ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΕΡΕΥΝΗΣΗ ΤΗΣ ΣΥΣΧΕΤΙΣΗΣ ΜΕΤΑΞΥ ΤΕΧΝΙΚΟΥ ΧΡΕΟΥΣ ΚΑΙ ΤΗΣ ΕΥΠΑΘΕΙΑΣ ΜΟΝΑΔΩΝ ΛΟΓΙΣΜΙΚΟΥ

A STUDY ON THE CORRELATION BETWEEN THE TECHNICAL DEBT AND THE VULNERABILITY OF THE SOFTWARE MODULES

Διπλωματική εργασία από τον
Παπαδόπουλο Δημήτριο

Θεσσαλονίκη,
Φεβρουάριος 2023

Επιβλέπων Καθηγητής
Χατζηγεωργίου Αλέξανδρος

Περιεχόμενα

Περίληψη

Στη παρούσα διπλωματική εργασία, ο αναγνώστης θα μπορεί να αναλύσει τα αποτελέσματα μίας έρευνας η οποία πραγματοποιήθηκε με τη βοήθεια χρήσης των SDK4ED και του VM4SEC. Η συγκεκριμένη έρευνα, σκοπό είχε να μελετήσει τη συσχέτιση του τεχνικού χρέους ενός repository κατά αντιστοιχία με τα πιθανά προβλήματα ασφάλειάς του.

Έτσι, μετά την ανάλυση των εργαλείων, το πως χτίστηκαν, καθώς και το σε τι αποσκοπούν, θα μοιραστούν τα αποτελέσματα της προαναφερθείσας έρευνας. Ο αναγνώστης θα είναι σε θέση να γνωρίζει την λειτουργικότητα των δύο εργαλείων, πως μπορούν να τον βοηθήσουν στο να συγκεντρώσει αντίστοιχα αποτελέσματα αλλά και τέλος, θα μπορεί να διακρίνει κατά πόσο υπάρχει συσχέτιση μεταξύ τεχνικού χρέους και ασφάλειας σε ένα repository μέσω των μεθοδολογιών που ακολουθήθηκαν.

Τέλος, μετά τη συλλογή των δεδομένων και το φιλτράρισμα τους, έγινε στατιστική καθώς και ανάλυση αυτών, με σκοπό την εξακρίβωση ή μη, ύπαρξη συσχέτισης μεταξύ του τεχνικού χρέους και της ευπάθειας του κάθε repository. Οι μεθοδολογίες που χρησιμοποιήθηκαν ήταν οι Pearson, Spearman, και Permutation test.

Abstract

In this thesis, the reader will be able to analyze the results of a research that was conducted with the use of SDK4ED and VM4SEC. This research aimed to study the correlation of the technical debt of a repository in correspondence with its possible security problems.

Thus, after analyzing the tools, how they were built, and what they aim at, the results of the aforementioned research will be shared. At the end of the paper, the reader will be able to know the functionality of the two tools, how they can help him to gather corresponding results and finally, he will be able to distinguish whether there is a correlation between technical debt and security in a repository through the methodologies followed.

Finally, after collecting the data and filtering them, statistical as well as analysis was performed to verify, or not, the existence of a correlation between technical debt and vulnerability of each repository. The methodologies used were Pearson, Spearman, and Permutation test.

Λέξεις Κλειδιά: Τεχνικό Χρέος, Ασφάλεια, Ευπάθεια λογισμικού, SDK4ED, VM4SEC, Pearson, Spearman, Permutation test, Συσχέτιση

1. Εισαγωγή

1.1 Πρόβλημα

Το ερώτημα που καλείται να απαντήσει η παρούσα έρευνα είναι κατά πόσο υπάρχει συσχέτιση μεταξύ τεχνικού χρέους και ευπάθειας για μία αποθήκη λογισμικού. Ένα ερώτημα που είναι ευρέως γνωστό στην επιστημονική κοινότητα και ταλανίζει τον ακαδημαϊκό, αλλά και επιχειρηματικό κόσμο, καθώς σε περίπτωση συσχέτισης, κάθε πρόβλημα τεχνικού χρέους θα σήμαινε παράλληλο πρόβλημα και σε ζητήματα ασφάλειας.

Έτσι, το αποτέλεσμα της εργασίας θέλει να αποδείξει πως υπάρχει συσχέτιση, καθώς η ύπαρξη τεχνικού χρέους μπορεί να σημαίνει έμμεσα (ακόμα και άμεσα) πως υπάρχουν και παραβιάσεις ασφάλειας στα αντίστοιχα σημεία του λογισμικού.

1.2 Σκοπός

Ο σκοπός της εργασίας είναι να αναλύσει την χρήση των δύο εργαλείων, SDK4ED και VM4SEC, καθώς ο τρόπος εξαγωγής των αποτελεσμάτων παρουσιάζει πολύ μεγάλο ενδιαφέρον, συνδυαστικά με την ανάλυση των παραπάνω αποτελεσμάτων, με τη χρήση βιβλιοθηκών που βοηθούν στο τελικό στατιστικό αποτέλεσμα. Ο απώτερος σκοπός, είναι να υπάρχει συσχέτιση μεταξύ των δύο ανεξάρτητων μεταβλητών (τεχνικό χρέος και ευπάθεια αντίστοιχα).

1.3 Συνεισφορά

Κατά τη βιβλιογραφική επισκόπηση, έγινε μελέτη αντίστοιχων ερευνητικών εγγράφων καθώς και των εγχειριδίων των ίδιων των εργαλείων. Το δεύτερο σκέλος βοήθησε ώστε να γίνει αντιληπτός ο τρόπος ανάλυσης του τεχνικού χρέους και της ευπάθειας, κάτι που με τη σειρά του συνέδραμε στο φιλτράρισμα των αποτελεσμάτων κατά τη στατιστική ανάλυση.

1.4 Μεθοδολογία

Για να εξακριβωθεί εάν υπάρχει σχέση εξάρτησης μεταξύ του Τεχνικού Χρέους και της Ευπάθειας λογισμικού, η μεθοδολογία της έρευνας βασίστηκε στη χρήση τριών βασικών στατιστικών μεθοδολογιών. Αυτές είναι οι Pearson, Spearman και Permutation Test. Με τη χρήση της γλώσσας προγραμματισμού Python κατασκευάστηκαν τα κατάλληλα αρχεία κώδικα, τα οποία εκτέλεσαν τις αντίστοιχες στατιστικές αναλύσεις με σκοπό την εξόρυξη αποτελεσμάτων που θα αναφερθούν εντός της παρούσας εργασίας.

2. Βιβλιογραφική Ανασκόπηση

Τι είναι το Τεχνικό Χρέος

Το τεχνικό χρέος είναι ένας όρος που χρησιμοποιείται στην ανάπτυξη λογισμικού για να περιγράψει το κόστος συντήρησης και ενημέρωσης συστημάτων λογισμικού που έχουν αναπτυχθεί με συντομεύσεις, συμβιβασμούς ή προσωρινές λύσεις. Είναι το αποτέλεσμα της επιλογής μιας γρήγορης και εύκολης λύσης έναντι μιας πιο ολοκληρωμένης, μεθοδικής και επεκτάσιμης.

Ο όρος "τεχνικό χρέος" χρησιμοποιείται για να περιγράψει το μακροπρόθεσμο κόστος των κακών σχεδιαστικών αποφάσεων, της ελλιπούς ή ανεπαρκούς τεκμηρίωσης, του μη δοκιμασμένου κώδικα ή άλλων τεχνικών παραχωρήσεων που γίνονται κατά τη διαδικασία ανάπτυξης. Αυτές οι αποφάσεις μπορεί να κάνουν το λογισμικό πιο δύσκολο στη συντήρηση, πιο δύσκολο στην ενημέρωση και πιο επιρρεπές σε λάθη και σφάλματα.

Το χρέος μπορεί να συσσωρευτεί με την πάροδο του χρόνου και να οδηγήσει σε σημαντικό κόστος για μια εταιρεία ή έναν οργανισμό. Σε ακραίες περιπτώσεις, το τεχνικό χρέος μπορεί να καταστήσει δύσκολη ή ακόμη και αδύνατη την πραγματοποίηση σημαντικών αλλαγών σε ένα σύστημα λογισμικού, απαιτώντας την πλήρη επανεγγραφή του συστήματος από την αρχή. Ως εκ τούτου, είναι σημαντικό για τις ομάδες ανάπτυξης λογισμικού να γνωρίζουν το τεχνικό χρέος και να θέτουν προτεραιότητα στην αντιμετώπισή του, ώστε να ελαχιστοποιήσουν τις επιπτώσεις του με την πάροδο του χρόνου.

Το Τεχνικό Χρέος αποτελεί έναν επιστημονικό χώρο ο οποίος εάν αξιοποιηθεί ορθά από οργανισμούς και εταιρείες, μπορεί να εξοικονομήσει σημαντικούς πόρους και χρόνο. Είναι μία από τις μεγαλύτερες ανάγκες, ιδίως τα τελευταία χρόνια, καθώς η ταχύτητα εξέλιξης του κλάδου της πληροφορικής δημιουργεί την υποχρέωση για δημιουργία ορθότερης, ποιοτικότερης και πιο αποτελεσματικής δομής του λογισμικού. [9]

Ο Ward Cunningham, εκδότης του Agile Manifesto, χαρακτήρισε το τεχνικό χρέος ως εξής:

«*Η αποστολή ενός κώδικα για πρώτη φορά είναι σαν να χρεώνεσαι. Ένα μικρό χρέος επιταχύνει την ανάπτυξη, αρκεί να αποπληρωθεί άμεσα με μια επανεγγραφή. Ο κίνδυνος εμφανίζεται όταν το χρέος δεν εξοφλείται. Κάθε λεπτό που ξοδεύεται σε κώδικα που δεν είναι αρκετά σωστός μετράει ως τόκος του χρέους. Ολόκληροι οργανισμοί με εξειδίκευση στον τομέα του λογισμικού μπορούν να ακινητοποιηθούν κάτω από το φορτίο χρέους μιας μη παγιωμένης υλοποίησης, αντικειμενοστραφούς ή μη.*» [34]

Πώς μετράται το Τεχνικό Χρέος

Ο υπολογισμός του τεχνικού χρέους δεν είναι μια απλή διαδικασία, καθώς δεν υπάρχει μια ενιαία συμφωνημένη μέθοδος για την ποσοτικοποίησή του. Ωστόσο, υπάρχουν διάφορες προσεγγίσεις που μπορούν να χρησιμοποιηθούν για την εκτίμηση του τεχνικού χρέους. Ακολουθούν μερικές κοινώς χρησιμοποιούμενες μέθοδοι:

(1) Εργαλεία ανάλυσης κώδικα: Υπάρχουν διάφορα διαθέσιμα εργαλεία που μπορούν να αναλύσουν κώδικα και να εντοπίσουν πιθανό τεχνικό χρέος. Αυτά τα εργαλεία μπορούν να παρέχουν μετρήσεις όπως η πολυπλοκότητα του κώδικα, η επανάληψη κώδικα και η κάλυψη κώδικα, οι οποίες μπορούν να χρησιμοποιηθούν για τον εντοπισμό περιοχών του κώδικα που μπορεί να απαιτούν προσοχή.

(2) Εκτιμήσεις προγραμματιστών: Οι προγραμματιστές είναι συχνά στην καλύτερη θέση για να εκτιμήσουν την ποσότητα του τεχνικού χρέους σε μια βάση κώδικα. Μπορούν να εκτιμήσουν τον χρόνο που θα χρειαστεί για τη διόρθωση γνωστών ζητημάτων ή την υλοποίηση ελλείπουσας λειτουργικότητας, καθώς και τον αντίκτυπο αυτών των αλλαγών στη συνολική βάση κώδικα.

(3) Επιχειρηματικός αντίκτυπος: Μια άλλη προσέγγιση είναι η εκτίμηση του επιχειρηματικού αντίκτυπου του τεχνικού χρέους. Αυτό μπορεί να περιλαμβάνει παράγοντες όπως η απώλεια παραγωγικότητας, το αυξημένο κόστος υποστήριξης και η μειωμένη ικανοποίηση των πελατών.

(4) Εξωτερικά σημεία αναφοράς: Οι οργανισμοί μπορούν επίσης να συγκρίνουν την κωδικοποιημένη βάση τους με εξωτερικά σημεία αναφοράς, όπως τα πρότυπα του κλάδου ή τις βέλτιστες πρακτικές. Συγκρίνοντας τον κώδικά τους με αυτά τα σημεία αναφοράς, μπορούν να εντοπίσουν τις περιοχές στις οποίες ο κώδικάς τους μπορεί να υστερεί και να εκτιμήσουν το ποσό του τεχνικού χρέους που πρέπει να αντιμετωπιστεί.

Είναι σημαντικό να σημειωθεί ότι το τεχνικό χρέος δεν είναι μια σταθερή ή στατική τιμή, αλλά μάλλον ένα συνεχές κόστος που μπορεί να συσσωρευτεί με την πάροδο του χρόνου. Ως εκ τούτου, είναι σημαντικό να αξιολογείται και να αντιμετωπίζεται τακτικά το τεχνικό χρέος για να ελαχιστοποιείται ο αντίκτυπός του στα έργα ανάπτυξης λογισμικού. [9]

Στα παραπάνω αξίζει να προστεθεί και ο κλάδος της Μηχανικής Μάθησης. Σύμφωνα με [10], υπάρχουν αλγόριθμοι όπως αυτός του Scott-Knott, οι οποίοι μπορούν να εντοπίσουν τεχνικό χρέος σε κλάσεις. Με βάση αυτά τα κριτήρια ο αλγόριθμος γίνεται πιο έμπειρος με αποτέλεσμα να βρίσκει τεχνικό χρέος πιο εύκολα και με καλύτερο τρόπο συν τον χρόνο σε έργα λογισμικού. Επίσης, από το [11] αξίζει να σημειωθεί πως μόνο 4 εργαλεία υπάρχουν τα οποία είναι κατασκευασμένα αμιγώς για τη διαχείριση του τεχνικού χρέους, ενώ τα υπόλοιπα 25 της έρευνας δημιουργήθηκαν για άλλο σκοπό, απλώς καλύπτουν και αυτόν του τεχνικού χρέους. Αυτό δείχνει και την ανάγκη για περαιτέρω ανάπτυξη εργαλείων γύρω από αυτόν τον τεχνολογικό τομέα.

Τι είναι η Ευπάθεια Λογισμικού

Μια ευπάθεια λογισμικού είναι μια αδυναμία ή ένα ελάττωμα στο λογισμικό που μπορεί να αξιοποιηθεί από έναν εισβολέα για να εκτελέσει μη εξουσιοδοτημένες ενέργειες στο σύστημα. Οι ευπάθειες μπορούν να εμφανιστούν σε οποιοδήποτε στάδιο της διαδικασίας ανάπτυξης λογισμικού, συμπεριλαμβανομένου του σχεδιασμού, της κωδικοποίησης, των δοκιμών και της ανάπτυξης.

Οι ευπάθειες μπορούν να λάβουν πολλές μορφές, αλλά ορισμένοι συνηθισμένοι τύποι περιλαμβάνουν:

- Ευπάθειες υπερχείλισης ρυθμιστικού διαστήματος: Αυτά συμβαίνουν όταν ένα πρόγραμμα προσπαθεί να αποθηκεύσει περισσότερα δεδομένα σε ένα ρυθμιστή (buffer) από όσα μπορεί να διαχειριστεί. Αυτό μπορεί να προκαλέσει κατάρρευση του προγράμματος ή να επιτρέψει σε έναν εισβολέα να εκτελέσει αυθαίρετο κώδικα.

- Τρωτά σημεία εισόδου: Αυτά συμβαίνουν όταν ένας εισβολέας μπορεί να εισάγει κώδικα σε ένα σύστημα μέσω ενός πεδίου εισαγωγής χρήστη, όπως ένα πλαίσιο αναζήτησης ή μια φόρμα σύνδεσης. Αυτό μπορεί να επιτρέψει στον εισβολέα να εκτελέσει εντολές στο σύστημα ή να υποκλέψει δεδομένα.

- Ευπάθειες παράκαμψης αυθεντικοποίησης: Αυτές συμβαίνουν όταν ένα σύστημα δεν πιστοποιεί σωστά τους χρήστες πριν χορηγήσει πρόσβαση σε πόρους. Αυτό μπορεί να επιτρέψει σε έναν εισβολέα να παρακάμψει τα μέτρα ασφαλείας και να αποκτήσει μη εξουσιοδοτημένη πρόσβαση σε ευαίσθητα δεδομένα ή συστήματα.

- Ευπάθειες λογισμικού διασταυρούμενων ιστότοπων: Αυτά συμβαίνουν όταν ένας εισβολέας μπορεί να εισάγει κακόβουλα σενάρια σε έναν ιστότοπο, τα οποία στη συνέχεια εκτελούνται από τους χρήστες που επισκέπτονται τον ιστότοπο. Αυτό μπορεί να επιτρέψει στον εισβολέα να κλέψει δεδομένα του χρήστη ή να εκτελέσει ενέργειες εκ μέρους του χρήστη.

Οι ευπάθειες μπορεί να έχουν σοβαρές συνέπειες, όπως κλοπή δεδομένων, παραβίαση του συστήματος και διακοπή των υπηρεσιών. Για να μειωθεί ο κίνδυνος ευπαθειών, οι προγραμματιστές θα πρέπει να ακολουθούν ασφαλείς πρακτικές κωδικοποίησης και να διεξάγουν τακτικές δοκιμές ασφαλείας. Οι χρήστες θα πρέπει επίσης να ενημερώνουν το λογισμικό τους με τις πιο πρόσφατες διορθώσεις ασφαλείας και να ακολουθούν τις βέλτιστες πρακτικές για την ασφάλεια στο διαδίκτυο. [12]

Αντιμετώπιση της Ευπάθειας Λογισμικού

Υπάρχουν διάφοροι μηχανισμοί που μπορούν να χρησιμοποιηθούν για την άμυνα κατά των ευπαθειών λογισμικού, όπως:

(1) Ασφαλείς πρακτικές κωδικοποίησης: Οι προγραμματιστές μπορούν να ακολουθήσουν πρακτικές ασφαλούς κωδικοποίησης, όπως επικύρωση εισόδου, χειρισμό σφαλμάτων και διαχείριση μνήμης, για να μειώσουν τον κίνδυνο εισαγωγής ευπαθειών στον κώδικά τους.

