ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΕΧΝΙΚΕΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΣΤΟΝ ΕΛΕΓΧΟ ΟΡΘΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΛΟΓΙΣΜΙΚΟΥ

Παναγιώτης Κεπεσίδης

Επιβλέπων: Σταύρος Π. Αδάμ

Επίκουρος Καθηγητής

Άρτα, Σεπτέμβριος 2022

MACHINE LEARNING TECHNIQUES IN SOFTWARE TEST AND VERIFICATION

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, 27/09/2022

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής
   Σταύρος Π. Αδάμ,

2. Μέλος επιτροπής
   Λιαροκάπης Δημήτριος,

3. Μέλος επιτροπής
   Στεργίου Ελευθέριος

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στην βιβλιογραφία.

Κεπεσίδης, Παναγιώτης

Υπογραφή

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα αρχικά να ευχαριστήσω όλους τους καθηγητές για το έργο τους όλα αυτά τα χρόνια και ιδιαίτερα τον επιβλέποντα καθηγητή μου, Σταύρο Αδάμ για την συνεργασία, την υποστήριξη αλλά και για την ευκαιρία που μου επέτρεψε να συγγράψω την συγκεκριμένη πτυχιακή. Επίσης θα ήθελα να ευχαριστήσω τους γονείς μου και τα αδέρφια μου για την συνεχή υποστήριξη και βοήθεια που μου παρείχαν σε όλες τις στιγμές αυτού του μοναδικού ταξιδιού.

ΠΕΡΙΛΗΨΗ

Ο έλεγχος ορθής λειτουργίας λογισμικού είναι μια διαδικασία απαραίτητη κατά την διάρκεια ανάπτυξης λογισμικού. Ο έλεγχος όμως δημιουργεί ένα μεγάλο πρόβλημα διαχείρισης πόρων καθώς κοστίζει σε χρόνο και κόστος. Για την αντιμετώπιση αυτού του προβλήματος εξετάζεται η αυτοματοποιημένη δημιουργία δοκιμών βασιζόμενη στα δεδομένα που προκύπτουν από τους χρήστες του λογισμικού και με την βοήθεια τεχνητής νοημοσύνης. Αλγόριθμοι όπως ο K-means και μοντέλα όπως ένα Hidden Markov Model από την μηχανική μάθηση χρησιμοποιούνται για την ανάλυση και επεξεργασία των δεδομένων από τους χρήστες για την αποτελεσματικότερη δημιουργία δοκιμών. Η παρούσα πτυχιακή αποτελεί μελέτη της διαδικασίας ελέγχου λογισμικού, της τεχνητής νοημοσύνης και των αλγορίθμων που προαναφέρθηκαν. Τέλος, θα γίνει επεξεργασία των δεδομένων από τους χρήστες με την βοήθεια μηχανικής μάθησης και σύγκριση με αποτελέσματα πριν την εφαρμογή της, όπως ακόμα και εκτίμηση ομαδοποίησης των δεδομένων με βάση τα αποτελέσματα των αυτοματοποιημένων δοκιμών τους.

Λέξεις-κλειδιά: HMM, K-means, Μηχανική Μάθηση, Τεχνητή Νοημοσύνη

ABSTRACT

Software testing is an essential process during software development. But the testing creates a big resource management problem as it costs in time and money. To address this problem, automated test generation is being considered, based on data obtained from software users and with the help of artificial intelligence. Algorithms such as K-means and models such as a Hidden Markov Model from machine learning are used to analyze and process data from users to create tests more efficiently. This thesis is a study of the software testing process, the artificial intelligence and the above-mentioned algorithms. Finally, the data from the users will be processed with the help of machine learning and the results will be compared with the results before its application, and an evaluation of clustering the data based on the results of their automated tests will be implemented as well.

Keywords: HMM, K-means, Machine Learning, Artificial Intelligence

Πίνακας περιεχομένων

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Κεφάλαιο 1 Εισαγωγή

1.1 Στόχος της πτυχιακής

Ο στόχος της πτυχιακής είναι η ανάδειξη των προβλημάτων που προκύπτουν από τον έλεγχο ορθής λειτουργίας λογισμικού, την αναφορά σε μεθόδους και στρατηγικές που χρησιμοποιούνται για τον έλεγχο, την αναφορά της τεχνητής νοημοσύνης και της μηχανικής μάθησης, όπως επίσης και την περιγραφή αλγορίθμων μηχανικής μάθησης όπως του K-means, του Hidden Markov Model, του Bag of Words και του Word2Vec. Τέλος γίνεται η παρουσίαση της υλοποίησης που πραγματοποιήθηκε για την εφαρμογή των αλγορίθμων αυτών στον έλεγχο και τις επιδόσεις που προκύπτουν στην δημιουργία αυτοματοποιημένων test.

