκλίσεων, ειδικότερα των Null τιμών ή άλλων απουσιαζουσών τιμών στα δεδομένα, προκειμένου να αφαιρέσουν αυτές τις τιμές πριν από την ανάλυση των δεδομένων.

- **Summaries and sketches**

Στη μελέτη [ABGN15] από τους Abedjan, Golab και Naumann, επισημαίνει η σημασία των περιλήψεων (summaries) και των σκίτσων (sketches) ως σημαντικών εργαλείων για τη δημιουργία συνοπτικών αναπαραστάσεων των δεδομένων. Αυτές οι τεχνικές επιτρέπουν στους ερευνητές δεδομένων να κατανοήσουν τα χαρακτηριστικά των δεδομένων χωρίς την ανάγκη ανάλυσης των πλήρων συνόλων δεδομένων.

Η δημιουργία περιλήψεων και σκίτσων μπορεί να γίνει με διάφορες τεχνικές, όπως η δειγματοληψία και το hashing τιμών δεδομένων. Μέσω αυτών των τεχνικών, μπορεί να μειωθεί το μέγεθος των δεδομένων, ενώ ταυτόχρονα διατηρούνται σημαντικά χαρακτηριστικά για την κατανόηση τους.

Τέλος, οι περιλήψεις και τα σκίτσα χρησιμοποιούνται ευρέως σε διάφορους τομείς, όπως η απάντηση προσεγγιστικών ερωτημάτων, η επεξεργασία ροής δεδομένων και η εκτίμηση μεγεθών συνδυασμών.

Dependencies

Όπως αναφέρουν και οι Abedjan, Golab, Nauman [AGNP18][ABGN15] η ανίχνευση εξαρτήσεων είναι η πολυπλοκότερη κατηγορία εργασιών όσον αφορά την διαδικασία της δημιουργίας του προφίλ δεδομένων. Για το λόγο αυτό την διαχωρίζουν από την κατηγορία δημιουργίας προφίλ πολλαπλών στηλών και την μελετούν ξεχωριστά. Οι εξαρτήσεις σε ένα σύνολο δεδομένων αντιπροσωπεύουν τις σχέσεις μεταξύ των διαφόρων στηλών. Η ανίχνευση αυτών των εξαρτήσεων είναι κρίσιμη, καθώς επιτρέπει την κατανόηση της δομής και των συσχετίσεων μεταξύ των δεδομένων. Η αναγνώριση μοναδικών συνδυασμών στηλών μπορεί να οδηγήσει στον εντοπισμό κατάλληλου κλειδιού για έναν πίνακα, ενώ η ανίχνευση εξαρτήσεων ενσωμάτωσης μπορεί να προτείνει πώς να συνδυάσουμε διαφορετικά σύνολα δεδομένων. Επιπλέον, οι λειτουργικές εξαρτήσεις μπορούν να βοηθήσουν στην κατανόηση του πώς οι τιμές σε μια στήλη επηρεάζουν λειτουργικά τις τιμές μιας άλλης στήλης.

Όπως φαίνεται και στην Εικόνα 1, χωρίζουν [AGNP18],[ABGN15] την λειτουργία ανακάλυψης εξαρτήσεων σε 3 υποκατηγορίες λειτουργιών :

1. **Ανακάλυψη μοναδικών συνδυασμών στηλών (UCCs)**

   Η λειτουργία αυτή έχει ως σκοπό την αναγνώριση ή μη των μοναδικών και μη-μοναδικών συνδυασμών στηλών σε ένα σύνολο δεδομένων.

   Για παράδειγμα έστω ότι έχουμε τον παρακάτω πίνακα δεδομένων :

OrderID	CustomerID	ProductID	OrderDate
1	1001	101	2024-03-15
2	1002	102	2024-03-16
3	1001	103	2024-03-16

   Ένας μοναδικός συνδυασμός στηλών θα μπορούσε να είναι ο συνδυασμός των στηλών (CustomerID, OrderDate).

   Για την ανακάλυψη τέτοιου είδους εξαρτήσεων στο βιβλίο[AGNP18] αναφέρονται δύο κατηγορίες αλγορίθμων ανακάλυψης για μοναδικούς συνδυασμούς στηλών :

   - **Αλγόριθμοι βασισμένοι στις στήλες (column-based approaches)**

Αυτή η προσέγγιση αλγορίθμων επικεντρώνεται στην επιβεβαίωση μιας υποψήφιας μοναδικής στήλης κάθε φορά. Διαδοχικά ελέγχει κάθε στήλη για το αν αποτελεί μοναδικό συνδυασμό ή όχι. Χαρακτηριστικοί αλγόριθμοι είναι οι HCA(Hierarchical Column-based Algorithm), DUCC (Discovery of Unique Column Combinations) και Swan.

- **Αλγόριθμοι βασισμένοι στις γραμμές (row-based approaches)**

  Αυτή η προσέγγιση παράγει μοναδικούς συνδυασμούς στηλών από συγκρίσεις εγγραφών ζευγαριών. Επικεντρώνεται στην ανάλυση των επιμέρους εγγραφών και τη σύγκρισή τους μεταξύ τους για την εντοπισμό μοναδικών συνδυασμών. Χαρακτηριστικό παράδειγμα αλγορίθμου είναι ο GORDIAN.

2. Ανακάλυψη λειτουργικών εξαρτήσεων (FDs)

Η λειτουργία ανακάλυψης λειτουργικών εξαρτήσεων (Functional Dependencies - FDs) στοχεύει στον εντοπισμό σχέσεων μεταξύ των γνωρισμάτων ενός πίνακα. Αυτό σημαίνει ότι η λειτουργία αυτή προσπαθεί να αναγνωρίσει ποια γνωρίσματα εξαρτώνται λειτουργικά από άλλα, προσδιορίζοντας ποιες τιμές σε ένα σύνολο στηλών καθορίζουν λειτουργικά τις τιμές άλλων στηλών. Για παράδειγμα έστω ότι έχουμε το παρακάτω πίνακα:

Κωδικός Προϊόντος	Τιμή	Ποσότητα
001	1000	10
002	25000	20
003	300	30
004	20	40

Στον παραπάνω πίνακα, μπορούμε να παρατηρήσουμε τις εξής λειτουργικές εξαρτήσεις:

Τιμή -> Ποσότητα: Η τιμή ενός προϊόντος εξαρτάται από την ποσότητα του προϊόντος. Όσο μεγαλύτερη είναι η ποσότητα που αγοράζεται, τόσο χαμηλότερη είναι η τιμή ανά μονάδα. Αυτή η λειτουργική εξάρτηση μπορεί να εκφραστεί ως: {Τιμή} -> {Ποσότητα}.

Είναι σημαντικό να σημειωθεί ότι δεν υπάρχει λειτουργική εξάρτηση ανάποδα σε αυτήν την περίπτωση, δηλαδή η ποσότητα δεν εξαρτάται από την τιμή.

Βάσει του βιβλίου [AGNP18] η εύρεση λειτουργικών εξαρτήσεων στα δεδομένα απαιτεί τη χρήση ποικίλων αλγορίθμων, οι οποίοι μπορούν να ταξινομηθούν σε διάφορες κατηγορίες βάσει της στρατηγικής που χρησιμοποιούν.

Οι αλγόριθμοι διάσχισης πλέγματος (lattice traversal) βασίζονται σε διάσχιση πλέγματος στήλης και αναζητούν υποψήφιες εξαρτήσεις στην πλεγματική δομή των δεδομένων. Παραδείγματα αυτών των αλγορίθμων είναι οι Tane, Fun, Fd_Mine και Dfd.

Οι αλγόριθμοι συνόλων διαφοράς και συμφωνίας (difference- and agree-set algorithms) εξετάζουν σύνολα συμφωνίας και διαφοράς για την εύρεση λειτουργικών εξαρτήσεων. Ορισμένοι από αυτούς είναι οι Dep-Miner και FastFDs.

Οι αλγόριθμοι εισαγωγής εξαρτήσεων (dependency induction algorithms) συγκρίνουν εγγραφές δια ζευγών για να εντοπίσουν μη έγκυρες λειτουργικές εξαρτήσεις. Ένας γνωστός αλγόριθμος αυτής της κατηγορίας είναι ο Fdep.

Οι αλγόριθμοι υβριδικής προσέγγισης (hybrid algorithms) ενσωματώνουν ανεξάρτητες στρατηγικές εύρεσης εξαρτήσεων σε μία νέα στρατηγική, συνδυάζοντας τα πλεονεκτήματα διαφορετικών προσεγγίσεων. Ένας τέτοιος αλγόριθμος είναι ο HyFD.

Οι παράλληλοι αλγόριθμοι (parallel algorithms) βασίζονται στη μαζική παραλληλοποίηση για την αύξηση της απόδοσης κατά την ανακάλυψη λειτουργικών εξαρτήσεων, επιτρέποντας την επεξεργασία μεγάλων ποσοτήτων δεδομένων πιο αποτελεσματικά.

3. Ανακάλυψη εξαρτήσεων ενσωμάτωσης (INDs)

Οι εξαρτήσεις ενσωμάτωσης (INDs) αναφέρονται σε σχέσεις μεταξύ στηλών σε ένα σύνολο δεδομένων και καταδεικνύουν ότι όλες οι τιμές ή οι συνδυασμοί τιμών από ένα σύνολο στηλών εμφανίζονται και σε ένα διαφορετικό σύνολο στηλών.

Η ανακάλυψη εξαρτήσεων ενσωμάτωσης (INDs) στην διαδικασίας προφίλ δεδομένων αποσκοπεί στον εντοπισμό αυτών των σχέσεων μεταξύ στηλών σε ένα σύνολο δεδομένων. Όπως αναφέρεται και στο βιβλίο[AGNP18] συνήθως εξετάζουμε τις εξαρτήσεις ενσωμάτωσης μεταξύ διαφορετικών σχέσεων P_i και P_j, διότι αυτές οι εξαρτήσεις υποδηλώνουν τις σχέσεις των ξένων κλειδιών στο σχήμα μιας βάσης δεδομένων. Για παράδειγμα:

Πίνακας Παραγγελίες (Orders):

OrderID	ProductID	Quantity
1	101	2
2	102	1
3	103	3

Πίνακας Προϊόντα (Products):

ProductID	ProductName	Price
101	Μήλο	2.50
102	Πορτοκάλι	1.50
103	Μπανάνα	3.50

Εδώ, το πεδίο ProductID στον πίνακα Παραγγελίες εξαρτάται από το πεδίο ProductID στον πίνακα Προϊόντα. Κάθε τιμή του ProductID στον πίνακα Παραγγελίες υπάρχει στον πίνακα Προϊόντα.

Με βάση το βιβλίο [AGNP18] περιγράφονται δύο κύριες κατηγορίες αλγορίθμων για την ανακάλυψη Αλληλεξαρτήσεων Συμπερίληψης (INDs) όπως παρακάτω:

- **Αλγόριθμοι Επικύρωσης Υποψήφιων INDs**

  Οι αλγόριθμοι επικύρωσης υποψήφιων INDs περιλαμβάνουν τους B&B (Bell and Brockhausen, 1995), DeMarchi (Marchi et al., 2009), Binder (Papenbrock et al., 2015d), Spider (Bauckmann et al., 2006), S-IndD (Shaabani and Meinel, 2015) και Sindy (Kruse et al., 2015a). Αυτοί οι αλγόριθμοι εξετάζουν αποτελεσματικά την ορθότητα των πιθανών αλληλεξαρτήσεων, βασιζόμενοι σε στρατηγικές που ελαχιστοποιούν τις περιττές επιβεβαιώσεις και εκμεταλλεύονται την παράλληλη επεξεργασία για αποτελεσματικότερη ανίχνευση.

- **Αλγόριθμοι Διατριβής στο Πλέγμα**

  Οι αλγόριθμοι διατριβής στο πλέγμα περιλαμβάνουν τους Mind (Marchi et al., 2009), Find2 (Koeller and Rundensteiner, 2003), ZigZag (Marchi and Petit, 2003) και Mind2 (Shaabani and Meinel, 2016). Αυτοί οι αλγόριθμοι εφαρμόζουν αξιωματικούς κανόνες για την αποτελεσματική περικοπή του χώρου αναζήτησης των INDs, βασιζόμενοι σε στρατηγικές που επιλέγουν την

πιο κατάλληλη πορεία εξερεύνησης του πλέγματος για κάθε συγκεκριμένο σενάριο.

2.2.2 Τεχνολογικό Υπόβαθρο

Pythia

Η Pythia (Available : https://github.com/DAINTINESS-Group/Pythia) είναι ένα λογισμικό που αναπτύσσεται με σκοπό την δημιουργία ενός προφίλ δεδομένων δοθέντος ενός αρχείου που περιέχει τα δεδομένα σε μορφή πίνακα. Με βάση την θεωρητική ανάλυση το εργαλείο αυτό ακολουθεί την δομή και τις λειτουργίες περιεγραφήκαν.

Σχετικά με τις λειτουργίες-εργασίες της ανάλυση μίας στήλης το λογισμικό Pythia περιλαμβάνει:

- Υποσύστημα υπολογισμού κάποιων περιγραφικών στατιστικών όπως αριθμό των στοιχείων, τον μέσο όρο, τυπική απόκλιση , την διάμεσο των τιμών , την ελάχιστη και μέγιστη τιμή.
- Υποσύστημα παραγωγής ιστογραμμάτων.
- Υποσύστημα εντοπισμού αποκλίσεων (OutLiers)
- Υποσύστημα ομαδοποίησης δεδομένων (Clustering)

Σχετικά με τις λειτουργίες-εργασίες της πολλαπλών στηλών ανάλυσης περιλαμβάνει:

- Υποσύστημα εντοπισμού συσχετίσων (Correlations)
- Υποσύστημα παραγωγής δέντρου αποφάσεων (DecisionTree)
- Υποσύστημα υπολογισμού παλινδρόμησης(Regression)

Το λογισμικό αυτό επιπλέον είναι σε θέση να παράγει αναφορές με τα αποτέλεσμα της ανάλυσης του δοθέντος αρχείου.

Έως τώρα το λογισμικό είναι διαθέσιμο υπό την μορφή βιβλιοθήκης. Αυτό σημαίνει ότι δεν υπάρχει κάποια υλοποίηση γραφικής διεπαφής με το χρήστη και για να το χρησιμοποιήσει κάποιος χρειάζεται να διαθέτει προγραμματιστικές ικανότητες στην γλώσσα που είναι υλοποιημένο το σύστημα.

Spark

Το Apache Spark είναι μια ενιαία πλατφόρμα για τη επεξεργασία μεγάλων όγκων δεδομένων σε κατανεμημένα περιβάλλοντα, είτε σε τοπικούς υπολογιστές είτε στο νέφος. Η κύρια δύναμή του είναι η ικανότητα επεξεργασίας δεδομένων στη μνήμη, που το καθιστά πολύ ταχύτερο από το Hadoop MapReduce. Σε αντίθεση με το Hadoop, το οποίο αποθηκεύει τα ενδιάμεσα αποτελέσματα στον δίσκο μεταξύ των φάσεων map και reduce, το Spark διατηρεί τα δεδομένα στη μνήμη RAM, μειώνοντας σημαντικά το κόστος

εισόδου/εξόδου (I/O) και επιταχύνοντας την εκτέλεση. Επιπλέον, περιλαμβάνει βιβλιοθήκες με εύκολα διαθέσιμες διεπαφές για μηχανική μάθηση (MLlib), δια δραστικές ερωτήσεις SQL (Spark SQL), επεξεργασία ροών (Structured Streaming) για αλληλεπίδραση με δεδομένα πραγματικού χρόνου και επεξεργασία γράφων (GraphX) [DWDL20].

Spark RDD(Resilient Distributed Datasets)

Τα RDD είναι ένα από τα κύρια χαρακτηριστικά του Apache Spark. Κάθε εφαρμογή Spark αποτελείται από ένα πρόγραμμα οδήγησης (driver program) που εκτελεί την κύρια συνάρτηση του χρήστη και εκτελεί διάφορες παράλληλες λειτουργίες σε ένα σύνολο υπολογιστών (cluster). Το κύριο εργαλείο που παρέχει το Spark για τη διαχείριση των δεδομένων είναι τα ανθεκτικά κατανεμημένα σύνολα δεδομένων (RDD). Τα RDD είναι ουσιαστικά συλλογές δεδομένων που χωρίζονται σε τμήματα και κατανέμονται σε διάφορους υπολογιστές του συνόλου. Αυτά τα τμήματα μπορούν να επεξεργαστούν παράλληλα, προσφέροντας ταχύτητα και αποδοτικότητα στις λειτουργίες επεξεργασίας δεδομένων. Μπορούν να δημιουργηθούν από αρχεία στο σύστημα αρχείων Hadoop ή οποιοδήποτε άλλο συμβατό σύστημα αρχείων, αλλά και από υπάρχουσες συλλογές δεδομένων στο πρόγραμμα οδήγησης (driver program) που μετατρέπονται και μοιράζονται.

Το σημαντικότερο χαρακτηριστικό των RDD είναι η ανθεκτικότητά τους. Αυτό σημαίνει ότι αν ένας από τους υπολογιστές του συνόλου αποτύχει, το Spark μπορεί αυτόματα να ανακτήσει τα δεδομένα από τους υπόλοιπους υπολογιστές, εξασφαλίζοντας την ολοκλήρωση των λειτουργιών ακόμα και σε περιπτώσεις αποτυχίας. Με αυτόν τον τρόπο, τα RDD παρέχουν αξιοπιστία και αποτελεσματικότητα στην επεξεργασία δεδομένων σε μεγάλη κλίμακα [Apac21]. Ακολουθεί ένα μικρό απλοποιημένο παράδειγμα λειτουργίας:

```java
import org.apache.spark.api.java.*;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.function.Function;

import java.util.Arrays;
import java.util.List;

public class SimpleSparkExample {
    public static void main(String[] args) {

        // Set up Spark configuration
        // "local" means the program will run locally on your machine
        SparkConf conf = new SparkConf().setAppName("Simple Spark Example").setMaster("local"

        // Create a JavaSparkContext, which connects to the Spark cluster
        JavaSparkContext sc = new JavaSparkContext(conf);

        // Create a simple list of integers
        List<Integer> data = Arrays.asList(1, 2, 3, 4, 5);

        // Parallelize the list into an RDD (Resilient Distributed Dataset)
        // This distributes the data across Spark workers
        JavaRDD<Integer> distData = sc.parallelize(data);

        // Apply a transformation using map() to add 10 to each element of the RDD
        // map() applies the function to each element in the RDD
        JavaRDD<Integer> result = distData.map(new Function<Integer, Integer>() {
            @Override
            public Integer call(Integer number) throws Exception {
                return number + 10;  // Add 10 to each number
            }
        });

        // Collect the results from the RDD and print them
        // collect() gathers the results back to the local driver node
        System.out.println(result.collect());  // Output: [11, 12, 13, 14, 15]

        // Stop the SparkContext to release resources
        sc.stop();
    }
}
```

Εικόνα 4 Παράδειγμα κώδικα χρήσης RDD

Η μέθοδος sc.parallelize(data) χρησιμοποιείται για να μετατρέψει μια κανονική λίστα σε RDD (Resilient Distributed Dataset). Η μέθοδος map() εφαρμόζεται σε κάθε στοιχείο του RDD και δημιουργεί ένα νέο RDD. Η συνάρτηση που παρέχεται στο map() εκτελεί μια συγκεκριμένη ενέργεια για κάθε στοιχείο. Στο παράδειγμα, προστίθεται το 10 σε κάθε στοιχείο του RDD. Η μέθοδος collect() συγκεντρώνει τα δεδομένα του RDD και τα επιστρέφει στον τοπικό υπολογιστή (στην περίπτωση αυτή, στον driver). Τα δεδομένα επιστρέφονται ως κανονική λίστα, η οποία μπορεί να εκτυπωθεί. Στο παράδειγμα, τα δεδομένα εκτυπώνονται ως [11, 12, 13, 14, 15].

Spark DataFrame, Dataset and Sql

Το Spark DataFrame και το Dataset είναι δύο ισχυρές δομές δεδομένων που προσφέρει το Apache Spark για την αποθήκευση και επεξεργασία κατανεμημένων δεδομένων. Το DataFrame είναι ένας τύπος κατανεμημένου πίνακα που επιτρέπει την ανάλυση δεδομένων μέσω SQL queries και είναι ιδιαίτερα χρήσιμο για επεξεργασία με μεγάλα σύνολα δεδομένων. Από την άλλη, το Dataset παρέχει μεγαλύτερη αυστηρότητα στον

τύπο των δεδομένων, επιτρέποντας στους προγραμματιστές να εργάζονται με αντικείμενα τύπου Java [Apac21|17]. Αυτό σημαίνει ότι σε κάθε στήλη του Dataset θα αποθηκεύονται μόνο δεδομένα του συγκεκριμένου τύπου, όπως μόνο αριθμοί ή μόνο κείμενο. Αυτή η αυστηρότητα βοηθά στην αποφυγή λαθών κατά την εκτέλεση του προγράμματος. Στην πράξη, το Dataset μπορεί να μετατραπεί σε DataFrame και το αντίθετο, παρέχοντας στους προγραμματιστές ευελιξία στην επεξεργασία δεδομένων με ισχυρούς μετασχηματισμούς και βελτιστοποιημένες εκτελέσεις[Danc21]. Το Spark SQL είναι ένα υποσύστημα του Apache Spark που προσφέρει δυνατότητες επεξεργασίας δεδομένων σε μορφή δομημένων ερωτημάτων SQL. Βασιζόμενο σε αυτό, οι χρήστες μπορούν να αλληλοεπιδρούν με τα δεδομένα τους χρησιμοποιώντας παρόμοια με SQL ερωτήματα και λειτουργίες. Επιπρόσθετα, το Spark SQL επιτρέπει την εκτέλεση SQL ερωτημάτων σε πραγματικό χρόνο πάνω στα δεδομένα που βρίσκονται στο Spark, ενώ παρέχει και προηγμένες λειτουργίες για την επεξεργασία, την ανάλυση, και την ανάκτηση δεδομένων [Apac21|17]. Τέλος, το Spark SQL είναι συμβατό με το πρότυπο ANSI SQL:2003, και μπορεί να λειτουργήσει ως ένας αυτόνομος κινητήρας SQL για την εκτέλεση πλήρων ερωτημάτων SQL [DWDL20]. Ακολουθεί ένα παράδειγμα δημιουργίας dataFrame, dataset από υπάρχων Rdd:

```java
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import org.apache.spark.api.java.function.FilterFunction;
import org.apache.spark.sql.*;
import org.apache.spark.sql.types.*;
import java.util.Arrays;
import java.util.List;

public class RDDToDataFrameAndDatasetExample {

    public static void main(String[] args) {
        // Δημιουργία SparkConf και JavaSparkContext
        SparkConf conf = new SparkConf().setAppName("RDD to DataFrame and Dataset Example").setMaster("local");
        JavaSparkContext sc = new JavaSparkContext(conf);
        // Δημιουργία SparkSession
        SparkSession spark = SparkSession.builder()
                .appName("RDD to DataFrame and Dataset")
                .master("local")
                .getOrCreate();
        // Δημιουργία λίστας με αριθμούς
        List<Integer> data = Arrays.asList(1, 2, 3, 4, 5);
        // Δημιουργία RDD από τη λίστα
        JavaRDD<Integer> rdd = sc.parallelize(data);
        // --- Δημιουργία DataFrame από RDD ---
        // Δημιουργία RDD από Row (με τη χρήση map για τη μετατροπή σε Row)
        JavaRDD<Row> rowRDD = rdd.map(num -> RowFactory.create(num));
        // Ορισμός schema για το DataFrame
        StructType schema = new StructType(new StructField[]{
                new StructField( name: "Number", DataTypes.IntegerType,  nullable: false, Metadata.empty())
        });
        // Δημιουργία DataFrame από το RowRDD και το schema
        Dataset<Row> df = spark.createDataFrame(rowRDD, schema);
        // Εμφάνιση του DataFrame
        System.out.println("DataFrame:");
        df.show();
        // Εκτέλεση SQL query στο DataFrame
        df.createOrReplaceTempView( viewName: "numbers");
        Dataset<Row> resultDF = spark.sql( sqlText: "SELECT * FROM numbers WHERE Number > 2");
        System.out.println("SQL Query on DataFrame:");
        resultDF.show();
        // --- Δημιουργία Dataset από DataFrame ---
        // Μετατροπή του DataFrame σε Dataset<Integer>
        Dataset<Integer> dataset = df.select( col: "Number").as(Encoders.INT());
        // Εμφάνιση του Dataset
        System.out.println("Dataset:");
        dataset.show();
        // Χρήση Dataset API για φιλτράρισμα
        Dataset<Integer> filteredDataset = dataset.filter((FilterFunction<Integer>) num -> num > 2);
        System.out.println("Filtered Dataset (using Dataset API):");
        filteredDataset.show();
        // Τερματισμός του SparkContext
        sc.stop();
    }
}
```

Εικόνα 5 Παράδειγμα κώδικα χρήσης dataFrame, dataset, Sql.

Στον παραπάνω κώδικα, δημιουργούμε ένα DataFrame και ένα Dataset από ένα υπάρχον RDD με τη χρήση του Apache Spark. Αρχικά, από το RDD δημιουργούμε ένα DataFrame μέσω του RowRDD και ενός schema που ορίζει τη στήλη "Number" ως τύπο Integer. Στη συνέχεια, πραγματοποιούμε εκτέλεση SQL query πάνω στο DataFrame για να φιλτράρουμε τις τιμές του αριθμού που είναι μεγαλύτερες από 2. Από το DataFrame, μετατρέπουμε τα δεδομένα σε Dataset<Integer> χρησιμοποιώντας την μέθοδο select().as(Encoders.INT()). Στο Dataset, χρησιμοποιούμε το API του Dataset για φιλτράρισμα των δεδομένων με βάση μια συνθήκη (όπου ο αριθμός είναι μεγαλύτερος από 2).

MLlib

Η MLlib είναι η βιβλιοθήκη μηχανικής μάθησης (ML) του Apache Spark, η οποία στοχεύει στο να καταστήσει την πρακτική μηχανική μάθηση πιο επεκτάσιμη και εύκολη στη χρήση. Αυτή η βιβλιοθήκη παρέχει ένα ευρύ φάσμα εργαλείων και αλγορίθμων ML για την εκπαίδευση, την αξιολόγηση και την εφαρμογή μοντέλων μηχανικής μάθησης σε μεγάλα σύνολα δεδομένων. Με την επικέντρωσή της στην απόδοση και την ευκολία χρήσης, η MLlib περιλαμβάνει αλγόριθμους για ταξινόμηση, παλινδρόμηση, ομαδοποίηση, συνεργατικό φιλτράρισμα και άλλες δημοφιλείς εργασίες μηχανικής μάθησης, καθώς και εργαλεία για τη διαχείριση χαρακτηριστικών, την κατασκευή σωλήνων επεξεργασίας και την αποθήκευση των μοντέλων και των πιθανοτήτων[DWDL20]. Ακολουθεί παρακάτω ένα παράδειγμα:

```java
import java.util.Arrays;

import org.apache.spark.api.java.JavaDoubleRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.linalg.Matrix;
import org.apache.spark.mllib.linalg.Vector;
import org.apache.spark.mllib.linalg.Vectors;
import org.apache.spark.mllib.stat.Statistics;

JavaDoubleRDD seriesX = jsc.parallelizeDoubles(
  Arrays.asList(1.0, 2.0, 3.0, 3.0, 5.0));  // a series

// must have the same number of partitions and cardinality as seriesX
JavaDoubleRDD seriesY = jsc.parallelizeDoubles(
  Arrays.asList(11.0, 22.0, 33.0, 33.0, 555.0));

// compute the correlation using Pearson's method. Enter "spearman" for Spearman's method.
// If a method is not specified, Pearson's method will be used by default.
double correlation = Statistics.corr(seriesX.srdd(), seriesY.srdd(), "pearson");
System.out.println("Correlation is: " + correlation);

// note that each Vector is a row and not a column
JavaRDD<Vector> data = jsc.parallelize(
  Arrays.asList(
    Vectors.dense(1.0, 10.0, 100.0),
    Vectors.dense(2.0, 20.0, 200.0),
    Vectors.dense(5.0, 33.0, 366.0)
  )
);

// calculate the correlation matrix using Pearson's method.
// Use "spearman" for Spearman's method.
// If a method is not specified, Pearson's method will be used by default.
Matrix correlMatrix = Statistics.corr(data.rdd(), "pearson");
System.out.println(correlMatrix.toString());
```

*Εικόνα 6 Παράδειγμα χρήσης κώδικα από **Spark Documentation***

Το παραπάνω πρόγραμμα δείχνει πώς μπορεί να υπολογιστεί συσχέτιση και πίνακα συσχέτισης χρησιμοποιώντας την MLlib βιβλιοθήκη του Apache Spark στην Java. Αρχικά, δημιουργούμε δύο σειρές δεδομένων (seriesX και seriesY) και υπολογίζουμε τη συσχέτιση μεταξύ τους χρησιμοποιώντας τη μέθοδο Pearson. Στη συνέχεια, δημιουργούμε ένα RDD με πολλαπλά χαρακτηριστικά (στήλες) και υπολογίζουμε τον πίνακα συσχέτισης για αυτά τα χαρακτηριστικά.

GraphX

Το GraphX αποτελεί ένα νέο στοιχείο του Apache Spark, σχεδιασμένο ειδικά για τη διαχείριση γράφων και την εκτέλεση παράλληλων υπολογισμών πάνω σε αυτούς. Σε γενικές γραμμές, το GraphX επεκτείνει τη λειτουργικότητα του Spark RDD εισάγοντας ένα νέο αφαιρετικό αντικείμενο γράφου: έναν κατευθυνόμενο πολυγράφο με ιδιότητες που συνδέονται με κάθε κορυφή και ακμή [DWDL20]. Για να υποστηρίξει τους υπολογισμούς πάνω σε γράφους, το GraphX παρέχει ένα σύνολο βασικών τελεστών όπως οι subgraph, joinVertices, και aggregateMessages, καθώς και μια βελτιστοποιημένη εκδοχή του API Pregel. Επιπλέον, το GraphX περιλαμβάνει μια εκτεταμένη συλλογή αλγορίθμων γράφων και εργαλεία για την εύκολη ανάλυση και την κατασκευή γράφων [Apac21|19]. Ακολουθεί ένα παράδειγμα παρακάτω:

```
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
package info.debatty.spark;

import java.util.LinkedList;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.graphx.Edge;
import org.apache.spark.graphx.Graph;
import org.apache.spark.rdd.RDD;
import org.apache.spark.storage.StorageLevel;
import scala.Tuple2;
import scala.reflect.ClassTag;

/**
 *
 * @author tibo
 */
public class GraphXTest extends SparkCase {
    public void testGraphX() {
        JavaSparkContext spark = getSpark();


        LinkedList<Tuple2<Object, String>> nodes =
                new LinkedList<>();
        nodes.add(new Tuple2<>((Object) 0L, "0"));
        nodes.add(new Tuple2<>((Object) 1L, "1"));
        RDD<Tuple2<Object, String>> nodes_rdd = spark.parallelize(nodes).rdd();

        LinkedList<Edge<String>> edges = new LinkedList<>();
        edges.add(new Edge<>(0L, 1L, ""));
        RDD<Edge<String>> edges_rdd = spark.parallelize(edges).rdd();

        String default_node = "Default";

        StorageLevel storage1 = StorageLevel.MEMORY_ONLY();
        StorageLevel storage2 = StorageLevel.MEMORY_ONLY();

        ClassTag<String> string_class_tag =
                scala.reflect.ClassManifestFactory.fromClass(String.class);

        Graph<String, String> graph = Graph.apply(
                nodes_rdd,
                edges_rdd,
                default_node,
                storage1,
                storage2,
                string_class_tag,
                string_class_tag);

        System.out.println(graph.edges().collect());

        Graph<Object, String> connected_components =
                graph.ops().connectedComponents();
        System.out.println(connected_components.vertices().collect());
    }
}
```

*Εικόνα 7 Παράδειγμα χρήσης κώδικα σε αποθετήριο στο **github**.*

Ο κώδικας δημιουργεί έναν γράφημα με δύο κορυφές και μία ακμή χρησιμοποιώντας το GraphX σε Java. Δημιουργούνται τα RDD για τις κορυφές και τις ακμές και στη συνέχεια δημιουργείται το γράφημα με τη μέθοδο Graph.apply. Μετά, εκτελείται ο αλγόριθμος connectedComponents για να βρεθούν οι συνδεδεμένες ακμές του γράφου. Ο αλγόριθμος

επιστρέφει ένα νέο γράφημα με τις κορυφές που συνδέονται μεταξύ τους με ακμές, και εκτυπώνονται τα αποτελέσματα, όπως οι ακμές του γράφου και οι συνδεδεμένες ακμές. Το αποτέλεσμα δείχνει ότι οι κορυφές 0 και 1 είναι συνδεδεμένες μέσω μίας ακμής και ανήκουν στην ίδια ομάδα συνδεδεμένων ακμών.

Sparksession

Το SparkSession αποτελεί κρίσιμο στοιχείο στο πλαίσιο της ανάπτυξης εφαρμογών στο Apache Spark. Αντί να αναγκαστεί ο προγραμματιστής να διαχωρίζει τις λειτουργίες της εφαρμογής σε διάφορα περιβάλλοντα, όπως το SparkContext και το SQLContext, το SparkSession επιτρέπει την ολοκληρωμένη διαχείριση τους μέσω ενός ενιαίου αντικειμένου. Μέσω του SparkSession, ο προγραμματιστής μπορεί να δημιουργήσει DataFrames, να εκτελέσει ερωτήματα SQL και να διαχειριστεί παραμέτρους εκτέλεσης, παρέχοντας ένα ολοκληρωμένο περιβάλλον ανάπτυξης για την ανάπτυξη εφαρμογών Spark. Η ενοποίηση των λειτουργιών αυτών σε ένα σημείο εισόδου απλοποιεί την ανάπτυξη και τη διαχείριση των εφαρμογών Spark, επιτρέποντας στους προγραμματιστές να επικεντρωθούν στην υλοποίηση της λογικής της εφαρμογής τους [DWDL20].

Java

Η Java έχει αναδειχθεί ως μια από τις πιο δημοφιλείς και ευέλικτες γλώσσες προγραμματισμού στον κόσμο, χάρη στην αξιοσημείωτη συνδυασμένη δύναμη της αντικειμενοστρεφούς αρχιτεκτονικής της και της ευκολίας χρήσης της. Με την ικανότητά της να προσφέρει μια πλατφόρμα ανεξάρτητη από το λειτουργικό σύστημα εκτέλεσης, η Java έχει καταφέρει να αποτελέσει τη βάση για πολλούς τομείς της πληροφορικής, από τις διαδικτυακές εφαρμογές έως τις επιχειρηματικές λύσεις και τις εφαρμογές κινητών συσκευών.

Ένα από τα κύρια χαρακτηριστικά που καθιστούν τη Java τόσο δημοφιλή είναι η Εικονική Μηχανή Java (JVM), που επιτρέπει τη μεταφορά του κώδικα Java σε διαφορετικά συστήματα χωρίς την ανάγκη επανασυγγραφής του. Η JVM μεταφράζει τον κώδικα σε ένα ενδιάμεσο μορφής bytecode, ο οποίος εκτελείται στη συνέχεια σε κάθε συσκευή που διαθέτει ένα αντίστοιχο JVM.

Ένα άλλο σημαντικό χαρακτηριστικό είναι η αυτόματη διαχείριση μνήμης που προσφέρει η Java με τη συλλογή σκουπιδιών. Αυτό ελαχιστοποιεί την ανάγκη για χειροκίνητη διαχείριση μνήμης από τους προγραμματιστές και διασφαλίζει την αποτελεσματική χρήση των πόρων του συστήματος.

Επιπλέον, η Java υποστηρίζει τον αντικειμενοστρεφή προγραμματισμό, που επιτρέπει τη δημιουργία πολύπλοκων εφαρμογών με οργανωμένο τρόπο και επαναχρησιμοποίηση

κώδικα. Οι έννοιες όπως η κληρονομικότητα, η ενθυλάκωση και ο πολυμορφισμός επιτρέπουν τη δημιουργία ευέλικτων και επεκτάσιμων εφαρμογών.

Τέλος, η Java προσφέρει εκτεταμένο οικοσύστημα βιβλιοθηκών και εργαλείων που καλύπτουν μια ευρεία γκάμα αναγκών ανάπτυξης εφαρμογών. Από βιβλιοθήκες για γραφικά και δικτύωση έως βάσεις δεδομένων και επεξεργασία φυσικής γλώσσας, η Java παρέχει όλα τα εργαλεία που απαιτούνται για τη δημιουργία πολύπλοκων και αξιόπιστων εφαρμογών [Das22].

Maven

Το Maven είναι ένα εργαλείο διαχείρισης λογισμικού και αυτοματοποιημένης κατασκευής αυτού που χρησιμοποιείται κυρίως στη Java και άλλες γλώσσες προγραμματισμού. Το Maven διευκολύνει τον προγραμματιστή, καθώς αναλαμβάνει τη διαχείριση εξαρτήσεων, την κατασκευή και τη δημιουργία της τελικής εφαρμογής χωρίς να χρειάζεται να ασχολείται με τις βιβλιοθήκες ή άλλες τεχνικές λεπτομέρειες. Όταν αναπτύσσεται μια εφαρμογή με το Maven, δημιουργείται ένα αρχείο pom.xml (Project Object Model), το οποίο περιέχει όλες τις ρυθμίσεις για την κατασκευή της, όπως εξαρτήσεις (dependencies), πηγές, στόχους κατασκευής και πλατφόρμες εκτέλεσης. Μέσω του αρχείου αυτού, το Maven διασφαλίζει ότι όλες οι απαιτούμενες βιβλιοθήκες είναι διαθέσιμες και ότι η εφαρμογή κατασκευάζεται με τον σωστό τρόπο. Επιπλέον, το Maven παρέχει έναν μηχανισμό για την εκτέλεση αυτοματοποιημένων test και την αποστολή της εφαρμογής σε απομακρυσμένα αποθετήρια (repositories), όπου μπορεί να αναζητήσει και να κατεβάσει εξαρτήσεις από άλλους χρήστες ή βιβλιοθήκες τρίτων. Αυτό καθιστά τη διαδικασία ανάπτυξης πιο γρήγορη, αξιόπιστη και συνεπή, επιτρέποντας στους προγραμματιστές να εστιάσουν περισσότερο στον κώδικα και λιγότερο στην υποδομή [ApMa |22].

Git

Το Git είναι ένα κατανεμημένο σύστημα ελέγχου εκδόσεων (dVCS), που χρησιμοποιείται για την παρακολούθηση και τον έλεγχο των αλλαγών σε αρχεία και έργα λογισμικού. Το Git επιτρέπει σε ομάδες προγραμματιστών και σε ατομικούς χρήστες να εργαστούν ταυτόχρονα σε διαφορετικές εκδόσεις του ίδιου έργου, να παρακολουθούν τις αλλαγές που γίνονται και να ενσωματώνουν τις αλλαγές από διάφορες πηγές. Με το Git, οι χρήστες μπορούν να δημιουργούν αντίγραφα ασφαλείας του κώδικα τους, να δοκιμάζουν νέες λειτουργίες σε ξεχωριστές "κλάδους" (branches) και να συγχωνεύουν τις αλλαγές τους με την κύρια έκδοση του έργου. Με αυτόν τον τρόπο, το Git επιτρέπει την αποτελεσματική διαχείριση του κώδικα και τη συνεργασία μεταξύ των μελών μιας ομάδας ανάπτυξης λογισμικού [Wors23].