(2) Ανασκοπήσεις και δοκιμές κώδικα: Οι προγραμματιστές μπορούν να χρησιμοποιούν αυτοματοποιημένα εργαλεία και χειροκίνητες αναθεωρήσεις κώδικα για να εντοπίζουν και να διορθώνουν ευπάθειες στον κώδικά τους πριν από την ανάπτυξή του. Αυτό μπορεί να περιλαμβάνει τεχνικές όπως η στατική ανάλυση, ο έλεγχος fuzz και ο έλεγχος διείσδυσης.

(3) Τακτικές ενημερώσεις λογισμικού: Οι πωλητές λογισμικού κυκλοφορούν τακτικά ενημερώσεις που περιλαμβάνουν διορθώσεις ασφαλείας για την αντιμετώπιση γνωστών ευπαθειών. Είναι σημαντικό να διατηρείτε το λογισμικό ενημερωμένο για να μειώσετε τον κίνδυνο εκμετάλλευσης.

(4) Ασφάλεια δικτύου: Τα μέτρα ασφάλειας δικτύου, όπως τα τείχη προστασίας, τα συστήματα ανίχνευσης και πρόληψης εισβολών (IDPS) και η τμηματοποίηση του δικτύου, μπορούν να βοηθήσουν στην αποτροπή της εκμετάλλευσης ευπαθειών των εφαρμογών λογισμικού από τους επιτιθέμενους.

(5) Εκπαίδευση και ευαισθητοποίηση των χρηστών: Η εκπαίδευση των χρηστών σχετικά με τους κινδύνους των ευπαθειών λογισμικού και τον τρόπο προστασίας τους μπορεί να συμβάλει στη μείωση της πιθανότητας επιτυχών επιθέσεων. Αυτό μπορεί να περιλαμβάνει τεχνικές όπως η εκπαίδευση ευαισθητοποίησης σε θέματα phishing και οι βέλτιστες πρακτικές υγιεινής των κωδικών πρόσβασης.

Με την εφαρμογή αυτών των μηχανισμών, οι οργανισμοί μπορούν να συμβάλλουν στη μείωση του κινδύνου των ευπαθειών λογισμικού και στην καλύτερη προστασία των συστημάτων και των δεδομένων τους από την εκμετάλλευση από επιτιθέμενους. [12]

2.1 Το VM4SEC

Σε αυτό το σημείο, θα γίνει ανάλυση του VM4SEC, μιας πλατφόρμας η οποία χτίστηκε ώστε να βοηθήσει τους διαχειριστές έργων και τους προγραμματιστές αυτών, να βελτιστοποιούν το επίπεδο ασφάλειας του λογισμικού τους, μέσω μιας συνεχούς παρακολούθησης των σχετικών έργων. Η συγκεκριμένη πλατφόρμα διαχωρίζει τις ιδιότητες της σε δύο υπο-εργαλεία για (1) *την παροχή ποσοτικών εκφράσεων του επιπέδου ασφάλειας του λογισμικού (Security Analysis)* και (2) *τον εντοπισμό πιθανών σημείων πρόσβασης ασφάλειας (Vulnerability Analysis)*. [7] [8]

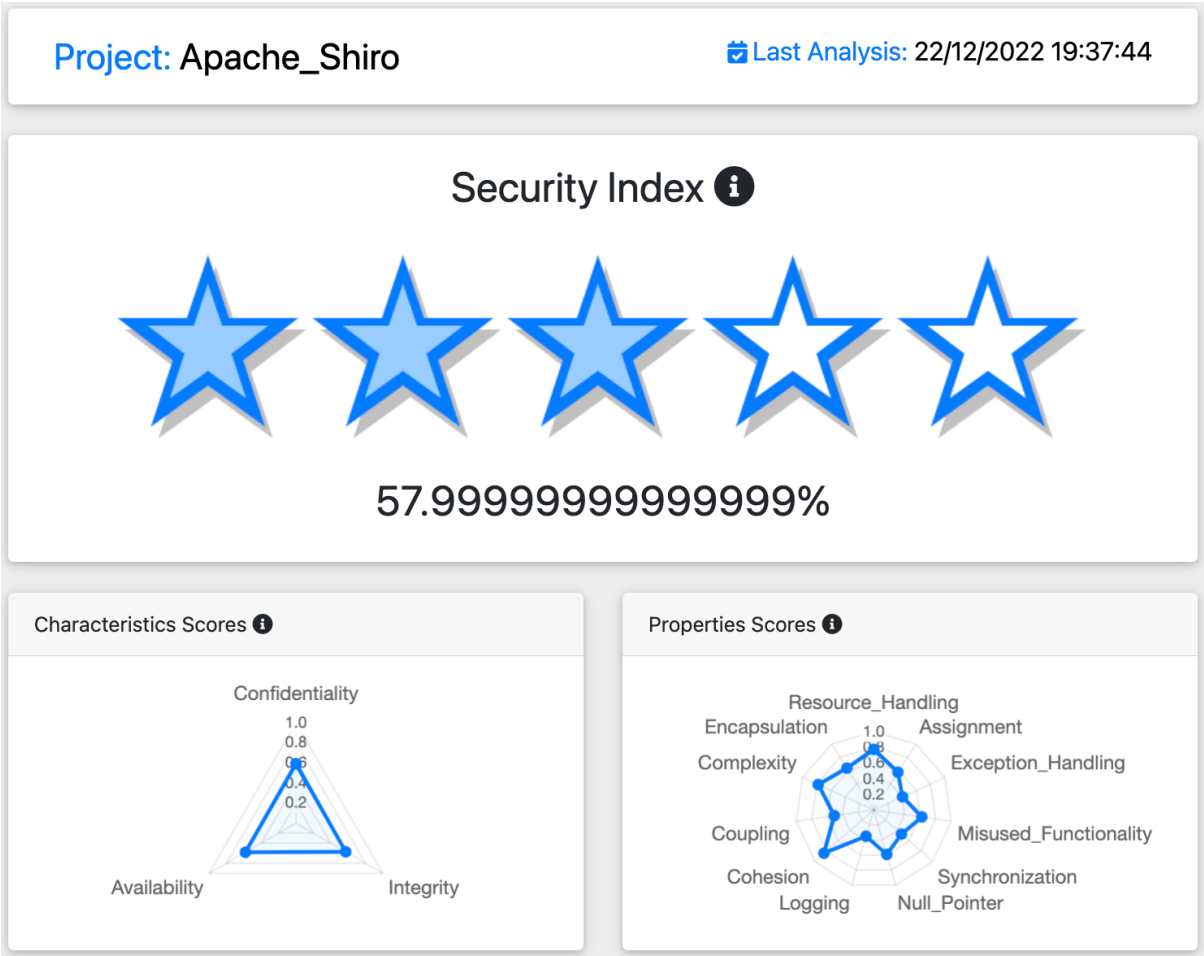
Πιο αναλυτικά, οι παραπάνω ιδιότητες:

Ποσοτική Αξιολόγηση Ασφάλειας (Quantitative Security Assessment – QSA): Ο συγκεκριμένος μηχανισμός αξιολογεί το επίπεδο εσωτερικής ασφάλειας ενός λογισμικού, με εστίαση στην ποσοτικοποίηση. Με τη χρήση στατικής ανάλυσης, εντοπίζει ζητήματα με πιθανές επιπτώσεις στην ασφάλεια του λογισμικού. Στη συνέχεια συγκεντρώνει τα αποτελέσματα της ανάλυσης χρησιμοποιώντας σύγχρονα μοντέλα προκειμένου να υπολογίσει μετρήσεις υψηλού επιπέδου που αντικατοπτρίζουν πολύ σημαντικές πτυχές ασφάλειας του εκάστοτε λογισμικού (όπως Εμπιστευτικότητα, Διαθεσιμότητας, Αξιοπιστία κλπ.). [6]

Πρόβλεψη ευπάθειας (Vulnerability Prediction – VP): Ο μηχανισμός αυτός επισημαίνει τα στοιχεία λογισμικού που είναι πιθανό να περιέχουν ευπάθειες, δηλαδή τα λεγόμενα hotspots που βρίσκονται στο λογισμικό. Χρησιμοποιεί μοντέλα μηχανικής μάθησης (ML) τα οποία λαμβάνουν ως είσοδο χαρακτηριστικά που εξάγονται από το εκάστοτε λογισμικό. Τα παραπάνω χαρακτηριστικά είτε εξάγονται μέσω εξόρυξης κειμένου είτε μέσω στατικής ανάλυσης. Μετά, τα μοντέλα αποφασίζουν εάν κάθε στοιχείο είναι πιθανό να περιέχει ευπάθεια. [2]

Οι παραπάνω μηχανισμοί, χρησιμοποιήθηκαν μέσω του front-end της πλατφόρμας, αλλά και μέσω των υποστηριζόμενων APIs (RESTfull) για πιο άμεση και γρήγορη χρήση των δεδομένων. Επίσης, οι μηχανισμοί αυτοί είναι αυτόνομα Microservices που μπορούν να χρησιμοποιηθούν μεμονωμένα παρέχοντας μεγαλύτερη ευκολία και ευχρηστία για τον τελικό χρήστη.

Το αποτέλεσμα της συγκεκριμένης υπηρεσίας είναι ένα JSON αρχείο. Το αρχείο αυτό, περιέχει την αναφορά αξιολόγησης ασφάλειας. Με τη σειρά της η αξιολόγηση περιλαμβάνει τον δείκτη ασφάλειας, τις βαθμολογίες ασφάλειας των ιδιοτήτων & χαρακτηριστικών του μοντέλου και τα λεπτομερή αποτελέσματα στατικής ανάλυσης. Αυτό το επίπεδο λεπτομέρειας, επιτρέπει τον τελικό χρήστη να φιλτράρει τα εκάστοτε JSON αρχεία του με τις προσωπικές του απαιτήσεις και παραμέτρους.

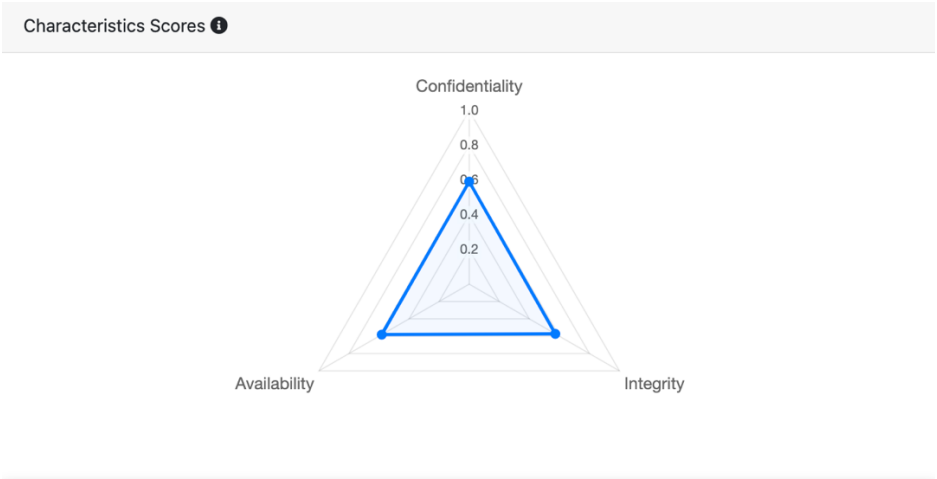


Εικόνα 1: VM4SEC Report Dashboard

Από την παραπάνω εικόνα, μπορούμε να παρατηρήσουμε τον σκελετό της αναφοράς αποτελεσμάτων, καθώς και τις τρεις κατηγορίες αναφορών του.

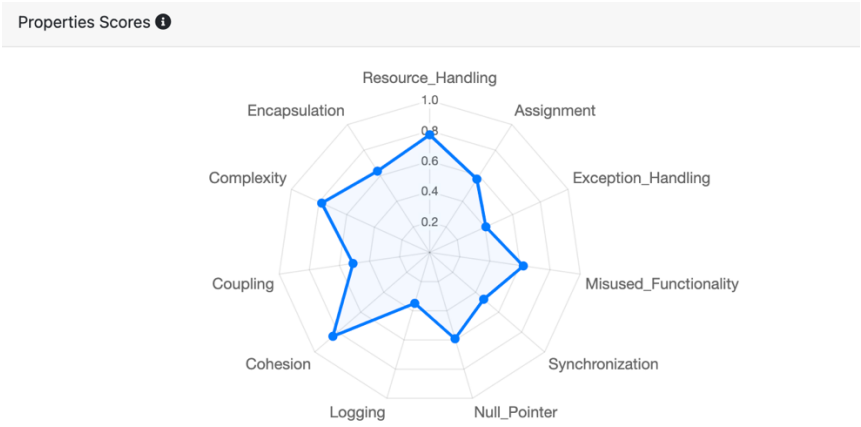
Αρχικά, στο πάνω σκέλος υπάρχει ο δείκτης Security Index ο οποίος μας δίνει μία γενική και καθολική βαθμολογία για την ασφάλεια της συγκεκριμένης αποθήκης λογισμικού. Πρακτικά είναι ο Δείκτης Ασφαλείας του έργου τόσο σε αριθμητική όσο και σε διακριτή μορφή. Αναφορικά με το τελευταίο, είναι μια σχετική βαθμολογία που καθορίζεται με βάση την κανονική κατανομή των βαθμολογιών ασφαλείας δημοφιλών έργων λογισμικού ανοιχτού κώδικα.

Στο αριστερό σκέλος, υπάρχει ένα γράφημα το οποίο ορίζει την βαθμολογία των χαρακτηριστικών του μοντέλου. Η συγκεκριμένη βαθμολογία βασίζεται στα εξής χαρακτηριστικά: Εμπιστευτικότητα, Διαθεσιμότητα και Ακεραιότητα.



Εικόνα 2: Βαθμολογία Χαρακτηριστικών

Τέλος, στο δεξιό σκέλος υπάρχει το γράφημα της βαθμολογίας των ιδιοτήτων ασφάλειας της εκάστοτε αποθήκης λογισμικού. Η συγκεκριμένη βαθμολογία πηγάζει από τις παρακάτω ιδιότητες: Πολυπλοκότητα, Σύζευξη, Συνεκτικότητα, Καταγραφή, Μηδενικός Δείκτης, Συγχρονισμός, Κακή χρήση Λειτουργικότητας, Διαχείριση Εξαιρέσεων, Εκχώρηση, Διαχείριση Πόρων και Ενθυλάκωση.



Εικόνα 3: Βαθμολογία ιδιοτήτων

2.2 Το μοντέλο του VM4SEC

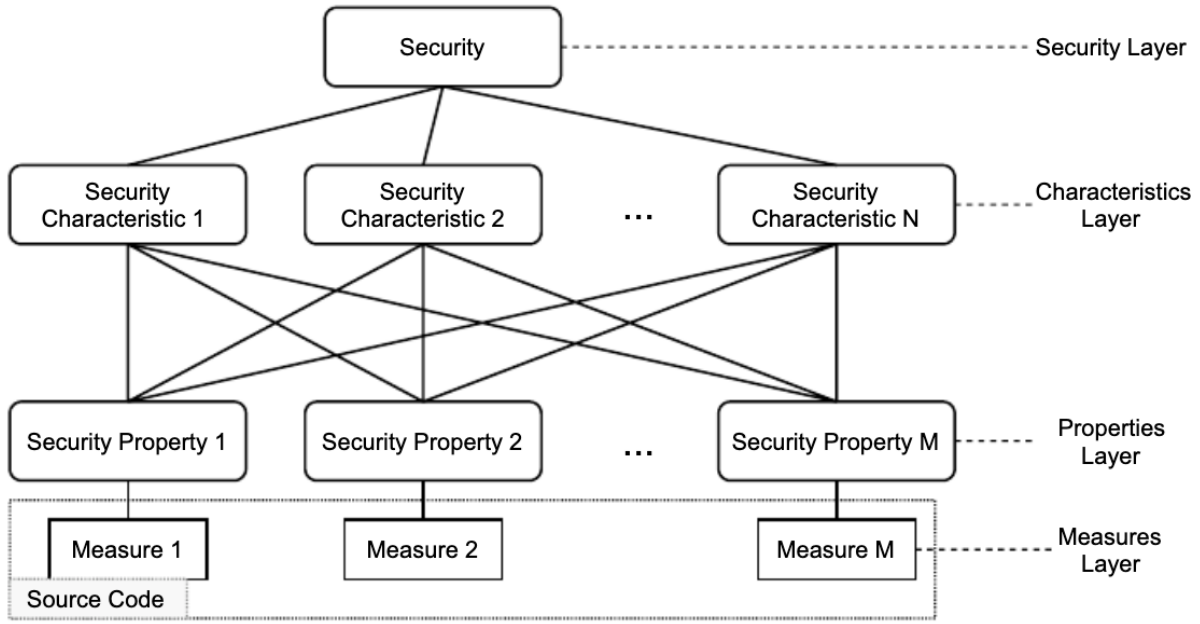
Το μοντέλο του εργαλείου είναι βασισμένο σε τέσσερα επίπεδα: (1) το επίπεδο των μετρήσεων, (2) το επίπεδο των ιδιοτήτων, (3) το επίπεδο των χαρακτηριστικών και τέλος (4) το επίπεδο της γενικής ασφάλειας.

Εν συντομία, το μοντέλο ξεκινά με τον υπολογισμό ενός συνόλου μέτρων ασφαλείας χαμηλού επιπέδου (στατική ανάλυση και μετρήσεις λογισμικού) από τον πηγαίο κώδικα και χρησιμοποιεί τις τιμές τους μαζί με ένα σύνολο ορίων, προκειμένου να εκχωρήσει αξιολογήσεις σε μία ομάδα ιδιοτήτων υψηλότερου επιπέδου (π.χ. Πολυπλοκότητα).

Στη συνέχεια, αυτές οι αξιολογήσεις συγκεντρώνονται χρησιμοποιώντας ένα σύστημα σταθμισμένου μέσου όρου προκειμένου να υπολογιστούν οι αξιολογήσεις ενός συνόλου χαρακτηριστικών ασφαλείας (π.χ. Εμπιστευτικότητα). Τέλος, υπολογίζεται ο μέσος όρος αυτών των αξιολογήσεων, προκειμένου να υπολογιστεί η συνολική βαθμολογία ασφαλείας του προϊόντος λογισμικού. Η βαθμολογία κυμαίνεται στο διάστημα [0.1].