1.2 Δομή

Στο πρώτο κεφάλαιο αναφέρεται ο στόχος της πτυχιακής και περιγράφεται η δομή της με στόχο την προετοιμασία του αναγνώστη για τις θεματικές ενότητες που θα ακολουθήσουν.

Το δεύτερο κεφάλαιο αφορά την θεωρία του ελέγχου ορθής λειτουργίας του λογισμικού. Περιγράφονται η έννοια του ελέγχου λογισμικού, οι στόχοι τους οποίους έχει και τις γενικές αρχές του. Επίσης αναφέρονται οι τεχνικές με τις οποίες γίνεται ο έλεγχος, καθώς ακόμα και τις στρατηγικές που εφαρμόζονται.

Το τρίτο κεφάλαιο αναφέρει την θεωρία της Τεχνητής Νοημοσύνης. Αναλύεται η έννοια της τεχνητής νοημοσύνης, οι στόχοι της και οι υποκατηγορίες της. Γίνεται και αναλυτική αναφορά στην θεωρία της μηχανικής μάθησης.

Στο τέταρτο κεφάλαιο περιγράφονται οι αλγόριθμοι μηχανικής μάθησης HMM και k-means που χρησιμοποιήθηκαν, αλλά και η θεωρία αλγορίθμων επεξεργασίας κειμένου Bag of Words και Word2Vec.

Το πέμπτο κεφάλαιο αφορά την εφαρμογή όλων των εννοιών που περιγράφονται στο τέταρτο κεφάλαιο και που χρησιμοποιήθηκαν. Αναφέρονται επίσης τα τεχνικά χαρακτηριστικά της εφαρμογής τους όπως το περιβάλλον στο οποίο υλοποιήθηκαν, το hardware στο οποίο εκτελέστηκαν αλλά και τα αποτελέσματα που παράχθηκαν.

Στο έκτο κεφάλαιο γίνεται μια ανασκόπηση της πτυχιακής και αναφορά συμπερασμάτων από όλη την υλοποίηση.

Κεφάλαιο 2 Έλεγχος Λογισμικού

2.1 Ο έλεγχος και η αναγκαιότητά του

Ο έλεγχος λογισμικού είναι μια διαδικασία μέσα από την οποία γίνεται η αξιολόγηση ενός λογισμικού ώστε να βρεθούν λάθη και δυσλειτουργίες. Αποσκοπεί στην διαπίστωση της σωστής λειτουργίας του με βάση τον σχεδιασμό του, αλλά και με τις απαιτήσεις που ορίζει ο χρήστης. Με τον έλεγχο της ορθής λειτουργίας του λογισμικού εξασφαλίζονται επίσης η αξιοπιστία του, η ασφάλειά του, η αποδοτικότητά του και η χρηστικότητά του. Χαρακτηρίζεται ως ένα αναπόσπαστο και πολύ σημαντικό κομμάτι στην παραγωγή λογισμικού και πολλές φορές απαιτεί το 40-50% του χρόνου και του κόστους ενός οργανισμού δημιουργώντας έτσι ένα μεγάλο πρόβλημα διαχείρισης πόρων. (Abhijit, 2012; Bertolino, 2007; Muhammad Abid Jamil, 2016; Syed, 2016)

2.2 Οι Στόχοι του Ελέγχου

Οι στόχοι του ελέγχου μπορούν να αναφερθούν συνοπτικά στους εξής (Syed, 2016) :

i) Στην ανίχνευση σφαλμάτων,
ii) Στην πρόληψη νέων σφαλμάτων,
iii) Στην αναβάθμιση της ποιότητας,
iv) Στην επαλήθευση για το εάν πληρούνται οι απαιτήσεις, και
v) Στην επικύρωση, ώστε να παραχθεί λογισμικό βάσει των απαιτήσεων του χρήστη.