2.3 Ανάλυση απαιτήσεων

Η παρούσα υπό ενότητα παρουσιάζει μια σειρά από user stories που αποτελούν μέρος της επέκτασης του ήδη υπάρχοντος εργαλείου για την αυτόματη παραγωγή στατιστικών προφίλ για αλφαριθμητικά σύνολα δεδομένων. Τα παρακάτω user stories προσδιορίζουν τις λειτουργικές απαιτήσεις που αναμένεται να επεκτείνουν την υπάρχουσα λειτουργικότητα του συστήματος, δημιουργώντας ένα πιο ολοκληρωμένο και ευέλικτο εργαλείο ανάλυσης δεδομένων. Με την υλοποίηση αυτών των user stories, η αναμενόμενη επέκταση του συστήματος θα παρέχει στους χρήστες μια ακόμη πιο αποτελεσματική και εύχρηστη εμπειρία δημιουργίας προφίλ δεδομένων.

Εδώ είναι οι λειτουργικές απαιτήσεις υπό μορφή user stories:

- [UC1] Ως χρήστης, θέλω να εκτιμώ ημιαυτόματα τον τύπο των στηλών ενός dataset, βασιζόμενος σε δείγματα από τα δεδομένα.
- [UC2] Ως χρήστης, θέλω να βλέπω πληροφορίες για κάθε dataset, συμπεριλαμβανομένου του αριθμού των γραμμών, του μεγέθους του αρχείου και της ημερομηνίας δημιουργίας του προφίλ.
- [UC3] Ως χρήστης, θέλω να βλέπω λεπτομερείς αναφορές για κάθε στήλη του datatset, περιλαμβάνοντας τον αριθμό των null τιμών, τον αριθμό των διακριτών τιμών και mode τιμών.
- [UC4] Ως χρήστης θέλω να βλέπω αναφορές που περιλαμβάνουν τα τεταρτημόρια q1,q2,q3 και ισοϋψές ιστόγραμμα με την μορφή quartiles.
- [UC4] Ως χρήστης, θέλω να έχω μια φιλική προς τον χρήστη γραφική διεπαφή που θα επιτρέπει τη διαχείριση των δεδομένων και την προβολή αναφορών.

Κεφάλαιο 3. Σχεδίαση & Υλοποίηση

3.1 Ορισμός προβλήματος και αλγόριθμοι επίλυσης

3.1.1 Ορισμός Προβλήματος

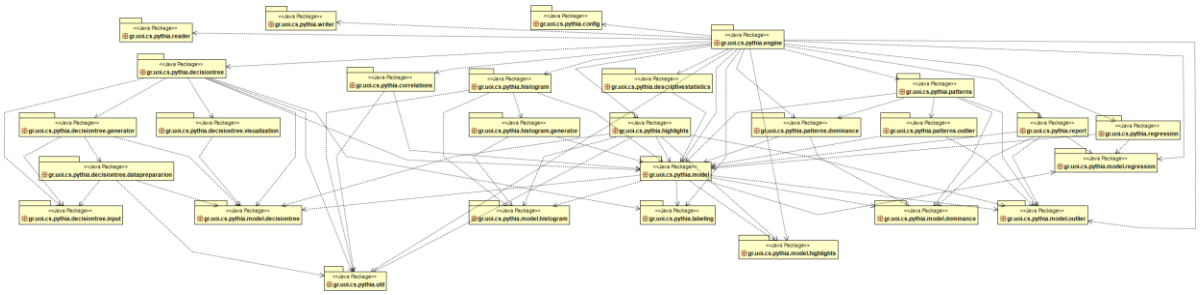
Όπως έχει ήδη αναφερθεί το λογισμικό Pythia είναι ένα ήδη υπάρχων λογισμικό υπό-ανάπτυξη που στόχο έχει την δημιουργία ενός στατικού αυτόματου προφίλ δοθέντος ενός συνόλου δεδομένων. Ωστόσο όπως προκύπτει από την ανάλυση στο κεφάλαιο 2 και την υπάρχουσα δομή και λειτουργεία του, το λογισμικό χρειάζεται να επεκταθεί κατάλληλα ώστε καλύπτει όλες τις πληροφορίες που χρειάζονται για την επιτυχή δημιουργία του προφίλ δεδομένων. Αναλυτικότερα :

- Τίθεται ανάγκη για μια λειτουργία ημιαυτόματης εκτίμησης του τύπου των στηλών των δεδομένων αυτοματοποιημένα από το σύστημα, με το οποίο ο χρήστης θα αποφασίζει πώς θα χαρακτηρίσει τους τύπους κάθε στήλης.
- Χρειάζεται επέκταση που αφορά την καταγραφή χαρακτηριστικών που αφορούν ολόκληρο το σετ δεδομένων όπως ο αριθμός των γραμμών, ολόκληρη η διαδρομή του αρχείου που φορτώνεται στο σύστημα (filepath), το μέγεθος του αρχείου που φορτώνεται, καθώς και την αναλυτική ημερομηνία και ώρα δημιουργίας του προφίλ δεδομένων.
- Χρειάζεται επέκταση που αφορά την καταγραφή κρίσιμων χαρακτηριστικών όπως των αριθμό των null τιμών για κάθε στήλη, τον αριθμό διακριτών τιμών(distinct value), τις τιμές με την μεγαλύτερη συχνότητα εμφάνισης σε μία στήλη(mode), καθώς και την δημιουργία ισοϋψές ιστόγραμμα με τη μορφή quartiles.
- Τροποποίηση των υπαρχόντων αναφορών που παράγει το λογισμικό με τις παραπάνω επεκτάσεις.
- Ανακατασκευή και νέος σχεδιασμός στο πακέτο Outliers καθώς συμπεριλαμβάνεται σε λανθασμένο πακέτο και δεν είναι ξεκάθαρη η λειτουργία του.

– Δημιουργία μια γραφικής διαπροσωπείας με το χρήστη, εφόσον μέχρι στιγμής απευθύνεται σε χρήστες με προγραμματιστικές ικανότητες.

3.1.2 Περιγραφή Υπάρχουσας Υλοποίησης Και Δομών Συστήματος

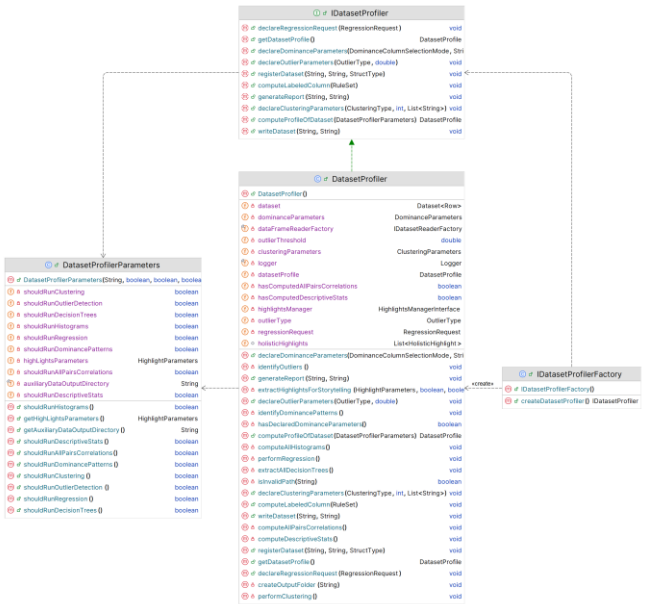
Στην ενότητα αυτή περιγράφεται η υπάρχουσα δομή του λογισμικού και η σύντομη επεξήγηση αυτών. Η Pythia λοιπόν αποτελείται :



Εικόνα 8 packages uml diagram

Κάθε πακέτο περιλαμβάνει απαραίτητα υποσυστήματα υπολογισμού με το πακέτο engine να αποτελεί την κύρια μηχανή οργάνωσης και λειτουργίας του συστήματος.

Αναλυτικότερα το πακέτο **engine:**



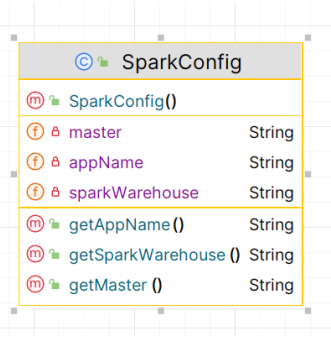
Εικόνα 9engine package uml diagram

Το πακέτο περιλαμβάνει τις βασικές κλάσεις που αναλαμβάνουν τη λειτουργία του εργαλείου. Υπάρχει ένα κεντρικό interface με την ονομασία IDatasetProfiler, το οποίο

καθορίζει τις βασικές λειτουργίες του API. Η κλάση DatasetProfiler υλοποιεί αυτήν τη διεπαφή και τις αντίστοιχες λειτουργίες. Αυτές οι λειτουργίες καλούν τα αντίστοιχα υποσυστήματα στα διάφορα πακέτα προκειμένου να εκτελέσουν τις αντίστοιχες εργασίες. Έως τώρα, οι υπάρχοντες clients χρησιμοποιούν τις κλήσεις-λειτουργίες που προσφέρει αυτή η διεπαφή.

Η κλάση DatasetProfilerParameters έχει τον βοηθητικό ρόλο ως προς τους clients της DatasetProfiler. Χρησιμοποιείται με σκοπό να ορίσει ποιες λειτουργίες υπολογισμού στατιστικών θα εκτελεστούν .

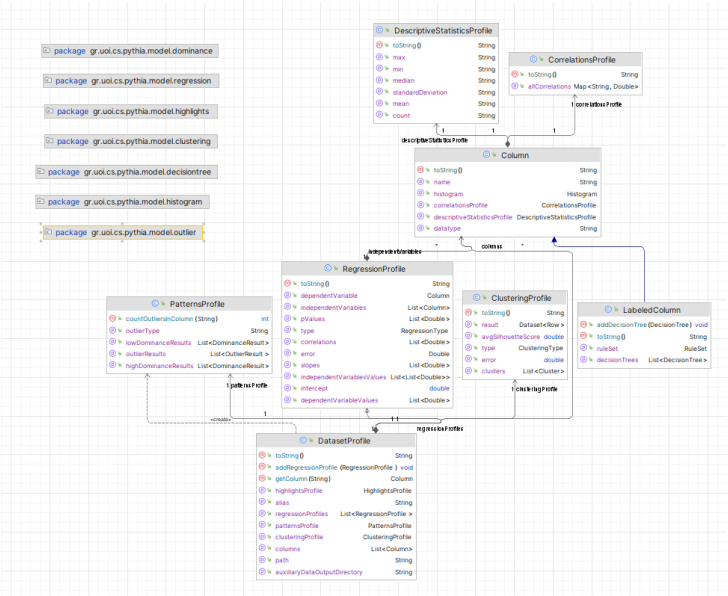
Πακέτο **config:**



Εικόνα 10 config package uml/ class diagram

Το πακέτο αυτό περιέχει την κλάση SparkConfig, η οποία αναλαμβάνει τη διαχείριση των ρυθμίσεων σύνδεσης και διαμόρφωσης του Apache Spark. Η κλάση αυτή φορτώνει τις ρυθμίσεις από το αρχείο spark.properties. Στη συνέχεια, παρέχει μεθόδους για την ανάκτηση των τιμών των παραμέτρων master, appName, και sparkWarehouse, οι οποίες χρησιμοποιούνται από τον datasetProfiler για τη διαμόρφωση της εκτέλεσης εφαρμογών Spark. Το αρχείο spark.properties περιέχει τις ρυθμίσεις σύνδεσης και διαμόρφωσης για το Apache Spark. Αυτές οι ρυθμίσεις καθορίζουν το URI του master node (spark.master.uri) ως local[*], που υποδηλώνει τη χρήση του τοπικού cluster με όλους τους διαθέσιμους πυρήνες του υπολογιστή, καθώς και το όνομα της εφαρμογής (spark.app.name) ως "Pythia".

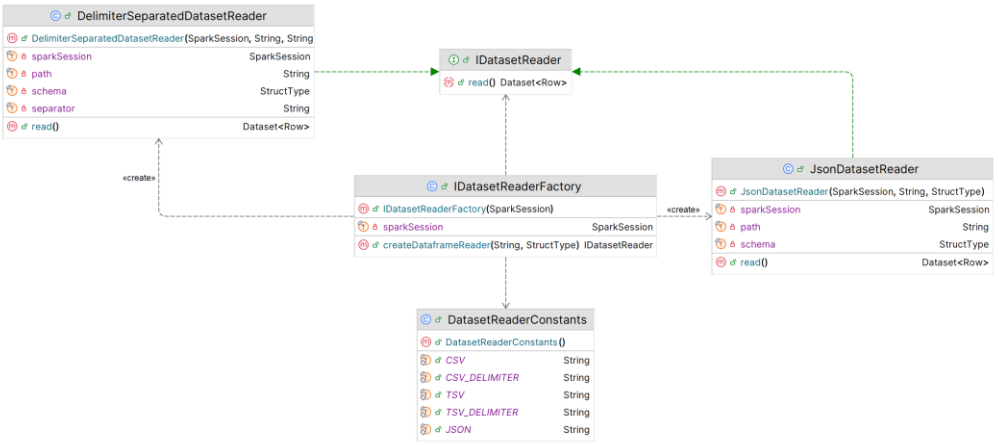
Το πακέτο **model:**



Εικόνα 11 model package uml diagram

Αυτό το πακέτο αποτελεί τον πυρήνα του μοντέλου του λογισμικού και οι κλάσεις παρέχουν μια αφαιρετική προσέγγιση των δεδομένων και των λειτουργιών που σχετίζονται με αυτά. Μερικές από τις βασικές κλάσεις στο πακέτο είναι η DatasetProfile, η οποία αναπαριστά ένα προφίλ για ένα σύνολο δεδομένων, συμπεριλαμβανομένων των στηλών του, προφίλ προτύπων, προφίλ παλινδρόμησης και προφίλ ομαδοποίησης. Η Column αναπαριστά μια στήλη δεδομένων μέσα σε ένα σύνολο δεδομένων, συμπεριλαμβανομένων των χαρακτηριστικών της όπως το όνομα, ο τύπος δεδομένων και διάφορες άλλες πληροφορίες. Η PatternsProfile αναπαριστά ένα προφίλ για τα πρότυπα που εντοπίζονται στα δεδομένα. Η RegressionProfile αναπαριστά ένα προφίλ παλινδρόμησης για μια στήλη δεδομένων, ενώ η ClusteringProfile αναπαριστά ένα προφίλ ομαδοποίησης για τα δεδομένα. Στο διάγραμμα της εικόνας διακρίνονται επίσης υποπακέτα που είτε έχουν κλάσεις τύπου ResultProfile, όπως για παράδειγμα τα OutlierResult, DominanceResult, είτε περιέχουν σχετικά Enums, τα οποία καθορίζουν τους διαφορετικούς αλγορίθμους υπολογισμού μιας συγκεκριμένης κατηγορίας εργασιών που υποστηρίζει το πρόγραμμα. Για παράδειγμα, κάποιες από τις κλάσεις είναι οι RegressionType, ClusteringType, OutlierType.

Το πακέτο **reader:**

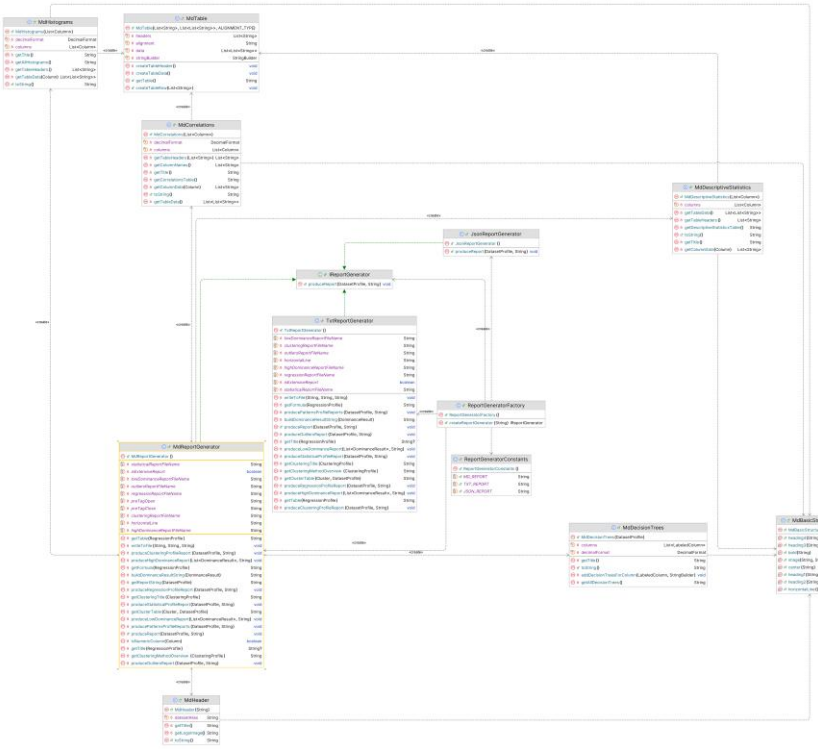


Εικόνα 12 reader package uml diagram

Το πακέτο περιλαμβάνει ένα σύνολο κλάσεων και διεπαφών που αναλαμβάνουν τη διαχείριση της ανάγνωσης δεδομένων στην εφαρμογή, επιτρέποντας τη φόρτωση δεδομένων από αρχεία κειμένου. Η διεπαφή IDatasetReader ορίζει μια μέθοδο read(), η οποία πρέπει να υλοποιείται από τις κλάσεις που διαβάζουν τα δεδομένα. Κάθε υλοποίηση αυτής της διεπαφής αντιστοιχεί σε έναν συγκεκριμένο τύπο αρχείου, όπως CSV, TSV ή JSON. Η κλάση IDatasetReaderFactory αναλαμβάνει τη δημιουργία των κατάλληλων αντικειμένων IDatasetReader ανάλογα με τον τύπο του αρχείου που πρόκειται να διαβαστεί.

Οι κλάσεις JsonDatasetReader και DelimiterSeparatedDatasetReader είναι υλοποιήσεις της διεπαφής IDatasetReader και αναλαμβάνουν τη διαχείριση της ανάγνωσης δεδομένων από αρχεία JSON και από αρχεία με διαχωριστικό χαρακτήρα, όπως CSV ή TSV, αντίστοιχα. Τέλος, η κλάση DatasetReaderConstants παρέχει σταθερές που περιγράφουν τους διάφορους τύπους αρχείων που υποστηρίζονται από το σύστημα ανάγνωσης δεδομένων, καθώς και τους αντίστοιχους χαρακτήρες διαχωρισμού για αρχεία CSV και TSV.

Το πακέτο **report:**



Εικόνα 13 report package uml diagram

Το πακέτο περιλαμβάνει τις κλάσεις που αναλαμβάνουν τη δημιουργία αναφορών για τα στατιστικά και τα αποτελέσματα ανάλυσης των δεδομένων. Η διεπαφή IReportGenerator ορίζει τη μέθοδο produceReport(), η οποία παράγει την αναφορά για ένα προφίλ συνόλου δεδομένων και τοποθετεί το αποτέλεσμα σε έναν καθορισμένο κατάλογο εξόδου. Οι κλάσεις JsonReportGenerator, TxtReportGenerator και MdReportGenerator αναλαμβάνουν τη δημιουργία αναφορών σε μορφή Json, κειμένου και Markdown αντίστοιχα. Κάθε μία από αυτές τις κλάσεις υλοποιεί τη μέθοδο produceReport() της διεπαφής IReportGenerator, διαχειριζόμενη τη δημιουργία και τη μορφοποίηση της αναφοράς σύμφωνα με τον συγκεκριμένο τύπο που αντιστοιχεί σε κάθε κλάση. Η κλάση ReportGeneratorFactory αναλαμβάνει τη δημιουργία του κατάλληλου αντικειμένου για τη δημιουργία αναφοράς, με βάση τον τύπο αναφοράς που πρέπει να παραχθεί. Χρησιμοποιείται για τη διευκόλυνση της δημιουργίας των αντικειμένων για την παραγωγή αναφορών, επιτρέποντας την επιλογή του σωστού αλγορίθμου ανάλογα με τον τύπο της αναφοράς που επιθυμούμε.

Το πακέτο **clustering:**

Εικόνα 14 clustering package uml diagram

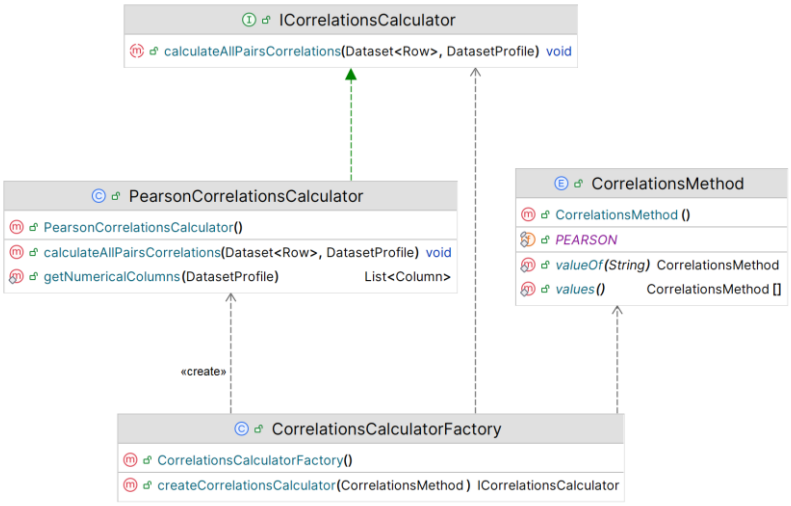
Το πακέτο περιλαμβάνει μια συλλογή κλάσεων και διεπαφών που έχουν σχεδιαστεί ειδικά για την εκτέλεση αλγορίθμων ομαδοποίησης πάνω σε σύνολα δεδομένων. Η διεπαφή IClusteringPerformer έχει σχεδιαστεί για να παρέχει ένα πρότυπο εκτέλεσης αλγορίθμων ομαδοποίησης. Κάθε κλάση που την υλοποιεί περιλαμβάνει τη μέθοδο performClustering, η οποία αναλαμβάνει την εκτέλεση του εκάστοτε αλγορίθμου ομαδοποίησης πάνω σε ένα δεδομένο σύνολο και επιστρέφει ένα προφίλ ομαδοποίησης.

Το πακέτο περιέχει κλάσεις όπως οι KmeansClusteringPerformer, DivisiveClusteringPerformer και GraphBasedClusteringPerformer, οι οποίες υλοποιούν διάφορους αλγορίθμους ομαδοποίησης, όπως ο K-Means, ο αλγόριθμος Divisive και ο Graph-Based αλγόριθμος. Κάθε μία από αυτές τις κλάσεις παρέχει τις απαραίτητες μεθόδους για την εκτέλεση του αντίστοιχου αλγορίθμου ομαδοποίησης.

Επιπλέον, περιλαμβάνει τις κλάσεις ClusteringParameters και Cluster, οι οποίες χρησιμοποιούνται για τη διαχείριση και την παραμετροποίηση των αλγορίθμων ομαδοποίησης, καθώς και για τη δημιουργία προφίλ ομαδοποίησης που περιλαμβάνουν τα αποτελέσματα της διαδικασίας.

Τέλος, η αφηρημένη κλάση GeneralClusteringPerformer έχει σχεδιαστεί με στόχο την αποφυγή επαναλαμβανόμενου κώδικα στις κλάσεις που είναι υπεύθυνες για την υλοποίηση συγκεκριμένων αλγορίθμων ομαδοποίησης, προσφέροντας κοινή λειτουργικότητα και διευκολύνοντας την ανάπτυξη νέων υλοποιήσεων.

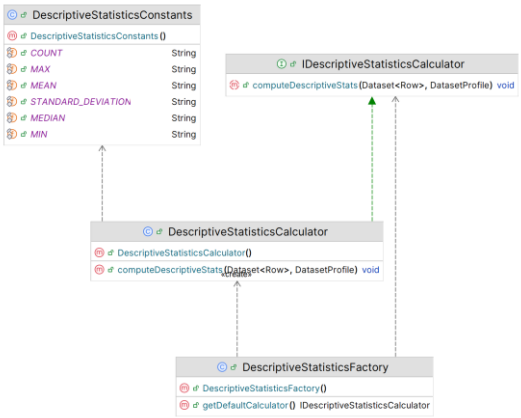
Το πακέτο **correlations:**



Εικόνα 15 correlations package uml diagram

Το πακέτο περιέχει κλάσεις και διεπαφές που σχετίζονται με τον υπολογισμό και την ανάλυση συσχετίσεων (correlations) σε δεδομένα. Η κλάση CorrelationsCalculatorFactory αναλαμβάνει τη δημιουργία αντικειμένων του τύπου ICorrelationsCalculator βάσει του τύπου της μεθόδου που προσδιορίζεται, ενώ προβλέπει μια εξαίρεση σε περίπτωση που ο τύπος της μεθόδου δεν είναι έγκυρος. Η κλάση CorrelationsMethod καθορίζει τους διαφορετικούς τύπους μεθόδων συσχετίσεων, με τον μόνο διαθέσιμο τύπο προς το παρόν να είναι ο "PEARSON". Η διεπαφή ICorrelationsCalculator ορίζει τη μέθοδο calculateAllPairsCorrelations, η οποία χρησιμοποιείται για τον υπολογισμό των διασυνδέσεων μεταξύ όλων των ζευγών στο σύνολο δεδομένων. Τέλος, η κλάση PearsonCorrelationsCalculator υλοποιεί τη διεπαφή ICorrelationsCalculator και παρέχει συγκεκριμένα τον υπολογισμό των συσχετίσεων Pearson μεταξύ των αριθμητικών στηλών του συνόλου δεδομένων.

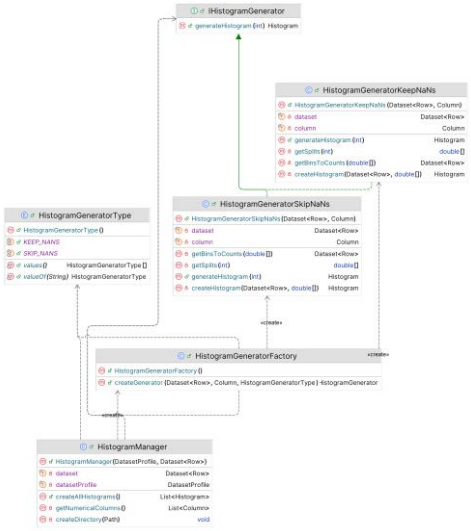
Το πακέτο **descriptivestatistics:**



Εικόνα 16 descriptivestatistics package uml diagram

Το πακέτο παρέχει λειτουργικότητα για τον υπολογισμό και την ανάλυση περιγραφικών στατιστικών σε ένα σύνολο δεδομένων. Η διεπαφή IDescriptiveStatisticsCalculator ορίζει τη μέθοδο computeDescriptiveStats, η οποία υπολογίζει τα ελάχιστα, μέγιστα, μέση τιμή, διάμεσο, πλήθος και τυπική απόκλιση για κάθε στήλη στο σύνολο δεδομένων και καθορίζει το προφίλ περιγραφικών στατιστικών για κάθε στήλη. Η κλάση DescriptiveStatisticsFactory παρέχει έναν προεπιλεγμένο υπολογιστή περιγραφικών στατιστικών μέσω της μεθόδου getDefaultCalculator. Η κλάση DescriptiveStatisticsConstants περιλαμβάνει σταθερές που χρησιμοποιούνται για τις περιγραφικές στατιστικές, όπως πλήθος, μέση τιμή και τυπική απόκλιση. Τέλος, η κλάση DescriptiveStatisticsCalculator υλοποιεί τη διεπαφή IDescriptiveStatisticsCalculator και προσφέρει τον υπολογισμό περιγραφικών στατιστικών για κάθε στήλη στο σύνολο δεδομένων, όπως ελάχιστη, μέγιστη, μέση τιμή, διάμεσο, πλήθος τιμών και τυπική απόκλιση.

Το πακέτο **histogram**:

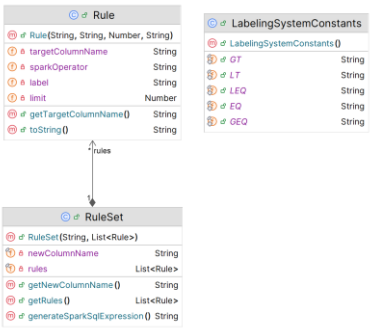


Εικόνα 17 histogram uml package diagram

Το πακέτο περιέχει κλάσεις που είναι υπεύθυνες για τη δημιουργία ιστογραμμάτων από δεδομένα. Η κλάση HistogramGeneratorFactory παρέχει μια μέθοδο για τη δημιουργία αντικειμένων τύπου IHistogramGenerator και, ανάλογα με τον τύπο του ιστογράμματος που καθορίζεται, επιστρέφει το αντίστοιχο αντικείμενο. Η διεπαφή IHistogramGenerator ορίζει τη μέθοδο generateHistogram, η οποία χρησιμοποιείται για τον υπολογισμό ενός ιστογράμματος με έναν συγκεκριμένο αριθμό bins. Η κλάση HistogramGeneratorKeepNaNs υλοποιεί έναν γεννήτορα ιστογράμματος που διατηρεί τις τιμές NaN και τις μετράει σε ένα επιπλέον bin, ενώ η κλάση HistogramGeneratorSkipNaNs υλοποιεί έναν γεννήτορα ιστογράμματος που παραλείπει τις τιμές NaN κατά τη δημιουργία του ιστογράμματος. Τέλος, η κλάση

HistogramGeneratorType ορίζει τους διαφορετικούς τύπους ιστογράμματος, όπως η διατήρηση ή η παράλειψη των τιμών NaN.

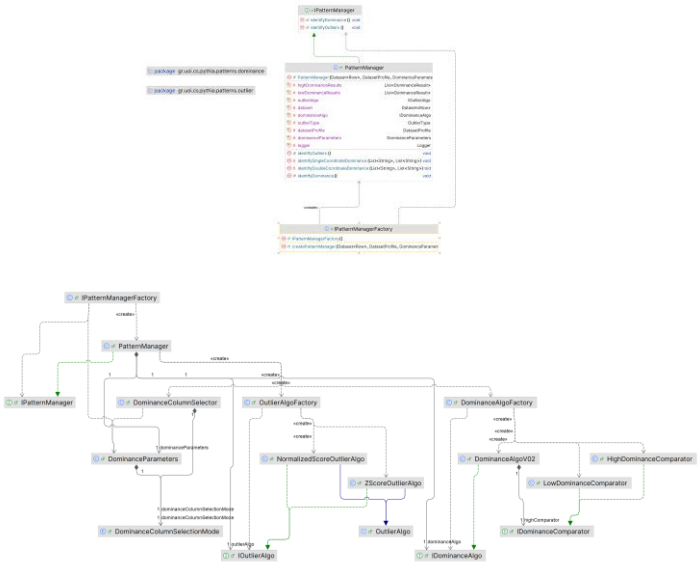
Το πακέτο **labeling:**



Εικόνα 18 labeling package uml diagram

Το πακέτο περιλαμβάνει κλάσεις που χρησιμοποιούνται για τον καθορισμό ετικετών σε στήλες. Η κλάση LabelingSystemConstants περιέχει σταθερές που αντιστοιχούν σε λεκτικά σύμβολα για συγκρίσεις, όπως μεγαλύτερο από (GT), μεγαλύτερο ή ίσο από (GEQ), μικρότερο από (LT), μικρότερο ή ίσο από (LEQ) και ισότητα (EQ). Η κλάση Rule αντιπροσωπεύει έναν κανόνα κατά τον οποίο εφαρμόζεται μια ετικέτα σε μια στήλη, με βάση έναν περιορισμό και έναν τελεστή. Τέλος, η κλάση RuleSet αντιπροσωπεύει ένα σύνολο κανόνων που αντιστοιχούν σε μια συγκεκριμένη στήλη.

Το πακέτο **patterns:**



Εικόνα 19 patterns package uml diagram

Το πακέτο περιλαμβάνει κλάσεις υπεύθυνες για την αναγνώριση προτύπων σε σύνολα δεδομένων. Η διεπαφή IPatternManager ορίζει τη λειτουργικότητα για την αναγνώριση προτύπων στα δεδομένα και καθορίζει τις βασικές λειτουργίες μίας κλάσης manager που εντοπίζει πρότυπα. Η κλάση PatternManager υλοποιεί τη διεπαφή IPatternManager και παρέχει τη βασική λειτουργικότητα για την αναγνώριση προτύπων. Η κλάση IPatternManagerFactory δημιουργεί κατάλληλα αντικείμενα τύπου IPatternManager και περιλαμβάνει μεθόδους, όπως η createPatternManager(), η οποία επιστρέφει ένα αντικείμενο κατάλληλο δοθέντων παραμέτρων. Το πακέτο περιλαμβάνει επίσης δύο υποπακέτα, τα οποία περιέχουν τους αλγορίθμους εύρεσης προτύπων. Στο εσωτερικό αυτών των πακέτων υπάρχουν κλάσεις όπως η DominanceAlgoFactory, η οποία δημιουργεί αλγορίθμους κυριαρχίας, και η OutlierAlgoFactory, η οποία δημιουργεί αλγορίθμους ανίχνευσης outlier τιμών.

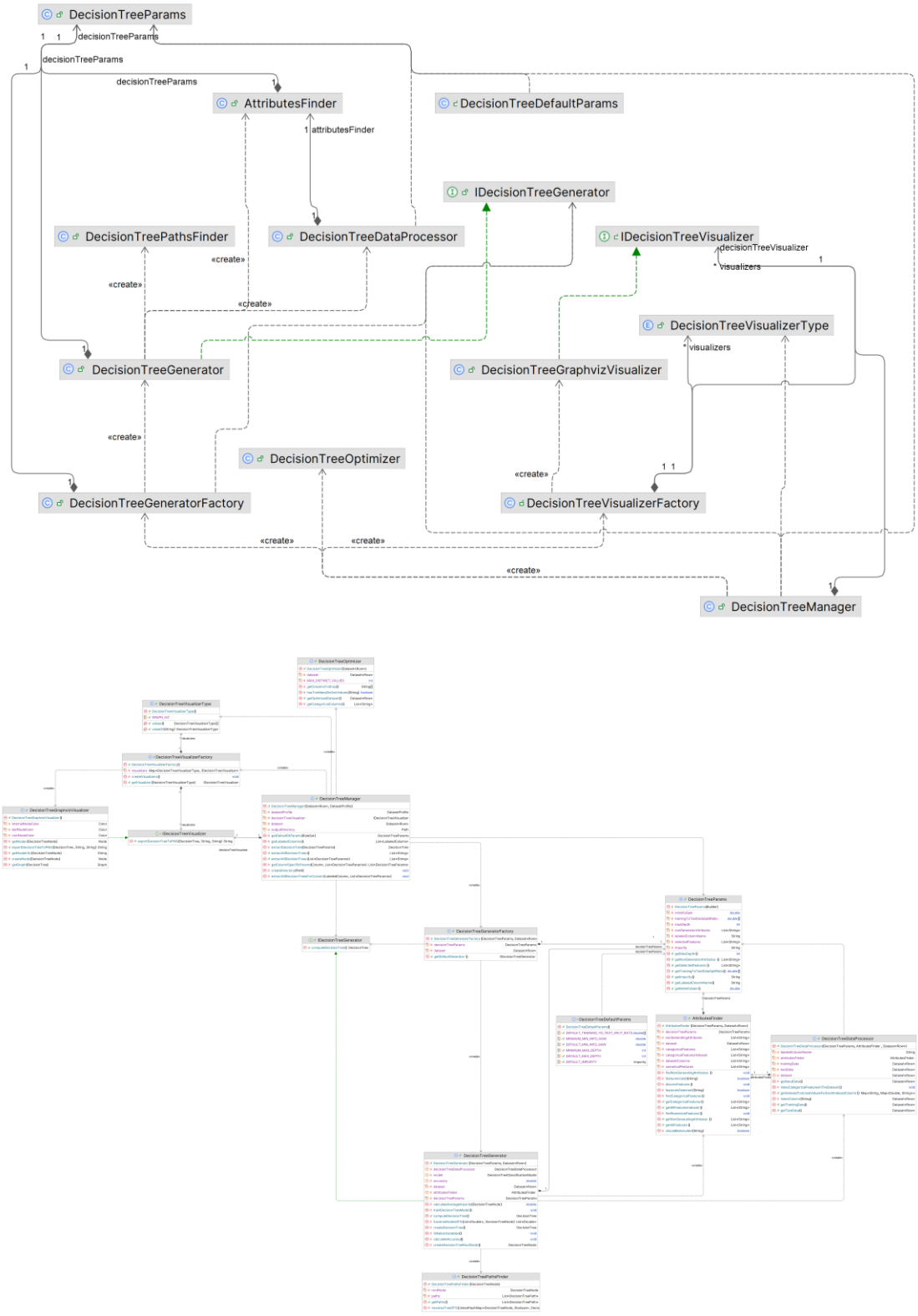
Το πακέτο **regression:**

Εικόνα 20 regression package uml diagram

Το πακέτο αυτό περιλαμβάνει εργαλεία για την εκτέλεση παλινδρομήσεων σε δεδομένα, με μια δομή που επιτρέπει επαναχρησιμοποίηση και εύκολη επέκταση. Η διεπαφή IRegressionPerformer ορίζει τις βασικές μεθόδους κάθε τύπου παλινδρόμησης, ενώ η αφηρημένη κλάση GeneralRegression περιέχει κοινές λειτουργίες για αποφυγή επανάληψης κώδικα. Η LinearRegressionPerformer υλοποιεί γραμμική παλινδρόμηση μεταξύ μιας ανεξάρτητης και μιας εξαρτημένης μεταβλητής, ενώ η MultipleLinearRegressionPerformer την επεκτείνει για πολλές ανεξάρτητες μεταβλητές. Η PolynomialRegressionPerformer εκτελεί πολυωνυμική παλινδρόμηση προσαρμόζοντας ένα πολυώνυμο στα δεδομένα, και η AutomatedRegressionPerformer επιλέγει αυτόματα τις ανεξάρτητες μεταβλητές βάσει κριτηρίων ακρίβειας. Η RegressionParameters διαχειρίζεται τις απαραίτητες παραμέτρους, ενώ η

RegressionPerformerFactory επιλέγει και δημιουργεί τον κατάλληλο τύπο παλινδρόμησης, ανάλογα τα ορίσματα που δέχεται.