Στην πραγματικότητα, το προτεινόμενο μοντέλο ασφαλείας συγκεντρώνει τους δείκτες ασφάλειας χαμηλού επιπέδου με έναν περίπλοκο τρόπο, προκειμένου να αποκτήσει μια ενιαία βαθμολογία ασφαλείας, δηλαδή τον δείκτη ασφαλείας, ο οποίος αντικατοπτρίζει το εσωτερικό επίπεδο ασφάλειας του προϊόντος. Η τελική βαθμολογία ασφαλείας παρέχει καλύτερη κατανόηση του επιπέδου ασφάλειας του προϊόντος στους προγραμματιστές και τους διαχειριστές έργων, σε σύγκριση με τα μεμονωμένα μέτρα χαμηλού επιπέδου.

Η παρακάτω εικόνα μπορεί να αποτυπώσει πλήρως τη δομή του μοντέλου:



Εικόνα 4: Διαγραμματικά η δομή του μοντέλου

Η συνολική δομή του προτεινόμενου μοντέλου είναι σύμφωνη με τις κατευθυντήριες γραμμές που παρέχονται από το διεθνές πρότυπο ISO/IEC 25010, για την κατασκευή μοντέλων ποιότητας. Σύμφωνα με το ISO/IEC 25010, η σύνθετη έννοια της ποιότητας λογισμικού θα πρέπει να αποσυντίθεται ιεραρχικά σε ένα σύνολο ποιοτικών χαρακτηριστικών (π.χ. Ασφάλεια), τα οποία μπορούν να αναλυθούν περαιτέρω σε ένα σύνολο υπο-χαρακτηριστικών ποιότητας (π.χ. Εμπιστευτικότητα και Ακεραιότητα). [3]

Αν και αυτά τα χαρακτηριστικά είναι πιο απτά σε σύγκριση με την πλήρη έννοια της ποιότητας, δεν μπορούν να μετρηθούν απευθείας από τον πηγαίο κώδικα ενός προϊόντος λογισμικού και επομένως θα πρέπει να ποσοτικοποιηθούν έμμεσα μέσω ενός συνόλου ιδιοτήτων χαμηλού επιπέδου που μπορούν να ποσοτικοποιηθούν άμεσα από τον πηγαίο κώδικα.

Τέλος, χρειάστηκαν τρία χαρακτηριστικά για να θεσπίσουν το συγκεκριμένο μοντέλο ασφάλειας. Τα χαρακτηριστικά αυτά είναι τα παρακάτω (ISO, 2011):

- **Εμπιστευτικότητα**: Ο βαθμός στον οποίο το προϊόν λογισμικού διασφαλίζει ότι τα δεδομένα είναι προσβάσιμα μόνο σε εκείνους που είναι εξουσιοδοτημένοι να έχουν πρόσβαση.
- **Ακεραιότητα**: Ο βαθμός στον οποίο ένα προϊόν λογισμικού (σύστημα ή στοιχείο) αποτρέπει τη μη εξουσιοδοτημένη τροποποίηση ευαίσθητων δεδομένων.
- **Διαθεσιμότητα**: Ο βαθμός στον οποίο ένα προϊόν λογισμικού (σύστημα ή στοιχείο) είναι λειτουργικό και προσβάσιμο όταν απαιτείται για χρήση.

2.3 Το SDK4ED

Το SDK4ED χτίστηκε πάνω σε ένα όραμα που ως σκοπό είχε την ελαχιστοποίησή του κόστους, τον χρόνο ανάπτυξης και την πολυπλοκότητα των διαδικασιών ανάπτυξης λογισμικού χαμηλής ενέργειας, παρέχοντας εργαλεία για αυτόματη βελτιστοποίηση πολλαπλών απαιτήσεων ποιότητας, όπως το τεχνικό χρέος, την ενεργειακή απόδοση, την αξιοπιστία και την απόδοση. Τα αντίστοιχα εργαλεία με τη σειρά τους θα παρέχουν αναφορές με ελλείψεις, ταξινομημένες με βάση τη σημασία και τον επείγοντα χαρακτήρα που πρέπει να επιλυθούν, λαμβάνοντας υπόψιν παρελθοντικές αλλαγές και την μελλοντική τους πιθανή συντήρηση.

Το εργαλείο, θα εκτιμήσει μέσω καινοτόμων τεχνικών, το κόστους και τους περιορισμούς που συνδέονται με τις υποχρεώσεις του τεχνικού χρέους που με τη σειρά τους σχετίζονται με την ενέργεια που απαιτείται για τον κύκλο ζωής ανάπτυξης ενός λογισμικού έργου.

Στην παρούσα εργασία, από το σύνολο των εργαλείων της πλατφόρμας, έγινε χρήση του υπο-εργαλείου που σχετίζεται με το τεχνικό χρέος (Technical Debt Management). Το συγκεκριμένο υπο-εργαλείο, είναι υπεύθυνο για την παρακολούθηση και τη βελτιστοποίηση της Συντηρησιμότητας (Maintainability) των προϊόντων λογισμικού μέσω της έννοιας του Τεχνικού Χρέους. Συγκεκριμένα, παρέχει νέους μηχανισμούς και υπηρεσίες για την εκτίμηση του Βασικού Τεχνικού Χρέους (Principal) και του Χ, καθώς και για την εξαγωγή ευκαιριών ανακατασκευής (refactoring). [5]

2.4 Το μοντέλο του SDK4ED – Technical Debt Management

Το συγκεκριμένο εργαλείο χρησιμοποιεί πολλές υπο-υπηρεσίες, οι οποίες ενώθηκαν κάτω από ένα Docker container. Οι συγκεκριμένες υπο-υπηρεσίες είναι οι παρακάτω:

- **TD Analysis**: Η συγκεκριμένη υπηρεσία είναι υπεύθυνη για να παρακολουθεί και να ελέγχει το τεχνικό χρέος σε επίπεδο αρχείου, πακέτου και έργου από ένα δοσμένο έργο λογισμικού. Πρακτικά είναι ένα εργαλείο που παρέχει ανάλυση εκδόσεων του έργου, ελέγχοντας το πως εξελίσσεται από τη μία έκδοση του στην άλλη. Επιπρόσθετα, η υπηρεσία περιλαμβάνει ανακατασκευές του σχεδίου του έργου. Οι ανακατασκευές σχεδίου περιλαμβάνουν την ανακατασκευή τύπου Long Method αλλά και Move Class. Και οι δύο καλούνται να βελτιώσουν την αναγνωσιμότητα και την δυνατότητα συντήρησης του λογισμικού.
- **TD New Code**: Η υπηρεσία αυτή είναι υπεύθυνη για την πρόληψη τεχνικού χρέους σχετικά με μία νέα παράδοση κώδικα και έργου γενικότερα, λειτουργώντας ως πύλη ποιότητας και ασφάλειας.

Οι υπηρεσίες έχουν υλοποιηθεί ως μεμονωμένες Docker εικόνες που αναπτύσσονται μέσω μιας μεμονωμένης Docker σύνθεσης. Η προαναφερθείσα σύνθεση είναι ένα εργαλείο που βοηθάει στον καθορισμό και την εκτέλεση εφαρμογών Docker πολλαπλών containers. Επιπρόσθετα δύο βάσεις δεδομένων έχουν υλοποιηθεί για την αποθήκευση των αποτελεσμάτων των δύο υπηρεσιών που παρέχει το Technical Debt Toolbox. Παρόλο που οι βάσεις δεδομένων έχουν αναπτυχθεί ως αυτόνομα Docket containers, η πρόσβαση σε αυτές τις βάσεις δεδομένων είναι εφικτή μόνο μέσω αποκλειστικών API που παρέχονται από το back-end Technical Debt Toolbox. [33]

2.5 Ο TD Classifier

Στο παραπάνω εργαλείο, προστέθηκε και η λειτουργία του **TD Classifier**. Είναι μία επιπρόσθετη λειτουργία που δεν υπάρχει ως αναφορά στο επίσημο αποθετήριο λογισμικού του συγκεκριμένου εργαλείου. Οι τεχνικές αναγνώρισης του TX που υιοθετούνται από τα περισσότερα από τα υπάρχοντα εργαλεία βασίζονται σε προκαθορισμένους κανόνες που μπορούν να επιβεβαιωθούν με τεχνικές στατιστικής ανάλυσης πηγαίου κώδικα. [4]

Ωστόσο το γεγονός ότι κάθε εργαλείο χρησιμοποιεί τα δικά του σύνολα κανόνων για τον εντοπισμό προβλημάτων τεχνικού χρέους οδηγεί σε σημαντικές ελλείψεις που επηρεάζουν τόσο τον ακαδημαϊκό όσο και τον πρακτικό τομέα. Όσον αφορά την ακαδημαϊκή κοινότητα, η έλλειψη ενός εργαλείου που λειτουργεί ως βασική αλήθεια οδηγεί σε προβλήματα εγκυρότητας της κατασκευής σε εμπειρικές μελέτες. Από την άλλη, οι επαγγελματίες είναι πάντα επιφυλακτικοί σχετικά με το ποιο εργαλείο να εμπιστευτούν για τον αποτελεσματικό εντοπισμού του τεχνικού χρέους.

Λαμβάνοντας υπόψη τα παραπάνω, έγινε αξιολόγηση εμπειρικών στατιστικών αλγορίθμων και αλγορίθμων μηχανικής μάθησης ως προς την ικανότητα τους να ταξινομούν κατηγορίες λογισμικού ως υψηλού/μη υψηλού τεχνικού χρέους. Ως βασική αλήθεια για την ανάπτυξη του πλαισίου ταξινόμησης, θεωρήθηκε μια «κοινά συμφωνημένη βάση γνώσης τεχνικού χρέους», δηλαδή ένα εμπειρικό σημείο αναφοράς των κλάσεων που παρουσιάζουν υψηλά επίπεδα τεχνικού χρέους. Με βάση τη σύγκλιση τριών ευρέως αποδεκτών εργαλείων αξιολόγησης τεχνικού χρέους: 1) SonarQube, 2) CAST & 3) Squore.

Έτσι, με βάση την παραπάνω ανάλυση αλλά και το πόρισμα ερευνητικής εργασίας που έγινε [12], επιχειρήθηκε να κατασκευαστεί ένα νέο εργαλείο, το TD Classifier, ένα εργαλείο που χρησιμοποιείται για την ταξινόμηση κλάσεων λογισμικού ως Υψηλό/Μη Υψηλό τεχνικό χρέος για οποιοδήποτε έργο Java, απλά δείχνοντας το git αποθετήριο του. Το εργαλείο περιλαμβάνει τη συλλογική γνώση που θα εξάγονταν από το συνδυασμό των αποτελεσμάτων των τριών προαναφερθέντων εργαλείων αξιολόγησης τεχνικού χρέους και βασίζεται σε τέσσερα εργαλεία ανοιχτού κώδικα για την αυτόματη ανάκτηση όλων των ανεξάρτητων μεταβλητών και την απόδοση των αναγνωρισμένων κλάσεων με υψηλό τεχνικό χρέος.

Κατά αυτόν τον τρόπο, επιτρέπει τον εύκολο εντοπισμό και τον περαιτέρω πειραματισμό των ζητημάτων τεχνικού χρέους, χωρίς να χρειάζεται να καταφύγει κανείς σε πλήθος εμπορικών εργαλείων και εργαλείων ανοιχτού κώδικα. Ο TD Classifier υλοποιείται ως εφαρμογή ιστού, που περιλαμβάνει τόσο ένα backend όσο και το σχετικό frontend. Προσφέρει διαδραστικές απεικονίσεις που επιτρέπουν τον άμεσο εντοπισμό των κλάσεων που είναι πιθανότερο να είναι προβληματικές.

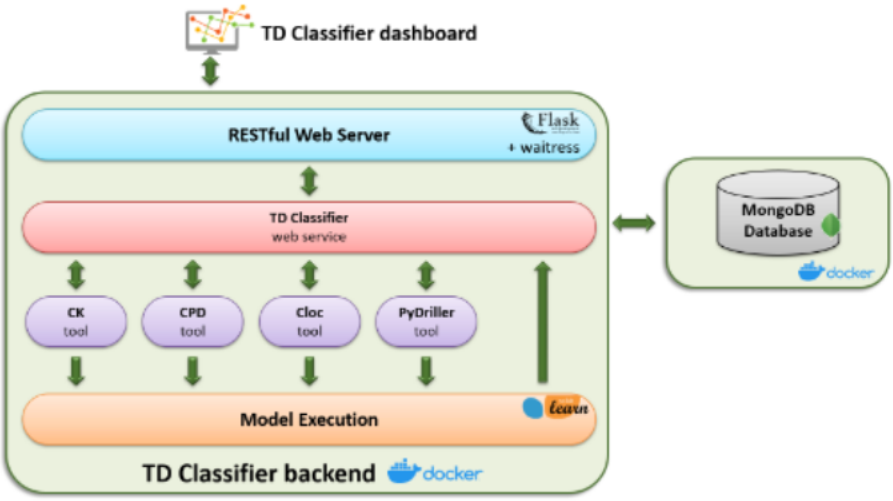
2.6 Η Μεθοδολογία του TD Classifier

Αρχικά, το πρώτο στάδιο ήταν αυτό της συλλογής δεδομένων, που έγινε μέσω της παρακάτω μελέτης. Συγκεκριμένα στη μελέτη έγινε εξέταση της ικανότητας αξιολόγησης Τεχνικού Χρέους τριών εργαλείων (SonarQube, CAST & Squore όπως αναφέρθηκε και παραπάνω) σε 25 έργα ανοιχτού κώδικα (Java), με σκοπό να αξιολογηθεί ο βαθμός συμφωνίας μεταξύ τους και να εντοπίσουν προφίλ κλάσεων/αρχείων που μοιράζονται παρόμοια επίπεδα τεχνικού χρέους μέσω της ανάλυσης αρχέτυπων. [4]

Αξιοποιώντας αυτό το εμπειρικό σημείο αναφοράς, χαρακτηρίστηκαν οι κλάσεις λογισμικού που ανήκουν στο προφίλ υψηλού επιπέδου τεχνικού χρέους ως «Υψηλό ΤΧ», ενώ οι υπόλοιπες κλάσεις χαρακτηρίστηκαν ως «όχι Υψηλού ΤΧ», καθιερώνοντας με αυτόν τον τρόπο τη «βασική αλήθεια» για το έργο της ταξινόμησης. Να σημειωθεί πως το σύνολο των δεδομένων περιείχε πάνω από 18,000 κλάσεις με τις 1,200 περίπου, να χαρακτηρίζονται ως κλάσεις υψηλού τεχνικού χρέους.

Οι τεχνολογίες που χρησιμοποιήθηκαν για την υπηρεσία του TD Classifier χωρίζονται σε BE και FE. Για το BE, χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python. Χτίστηκε με τέτοιο τρόπο ώστε να είναι προσβάσιμη ως υπηρεσία από τον οποιοδήποτε μέσω ενός API port. Με τον τρόπο αυτό, η υπηρεσία μπορεί να επικοινωνήσει με τρίτες υπηρεσίες και πλατφόρμες.

Επιπρόσθετα, όπως φαίνεται και στην παρακάτω εικόνα, το σημείο εισόδου του TD Classifier είναι ένας RESTful διακομιστής που χρησιμοποιεί Flask, το οποίο είναι ενσωματωμένο μέσα στο Waitress8, ένα WSGI της Python (διακομιστής prod-ready). Σε χαμηλότερο επίπεδο, ο διακομιστής εκθέτει το API του TD Classifier, το οποίο υλοποιείται ως μεμονωμένη υπηρεσία ιστού. Αυτή η υπηρεσία παίζει τον ρόλο ενός ενορχηστρωτή που είναι υπεύθυνος για (α) την κλωνοποίηση ενός έργου, (β) για την κλήση των εργαλείων ανάλυσης όπως PyDriller, (γ) την εκτέλεση του προ-εκπαιδευμένου ταξινομητή και (δ) την επιστροφή των αποτελεσμάτων.



Για τη διευκόλυνση της διαδικασίας κατασκευής και ανάπτυξης, το εργαλείο του backend έχει υλοποιηθεί ως μεμονωμένη Docker εικόνα και έχει αναπτυχθεί ως μεμονωμένο Docker Container. Για τον σκοπό αυτό, έχει δημιουργηθεί ένα Docker αρχείο το οποίο είναι διαθέσιμο στο διαδίκτυο στο αποθετήριο του εργαλείου. Κατά αυτόν τον τρόπο, οι δυνητικοί χρήστες μπορούν να δημιουργήσουν το δικό τους TD Classifier container (backend) εύκολα, δημιουργώντας τη δική τους εικόνα και να το φιλοξενήσουν τοπικά.

Επιπρόσθετα από τα σενάρια του TD Classifier (backend), όλα τα εργαλεία ανάλυσης τρίτων κατασκευαστών που είναι υπεύθυνα για τη συλλογή των απαιτούμενων εισροών του μοντέλου ενσωματώνονται επίσης στην Docker εικόνα ως αυτόνομα εκτελέσιμα αρχεία (σε μορφή jar ή με μορφή σεναρίου). Αυτή η ρύθμιση όχι μόνο ενισχύει τη φορητότητα καθιστώντας το εργαλείο εύκολο στην εγκατάσταση, αλλά επιταχύνει επίσης την εκτέλεση καθώς δεν απαιτούνται εξωτερικές κλήσεις για την εκτέλεσή τους.

Αξίζει να σημειωθεί ότι τα εργαλεία ανάλυσης εκτελούνται παράλληλα, προκειμένου να μειωθεί ο συνολικός χρόνος εκτέλεσης του εργαλείου. Τέλος, το εργαλείο χρησιμοποιεί μία βάση δεδομένων MongoDB η οποία έχεις ως σκοπό την αποθήκευση δεδομένων με τέτοιον τρόπο, που επιτρέπει στην υπηρεσία TD Classifier να ανακτά γρήγορα τα προηγούμενα αποτελέσματα κατόπιν αιτήματος, χωρίς να χρειάζεται διαδικασία επανεκτέλεσης της ανάλυσης.