2.3 Αρχές του Ελέγχου

Η διαδικασία του ελέγχου είναι αναγκαίο να βασίζεται πάνω σε βασικές αρχές ώστε να γίνεται σωστά και επιτυχημένα. Οι βασικές αρχές που πρέπει να ληφθούν υπόψη είναι οι εξής (Abhijit, 2012):

i) Πρέπει να γίνεται προσπάθεια αποτυχίας του λογισμικού

ii) Πρέπει να γίνονται έλεγχοι από τα αρχικά στάδια δημιουργίας του λογισμικού
iii) Πρέπει οι έλεγχοι να είναι διαφορετικοί και κατάλληλοι σε κάθε χρονική στιγμή
iv) Πρέπει να υπάρχει ένα πλάνο με λεπτομέρειες όπου θα βασιστούν οι έλεγχοι
v) Πρέπει τα αποτελέσματα δοκιμών να είναι σαφή
vi) Πρέπει να δοκιμάζονται έγκυρες αλλά και μη έγκυρες είσοδοι στο σύστημα
vii) Πρέπει οι έλεγχοι να γίνονται από πολλά και διαφορετικά άτομα σε διαφορετικές φάσεις του ελέγχου, και τέλος
viii) Πρέπει να καθορίζεται το τέλος της διαδικασίας ελέγχου με βάση κάποιο όριο ρίσκου, εάν υπάρχει όριο.

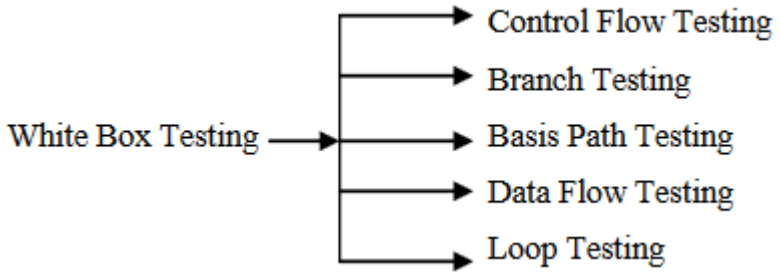
2.4 Μέθοδοι Ελέγχου

Ο έλεγχος ενός συστήματος κατηγοριοποιείται σε τρείς μεθόδους, την μέθοδο White Box Testing, την μέθοδο Black Box Testing και την μέθοδο Grey Box Testing.

Πιο αναλυτικά η μέθοδος WhiteBoxTesting είναι μια μέθοδος στην οποία η εσωτερική δομή, αρχιτεκτονική και η σχεδίαση του κώδικα μπαίνει σε διαδικασία δοκιμής . Η δοκιμή αυτή είναι απαραίτητη για να ανιχνευθούν σφάλματα κατά την είσοδο και έξοδο του προγράμματος και έτσι να βελτιωθούν η ασφάλεια καθώς και η χρηστικότητα. Τα σφάλματα για τα οποία ελέγχεται ο κώδικας , μεταξύ άλλων είναι τα εξής:

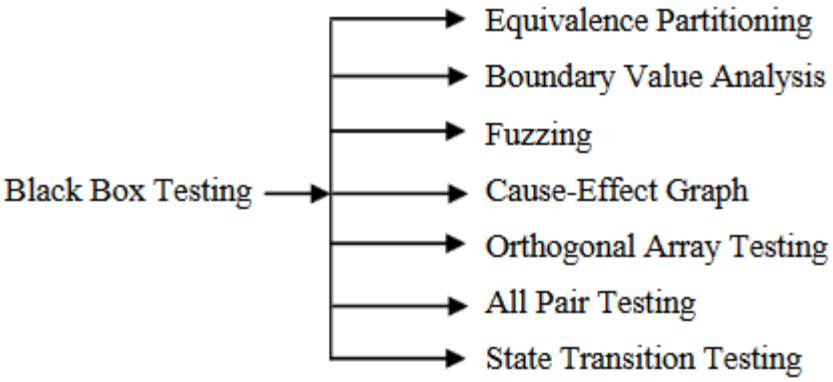
- Κενά ασφαλείας
- Αναμενόμενη και μη αναμενόμενη Έξοδος
- Η λειτουργικότητα των βρόχων και των περιπτώσεων που ελέγχουν
- Την ‘αντίδραση’ του κώδικα σε συγκεκριμένες εισόδους κ.ο.κ

Όλη αυτή η διαδικασία είναι ορατή στον tester για αυτό και η διαδικασία αυτή επονομάστηκε με τον όρο «White Box Testing», ενώ μπορεί να συναντηθεί στη βιβλιογραφία και με τους δόκιμους όρους όπως Clear Box Testing , Open Box Testing κ.ο.κ.