Το πακέτο **decisiontree:**



Εικόνα 21 decisiontree package uml diagram

Το πακέτο παρέχει ένα σύνολο λειτουργιών για τη διαχείριση, την εκπαίδευση, την αξιολόγηση και την οπτικοποίηση των μοντέλων δέντρων απόφασης. Αναλυτικότερα:

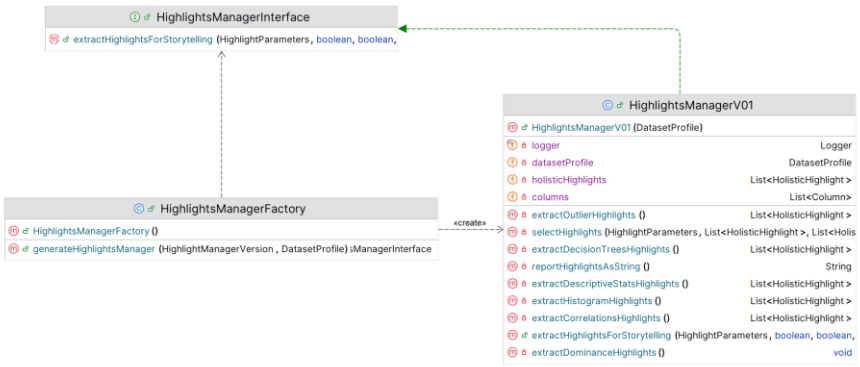
Διαχείριση Δέντρων Απόφασης: Η κλάση DecisionTreeManager αναλαμβάνει τη δημιουργία, την εξαγωγή και την οπτικοποίηση δέντρων απόφασης για κάθε στήλη με ετικέτα(labeled column). Η κλάση DecisionTreeOptimizer βελτιστοποιεί το σύνολο δεδομένων πριν από την εκπαίδευση, αφαιρώντας χαρακτηριστικά που μπορεί να επηρεάσουν αρνητικά την απόδοση.

Εκπαίδευση και προεπεξεργασία Δεδομένων: Η κλάση DecisionTreeGenerator εκπαιδεύει μοντέλα δέντρων απόφασης χρησιμοποιώντας το Apache Spark και παρέχει μεθόδους προεπεξεργασίας δεδομένων. Οι κλάσεις AttributesFinder και DecisionTreeDataProcessor αναλαμβάνουν τον εντοπισμό και την προ-επεξεργασία των δεδομένων για την κατασκευή του δέντρου.

Οπτικοποίηση Δέντρων Απόφασης: Η κλάση DecisionTreeGraphvizVisualizer χρησιμοποιεί τη βιβλιοθήκη Graphviz για τη δημιουργία γραφικών αναπαραστάσεων των δέντρων απόφασης σε μορφή εικόνας PNG. Οι κλάσεις DecisionTreeVisualizerFactory και IDecisionTreeVisualizer παρέχουν ένα εύκολο μηχανισμό δημιουργίας διαφόρων τύπων οπτικοποίησης.

Διαχείριση Παραμέτρων: Οι κλάσεις DecisionTreeDefaultParams και DecisionTreeParams διαχειρίζονται τις παραμέτρους που απαιτούνται για την εκπαίδευση του μοντέλου, επιτρέποντας την εύκολη προσαρμογή ανάλογα με τις ανάγκες της εφαρμογής.

Το πακέτο **highlights:**



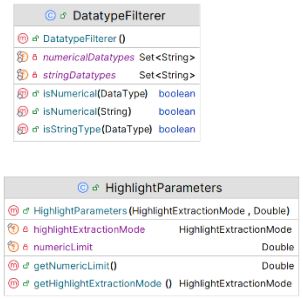
Εικόνα 22 highlights package uml diagram

Το πακέτο προσφέρει εργαλεία για την εξαγωγή σημαντικών σημείων (highlights) από αναλύσεις δεδομένων. Αυτά τα σημεία μπορεί να σχετίζονται με περιγραφικά στατιστικά, ιστογράμματα, συσχετίσεις, δέντρα απόφασης, ανίχνευση αποκλίσεων και

κυριαρχία (dominance). Η επιλογή των σημείων ενδιαφέροντος γίνεται με βάση παραμέτρους, όπως ο τύπος εξαγωγής, το κατώφλι τιμών και η επιστροφή των κορυφαίων αποτελεσμάτων. Τα αποτελέσματα μπορούν να παρουσιαστούν ως κείμενο ή με ειδικές αναφορές.

Η HighlightsManagerInterface είναι η διεπαφή που καθορίζει τις μεθόδους για την εξαγωγή αυτών των σημείων από τις αναλύσεις δεδομένων. Η HighlightsManagerFactory δημιουργεί ένα κατάλληλο αντικείμενο σημείων ενδιαφέροντος, ανάλογα με την έκδοση και το προφίλ δεδομένων. Η HighlightsManagerV01 είναι μια υλοποίηση αυτής της διεπαφής και παρέχει τις λειτουργίες εξαγωγής σημείων από τα δεδομένα.

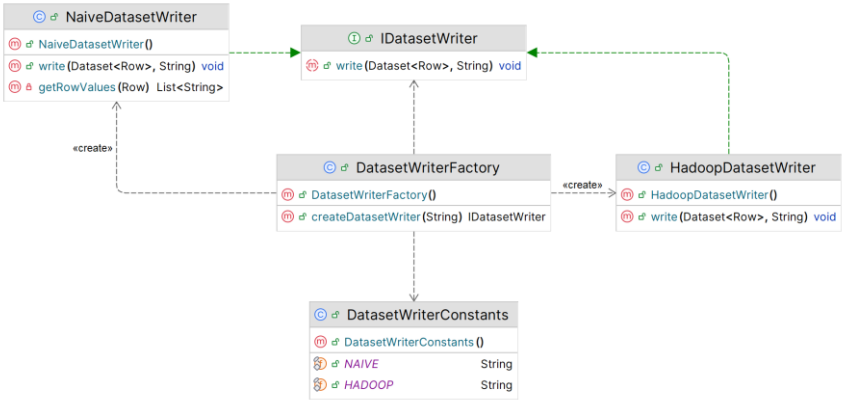
Το πακέτο **util:**



Εικόνα 23 util package uml diagram

Το πακέτο περιέχει χρήσιμες κλάσεις για τη διαχείριση δεδομένων και παραμέτρων που χρησιμοποιούνται. Η κλάση DatatypeFilterer παρέχει μια σειρά από μεθόδους για τον έλεγχο του τύπου δεδομένων. Η κλάση HighlightParameters ορίζει τις παραμέτρους που απαιτούνται για την εξαγωγή σημείων ενδιαφέροντος (highlights). Το εμφωλευμένο enum HighlightExtractionMode ορίζει τους διαθέσιμους τρόπους εξαγωγής highlights.

Το πακέτο **writer:**



Εικόνα 24 writer package uml diagram

Το πακέτο περιλαμβάνει κλάσεις που ασχολούνται με την εγγραφή δεδομένων σε αρχεία. Η διεπαφή IDatasetWriter ορίζει τη μέθοδο write, η οποία χρησιμοποιείται για την εγγραφή ενός Dataset<Row> σε ένα δεδομένο filePath. Η κλάση NaiveDatasetWriter υλοποιεί τη διεπαφή IDatasetWriter και η μέθοδος write διατρέχει το Dataset<Row> και εγγράφει κάθε γραμμή στο αρχείο χρησιμοποιώντας έναν απλό τρόπο εγγραφής, όπου τα δεδομένα διαχωρίζονται με κόμμα. Η κλάση HadoopDatasetWriter επίσης υλοποιεί τη διεπαφή IDatasetWriter, και η μέθοδος write εγγράφει το Dataset<Row> στο δίσκο χρησιμοποιώντας το Hadoop File System. Η κλάση DatasetWriterFactory δημιουργεί αντικείμενα IDatasetWriter βάσει του τύπου που περνά ως παράμετρο, ενώ η κλάση DatasetWriterConstants περιέχει σταθερές για τους διαφορετικούς τύπους writers, όπως Hadoop και Naive.

3.1.3 Επίλυση των προβλημάτων

Στην ενότητα αυτή παρουσιάζονται συνοπτικά οι τεχνικές και οι μεθοδολογίες που χρησιμοποιήθηκαν για την αντιμετώπιση των προβλημάτων που αναφέρθηκαν. Στην επόμενη ενότητα, οι διαδικασίες αυτές θα αναλυθούν με περισσότερη λεπτομέρεια.

Για την επέκταση του προφίλ του αρχείου με χαρακτηριστικά που αφορούν ολόκληρο το σύνολο δεδομένων (π.χ. αριθμός γραμμών, τοποθεσία αρχείου, μέγεθος αρχείου), δημιουργήθηκε ένα κατάλληλο πακέτο που είναι υπεύθυνο για τον υπολογισμό των παραπάνω χαρακτηριστικών. Για τον αριθμό των γραμμών, χρησιμοποιήθηκε το Dataset<Row> που παρέχεται από το Apache Spark. Το μέγεθος του αρχείου υπολογίστηκε μέσω των κλάσεων FileSystem και FileStatus, ενώ για την πλήρη ημερομηνία δημιουργίας του προφίλ χρησιμοποιήθηκε η κλάση Timestamp.

Για τον υπολογισμό των null και των διακριτών τιμών, δημιουργήθηκε ένα υποσύστημα που αναλαμβάνει αυτή τη λειτουργία. Ο υπολογισμός πραγματοποιείται με τη χρήση του Dataset<Row> από το Apache Spark, καθώς και SQL ερωτημάτων που υλοποιούνται μέσω του Spark DataFrame API. Μετά τον υπολογισμό, ενημερώνεται η υπάρχουσα δομή δεδομένων που διατηρεί τις πληροφορίες του προφίλ.

Για τον υπολογισμό των τεταρτημορίων (Q1, Q2, Q3), χρησιμοποιήθηκε η μέθοδος summary() του Spark, η οποία παρέχει αυτές τις τιμές αυτόματα. Οι τιμές αυτές προσαρτήθηκαν στην υπάρχουσα δομή DescriptiveStatistics της Pythias. Επιπλέον, για τον υπολογισμό των τιμών mode, δημιουργήθηκε μια εξειδικευμένη μέθοδος εντός της δομής DescriptiveStatistics, η οποία περιγράφεται με περισσότερες λεπτομέρειες στην επόμενη ενότητα. Για τη δημιουργία ισοϋψούς διαγράμματος, πραγματοποιήθηκε μια

ελαφρά τροποποίηση στο υπάρχον πακέτο histogram της Pythias, ώστε να ανταποκρίνεται στις ανάγκες της εφαρμογής.

Για την καταγραφή του score που αφορά τον τύπο των στηλών, δημιουργήθηκε ένα πακέτο που ελέγχει τις τιμές του dataset σε ένα σύνολο κανονικών εκφράσεων (regex). Το score καταγράφεται ανάλογα με το βαθμό ταιριάσματος των τιμών με τα καθορισμένα πρότυπα.

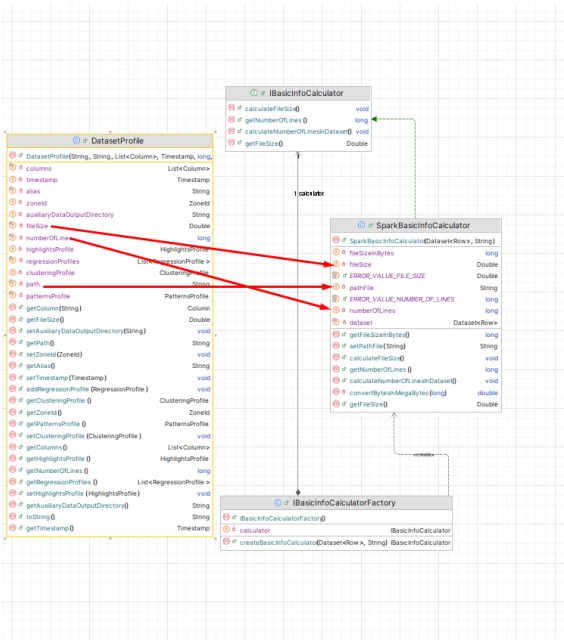
Για τη δημιουργία της γραφικής διεπαφής, χρησιμοποιήθηκε η βιβλιοθήκη Java Swing. Μέσω αυτής, δημιουργήθηκαν φιλικά παράθυρα προς τον χρήστη επιτρέποντάς του την εύκολη και γρήγορη χρήση της Pythias.

3.2 Σχεδίαση και αρχιτεκτονική λογισμικού

Σε αυτή την ενότητα θα αναλυθούν και θα επεξηγηθούν οι νέες δομές λογισμικού που προστέθηκαν.

3.2.1 Το πακέτο generalinfo

Για την προσθήκη των γενικών χαρακτηριστικών του σετ δεδομένων δημιουργήθηκε το πακέτο **generalinfo**. Αυτό είναι υπεύθυνο για τον υπολογισμό των αριθμό των γραμμών του αρχείου προς φόρτωση, το μέγεθος αυτού και την καταγραφή ολόκληρης τοποθεσίας του αρχείου. Παρακάτω φαίνεται η δομή του :



Εικόνα 25 generalinfo uml package diagram

Στο πακέτο αυτό είναι ορισμένη μια διεπαφή IBasicInfoCalculator (Interface) που ορίζει τις λειτουργίες υπολογισμού των δεδομένων. Μια κλάση IBasicInfoCalculatorFactory που

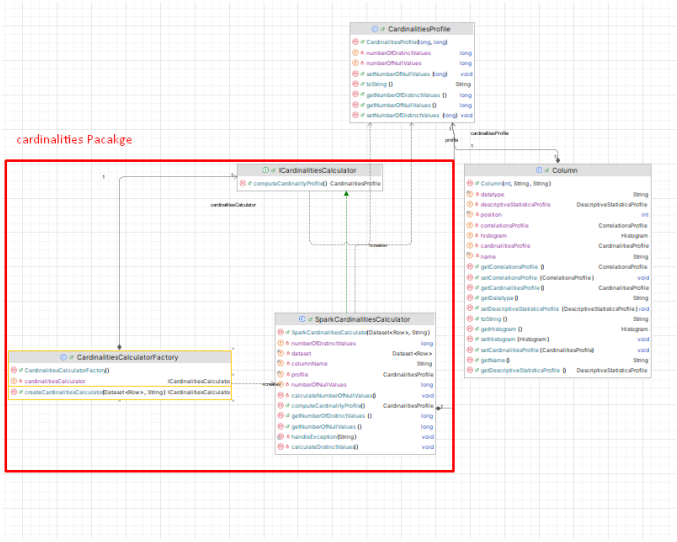
είναι υπεύθυνη για την δημιουργία αντικειμένων που θα εκτελέσουν τις λειτουργίες που ορίζει η διεπαφή, αλλά και μία κλάση SparkBasicInfoCalculator που υλοποιεί τη διεπαφή. Επιπλέον, οι μεταβλητές που υπολογίζονται από αυτό το πακέτο είναι πλέον πεδία στην ήδη υπάρχουσα δομή DatasetProfile και αφού υπολογιστούν καταγράφονται στο datasetProfile. Βέβαια έπρεπε να ενημερωθεί η registerDataset() του Engine ώστε πριν τη δημιουργία του datasetProfile αντικειμένου να υπολογιστούν τα απαραίτητα δεδομένα ώστε να ενημερωθεί πλήρως με τις υπολογισμένες τιμές. Παρακάτω φαίνονται οι τροποποιήσεις.

```
@Override  A lamprosvlax13 +3
public void registerDataset(String alias, String path, StructType schema) throws AnalysisException {

    Timestamp executionDateTime = new Timestamp(sparkSession.sparkContext().startTime());
    Instant start = Instant.now();
    File file = new File(path);
    String absolutePath = file.getAbsolutePath();
    dataset = dataFrameReaderFactory.createDataframeReader(absolutePath, schema).read();

    List<Column> columns = new ArrayList<>();
    StructField[] fields = dataset.schema().fields();
    for (int i = 0; i < fields.length; ++i) {
        String columnName = fields[i].name();
        String dataType = fields[i].dataType().toString();
        Column column = new Column(i, columnName, dataType);
        ICardinalitiesCalculator cardinalitiesCalculator = cardinalitiesCalculatorFactory.createCardinalitiesCalculator(dataset,columnName);
        column.setCardinalitiesProfile(cardinalitiesCalculator.computeCardinalityProfile());
        columns.add(column);
    }
    IBasicInfoCalculator calculator = basicInfoCalculatorFactory.createBasicInfoCalculator(dataset,absolutePath);
    long numberOfLines = calculator.getNumberOfLines();
    Double fileSize = calculator.getFileSize();
    datasetProfile = new DatasetProfile(alias, absolutePath, columns,executionDateTime,numberOfLines,fileSize);
    logger.info(String.format("Registered Dataset file with alias '%s' at %s", alias, absolutePath));

    Instant end = Instant.now();
    Duration duration = Duration.between(start, end);
    logger.info(String.format("Duration of registerDataset: %s / %sms", duration, duration.toMillis()));

    //TODO PRINT add-ons- Trial!!
    logger.info(String.format("DateTime of Profiling %s",datasetProfile.getTimestamp())); //Preview: 24/04/18 19:31:50 INFO DatasetProfiler: DateTime of Pr
    logger.info(String.format("numOfLines: %d",datasetProfile.getNumberOfLines() ));
    logger.info(String.format("fileSize in MB: %s", datasetProfile.getFileSize()));

}
```

Εικόνα 26 engine/DatasetProfiler.java

3.2.2 Το πακέτο cardinalities

Για τον υπολογισμό των null, διακριτών τιμών δημιουργήθηκε το πακέτο cardinalities που είναι υπεύθυνο για τον υπολογισμό των τιμών αυτών. Παρακάτω φαίνεται η δομή του:



Εικόνα 27 cardinalities uml package

Στο πακέτο υπάρχει μια διεπαφή ICardinalitiesCalculator που ορίζει τις λειτουργίες υπολογισμού, κλάση CardinalitiesCalculatorFactory που είναι υπεύθυνη για την δημιουργία αντικειμένων που υλοποιούν την διεπαφή αλλά και κλάση SparkCardinalitiesCalculator που υλοποιεί την διεπαφή και είναι υπεύθυνη για την εκτέλεση των υπολογισμών. Επιπλέον, έχει δημιουργηθεί ένα CardinalitieProfile model class που περιέχει τις υπολογισμένες πληροφορίες. Αυτό το CardinalitieProfile θα είναι ως πεδίο στην υπάρχουσα model class Column έτσι ώστε για κάθε στήλη να αποθηκεύονται οι πληροφορίες. Επιπλέον, χρειάζεται τροποποίηση η registerDataset στην Engine ώστε για κάθε στήλη να υπολογίζεται αυτό το CardinalitiesProfile και να ενημερώνεται η στήλη με το προφίλ αυτό με τις υπολογισμένες τιμές.

```
@Override
public void registerDataset(String alias, String path, StructType schema) throws AnalysisException {

    Timestamp executionDateTime = new Timestamp(sparkSession.sparkContext().startTime());
    Instant start = Instant.now();
    File file = new File(path);
    String absolutePath = file.getAbsolutePath();
    dataset = dataFrameReaderFactory.createDataFrameReader(absolutePath, schema).read();

    List<Column> columns = new ArrayList<>();
    StructField[] fields = dataset.schema().fields();
    for (int i = 0; i < fields.length; ++i) {
        String columnName = fields[i].name();
        String dataType = fields[i].dataType().toString();
        Column column = new Column(i, columnName, dataType);
        ICardinalitiesCalculator cardinalitiesCalculator = cardinalitiesCalculatorFactory.createCardinalitiesCalculator(dataset,columnName);
        column.setCardinalitiesProfile(cardinalitiesCalculator.computeCardinalityProfile());
        columns.add(column);
    }
    IBasicInfoCalculator calculator = basicInfoCalculatorFactory.createBasicInfoCalculator(dataset,absolutePath);
    long numberOfLines = calculator.getNumberOfLines();
    Double fileSize = calculator.getFileSize();
    datasetProfile = new DatasetProfile(alias, absolutePath, columns,executionDateTime,numberOfLines,fileSize);
    logger.info(String.format("Registered Dataset file with alias '%s' at %s", alias, absolutePath));

    Instant end = Instant.now();
    Duration duration = Duration.between(start, end);
    logger.info(String.format("Duration of registerDataset: %s / %sms", duration, duration.toMillis()));

    //TODO PRINT add-ons- Trial!!
    logger.info(String.format("DateTime of Profiling %s",datasetProfile.getTimestamp())); //Preview: 24/04/18 19:31:50 INFO DatasetProfiler: DateTime of Profiling 2024-04-18 19:31:45.5
    logger.info(String.format("numOfLines: %d",datasetProfile.getNumberOfLines() ));
    logger.info(String.format("fileSize in MB: %s", datasetProfile.getFileSize()));
}
```

Εικόνα 28 engine/DatasetProfiler.java

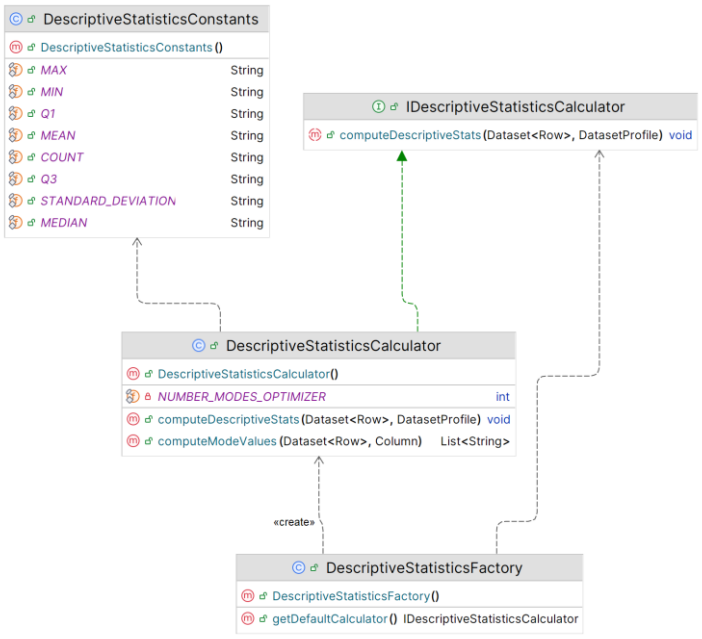
Για την ανάλυση των null και κενών τιμών, χρησιμοποιούνται μετασχηματισμοί όπως select, filter (με συνθήκες isNull και trim().equalTo("")) και την τελική συνάρτηση count. Η λογική που ακολουθεί είναι η εξής: “Επιλέγουμε τη στήλη, φιλτράρουμε τις εγγραφές

όπου η τιμή είναι null ή περιέχει μόνο κενά διαστήματα, και μετράμε αυτές τις εγγραφές." Αντίστοιχα, για την εξαγωγή των διακριτών τιμών, χρησιμοποιεί filter (με συνθήκες isNotNull και trim().notEqual("")), distinct για την απομάκρυνση των ίδιων τιμών, και count για την τελική μέτρηση.

3.2.3 Το πακέτο DescriptiveStatistics

Το πακέτο αυτό επεκτείνεται με την προσθήκη των υπολογισμών για τα τεταρτημόρια (Q1, Q2, Q3) για τα δεδομένα της στήλης καθώς και την λίστα των modes(τιμών με την μεγαλύτερη εμφάνιση στη στήλη). Για τον υπολογισμό των Q1 και Q3, καθώς και της διάμεσου (Q2), γίνεται χρήση της μεθόδου summary του Spark, η οποία παρέχει αυτά τα στατιστικά άμεσα για κάθε στήλη του dataset.

Παράλληλα, για τον υπολογισμό των modes, δηλαδή των τιμών με τη μεγαλύτερη συχνότητα, εφαρμόζεται η μέθοδος computeModeValues. Στη μέθοδο αυτή, τα δεδομένα ομαδοποιούνται κατά τιμή και υπολογίζεται η συχνότητα εμφάνισης κάθε τιμής. Οι τιμές με τη μεγαλύτερη συχνότητα επιλέγονται ως modes και επιστρέφονται σε μία λίστα. Ο αριθμός των modes περιορίζεται με τη σταθερά NUMBER_MODES_OPTIMIZER για να αποφευχθεί η επιστροφή υπερβολικά μεγάλου αριθμού modes σε περιπτώσεις όπου κάθε τιμή στης στήλης έχει συχνότητα εμφάνισης 1 δηλαδή κάθε τιμή της στήλης είναι μοναδική. Η σχεδίαση του πακέτου δεν αλλάζει όπως φαίνεται παρακάτω:



Εικόνα 29 descriptivestatistics package uml diagram

Οι κλάσεις όμως DescriptiveStatisticsCalculator, DescriptiveStatisticsConstants είναι ελαφρώς τροποποιημένες ώστε να υποστηρίζουν τον υπολογισμό των παραμέτρων

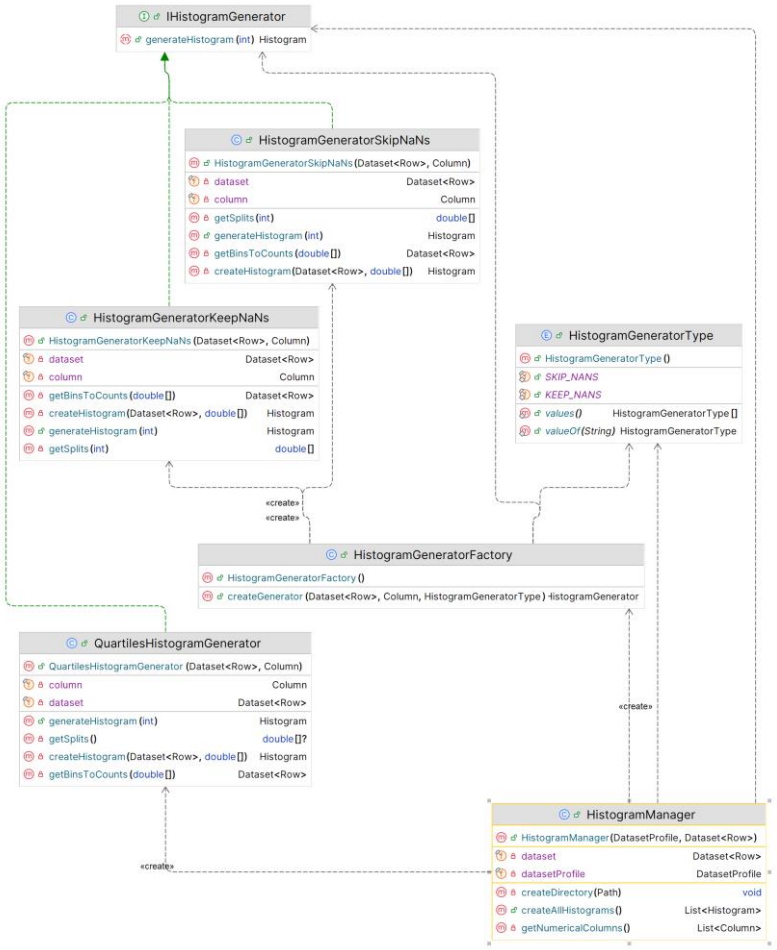
q1,q2,modes όπως εξηγήθηκε παραπάνω. Επίσης χρειάζεται ελαφρά τροποποίηση η κλάση DescriptiveStatisticsProfile ώστε να διατηρεί και τις νέες υπολογισμένες τιμές ως πεδία.

3.2.4 Το πακέτο histogram

Στο πακέτο προστέθηκε η κλάση QuartileHistogramCalculator, με σκοπό τη δημιουργία ισοϋψούς ιστογράμματος βασισμένου στα τεταρτημόρια των δεδομένων. Το ιστόγραμμα αυτό χωρίζει τα δεδομένα σε bins, όπου κάθε bin αντιστοιχεί στην περιοχή μεταξύ δύο διαδοχικών σημείων, όπως π.χ. από το Q1 έως το Q2, από το Q2 έως το Q3, κ.λπ.

Ένα σημαντικό στοιχείο είναι η χρήση του epsilon, μιας πολύ μικρής θετικής τιμής, για τον υπολογισμό των ορίων των bins. Αυτό βοηθά στην αποφυγή προβλημάτων που ενδέχεται να προκύψουν από την ύπαρξη ίδιων τιμών στα όρια των bins. Για παράδειγμα, όταν δύο διαδοχικά όρια έχουν την ίδια τιμή, όπως όταν το Q1 είναι ίσο με το Q2, χωρίς τη χρήση του epsilon, το bin μεταξύ αυτών των δύο ορίων θα είχε μηδενικό εύρος, κάτι που θα προκαλούσε σφάλμα στο Bucketizer του Spark. Για παράδειγμα ας υποθέσουμε ότι τα υπολογισμένα τεταρτημόρια είναι τα εξής: Min = 10, Q1 = 15, Q2 = 15 (διάμεσος ίσος με Q1), Q3 = 30 και Max = 35. Χωρίς την προσθήκη του epsilon, τα splits θα ήταν: [10, 15, 15, 30, 35]. Αυτό θα προκαλούσε πρόβλημα, επειδή τα όρια 15 και 15 θα οδηγούσαν σε ένα bin με μηδενικό εύρος, το οποίο δεν μπορεί να διαχειριστεί σωστά το Bucketizer του Spark. Με την προσθήκη του epsilon, το αποτέλεσμα των splits θα γίνει: [10, 15, 15.0001, 30, 35]. Έτσι, τα bins θα είναι από 10 έως 15, από 15 έως 15.0001, από 15.0001 έως 30 και από 30 έως 35. Αυτή η προσέγγιση εξασφαλίζει ότι το ιστόγραμμα δημιουργείται σωστά χωρίς να προκύπτουν προβλήματα από διπλότυπες τιμές.

Η σχεδίαση του πακέτου δεν άλλαξε και φαίνεται παρακάτω:



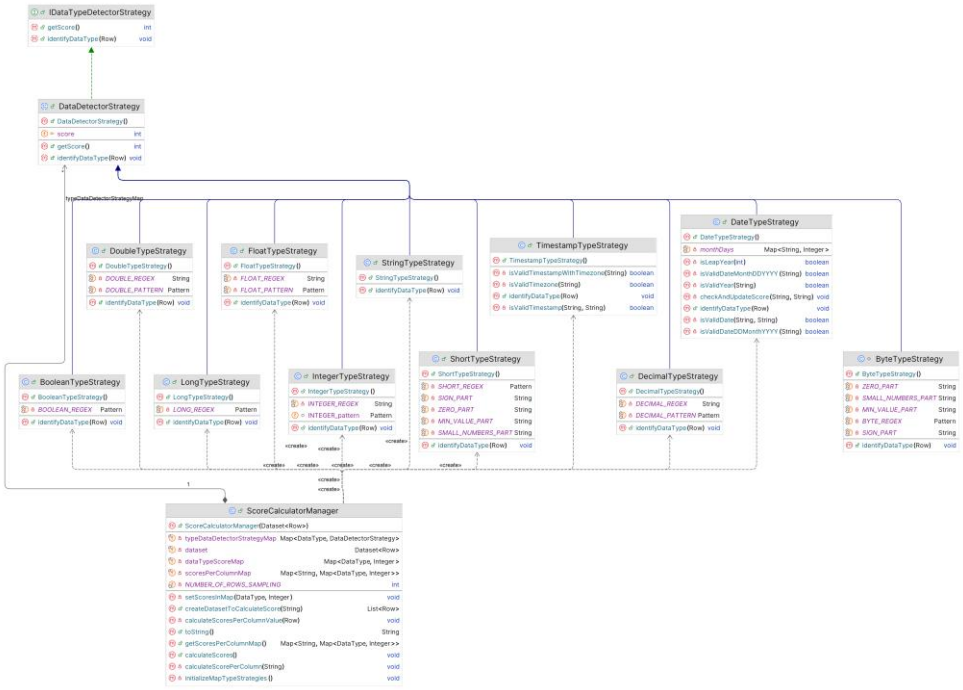
Εικόνα 30 histogram package uml diagram

Η κλάση HistogramManager πλέον είναι υπεύθυνη στον να δημιουργεί και τα ισοϋψείς ιστογράμματα με την μορφή quartiles πέρα από τα κλασικά ιστογράμματα. Η κλάση QuartileHistogramGenerator δεν δημιουργεί συσχέτιση ή εξάρτηση με το πακέτο descriptiveStatistics που υπολογίζουν τα q1, q2, q3. Η κλάση αυτή κάνει χρήση του descriptiveProfile που έχει κάθε στήλη και ελέγχει αν είναι τα δεδομένα που χρειάζεται είναι υπολογισμένα για να δημιουργήσει τα ισοϋψή διαγράμματα όπως ακριβώς γίνεται και για τα απλά ιστογράμματα. Μετά το τέλος των υπολογισμών κάθε στήλη διατηρεί όλες τις υπολογισμένες πληροφορίες.

3.2.5 Το πακέτο datatypeIdentifier

Αυτό το πακέτο δημιουργήθηκε για να καταγράφει ένα score για ορισμένους τύπους δεδομένων που υποστηρίζει το Spark, αξιολογώντας κατά πόσο οι τιμές μιας στήλης μπορούν να χαρακτηριστούν ως αυτού του τύπου. Συγκεκριμένα, κάθε τιμή μιας στήλης ελέγχεται με τη χρήση διαφόρων κανονικών εκφράσεων, οι οποίες αντιπροσωπεύουν διαφορετικούς τύπους δεδομένων. Όταν μια τιμή κάνει match (ταιριάζει) με μια κανονική έκφραση, αυξάνεται το score του αντίστοιχου τύπου δεδομένων. Με αυτόν

τον τρόπο, στο τέλος των υπολογισμών, προκύπτει ένα score για κάθε τύπο δεδομένων, το οποίο δείχνει πόσες από τις τιμές της στήλης θα μπορούσαν να χαρακτηριστούν, για παράδειγμα, ως Numeric types (όπως IntegerType, DoubleType κ.λπ.), ως String types και άλλους τύπους δεδομένων. Παρακάτω φαίνεται η σχεδίαση του πακέτου:



Εικόνα 31 datatypeIdentifier package uml diagram

Το πακέτο περιλαμβάνει τη διεπαφή (IDataTypeDetectorStrategy), την αφηρημένη κλάση (DataDetectorStrategy) και τις υποκλάσεις που υλοποιούν τον έλεγχο των τύπων δεδομένων. Η διεπαφή ορίζει δύο μεθόδους: μία για τον έλεγχο του τύπου δεδομένων μιας τιμής και μία για την επιστροφή του score, δηλαδή πόσες τιμές ταιριάζουν με τον τύπο. Η αφηρημένη κλάση υλοποιεί τη λογική υπολογισμού του score και απαιτεί από τις υποκλάσεις να υλοποιήσουν τον έλεγχο του τύπου δεδομένων.

Όπως αναφέρθηκε για τον έλεγχο του τύπου δεδομένων, γίνεται χρήση κανονικών εκφράσεων. Η κανονική έκφραση (regex ή regexp) αποτελεί ένα εργαλείο που χρησιμοποιείται ευρέως στην επεξεργασία κειμένων και δεδομένων. Οι κανονικές εκφράσεις επιτρέπουν στους προγραμματιστές να καθορίσουν μοτίβα, τα οποία μπορούν να αντιστοιχούν σε συγκεκριμένες ακολουθίες χαρακτήρων μέσα σε κείμενα. Με τη χρήση αυτών των μοτίβων, είναι δυνατές λειτουργίες όπως η εύρεση συγκεκριμένων πληροφοριών ή η αντικατάσταση τμημάτων κειμένου με άλλες τιμές. Ένα παράδειγμα κανονικής έκφρασης για έναν απλό αριθμό τηλεφώνου, χωρίς να απαιτείται ο πρόθεμα ή η εναλλαγή μεταξύ με ή χωρίς διάστημα, η κανονική έκφραση

μπορεί να είναι: ^\d{10}$. Αναλυτικά, ^ σημαίνει "αρχή γραμμής", \d{10} αναζητά ακριβώς 10 ψηφία, και $ σημαίνει "τέλος γραμμής".

Πιο συγκεκριμένα στο πακέτο :

Για την αναγνώριση τύπων Boolean χρησιμοποιείται η κανονική έκφραση "^(true|false|1|0|yes|no|TRUE|FALSE|YES|NO|on|off|enabled|disabled|ok|not\sok)$", η οποία επιτρέπει τις τιμές "true", "false", "1", "0", "yes", "no", "TRUE", "FALSE", "YES", "NO", "on", "off", "enabled", "disabled", "ok" και "not ok", οι οποίες αντιπροσωπεύουν Boolean τύπους δεδομένων.

Για την αναγνώριση του τύπου Byte, η κανονική έκφραση "^(//+|-)?" καλύπτει το προαιρετικό πρόσημο, ενώ η "0" αναγνωρίζει την ειδική περίπτωση του αριθμού μηδέν. Η κανονική έκφραση "[1-9]\d?|1[0-1]\d|12[0-7]" καλύπτει τους αριθμούς από 1 έως 127, και η "-128" αναγνωρίζει την ελάχιστη τιμή. Συνδυασμένα, αυτά δημιουργούν την κανονική έκφραση "^" + SIGN_PART + "(" + ZERO_PART + "|" + SMALL_NUMBERS_PART + "|" + MIN_VALUE_PART + ")$", η οποία αναγνωρίζει τιμές τύπου byte, δηλαδή από -128 έως 127."

Για την αναγνώριση του τύπου Decimal, η κανονική έκφραση "^[-+]?(\\d+\\.\\d*|\\.\\d+)$" αναγνωρίζει δεκαδικούς αριθμούς με μεγάλη ακρίβεια, ανεξάρτητα από το πλήθος των δεκαδικών ψηφίων. Επιτρέπει προαιρετικό πρόσημο ([-+]?) και καλύπτει δεκαδικούς με ακέραιο μέρος (\\d+\\.\\d*), και δεκαδικούς χωρίς ακέραιο μέρος (\\.\\d+), αναγνωρίζοντας έτσι όλους τους δεκαδικούς αριθμούς με οποιοδήποτε πλήθος δεκαδικών ψηφίων.

Για την αναγνώριση του τύπου Float, η κανονική έκφραση "^[-+]?(\\d{1,6}\\.\\d{1,7}|\\d{1,6}[eE][-+]?\\d{1,2}|\\d{1,6}\\.\\d{1,7}[eE][-+]?\\d{1,2})$" αναγνωρίζει αριθμούς κινητής υποδιαστολής (float) με συγκεκριμένο εύρος και ακρίβεια, αλλά χωρίς να επιτρέπει ακέραιους αριθμούς. Επιτρέπει προαιρετικό πρόσημο ([-+]?). Απαιτεί την παρουσία είτε δεκαδικού μέρους, είτε εκθέτη, είτε και των δύο. Καλύπτει τις εξής περιπτώσεις: αριθμούς με ακέραιο μέρος (1-6 ψηφία) και δεκαδικό μέρος (1-7 ψηφία) (\\d{1,6}\\.\\d{1,7}), αριθμούς με ακέραιο μέρος (1-6 ψηφία) και επιστημονική σημείωση (εκθέτης 1-2 ψηφία) (\\d{1,6}[eE][-+]?\\d{1,2}), και αριθμούς με ακέραιο μέρος (1-6 ψηφία), δεκαδικό μέρος (1-7 ψηφία) και επιστημονική σημείωση (εκθέτης 1-2 ψηφία) (\\d{1,6}\\.\\d{1,7}[eE][-+]?\\d{1,2}). Το ^ και το $ ορίζουν την αρχή και το τέλος της συμβολοσειράς, εξασφαλίζοντας ότι ολόκληρη η συμβολοσειρά αντιστοιχεί στην έκφραση. Είναι ορισμένη έτσι ώστε να καλύπτει τα όρια ενός float αριθμού.