Επίσης εκτός από το backend του εργαλείου, ένα διαισθητικό frontend έχει υλοποιηθεί προκειμένου να διευκολυνθεί η υιοθέτηση του στην πράξη. Η ανάπτυξη έχει γίνει με τη χρήση του React9 πλαισίου, και επικοινωνεί αδιάκοπα με το backend. Αυτό επιτρέπει την εύκολη κλήση των κύριων λειτουργιών που παρέχει το εργαλείο, καθώς και την οπτικοποίηση των παραγόμενων αποτελεσμάτων. [4]

3. Συλλογή και Ανάλυση δεδομένων

3.1 Επιλογή των κατάλληλων αποθετηρίων

Για να γίνει η κατάλληλη ερευνητική ανάλυση, συλλέχθηκαν και τα αντίστοιχα δεδομένα από ανοιχτά αποθετήρια. Πιο συγκεκριμένα χρησιμοποιήθηκε κώδικας από συνολικά 30 αποθετήρια. Η επιλογής τους έγινε με βάση τη δημοσιότητά τους και την πρακτική τους χρήση από την κοινότητα.

Οι κατηγορίες των αποθετηρίων είναι οι παρακάτω:

- Quality Assurance
- Auth & Security
- Mobile
- Other (CI/CD, Mobile, Crypto, etc.)

Οι κατηγορίες όπως Quality Assurance, Auth & Security and Mobile επιλέχθηκαν καθώς έχουν υψηλές απαιτήσεις όσον αφορά την ποιότητα των υπηρεσιών και την ασφάλεια αυτών αλλά και των χρηστών τους. Επομένως, θεωρητικά, τα συγκεκριμένα αποθετήρια θα παρέχουν ένα πολύ ρεαλιστικό πλαίσιο για έρευνα, και κατά πόσο υπάρχει συσχέτιση μεταξύ ασφάλειας και τεχνικού χρέους.

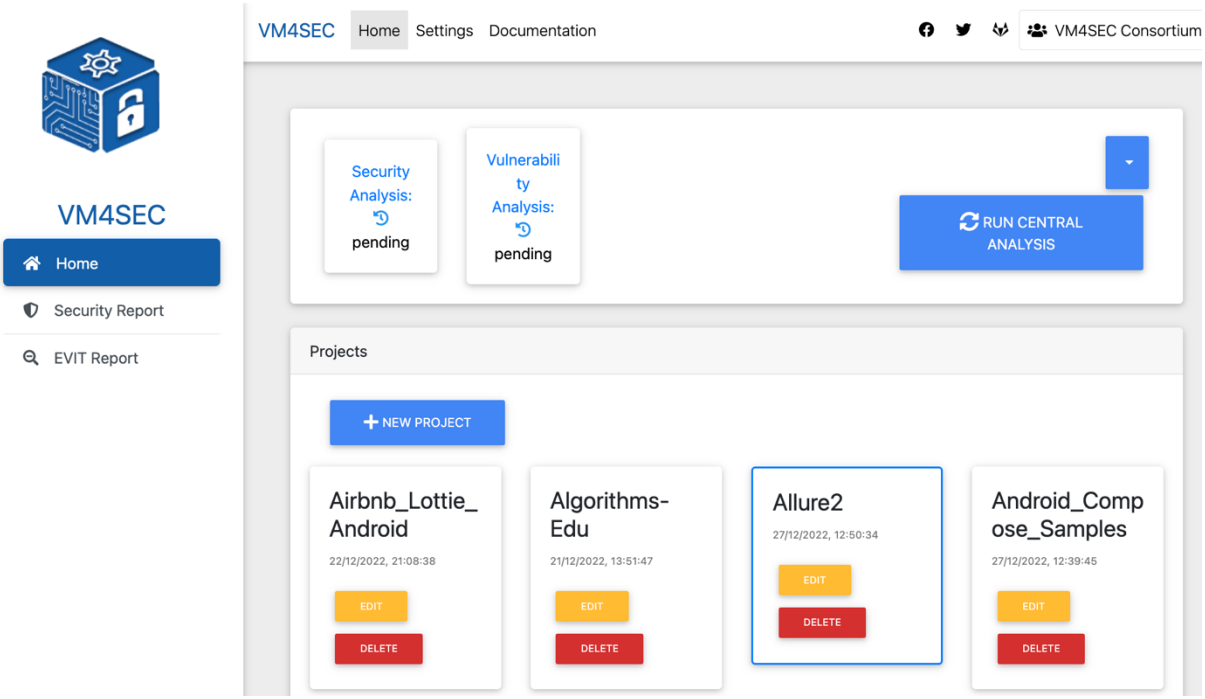
Όσον αφορά την κατηγορία Other, έχουν επιλεχθεί αποθετήρια τα οποία μπορεί να ανήκουν στις προηγούμενες κατηγορίες ή όχι, αλλά έχουν πολύ δυνατή απήχηση στον δικό τους τομέας, όπως το Jenkins, Signal κ.α.

3.2 Συλλογή δεδομένων από το VM4SEC

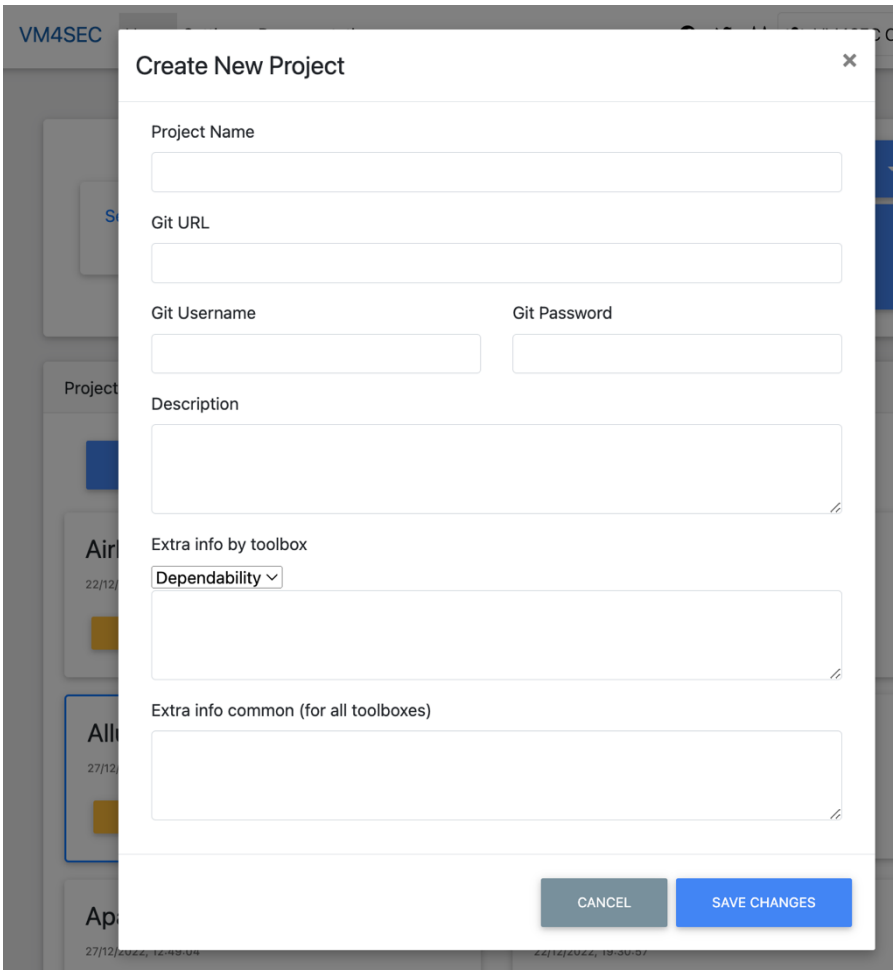
Για την συλλογή των δεδομένων μέσω της πλατφόρμας VM4SEC, χρειάζονται δύο ενέργειες από τη μεριά του χρήστη, οι οποίες είναι οι παρακάτω:

1) Χρήση του UI για την εισαγωγή του επιθυμητού project και εκτέλεση της ανάλυσης (είτε Security, είτε Vulnerability, είτε και τα δύο)
2) Χρήση των κατάλληλων API endpoints με σκοπό την αίτηση των αποτελεσμάτων από τις βάσεις δεδομένων του εργαλείου

Όσον αφορά την πρώτη ενέργεια, ο χρήστης μόλις βρεθεί στην αρχική σελίδα της πλατφόρμας θα πρέπει να εισάγει το αποθετήριο που επιθυμεί, πατώντας στο κουμπί «+New Project» όπως φαίνεται στις παρακάτω εικόνες:



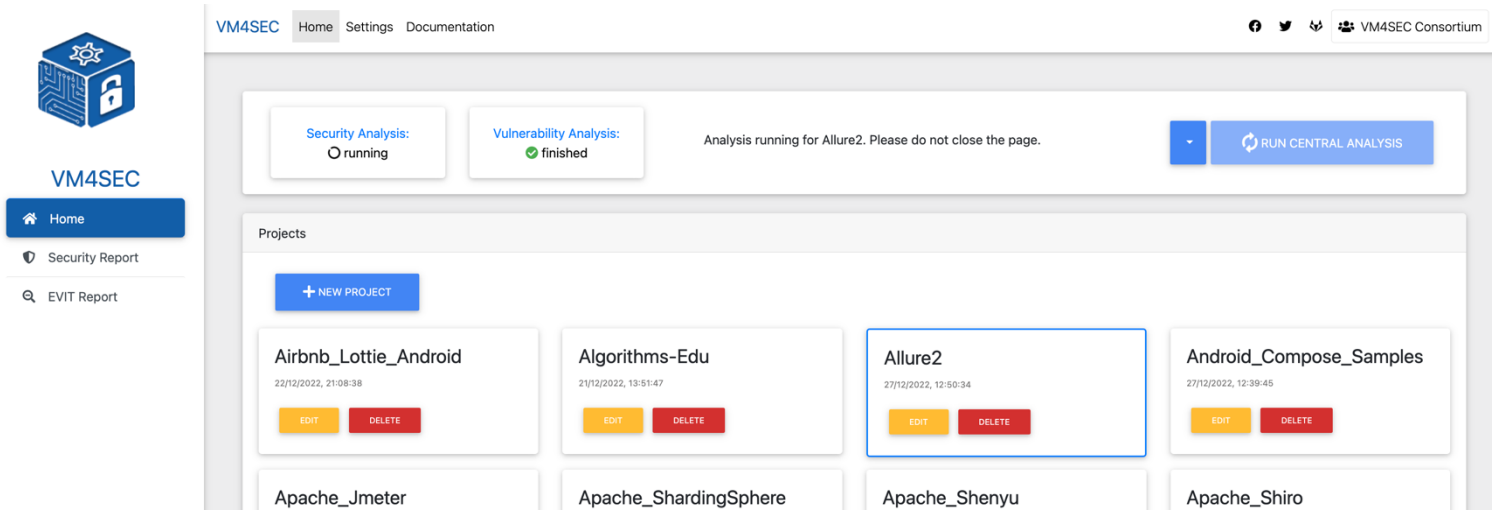
Αφού ο χρήστης πατήσει στο συγκεκριμένο κουμπί, θα εμφανιστεί η φόρμα εισαγωγής ενός αποθετηρίου στην πλατφόρμα. Μόνο όταν θα δημιουργηθεί είναι δυνατή η χρήση των λειτουργιών της πλατφόρμας για το συγκεκριμένο αποθετήριο. Έτσι, αφού ο χρήστης πατήσει το κουμπί που αναφέρθηκε παραπάνω, θα εμφανιστεί η παρακάτω φόρμα:



Στην παραπάνω φόρμα ο χρήστης θα πρέπει να εισάγει το όνομα που επιθυμεί για το συγκεκριμένο πρότζεκτ, τον σύνδεσμο από το αποθετήριο και τέλος στο πεδίο «Extra info common (for all toolboxes)» την παρακάτω απάντηση «{“language”:”java”}». Αυτό θα βοηθήσει την αντίστοιχη λειτουργία της πλατφόρμας να καταλάβει αν είναι πρότζεκτ Java ή όχι. Όπως αναφέρθηκε και στην αρχή, η πλατφόρμα δεν μπορεί να υποστηρίξει αποθετήρια που δεν είναι γραμμένα με τη γλώσσα προγραμματισμού Java. Εφόσον ο χρήστης συμπληρώσει αυτά τα πεδία που αποτελούν υποχρεωτική προϋπόθεση για τη δημιουργία του πρότζεκτ, τότε το συγκεκριμένο πρότζεκτ θα εμφανίζεται στην αρχική σελίδα.

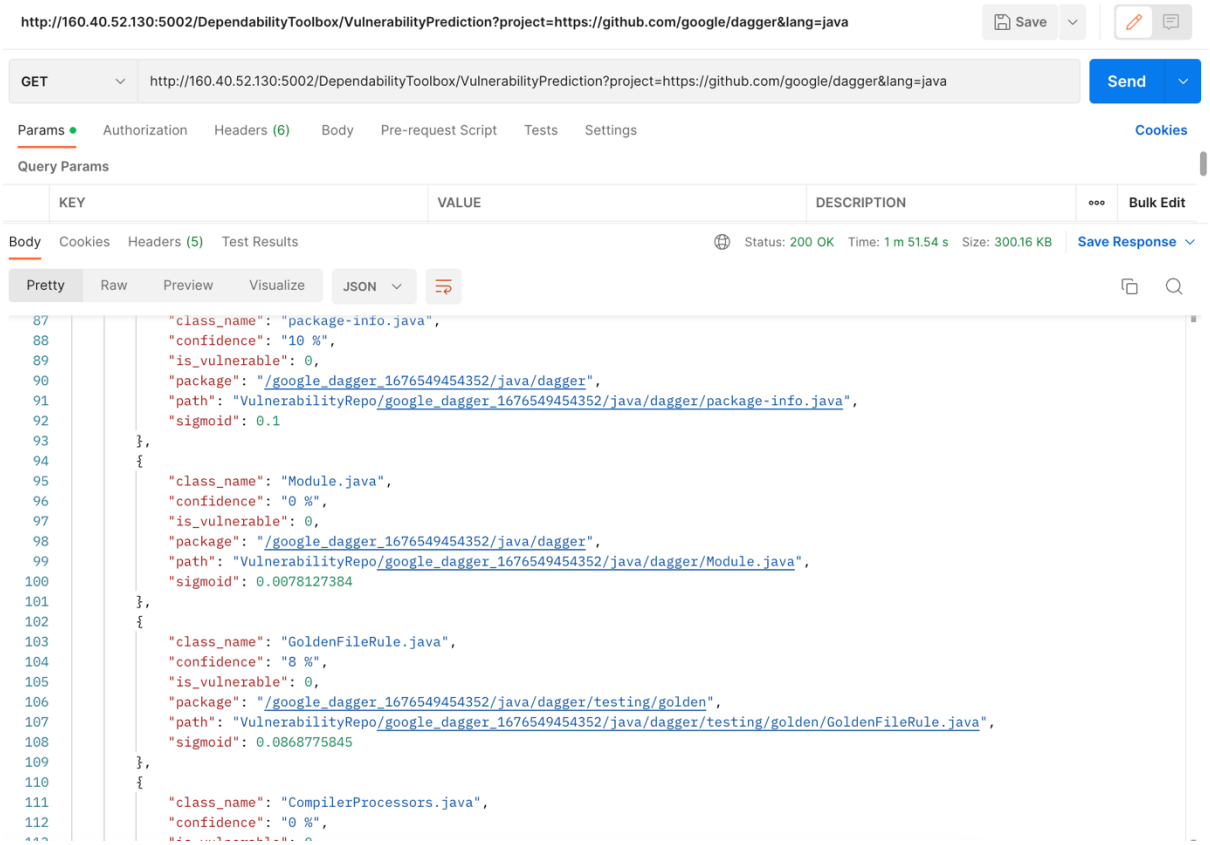
Στη συνέχεια, εφόσον ο χρήστης μπορεί να χρησιμοποιήσει το πρότζεκτ που μόλις προσέθεσε, πλέον θα μπορεί να επιλέξει το μενού πάνω δεξιά (αριστερά από το κουμπί «Run Central Analysis») εάν επιθυμεί ανάλυση σχετικά με το Security ή το Vulnerability (είτε και τα δύο).

Αφού ορίσει την επιλογή του, μπορεί επιλέγοντας το κουμπί Run Central Analysis να ξεκινήσει η αντίστοιχη εκτέλεση για το επιλεγμένο πρότζεκτ. Μόλις το κάνει, θα εμφανιστεί, σύμφωνα με την παρακάτω εικόνα, η τρέχουσα κατάσταση της εκτέλεσης:



Όπως είναι εμφανές, όταν μία από τις δύο λειτουργίες τελειώσει, τότε θα είναι σε κατάσταση “Finished” ενώ όσο εκτελείται θα είναι σε κατάσταση “Running”. Μόλις τελειώσουν και οι δύο λειτουργίες, τότε τα αποτελέσματα θα έχουν σωθεί στην αντίστοιχη βάση δεδομένων. Με την κατάλληλη χρήση των API endpoints ο χρήστης θα μπορεί να τα κατεβάσει τοπικά προς χρήση.

Με τη βοήθεια του Postman και έχοντας ένα API endpoint query (*http://160.40.52.130:5002/DependabilityToolbox/VulnerabilityPrediction?project=https://github.com/google/dagger&lang=java*) μπορούμε να ζητήσουμε τα Security & Vulnerability αποτελέσματα του Google/Dagger αποθετηρίου, για παράδειγμα, σε μορφή JSON όπως φαίνεται παρακάτω:



Μέσω των αποτελεσμάτων γίνεται κατανοητή η μορφή του JSON. Αυτό που χρειάζεται να εξαχθεί από αρχείο είναι το όνομα της κλάσης και η τιμή της μεταβλητής “sigmoid” η οποία αντικατοπτρίζει την πιθανότητα να υπάρχει Security & Vulnerability πρόβλημα στην αντίστοιχη κλάση. Η συγκεκριμένη μεταβλητή θα αποτελεί τη μία από τις δύο μεταβλητές που θα χρησιμοποιηθούν για τον έλεγχο συσχέτισης, κάτι που θα αναλυθεί στα επόμενα κεφάλαια.