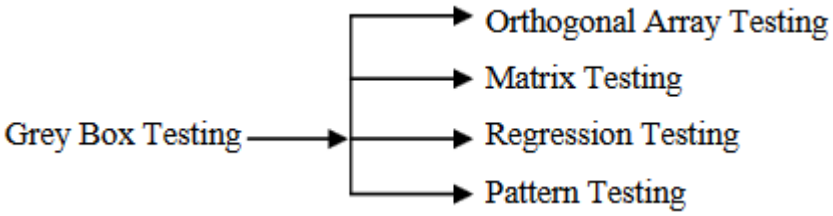
Η μέθοδος του BlackBoxTesting είναι μία πιο απλή μέθοδος και πιο διαδεδομένη. Αυτός που πραγματοποιεί την δοκιμή, ο tester, δεν χρειάζεται να έχει γνώση ή ακόμα και πρόσβαση στον κώδικα που θα ελέγξει, χρειάζεται όμως να έχει γνώση της αρχιτεκτονικής του. Στη μέθοδο αυτή, ο προγραμματιστής και ο tester λειτουργούν ανεξάρτητα ο ένας από τον άλλον.

Στο BlackBox Testing ελέγχονται μόνο αν η είσοδος γίνεται δεκτή στο σύστημα και αν η έξοδος που παράγεται είναι η επιθυμητή. Δοκιμάζονται όλες οι λειτουργικές δυνατότητες του λογισμικού ώστε να διαπιστωθεί αν η λειτουργικότητα του συστήματος συμφωνεί με τις απαιτήσεις του χρήστη. Και αυτό αποτελεί ένα από τα βασικότερα πλεονεκτήματα αυτή της μεθόδου, ό,τι στο τέλος της διαδικασίας θα αποκαλυφθούν τυχόν αστοχίες στις προδιαγραφές απαιτήσεων. (Dondeti, 2012)



Στην μέθοδο Grey Box Testing συνδυάζονται τα προτερήματα των μεθόδων White Box Testing και της Black Box Testing. Ο Tester δοκιμάζει ένα κομμάτι λογισμικού έχοντας όμως γνώση και για την εσωτερική λογική. Ελέγχεται αν το λογισμικό συναντά τις προδιαγραφές που

έχουν οριστεί. Μπορεί να εφαρμοστεί σε real-time και δεν εξαρτάται από κάποια πλατφόρμα ή από κάποια γλώσσα προγραμματισμού. (Abhijit, 2012; Muhammad Abid Jamil, 2016; Syed, 2016)



2.4.1 Σύγκριση Μεθόδων Ελέγχου

Στην παρούσα ενότητα, θα γίνει μια σύγκριση των προαναφερόμενων τριών μεθόδων αναφέροντας τα πλεονεκτήματα τους καθώς και τα μειονεκτήματα τους.

White Box Testing:

Πλεονεκτήματα:

- Ανιχνεύει οποιοδήποτε πιθανό λάθος απαλείφοντας τις γραμμές κώδικα που μπορεί να το δημιουργούν
- Μέγιστη κάλυψη κατά τη διάρκεια του σεναρίου ελέγχου

Μειονεκτήματα:

- Είναι ιδιαίτερη ακριβή επιλογή αφού ακόμα και ο tester πρέπει να έχει γνώσεις προγραμματισμού για την εκτελέσει
- Αρκετά σημεία του κώδικα δεν θα δοκιμαστούν ποτέ

Black Box Testing:

Πλεονεκτήματα

- Αποδοτικό σε μεγάλα τμήματα κώδικα
- Ο tester δεν χρειάζεται να έχει γνώσεις προγραμματισμού
- Γρήγορη εκτέλεση δοκιμών

Μειονεκτήματα

- Μόνο ένα συγκεκριμένο αριθμό σεναρίων δοκιμάζονται
- Χωρίς συγκεκριμένες προδιαγραφές δεν μπορούν να σχεδιαστούν σενάρια δοκιμών
- Μη αποδοτική δοκιμή

Grey Box Testing:

Πλεονεκτήματα:

- Συνδυάζει τα πλεονεκτήματα των δύο παραπάνω μεθόδων ελέγχου (Black Box Testing και White Box Testing).
- Η όλη διαδικασία δοκιμής του λογισμικού επικεντρώνεται στη διεπαφή παρά στον κώδικα και στη λειτουργικότητα του.
- Ο tester μπορεί να σχεδιάσει και αυτός τα δικά του σενάρια δοκιμών.
- Η όλη δοκιμή γίνεται με βάση τις απαιτήσεις του χρήστη.