Για την αναγνώριση του τύπου Double, η κανονική έκφραση "^[-+]?(\\d{1,15}\\.\\d{1,16}|\\d{1,15}[eE][-+]?\\d{1,3}|\\d{1,15}\\.\\d{1,16}[eE][-+]?\\d{1,3})$" αναγνωρίζει αριθμούς διπλής ακρίβειας (double) με συγκεκριμένο εύρος και ακρίβεια, αλλά χωρίς να επιτρέπει ακέραιους αριθμούς. Επιτρέπει προαιρετικό πρόσημο ([-+]?). Απαιτεί την παρουσία είτε δεκαδικού μέρους, είτε εκθέτη, είτε και των δύο. Καλύπτει τις εξής περιπτώσεις: αριθμούς με ακέραιο μέρος (1-15 ψηφία) και δεκαδικό μέρος (1-16 ψηφία) (\\d{1,15}\\.\\d{1,16}), αριθμούς με ακέραιο μέρος (1-15 ψηφία) και επιστημονική σημείωση (εκθέτης 1-3 ψηφία) (\\d{1,15}[eE][-+]?\\d{1,3}), και αριθμούς με ακέραιο μέρος (1-15 ψηφία), δεκαδικό μέρος (1-16 ψηφία) και επιστημονική σημείωση (εκθέτης 1-3 ψηφία) (\\d{1,15}\\.\\d{1,16}[eE][-+]?\\d{1,3}). Το ^ και το $ ορίζουν την αρχή και το τέλος της συμβολοσειράς, εξασφαλίζοντας ότι ολόκληρη η συμβολοσειρά αντιστοιχεί στην έκφραση. Αυτή η έκφραση περιορίζει το μέγεθος του ακέραιου, του δεκαδικού μέρους και του εκθέτη, καθορίζοντας έτσι την ακρίβεια και το μέγιστο μέγεθος των αριθμών double που θα αναγνωριστούν, ενώ ταυτόχρονα αποκλείει την αναγνώριση ακέραιων αριθμών.

Για την αναγνώριση του τύπου Integer, η κανονική έκφραση "^[-+]?" καλύπτει το προαιρετικό πρόσημο. Η "\\d{1,9}" αναγνωρίζει αριθμούς από ένα έως εννέα ψηφία. Οι εκφράσεις "214748364[0-7]" και "-2147483648" αναγνωρίζουν τις μέγιστες και ελάχιστες τιμές που υποστηρίζει ο τύπος Integer. Συνδυαστικά, αυτά δημιουργούν την κανονική έκφραση "^[-+]?(\\d{1,9}|214748364[0-7]|-2147483648)$", η οποία αναγνωρίζει ακαιρέους, δηλαδή από -2.147.483.648 έως 2.147.483.647.

Για την αναγνώριση του τύπου long, η κανονική έκφραση "^(0|" αναγνωρίζει την ειδική περίπτωση του μηδέν. Η " [+-]?([1-9]\\d{0,17}|" καλύπτει το προαιρετικό πρόσημο και αριθμούς με 1 έως 18 ψηφία. Οι εκφράσεις "922337203685477580[0-7]|" και "-9223372036854775808)" αναγνωρίζουν τις μέγιστες και ελάχιστες τιμές που υποστηρίζει ο τύπος. Συνδυαστικά, αυτά δημιουργούν την κανονική έκφραση "^(0|[+-]?([1-9]\\d{0,17}|922337203685477580[0-7]|-9223372036854775808))$", η οποία αναγνωρίζει τιμές τύπου long, δηλαδή από -9.223.372.036.854.775.808 έως 9.223.372.036.854.775.807.

Για την αναγνώριση του τύπου short, η κανονική έκφραση "(\\+|-)?" καλύπτει το προαιρετικό πρόσημο. Η "0" αναγνωρίζει την ειδική περίπτωση του αριθμού μηδέν. Η κανονική έκφραση "[1-9]\\d{0,3}|[12]\\d{4}|3[0-1]\\d{3}|32[0-6]\\d{2}|327[0-5]\\d|3276[0-7]" καλύπτει τους αριθμούς από 1 έως 32.767, και η "-32768" αναγνωρίζει την ελάχιστη τιμή. Συνδυασμένα, αυτά δημιουργούν την κανονική έκφραση "^" +

SIGN_PART + "(" + ZERO_PART + "|" + SMALL_NUMBERS_PART + "|" + MIN_VALUE_PART + ")$", η οποία αναγνωρίζει τιμές τύπου short, δηλαδή από -32.768 έως 32.767.

Για την αναγνώριση του τύπου String, έχουμε την εξής λογική: Χρησιμοποιούνται κανονικές εκφράσεις (regex) για τον έλεγχο και την ανάλυση συμβολοσειρών. Συγκεκριμένα, η regex ".*[a-zA-Z].*" ελέγχει αν η συμβολοσειρά περιέχει τουλάχιστον ένα γράμμα (μικρό ή κεφαλαίο). Η regex ".*[\\w\\s\\p{Punct}].*" ελέγχει αν η συμβολοσειρά περιέχει και άλλους χαρακτήρες (όπως αριθμούς, σύμβολα στίξης, κενά διαστήματα). Ωστόσο, οι regex "^[0-9]+$" και "^[+-]?\\d*\\.\\d+$" χρησιμοποιούνται για να εξασφαλιστεί ότι η συμβολοσειρά δεν αποτελείται μόνο από αριθμούς ή δεκαδικό αριθμό, αντίστοιχα.

Για την αναγνώριση της μορφής ημερομηνίας και ώρας (Timestamp) σε κείμενο, χρησιμοποιούνται εκφράσεις που βοηθούν στον έλεγχο για το αν μια σειρά χαρακτήρων (string) ταιριάζει σε μια συγκεκριμένη μορφή ημερομηνίας και ώρας. Για παράδειγμα, για να ελέγξουμε αν μια συμβολοσειρά έχει τη μορφή YYYY-MM-DD HH:MM:SS (δηλαδή, έτος-μήνας-ημέρα ώρες:λεπτά:δευτερόλεπτα), χρησιμοποιούμε την εξής regex: ^\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}$. Αυτή η regex "λέει" ότι η συμβολοσειρά πρέπει να ξεκινά (^) με 4 ψηφία (\\d{4}) που αντιστοιχούν στο έτος, να ακολουθείται από μια παύλα (-), μετά 2 ψηφία (\\d{2}) για τον μήνα, άλλη μια παύλα (-), 2 ψηφία (\\d{2}) για την ημέρα, ένα κενό (), και τέλος την ώρα σε μορφή \\d{2}:\\d{2}:\\d{2} (ώρες:λεπτά:δευτερόλεπτα) και να τελειώνει εκεί ($). Αν θέλουμε να αναγνωρίσουμε μορφές όπως YYYY-MM-DD HH:MM:SS.SSS (με χιλιοστά του δευτερολέπτου), χρησιμοποιούμε την regex: ^\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}\\.\\d{3}$. Η μόνη διαφορά είναι η προσθήκη \\.\\d{3} για τα χιλιοστά του δευτερολέπτου (3 ψηφία μετά την τελεία). Για μορφές που περιλαμβάνουν τη ζώνη ώρας, όπως YYYY-MM-DD HH:MM:SS±HH:MM, χρησιμοποιούμε την regex: ^\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}[+-]\\d{2}:\\d{2}$. Εδώ, το [+-] σημαίνει "συν ή μείον" και το \\d{2}:\\d{2} αναπαριστά την ώρα και τα λεπτά της ζώνης ώρας. Αυτές οι regex ελέγχουν αν η μορφή της συμβολοσειράς είναι σωστή. Δεν ελέγχουν αν η ημερομηνία και η ώρα είναι πραγματικά έγκυρες (π.χ. δεν θα αποδεχτούν την 30η Φεβρουαρίου).

Για την αναγνώριση της μορφής ημερομηνίας (Date type) σε κείμενο, χρησιμοποιούμε κανονικές εκφράσεις (regex) που μας επιτρέπουν να ελέγξουμε αν μια συμβολοσειρά (string) αντιστοιχεί σε μια συγκεκριμένη μορφή ημερομηνίας. Η πρώτη μορφή που ελέγχεται είναι η YYYY-MM-DD, η οποία αντιστοιχεί σε έτος-μήνας-ημέρα. Η regex ^\\d{4}-\\d{2}-\\d{2}$ ελέγχει αν η συμβολοσειρά ξεκινά με 4 ψηφία για το έτος, ακολουθούμενα από μια παύλα, 2 ψηφία για τον μήνα, άλλη μια παύλα και 2 ψηφία για

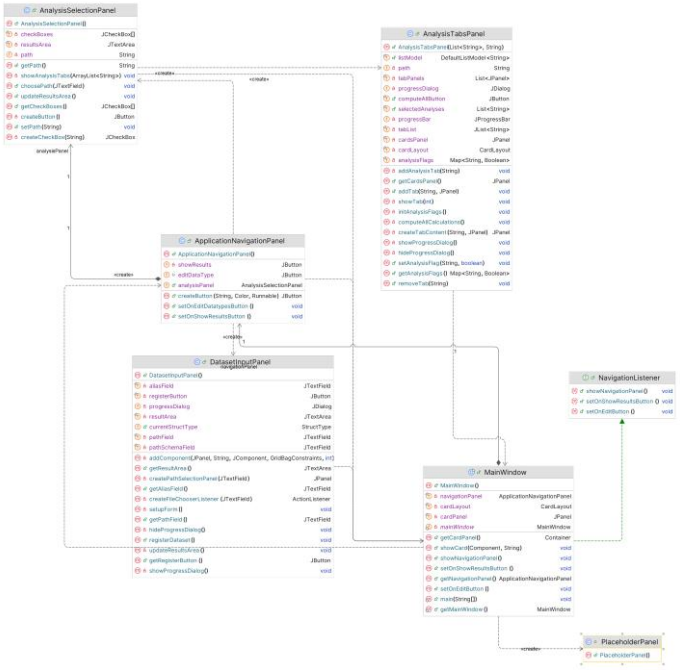
την ημέρα. Η δεύτερη μορφή είναι η DD-MM-YYYY, η οποία αντιστοιχεί σε ημέρα-μήνας-έτος. Η regex ^\\d{2}-\\d{2}-\\d{4}$ ελέγχει αν η συμβολοσειρά ξεκινά με 2 ψηφία για την ημέρα, ακολουθούμενα από μια παύλα, 2 ψηφία για τον μήνα, άλλη μια παύλα και 4 ψηφία για το έτος. Η τρίτη μορφή είναι η DD/MM/YYYY, η οποία αντιστοιχεί σε ημέρα/μήνας/έτος. Η regex ^\\d{2}/\\d{2}/\\d{4}$ ελέγχει αν η συμβολοσειρά ξεκινά με 2 ψηφία για την ημέρα, ακολουθούμενα από μια κάθετο, 2 ψηφία για τον μήνα, άλλη μια κάθετο και 4 ψηφία για το έτος. Η τέταρτη μορφή είναι η DD/MM/YY, η οποία αντιστοιχεί σε ημέρα/μήνας/έτος με δύο ψηφία για το έτος. Η regex ^\\d{2}/\\d{2}/\\d{2}$ ελέγχει αν η συμβολοσειρά ξεκινά με 2 ψηφία για την ημέρα, ακολουθούμενα από μια κάθετο, 2 ψηφία για τον μήνα, άλλη μια κάθετο και 2 ψηφία για το έτος. Η πέμπτη μορφή είναι η MM-DD-YY, η οποία αντιστοιχεί σε μήνας-ημέρα-έτος με δύο ψηφία για το έτος. Η regex ^\\d{2}-\\d{2}-\\d{2}$ ελέγχει αν η συμβολοσειρά ξεκινά με 2 ψηφία για τον μήνα, ακολουθούμενα από μια παύλα, 2 ψηφία για την ημέρα, άλλη μια παύλα και 2 ψηφία για το έτος. Η έκτη μορφή είναι η YYYY/MM/DD, η οποία αντιστοιχεί σε έτος/μήνας/ημέρα. Η regex ^\\d{4}/\\d{2}/\\d{2}$ ελέγχει αν η συμβολοσειρά ξεκινά με 4 ψηφία για το έτος, ακολουθούμενα από μια κάθετο, 2 ψηφία για τον μήνα, άλλη μια κάθετο και 2 ψηφία για την ημέρα. Η έβδομη μορφή είναι η DD Month YYYY, η οποία αντιστοιχεί σε ημέρα, μήνας σε γράμματα και έτος. Η regex ^(0[1-9]|[12][0-9]|3[01])\\s(January|February|March|April|May|June|July|August|September|October|November|December)\\s\\d{4}$ ελέγχει αν η συμβολοσειρά ξεκινά με 1 ή 2 ψηφία για την ημέρα (0[1-9] για τις ημέρες 01-09, [12][0-9] για τις ημέρες 10-29 και 3[01] για τις ημέρες 30-31), ακολουθούμενα από ένα κενό, τον μήνα σε γράμματα (January, February, κ.λπ.), άλλο ένα κενό και 4 ψηφία για το έτος. Η όγδοη μορφή είναι η Month DD, YYYY, η οποία αντιστοιχεί σε μήνας σε γράμματα, ημέρα και έτος. Η regex ^(January|February|March|April|May|June|July|August|September|October|November|December)\\s(0[1-9]|[12][0-9]|3[01]),\\s\\d{4}$ ελέγχει αν η συμβολοσειρά ξεκινά με τον μήνα σε γράμματα, ακολουθούμενο από ένα κενό, 1 ή 2 ψηφία για την ημέρα, ένα κόμμα, άλλο ένα κενό και 4 ψηφία για το έτος. Αυτές οι regex ελέγχουν αν η μορφή της συμβολοσειράς είναι σωστή. Ωστόσο, δεν ελέγχουν αν η ημερομηνία είναι πραγματικά έγκυρη (π.χ. θα αποδεχτούν την 30η Φεβρουαρίου). Έχουν όμως προστεθεί έλεγχοι που ελέγχουν αν η ημερομηνία είναι έγκυρη πριν ενημερώσουν το score για την κανονική έκφραση.

Για τον συγχρονισμό όμως όλου αυτού του υποσυστήματος δημιουργήθηκε η κλάση ScoreCalculatorManager. Για κάθε στήλη, δημιουργεί ένα υποσύνολο δεδομένων και υπολογίζει βαθμολογίες για πιθανούς τύπους δεδομένων. Τα score αποθηκεύονται στο

map scoresPerColumnMap, όπου το κλειδί είναι το όνομα της στήλης και η τιμή είναι ένα map με τους τύπους δεδομένων και τα αντίστοιχα score τους.

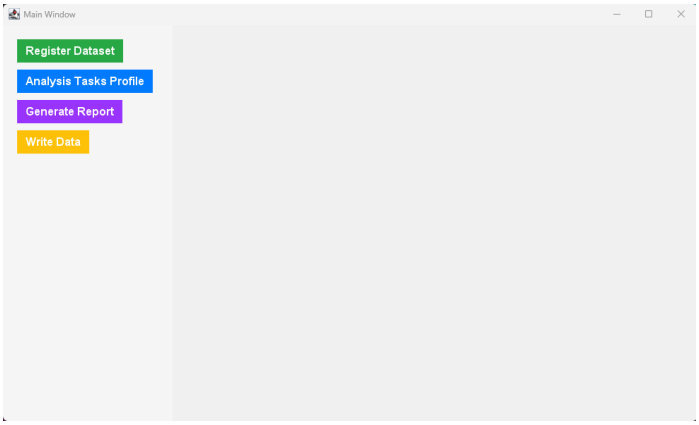
3.2.6 Το πακέτο gui

Το πακέτο αυτό δημιουργήθηκε για να προσφέρει γραφική διεπαφή στο χρήστη ώστε να μπορεί να χρησιμοποιεί την Pythia μέσω αυτής χωρίς να απαιτούνται προγραμματικές ικανότητες. Το πακέτο περιλαμβάνει 4 υπό πακέτα. Για το πρώτο πακέτο με όνομα guiButtonPanesl φαίνεται παρακάτω η σχεδίαση:



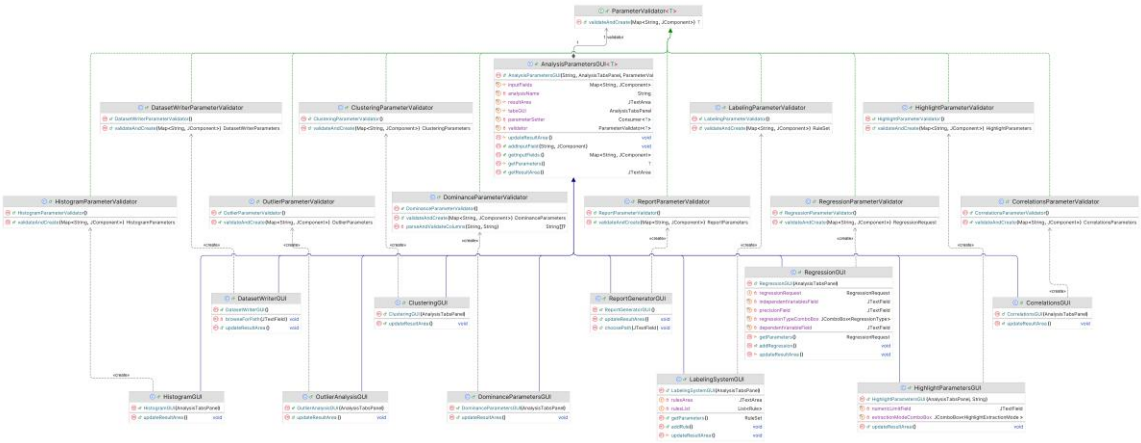
Εικόνα 32 guiButtonPanel package diagram

Το πακέτο είναι υπεύθυνο για την δημιουργία του κεντρικού παραθύρου της εφαρμογής προσφέροντας στο χρήστη τις επιλογές που υποστηρίζει η εφαρμογή. Παρακάτω φαίνεται το παράθυρο που εμφανίζεται στο χρήστη:



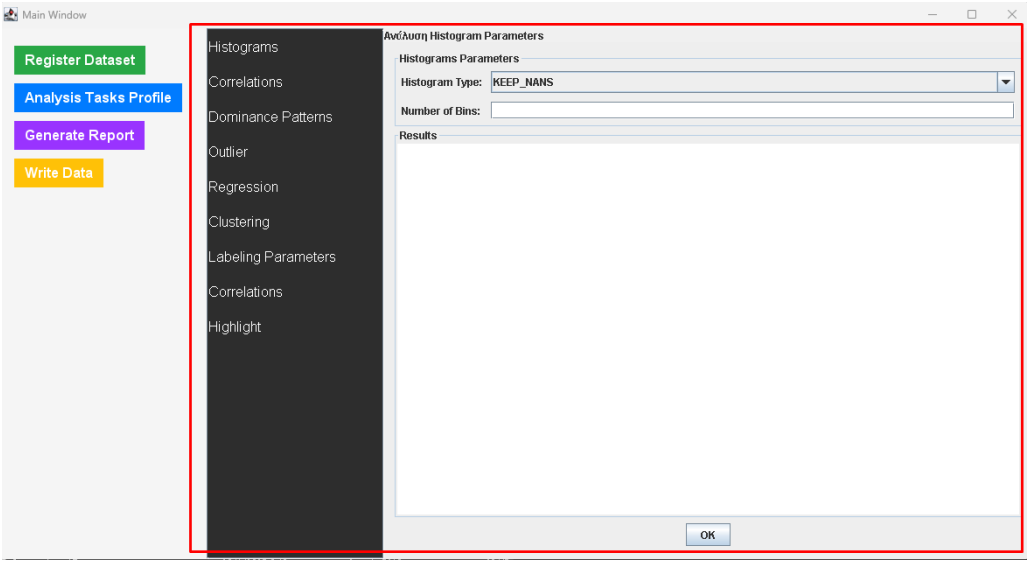
Εικόνα 33 Παράθυρο εμφάνισης στο χρήστη

Για το δεύτερο πακέτο με όνομα analysisTasksGuiPanels η σχεδίαση του είναι η παρακάτω:



Εικόνα 34 analysisTasksGuiPanels package uml diagram

Το πακέτο αυτό είναι υπεύθυνο στο να δημιουργήσει καρτέλες για κάθε ανάλυση που υποστηρίζει η Pythia με σκοπό ο χρήστης να εισάγει τις παραμέτρους για κάθε ανάλυση που θέλει. Το κεντρικό παράθυρο μετατρέπεται όπως φαίνεται παρακάτω:



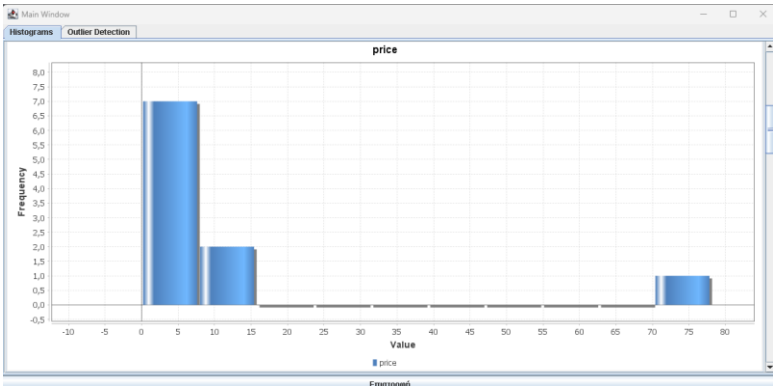
Εικόνα 35 Παράθυρο εμφάνισης καρτελών στον χρήστη για εισαγωγή παραμέτρων ανάλυσης

Το τρίτο πακέτο resultsGui που υπάρχει είναι υπεύθυνο στο να δημιουργήσει καρτέλες εμφάνισης αποτελεσμάτων και η σχεδίαση του είναι η παρακάτω:

Εικόνα 36 resultsGui package uml diagram

Το πακέτο αυτό είναι υπεύθυνο για την δημιουργία καρτελών που έχουν να κάνουν με τα αποτελέσματα των αναλύσεων που επιλέχθηκαν από το χρήστη. Το πακέτο δημιουργεί ιστογράμματα, outlier results με την μορφή scater plot, ενώ για τις υπόλοιπες αναλύσεις δημιουργεί πίνακες με τα αποτελέσματα της ανάλυσης. Όσο αναφορά τα αποτελέσματα των decisionTrees που υπολογίζονται, τα αποτελέσματα προβάλλονται σε μορφή εικόνων σε ένα παράθυρο.

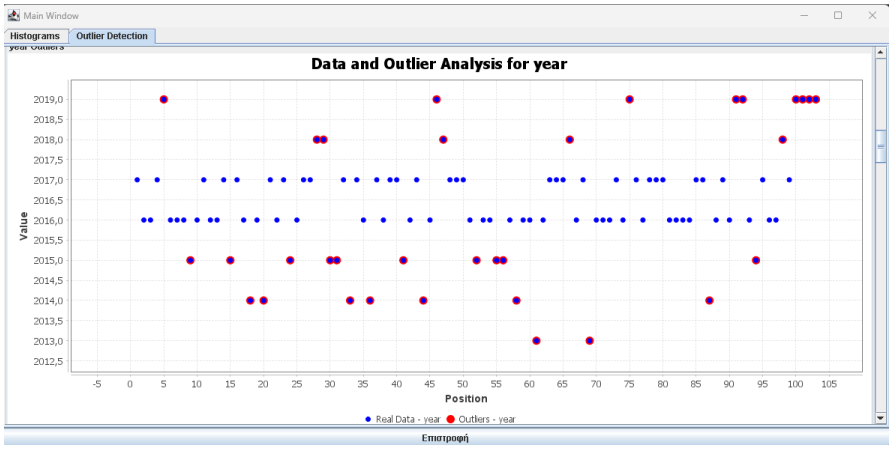
Τα ιστογράμματα φαίνονται όπως παρακάτω:



Εικόνα 37 Παράδειγμα προβολής ιστογραμμάτων

Το διάγραμμα που δημιουργείται είναι ένα ιστόγραμμα (histogram), το οποίο απεικονίζει την κατανομή τιμών μιας στήλης του dataset χρησιμοποιώντας JFreeChart. Στον άξονα X εμφανίζονται οι τιμές των δεδομένων ομαδοποιημένες σε διαστήματα (bins), ενώ στον άξονα Y η συχνότητα εμφάνισης κάθε διαστήματος. Τα δεδομένα φορτώνονται από την κλάση HistogramDataset, και το διάγραμμα προσαρμόζεται με XYPlot και XYBarRenderer για καλύτερη οπτικοποίηση. Υποστηρίζονται τόσο τα κανονικά ιστογράμματα όσο και τα ιστογράμματα τεταρτημόριων, ενώ η κύλιση μέσω JScrollPane επιτρέπει την εύκολη προβολή πολλαπλών διαγραμμάτων.

Για την προβολή των outlier αποτελεσμάτων τα scatterplot προβάλουν τις πραγματικές θέσεις και τις θέσεις των outlier που υπολογίζονται όπως φαίνεται παρακάτω:

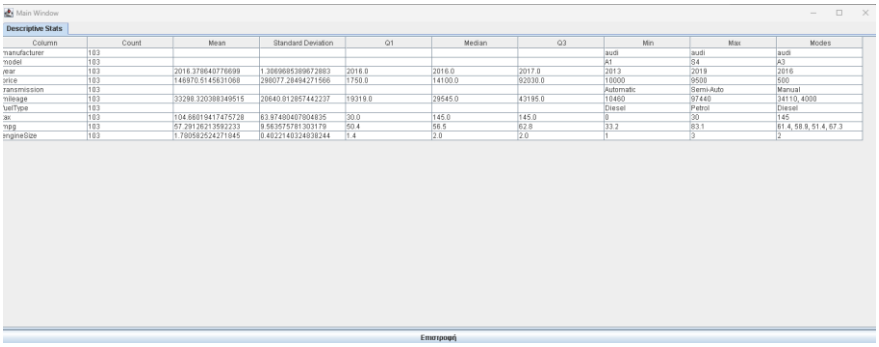


Εικόνα 38 Παράδειγμα προβολής outlier

Το διάγραμμα scatter plot (XY) δημιουργείται με τη JFreeChart και απεικονίζει τις τιμές μιας στήλης του dataset. Στον άξονα X εμφανίζεται η θέση κάθε τιμής στο dataset, ενώ στον άξονα Y η αντίστοιχη αριθμητική τιμή. Οι πραγματικές τιμές σχεδιάζονται ως μπλε κύκλοι, ενώ οι τιμές outliers επισημαίνονται με κόκκινους, μεγαλύτερους κύκλους για ευκολότερη διάκριση. Το διάγραμμα δημιουργείται μέσω της XYSeriesCollection και προσαρμόζεται οπτικά με XYPlot και XYLineAndShapeRenderer. Επιπλέον,

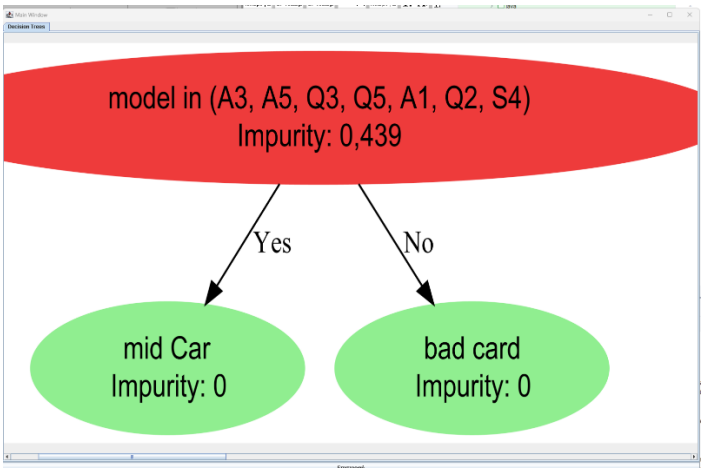
χρησιμοποιείται JScrollPanel για την υποστήριξη κύλισης όταν υπάρχουν πολλές στήλες προς ανάλυση.

Για τις υπόλοιπες αναλύσεις εμφανίζονται πίνακες πχ, σαν και το παρακάτω που αφορά υπολογισμό Descriptive Statictics:



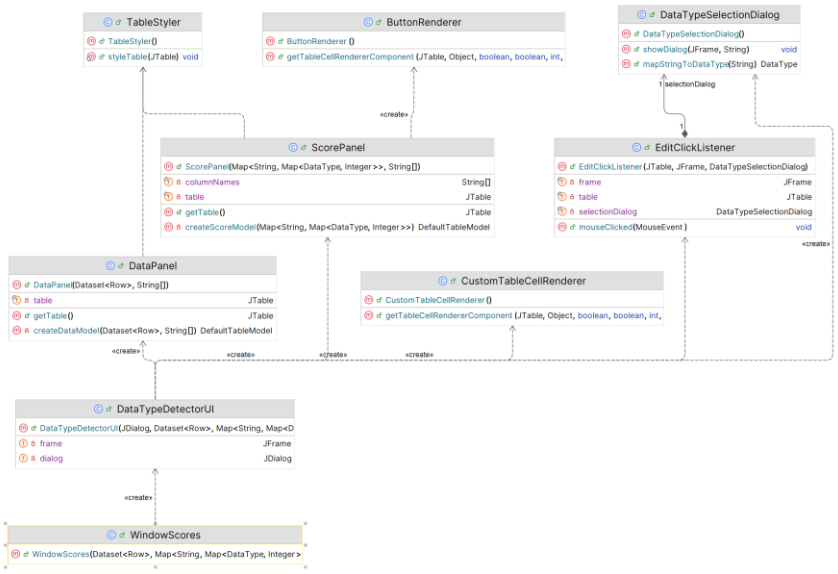
Εικόνα 39 Παράδειγμα προβολής πινάκων για την προβολή αποτελεσμάτων Descriptive Statistics

Ένα παράδειγμα παρουσίασης δέντρων αποφάσεων είναι το παρακάτω:



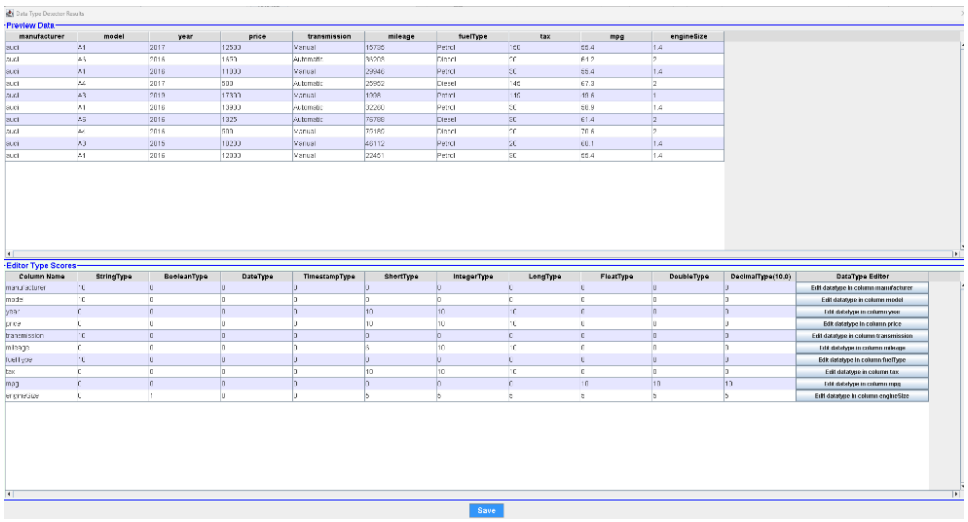
Εικόνα 40 Παράδειγμα προβολής εικόνων για αποτελέσματα decision Trees

Και τέλος το τελευταίο που περιέχει το πακέτο gui είναι το πακέτο guiScores, το οποίο παρουσιάζει τα αποτελέσματα των score στο χρήστη δίνοντάς του την δυνατότητα να καθορίσει αυτός τον τύπο για κάθε στήλη. Παρακάτω φαίνεται η σχεδίαση του πακέτου αυτού:



Εικόνα 41 guiScores package uml diagram

Το πακέτο αυτό περιλαμβάνει κλάσεις που είναι υπεύθυνες να δημιουργήσουν μια γραφική διεπαφή με το χρήστη προβάλλοντάς του σε ένα panel 2 πίνακες, έναν πίνακα με 50 εγγραφές που δείχνουν κάποια από τα δεδομένα που υπάρχουν στο αρχείο που ανέβασε ο χρήστης για δημιουργία προφίλ και έναν πίνακα με τα σκορ για κάθε τύπο δεδομένων και κάθε στήλη. Επίσης στο πακέτο αυτό περιλαμβάνονται και κλάσεις που ορίζουν το styling των πινάκων. Στον χρήστη ο πίνακας φαίνεται όπως παρακάτω:

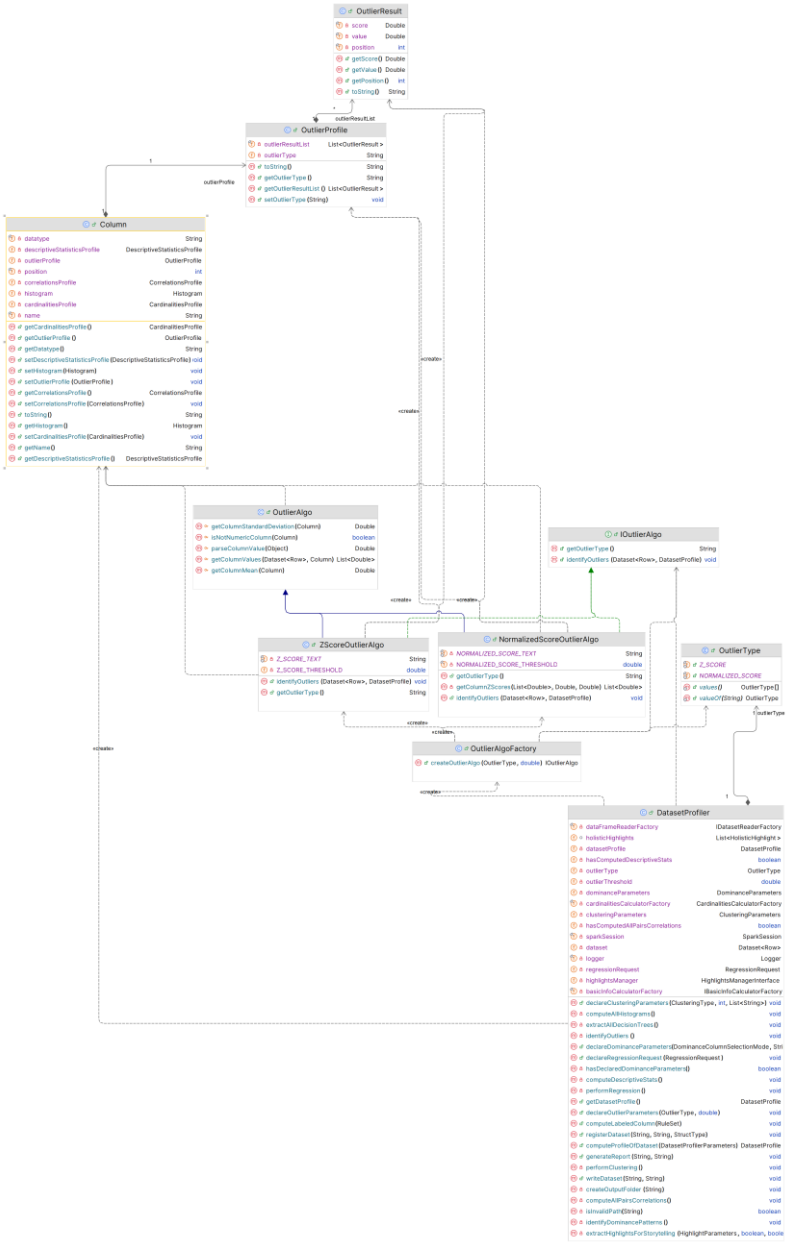


Εικόνα 42 Παράδειγμα παραθύρου για αποτελέσματα score και τροποποίηση από το χρήστη του τύπου των στηλών.

Ο χρήστης μπορεί να επιλέξει να χαρακτηρίσει τον τύπο για κάθε στήλη ανάλογα με τα score που βλέπει.

3.2.7 Το πακέτο outliers

Για την ανακατασκευή του outlier package η ιδέα είναι ότι πρέπει να είναι ανεξάρτητο να μην εμπλέκεται σε άλλους υπολογισμούς αλλά και κάθε στήλη να περιέχει πληροφορίες σχετικές με τα outlier του. Για την ανακατασκευή του πακέτου αρχικά το πακέτο outlier όπως περιεγράφηκε παραπάνω στην 3.1.2 ενότητα αφαιρέθηκε από το πακέτο patterns όπου βρισκόταν καθώς και οι εξαρτήσεις του από αυτό. Η νέα σχεδίαση του είναι η παρακάτω:



Εικόνα 43 outlier package uml diagram

Το νέο ανεξάρτητο πακέτο είναι τροποποιημένο με τις προσθήκες του OutlierResult model class όπου σε αυτή την κλάση περιέχονται οι πληροφορίες που αφορούν τα outlier

της στήλης καθώς και τον τύπο-αλγόριθμο υπολογισμού αυτών. Πλέον κρατείτε αντικείμενο στην model class Column και καταγράφεται η πληροφορία των outliers για κάθε στήλη που είναι το επιθυμητό. Επιπλέον χρειάστηκε να τροποποιηθεί η identifyOutliers() της κλάσης DatasetProfiler στο Engine πακέτο με τον παρακάτω τρόπο

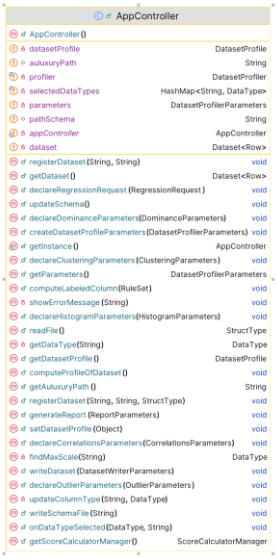
```
/*
private void identifyOutliers() throws IOException {
    Instant start = Instant.now();

    if (!hasComputedDescriptiveStats) computeDescriptiveStats();
    if (!hasComputedAllPairsCorrelations) computeAllPairsCorrelations();
    IPatternManagerFactory factory = new IPatternManagerFactory();
    IPatternManager patternManager = factory.createPatternManager(
            dataset, datasetProfile, dominanceParameters, outlierType, outlierThreshold);
    patternManager.identifyOutliers();
    logger.info(String.format("Identified outliers for dataset %s", datasetProfile.getAlias()));

    Instant end = Instant.now();
    Duration duration = Duration.between(start, end);
    logger.info(String.format("Duration of identifyOutliers: %s / %sms", duration, duration.toMillis()));
}*/

private void identifyOutliers() throws IOException {  1 usage   ± lamprosvlax13 +1
    Instant start = Instant.now();
    if (!hasComputedDescriptiveStats) computeDescriptiveStats();
    OutlierAlgoFactory factory = new OutlierAlgoFactory();

    //Todo check this call.
    // in case who have forgot call  declareOutlierParameters in clients , we take nulls. e.g not initialize threshold
    //declareOutlierParameters(outlierType,outlierThreshold);

    IOutlierAlgo algoOutlier = factory.createOutlierAlgo(this.outlierType,this.outlierThreshold);
    algoOutlier.identifyOutliers(this.dataset, this.datasetProfile);
    logger.info(String.format("Identified outliers for dataset %s", datasetProfile.getAlias()));
    Instant end = Instant.now();
    Duration duration = Duration.between(start, end);
    logger.info(String.format("Duration of identifyOutliers: %s / %sms", duration, duration.toMillis()));
}


private void performRegression() {  1 usage   ± georgekarathanos +1
    Instant start = Instant.now();
```

Εικόνα 44 engine/DatasetProfile.java

Πλέον καλείτε αυτόνομα η διαδικασία εύρεσης των outlier χωρίς την εξάρτηση τους από το pattern package.