3.3 Συλλογή δεδομένων από το SDK4ED (TD Classifier)

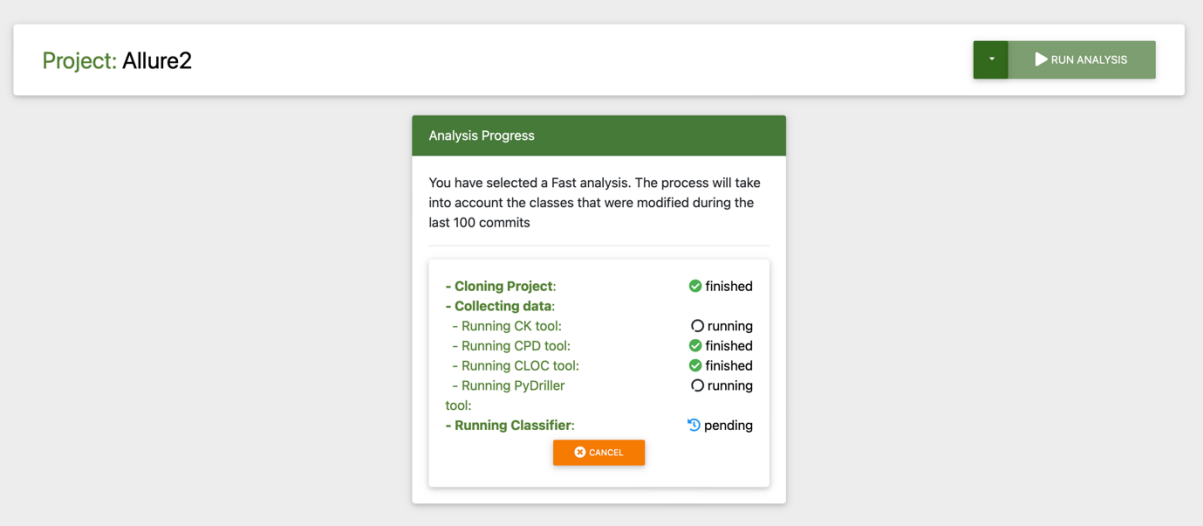
Σε αντίθεση με την πλατφόρμα VM4SEC, η SDK4ED πλατφόρμα παρέχει τη δυνατότητα συλλογής των αποτελεσμάτων σε μορφή JSON και μέσω του UI της, παρότι υποστηρίζει και τα δικά της API endpoints.

Έτσι, για να δημιουργήσει ο χρήστης το δικό του πρότζεκτ θα πρέπει να επιλέξει το κουμπί «+New Project». Στη συνέχεια θα εμφανιστεί η αντίστοιχη φόρμα, όπως στο VM4SEC, για να συμπληρώσει ο χρήστης τις απαραίτητες πληροφορίες από το σχετικό αποθετήριο.

Στη συνέχεια, ο χρήστης πρέπει να επιλέξει ένα από τα πρότζεκτ του και αφού φανεί το περίγραμμα πέριξ αυτού, να επιλέξει την επιλογή Technical Debt. Η συγκεκριμένη επιλογή έχει τρεις υπο-επιλογές, τις (α) TD Analysis, (β) TD New Code και (γ) TD Classifier.

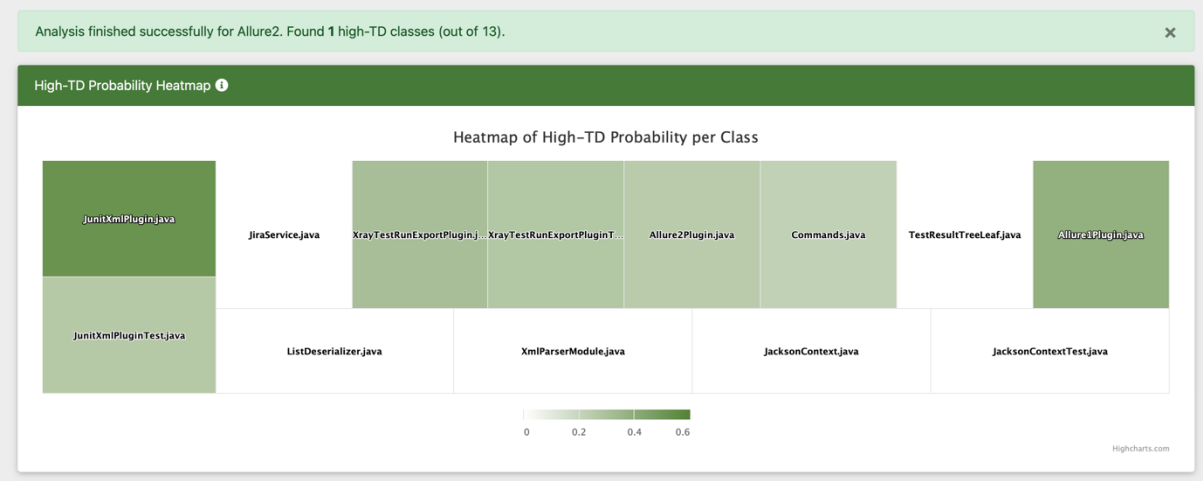
Ο χρήστης θα πρέπει να επιλέξει την τρίτη επιλογή (TD Classifier). Μόλις το κάνει, θα μεταφερθεί αυτόματα σε μία υπο-σελίδα στην οποία θα εμφανιστούν τα αποτελέσματα, μετά την ανάλυση. Πριν ξεκινήσει, ο χρήστης θα πρέπει να θέσει από τη λίστα επιλογών εάν επιθυμεί ανάλυση μικρής, μεσαίας ή μεγάλης εμβέλειας. Για την παρούσα εργασία επιλέχθηκε

η πρώτη επιλογή της μικρής εμβέλειας για να ακολουθεί την ίδια αναλογία με αυτή του VM4SEC καθώς δεν παρέχει παρόμοια επιλογή στην συγκεκριμένη πλατφόρμα, και αναλύει όλα τα πρότζεκτ με μικρή εμβέλεια εξαρχής. Μόλις ο χρήστης επιλέξει την εμβέλεια ανάλυσης και πατήσει το κουμπί «Run Analysis», τότε η ανάλυση θα ξεκινήσει ανανεώνοντας την κατάσταση της εκτέλεσης όπως φαίνεται στην παρακάτω εικόνα:



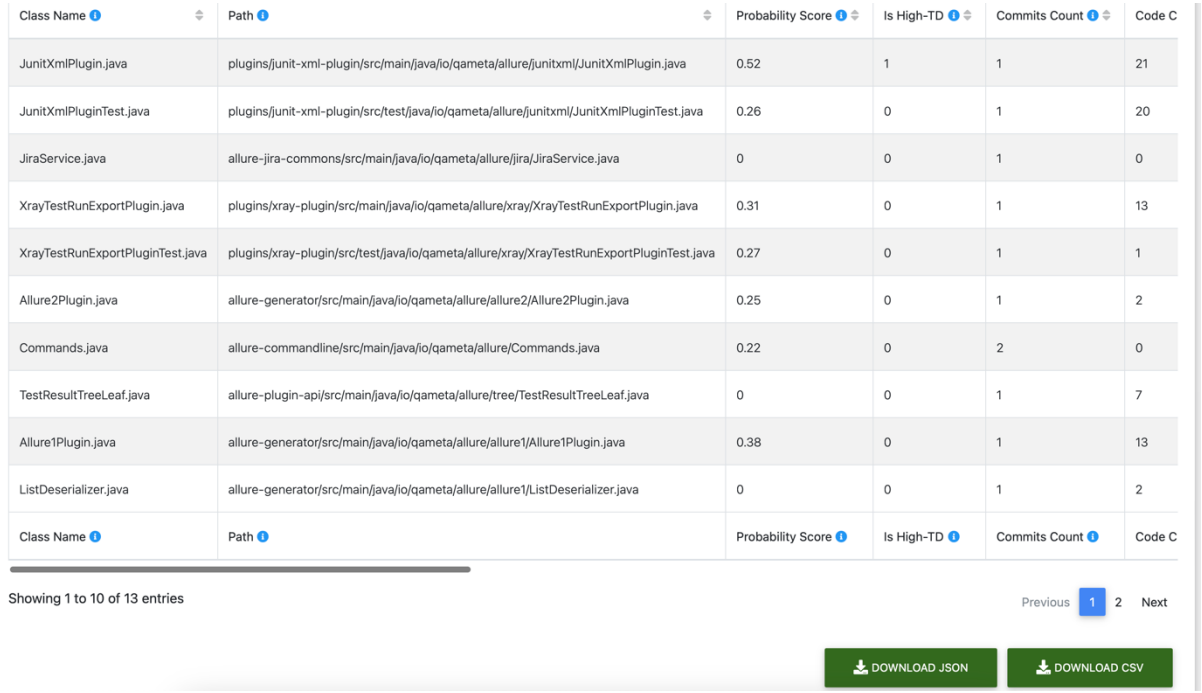
Όπως γίνεται κατανοητό, οι υπολειτουργίες με κατάσταση «Finished» υποδεικνύουν πως τελείωσαν επιτυχώς ενώ όσες είναι με την κατάσταση «Running» μας δείχνουν πως είναι ακόμα υπό εκτέλεση.

Μόλις τελειώσει επιτυχώς η ανάλυση, ο χρήστης θα δει ένα heatmap αντίστοιχο όπως το παρακάτω:



Από το παραπάνω heatmap ο χρήστης μπορεί με μία γρήγορη ματιά να καταλάβει ποιες κλάσεις έχουν σοβαρό πρόβλημα με το τεχνικό τους χρέος και ποιες όχι. Επίσης, με την ειδοποίηση πάνω από το heatmap, γίνεται και μία γρήγορη αναφορά στο αποτέλεσμα, ενημερώνοντας τον χρήστη για το πόσες κλάσεις έχουν σοβαρό ζήτημα συνολικά στο πρότζεκτ.

Μετά την εκτέλεση εάν ο χρήστης επιθυμεί να κατεβάσει τοπικά το αντίστοιχο JSON αρχείο με τα αποτελέσματα θα πρέπει να επιλέξει το κουμπί «DOWNLOAD JSON», όπως φαίνεται και στην παρακάτω εικόνα.



Class Name	Path	Probability Score	Is High-TD	Commits Count	Code C
JunitXmlPlugin.java	plugins/junit-xml-plugin/src/main/java/io/qameta/allure/junitxml/JunitXmlPlugin.java	0.52	1	1	21
JunitXmlPluginTest.java	plugins/junit-xml-plugin/src/test/java/io/qameta/allure/junitxml/JunitXmlPluginTest.java	0.26	0	1	20
JiraService.java	allure-jira-commons/src/main/java/io/qameta/allure/jira/JiraService.java	0	0	1	0
XrayTestRunExportPlugin.java	plugins/xray-plugin/src/main/java/io/qameta/allure/xray/XrayTestRunExportPlugin.java	0.31	0	1	13
XrayTestRunExportPluginTest.java	plugins/xray-plugin/src/test/java/io/qameta/allure/xray/XrayTestRunExportPluginTest.java	0.27	0	1	1
Allure2Plugin.java	allure-generator/src/main/java/io/qameta/allure/allure2/Allure2Plugin.java	0.25	0	1	2
Commands.java	allure-commandline/src/main/java/io/qameta/allure/Commands.java	0.22	0	2	0
TestResultTreeLeaf.java	allure-plugin-api/src/main/java/io/qameta/allure/tree/TestResultTreeLeaf.java	0	0	1	7
Allure1Plugin.java	allure-generator/src/main/java/io/qameta/allure/allure1/Allure1Plugin.java	0.38	0	1	13
ListDeserializer.java	allure-generator/src/main/java/io/qameta/allure/allure1/ListDeserializer.java	0	0	1	2
Class Name	Path	Probability Score	Is High-TD	Commits Count	Code C

Showing 1 to 10 of 13 entries

Previous 1 2 Next

DOWNLOAD JSON DOWNLOAD CSV

Η δομή των JSON αρχείων από την SDK4ED πλατφόρμα έχει ως εξής:


```
[
  {
    "class_path": "lottie\/src\/main\/java\/com\/airbnb\/lottie\/network\/NetworkFetcher.java",
    "class_name": "NetworkFetcher.java",
    "commits_count": 4,
    "code_churn_count": 3,
    "code_churn_max": 2,
    "code_churn_avg": 1,
    "contributors_count": 2.0,
    "contributors_experience": 72.09,
    "hunks_count": 4.0,
    "cbo": 15.0,
    "wmc": 29.0,
    "dit": 1.0,
    "rfc": 23.0,
    "lcom": 0.0,
    "total_methods": 7.0,
    "max_nested_blocks": 3.0,
    "loc": 103.0,
    "total_variables": 15.0,
    "duplicated_lines": 0.0,
    "comment_lines": 2.0,
    "ncloc": 131.0,
    "total_lines": 152.0,
    "high_td": 0,
    "high_td_proba": 0.11
  },
```

Στο αρχείο, εμπεριέχονται πολλών ειδών πληροφορίες για κάθε κλάση, όπως ο αριθμός των commits, ο αριθμός των συνολικών μεταβλητών, οι σειρές κ.α. Η μεταβλητή που αποτελεί την αντίστοιχη της «sigmoid», που είδαμε και στο αρχείο της VM4SEC, είναι η «high_td_proba». Μαζί με την «sigmoid» θα αποτελούν τις μεταβλητές της επικείμενης ανάλυσης που θα πραγματοποιηθεί για να εξακριβωθεί κατά πόσο υπάρχει συσχέτιση.

3.4 Φιλτράρισμα των JSON αρχείων

Επομένως, η πληροφορία από τα δύο είδη JSON αρχεία πέρασαν από φιλτράρισμα. Νέα JSON αποτελέσματα δημιουργήθηκαν με σκοπό την ευκολότερη χρήση τους από ειδικές βιβλιοθήκες στατιστικής ανάλυσης, όπως θα φανεί και στη συνέχεια.

Για το φιλτράρισμα των αποτελεσμάτων της SDK4ED χρησιμοποιήθηκε το παρακάτω κομμάτι κώδικα:

```
def filter_data(file_name):

    # Load the JSON data from a file
    global var1
    with open(f'' + file_name, 'r') as f:
        data = json.load(f)

    # Filter the data
    filtered_data = []
    for item in data:
        if item['high_td_proba'] > 0 or item['high_td_proba'] == 0:
            filtered_data.append({
                "class_name": item["class_name"],
                "high_td_proba": item["high_td_proba"]
            })

            # Output the filtered data to a new file
            var1 = file_name[:file_name.index(".")]
    with open(f'/Users/dimpap/PycharmProjects/MST/TD_JSON/Outcome/{var1} - Filtered.json', 'w') as f:
        json.dump(filtered_data, f, indent=4)


def iterator():
    path = r'/Users/dimpap/PycharmProjects/MST/PlayGround2'
    os.chdir(path)
    for file in os.listdir():
        if file.endswith('.json'):
            print(file)
            filter_data(file)
```

Αξίζει αν σημειωθεί πως ο έλεγχος έγινε αρχικά με βάση μόνο τις κλάσεις που έχουν την μεταβλητή «» μεγαλύτερη του μηδενός. Στην πορεία της εργασίας θεωρήθηκε λάθος προσέγγιση καθώς θα δημιουργούσε λογικά κενά και παράλληλα θα αλλοίωνε τα αποτελέσματα της συσχέτισης. Έτσι στην πορεία προστέθηκε και το κομμάτι της ισότητας με το μηδέν.

Οι δύο μεταβλητές που θα χρειαστούν είναι μόνο το όνομα της κλάσεις και το ποσοστό τεχνικού χρέους της. Έτσι, τα αποτελέσματα του νέου JSON αρχείου, μετά το φιλτράρισμά, θα έχουν την παρακάτω δομή:


```
{
    "class_name": "CryptoHelper.java",
    "high_td_proba": 0.01
},
{
    "class_name": "NoneAlgorithm.java",
    "high_td_proba": 0.01
},
{
    "class_name": "RSAAlgorithm.java",
    "high_td_proba": 0.01
},
{
    "class_name": "ClaimsSerializer.java",
    "high_td_proba": 0.03
},
{
    "class_name": "JWTParser.java",
    "high_td_proba": 0.02
},
{
    "class_name": "PayloadImpl.java",
    "high_td_proba": 0.01
},
{
    "class_name": "ConcurrentVerifyTest.java",
    "high_td_proba": 0.07
},
```


Αντίστοιχα, για το φιλτράρισμα των αποτελεσμάτων της VM4SEC χρησιμοποιήθηκε το παρακάτω κομμάτι κώδικα:

```
def filter_data(file_name):

    # Load the JSON data from a file
    global var1
    with open(f'' + file_name, 'r') as f:
        data = json.load(f)

    # Filter the data
    filtered_data = []
    for item in data['results']:
        if item['sigmoid'] > 0 or item['sigmoid'] == 0 :
            filtered_data.append({
                "class_name": item["class_name"],
                "sigmoid": item["sigmoid"]

            })

        # Output the filtered data to a new file
        var1 = file_name[:file_name.index(".")]
    with open(f'/Users/dimpap/PycharmProjects/MST/Security_JSON/Outcome/{var1} - Filtered.json', 'w') as f:
        json.dump(filtered_data, f, indent=4)

def iterator():
    path = r'/Users/dimpap/PycharmProjects/MST/PlayGround'
    os.chdir(path)
    for file in os.listdir():
        if file.endswith('.json'):
            print(file)
            filter_data(file)
```

Για το αποτέλεσμα της VM4SEC, οι δύο μεταβλητές που κρατήθηκαν είναι το όνομα της κλάσης και το αντίστοιχο ποσοστό προβλήματος ασφάλειας και ευπάθειας, όπως φαίνεται και στην παρακάτω εικόνα που δείχνει τη δομή του JSON αρχείου με τα αποθηκευμένα και φιλτραρισμένα αποτελέσματα:


```
{
    "class_name": "ApplicationTest.java",
    "sigmoid": 0.1
},
{
    "class_name": "NavigationDrawerAdapter.java",
    "sigmoid": 0.8915296793
},
{
    "class_name": "Connections.java",
    "sigmoid": 0.2167164981
},
{
    "class_name": "IReceivedMessageListener.java",
    "sigmoid": 0.1
},
{
    "class_name": "PersistenceException.java",
    "sigmoid": 0.6741962433
},
{
    "class_name": "Persistence.java",
    "sigmoid": 0.4138348997
},
```


Συμπερασματικά, μέσω του φιλτραρίσματος, για τα 30 αποθετήρια υπάρχουν 60 JSON αρχεία, 30 για την Ασφάλεια & Ευπάθεια καθώς και άλλα 30 για το Τεχνικό Χρέος. Με αυτά τα αρχεία, στα επόμενα κεφάλαια, θα γίνουν οι απαραίτητοι στατιστικοί έλεγχοι χρησιμοποιώντας τις κατάλληλες μεθοδολογίες. Με βάση τα αποτελέσματα θα φανεί εάν υπάρχει συσχέτιση μεταξύ των δύο κατηγοριών, και αν ναι, σε τι βαθμό.