Μειονεκτήματα:

- Οι δοκιμές είναι περιορισμένες και δεν έχουν πρόσβαση στον κώδικα.
- Αρκετά κομμάτια κώδικα παραμένουν ‘κρυφά’ από τη διαδικασία δοκιμής.

Εδώ αξίζει να αναφερθεί ότι εάν ο προγραμματιστής έχει ήδη τρέξει ένα σενάρια ελέγχου τότε οποιαδήποτε άλλη δοκιμή και έλεγχος εκ μέρους τρίτων είναι περιττό. (Mohd. Ehmer Khan, 2012)

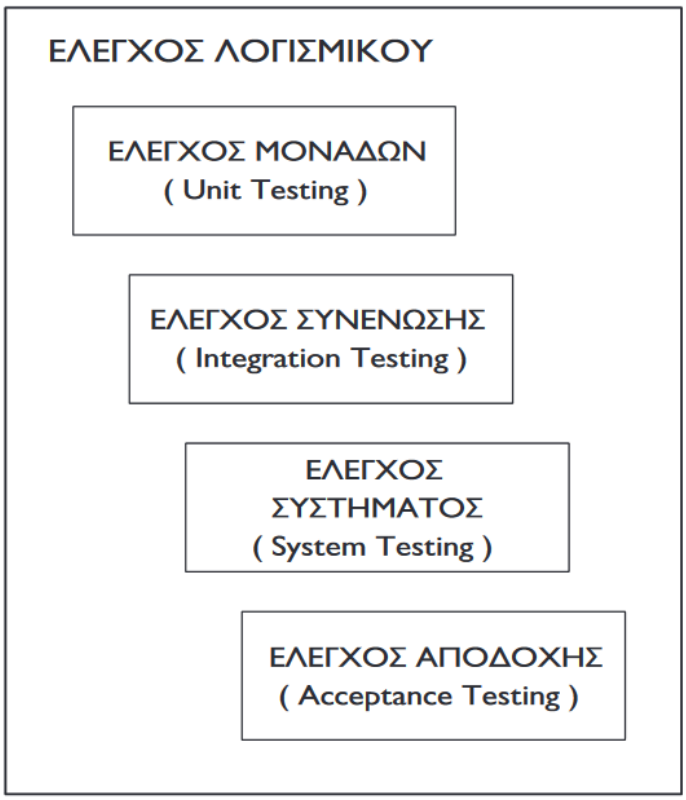
2.5 Στρατηγικές Ελέγχου

Ένα από τα κυριότερα στάδια κατά την ανάπτυξη ενός λογισμικού είναι ο έλεγχος της ορθής λειτουργίας του και κατ' επέκταση ο έλεγχος της λειτουργίας του με βάση τις απαιτήσεις (requirements) που έχουν καθοριστεί σε επίπεδο χρήστη αλλά όχι μόνο. Ο χρήστης είναι αυτός που κρίνει τη λειτουργικότητα με βάση τις απαιτήσεις του κάθε λογισμικού αλλά από εκεί και πέρα πρέπει να ελεγχθούν και άλλες παράμετροι όπως π.χ. η ασφάλεια. Ένα βασικό κομμάτι του ελέγχου είναι η εξασφάλιση αυτής της ασφάλειας. Όπως και της δυνατότητας του λογισμικού να ανταποκρίνεται σε ακραίες καταστάσεις όπως είναι η μεγάλη αύξηση ροής δεδομένων ή μια αναπάντεχη διακοπή ρεύματος ή κατάρρευσης (απροσδόκητο κλείσιμο) του λογισμικού. Για αυτό και ο έλεγχος λογισμικού καταλαμβάνει το 25% με 50% του συνολικού κόστους στη διαδικασία παραγωγής ενός λογισμικού και αυτός είναι και ένας λόγος παραπάνω ο έλεγχος να γίνεται με βάση τεχνικές και στρατηγικές ελέγχων.