3.2.8 Το πακέτο appController

Το πακέτο αυτό δημιουργήθηκε για να αποτελέσει τον συνδετικό κρίκο μεταξύ του gui και του backend της pythias. Αποτελείται από μία κλάση AppController που σκοπό έχει να ενημερώνει, καλεί τα απαραίτητα υποσυστήματα ανάλογα με τις επιλογές του χρήστη. Παρακάτω φαίνεται η σχεδίαση του



Εικόνα 45 appController uml diagram

3.2.9 Επεξεργασία αναφορών Reports

Όπως αναφέρθηκε στην ενότητα 3.1.2 το πακέτο report περιλαμβάνει το υποσύστημα παραγωγής αναφορών με τα στατιστικά και τους υπολογισμούς που εκτελεί η Pythia. Σε αυτή την ενότητα θα αναλυθούν οι τροποποιήσεις που χρειάστηκαν σε αυτό το πακέτο αλλά και σε κάποιες model class ώστε οι αναφορές να συμπεριλαμβάνουν τις επεκτάσεις κώδικα που σχεδιάστηκαν και αναλυθήκαν στις ενότητες 3.2.1 ,3.2.2 , 3.2.3, 3.2.4, 3.4.7.

Για την προσθήκη των πληροφοριών που υπολογίζονται στο πακέτο generalinfo τροποποιήθηκε η model class DatasetProfile στην μέθοδό της toString(), που επιστρέφει ένα String με τις πληροφορίες του profile. Παρακάτω φαίνονται η τροποποίηση:

```
@Override  ▲ lamprosvlax13 +2
public String toString() {
  StringBuilder stringBuilder = new StringBuilder();
  for (Column column : columns) {
    stringBuilder.append(column.toString());
  }
  return "DatasetProfile\n" +
          "Alias: " + alias + "\n" +
          "Path: " + path + "\n" +
          "Number of Lines: " + numberOfLines + "\n" +
          "File Size: " + fileSize +" MB" +"\n" +
          "Timestamp: " + timestamp +" "+zoneId +"\n\n" +
          "Column Profiles:\n" + stringBuilder;
}
```

Εικόνα 46 model/DatasetProfile.java , μέθοδος toString()

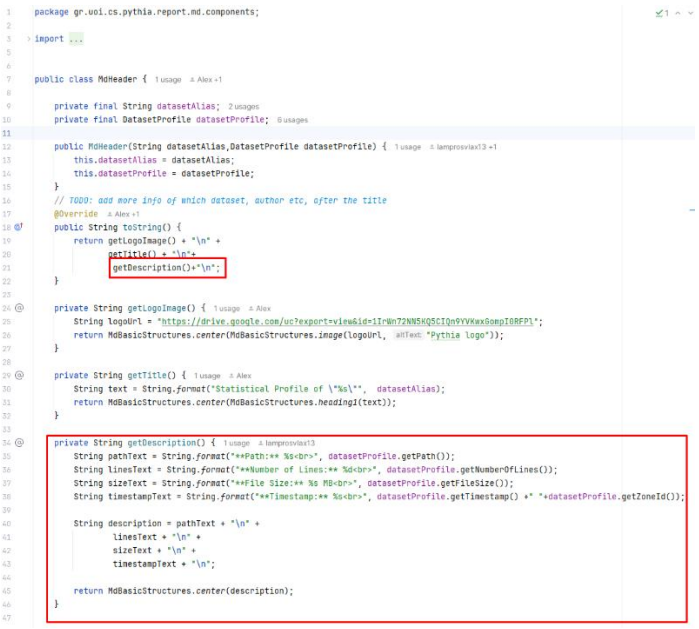
Αυτή η τροποποίηση αρκεί για να συμπεριλάβει στο αρχείο statistical_report.txt (παράγεται από το report πακέτο) τα υπολογισμένα δεδομένα καθώς για την δημιουργία αυτού του, η Pythia κάνει χρήση της toString() που τροποποιήθηκε όπως φαίνεται παρακάτω:

```
private void produceStatisticalProfileReport(DatasetProfile datasetProfile,  1 usage  ch-ant
        String outputDirectoryPath) throws IOException {
    writeToFile(outputDirectoryPath, statisticalReportFileName, datasetProfile.toString());
}
```

Εικόνα 47 report/TxtReportGenerator.java , μέθοδος produceStatisticalProfileReport()

Η μέθοδος αυτή βρίσκεται στο report πακέτο στην κλάση TxtReportGenerator και είναι υπεύθυνη για την δημιουργία του αρχείου.

Για να συμπεριληφθούν όμως στο αρχείο statistical_report.md , τροποποιήθηκε η κλάση MdHeader του πακέτου report/md/components ως εξής :



```
package gr.uoi.cs.pythia.report.md.components;

import ...

public class MdHeader {  1 usage  Alex +1

    private final String datasetAlias;  2 usages
    private final DatasetProfile datasetProfile;  6 usages

    public MdHeader(String datasetAlias,DatasetProfile datasetProfile) {  1 usage  lamprosvlax13 +1
        this.datasetAlias = datasetAlias;
        this.datasetProfile = datasetProfile;
    }
    // TODO: add more info of which dataset, author etc, after the title
    @Override  Alex +1
    public String toString() {
        return getLogoImage() + "\n" +
                getTitle() + "\n" +
                getDescription()+"\n";
    }

    private String getLogoImage() {  1 usage  Alex
        String logoUrl = "https://drive.google.com/uc?export=view&id=1IrWn72NN5KQ5CIQn9YVKwx9ompI0RPP1";
        return MdBasicStructures.center(MdBasicStructures.image(logoUrl,  altText: "Pythia logo"));
    }

    private String getTitle() {  1 usage  Alex
        String text = String.format("Statistical Profile of \"%s\"",  datasetAlias);
        return MdBasicStructures.center(MdBasicStructures.heading1(text));
    }

    private String getDescription() {  1 usage  lamprosvlax13
        String pathText = String.format("**Path:** %s<br>", datasetProfile.getPath());
        String linesText = String.format("**Number of Lines:** %d<br>", datasetProfile.getNumberOfLines());
        String sizeText = String.format("**File Size:** %s MB<br>", datasetProfile.getFileSize());
        String timestampText = String.format("**Timestamp:** %s<br>", datasetProfile.getTimestamp() +" "+datasetProfile.getZoneId());

        String description = pathText + "\n" +
                linesText + "\n" +
                sizeText + "\n" +
                timestampText + "\n";

        return MdBasicStructures.center(description);
    }
```

Εικόνα 48 report/md/components/MdHeader.java

Προστέθηκε όπως φαίνεται η μέθοδος getDescription() , η οποία μορφοποιεί κατάλληλα τις πληροφορίες κάνοντας χρήση υπάρχουσες δομές της Pythias MdBasicStrutures και συμπληρώνει το header του report με τις υπολογισμένες τιμές.

Για την προσθήκη των δεδομένων που υπολογίζονται στο πακέτο cardinalities που αναλύθηκε στην 3.2.2 ενότητα τροποποιήθηκε αρχικά η model class Column όπως φαίνεται παρακάτω :

Εικόνα 49 model/Column.java

Ενημερώνεται η toString() με το cardinalitieProfile και ενσωματώνει τις πληροφορίες του. Αυτή η τροποποίηση είναι αρκετή για να συμπεριλάβει στο αρχείο statistical_report.txt τις πληροφορίες που αφορούν τις null, διακριτές τιμές για κάθε στήλη. Αντίθετα για την ενσωμάτωση των πληροφοριών στο statistical_report.md χρειάστηκε η δημιουργία της κλάσης MdCardinalities στο πακέτο report/md/components όπως φαίνεται παρακάτω:

Εικόνα 50 report/md/components/MdCardinalities.java

Η κλάση αυτή είναι υπεύθυνη για την δημιουργίας ενός πίνακα που περιέχει για κάθε στήλη τις πληροφορίες των null, διακριτών τιμών. Κάνει χρήση υπαρχόντων δομών της Pythias όπως MdBasicStructures, MdTable και επιστρέφει μέσω της toString() τον πίνακα σε μορφή MarkDown για να προστεθεί στην αναφορά. Ωστόσο για να ενσωματωθεί στην αναφορά χρειαζόταν η τροποποίηση στην κλάση MdReportGenerator του πακέτου report στην μέθοδο getReportString() όπως φαίνεται παρακάτω:

```
private String getReportString(DatasetProfile datasetProfile) {  1 usage   Alex +2
    StringBuilder bobOMastoras = new StringBuilder();
    bobOMastoras.append(new MdHeader(datasetProfile.getAlias(),datasetProfile));
    bobOMastoras.append(new MdCardinalities(datasetProfile));
    bobOMastoras.append(new MdDescriptiveStatistics(datasetProfile.getColumns()));
    //Todo check it
    // move outliers separate file
    bobOMastoras.append(new MdOutlierStatistics(datasetProfile));

    bobOMastoras.append(new MdCorrelations(datasetProfile.getColumns()));
    bobOMastoras.append(new MdDecisionTrees(datasetProfile));
    bobOMastoras.append(new MdHistograms(datasetProfile.getColumns()));
    return bobOMastoras.toString();
}
```

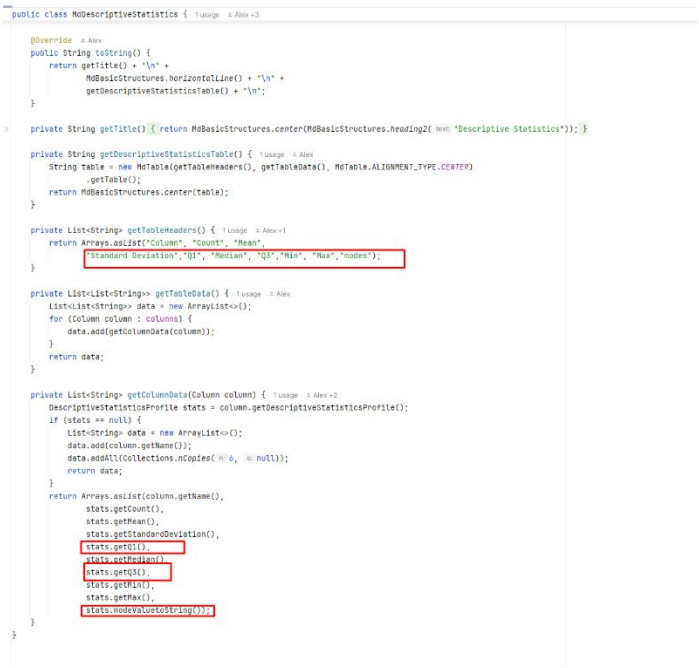
Εικόνα 51 report/MdReportGenerator.java

Για την προσθήκη των νέων δεδομένων που περιγράφτηκαν στην ενότητα 3.2.3 και αφορά των υπολογισμό των q1, q2 ,q3 , mode τιμές ενημερώθηκε κατάλληλα η κλάση DescriptiveStatisticsProfile που κρατά αυτές τις πληροφορίες και για την εισαγωγή αυτών στα report χρειάστηκε τροποποίηση της μεθόδου αρχικά toString() όπως φαίνεται παρακάτω:

```
public String modeValuetoString(){  3 usages   lamprosvlax13
  StringBuilder modeValuesAsString = new StringBuilder();
  if (modeList == null || modeList.isEmpty()) {
    modeValuesAsString = new StringBuilder("null");
  }
  else{
    for(String mode: modeList){
      modeValuesAsString.append(mode).append(", ");
    }
    if (modeValuesAsString.length() > 0) {
      modeValuesAsString.setLength(modeValuesAsString.length() - 2);
    }
  }
  return modeValuesAsString.toString();
}

@Override   AlexandrosAlexiou +1
public String toString(){
  return "count: "
          +count
          +'\n'
          +"mean: "
          +mean
          +'\n'
          +"standardDeviation: "
          +standardDeviation
          +'\n'
          +"q1: "
          +q1
          +'\n'
          +"median: "
          +median
          +'\n'
          +"q3: "
          +q3
          +"\n"
          +"min: "
          +min
          +'\n'
          +"max: "
          +max
          +'\n'
          +"mode: "
          +modeValuetoString()
```

Εικόνα 52 DescriptiveStatisticsProfile.java

Η μέθοδος πλέον διατηρεί και τις πληροφορίες που αφορούν τις νέες υπολογισθείσες τιμές. Αυτή η αρχική τροποποίηση είναι αρκετή ώστε να συμπεριληφθούν στις παραγόμενες αναφορές μορφής Txt. Ωστόσο για να συμπεριληφθούν στις αναφορές markdown χρειάστηκε τροποποίηση η κλάση MdDescriptiveStatistics όπως φαίνεται παρακάτω:



Εικόνα 53 MdDescriptiveStatistics.java

Ενημερώνεται με τις πληροφορίες που προστέθηκαν.

Για την προσθήκη των ισοϋψών διαγραμμάτων που αναλύθηκαν στην ενότητα 3.2.4 στις αναφορές μορφής Txt χρειάστηκε απλά τροποποίηση η μέθοδος toString() της model class Column όπως φαίνεται παρακάτω:

Εικόνα 54 Column.java

Η τροποποίηση αυτή απλά προσθέτει την πληροφορία του ισοϋψούς διαγράμματος.

Ενώ για να συμπεριληφθούν στις αναφορές μορφής markdown χρειάστηκε η τροποποίηση της κλάσης MdHistograms όπως φαίνεται παρακάτω:

Εικόνα 55 MdHistograms.java

Γίνεται προσθήκη ενός πίνακα που περιέχει και τις πληροφορίες για το νέο ισοϋψές ιστόγραμμα.

Για την προσθήκη των τροποποιήσεων του πακέτου outliers στις αναφορές που αναλύθηκε στην ενότητα 3.2.7 χρειάστηκαν τροποποιήσεις που αφορούν την toString() μέθοδο της model class Column όπως φαίνεται παρακάτω:

Εικόνα 56 Column.java

Προστίθενται η πληροφορία σχετικά με τα την ανάλυση των outlier για την συγκεκριμένη στήλη των δεδομένων και αυτό είναι αρκετό ώστε να συμπεριλάβει τα αποτελέσματα στις αναφορές μορφής .Txt. Για να συμπεριληφθούν σε αναφορές markdown χρειάστηκε η δημιουργία κλάσης MdOutlierStatistics η οποία φαίνεται παρακάτω:

```
package gr.uoi.cs.pythia.report.md.components;

import ...

public class MdOutlierStatistics {  1 usage   ± lamprosvlax13 *
    private final List<Column> columns;  2 usages
    public MdOutlierStatistics(DatasetProfile datasetProfile) { this.columns = datasetProfile.getColumns(); }

    @Override  ± lamprosvlax13 *
    public String toString() {
        String tableData = getOutlierStatisticsTable();
        return getTitle() + "\n" +
                MdBasicStructures.horizontalLine() + "\n" +
                tableData + "\n";
    }
    private String getTitle() { return MdBasicStructures.center(MdBasicStructures.heading2( text: "Outliers Statistics")); }
    private String getOutlierStatisticsTable() {  1 usage   ± lamprosvlax13
        List<List<String>> listData = getTableData();
        if (!listData.isEmpty()) {
            String table = new MdTable(getTableHeaders(), listData, MdTable.ALIGNMENT_TYPE.CENTER).getTable();
            return MdBasicStructures.center(table);
        }
        String notFound = "Not found Outlier Statistics";
        return MdBasicStructures.center(notFound);
    }
    private List<String> getTableHeaders() {  1 usage   ± lamprosvlax13
        return Arrays.asList("Column","outlierType", "value", "score",
                "position");
    }
    private List<List<String>> getTableData() {  1 usage   ± lamprosvlax13
        List<List<String>> data = new ArrayList<>();
        for (Column column : columns) {
            List<List<String>> columnData = getColumnData(column);
            if (!columnData.isEmpty()) {
                data.addAll(columnData);
            }
        }
        return data;
    }
    private List<List<String>> getColumnData(Column column) {  1 usage   ± lamprosvlax13 *
        List<List<String>> rows = new ArrayList<>();
        OutlierProfile outlierProfile = column.getOutlierProfile();
        if (outlierProfile == null) {
            return Collections.emptyList();
        }
        List<OutlierResult> outlierResults = outlierProfile.getOutlierResultList();
        for (OutlierResult result : outlierResults) {
            List<String> rowData = new ArrayList<>();
            rowData.add(column.getName());
            rowData.add(outlierProfile.getOutlierType());
            rowData.add(String.valueOf(result.getValue()));
            rowData.add(String.valueOf(result.getScore()));
            rowData.add(String.valueOf(result.getPosition()));
            rows.add(rowData);
        }
        return rows;
    }
```

Εικόνα 57 MdOutlierStatistics.java

Η κλάση αυτή θα δημιουργήσει τους κατάλληλους πίνακες με τα υπολογισμένα δεδομένα τα οποία θα ενσωματωθούν στο αρχείο markdown.

3.2.10 Λοιπές προσθήκες

Για την αξιοποίηση του JFreeChart και της βιβλιοθήκης JSON στη Pythia, προσθέτουμε τις ακόλουθες εξαρτήσεις στο αρχείο pom.xml:

Εικόνα 58 pom.xml

Αυτό επιτρέπει την αυτόματη λήψη και διαχείριση των βιβλιοθηκών αυτόματα από το maven.

Επιπλέον για την λήψη των παραμέτρων που αφορούν την δημιουργία ιστογράμματος από το χρήστη δημιουργείτε η κλάση HistogramParameters.

3.3 Σχεδίαση και αποτελέσματα ελέγχου του λογισμικού

Η ενότητα αυτή περιλαμβάνει τους ελέγχους ορθής λειτουργίας του συστήματος, την μέθοδο με την οποία ελέγχεται η ορθότητα αλλά και η αναλυτική παρουσίαση των ελέγχων που πραγματοποιήθηκαν.

3.3.1 Μεθοδολογία ελέγχου

Η συγγραφή λογισμικού δεν αρκεί από μόνη της για να διασφαλίσει την ποιότητα και την αξιοπιστία του συστήματος. Απαραίτητος είναι και ο συστηματικός έλεγχος μέσω test. Τα test βοηθούν στον έγκαιρο εντοπισμό σφαλμάτων και διευκολύνουν τη μελλοντική συντήρηση του κώδικα. Μέσω της αυτοματοποίησης, εξασφαλίζεται η σταθερότητα του συστήματος, επιτρέποντας ασφαλείς αλλαγές και βελτιώσεις. Έτσι και η Pythia δε θα μπορούσε να μην καλύπτεται από ελέγχους ορθής λειτουργίας. Πιο συγκεκριμένα η Pythia περιλαμβάνει πακέτο ελέγχων που ελέγχουν το κάθε ένα ξεχωριστά, τις επιμέρους λειτουργίες και κρίσιμους υπολογισμούς που εκτελεί. Αποτελείται από ελέγχους, οι οποίοι αφού έχουν έτοιμα δεδομένα εισόδου από τον προγραμματιστή, τα προωθούν στα υποσυστήματα και περιμένουν την έξοδο. Αν η έξοδος ταυτίζεται με τα δεδομένα εισόδου(αναμενόμενες τιμές) τότε έχουμε ορθή λειτουργία, σε αντίθετη περίπτωση η λειτουργία του υποσυστήματος που λαμβάνει τα δεδομένα είναι εσφαλμένη.

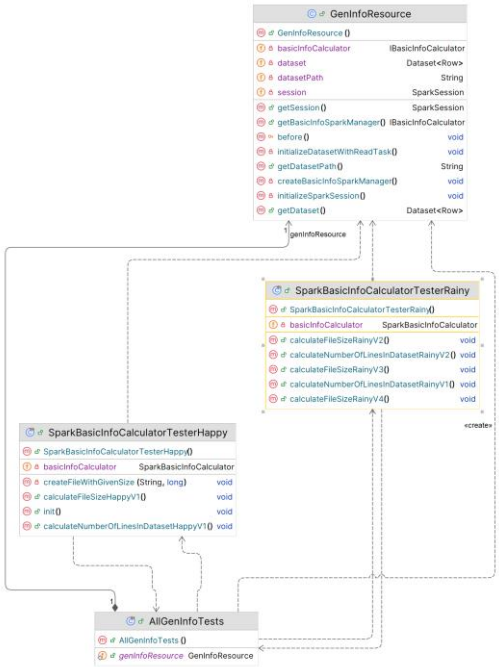
3.3.2 Αναλυτική παρουσίαση ελέγχου

Όπως αναφέρθηκε η Pythia περιλαμβάνει ελέγχους λειτουργίας. Για την δημιουργία αυτών γίνεται χρήση του Junit framework που διασφαλίζει ότι οι επιμέρους μέθοδοι ή κλάσεις λειτουργούν σωστά. Ο κώδικας χρησιμοποιεί το JUnit για να εκτελέσει μια σουίτα τεστ, αξιολογώντας τη συνολική λειτουργικότητα του συστήματος. Με το @RunWith(Suite.class), η κλάση AllTests εκτελεί μια συλλογή από τεστ, που οργανώνονται σε διαφορετικές κλάσεις (@SuiteClasses({...})), καθεμία από τις οποίες ελέγχει συγκεκριμένες λειτουργίες, όπως στατιστικά, clustering, regression και συσχετίσεις όπως φαίνεται και παρακάτω:

```java
package gr.uoi.cs.pythia;

import gr.uoi.cs.pythia.cardinalities.AllCardinalitiesTests;
import gr.uoi.cs.pythia.clustering.AllClusteringTests;
import gr.uoi.cs.pythia.correlations.AllCorrelationsTests;
import gr.uoi.cs.pythia.decisiontree.AllDecisionTreeTests;
import gr.uoi.cs.pythia.generalinfo.AllGenInfoTests;
import gr.uoi.cs.pythia.highlights.AllHighlightsTests;
import gr.uoi.cs.pythia.histogram.AllHistogramTests;
import gr.uoi.cs.pythia.labeling.LabelingSystemTests;
import gr.uoi.cs.pythia.outliers.AllOutlierTests;
import gr.uoi.cs.pythia.patterns.AllPatternTests;
import gr.uoi.cs.pythia.regression.AllRegressionTests;
import gr.uoi.cs.pythia.report.AllReportTests;
import gr.uoi.cs.pythia.writer.AllWriterTests;
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)  AlexandrosAlexiou
@SuiteClasses({
    AllCardinalitiesTests.class,
    AllClusteringTests.class,
    AllCorrelationsTests.class,
    AllDecisionTreeTests.class,
    AllGenInfoTests.class,
    AllHighlightsTests.class,
    AllHistogramTests.class,
    LabelingSystemTests.class,
    AllOutlierTests.class,
    AllPatternTests.class,
    AllRegressionTests.class,
    AllReportTests.class,
    AllWriterTests.class
})
public class AllTests {}
```

Εικόνα 59 test/java/pythia/AllTests.java

Για το πακέτο **generalinfo** δημιουργήθηκε αντίστοιχο για τους ελέγχους και ενσωματώθηκε στην εκτέλεση της Suite. Το διάγραμμα του πακέτου test φαίνεται παρακάτω:



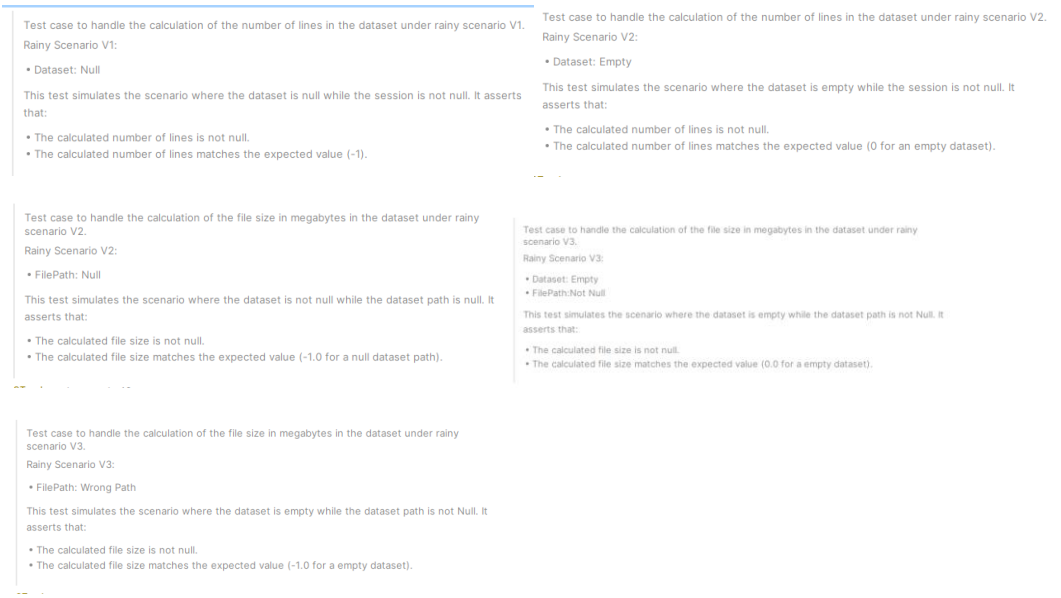
Εικόνα 60 test/java/pythia/generalinfo uml diagram tests

Δημιουργήθηκαν όπως φαίνεται Happy, Rainy class σενάρια που εξετάζουν αντίστοιχα τα ενδεχόμενα μεθόδων που υπολογίζουν τις συνολικές γραμμές του αρχείου καθώς και συνολικό του μέγεθος. Αναλυτικότερα έχουμε τα testCases:

Για HappyScenarios:

Εικόνα 61 testcase HappyScenarios

Για RainnyScenarios



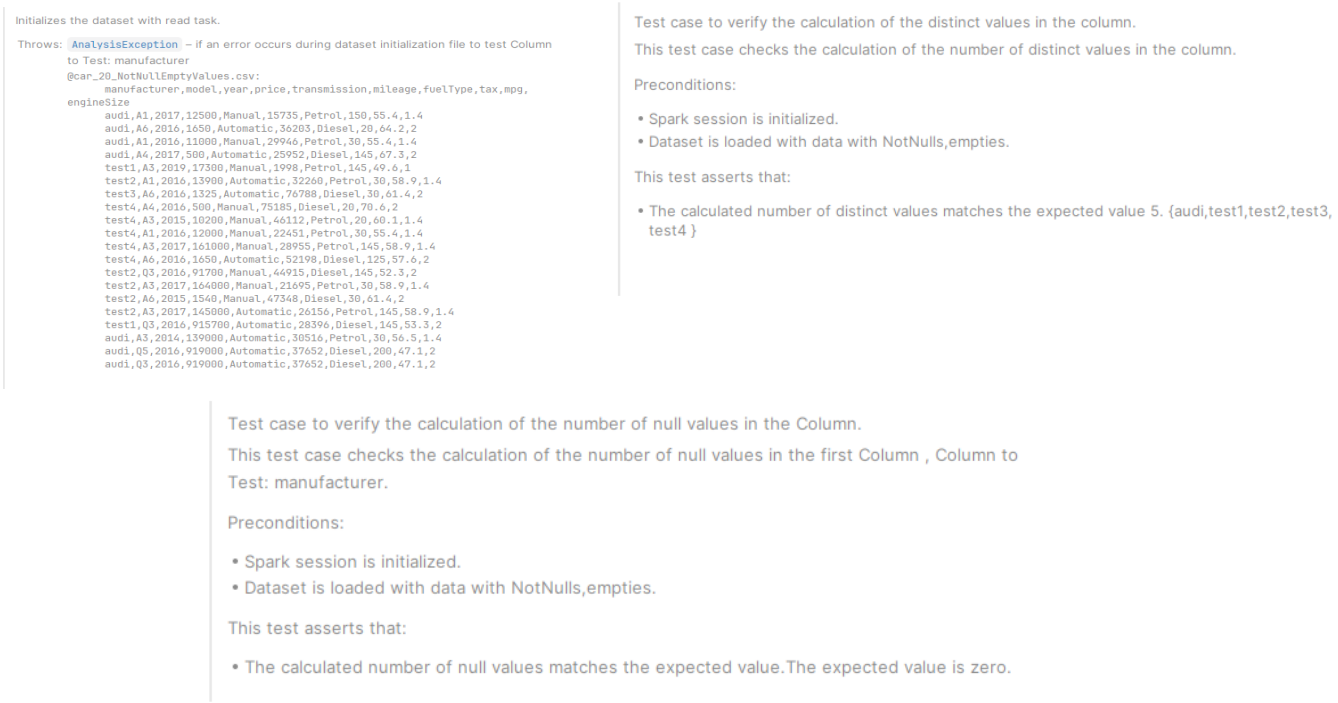
Εικόνα 62 testcase RainyScenarios

Για το πακέτο **cardinalities** που υπολογίζει τις τιμές null, distinct προστέθηκε αντίστοιχο πακέτο που περιλαμβάνει:

Εικόνα 63 test/java/pythia/cardinalities uml diagram tests

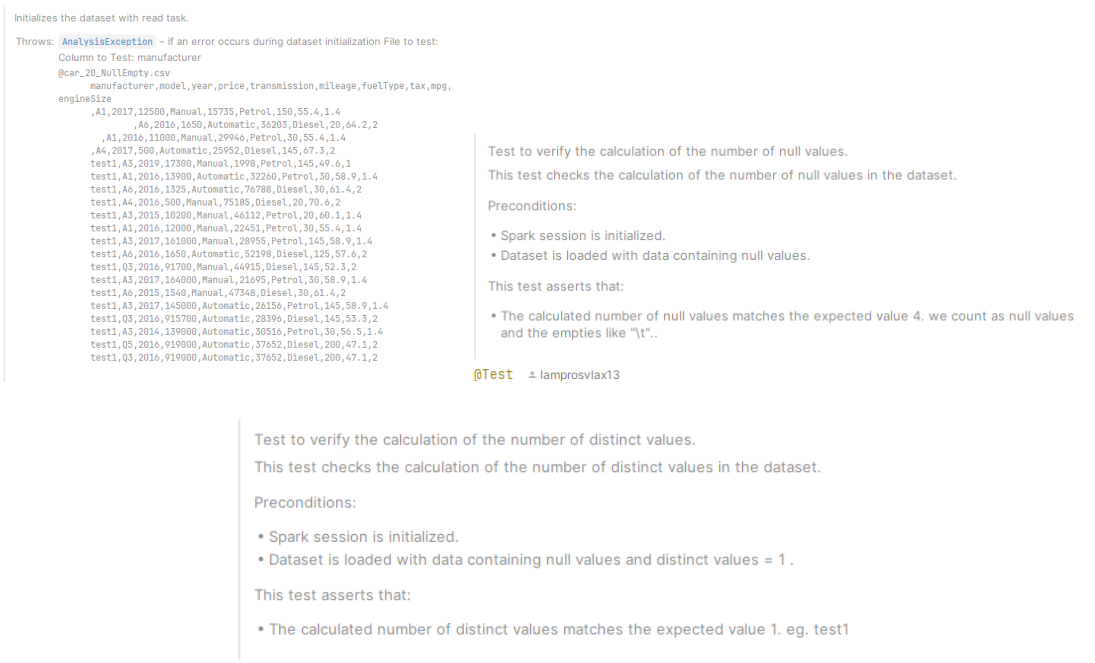
Το πακέτο περιλαμβάνει Happy, Rainy Scenarios που καλύπτουν περιπτώσεις υπολογισμού των τιμών όπως περιγράφετε παρακάτω από τα testCase:

HappyScenariosV1:



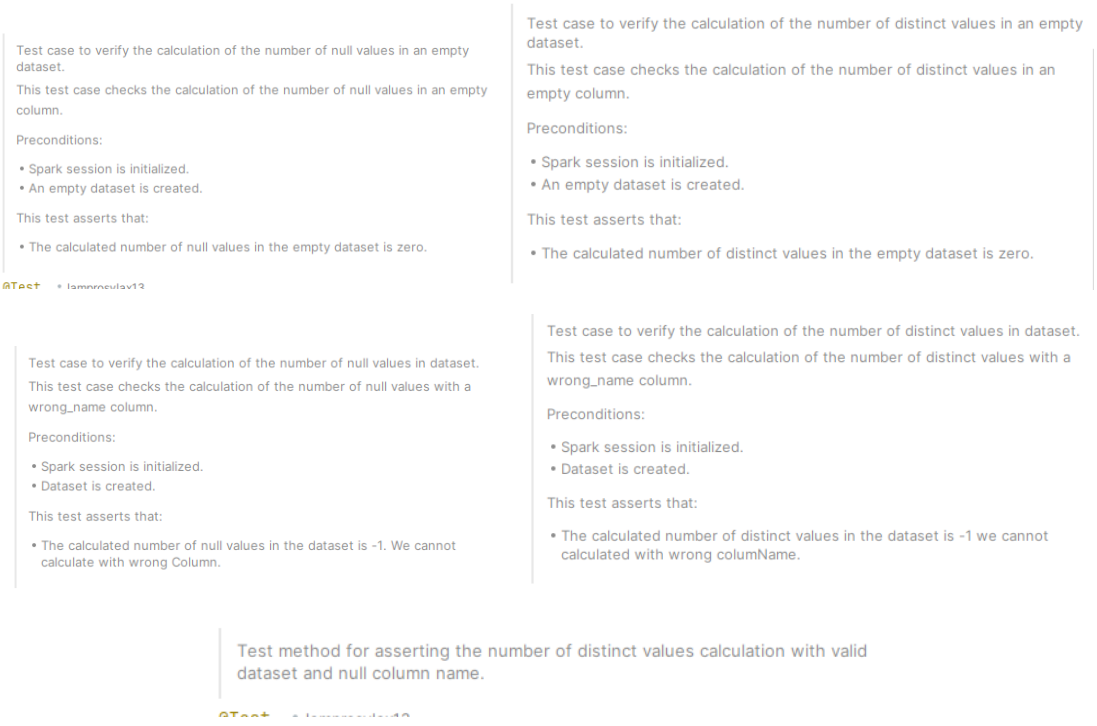
Εικόνα 64 testcase Happy ScenariosV1

HappyScenariosV2:



Εικόνα 65 testcase Happy ScenariosV2

RainyScenarios:

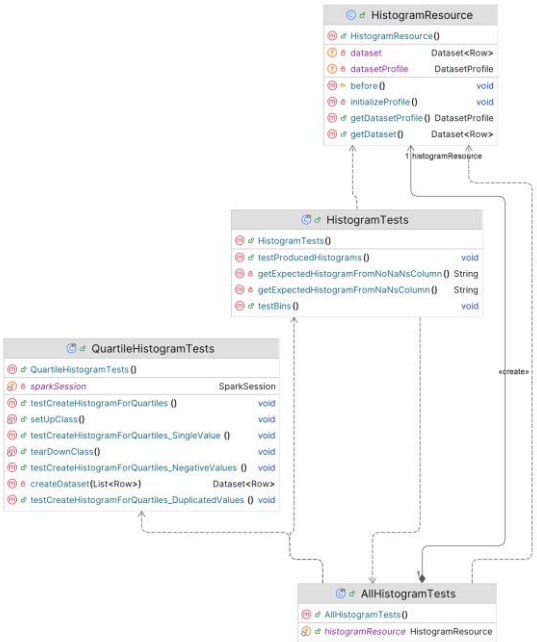


Εικόνα 66 testcase Rainy Scenarios

Για το πακετο **DescriptiveStatistics** δημιουργήθηκαν έλεγχοι που αφορούν μόνο τον υπολογισμό των mode τιμών. Το πακέτο περιλαμβάνει την κλάση DescriptiveStatisticsTest που ελέγχει τον υπολογισμό του mode (της πιο συχνής τιμής) σε δύο σενάρια: Στο πρώτο, δημιουργείται ένα σύνολο δεδομένων με διάφορες τιμές και null τιμές. Επαληθεύει ότι η πιο συχνή τιμή αναγνωρίζεται σωστά και ότι ο αριθμός των

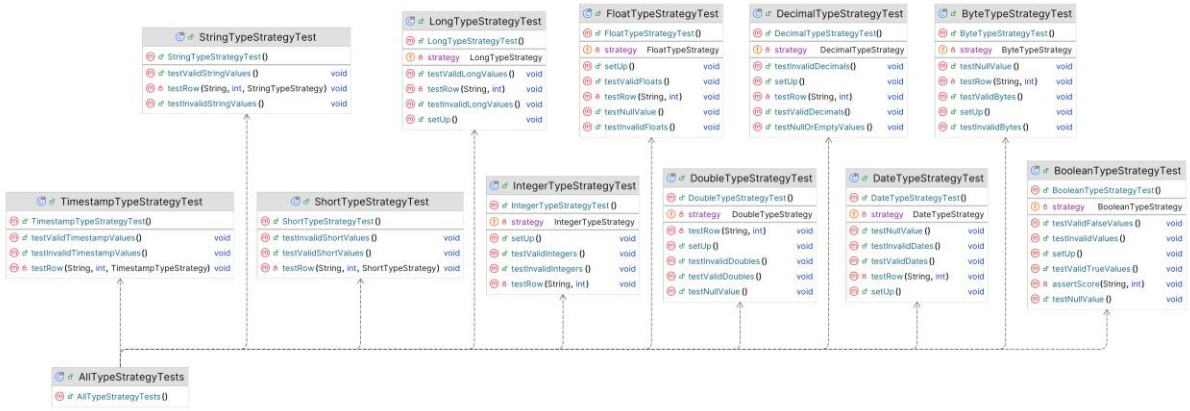
εμφανίσεων είναι σωστός. Ενώ στο δεύτερο, το σύνολο δεδομένων περιλαμβάνει τιμές με την ίδια συχνότητα , καθώς και άλλες μη χρήσιμες τιμές όπως κενά ή άκυρες. Ελέγχεται ότι μόνο οι τιμές με την υψηλότερη συχνότητα αναγνωρίζονται ως modes και ότι οι ακατάλληλες τιμές αγνοούνται.