4. Στατιστική Ανάλυση

Στο κεφάλαιο αυτό θα γίνει εκτενής ανάλυση αναφορικά με τις μεθοδολογίες που χρησιμοποιήθηκαν με σκοπό την εξακρίβωση συσχέτισης ή μη μεταξύ του Τεχνικού Χρέους και της Ασφάλειας & Ευπάθειας, αναφορικά με τα 30 αποθετήρια.

Θεωρήθηκε πως το δείγμα των 30 αποθετηρίων είναι ένας ικανοποιητικός αριθμός για να δώσει στην κοινότητα ένα αρκετά ρεαλιστικό αποτέλεσμα ώστε να χρησιμοποιηθεί ως βάση για περεταίρω έρευνα και ανάλυση μελλοντικά. Συνολικά χρησιμοποιήθηκαν τρεις μεθοδολογίες, οι: Pearson, Spearman και Permutation Test.

4.1 Μεθοδολογία Pearson

Αρχικά, χρησιμοποιήθηκε η Pearson μεθοδολογία, γνωστή και ως συντελεστής συσχέτισης Pearson. Είναι ένα στατιστικό μέτρο που ποσοτικοποιεί το βαθμό γραμμικής σχέσης μεταξύ δύο μεταβλητών.

Ο συντελεστής συσχέτισης Pearson θεωρείται παραμετρικός και συμβολίζεται με το σύμβολο «» και κυμαίνεται από -1 έως και +1. Η τιμή +1 υποδηλώνει τέλεια θετική συσχέτιση, που σημαίνει ότι καθώς αυξάνεται η μία μεταβλητή, αυξάνεται γραμμικά και η άλλη μεταβλητή. Η τιμή -1 υποδηλώνει τέλεια αρνητική συσχέτιση, που σημαίνει ότι καθώς αυξάνεται η μία μεταβλητή, η άλλη μεταβλητή μειώνεται γραμμικά. Η τιμή 0 υποδηλώνει ότι δεν υπάρχει συσχέτιση μεταξύ των δύο μεταβλητών.

Με τον όρο παραμετρικός αναφερόμαστε σε μια σειρά δεικτών που ικανοποιούν κάποιες προϋποθέσεις. Αν αυτές δεν πληρούνται, τότε γίνεται χρήση μη-παραμετρικών δεικτών, οι οποίοι επηρεάζονται από την τήρηση των προϋποθέσεων (ή όχι). Αυτές είναι (i) οι μεταβλητές να είναι συνεχείς σε ίσα διαστήματα και να έχουν κανονική κατανομή και (ii) να υπάρχει γραμμική σχέση μεταξύ των δύο μεταβλητών (δεν υπάρχουν ακραίες τιμές ή παρατηρήσεις με επιρροή στα δεδομένα).

Ο συγκεκριμένος συντελεστής χρησιμοποιείται συνήθως σε πολλούς τομείς, όπως η ψυχολογία, η οικονομία, η βιολογία και οι κοινωνικές επιστήμες, για μελέτη της σχέσης μεταξύ δύο μεταβλητών. Είναι ένα ευρέως χρησιμοποιούμενο μέτρο συσχέτισης λόγω της απλότητας και της ερμηνευσιμότητας του και επειδή υποθέτει ότι τα δεδομένα είναι κανονικά κατανεμημένα, γεγονός που το καθιστά εφαρμόσιμο σε ένα ευρύ φάσμα συνόλων δεδομένων. [22]

Ο μαθηματικός τύπος που ορίζει τη συγκεκριμένη μεθοδολογία είναι ο παρακάτω:

$$r = \frac{n\sum xy - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

όπου,
r = συντελεστής συσχέτισης Pearson
x και y είναι οι τιμές των δύο μεταβλητών που συγκρίνονται

Στην παρούσα εργασία έγινε χρήση της βιβλιοθήκης scipy και πιο συγκεκριμένα της συνάρτησης pearson από το πακέτο stats της βιβλιοθήκης. Η συγκεκριμένη συνάρτηση αυτό που δέχεται πρακτικά είναι δύο πίνακες. Αυτοί οι πίνακες θα πρέπει να είναι ίδιου μεγέθους για να μπορεί να υπάρχει 1-1 αντιστοιχία μεταξύ των τιμών τους. Ο ένας πίνακας αντικατοπτρίζει τη μεταβλητή x και ο άλλος τη μεταβλητή y από τον παραπάνω μαθηματικό τύπο. [23]

Μέσω της χρήσης Python και των κατάλληλων βιβλιοθηκών κατασκευάστηκε ο παρακάτω κώδικας:

```
def filter_data(file_name):
    global var1
    with open(file_name, 'r') as f:
        data = json.loads(''.join(f.readlines()))

    # Filter the data
    x = []
    y = []

    for item in data:
        x.append(item[1])
        y.append(item[2])

    r_p = pearsonr(x, y)

    print("--= Repos outcome =--  ")
    print("Correlation coefficient: \n" + str(np.corrcoef(x, y)))
    print("P-value: " + str(r_p.pvalue))
    print("Statistic: " + str(r_p.statistic))
    print("Confidence-high: " + str(r_p.confidence_interval().high))
    print("Confidence-low: " + str(r_p.confidence_interval().low))


def iterator():
    path = r'/Users/dimpap/PycharmProjects/MST/Outcome'
    os.chdir(path)
    for file in os.listdir():
        filter_data(file)
```

Ο κώδικας διαβάζει αρχεία τύπου JSON από ένα συγκεκριμένο φάκελο. Αφού τα διαβάσει, εισάγει τα στοιχεία τους σε δύο πίνακες (x, y). Στη συνέχεια, αποθηκεύεται το αποτέλεσμα της συνάρτησης pearsonr() στη μεταβλητή r_p. Όπως γίνεται κατανοητό, η συνάρτηση χρειάζεται μόνο τις δύο μεταβλητές που έμμεσα είναι οι δύο πίνακες μεταβλητών που διαβάστηκαν από τα JSON αρχεία. [24]

Στη συνέχεια, εφόσον η μεταβλητή r_p έχει αποθηκευμένο το αποτέλεσμα της pearson συνάρτησης, μπορεί να χρησιμοποιηθεί με τρόπο τέτοιο ώστε να εξαχθούν όλα τα απαραίτητα συμπεράσματα μέσω των διαθέσιμων στατιστικών μεταβλητών.

Έτσι έχουμε,

r_p.pvalue = είναι η τιμή του P από τη συγκεκριμένη εκτέλεση
r_p.statistic = είναι η τιμή του στατιστικού από τη συγκεκριμένη εκτέλεση
r_p.confidence_internval() = η μέθοδος αυτή υπολογίζει το διάστημα εμπιστοσύνης της στατιστικής του συντελεστή συσχέτισης για το δεδομένο επίπεδο εμπιστοσύνης. Για να γίνει αυτό εύκολα από τους χρήστες, υπάρχουν οι μεταβλητές high και low (βλ. κώδικα). [24]

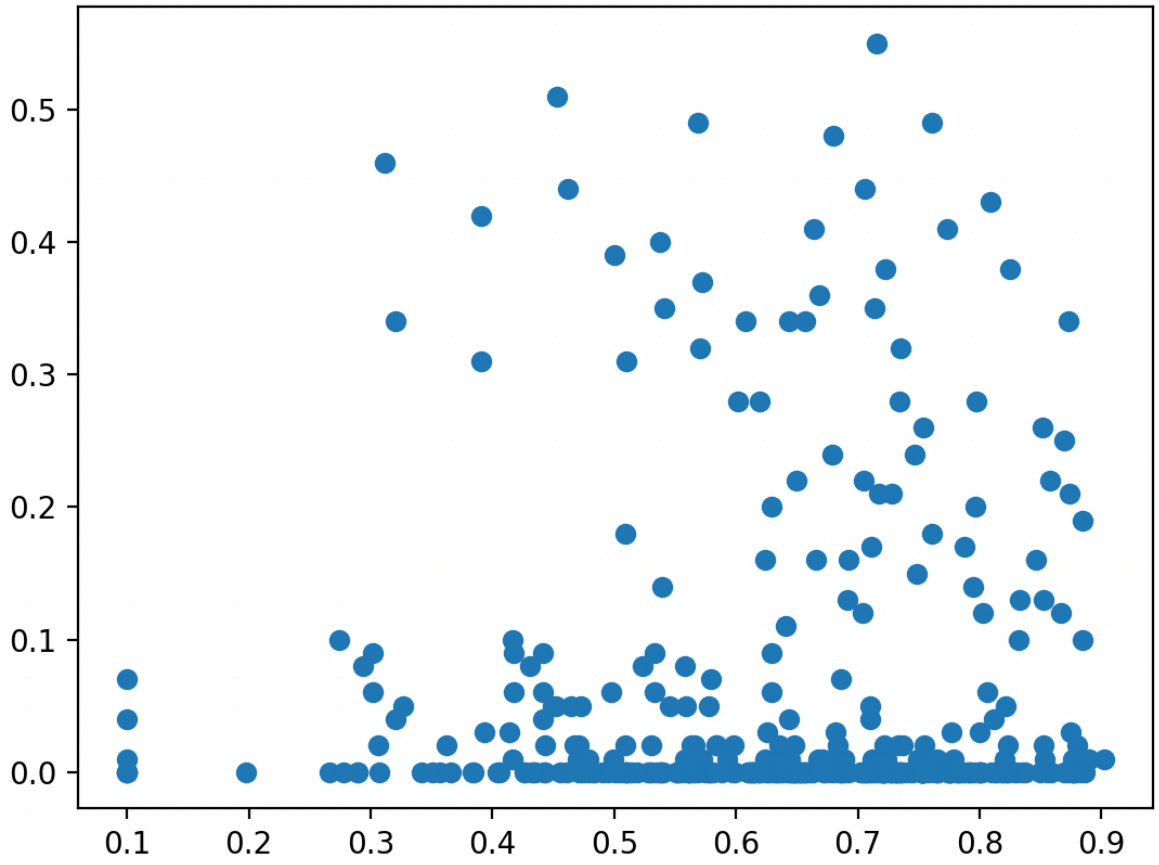
4.2 Αποτελέσματα Μεθοδολογίας Pearson

Μετά την εκτέλεση του κώδικα, έχουμε τα παρακάτω αποτελέσματα:

Repository	P-Value	Statistic	Conf. High	Conf. Low
Apache Shiro	0,851	0,034	0,384	-0,323
Apache_ShardingSphere	0,438	0,064	0,223	-0,098
Cucumber_JVM	0,862	-0,030	0,310	-0,365
Apache Spark	0,416	0,583	0,989	-0,859
Selenide	0,493	0,018	0,285	-0,141
Supertokens_Core	0,003	0,291	0,460	0,102
Junit5	0,477	0,08	0,365	-0,192
Auth0_Java_Jwt	0,233	0,216	0,525	-0,142
Keycloak	0,850	-0,010	0,099	-0,120
Facebook Fresco	0,602	0,080	0,165	-0,538
Appium_Java_Client	0,713	0,044	0,276	-0,192
Mockito	0,831	0,019	0,195	-0,314
Google Tsunami	0,221	0,134	0,345	-0,302
Google Dagger	0,354	0,050	-0,049	-0,394
MS Playwright	0,925	-0,070	0,118	-0,254
Apache_Shenyu	0,820	0,011	0,210	-0,090
Bumptech_Glide	0,747	0,036	0,177	-0,353
Selenium	0,170	-0,161	0,251	-0,304
SoapUI	0,036	-0,302	0,072	-0,551
Allure2	0,597	-0,161	0,427	-0,654
Eclipse_Paho_Mqtt	0,507	0,095	0,274	-0,440
Apache Jmeter	0,403	0,071	0,234	-0,095
Airbnb_Lottie_Android	0,359	0,110	0,335	-0,126
Termux_App	0,309	0,098	0,186	-0,285
Signal Android	0,526	-0,073	0,215	-0,260
Jenkins	0,338	-0,110	0,253	-0,277
Google_ExoPlayer	0,567	0,040	0,157	-0,168
TestNG	0,048	0,060	0,158	-0,174
Spring_Security	0,162	0,037	0,090	-0,015
Thoughtbot_Expandable	0,012	-0,558	-0,139	-0,807

Με μια γρήγορη ανάλυση, γίνεται φανερό πω το μοντέλο του Pearson έχει εφαρμογή στο συγκεκριμένο μοντέλο, καθώς η τιμή statistic είναι κοντά στην τιμή 0.05 στα 9 από τα 30 συνολικά αποθετήρια, δηλαδή το 30%. Αυτό μας οδηγεί στο συμπέρασμα πως μπορούμε να εξάγουμε υγιή συμπεράσματα για το δείγμα μας με τη συγκεκριμένη μεθοδολογία. Επίσης, 8 από τα αποθετήρια φαίνεται να έχουν μέτρια συσχέτιση (0,05-0,10), ποσοστό που αποτελεί το 26,6% του δείγματος των αποθετηρίων. Συνδυαστικά, πάνω από το 50% των αποθετηρίων φαίνεται να έχουν από μέτρια ως τέλεια συσχέτιση.

Παρακάτω δίνεται το Scatter διάγραμμα του αποθετηρίου Apache_Shenyu που είχε τη μικρότερη τιμή σε σχέση με τα υπόλοιπα αποθετήρια:



4.3 Μεθοδολογία Spearman

Η μεθοδολογία Spearman, επίσης γνωστή ως συντελεστής συσχέτισης Spearman, είναι ένα στατιστικό μέτρο που χρησιμοποιείται για την αξιολόγηση της ισχύος της σχέσης μεταξύ δύο μεταβλητών.

Η μεθοδολογία βασίζεται στην έννοια της κατάταξης. Περιλαμβάνει τον υπολογισμό της διαφοράς μεταξύ των κατατάξεων κάθε ζεύγους αντίστοιχων σημείων δεδομένων για τις δύο μεταβλητές. Στη συνέχεια, οι διαφορές τετραγωνίζονται και αθροίζονται, με αποτέλεσμα τον συντελεστή συσχέτισης κατάταξης του Spearman.

Χρησιμοποιείται συχνά στις κοινωνικές επιστήμες, την ψυχολογία και άλλους τομείς όπου τα δεδομένα μπορεί να μην πληρούν τις προϋποθέσεις που απαιτούνται για άλλα μέτρα συσχέτισης όπως ο συντελεστής συσχέτισης Pearson, όπως όταν τα δεδομένα δεν είναι κανονικά κατανεμημένα ή όταν υπάρχουν ακραίες τιμές. [26]

Ο μαθηματικός τύπος έχει ως εξής:

$$r = 1 - (6 \times \Sigma d2) \div (n \times (n^2 - 1))$$

όπου:

r είναι ο συντελεστής συσχέτισης Spearman's rank correlation coefficient
Σd^2 είναι το άθροισμα των τετραγωνικών διαφορών μεταξύ των τάξεων των δύο μεταβλητών
n είναι ο αριθμός των ζευγών δεδομένων

Ο τύπος περιλαμβάνει τέσσερα βασικά βήματα:

(1) Κατάταξη των δεδομένων για κάθε μεταβλητή χωριστά, αποδίδοντας στη χαμηλότερη τιμή την κατάταξη 1, στην αμέσως χαμηλότερη την κατάταξη 2 κ.ο.κ. Εάν υπάρχουν ισοβαθμίες, αποδίδεται ο μέσος όρος της κατάταξης.
(2) Υπολογισμός των διαφορών μεταξύ των βαθμών κάθε ζεύγους αντίστοιχων σημείων δεδομένων για τις δύο μεταβλητές.
(3) Τετραγωνισμός κάθε διαφοράς.
(4) Χρήση του παραπάνω τύπου για τον υπολογισμό του συντελεστή συσχέτισης κατάταξης Spearman.

Το αποτέλεσμα του τύπου είναι ένας αριθμός μεταξύ -1 και 1. Η τιμή -1 υποδηλώνει τέλεια αρνητική συσχέτιση, το 0 υποδηλώνει καμία συσχέτιση και το 1 υποδηλώνει τέλεια θετική συσχέτιση. Μια τιμή μεταξύ -1 και 1 υποδεικνύει την ισχύ και την κατεύθυνση της συσχέτισης, με υψηλότερες απόλυτες τιμές να υποδεικνύουν ισχυρότερες συσχετίσεις. [25]

Για την χρήση της Spearman μεθοδολογίας στην πράξη, χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python και η βιβλιοθήκη scipy και πιο συγκεκριμένα από το πακέτο stats η συνάρτηση spearmanr. [27]

Ο κώδικας είναι ο παρακάτω:

```
def filter_data(file_name):
    global var1
    with open(file_name, 'r') as f:
        data = json.loads(''.join(f.readlines()))

    # Filter the data
    x = []
    y = []

    for item in data:
        x.append(item["sigmoid"])
        y.append(item["high_td_proba"])

    print(" ")
    print("Repository: " + file_name)
    result = stats.spearmanr(x, y)
    print("Correlation result: ", str(result.correlation)[:5])

def iterator():
    path = r'/Users/dimpap/PycharmProjects/MST/Outcome'
    os.chdir(path)
    for file in os.listdir():
        filter_data(file)
```

Οι τιμές «sigmoid» και «high_td_proba» αντικατοπτρίζουν τις μεταβλητές x και y αντίστοιχα. Οι x και y αντίστοιχα, θα χρησιμοποιηθούν μέσω της συνάρτησης spearmanr για τον υπολογισμό της συσχέτισης και της P μεταβλητής.