Ο έλεγχος, λοιπόν, είναι η διαδικασία κατά την οποία εξετάζεται το λογισμικό με το συνδυασμό διάφορων τεχνικών και στρατηγικών για την εύρεση και αντιμετώπιση των σφαλμάτων. Ο όρος «στρατηγική» προσδιορίζει το συνδυασμό των πόρων και των τεχνικών για την ικανοποίηση των στόχων.

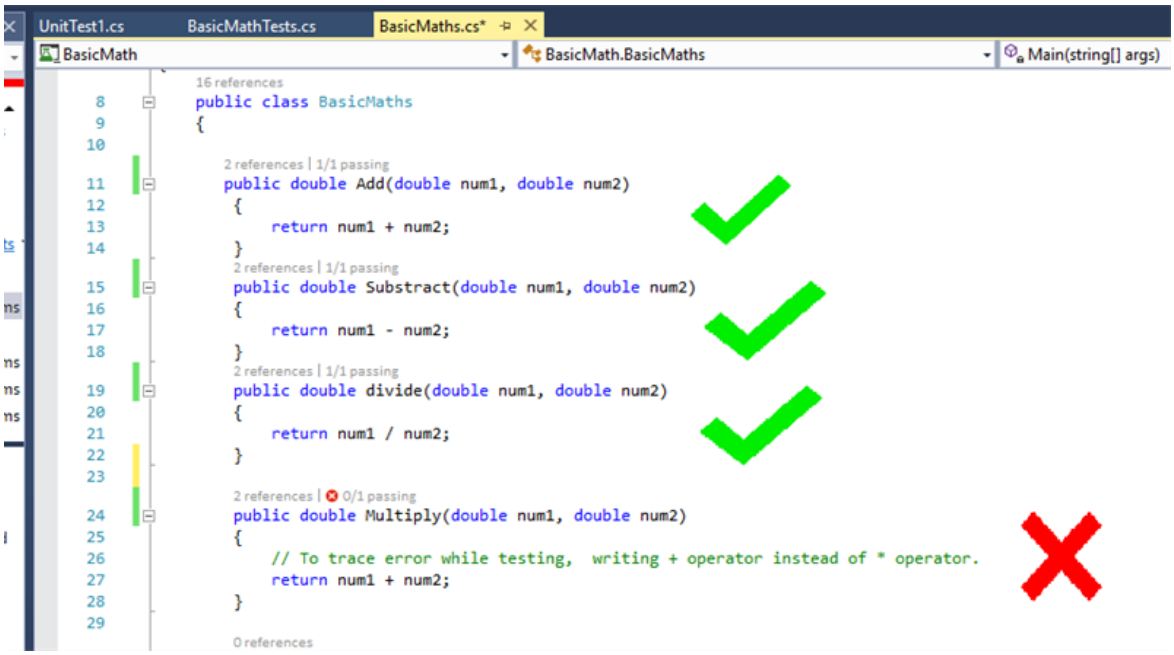
Σε αυτήν την ενότητα , θα γίνει αναφορά για τις στρατηγικές που ακολουθούνται για τις δοκιμές των λογισμικών, συνδυάζοντας και χρησιμοποιώντας τις τεχνικές που προαναφέρθηκαν.

Οι στρατηγικές, λοιπόν, πάνω στις οποίες βασίζονται οι δοκιμές του λογισμικού είναι 4 και αναλύονται στην συνέχεια, το Unit Testing, το Integration Testing, το System Testing και το Acceptance Testing.