Για το πακέτο **histograms** τροποποιήθηκε το ήδη υπάρχων πακέτο ελέγχων με την προσθήκη ελέγχων που αφορούν την δημιουργία του ισοϋψούς ιστογράμματος. Προστέθηκε η κλάση QuartileHistogramTests, η οποία ελέγχει τη σωστή δημιουργία ιστογραμμάτων για τα τεταρτημόρια με διάφορα δεδομένα. Η μέθοδος testCreateHistogramForQuartiles εξετάζει το ιστόγραμμα όταν τα δεδομένα περιλαμβάνουν πολλές διαφορετικές τιμές, ελέγχοντας αν τα όρια των bins υπολογίζονται σωστά και αν οι τιμές κατανέμονται σωστά στα bins. Η μέθοδος testCreateHistogramForQuartiles_SingleValue ελέγχει την περίπτωση όπου υπάρχει μόνο μία τιμή, διασφαλίζοντας ότι τα bins δημιουργούνται σωστά και η τιμή τοποθετείται στο σωστό bin. Η μέθοδος testCreateHistogramForQuartiles_DuplicatedValues εξετάζει πώς το ιστόγραμμα διαχειρίζεται τις διπλότυπες τιμές και αν καταμετρά σωστά την εμφάνιση αυτών. Τέλος, η μέθοδος testCreateHistogramForQuartiles_NegativeValues ελέγχει αν το ιστόγραμμα λειτουργεί σωστά με αρνητικές τιμές, εξασφαλίζοντας ότι τα bins και οι καταμετρήσεις υπολογίζονται σωστά. Το πακέτο παρουσιάζεται παρακάτω:



Εικόνα 67 histogram package test uml

Για το πακέτο **datatTypeIdentifier** που υπολογίζει τα score δημιουργήθηκε το κατάλληλο πακέτο ελέγχων όπως φαίνεται παρακάτω:



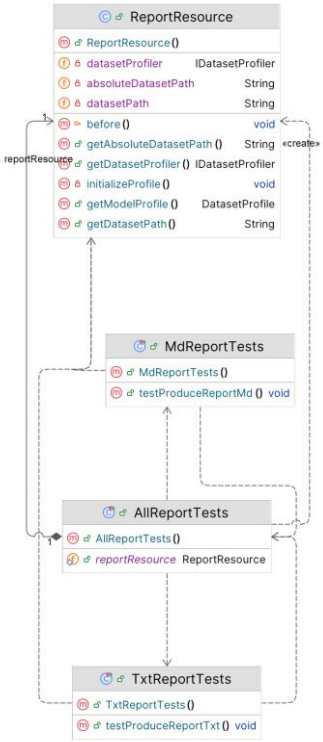
Εικόνα 68 dataTypeIdentifier package test uml diagram

Το πακέτο περιλαμβάνει tests που εξασφαλίζουν στην σωστή λειτουργία υπολογισμού του score. Πιο αναλυτικά για:

- **BooleanTypeStrategyTest:** Ελέγχει αν αναγνωρίζει σωστά τις λογικές τιμές (true/false) σε διάφορες μορφές (π.χ. "true", "1", "yes").
- **ByteTypeStrategyTest:** Ελέγχει αν αναγνωρίζει σωστά τις τιμές byte (ακέραιοι αριθμοί από -128 έως 127).
- **DateTypeStrategyTest:** Ελέγχει αν αναγνωρίζει σωστά τις ημερομηνίες σε διάφορες μορφές (π.χ. YYYY-MM-DD, DD/MM/YYYY).
- **DecimalTypeStrategyTest:** Ελέγχει αν αναγνωρίζει σωστά τους δεκαδικούς αριθμούς.
- **DoubleTypeStrategyTest:** Ελέγχει αν αναγνωρίζει σωστά τους αριθμούς διπλής ακρίβειας (double), συμπεριλαμβανομένων των αριθμών σε επιστημονική σημείωση.
- **FloatTypeStrategyTest:** Ελέγχει αν αναγνωρίζει σωστά τους αριθμούς κινητής υποδιαστολής (float).
- **IntegerTypeStrategyTest:** Ελέγχει αν αναγνωρίζει σωστά τους ακέραιους αριθμούς (integer).
- **LongTypeStrategyTest:** Ελέγχει αν αναγνωρίζει σωστά τους μεγάλους ακέραιους αριθμούς (long).
- **ShortTypeStrategyTest:** Ελέγχει αν αναγνωρίζει σωστά τους μικρούς ακέραιους αριθμούς (short).
- **StringTypeStrategyTest:** Ελέγχει αν αναγνωρίζει σωστά τις συμβολοσειρές (string).

- **TimestampTypeStrategyTest:** Ελέγχει αν αναγνωρίζει σωστά τις χρονοσημάνσεις (timestamp) σε διάφορες μορφές.

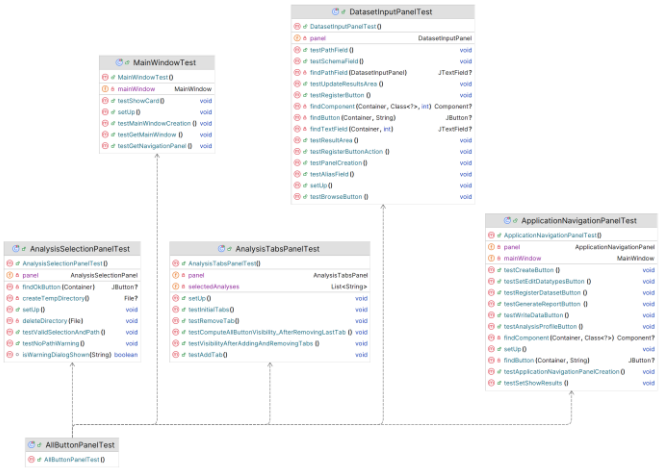
Για την προσθήκη των υπολογισμένων πληροφοριών στα reports τροποποιήθηκαν ελάχιστα οι υπάρχουσες δομές στα test στο πακέτο report. Η σχεδίαση του πακέτου παραμένει η ίδια:



Εικόνα 69 test/java/pythia/report uml diagram test

Ωστόσο έγιναν κάποιες τροποποιήσεις στις κλάσεις TxtReportTests, MdReportTests ώστε να ελέγχουν αν εγγράφονται οι πληροφορίες των πακέτων generalinfo, cardinalities, descriptiveStatistics, Histogram που αναλύθηκαν στις παραπάνω ενότητες. Επίσης ενημερώθηκαν τα αρχεία εισόδου για τα tests, τα αναμενόμενα αρχεία δηλαδή που χρειάζεται να δώσει το πακέτο reports ώστε να διαπιστωθεί η ορθή λειτουργία του.

Για το πακέτο **gui** και το πακέτο του **guibuttonPanels** έχουμε πακέτο με tests όπως φαίνεται παρακάτω:



Εικόνα 70 guiButtonPanels package test uml diagram

Το πακέτο αυτό είναι υπεύθυνο για την διασφάλιση της ορθής λειτουργίας του κεντρικού παραθύρου της εφαρμογής. Περιλαμβάνει ελέγχους με σενάρια χρήσης όπως παρακάτω:

- **AnalysisSelectionPanelTest:** Ελέγχει αν ο χρήστης μπορεί να επιλέξει σωστά την ανάλυση που θέλει να κάνει και τη φόρμα για την αποθήκευση του αρχείου αποτελεσμάτων. Επίσης, ελέγχει αν εμφανίζεται η προειδοποίηση "Give path" όταν ο χρήστης προσπαθεί να κάνει ανάλυση χωρίς να έχει δώσει διαδρομή αρχείου.
- **AnalysisTabsPanelTest:** Ελέγχει αν οι καρτέλες ανάλυσης δημιουργούνται, προστίθενται, αφαιρούνται και εμφανίζονται σωστά. Επίσης, ελέγχει αν το κουμπί "Compute All" εμφανίζεται μόνο όταν υπάρχουν καρτέλες ανάλυσης.
- **ApplicationNavigationPanelTest:** Ελέγχει αν τα κουμπιά πλοήγησης εμφανίζονται και λειτουργούν σωστά, και αν εμφανίζεται το σωστό περιεχόμενο όταν ο χρήστης τα πατάει. Επίσης, ελέγχει τη δυναμική δημιουργία του κουμπιού "EditDataType" και την αρχικοποίηση του κουμπιού "Show Results".
- **DatasetInputPanelTest:** Ελέγχει αν η οθόνη εισαγωγής dataset εμφανίζει σωστά τα πεδία για το alias, το path του αρχείου προς φόρτωση και το schema του dataset, και αν λειτουργούν σωστά τα κουμπιά "Register Dataset", και "Browse". Επίσης, ελέγχει αν εμφανίζονται σωστά οι πληροφορίες του dataset στην οθόνη αποτελεσμάτων.
- **MainWindowTest:** Ελέγχει αν το κυρίως παράθυρο της εφαρμογής δημιουργείται σωστά (τίτλος, διαστάσεις, κλπ.) και αν εμφανίζονται σωστά οι διάφορες οθόνες. Επίσης, ελέγχει αν οι μέθοδοι getMainWindow και getNavigationPanel λειτουργούν σωστά.

Για το πακέτο **analysisTasksGuiPanels** δημιουργήθηκε πακέτο ελέγχων όπως φαίνεται παρακάτω:

Εικόνα 71 analysisTasksGuiPanels package test uml diagram

Οι έλεγχοι εδώ στοχεύουν την ορθή λειτουργία των καρτελών μέσω των οποίων ο χρήστης δίνει τις παραμέτρους για κάθε ανάλυση. Πιο συγκεκριμένα:

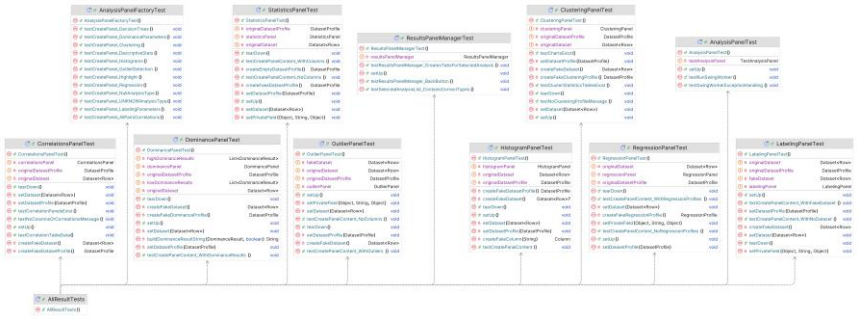
- **ClusteringGUITest**: Ελέγχει αν η οθόνη για την ανάλυση "Clustering" εμφανίζει σωστά τις επιλογές (π.χ. είδος "clustering", αριθμός ομάδων, χαρακτηριστικά) και αν οι πληροφορίες που δίνει ο χρήστης εμφανίζονται σωστά στην οθόνη αποτελεσμάτων.
- **ClusteringParameterValidatorTest**: Ελέγχει αν το πρόγραμμα που ελέγχει τις παραμέτρους για την ανάλυση "Clustering" λειτουργεί σωστά, δηλαδή αν δέχεται μόνο έγκυρες τιμές για τις παραμέτρους και απορρίπτει τις μη έγκυρες.
- **CorrelationsGUITest**: Ελέγχει αν η οθόνη για τις συσχετίσεις εμφανίζει σωστά την επιλογή μεθόδου συσχέτισης και αν ενημερώνεται σωστά η οθόνη αποτελεσμάτων.
- **CorrelationsParameterValidatorTest**: Ελέγχει αν το πρόγραμμα που ελέγχει τις παραμέτρους για τις συσχετίσεις λειτουργεί σωστά, δηλαδή αν δέχεται μόνο έγκυρες τιμές για τη μέθοδο συσχέτισης και απορρίπτει τις μη έγκυρες.
- **DatasetWriterGUITest**: Ελέγχει αν η οθόνη για την εγγραφή dataset εμφανίζει σωστά τις επιλογές (π.χ. alias dataset, τύπος εγγραφής, διαδρομή εξόδου) και αν ενημερώνεται σωστά η οθόνη αποτελεσμάτων.
- **DatasetWriterParameterValidatorTest**: Ελέγχει αν το πρόγραμμα που ελέγχει τις παραμέτρους για την εγγραφή dataset λειτουργεί σωστά, δηλαδή αν δέχεται μόνο έγκυρες τιμές για το alias του dataset και τη διαδρομή εξόδου και απορρίπτει τις μη έγκυρες.
- **DominanceParametersGUITest**: Ελέγχει αν η οθόνη για τις παραμέτρους κυριαρχίας εμφανίζει σωστά τις επιλογές (π.χ. τρόπος επιλογής, στήλες

μετρήσεων, στήλες συντεταγμένων) και αν ενημερώνεται σωστά η οθόνη αποτελεσμάτων.

- **DominanceParameterValidatorTest**: Ελέγχει αν το πρόγραμμα που ελέγχει τις παραμέτρους για την ανάλυση "Κυριαρχία" λειτουργεί σωστά, δηλαδή αν δέχεται μόνο έγκυρες τιμές για τις παραμέτρους και απορρίπτει τις μη έγκυρες.
- **HighlightParametersGUITest**: Ελέγχει αν η οθόνη για τις παραμέτρους επισήμανσης εμφανίζει σωστά τις επιλογές (π.χ. τρόπος εξαγωγής, αριθμητικό όριο) και αν ενημερώνεται σωστά η οθόνη αποτελεσμάτων.
- **HighlightParameterValidatorTest**: Ελέγχει αν το πρόγραμμα που ελέγχει τις παραμέτρους για την επισήμανση δεδομένων λειτουργεί σωστά, δηλαδή αν δέχεται μόνο έγκυρες τιμές για τις παραμέτρους και απορρίπτει τις μη έγκυρες.
- **HistogramGUITest**: Ελέγχει αν η οθόνη για τα ιστογράμματα εμφανίζει σωστά τις επιλογές (π.χ. τύπος ιστογράμματος, αριθμός bins) και αν ενημερώνεται σωστά η οθόνη αποτελεσμάτων.
- **HistogramParameterValidatorTest**: Ελέγχει αν το πρόγραμμα που ελέγχει τις παραμέτρους για τη δημιουργία ιστογράμματος λειτουργεί σωστά, δηλαδή αν δέχεται μόνο έγκυρες τιμές για τις παραμέτρους και απορρίπτει τις μη έγκυρες.
- **LabelingSystemGUITest**: Ελέγχει αν η οθόνη για το σύστημα ετικετών επιτρέπει την προσθήκη κανόνων με έγκυρες τιμές και αν εμφανίζει σωστά τους κανόνες που προστέθηκαν. Επίσης, ελέγχει την συμπεριφορά του συστήματος όταν ο χρήστης προσπαθήσει να προσθέσει κανόνες με μη έγκυρες τιμές.
- **OutlierAnalysisGUITest**: Ελέγχει αν η οθόνη για την ανάλυση outliers εμφανίζει σωστά τις επιλογές (π.χ. τύπος outlier, όριο) και αν ενημερώνεται σωστά η οθόνη αποτελεσμάτων.
- **OutlierParameterValidatorTest**: Ελέγχει αν το πρόγραμμα που ελέγχει τις παραμέτρους για την ανάλυση outliers λειτουργεί σωστά, δηλαδή αν δέχεται μόνο έγκυρες τιμές για τις παραμέτρους και απορρίπτει τις μη έγκυρες.
- **RegressionGUITest**: Ελέγχει αν η οθόνη για την ανάλυση παλινδρόμησης επιτρέπει την προσθήκη παλινδρομήσεων με έγκυρες τιμές και αν εμφανίζει σωστά τις παραμέτρους των παλινδρομήσεων που προστέθηκαν. Επίσης, ελέγχει την συμπεριφορά του συστήματος όταν ο χρήστης προσπαθήσει να προσθέσει παλινδρομήσεις με μη έγκυρες τιμές.
- **ReportGeneratorGUITest**: Ελέγχει αν η οθόνη για τη δημιουργία αναφορών εμφανίζει σωστά τις επιλογές (π.χ. τύπος αναφοράς, διαδρομή αποθήκευσης) και αν ενημερώνεται σωστά η οθόνη αποτελεσμάτων. Επίσης, ελέγχει αν λειτουργεί σωστά η επιλογή διαδρομής μέσω του κουμπιού "Browse".

- **ReportParameterValidatorTest**: Ελέγχει αν το πρόγραμμα που ελέγχει τις παραμέτρους για τη δημιουργία αναφορών λειτουργεί σωστά, δηλαδή αν δέχεται μόνο έγκυρες τιμές για τις παραμέτρους και απορρίπτει τις μη έγκυρες.

Για το πακέτο **resultsGui** που δημιουργεί καρτέλες με τα αποτελέσματα των αναλύσεων δημιουργήθηκε πακέτο ελέγχων όπως παρακάτω:



Εικόνα 72 resultGui package test uml diagram

Οι έλεγχοι εδώ στοχεύουν την ορθή λειτουργία των καρτελών μέσω των οποίων ο χρήστης βλέπει τα αποτελέσματα των αναλύσεων. Πιο συγκεκριμένα:

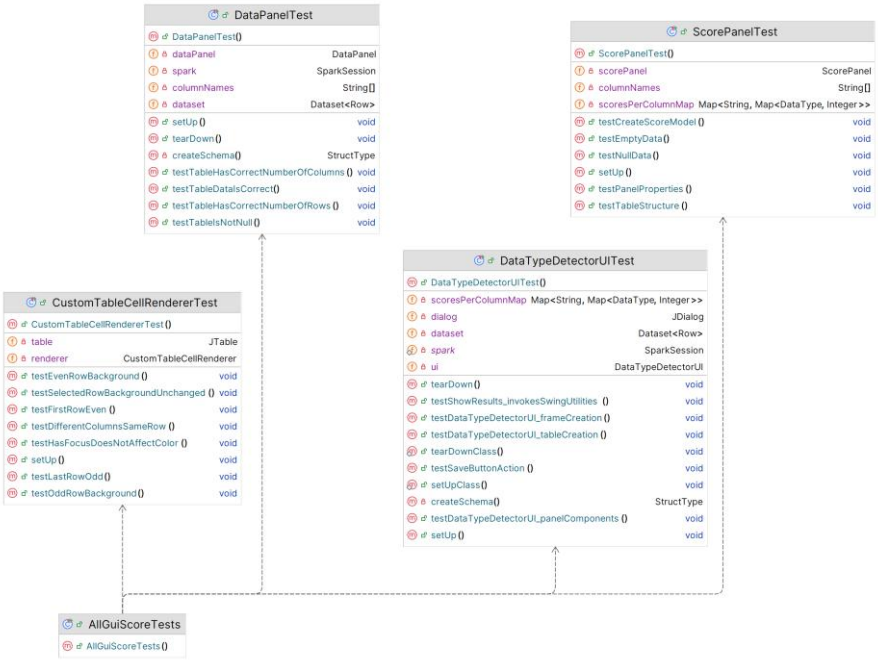
- Η AnalysisPanelFactoryTest ελέγχει αν η AnalysisPanelFactory.createPanel() επιστρέφει σωστά τις αντίστοιχες υλοποιήσεις των AnalysisPanel για κάθε AnalysisType.
- Η AnalysisPanelTest επαληθεύει τη συμπεριφορά της runSwingWorker() στην AnalysisPanel. Ελέγχει αν η createPanelContent() καλείται σωστά και αν οι εξαιρέσεις που προκύπτουν μέσα της διαχειρίζονται σωστά.
- Η ClusteringPanelTest ελέγχει αν η ClusteringPanel δημιουργεί σωστά τους πίνακες στατιστικών για τα clusters. Περιλαμβάνει δοκιμές για την ύπαρξη των πινάκων στατιστικών των clusters και την εμφάνιση κατάλληλου μηνύματος όταν δεν υπάρχει διαθέσιμο ClusteringProfile.
- Η CorrelationsPanelTest ελέγχει αν η CorrelationsPanel εμφανίζει σωστά τους πίνακες συσχετίσεων. Περιλαμβάνει δοκιμές για την ύπαρξη των πινάκων, την εμφάνιση μηνύματος όταν δεν υπάρχουν δεδομένα, και την ορθότητα των δεδομένων στον πίνακα συσχετίσεων.
- Η DominancePanelTest ελέγχει αν η DominancePanel εμφανίζει σωστά τα αποτελέσματα κυριαρχίας (dominance results). Περιλαμβάνει δοκιμές για την ύπαρξη των πάνελ υψηλής και χαμηλής κυριαρχίας, την ορθότητα των δεδομένων στις αντίστοιχες περιοχές κειμένου (JTextArea), και τη διαχείριση ψεύτικου προφίλ δεδομένων με reflection.

- Το τεστ HistogramPanelTest ελέγχει τη σωστή λειτουργία του HistogramPanel. Χρησιμοποιώντας reflection, εισάγει ψεύτικα δεδομένα (dataset και dataset profile με ιστογράμματα) στην κλάση AppController. Στη συνέχεια, καλεί τη μέθοδο createPanelContent() του HistogramPanel και ελέγχει αν δημιουργούνται σωστά τα γραφήματα ιστογραμμάτων, δηλαδή αν εμφανίζονται τα JScrollPane, JPanel και ChartPanel με τα δεδομένα. Τέλος, επαναφέρει την αρχική κατάσταση του AppController, πάλι με χρήση reflection, για να μην επηρεαστούν άλλα τεστ.
- Το τεστ για το LabelingPanel ελέγχει τη σωστή εμφάνιση του περιεχομένου του panel σε δύο περιπτώσεις. Στην πρώτη περίπτωση, όταν υπάρχει διαθέσιμο dataset, επαληθεύεται ότι το panel εμφανίζει έναν πίνακα με τις σωστές στήλες και δεδομένα. Στη δεύτερη περίπτωση, όταν το dataset είναι null, το panel πρέπει να εμφανίζει το μήνυμα "No data available." για να δείξει ότι δεν υπάρχουν διαθέσιμα δεδομένα.
- Η κλάση OutlierPanelTest ελέγχει τη λειτουργία της OutlierPanel σε δύο περιπτώσεις: πρώτα, όταν δεν υπάρχουν στήλες, επιβεβαιώνει ότι εμφανίζεται το μήνυμα "No columns found.", και δεύτερον, όταν υπάρχουν outliers, ελέγχει ότι δημιουργείται σωστά ένα JScrollPane για την εμφάνιση των δεδομένων. Χρησιμοποιεί reflection για να ορίσει ψεύτικα δεδομένα και επαναφέρει την αρχική κατάσταση μετά την ολοκλήρωση των ελέγχων.
- Η κλάση RegressionPanelTest ελέγχει τη λειτουργία της RegressionPanel σε δύο σενάρια: πρώτα, όταν υπάρχουν regression profiles, επιβεβαιώνει ότι δημιουργείται σωστά ένα JScrollPane για την εμφάνιση των δεδομένων, και δεύτερον, όταν δεν υπάρχουν regression profiles, ελέγχει ότι εμφανίζεται το μήνυμα "No regression profiles found." Χρησιμοποιεί reflection για να ορίσει ψεύτικα δεδομένα και επαναφέρει την αρχική κατάσταση μετά την ολοκλήρωση των ελέγχων.
- Ηκλάση ResultsPanelManagerTest ελέγχει τη συμπεριφορά της ResultsPanelManager. Πρώτα, δημιουργεί JCheckBoxes για να προσομοιώσει την επιλογή του χρήστη και αρχικοποιεί την ResultsPanelManager με αυτά. Στη συνέχεια, ελέγχει ότι δημιουργείται σωστά ένα tab μόνο για την επιλεγμένη ανάλυση ("Descriptive Stats") και ότι το κουμπί "Επιστροφή" υπάρχει και έχει το σωστό κείμενο. Τέλος, χρησιμοποιεί reflection για να εξαγάγει και να επιβεβαιώσει ότι η λίστα selectedAnalysis περιέχει μόνο τον σωστό τύπο ανάλυσης.
- Η κλάση StatisticsPanelTest ελέγχει τη συμπεριφορά της StatisticsPanel σε δύο σενάρια: πρώτα, όταν υπάρχουν στήλες στο DatasetProfile, επιβεβαιώνει ότι

δημιουργείται σωστά ένας πίνακας (JTable) με τις σωστές πληροφορίες για κάθε στήλη, και δεύτερον, όταν δεν υπάρχουν στήλες, ελέγχει ότι εμφανίζεται το μήνυμα "No columns found." Χρησιμοποιεί reflection για να ορίσει ψεύτικα δεδομένα και επαναφέρει την αρχική κατάσταση μετά την ολοκλήρωση των ελέγχων.

Είναι σημαντικό εδώ να τονιστεί ότι για την δημιουργία ελέγχων που εξετάζουν τα υποσυστήματα δημιουργίας γραφικών παραθύρων, κουμπιών κτλ., κυρίως στο πακέτο resultsGui χρησιμοποιήθηκε η μέθοδος reflection. Ο AppController είναι μια singleton instance κλάση που διαχειρίζεται τα δεδομένα της εφαρμογής. Στα τεστς, χρησιμοποιείται reflection για να τροποποιηθούν τα private πεδία του, όπως το dataset και το datasetProfile, ώστε να χρησιμοποιηθούν ψεύτικα δεδομένα. Μετά από κάθε δοκιμή, η αρχική κατάσταση του AppController επαναφέρεται, εξασφαλίζοντας ότι οι δοκιμές δεν επηρεάζουν η μία την άλλη ούτε την εφαρμογή. Επίσης κάποια test απαιτούν από το χρήστη μια αλληλεπίδραση με το παράθυρο σε κουμπιά okButtons, καθώς δε χρησιμοποιείτε κάποιο Framework που μπορεί να δημιουργήσει mock αντικείμενα. Τα test αυτά είναι μια προσωρινή λύση στο μέλλον μπορούν να αντικατασταθούν.

Για το πακέτο **guiScores** που υλοποιεί την γραφική διεπαφή με το χρήστη προβάλλοντας του τα υπολογισμένα score και δίνοντας του τη δυνατότητα να τροποποιήσει τους τύπους των στηλών, δημιουργήθηκε πακέτο ελέγχων όπως φαίνεται παρακάτω:



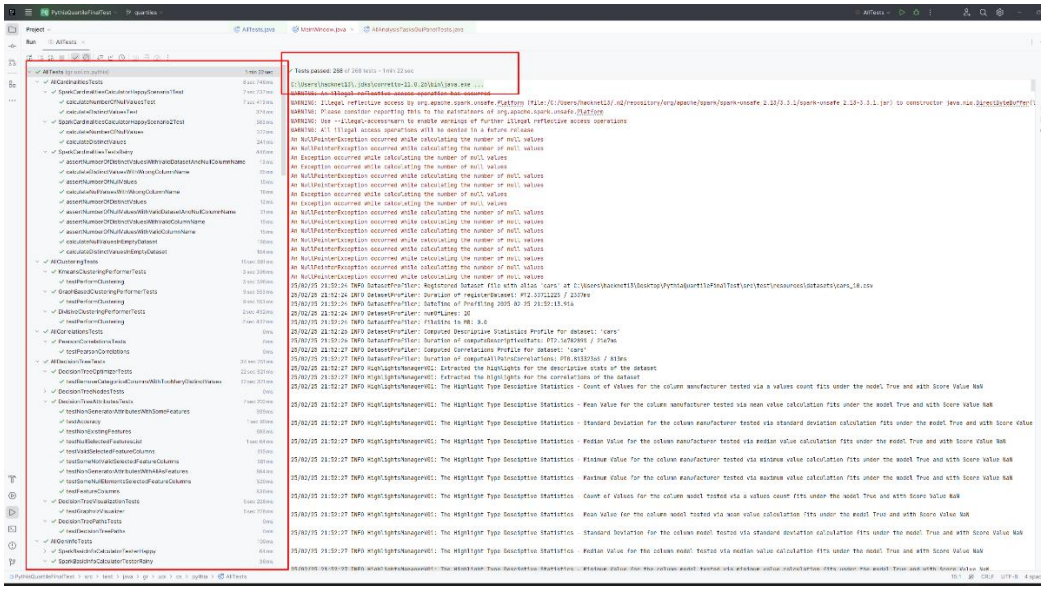
Εικόνα 73 guiScores package test uml diagram

Το πακέτο περιλαμβάνει ελέγχους εξασφαλίζουν την ορθή λειτουργία του υποσυστήματος και περιλαμβάνουν ελέγχους όπως παρακάτω:

- **CustomTableCellRendererTest:** Ελέγχει αν τα κελιά του πίνακα εμφανίζονται σωστά, με διαφορετικό χρώμα ανάλογα με το αν η σειρά είναι ζυγή ή μονή, αν το κελί είναι επιλεγμένο ή όχι, και αν έχει εστίαση δηλαδή δεν επηρεάζει το χρώμα του κελιού.
- **DataPanelTest:** Ελέγχει αν ο πίνακας που εμφανίζει τα δεδομένα από ένα Spark Dataset δημιουργείται και εμφανίζεται σωστά, με τον σωστό αριθμό γραμμών και στηλών, και με τα σωστά δεδομένα.
- **DataTypeDetectorUITest:** Το τεστ για το DataTypeDetectorUI επαληθεύει τη σωστή δημιουργία του γραφικού περιβάλλοντος και τη λειτουργικότητα των βασικών στοιχείων του. Ελέγχει ότι το JFrame και το JDialog δημιουργούνται σωστά, ότι τα δεδομένα εμφανίζονται στους αντίστοιχους πίνακες (JTable), και ότι η διάταξη των panels (DataPanel, ScorePanel, JPanel) είναι η αναμενόμενη.
- **HeaderClickListenerTest:** Ελέγχει τι συμβαίνει όταν ο χρήστης κάνει κλικ στην κεφαλίδα ενός πίνακα. Ελέγχει αν εμφανίζεται το σωστό παράθυρο διαλόγου με το σωστό όνομα στήλης.
- **ScorePanelTest:** Ελέγχει αν η οθόνη που εμφανίζει τις βαθμολογίες των διαφόρων τύπων δεδομένων για κάθε στήλη δημιουργείται σωστά, με το σωστό μοντέλο πίνακα και με τις σωστές ιδιότητες.

3.3.3 Αποτελέσματα Test

Μετά από όλες τις προσθήκες και τις υλοποιήσεις που τροποποιήθηκαν και προστέθηκαν χρειάζεται να ελεγχθεί η ορθότητα των όσων αναφέρθηκαν και κυρίως να διασφαλιστεί ότι δεν δημιουργούν προβλήματα στην ήδη υπάρχουσα δομή. Η εκτέλεση των συνολικών test του συστήματος δείχνει ότι όλα λειτουργούν σύμφωνα με τα αναμενόμενα αποτελέσματα.



Εικόνα 77 Αποτελέσματα εκτέλεσης όλων των ελέγχων της Pythias

Από την εικόνα φαίνεται ότι 268 tests εκτελέστηκαν όπως ακριβώς αναμενόταν.

3.4 Λεπτομέρειες εγκατάστασης και υλοποίησης

Όπως έχει ήδη αναφερθεί, το Pythia είναι ένα υπάρχον έργο διαθέσιμο στο GitHub. Οι προγραμματιστές(Contributors) που έχουν συνεισφέρει στην ανάπτυξή του επισημαίνουν ότι για την επιτυχή εγκατάσταση και εκτέλεση του λογισμικού, πρέπει να ακολουθηθούν συγκεκριμένα βήματα.

Για την εγκατάσταση και εκτέλεση του Pythia, απαιτούνται:

- **Java 8**: Απαραίτητη για την εκτέλεση του κώδικα. Μετά την εγκατάσταση, πρέπει να ρυθμιστεί η μεταβλητή περιβάλλοντος JAVA_HOME.
- **Ένα IDE για Java**: Μπορεί να χρησιμοποιηθεί το Eclipse, το IntelliJ IDEA ή οποιοδήποτε άλλο συμβατό περιβάλλον ανάπτυξης.
- **Git (προαιρετικά)**: Για την κλωνοποίηση του πηγαίου κώδικα από το GitHub απαιτείται η εγκατάσταση του Git και η ρύθμιση ενός προφίλ με όνομα χρήστη και email.
- **Apache Hadoop 3.2.2**: Χρησιμοποιείται για την επεξεργασία δεδομένων. Μετά την εγκατάσταση, πρέπει να ρυθμιστεί η μεταβλητή HADOOP_HOME.
- **WinUtils (μόνο για Windows)**: Απαιτείται για τη σωστή λειτουργία του Hadoop σε Windows.

1. Λήψη και Ρύθμιση του Pythia

Ο πηγαίος κώδικας του Pythia μπορεί να ληφθεί με δύο τρόπους:

1. **Με Git**: Εκτελώντας την εντολή git clone <repository-url>.
2. **Χωρίς Git**: Με απευθείας λήψη του .zip αρχείου από το GitHub και εξαγωγή του στον υπολογιστή.

Το Pythia χρησιμοποιεί Maven για τη διαχείριση των εξαρτήσεων και τη δημιουργία του έργου. Δεν απαιτείται ξεχωριστή εγκατάσταση του Maven, καθώς περιλαμβάνεται στον πηγαίο κώδικα μέσω του Maven Wrapper.

2. Κατασκευή και Εκτέλεση

Ανοίγοντας το έργο στο IDE της επιλογής, το Maven ανακτά αυτόματα τις απαραίτητες βιβλιοθήκες και προετοιμάζει το σύστημα για εκτέλεση. Για την κατασκευή του έργου, ανοίγει ένα τερματικό, γίνεται μετάβαση στον φάκελο του έργου και εκτελείται η αντίστοιχη εντολή:

- **Windows**: ./mvnw.cmd clean install
- **Linux/Mac**: ./mvnw clean install

Για την εκτέλεση των δοκιμών:

- **Windows**: ./mvnw.cmd test
- **Linux/Mac**: ./mvnw test
- Ορισμός παραμέτρου στον IDE στο πεδίο VM options: -Djava.awt.headless=false. Αυτή η προσθήκη οφείλεται ότι περιλαμβάνονται Test που χρειάζονται αλληλεπίδραση με το χρήστη.

3. Τελική Έξοδος – Αρχεία JAR

Η διαδικασία κατασκευής παράγει δύο εκδόσεις του Pythia σε μορφή JAR:

1. **Pythia-x.y.z-alldeps.jar**: Περιλαμβάνει όλες τις εξαρτήσεις και μπορεί να εκτελεστεί ανεξάρτητα.
2. **Pythia-x.y.z.jar**: Περιέχει μόνο τον βασικό κώδικα και απαιτεί την ξεχωριστή εγκατάσταση των εξαρτήσεων.

Ανάλογα με τις απαιτήσεις κάθε περίπτωσης, μπορεί να χρησιμοποιηθεί η κατάλληλη έκδοση για την ενσωμάτωση του Pythia σε άλλα έργα. Ακολουθώντας αυτά τα βήματα, διασφαλίζεται ότι το Pythia είναι σωστά εγκατεστημένο και έτοιμο προς χρήση.

3.5 Επεκτασιμότητα του λογισμικού

Στην ανάπτυξη λογισμικού, η προσοχή στη συγγραφή κώδικα που είναι επεκτάσιμος και συντηρήσιμος είναι υψίστης σημασίας. Γι' αυτό, η Pythia είναι υλοποιημένη με τεχνικές που επιτρέπουν εύκολα την προσθήκη, αφαίρεση και τροποποίηση λειτουργιών. Κάθε

υποσύστημα σχεδιάστηκε με ένα κεντρικό Interface και ένα Factory Class, διασφαλίζοντας την εύκολη τροποποίηση και επέκταση, ενώ κάποιες άλλες κλάσεις ορίζονται ως Abstract για την αποφυγή ίδιου κώδικα. Όλα τα νέα πακέτα που ενσωματώθηκαν ακολουθούν συνεπώς το ίδιο μοτίβο, διατηρώντας την αρχιτεκτονική ευέλικτη και προσαρμόσιμη. Κάποια παραδείγματα των παραπάνω αποτελούν τα πακέτα cardinalities, generalinfo, datatypeIdentifier που όπως παρουσιάστηκαν και αναλύθηκαν παραπάνω.

Κεφάλαιο 4. Πειραματική Αξιολόγηση

Στο κεφάλαιο αυτό, εξετάζεται η απόδοση των νέων λειτουργιών που προστέθηκαν στο πρόγραμμα Pythias. Για να γίνει αυτό, ελέγχεται κάθε νέα λειτουργία ξεχωριστά, σαν να ήταν ένα αυτόνομο κομμάτι κώδικα. Καταμετρείτε ο χρόνος που χρειάζεται κάθε λειτουργία για να ολοκληρώσει τη δουλειά της, και παρατηρείται πώς αλλάζει ο χρόνος αυτός όταν της δίνονται περισσότερα δεδομένα. Έτσι, γίνεται αντιληπτό αν οι νέες λειτουργίες είναι γρήγορες και ανθεκτικές, χωρίς να επηρεάζεται η αξιολόγηση από το πώς χρησιμοποιούνται μέσα στο υπόλοιπο πρόγραμμα Pythias. Με αυτόν τον τρόπο, αποκτάται μια καθαρή εικόνα για το πόσο καλά λειτουργούν οι νέες προσθήκες και αν χρειάζονται βελτιώσεις.

4.1 Μεθοδολογία πειραματισμού

Στην παρούσα ενότητα, περιγράφεται η μεθοδολογία που θα ακολουθηθεί για την αξιολόγηση των υποσυστημάτων. Θα διεξαχθούν πειραματικές μετρήσεις για να διαπιστωθεί η απόδοση των υποσυστημάτων σε διάφορες λειτουργίες. Συγκεκριμένα, θα μετρηθεί ο χρόνος που απαιτείται για την εύρεση στατιστικών πληροφοριών όπως οι τιμές null, distinct και mode σε αρχεία διαφορετικού μεγέθους, για να διαπιστωθεί αν το μέγεθος των αρχείων επηρεάζει την ταχύτητα. Επίσης, θα μετρηθεί ο χρόνος που χρειάζεται το πρόγραμμα για να μετρήσει τις γραμμές ενός αρχείου και το μέγεθός του. Επιπλέον, θα γίνει μέτρηση του χρόνου που απαιτείται για τον υπολογισμό score για κάθε στήλη που αφορά την αναγνώριση του τύπου δεδομένων της , και τέλος θα μετρηθεί ο χρόνος για τη δημιουργία ενός συγκεκριμένου διαγράμματος που δείχνει την κατανομή των δεδομένων. Για τους υπολογισμούς που αφορούν τις τιμές mode, distinct, null, total lines και file Size χρησιμοποιείται το σύνολο δεδομένων από το αρχείο που βρίσκεται στον σύνδεσμο: https://www.kaggle.com/datasets/mkechinov/ecommerce-behavior-data-from-multi-category-store?select=2019-Oct.csv&fbclid=IwY2xjawIm6rpleHRuA2FlbQIxMAABHZKsBsdnIc9tY6Zvcuuky3ZgdkGgWk81By6VUvITJXcs23KNeJ1PKTZXog_aem_R-Lgtkb1Akgvkqi5upOsmg, και συγκεκριμένα για τους ελέγχους χρησιμοποιήθηκε το 2019-Nov.csv. Σχετικά με το αρχείο δημιουργήθηκαν και αρχεία με 20.000, 200.000, 2.000.000, 20.000.000 εγγραφές αυτού, για να προσδιοριστεί όπως αναφέρθηκε παραπάνω το πώς επηρεάζει το μέγεθος δεδομένων τους υπολογισμούς. Το αρχείο αυτό αποτελείται από 9 στήλες και οι

συνολικές εγγραφές είναι 67.501.979. Κατά τη φόρτωση του αρχείου, το Spark ρυθμίστηκε να προσδιορίζει αυτόματα το σχήμα των δεδομένων, δηλαδή τους τύπους των στηλών. Αυτό έγινε γιατί η ανάλυση δεν επικεντρώνεται στην ακρίβεια των τύπων δεδομένων, αλλά στην απόδοση των λειτουργιών του Spark. Ο υπολογισμός των mode, null και distinct τιμών γίνεται μέσω συναρτήσεων του Spark όπως αναφέρθηκε στο κεφάλαιο 3. Επομένως, ο χρόνος που απαιτείται εξαρτάται από το πώς το Spark χειρίζεται τους τύπους δεδομένων, και όχι από τον εξωτερικό ορισμό τους. Με αυτόν τον τρόπο, εστιάζουμε στην αξιολόγηση της ταχύτητας και της αποτελεσματικότητας του Spark κατά την εκτέλεση των συγκεκριμένων λειτουργιών. Για τους υπολογισμούς των score χρησιμοποιούνται τα αρχεία Adult100K.csv, tweets.csv που βρίσκονται στα resources/datasets της Pythias καθώς και ένα αρχείο από τον σύνδεσμο https://www.kaggle.com/datasets/hopesb/hr-analytics-dataset?select=Messy_HR_Dataset_Detailed.csv&fbclid=IwY2xjawIoWN1leHRuA2FlbQIxMAABHbHfbRaEAFpHRBYSBbOJXnLP9U98gfZQwYuYJHOnjnfcpSZU7Z2w7L2Rjw_aem_eYZzAo5FsET8a4uEefOpGQ με όνομα Messy_HR_Dataset_Detailed.csv. Επίσης για την καταμέτρηση του χρόνου δημιουργίας ισοϋψών διαγραμμάτων χρησιμοποιούνται τα cars_100k.csv, Adult100K.csv από τα resources/datasets της Pythias, αλλά και ένα αρχείο από τον σύνδεσμο https://www.kaggle.com/datasets/picekl/geoplant με όνομα PA-test-landsat_time_series-blue.csv.