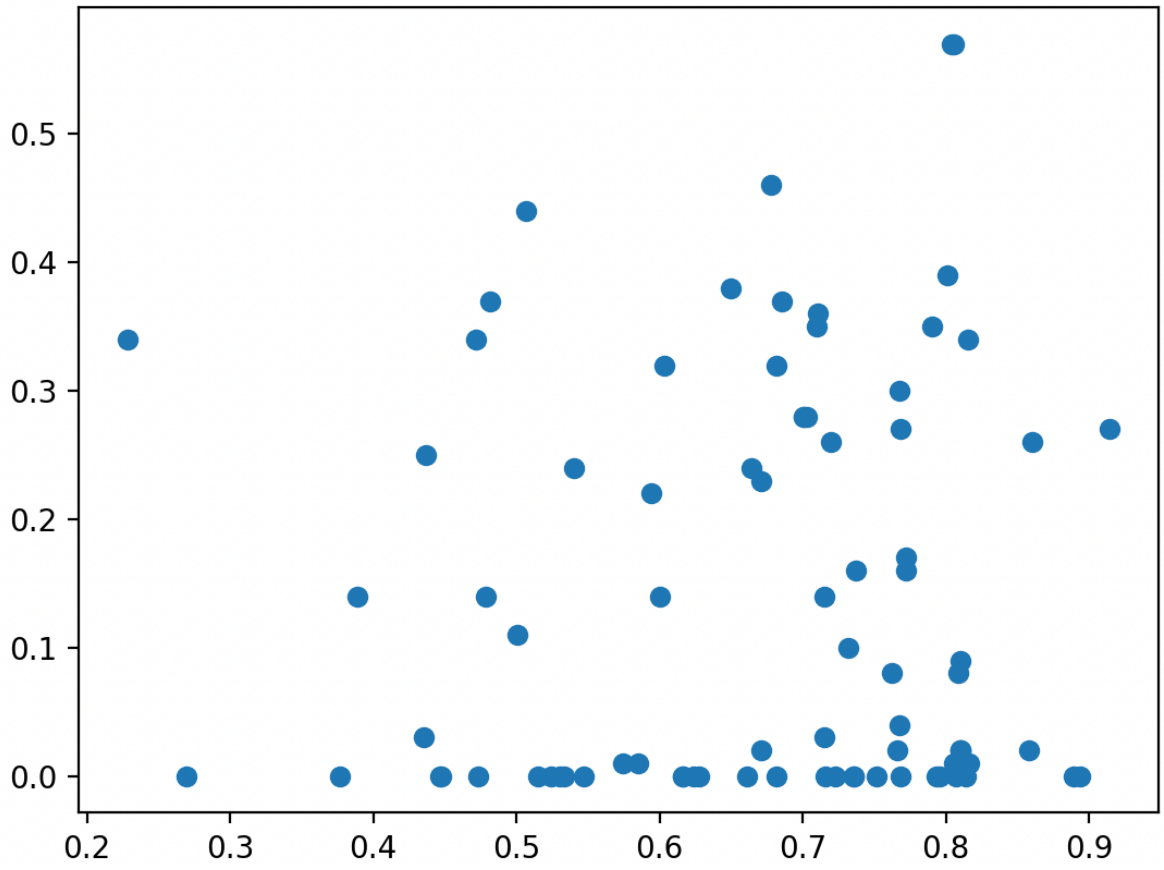
4.4 Αποτελέσματα Μεθοδολογίας Spearman

Για λόγους ευκολίας, θα κρατηθούν τα πρώτα τρία ψηφία του float αποτελέσματος από τη συνάρτηση, για να γίνει κατανοητό από τον χρήστη. Έτσι τα αποτελέσματα είναι τα παρακάτω:

Repository	Correlation
Apache Shiro	0,001
Apache_ShardingSphere	-0,10
Cucumber_JVM	-0,07
Apache Spark	0,737
Selenide	-0,02
Supertokens_Core	0,320
Junit5	0,069
Auth0_Java_Jwt	0,114
Keycloak	-0,10
Facebook Fresco	0,120
Appium_Java_Client	0,134
Mockito	0,044
Google Tsunami	0,185
Google Dagger	0,070
MS Playwright	-0,01
Apache_Shenyu	-0,03
Bumptech_Glide	0,053
Selenium	-0,21
SoapUI	-0,31
Allure2	-0,27
Eclipse_Paho_Mqtt	0,07
Apache Jmeter	0,081
Airbnb_Lottie_Android	0,047
Termux_App	0,159
Signal Android	-0,10
Jenkins	-0,12
Google_ExoPlayer	-0,01
TestNG	0,077
Spring_Security	0,026
Thoughtbot_Expandable	-0,53

Από τα αποτελέσματα μπορούμε να συμπεράνουμε πως 5 αποθετήρια έχουν στατιστικό μικρότερο από το 0.05, επομένως το δείγμα είναι στατιστικά σημαντικό. Πρακτικά από τα 30 αποθετήρια, το 16,5% αυτών φαίνεται να έχει δυνατές ενδείξεις για συσχέτιση μεταξύ του Τεχνικού Χρέους και της Ασφάλειας & Ευπάθειας. Επίσης, 4 από τα 30 αποθετήρια παρουσιάζουν μέτρια συσχέτιση αποτελώντας το 13,3% του δείγματος. Συνδυαστικά, σχεδόν το 30% του δείγματος φαίνεται να έχει μέτρια προς τέλεια συσχέτιση.

Παρακάτω δίνεται το Scatter διάγραμμά του αποθετηρίου με τη μεγαλύτερη, προς το 0.05, συσχέτιση:



4.5 Μεθοδολογία Permutation Test

Η μεθοδολογία του Permutation Test, ή αλλιώς του τεστ μετάθεσης, είναι ένα στατιστικό τεστ που χρησιμοποιείται για να προσδιοριστεί αν δύο ομάδες δεδομένων έχουν διαφορετικούς μέσους όρους ή αν μια παρατηρούμενη διαφορά μεταξύ δύο ομάδων είναι στατιστικά σημαντική. Είναι μια μη παραμετρική δοκιμή που δεν βασίζεται σε υποθέσεις σχετικά με την υποκείμενη κατανομή των δεδομένων.

Το τεστ μετάθεσης λειτουργεί με τυχαία μετάθεση των ετικετών των σημείων δεδομένων μεταξύ των δύο ομάδων και στη συνέχεια με τον υπολογισμό του στατιστικού ελέγχου (π.χ. διαφορά των μέσων όρων) για κάθε μετάθεση. Επαναλαμβάνοντας αυτή τη διαδικασία πολλές φορές, λαμβάνεται μια μηδενική κατανομή του στατιστικού ελέγχου, η οποία αντιπροσωπεύει την κατανομή του στατιστικού ελέγχου υπό την υπόθεση ότι δεν υπάρχει διαφορά μεταξύ των δύο ομάδων.

Η τιμή p-value του παρατηρούμενου στατιστικού ελέγχου υπολογίζεται στη συνέχεια ως το ποσοστό των τυχαίων μεταθέσεων που παρήγαγαν ένα στατιστικό ελέγχου εξίσου ακραίο ή πιο ακραίο από το παρατηρούμενο στατιστικό ελέγχου. Εάν η p-τιμή είναι μικρή (συνήθως μικρότερη από 0,05), υποδηλώνει ότι η παρατηρούμενη διαφορά μεταξύ των δύο ομάδων είναι στατιστικά σημαντική.

Το τεστ μετάθεσης είναι ένα ισχυρό εργαλείο σε περιπτώσεις όπου οι υποθέσεις των παραδοσιακών παραμετρικών δοκιμών (όπως η δοκιμή t) παραβιάζονται ή όταν το μέγεθος του δείγματος είναι μικρό. Χρησιμοποιείται συνήθως σε διάφορους τομείς, όπως η βιολογία, η ψυχολογία και οι κοινωνικές επιστήμες. [28] [29]

Το τεστ μετάθεσης είναι μια γενική μέθοδος που μπορεί να εφαρμοστεί σε μια ποικιλία στατιστικών δοκιμών, ανάλογα με το ερευνητικό ερώτημα και τα δεδομένα που εξετάζονται. Τα βασικά βήματα για τη διενέργεια μιας δοκιμής μεταστοιχείωσης είναι τα εξής:

(1) Πρέπει να οριστεί μια στατιστική δοκιμής που μετρά τη διαφορά μεταξύ των δύο ομάδων ενδιαφέροντος (π.χ. διαφορά μέσων όρων, διαφορά διαμέσων κ.λπ.).

(2) Υπολογισμός της παρατηρούμενης τιμής του στατιστικού ελέγχου από τα αρχικά δεδομένα.

(3) Τυχαία μετάθεση των ετικετών των σημείων δεδομένων μεταξύ των δύο ομάδων, δημιουργώντας ένα νέο σύνολο τυχαιοποιημένων δεδομένων.

(4) Εκ νέου υπολογισμός του στατιστικού ελέγχου για κάθε αντιμετάθεση.

(5) Επανάληψη των βημάτων 3-4 πολλές φορές (π.χ. 1000 φορές) για να εξαχθεί μια μηδενική κατανομή του στατιστικού ελέγχου υπό τη μηδενική υπόθεση ότι δεν υπάρχει διαφορά μεταξύ των ομάδων.

(6) Υπολογισμός της τιμής p ως το ποσοστό των τυχαιοποιημένων δειγμάτων που παρήγαγαν ένα στατιστικό ελέγχου εξίσου ακραίο ή πιο ακραίο από το παρατηρούμενο στατιστικό ελέγχου. [29]

Ο τύπος για τον υπολογισμό της p-τιμής του τεστ αντιμετάθεσης εξαρτάται από το συγκεκριμένο στατιστικό τεστ που χρησιμοποιείται. Ακολουθεί ένα παράδειγμα του τύπου για ένα t-test δύο δειγμάτων:

$$t = (\overline{x1} - \overline{x2}) \div (\sigma \times \sqrt{(1/n1 + 1/n2)})$$

όπου:

$\bar{x}1$ και $\bar{x}2$ είναι οι δειγματικοί μέσοι των δύο ομάδων
$n1$ και $n2$ είναι τα μεγέθη του δείγματος των δύο ομάδων
Σ είναι η συγκεντρωτική τυπική απόκλιση των δύο ομάδων

Για τον υπολογισμό της τιμής p-value, θα πρέπει να συγκριθεί η παρατηρούμενη τιμή του t με τη μηδενική κατανομή του t που προκύπτει από τα τυχαιοποιημένα δείγματα. Η τιμή p-value είναι το ποσοστό των τυχαιοποιημένων δειγμάτων που έχουν τιμή t εξίσου ακραία ή πιο ακραία από την παρατηρούμενη τιμή t.

Με την χρήση της γλώσσας προγραμματισμού Python, της βιβλιοθήκης scipy, του πακέτου stats της βιβλιοθήκης και τελικώς της συνάρτησης permutation_test κατασκευάστηκε ο κατάλληλος κώδικας για τον στατιστικό έλεγχο των 30 αποθετηρίων:

```
def __init__(self, given_x, given_y):
    self.data1 = given_x
    self.data2 = given_y

@staticmethod
def statistic(data1, data2, axis):
    return np.mean(data1, axis=axis) - np.mean(data2, axis=axis)

def get_permutation_values(self):
    res = permutation_test((self.data1, self.data2),
                           statistic=self.statistic, permutation_type='pairings',
                           vectorized=True, alternative='two-sided')

    print("Statistic: ", res.statistic)
    print("P-value: ", res.pvalue)
```

Όπως αναφέρθηκε και παραπάνω, το τεστ μετάθεσης χρειάζεται μια στατιστική δοκιμή μεταξύ των δύο μεταβλητών της υποψήφιας συσχέτισης, όπως τη διαφορά μέσων όρων, ή τη διαφορά διαμέσων. Έτσι, στη συνάρτηση statistic υπολογίζεται η διαφορά των διάμεσων μεταξύ των δεδομένων των δύο μεταβλητών.

Στη συνέχεια, στη συνάρτηση get_permutation_values υπολογίζεται το αποτέλεσμα του τεστ μετάθεσης και επιστρέφεται εκεί που το καλεί ο χρήστης (βλ. παρακάτω). Σε αυτό το σημείο θα αναλυθεί η σημασία της κάθε παραμέτρου που χρησιμοποιείται εντός της συνάρτησης permutation_test, με σκοπό να γίνει κατανοητό για ποιο λόγο επιλέχθηκαν για το συγκεκριμένο δείγμα δεδομένων:

Statistic: στη μεταβλητή αυτή πρέπει να μετατεθεί το αποτέλεσμα της συνάρτησης statistic

Permutation_type: η συνάρτηση διαθέτει τρεις επιλογές για τη συγκεκριμένη μεταβλητή, τις «samples», «pairings» και «independent».

- Η τρίτη επιλογή είναι για παρατηρήσεις που αντιστοιχίζονται σε διαφορετικά δείγματα. Τα δείγματα μπορεί να περιέχουν διαφορετικούς αριθμούς παρατηρήσεων. Αυτός ο τύπος μετάθεσης είναι κατάλληλος για ανεξάρτητες δοκιμές υποθέσεων όπως το Mann-Whitney και τα ανεξάρτητα δείγματα t-test.
- Αναφορικά με το «pairings», οι παρατηρήσεις συνδυάζονται με διαφορετικές παρατηρήσεις, αλλά παραμένουν στο ίδιο δείγμα. Αυτός ο τύπος είναι κατάλληλος για δοκιμές με στατιστικά στοιχεία όπως του Spearman, Kendall και Pearson.
- Τέλος, η επιλογή «» είναι κατάλληλη για δοκιμές υποθέσεων ζευγών δειγμάτων, όπως η δοκιμασία υπογεγραμμένης κατάταξης Wilcoxon και δοκιμή ζεύξης t.

Vectorized: αν η τιμή οριστεί ως False, τότε η μεταβλητή statistic δεν θα χρησιμοποιηθεί και θα εκτελεστεί η συνάρτηση λαμβάνοντας μόνο τον έναν από τους δύο πίνακες (1D).

Alternative: Η εναλλακτική υπόθεση για την οποία υπολογίζεται η τιμή P-value. Υπάρχουν τρεις επιλογές, «greater», «less» και «two-sided».

- Greater: το ποσοστό της μηδενικής κατανομής που είναι μεγαλύτερο ή ίσο με την παρατηρούμενη τιμή της στατιστικής δοκιμής.
- Less: το ποσοστό της μηδενικής κατανομής που είναι μικρότερο ή ίσο με την παρατηρούμενη τιμής της στατιστικής δοκιμής
- Two-sided: διπλάσια μικρότερη από τις παραπάνω τιμές (προεπιλογή και επιθυμητή επιλογή για το δείγμα της συγκεκριμένης έρευνας) [32]

```python
with open('Security_JSON/Outcome/Apache_Spark_V - Filtered.json', 'r') as f:
    sec_data = json.load(f)

with open('TD_JSON/Outcome/Apache_Spark_T - Filtered.json', 'r') as f:
    td_data = json.load(f)

max_array = sec_data if len(sec_data) > len(td_data) else td_data
min_array = td_data if len(sec_data) > len(td_data) else sec_data

final_array = []
x = []
y = []

for min1 in min_array:
    for max1 in max_array:
        if max1['class_name'].__eq__(min1['class_name']):
            try:
                final_array.append({

                    "class_name": max1["class_name"],
                    "sigmoid": max1["sigmoid"],
                    "high_td_proba": min1["high_td_proba"]

                })
                x.append(max1["sigmoid"])
                y.append(min1["high_td_proba"])

            except KeyError:
                if max1['high_td_proba'] > 0 and min1['sigmoid'] > 0:
                    correlation += 1

with open(f'/Users/dimpap/PycharmProjects/MST/Outcome/Apache_Spark.json', 'w') as f:
    json.dump(final_array, f, indent=4)

permutation_obj = PERMUTATION(x, y)
permutation_obj.get_permutation_values()
```

Ο παραπάνω κώδικας δημιουργήθηκε με σκοπό την εκτέλεση του τεστ μετάθεσης και την επιστροφή των τιμών πίσω στον χρήστη. Παρακάτω, θα παρουσιαστεί ο κώδικας που δέχεται, και φιλτράρει τα δεδομένα υπό μορφή JSON αρχείων και στη συνέχεια χρησιμοποιώντας τον παραπάνω κώδικα, εξάγει και το αντίστοιχο στατιστικό αποτέλεσμα.

Η λογική γύρω από το δεύτερο κομμάτι κώδικα, έχει ως εξής:

- Αποθηκεύονται τα δεδομένα των JSON αρχείων στους κατάλληλους πίνακες (sec_data για τα δεδομένα από το αρχείο που σχετίζεται με την Ασφάλεια & Ευπάθεια και td_data για τα δεδομένα από το αρχείο που σχετίζεται με το Τεχνικό Χρέος).
- Φιλτράρονται οι πίνακες, επιλέγονται οι αντίστοιχες τιμές (sigmoid & high_td_proba) και αποθηκεύονται σε ένα νέο πίνακα συγκεντρωτικά).
- Αποθηκεύεται η κάθε μεταβλητή σε έναν πίνακα (x & y) με σκοπό να σταλθούν ως είσοδος δεδομένων στην κλάση PERMUTATION().
- Μετά τη δημιουργία του αντικειμένου τύπου PERMUTATION καλείται η συνάρτηση get_permutation_values και εκτελείται η ανάλογη μεθοδολογία, επιστρέφοντας και τα κατάλληλα αποτελέσματα.
- Τέλος, ο νέος πίνακας με τα συγκεντρωτικά δεδομένα αποθηκεύεται σε ένα αρχείο με τη μορφή JSON.