Όλες οι μετρήσεις εκτελέστηκαν και υπολογίστηκαν σε ένα σύστημα με λειτουργικό σύστημα με τα παρακάτω χαρακτηριστικά:

Operating System	Windows 11
Cpu	Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz 3.70 GHz
Ram	12,0 GB 3200MHz
Disk	WDC WDS240G20B-00EPW0 READ: 500 mb/sec WRITE: 500 mb/sec

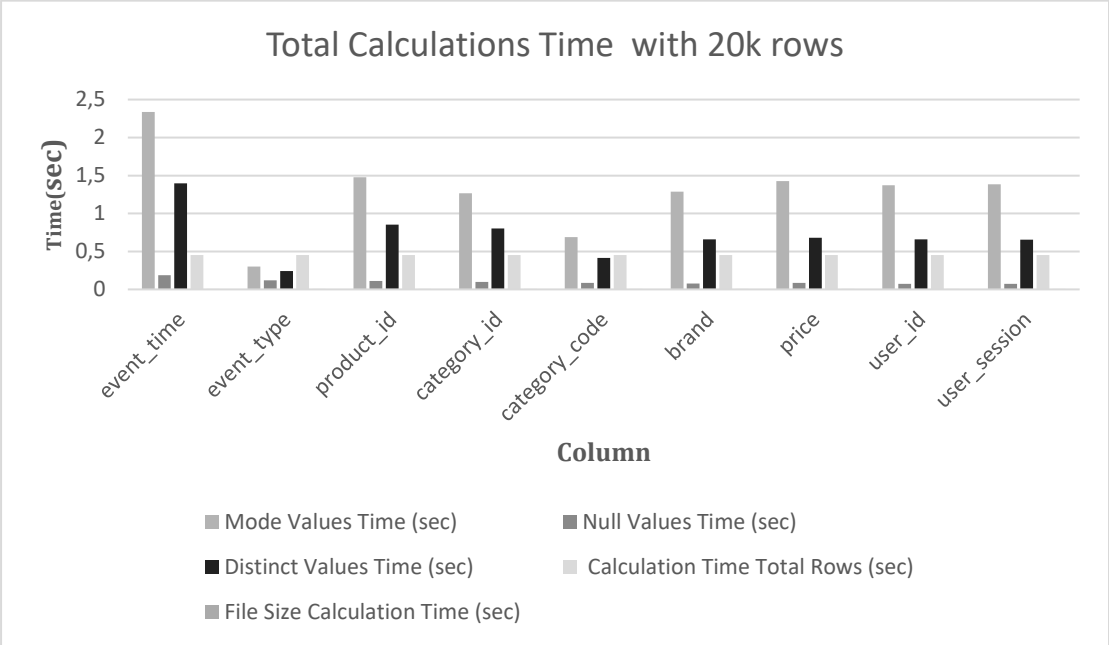
4.2 Αναλυτική παρουσίαση αποτελεσμάτων

Στην υπό ενότητα αυτή παρατίθενται τα αποτελέσματα των μετρήσεων που αναφέρθηκαν παραπάνω. Ο πίνακας 1 περιέχει τους συνολικούς χρόνους υπολογισμού των τιμών mode, distinct, null για dataset με 20.000 εγγραφές.

Experiment	Column	Mode Values Time (sec)	Null Values Time(sec)	Distinct Values Time (sec)	Calculation Time Total Rows (sec)	File Size Calculation Time (sec)
20k	event_time	2,336	0,189	1	0,453	0,007
20k	event_type	0,3	0,12	0,24	0,453	0,007
20k	product_id	1	0,11	0,852	0,453	0,007
20k	category_id	1,268	0,099	0,804	0,453	0,007
20k	category_code	0,69	0,085	0,414	0,453	0,007
20k	brand	1,288	0,078	0,661	0,453	0,007
20k	price	1,428	0,087	0,682	0,453	0,007
20k	user_id	1,37	0,075	0,66	0,453	0,007
20k	user_session	1,383	0,075	0,656	0,453	0,007

Πίνακας 1 Συνολικοί χρόνοι εκτέλεσης για την εύρεση mode, Null και Distinct τιμών για κάθε στήλη.

Στο διάγραμμα 1 παρακάτω φαίνονται επίσης οι χρόνοι εκτέλεσης.



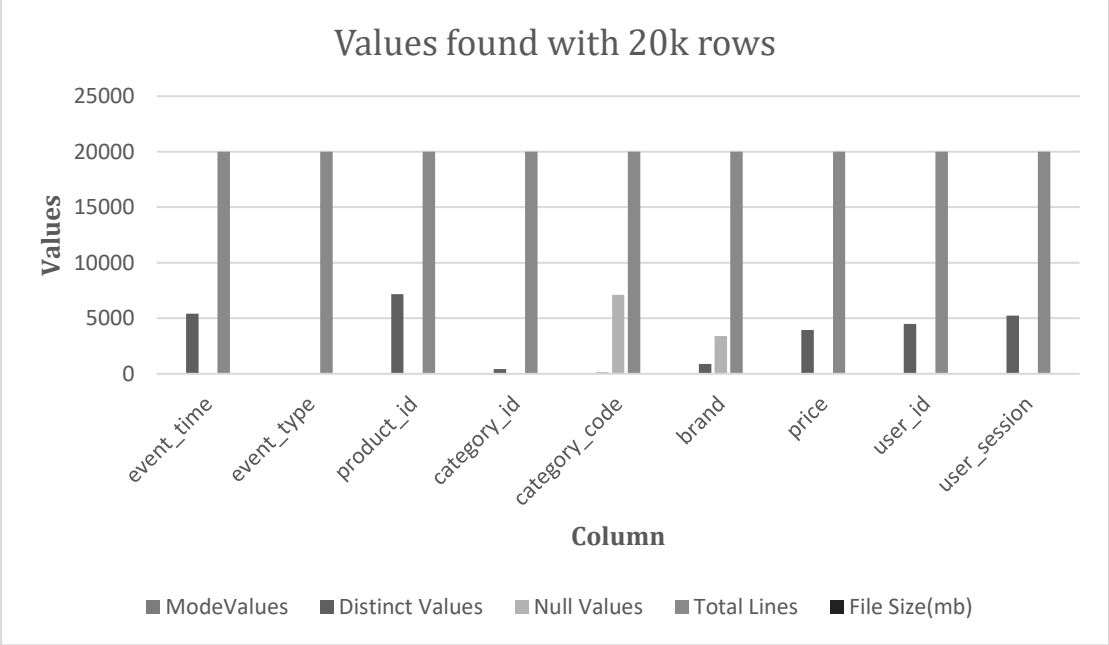
Διάγραμμα 1 Συνολικοί χρόνοι εκτέλεσης για την εύρεση mode, Null και Distinct τιμών για κάθε στήλη με εγγραφές 20.000.

Ο πίνακας 2 περιέχει τις τιμές που βρέθηκαν για την ανάλυση του πίνακα 1.

Experiment	Column	Mode Values	Distinct Values	Null Values	Total Lines	File Size(Mb)
20000	event_time	1	5428	0	20000	2,55
20000	event_type	1	3	0	20000	2,55
20000	product_id	1	7182	0	20000	2,55
20000	category_id	1	433	0	20000	2,55
20000	category_code	1	108	7110	20000	2,55
20000	brand	1	898	3401	20000	2,55
20000	price	1	3943	0	20000	2,55
20000	user_id	1	4503	0	20000	2,55
20000	user_session	1	5231	0	20000	2,55

Πίνακας 2 Υπολογισθέντες τιμές για εγγραφές 20000.

Στο διάγραμμα 2 φαίνονται οι τιμές που υπολογίστηκαν.



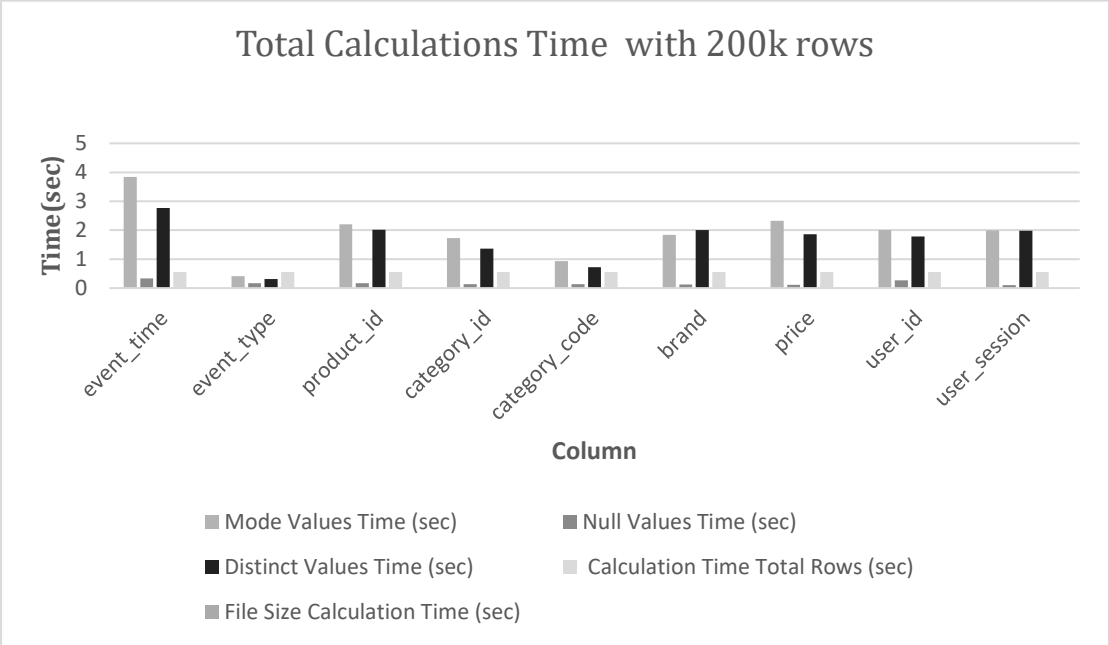
Διάγραμμα 2 Τιμές που υπολογίστηκαν στα δεδομένα 20.000 εγγραφών.

Ο πίνακας 3 περιέχει τους χρόνους εκτέλεσης που βρέθηκαν για dataset 200.000 εγγραφές.

Experiment	Column	Mode Values Time (sec)	Null Values Time(sec)	Distinct Values Time(sec)	Calculation Time Total Rows (sec)	File Size Calculation Time (sec)
200k	event_time	3,837	0,33	3	0,561	0,007
200k	event_type	0,418	0,169	0,309	0,561	0,007
200k	product_id	2,202	0,169	2	0,561	0,007
200k	category_id	1,724	0,133	1	0,561	0,007
200k	category_code	0,931	0,14	0,717	0,561	0,007
200k	brand	1,834	0,125	2	0,561	0,007
200k	price	2,33	0,115	2	0,561	0,007
200k	user_id	2,007	0,269	2	0,561	0,007
200k	user_session	1,999	0,1	2	0,561	0,007

Πίνακας 3 Συνολικοί χρόνοι εκτέλεσης για την εύρεση mode, Null και Distinct τιμών για κάθε στήλη με εγγραφές 200.000.

Το διάγραμμα 3 έχει τον συνολικό χρόνο εκτέλεσης για εγγραφές 200.000.



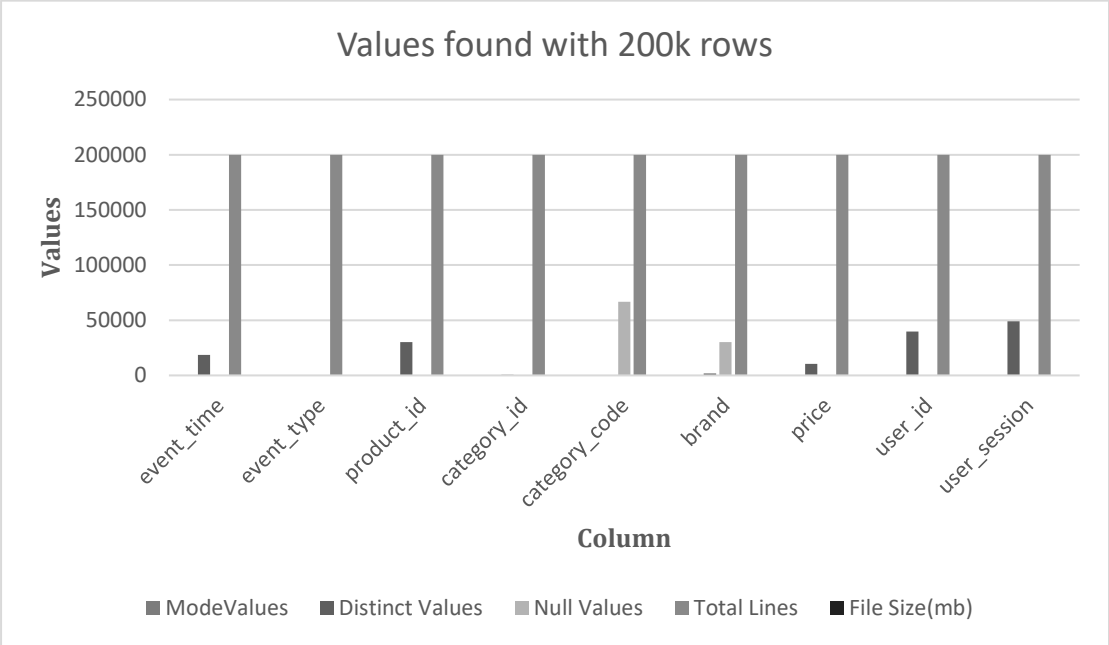
Διάγραμμα 3 Συνολικός χρόνος εκτέλεσης για 200.000 εγγραφές.

Ο πίνακας 4 περιέχει τις τιμές που υπολογίστηκαν.

Experiment	Column	Mode Values	Distinct Values	Null Values	Total Lines	File Size(Mb)
200000	event_time	1	12489	0	200000	25,5
200000	event_type	1	3	0	200000	25,5
200000	product_id	1	22000	0	200000	25,5
200000	category_id	1	512	0	200000	25,5
200000	category_code	1	179	71100	200000	25,5
200000	brand	1	1779	34010	200000	25,5
200000	price	1	10887	0	200000	25,5
200000	user_id	1	11156	0	200000	25,5
200000	user_session	1	14199	0	200000	25,5

Πίνακας 4 Υπολογισθέντες τιμές με εγγραφές 200.000.

Στο διάγραμμά 4 φαίνονται οι τιμές που υπολογίστηκαν για 200.000 εγγραφές.



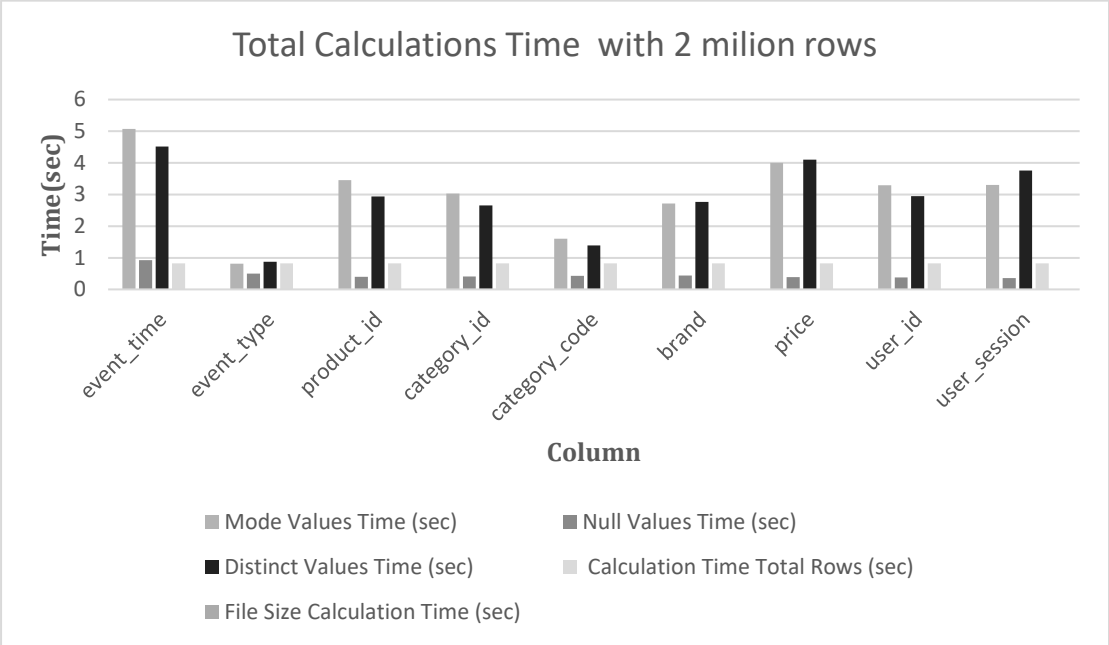
Διάγραμμα 4 Υπολογισθέντες τιμές για 200.000 εγγραφές.

Ο πίνακας 5 περιέχει του χρόνους εκτέλεσης και υπολογισμού για 2.000.000 εγγραφές.

Experiment	Column	Mode Values Time (sec)	Null Values Time(sec)	Distinct Values Time(sec)	Calculation Time Total Rows (sec)	File Size Calculation Time (sec)
2m	event_time	5,07	0,924	5	0,82	0,006
2m	event_type	0,813	0,504	0,876	0,82	0,006
2m	product_id	3,451	0,4	3	0,82	0,006
2m	category_id	3,03	0,413	3	0,82	0,006
2m	category_code	1,6	0,429	1	0,82	0,006
2m	brand	2,711	0,436	3	0,82	0,006
2m	price	3,997	0,389	4	0,82	0,006
2m	user_id	3,289	0,378	3	0,82	0,006
2m	user_session	3,3	0,361	4	0,82	0,006

Πίνακας 5 Συνολικοί χρόνοι εκτέλεσης για εγγραφές 2.000.000.

Το διάγραμμα 5 παρουσιάζει επίσης τους χρόνους εκτέλεσης για 2.000.000 εγγραφές.



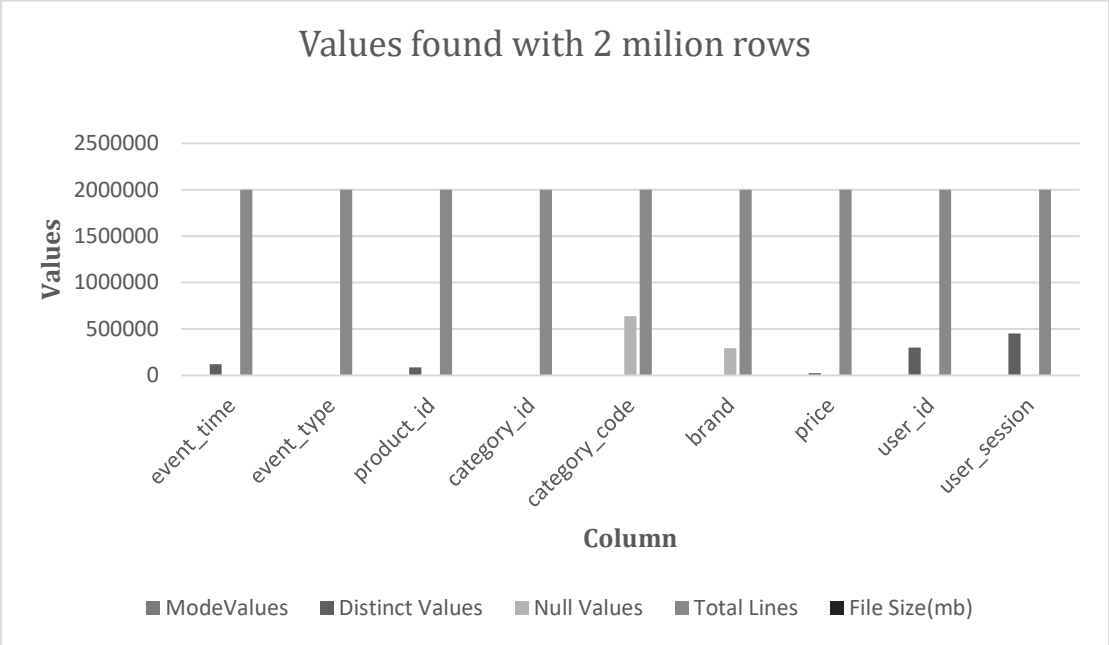
Διάγραμμα 5 Χρόνοι εκτέλεσης με 2.000.000 εγγραφές

Ο πίνακας 6 αποτελείται με τις τιμές που υπολογίστηκαν κατά το πείραμα στον πίνακα 5.

Experiment	Column	Mode Values	Distinct Values	Null Values	Total Lines	File Size(Mb)
2000000	event_time	1	168664	0	2000000	255
2000000	event_type	1	3	0	2000000	255
2000000	product_id	1	199998	0	2000000	255
2000000	category_id	1	512	0	2000000	255
2000000	category_code	1	224	711000	2000000	255
2000000	brand	1	2664	340100	2000000	255
2000000	price	1	25774	0	2000000	255
2000000	user_id	1	11156	0	2000000	255
2000000	user_session	1	200000	0	2000000	255

Πίνακας 6 Υπολογισθέντες τιμές για 2.000.000 εγγραφές.

Το διάγραμμα 6 παρουσιάζει τις τιμές αυτές που υπολογίστηκαν για 2.000.000 εγγραφές.



Διάγραμμα 6 Υπολογισθέντες τιμές για 2.000.000 εγγραφές.

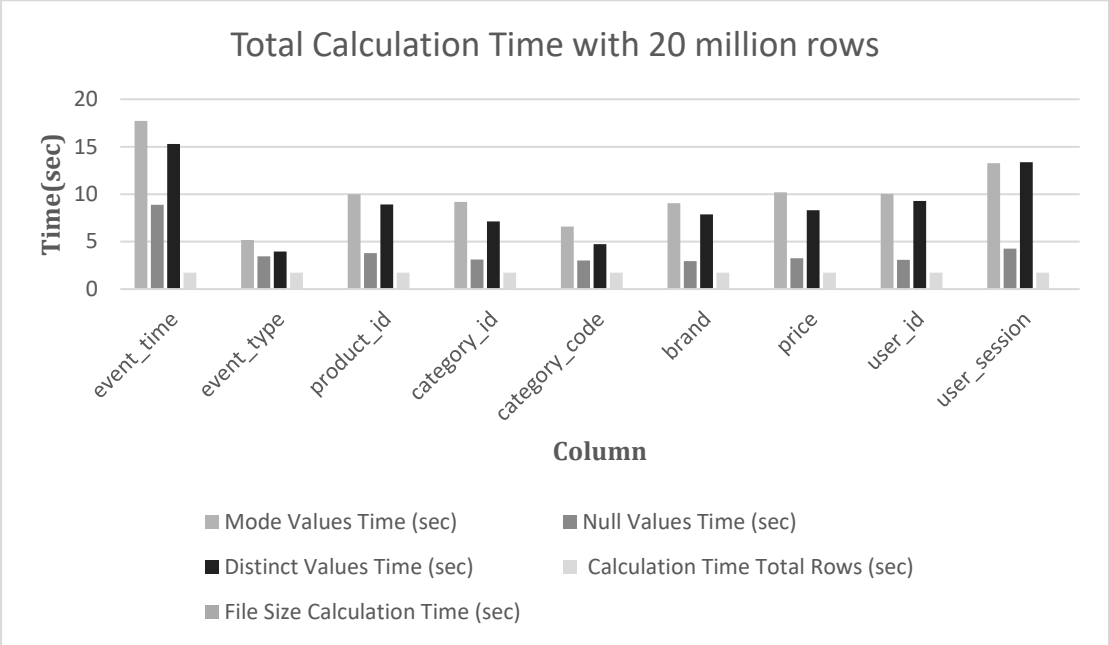
Ο πίνακας 7 αποτελείτε από τους υπολογισμένους χρόνους για 20.000.000 εγγραφές.

Experiment	Column	Mode Values Time (sec)	Null Values Time(sec)	Distinct Values Time (sec)	Calculation Time Total Rows (sec)	File Size Calculation Time (sec)
20m	event_time	17,707	9	15	1,724	0,008
20m	event_type	5,184	3	4	1,724	0,008
20m	product_id	9,979	4	9	1,724	0,008
20m	category_id	9,182	3	7	1,724	0,008
20m	category_code	6,593	3	5	1,724	0,008
20m	brand	9,064	3	8	1,724	0,008
20m	price	10,204	3	8	1,724	0,008
20m	user_id	10,015	3	9	1,724	0,008

20m	user_session	13,265	4	13	1,724	0,008

Πίνακας 7 Συνολικοί χρόνοι εκτέλεσης για 20.000.000 εγγραφές.

Στο διάγραμμα 7 φαίνονται οι χρόνοι αυτοί.



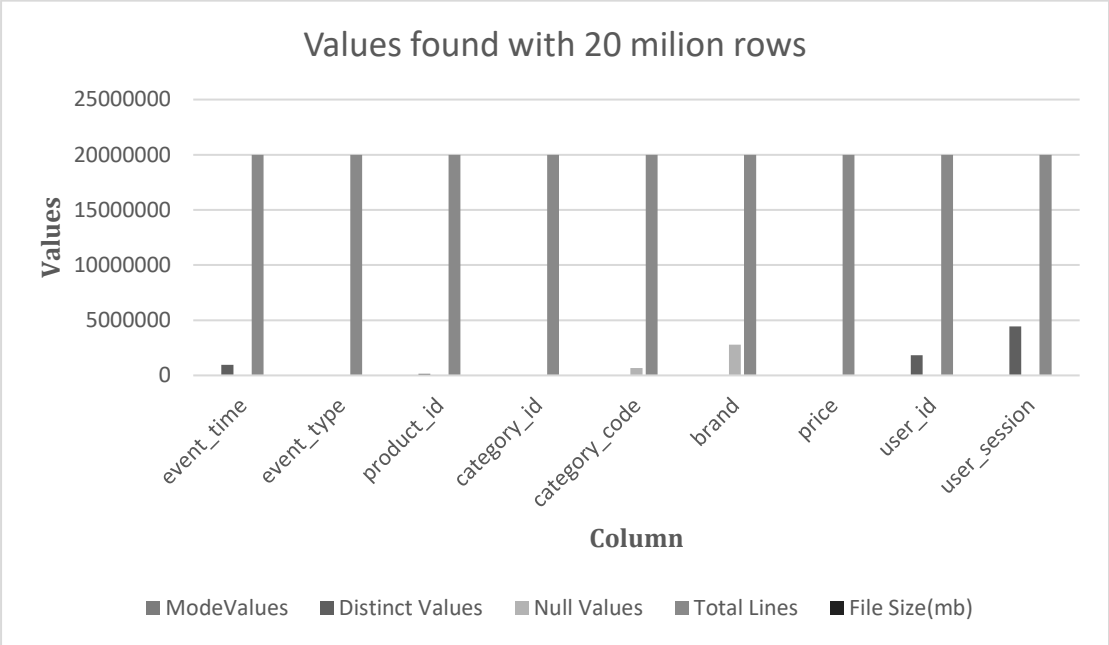
Διάγραμμα 7 Συνολικοί χρόνοι εκτέλεσης με εγγραφές 20.000.000.

Ο πίνακας 8 έχει τα αποτελέσματα τις τιμές που βρέθηκαν με εγγραφές 20.000.000.

Experiment	Column	Mode Values	Distinct Values	Null Values	Total Lines	File Size (Mb)
20000000	event_time	1	1709971	0	20000000	2553
20000000	event_type	1	3	0	20000000	2553
20000000	product_id	1	1999999	0	20000000	2553
20000000	category_id	1	512	0	20000000	2553
20000000	category_code	1	224	7110000	20000000	2553
20000000	brand	1	2664	3401000	20000000	2553
20000000	price	1	25774	0	20000000	2553
20000000	user_id	1	11156	0	20000000	2553
20000000	user_session	1	2000000	0	20000000	2553

Πίνακας 8 Υπολογισθέντες τιμές για 20.000.000 εγγραφές.

Στο διάγραμμα 8 φαίνονται οι τιμές που υπολογίσθηκαν για 20.000.000 εγγραφές.



Διάγραμμα 8 Υπολογισθέντες τιμές για 20.000.000 εγγραφές.

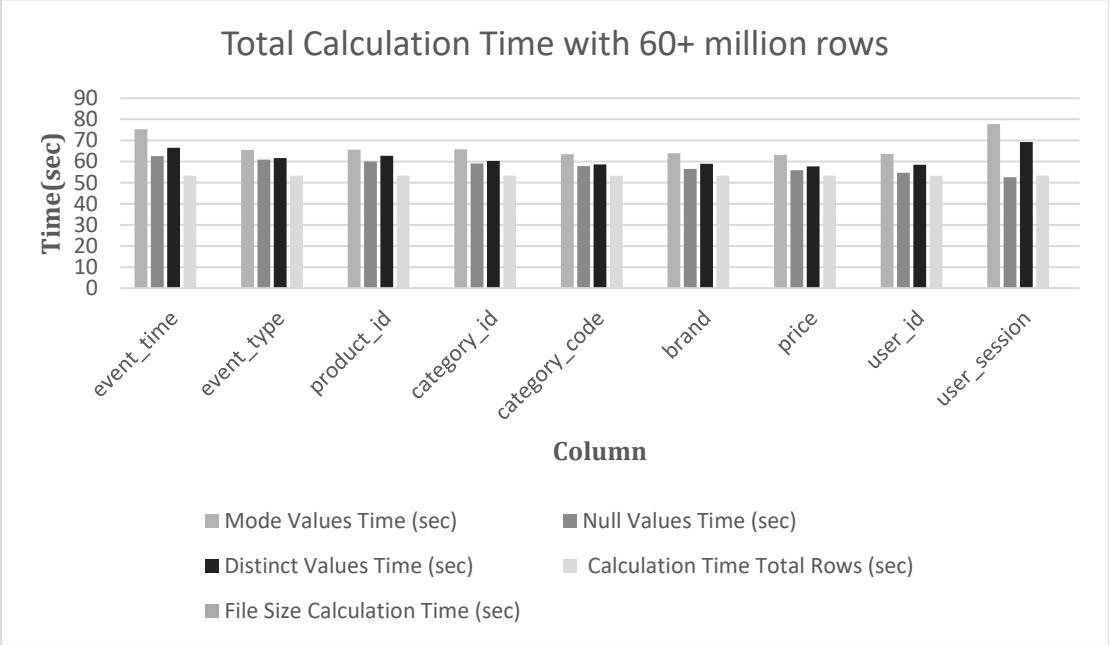
Ο πίνακας 9 έχει του συνολικούς χρόνου εκτέλεσης για ολόκληρο το dataset με εγγραφές 60 εκατομμυρίων και άνω.

Experiment	Column	Mode Values Time (sec)	Null Values Time(sec)	Distinct Values Time (sec)	Calculation Time Total Rows (sec)	File Size Calculation Time (sec)
60m+	event_time	75,416	63	66	53,319	0,01
60m+	event_type	65,411	61	62	53,319	0,01
60m+	product_id	65,696	60	63	53,319	0,01
60m+	category_id	65,828	59	60	53,319	0,01
60m+	category_code	63,51	58	59	53,319	0,01
60m+	brand	63,977	57	59	53,319	0,01

60m+	price	63,254	56	58	53,319	0,01
60m+	user_id	63,719	55	59	53,319	0,01
60m+	user_session	77,718	53	69	53,319	0,01

Πίνακας 9 Συνολικοί χρόνοι εκτέλεσης για εγγραφές άνω των 60 εκατομμυρίων.

Το διάγραμμα 9 παρουσιάζει τους χρόνους αυτούς.



Διάγραμμα 9 Συνολικοί χρόνοι εκτέλεσης για εγγραφές άνω των 60 εκατομμυρίων.

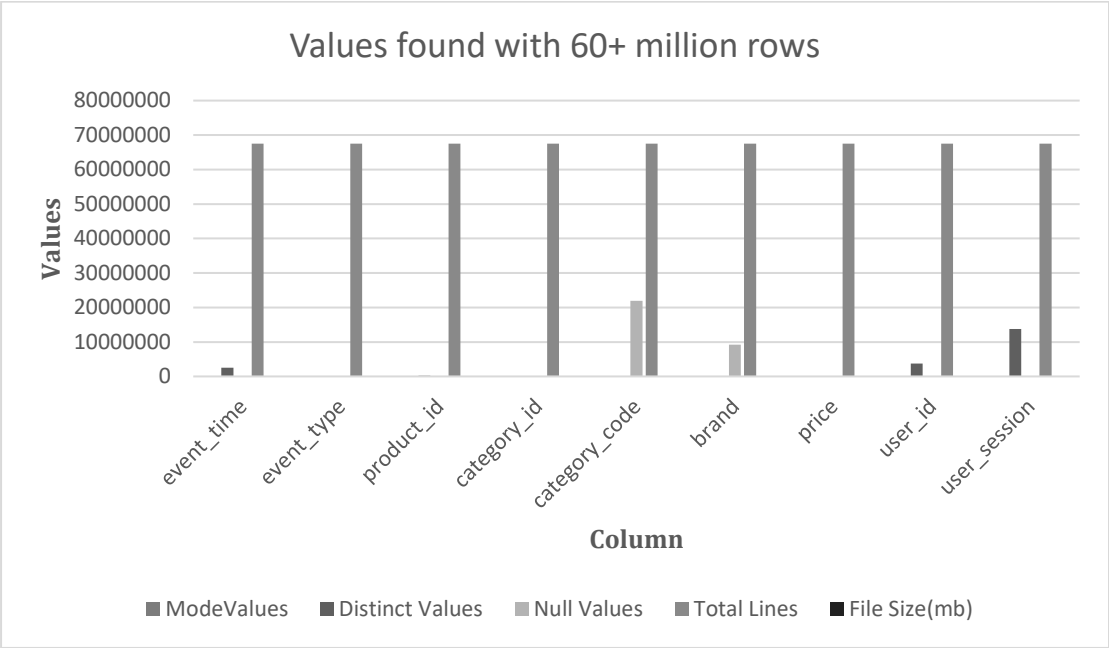
Ο πίνακας 10 έχει τα αποτελέσματα των υπολογισμών αυτών.

Experiment	Column	Mode Values	Distinct Values	Null Values	Total Lines	File Size (Mb)
60000000+	event_time	1	1712211	0	67501979	8593
60000000+	event_type	1	3	0	67501979	8593
60000000+	product_id	1	1999999	0	67501979	8593
60000000+	category_id	1	512	0	67501979	8593
60000000+	category_code	1	224	21330000	67501979	8593
60000000+	brand	1	2664	10203000	67501979	8593
60000000+	price	1	25774	0	67501979	8593

60000000+	user_id	1	11156	0	67501979	8593
60000000+	user_session	1	6750197	0	67501979	8593

Πίνακας 10 Υπολογισθέντες τιμές για εγγραφές άνω των 60.000.000.

Το διάγραμμα 10 παρουσιάζει τις τιμές αυτές.



Διάγραμμα 10 Υπολογισθέντες τιμές για εγγραφές άνω των 60.000.0000.

Ο πίνακας 11 έχει όλες τις πληροφορίες ομαδοποιημένες.

Experiment	Column	Mode Values Time(sec)	Null Values Time (sec)	Distinct Values Time (sec)	Calculation Time Total Rows (sec)	File Size Calculation Time (sec)	Mode Values	Distinct Values	Null Values	Total Lines	File Size (Mb)
20k	event_time	2,336	0,189	1	0,453	0,007	1	5428	0	20000	2,55
20k	event_type	0,3	0,12	0,24	0,453	0,007	1	3	0	20000	2,55

20k	product_id	1	0,11	0,852	0,453	0,007	1	7182	0	20000	2,55
20k	category_id	1,268	0,099	0,804	0,453	0,007	1	433	0	20000	2,55
20k	category_code	0,69	0,085	0,414	0,453	0,007	1	108	7110	20000	2,55
20k	brand	1,288	0,078	0,661	0,453	0,007	1	898	3401	20000	2,55
20k	price	1,428	0,087	0,682	0,453	0,007	1	3943	0	20000	2,55
20k	user_id	1,37	0,075	0,66	0,453	0,007	1	4503	0	20000	2,55
20k	user_session	1,383	0,075	0,656	0,453	0,007	1	5231	0	20000	2,55
200k	event_time	3,837	0,33	3	0,561	0,007	1	18597	0	200000	25,6
200k	event_type	0,418	0,169	0,309	0,561	0,007	1	3	0	200000	25,6
200k	product_id	2,202	0,169	2	0,561	0,007	1	30247	0	200000	25,6
200k	category_id	1,724	0,133	1	0,561	0,007	1	579	0	200000	25,6

200k	category_code	0,931	0,14	0,717	0,561	0,007	1	124	66642	200000	25,6
200k	brand	1,834	0,125	2	0,561	0,007	1	1899	30273	200000	25,6
200k	price	2,33	0,115	2	0,561	0,007	1	10601	0	200000	25,6
200k	user_id	2,007	0,269	2	0,561	0,007	1	39836	0	200000	25,6
200k	user_session	1,999	0,1	2	0,561	0,007	1	491800	0	200000	25,6
2m	event_time	5,07	0,924	5	0,82	0,006	1	1193330	0	2000000	256,48
2m	event_type	0,813	0,504	0,876	0,82	0,006	1	3	0	2000000	256,48
2m	product_id	3,451	0,4	3	0,82	0,006	1	871211	0	2000000	256,48
2m	category_id	3,03	0,413	3	0,82	0,006	1	611	0	2000000	256,48
2m	category_code	1,6	0,429	1	0,82	0,006	1	124	638812	2000000	256,48

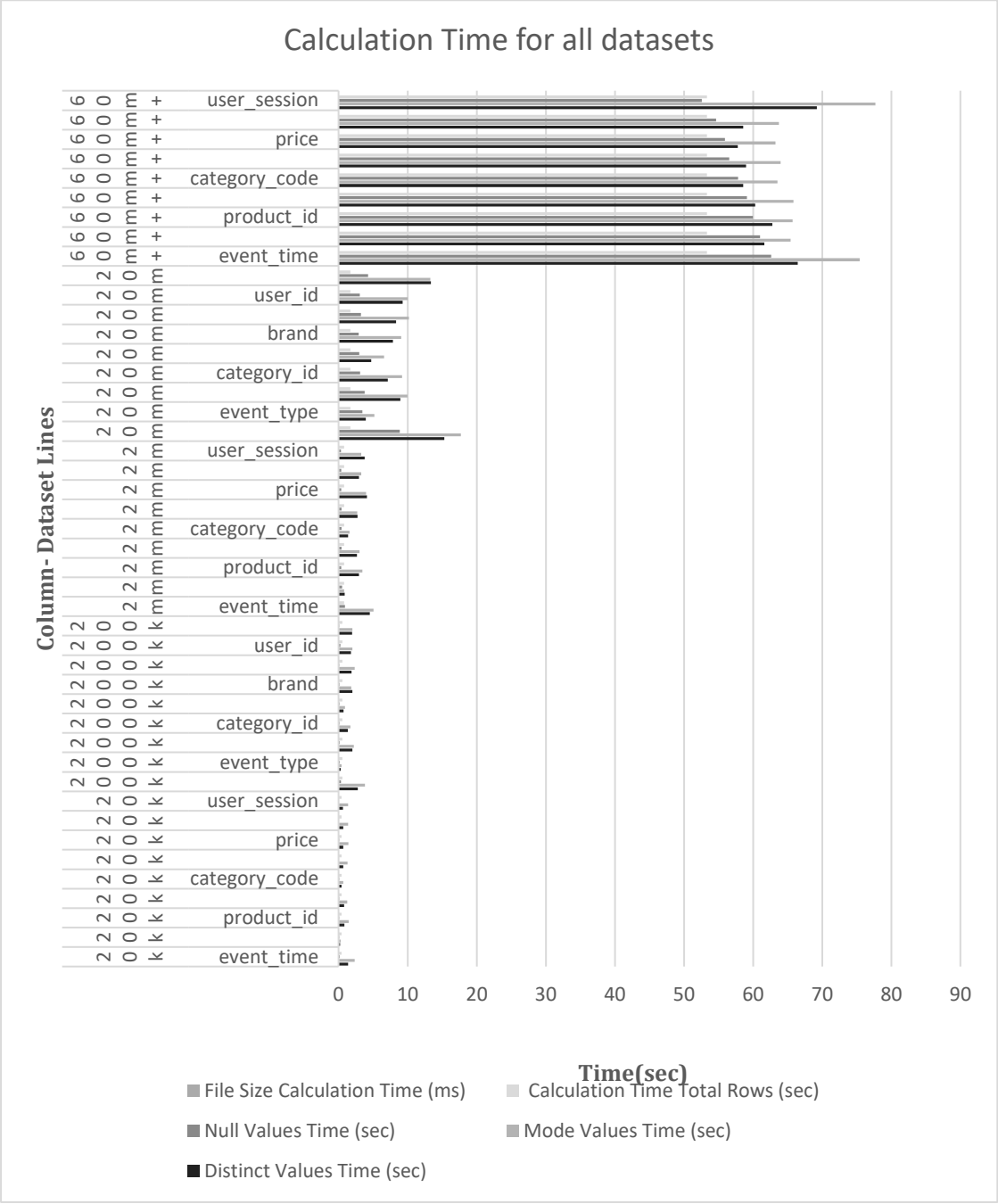
2m	brand	2,711	0,436	3	0,82	0,006	1	2922	293648	2000000	256,48
2m	price	3,997	0,389	4	0,82	0,006	1	23673	0	2000000	256,48
2m	user_id	3,289	0,378	3	0,82	0,006	1	300008	0	2000000	256,48
2m	user_session	3,3	0,361	4	0,82	0,006	1	451259	0	2000000	256,48
20m	event_time	17,707	9	15	1,724	0,008	1	964020	0	20000000	2560,13
20m	event_type	5,184	3	4	1,724	0,008	1	3	0	20000000	2560,13
20m	product_id	9,979	4	9	1,724	0,008	1	154539	0	20000000	2560,13
20m	category_id	9,182	3	7	1,724	0,008	1	640	0	20000000	2560,13
20m	category_code	6,593	3	5	1,724	0,008	1	127	657806	20000000	2560,13
20m	brand	9,064	3	8	1,724	0,008	1	3667	2801204	20000000	2560,13

20m	price	10,204	3	8	1,724	0,008	1	44663	0	20000000	2560,13
20m	user_id	10,015	3	9	1,724	0,008	1	18416686	0	20000000	2560,13
20m	user_session	13,265	4	13	1,724	0,008	1	44408851	2	20000000	2560,13
60m+	event_time	75,416	63	66	53,319	0,01	1	254955 9	0	67501979	8589,52
60m+	event_type	65,411	61	62	53,319	0,01	1	3	0	67501979	8589,52
60m+	product_id	65,696	60	63	53,319	0,01	1	190662	0	67501979	8589,52
60m+	category_id	65,828	59	60	53,319	0,01	1	684	0	67501979	8589,52
60m+	category_code	63,51	58	59	53,319	0,01	1	130	21898171	67501979	8589,52
60m+	brand	63,977	57	59	53,319	0,01	1	4202	9218235	67501979	8589,52
60m+	price	63,254	56	58	53,319	0,01	1	60435	0	67501979	8589,52

60m+	user_id	63,719	55	59	53,319	0,01	1	36961167	0	67501979	8589,52
60m+	user_session	77,718	53	69	53,319	0,01	1	13776051	10	67501979	8589,52

Πίνακας 11 Αποτελείται με όλα τα αποτελέσματα ομαδοποιημένα: α) Συνολικούς χρόνους εκτέλεσης για κάθε σενάριο εγγραφών αλλά και υπολογισμένων τιμών ανά περίπτωση.

Το διάγραμμα 11 παρουσιάζει τους συνολικούς χρόνους εκτέλεσης για όλα τα πειράματα δηλ., για κάθε σύνολο εγγραφών.



Διάγραμμα 11 Αποτελείται από όλους τους χρόνους εκτέλεσης για όλα τα σύνολα δεδομένων.

Αξιολογώντας τις παραπάνω μετρήσεις, διαπιστώνουμε ότι ο υπολογισμός των επικρατέστερων τιμών (mode) είναι ιδιαίτερα χρονοβόρος, ανεξαρτήτως του μεγέθους των δεδομένων. Αυτό οφείλεται στο γεγονός ότι η διαδικασία υπολογισμού απαιτεί την καταμέτρηση της συχνότητας εμφάνισης κάθε μοναδικής τιμής σε μια στήλη δεδομένων. Συγκεκριμένα, πρέπει να προσδιοριστεί πόσες φορές εμφανίζεται κάθε τιμή, ώστε να εντοπιστούν οι τιμές με τη μεγαλύτερη συχνότητα. Αυτή η διαδικασία καταμέτρησης και

σύγκρισης είναι υπολογιστικά απαιτητική, ειδικά για σύνολα δεδομένων με μεγάλο εύρος διαφορετικών τιμών.

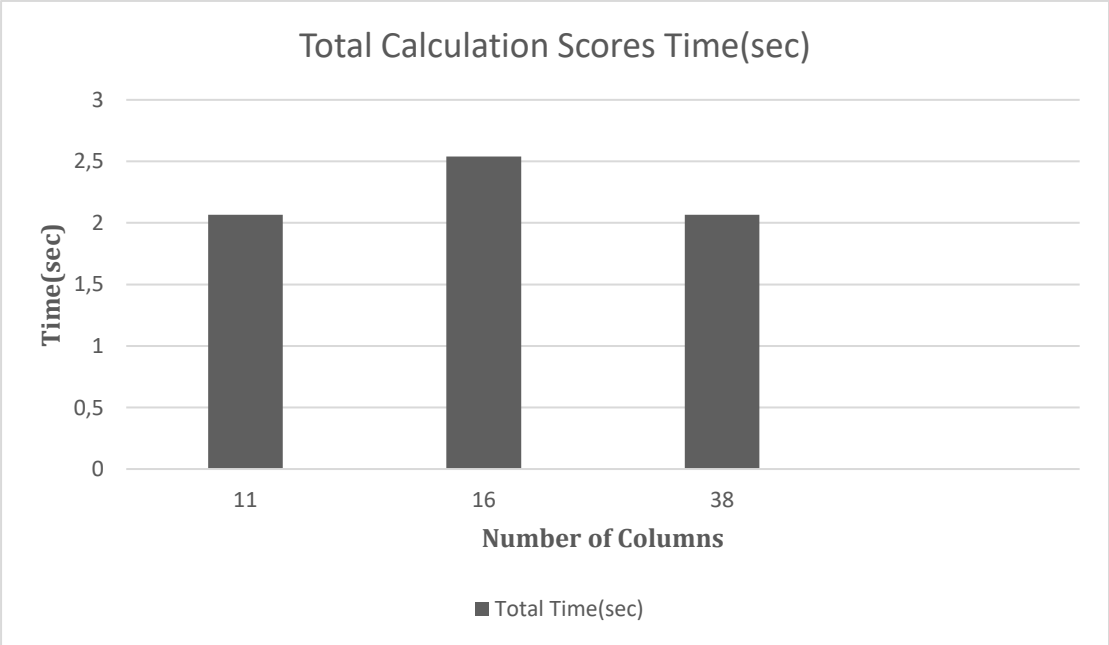
Επιπλέον, ο χρόνος υπολογισμού των null και distinct τιμών επηρεάζεται σημαντικά από το μέγεθος των αρχείων. Όσο μεγαλύτερο είναι το dataset, τόσο περισσότερος χρόνος απαιτείται για την επεξεργασία αυτών των τιμών. Για παράδειγμα, σε μικρά datasets (π.χ., 20k εγγραφές), ο χρόνος υπολογισμού των null και distinct τιμών είναι σχετικά μικρός. Ωστόσο, σε μεγάλα datasets (π.χ., 60m+ εγγραφές), ο χρόνος αυξάνεται δραματικά λόγω της αύξησης του όγκου των δεδομένων που πρέπει να εξεταστούν.

Ο πίνακας 12 παρουσιάζει τους χρόνους υπολογισμών των συνολικών score για διαφορετικά dataset με διαφορετικό αριθμό στηλών.

Experiment	Tota Time(sec)	Number of Columns
1	2,0675	11
2	2,5402	16
3	2,065	38

Πίνακας 12 Συνολικοί χρόνοι εύρεσης score Type για datasets με διαφορετικό αριθμό στηλών.

Το διάγραμμα 12 παρουσιάζει τους συνολικούς χρόνους υπολογισμού των score για κάθε dataset με διαφορετικό αριθμό στηλών.



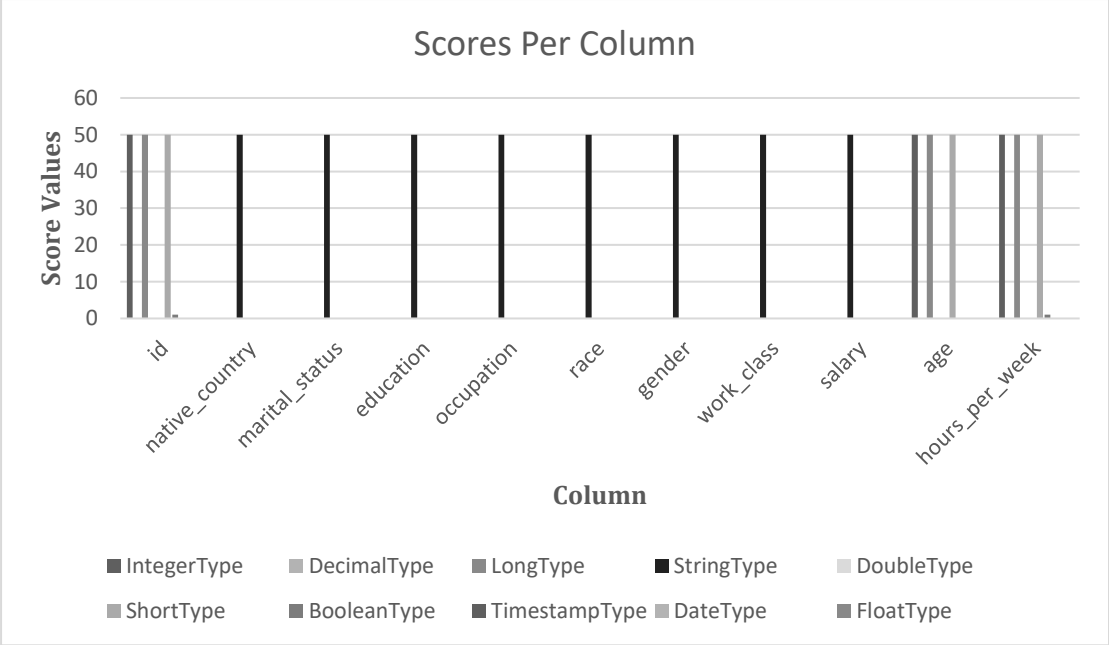
Διάγραμμα 12 Συνολικοί χρόνοι υπολογισμών scores σε διαφορετικά dataset με διαφορετικό αριθμό στηλών.

Ο πίνακας 13 δείχνει για τα αποτελέσματα των score για το dataset με 11 στήλες.

Column Name	Integer Type	Decimal Type	Long Type	String Type	Double Tupe	Short Type	Boolean Type	Timestamp Type	Date Type	Float Type	Total Time Calculations (sec)
id	50	0	50	0	0	50	1	0	0	0	1,968327
native_county	0	0	0	50	0	0	0	0	0	0	1,968327
marital_status	0	0	0	50	0	0	0	0	0	0	1,968327
education	0	0	0	50	0	0	0	0	0	0	1,968327
occupation	0	0	0	50	0	0	0	0	0	0	1,968327
race	0	0	0	50	0	0	0	0	0	0	1,968327
gender	0	0	0	50	0	0	0	0	0	0	1,968327
work_class	0	0	0	50	0	0	0	0	0	0	1,968327
salary	0	0	0	50	0	0	0	0	0	0	1,968327
age	50	0	50	0	0	50	0	0	0	0	1,968327
hours_per_week	50	0	50	0	0	50	1	0	0	0	1,968327

Πίνακας 13 Αποτελέσματα score για κάθε στήλη του dataset.

Το διάγραμμα 13 δείχνει τα score που υπολογίστηκαν για κάθε στήλη.



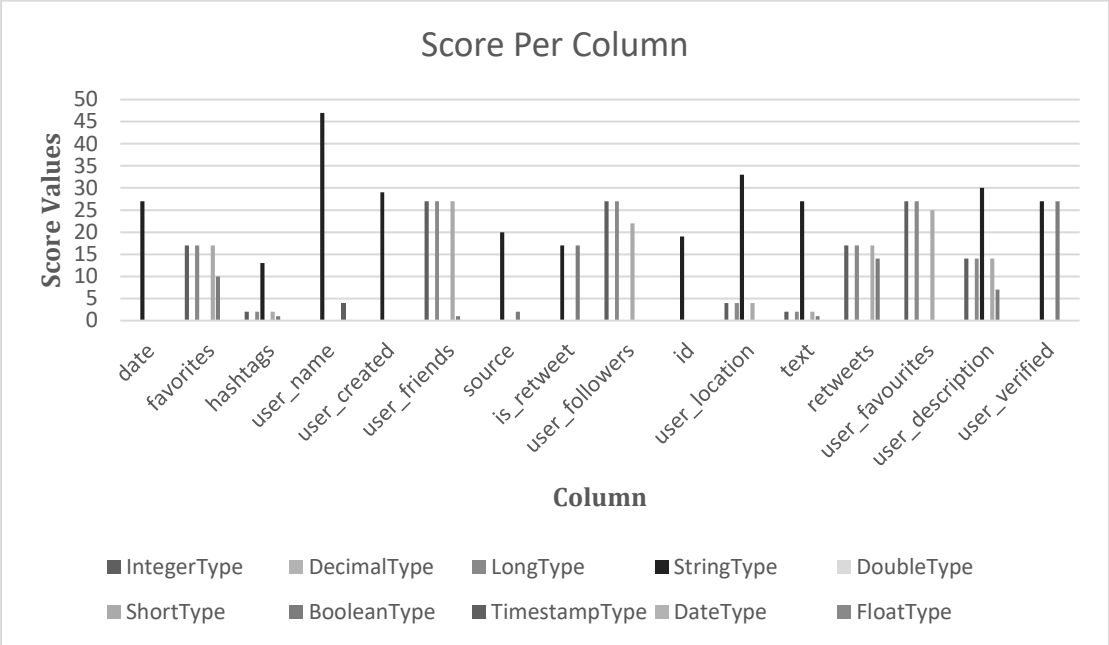
Διάγραμμα 13 Score για κάθε στήλη.

Ο πίνακας 14 δείχνει τα score που υπολογίστηκαν για το dataset που έχει 16 στήλες.

Column Name	Integer Type	Decimal Type	Long Type	String Type	Double Tupe	Short Type	Boolean Type	Timestamp Type	Date Type	Float Type
date	0	0	0	27	0	0	0	0	0	0
favorites	17	0	17	0	0	17	10	0	0	0
hashtags	2	0	2	13	0	2	1	0	0	0
user_name	0	0	0	47	0	0	0	4	0	0
user_created	0	0	0	29	0	0	0	0	0	0
user_friends	27	0	27	0	0	27	1	0	0	0
source	0	0	0	20	0	0	2	0	0	0
is_retweet	0	0	0	17	0	0	17	0	0	0
user_followers	27	0	27	0	0	22	0	0	0	0
id	0	0	0	19	0	0	0	0	0	0
user_location	4	0	4	33	0	4	0	0	0	0
text	2	0	2	27	0	2	1	0	0	0
retweets	17	0	17	0	0	17	14	0	0	0
user_favourites	27	0	27	0	0	25	0	0	0	0
user_description	14	0	14	30	0	14	7	0	0	0
user_verified	0	0	0	27	0	0	27	0	0	0

Πίνακας 14 Score για κάθε στήλη από dataset με 16 στήλες.

Το διάγραμμα 14 δείχνει τα score για κάθε στήλη.



Διάγραμμα 14 Score για κάθε στήλη για το αρχείο με 16 στήλες.

Ο πίνακας 15 δείχνει τα score που υπολογίστηκαν για dataset με 38 στήλες.

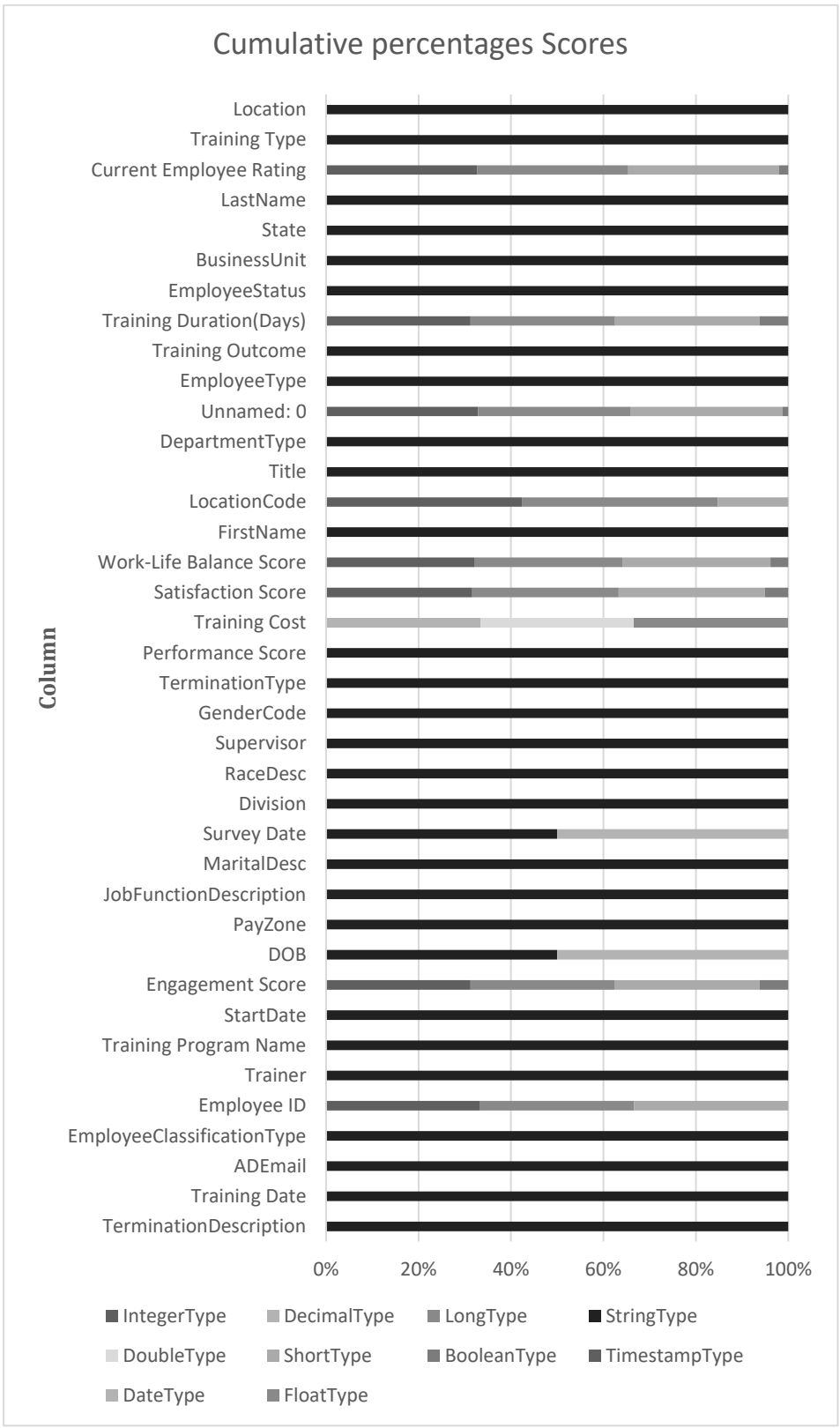
Column Name	Integer Type	Decimal Type	Long Type	String Type	Double Tupe	Short Type	Boolean Type	Timestamp Type	Date Type	Float Type
TerminationDescription	0	0	0	26	0	0	0	0	0	0
Training Date	0	0	0	50	0	0	0	0	0	0
ADEmail	0	0	0	50	0	0	0	0	0	0
EmployeeClassificationType	0	0	0	50	0	0	0	0	0	0
Employee ID	50	0	50	0	0	50	0	0	0	0
Trainer	0	0	0	50	0	0	0	0	0	0
Training Program Name	0	0	0	50	0	0	0	0	0	0
StartDate	0	0	0	50	0	0	0	0	0	0
Engagement Score	50	0	50	0	0	50	10	0	0	0
DOB	0	0	0	50	0	0	0	0	50	0
PayZone	0	0	0	50	0	0	0	0	0	0

JobFunctionDescription	0	0	0	50	0	0	0	0	0	0
MaritalDesc	0	0	0	50	0	0	0	0	0	0
Survey Date	0	0	0	50	0	0	0	0	50	0
Division	0	0	0	50	0	0	0	0	0	0
RaceDesc	0	0	0	50	0	0	0	0	0	0
Supervisor	0	0	0	50	0	0	0	0	0	0
GenderCode	0	0	0	50	0	0	0	0	0	0
TerminationType	0	0	0	50	0	0	0	0	0	0
Performance Score	0	0	0	50	0	0	0	0	0	0
Training Cost	0	50	0	0	50	0	0	0	0	50
Satisfaction Score	50	0	50	0	0	50	8	0	0	0
Work-Life Balance Score	50	0	50	0	0	50	6	0	0	0
FirstName	0	0	0	50	0	0	0	0	0	0
LocationCode	50	0	50	0	0	18	0	0	0	0
Title	0	0	0	50	0	0	0	0	0	0
DepartmentType	0	0	0	50	0	0	0	0	0	0
Unnamed: 0	50	0	50	0	0	50	2	0	0	0
EmployeeType	0	0	0	50	0	0	0	0	0	0
Training Outcome	0	0	0	50	0	0	0	0	0	0
Training Duration(Days)	50	0	50	0	0	50	10	0	0	0
EmployeeStatus	0	0	0	50	0	0	0	0	0	0
BusinessUnit	0	0	0	50	0	0	0	0	0	0
State	0	0	0	50	0	0	0	0	0	0
LastName	0	0	0	50	0	0	0	0	0	0
Current Employee Rating	50	0	50	0	0	50	3	0	0	0
Training Type	0	0	0	50	0	0	0	0	0	0

Location	0	0	0	50	0	0	0	0	0	0

Πίνακας 15 Score για κάθε στήλη.

Το διάγραμμα 15 δείχνει τα scores για κάθε στήλη.



Διάγραμμα 15 Συσσωρευμένες τιμές 100% για τα Score κάθε στήλης.

Αξιολογώντας τις μετρήσεις, διαπιστώνουμε ότι ο αριθμός των στηλών επηρεάζει τον συνολικό χρόνο εκτέλεσης των υπολογισμών των scores. Ωστόσο, ο χρόνος αυτός φαίνεται να είναι ανεξάρτητος από το συνολικό μέγεθος του αρχείου, καθώς για τον υπολογισμό των scores χρησιμοποιείται ένα σταθερό δείγμα 50 εγγραφών από το σύνολο δεδομένων.

Παρόλα αυτά, η παρουσία null ή κενών τιμών εντός αυτού του δείγματος επηρεάζει σημαντικά τον χρόνο εκτέλεσης. Αυτό είναι εμφανές στο πείραμα με τις 39 στήλες, όπου παρατηρούμε στον πίνακα των scores ότι για ορισμένες στήλες ο υπολογισμός έχει γίνει για λιγότερες από 50 εγγραφές. Αυτό δημιουργεί την ψευδαίσθηση ότι το πείραμα με τις 39 στήλες είναι ταχύτερο σε σύγκριση με το πείραμα με τις 16 στήλες, ενώ στην πραγματικότητα ο μικρότερος χρόνος οφείλεται στην επεξεργασία λιγότερων δεδομένων λόγω των null ή κενών τιμών.

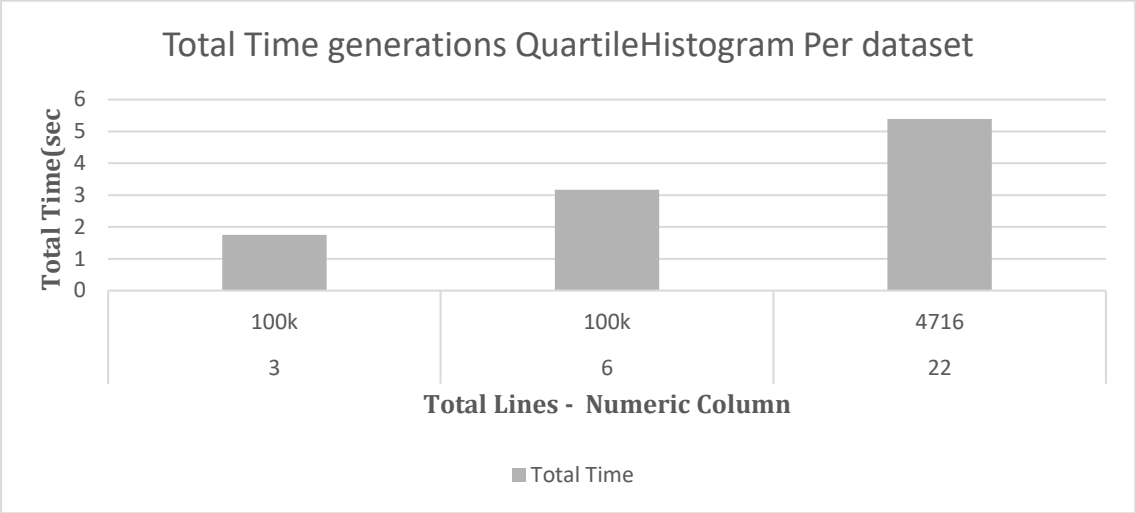
Επιπλέον, ο χρόνος επηρεάζεται άμεσα και από το είδος των δεδομένων σε κάθε στήλη. Όπως φαίνεται στο πείραμα με τις 16 στήλες, οι τιμές των δεδομένων σε ορισμένες στήλες αντιστοιχούν σε περισσότερες από μία κατηγορίες τύπων δεδομένων, γεγονός που αυξάνει τον χρόνο επεξεργασίας

Ο πίνακας 16 περιλαμβάνει συνολικούς χρόνους υπολογισμού δημιουργίας ισοϋψούς ιστογράμματος με την μορφή quartile για διαφορετικά dataset με διαφορετικό αριθμό σε στήλες που έχουν αριθμούς.

Numeric Columns	Total Lines	Total Time (sec)
3	100.000	1,753
6	100.000	3,1639
22	4716	5,3859

Πίνακας 16 Συνολικοί χρόνοι υπολογισμού διαγράμματος για διαφορετικά datasets με διαφορετικό αριθμό Numeric στηλών.

Το διάγραμμα 16 παρουσιάζει τους χρόνους αυτούς.



Διάγραμμα 16 Συνολικός χρόνος για κάθε dataset με διαφορετικό πλήθος Numeric στηλών

Αξιολογώντας τις μετρήσεις μπορεί να εξαχθεί το συμπέρασμα ότι ο αριθμός των Numeric στηλών σε ένα σύνολο δεδομένων επηρεάζει άμεσα το χρόνο δημιουργίας καθώς όσες πιο πολλές Numeric στήλες τόσα περισσότερα παραχθέντα διαγράμματα, άρα ο συνολικός χρόνος αυξάνεται.

Κεφάλαιο 5. Επίλογος

5.1 Σύνοψη και συμπεράσματα

Η συνεχής αύξηση της διαθεσιμότητας αλφαριθμητικών δεδομένων δημιούργησε την ανάγκη για αυτόματη και αποτελεσματική παραγωγή στατιστικού προφίλ. Η χειροκίνητη ανάλυση αυτών των δεδομένων είναι χρονοβόρα και επιρρεπής σε λάθη, ειδικά για μεγάλα και πολύπλοκα σύνολα δεδομένων.

Σε αυτή την εργασία, αναπτύχθηκε ένα εργαλείο που βελτίωσε και αυτοματοποίησε τη διαδικασία δημιουργίας στατιστικού προφίλ για αλφαριθμητικά σύνολα δεδομένων. Το εργαλείο προσφέρει ημιαυτόματη αναγνώριση του τύπου δεδομένων μιας στήλης, επιτρέποντας στον χρήστη να επιβεβαιώνει ή να τροποποιεί τους τύπους με βάση τα αποτελέσματα της αυτόματης ανάλυσης. Επιπλέον, υπολογίζει βασικές μετρικές, όπως τον συνολικό αριθμό γραμμών, το μέγεθος του αρχείου και την ημερομηνία δημιουργίας του προφίλ, ενώ παρέχει λεπτομερείς στατιστικές πληροφορίες, όπως τον αριθμό των null τιμών, τον αριθμό των διακριτών τιμών, την επικρατούσα τιμή (mode) και τα τεταρτημόρια (Q1, Q2, Q3).

Το εργαλείο δημιουργεί ισοϋψή ιστογράμματα και συνοπτικές αναφορές, οι οποίες περιλαμβάνουν όλες τις μετρικές που προστέθηκαν. Επιπλέον, υποστηρίζει διαδραστικές αιτήσεις profiling, επιτρέποντας στους χρήστες να αναλύουν τα δεδομένα τους με ευκολία και αποτελεσματικότητα. Η επέκταση αυτών των δυνατοτήτων καθιστά το εργαλείο ιδιαίτερα χρήσιμο για την ανάλυση και την κατανόηση μεγάλων και πολύπλοκων συνόλων δεδομένων

Με αυτόν τον τρόπο, το εργαλείο έλυσε το πρόβλημα της χρονοβόρας και επίπονης χειροκίνητης ανάλυσης, παρέχοντας μια γρήγορη, ακριβή και εύχρηστη λύση για την αυτόματη παραγωγή στατιστικού προφίλ αλφαριθμητικών δεδομένων.

5.2 Μελλοντικές επεκτάσεις

Στο μέλλον, το σύστημα μπορεί να επεκταθεί με την προσθήκη ενός υποσυστήματος για την εύρεση πιθανών κλειδιών από τον πίνακα, όπως αναφέρεται στο Κεφάλαιο 2. Αυτό θα βοηθούσε στην καλύτερη κατανόηση της δομής των δεδομένων και στην εύρεση των κατάλληλων κλειδιών για τον πίνακα αλλά και πώς θα μπορούσαν να χρησιμοποιηθούν δύο διαφορετικά datasets μαζί.

Επιπλέον, το πακέτο Highlight χρειάζεται ανακατασκευή, καθώς είναι δύσκολο στη χρήση για τους προγραμματιστές. Αυτή η αναβάθμιση θα βελτιώσει την ποιότητα κώδικα του υποσυστήματος και θα διευκολύνει τη χρήση του.

Όσον αφορά τις αναφορές που παράγονται από το σύστημα, το πακέτο Reports θα μπορούσε να τροποποιηθεί για να προσφέρει καλύτερη οργάνωση και διαχείριση των δεδομένων, κάνοντάς το πιο καθαρό και ευανάγνωστο. Τροποποίηση τόσο στην ποιότητα του κώδικα αλλά και στις ίδιες τις αναφορές.

Για να επιταχυνθεί η διαδικασία, προτείνεται η εφαρμογή παραλληλίας στον υπολογισμό των αναλύσεων. Η παραλληλία θα βελτιώσει την απόδοση του συστήματος, επιτρέποντας ταχύτερη επεξεργασία μεγάλων δεδομένων. Οι διάφορες αναλύσεις τις Pythias είναι απομονωμένες χωρίς κυκλικές εξαρτήσεις (ανεξάρτητα αν κάποιες προαπαιτούν τον υπολογισμό άλλων) και είναι αρκετά εύκολο να παραλληλοποιηθούν και να επιταχύνουν την συνολική δημιουργία του προφίλ δεδομένων.

Τέλος, το πακέτο GUI που προστέθηκε θα μπορούσε να επεκταθεί για να προσφέρει καλύτερη εμπειρία στο χρήστη και αυξημένη λειτουργικότητα, καθιστώντας το περιβάλλον εργασίας πιο φιλικό και αποτελεσματικό.

Επιπλέον, μπορεί να χρησιμοποιηθεί framework για τον έλεγχο κυρίως των test που εξετάζουν την ορθή λειτουργία του γραφικού περιβάλλοντος, ώστε να είναι πιο ολοκληρωμένη η διαδικασία ελέγχου και χωρίς να χρειάζεται αλληλεπίδραση με χρήστη.

Βιβλιογραφία

[ABGN15] Abedjan, Z., Golab, L., & Naumann, F. (2015). Profiling relational data: a survey. The VLDB Journal, 24(4), 557–581. doi: https://doi.org/10.1007/s00778-015-0389-y

[AGNP18] Abedjan, Z., Golab, L., Naumann, F., & Papenbrock, T. (2018). Data Profiling. Synthesis Lectures on Data Management. Springer Cham. doi: 10.1007/978-3-031-01865-7

[AgSr94] Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules. In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), doi: https://dl.acm.org/doi/10.5555/645920.672836

[Apac21] Apache Spark. "RDD Programming Guide." Apache Spark Documentation, Version 3.5.1, Apache Software Foundation, URL: https://spark.apache.org/docs/3.5.1/rdd-programming-guide.html

[Apac21] Apache Spark. "SQL Programming Guide." Apache Spark Documentation, Latest Version, Apache Software Foundation, URL: https://spark.apache.org/docs/latest/sql-programming-guide.html.

[Apac21] Apache Spark. "GraphX Programming Guide." Apache Spark Documentation, Latest Version, Apache Software Foundation, URL: https://spark.apache.org/docs/latest/graphx-programming-guide.html.

[Apac21] Apache Spark. "Structured Streaming Programming Guide." Apache Spark Documentation, Latest Version, Apache Software Foundation, URL: https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html.

[ApMa] Apache Maven. "What is Maven?" Apache Maven Documentation, Apache Software Foundation, URL: https://maven.apache.org/what-is-maven.html

[Danc21] Milica Dancuk "What Is a Spark DataFrame?" phoenixNAP Knowledge Base, March 16, 2021, https://phoenixnap.com/kb/spark-dataframe.

[Das22] Tulika Das. "Oracle Database Java Developer's Guide, 19c." Document ID: E96468-02, Copyright © 1999, 2022, Oracle and/or its affiliates.

URL: https://docs.oracle.com/en/database/oracle/oracle-database/19/jjdev/Java-overview.html#GUID-17B81887-C338-4489-924D-FDDF2468DEA7.

[DWDL20] Damji, J. S., Wenig, B., Das, T., Lee, D., & Zaharia, M. (2020). Learning Spark: Lightning-Fast Data Analytics (Second edition). Sebastopol, CA: O'Reilly Media.

[FaNK17] Fatima, A., Nazir, N., & Khan, M. G. (2017). Data Cleaning In Data Warehouse: A Survey of Data Pre-processing Techniques and Tools. International Journal of Information Technology and Computer Science, 9(3), 50-61. DOI: https://doi.org/10.5815/ijitcs.2017.03.06.

[Fern09] Fernau, H.: Algorithms for learning regular expressions from positive data. Inf. Comput. 207(4), 521–541 (2009)

[HaKP11] Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques (3rd ed.). The Morgan Kaufmann Series in Data Management Systems. Elsevier. ISBN 0123814804, 9780123814807.

[HDSM05] Hauswirth, M., Diwan, A., Sweeney, P. F., & Mozer, M. C. (2005). Automating Vertical Profiling. In Proceedings of the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (pp. 281–296). https://doi.org/10.1145/1094811.1094834

[KuFi16] Kusumasari, T., & Fitria. (2016). Data profiling for data quality improvement with OpenRefine. In 2016 International Conference on Information Technology Systems and Innovation (ICITSI) (pp. 1-6). DOI: 10.1109/ICITSI.2016.7858197

[LiZh20] Liu, Z., & Zhang, A. (2020). Sampling for Big Data Profiling: A Survey. IEEE Access, 8, 72713-72726. DOI: 10.1109/ACCESS.2020.2988120

[LKRV08] Li, Y., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., Jagadish, H.V.: Regular expression learning for information extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 21–30 (2008)

[NYKK18] Nazari, Z., Yu, S.-M., Kang, D., & Kawachi, Y. (2018). Comparative Study of Outlier Detection Algorithms for Machine Learning. At 2018 2nd International Conference. DOI: https://doi.org/10.1145/3234804.3234817

[RAHE01] Raman, V., Hellerstein, J.M.: Potters wheel: an interactive data cleaning system. In: Proceedings of the International Conference on Very Large Databases (VLDB), pp. 381–390 (2001)

[RoNi88] Rodgers, J. L., & Nicewander, W. A. (1988). Thirteen Ways to Look at the Correlation Coefficient. The American Statistician, 42(1), 59-66. https://doi.org/10.1080/00031305.1988.10475524

[VoNa11] Tobias Vogel and Felix Naumann. Instance-based "one-to-some" assignment of similarity measures to attributes. In Proc. of the International Conference on Cooperative Information Systems (CoopIS), pages 412–420, 2011. DOI: 10.1007/978-3-642-25109-2_27 15

[Wors23] Summer Worsley "All About Git." DataCamp Blog, DataCamp, October 2023, URL: https://www.datacamp.com/blog/all-about-git.

[ZhCh13] Zhang, M., Chakrabarti, K.: InfoGather+: semantic matching and annotation of numeric and time-varying attributes in web tables. In: Proceedings of the International Conference on Management of Data (SIGMOD), pp. 145–156 (2013)

[ZHO+11] Zhang, M., Hadjieleftheriou, M., Ooi, B.C., Procopiuc, C.M., Srivastava, D.: Automatic discovery of attributes in relational databases. In: Proceedings of the International Conference on Management of Data (SIGMOD), pp. 109–120 (2011)