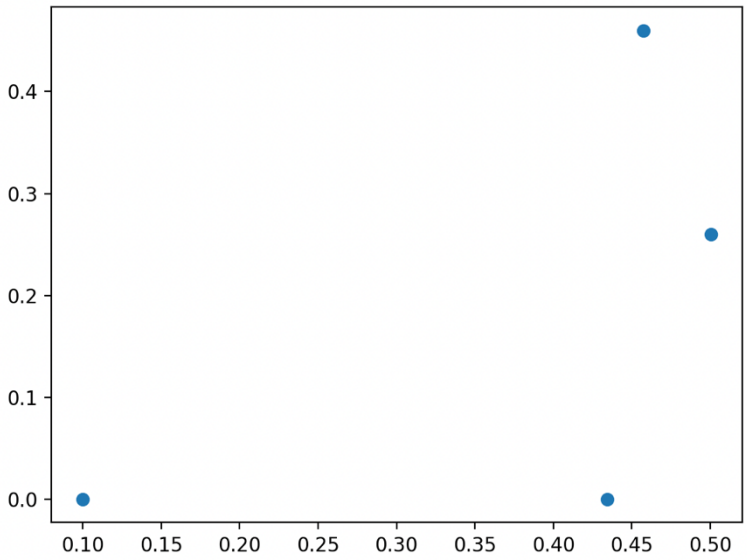
4.6 Αποτελέσματα του Permutation Test

Τα αποτελέσματα του ελέγχου μετάθεσης, για τα 30 αποθετήρια της έρευνας, είναι τα παρακάτω:

Repository	P-Value	Statistic
Apache Shiro	0,0002	0,621
Apache_ShardingSphere	0,0002	0,578
Cucumber_JVM	0,0002	0,489
Apache Spark	0,228	0,193
Selenide	0,0002	0,656
Supertokens_Core	0,0002	0,418
Junit5	0,0002	0,484
Auth0_Java_Jwt	0,0002	0,457
Keycloak	0,0002	0,513
Facebook Fresco	0,0002	0,520
Appium_Java_Client	0,0002	0,496
Mockito	0,0002	0,542
Google Tsunami	0,0002	0,550
Google Dagger	0,0002	0,469
MS Playwright	0,0002	0,393
Apache_Shenyu	0,0002	0,580
Bumptech_Glide	0,0002	0,530
Selenium	0,0002	0,456
SoapUI	0,0002	0,472
Allure2	0,0002	0,460
Eclipse_Paho_Mqtt	0,0002	0,456
Apache Jmeter	0,0002	0,407
Airbnb_Lottie_Android	0,0002	0,585

Termux_App	0,0002	0,483
Signal Android	0,0002	0,426
Jenkins	0,0002	0,512
Google_ExoPlayer	0,0002	0,437
TestNG	0,0002	0,561
Spring_Security	0,0002	0,586
Thoughtbot_Expandable	0,0002	0,530

Από τα παραπάνω αποτελέσματα μπορεί να γίνει αντιληπτό πως μόνο ένα αποθετήριο έχει τη τιμή του P-value του μεγαλύτερο του 5% (0,05), και πιο συγκεκριμένα το *Apache_Spark* του οποίου η τιμή είναι περίπου 0,22. Το τελευταίο, μπορεί να φανεί και από το διάγραμμα παρακάτω, στο οποίο μπορεί να γίνει διακριτό πως δεν υπάρχει συσχέτιση. Βέβαια, από τη στατιστική σκοπιά, λόγω του μικρού δείγματος δεδομένων, κάτι τέτοιο είναι λογικό. Το επίσης θετικό είναι πως το συγκεκριμένο αποθετήριο δεν παρουσιάζει διαφορετικά αποτελέσματα με τις άλλες μεθόδους, κάτι που θα ήταν πολύ οξύμωρο ίσχυε και θα έθετε μεγάλους προβληματισμούς για το τελικό συμπέρασμα.



Τα υπόλοιπα αποθετήρια, έχουν σχεδόν μηδενική τιμή για τη συγκεκριμένη μεταβλητή (0.0002). Αυτό σημαίνει πως όλα τα αποθετήρια έχουν θετική συσχέτιση μεταξύ των δύο μεταβλητών τους. Ο κυριότερος λόγος που η συγκεκριμένη μεθοδολογία έχει εντελώς διαφορετικά αποτελέσματα από τις δύο προηγούμενες, είναι ο βασικός τρόπος λειτουργίας της. Ουσιαστικά το τεστ μετάθεσης λειτουργεί με την λογική της επαναδειγματοληψίας, υπό την αρχική συνθήκη, τη μηδενική, η οποία υποστηρίζει πως όλα τα δείγματα προέρχονται από την ίδια κατανομή. [30] [31]

Αξίζει να σημειωθεί, πως το σύνολο των αποθετηρίων που πληρούν τα κριτήρια συσχέτισης, σύμφωνα με το τεστ μετάθεσης, μεταξύ Τεχνικού Χρέους και Ασφάλειας & Ευπάθειας, αποτελεί το 96,6% του δείγματος των αποθετηρίων της έρευνας.

Τα αποτελέσματα είναι κάπως αναμενόμενα καθώς η τεχνική του τεστ μετάθεσης λόγω της επαναδειγματοληψίας δημιουργεί ένα μοντέλο που κρατάει τις πιο θετικές τιμές της κάθε επανάληψης, κάτι που δε κάνει η τεχνική Pearson και η τεχνική Spearman.

5. Συμπέρασμα

Με το πέρας της εργασίας εξάγονται πολλά συμπεράσματα, τα οποία δημιουργούν χώρο προς μελέτη και επέκταση, αλλά και πληροφορία που μπορεί να χρησιμοποιηθεί από την κοινότητα πρακτικά

Επιπρόσθετα, με τις τρεις μεθοδολογίες που χρησιμοποιήθηκαν μπορεί ο αναγνώστης να καταλάβει το όφελος της κάθε μίας, σε τι διαφέρουν με σκοπό την κατάλληλη επιλογή για μελλοντικές έρευνες με αντίστοιχο σκοπό και δείγμα.

Με τη βοήθεια του παρακάτω πίνακα μπορούμε να δούμε στην πράξη το ποσοστό των συνολικών αποθετηρίων, από το δείγμα, που φαίνεται να έχουν θετική και αποδεκτή συσχέτιση:

Pearson	Spearman	Permutation Test
30% (9/30) – 26,6% (Μέτρια)	16,5% (5/30) – 13,3% (Μέτρια)	96,6% (29/30)

Από το πιο αυστηρό στατιστικά αποτέλεσμα μέχρι το πιο αισιόδοξο, γίνεται αντιληπτό πως το τεχνικό χρέος υπάρχει, και συνδυάζεται άμεσα με τα ζητήματα ασφάλειας. Αυτό επιβεβαιώνει, που είναι βέβαια ήδη γνωστό στη κοινότητα [1], πως όσο το τεχνικό χρέος αντιμετωπίζεται προληπτικά και με τις κατάλληλες τεχνικές, τότε και η πιθανότητα ύπαρξης σφάλματος ασφαλείας ή ευπάθειας μειώνεται.

Επιπρόσθετα, και συμπληρωματικά με το παραπάνω συμπέρασμα, η ύπαρξη ζητημάτων ασφαλείας και ευπαθειών, μπορεί επίσης να δημιουργήσει αύξηση του τεχνικού χρέους. Επομένως, η πιθανότητα ζητήματος ευπάθειας αυξάνεται με την αύξηση του τεχνικού χρέους και vice versa. Αυτό μας δίνει το συμπέρασμα πως οι κανόνες συγγραφής κώδικα αναφορικά με λογισμικό ασφάλειας πρέπει να αποτελούν μέρος των κανόνων του τεχνικού χρέους, καθώς άμεσα ή έμμεσα το επηρεάζουν.

Με τη σειρά τους, οι πλατφόρμες SDK4ED και VM4SEC, μπορούν να συνδράμουν στην εξόρυξη δεδομένων πολύ άμεσα και εύκολα. Συνδυαστικά με τις κατάλληλες στατιστικές βιβλιοθήκες, ένας μηχανικός λογισμικού μπορεί πολύ εύκολα να χτίσει ένα μοντέλο ελέγχου, διαχείρισης και εστίασης του τεχνικού χρέους σε κάποιο έργο λογισμικού.

Πολύ άμεσα μπορούν να εμφανιστούν οι ευπάθειες και οι κατάλληλοι συσχετισμοί σε κάθε εναλλαγή του SDLC (Software Development Life Cycle) αντιμετωπίζοντας σταδιακά και στον σωστό χρόνο το κάθε πιθανό πρόβλημα.

Εκτός της πρακτικής χρήσης από το σημείο της εξόρυξης δεδομένων, του φιλτραρίσματός τους, της ανάλυσης του αποτελέσματος και της τελικής εστίασης, με την κατάλληλη λύση, υπάρχει αρκετός χώρος και για την ερευνητική επικέντρωση στο συγκεκριμένο θέμα.

Όπως αναφέρθηκε στην αρχή, τα αποθετήρια επιλέχθηκαν ανά κατηγορία (QA, Security, κ.α.), με σκοπό την επιμέρους ανάλυση της κάθε κατηγορίας. Το συμπέρασμα

είναι πως τα 30 αποθετήρια, παρότι ήταν ικανοποιητικός αριθμός για την ανάλυση των συσχετίσεων δεν επαρκεί για να εξάγει αναλυτικά αποτελέσματα για την κάθε κατηγορία.

Έτσι, μελλοντικά θα ήταν πολύ καλή επέκταση η χρήση τουλάχιστον 20 αποθετηρίων ανά κατηγορία, με σκοπό την εξόρυξη ρεαλιστικών αποτελεσμάτων ώστε να φανεί εάν μία κατηγορία έχει μεγαλύτερο ζήτημα σε θέματα ασφάλειας ή σε τεχνικό χρέος και γιατί.

Θα ήταν πολύ χρήσιμο να αναλύονται και οι τεχνικές/αρχιτεκτονικές του κάθε αποθετηρίου, καθώς κατά αυτόν τον τρόπο μπορούν να συγκεντρωθούν οι πιο αποτελεσματικές τεχνικές για το τεχνικός χρέος και την ασφάλεια & ευπάθεια αντίστοιχα. Μετά τη συγκέντρωση αυτών, η χρήση τους σε επιμέρους αποθετήρια θα μείωναν τους παραπάνω δείκτες δημιουργώντας έτσι μία φόρμουλα αλληλεπίδρασης με σκοπό την εμπειρική αυτό-βελτίωση του κάθε αποθετηρίου εντός του ερευνητικού δείγματος.

6. Ευχαριστήρια

Θα ήθελα στο σημείο αυτό να ευχαριστήσω τον κύριο Χατζηγεωργίου Αλέξανδρο για την κατεύθυνση της εργασίας. Υπήρξαν κομβικά σημεία στα οποία η καθοδήγηση ήταν απαραίτητη για τη συνέχιση της εργασίας. Ο ερευνητικός της χαρακτήρας είχε μια ιδιαιτερότητα, και χωρίς την επίλυση των ζητημάτων που εμφανίστηκαν δε θα ήταν δυνατή η τελειοποίησή της.

Επιπρόσθετα, θα ήθελα να ευχαριστήσω θερμά τους Δημήτριο Τσουκαλά και Μιλτιάδη Σιάββα για την βοήθεια τους σχετικά με τα εργαλεία SDK4ED & VM4SEC αντίστοιχα. Η βοήθεια τους έπαιξε σημαντικό ρόλο στην εργασία και την έρευνα καθώς έμαθα να χρησιμοποιώ τα εργαλεία που αναφέρθηκαν, χωρίς τη χρήση των οποίων δε θα μπορούσε να γίνει η παρούσα έρευνα. Τα εργαλεία, με τις τεχνικές μηχανικής μάθησης που χρησιμοποιούν συνδυαστικά με τον όγκο των δεδομένων που συλλέχθηκαν από αυτά, συνέδραμαν στην έναρξη και, τελικώς, υλοποίηση της έρευνας.

ΠΗΓΕΣ

[1] Siavvas, Miltiadis, et al. “Technical Debt as an Indicator of Software Security Risk: A Machine Learning Approach for Software Development Enterprises.” *Enterprise Information Systems*, vol. 16, no. 5, 2020, https://doi.org/10.1080/17517575.2020.1824017.

[2] Tsoukalas, Dimitrios, et al. “TD Classifier.” *Proceedings of the International Conference on Technical Debt*, 2022, https://doi.org/10.1145/3524843.3528094.

[3] Kalouptsoglou, Ilias, et al. “Examining the Capacity of Text Mining and Software Metrics in Vulnerability Prediction.” *Entropy*, vol. 24, no. 5, 2022, p. 651., https://doi.org/10.3390/e24050651.

[4] D. Tsoukalas, A. Chatzigeorgiou, A. Ampatzoglou, N. Mittas and D. Kehagias, "TD Classifier: Automatic Identification of Java Classes with High Technical Debt," 2022 IEEE/ACM International Conference on Technical Debt (TechDebt), Pittsburgh, PA, USA, 2022, pp. 76-80, doi: 10.1145/3524843.3528094.

[5] Siavvas, Miltiadis, et al. “The SDK4ED Platform for Embedded Software Quality Improvement - Preliminary Overview.” *Computational Science and Its Applications – ICCSA 2020*, 2020, pp. 1035–1050., https://doi.org/10.1007/978-3-030-58811-3_73.

[6] Kalouptsoglou, Ilias, et al. “An Empirical Evaluation of the Usefulness of Word Embedding Techniques in Deep Learning-Based Vulnerability Prediction.” *Communications in Computer and Information Science*, 2022, pp. 23–37., https://doi.org/10.1007/978-3-031-09357-9_3.

[7] Siavvas, Miltiadis, et al. “A Hierarchical Model for Quantifying Software Security Based on Static Analysis Alerts and Software Metrics.” *Software Quality Journal*, vol. 29, no. 2, 2021, pp. 431–507., https://doi.org/10.1007/s11219-021-09555-0.

[8] “Home · Wiki · VM4SEC Wiki / Wiki Home · GITLAB.” *GitLab*, https://gitlab.seis.iti.gr/vm4sec-wiki/wiki-home/-/wikis/home.

[9] Kruchten, Philippe, et al. *Managing Technical Debt: Reducing Friction in Software Development*. 1st ed., Software Engineering Institute/Carnegie Mellon, 2019.

[10] D. Tsoukalas et al., "Machine Learning for Technical Debt Identification," in IEEE Transactions on Software Engineering, vol. 48, no. 12, pp. 4892-4906, 1 Dec. 2022, doi: 10.1109/TSE.2021.3129355.

[11] Li, Zengyang, et al. “A Systematic Mapping Study on Technical Debt and Its Management.” *Journal of Systems and Software*, vol. 101, 2015, pp. 193–220., https://doi.org/10.1016/j.jss.2014.12.027.

[12] Thompson, Herbert H., and Scott G. Chase. *The Software Vulnerability Guide*. 1st ed., Charles River Media, 2005.

[13] Arvanitou, Elvira-Maria, et al. “Monitoring Technical Debt in an Industrial Setting.” *Proceedings of the Evaluation and Assessment on Software Engineering*, 2019, https://doi.org/10.1145/3319008.3319019.

[14] Angeliki-Agathi Tsintzira, Areti Ampatzoglou, Oliviu Matei, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, & Robert Heb. (2019, May 30). “Technical Debt Quantification through Metrics: An Industrial Validation”. https://doi.org/10.5281/zenodo.3381247.

[15] Kouros, Panagiotis, et al. “JCaliper.” *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, https://doi.org/10.1145/3297280.3297448.

[16] Charalampidou, Sofia, et al. “Structural Quality Metrics as Indicators of the Long Method Bad Smell: An Empirical Study.” *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2018, https://doi.org/10.1109/seaa.2018.00046.

[17] Skiada, Peggy, et al. “Exploring the Relationship between Software Modularity and Technical Debt.” *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2018, https://doi.org/10.1109/seaa.2018.00072.

[18] Areti Ampatzoglou, Alexander Michailidis, Christos Sarikyriakidis, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, and Paris Avgeriou “A Framework for Managing Interest in Technical Debt: An Industrial Validation”, *2018 International Conference on Technical Debt (TechDEBT)*

[19] Amanatidis T., Mittas N., Chatzigeorgiou A., Ampatzoglou A., and Angelis L “The developer’s dilemma: Factors affecting the Decision to Repay Code Debt”, *2018 International Conference on Technical Debt (TechDEBT)*

[20] Tsantalis, Nikolaos, et al. “Ten Years of Jdeodorant: Lessons Learned from the Hunt for Smells.” *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2018, https://doi.org/10.1109/saner.2018.8330192.

[21] Digkas, Georgios, et al. “How Do Developers Fix Issues and Pay Back Technical Debt in the Apache Ecosystem?” *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2018, https://doi.org/10.1109/saner.2018.8330205.

[22] Turney, Shaun. “Pearson Correlation Coefficient (R): Guide & Examples.” *Scribbr*, 5 Dec. 2022, https://www.scribbr.com/statistics/pearson-correlation-coefficient/.

[23] “Libguides: SPSS Tutorials: Pearson Correlation.” *Pearson Correlation - SPSS Tutorials - LibGuides at Kent State University*, https://libguides.library.kent.edu/SPSS/PearsonCorr.

[24] “Scipy.stats.pearsonr.” *Scipy.stats.pearsonr - SciPy v1.10.1 Manual*, https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html.

[25] Bhat, Adi. “Spearman Correlation Coefficient: Formula & Calculation.” *QuestionPro*, 15 Jan. 2023, https://www.questionpro.com/blog/spearmans-rank-coefficient-of-correlation/.

[26] Bhandari, Pritha. “Correlation Coefficient: Types, Formulas & Examples.” *Scribbr*, 5 Dec. 2022, https://www.scribbr.com/statistics/correlation-coefficient/.

[27] “Scipy.stats.spearmanr.” *Scipy.stats.spearmanr - SciPy v1.10.1 Manual*, https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html.

[28] Phipson, Belinda, and Gordon K Smyth. “Permutation P-Values Should Never Be Zero: Calculating Exact p-Values When Permutations Are Randomly Drawn.” *Statistical Applications in Genetics and Molecular Biology*, vol. 9, no. 1, 2010, https://doi.org/10.2202/1544-6115.1585.

[29] Wilcox, Rand R. *Applying Contemporary Statistical Techniques*, 1st ed., Acad. Press, Amsterdam, 2010, pp. 237–284.

[30] Cobb, George W. “The Introductory Statistics Course: A Ptolemaic Curriculum?” *Technology Innovations in Statistics Education*, vol. 1, no. 1, 2007, https://doi.org/10.5070/t511000028.

[31] “The Permutation Test.” *Permutation Test: Visual Explanation*, https://www.jwilber.me/permutationtest/.

[32] “Scipy.stats.permutation_test.” *Scipy.stats.permutation_test - SciPy v1.10.1 Manual*, https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.permutation_test.html.

[33] “Technical Debt Toolbox Description · Wiki · SDK4ED Wiki / Wiki Home.” *GitLab*, https://gitlab.seis.iti.gr/sdk4ed-wiki/wiki-home/-/wikis/technical-debt-toolbox-description.

[34] “Technical Debt - 6 Things a Business Should Know about It.” *MasterBorn*, https://www.masterborn.com/blog/technical-debt-6-things-business-should-know.