Εικόνα 1 Έλεγχος Λογισμικού και Στρατηγικές Ελέγχου

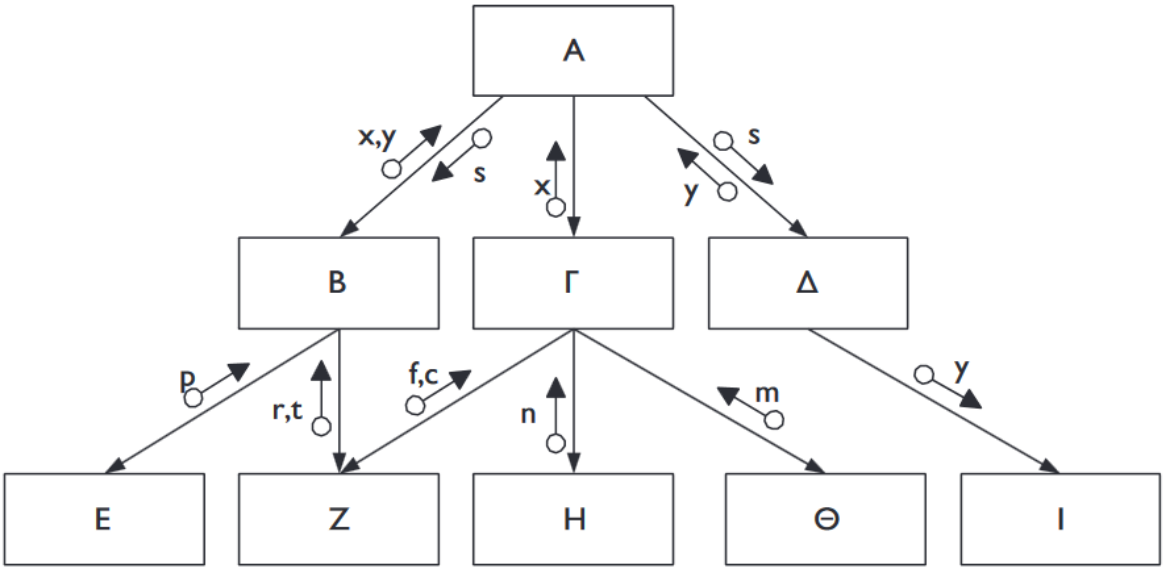
Στο Unit Testing (Έλεγχος Μονάδων) ελέγχονται συγκεκριμένες γραμμές κώδικα την φορά, αντιμετωπίζοντας τις γραμμές αυτές ως ξεχωριστές μονάδες και αυτή η στρατηγική βασίζεται στην μέθοδο του WhiteBoxTesting, δηλαδή εστιάζει στην δομή και την λογική του κώδικα. Ουσιαστικά, δοκιμάζει κάθε μονάδα για να διαπιστωθεί αν πληροί τις προδιαγραφές που έχουν οριστεί σε προηγούμενα βήματα. Ένα παράδειγμα τέτοιου ελέγχου, παρουσιάζεται στην παρακάτω εικόνα:



Εικόνα 2 Παράδειγμα Unit Testing σε C++

Το Integration Testing (Έλεγχος συνένωσης) λαμβάνει χώρα μετά το Unit Testing και πριν το System Testing(Έλεγχος Συστήματος). Στοχεύει στην εύρεση σφαλμάτων αλλά και λαθών δομής κατά την εκτέλεση μερών του κώδικα . Εδώ αξίζει να αναφερθεί ότι για αυτήν την στρατηγική ελέγχου μπορεί να χρησιμοποιηθεί και η στρατηγική του άσπρου κουτιού (White Box Testing) καθώς και του μαύρου κουτιού (Black Box Testing). Σε αυτόν τον έλεγχο συναντώνται και δύο τεχνικές , η τεχνική από πάνω προς τα κάτω (top down incremental) καθώς και η τεχνική από κάτω προς τα πάνω (bottom-up incremental). Για να γίνουν κατανοητές οι τεχνικές αυτές, παρουσιάζεται το εξής παράδειγμα:

Έστω η παρακάτω δομής ενός προγράμματος:



Εικόνα 3 Διάγραμμα ροής προγράμματος

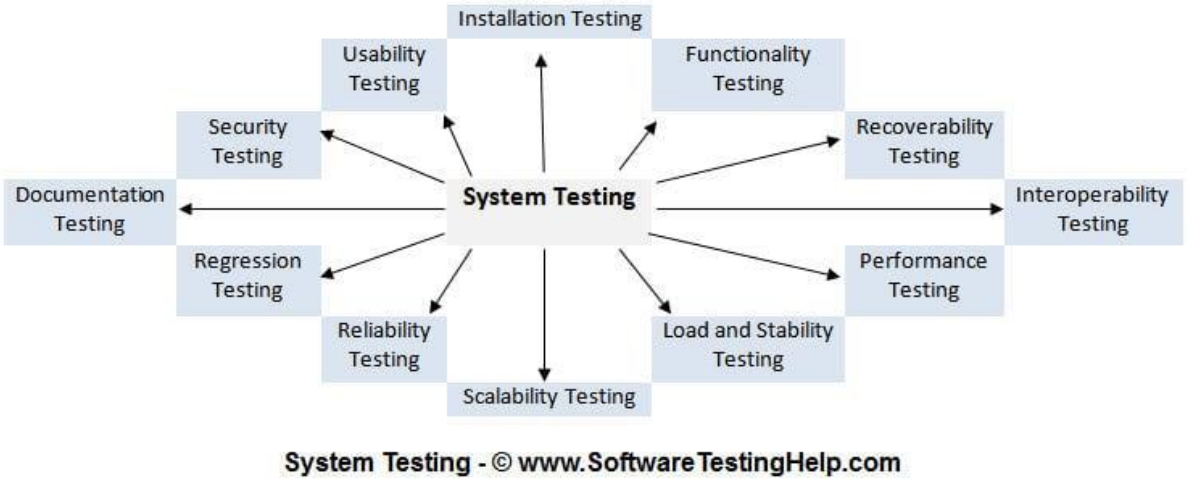
Μια σειρά εκτέλεσης με την τεχνική από πάνω προς τα κάτω (top down) είναι η εξής:

Βήμα	Υποσύστημα που ελέγχεται
I	A
2	AB
3	ΑΒΓ
4	ΑΒΓΔ
5	ΑΒΓΔΕ
6	ΑΒΓΔΕΖ
7	ΑΒΓΔΕΖΗ
8	ΑΒΓΔΕΖΗΘ
9	ΑΒΓΔΕΖΗΘΙ

Εικόνα 4 Τεχνική Top Down για Integration Testing

Στο System Testing (Έλεγχος συστημάτων) ελέγχεται ένα ολοκληρωμένο σύστημα για το αν τηρεί τις προκαθορισμένες απαιτήσεις. Είναι μια σειρά από ελέγχους και εφαρμόζει την μέθοδο BlackBox Testing, δηλαδή εστιάζονται οι έλεγχοι στις εισόδους και τις εξόδους του συστήματος ώστε να επαληθευτεί ότι λειτουργεί πλήρως και με βάση τις προδιαγραφές. Εδώ αξίζει να αναφερθεί , ότι πέραν των λειτουργικών απαιτήσεων που ελέγχονται , ταυτόχρονα εκτελούνται και επιπλέον , μη λειτουργικών απαιτήσεων, έλεγχοι όπως:

- Performance Testing (Έλεγχος Επίδοσης): Το λογισμικό ‘δοκιμάζεται με είσοδο δεδομένων με απώτερο σκοπό τη βελτιστοποίηση των επιδόσεων
- Stress Testing (Έλεγχοι πίεσης): Εδώ το λογισμικό δοκιμάζεται σε συνθήκες που δεν είναι συνήθεις και κρίνεται η ανταπόκριση του.
- Volume Testing (Έλεγχοι όγκου δεδομένων): Όπως αναφέρει και το όνομα του ελέγχου, το λογισμικό ελέγχεται για μεγάλα φορτία δεδομένων και πώς ανταποκρίνεται σε αυτά.
- Security Testing (Έλεγχοι ασφαλείας)
- Recovery Testing (Έλεγχοι ανάκτησης): ο έλεγχος αυτός εστιάζει στη δυνατότητα της ανάκτησης των δεδομένων σε περίπτωση ξαφνικής κατάρρευσης. (Βασίλειος, 2015)



Εικόνα 5 System Testing

Τέλος, στο Acceptance Testing (έλεγχος αποδοχής) ελέγχεται αν ο χρήστης είναι ευχαριστημένος με το σύστημα. Ο έλεγχος γίνεται από τον χρήστη που επιβεβαιώνει το γεγονός ότι το σύστημα λειτουργεί με βάση τις προϋποθέσεις που είχε θέσει ο ίδιος. Σε αυτήν την στρατηγική ακολουθείται η μέθοδος του Black BoxTesting καθώς ο χρήστης δείχνει ενδιαφέρον για τις εισόδους και τις παραγόμενες εξόδους αλλά και για την λειτουργικότητα συνολικά του συστήματος. (Abhijit, 2012; Syed, 2016)

Example #2	
Purpose	User able to log in with user ID and password
Preconditions	User has signed up. User has not logged in
Input	Correct user ID Correct password
Process (Actions & Triggers)	1. Enter user ID 2. Enter password 3. Click 'Login' button
Output	User logs in and sees the dashboard

Εικόνα 6 Παράδειγμα Acceptance Testing

Κεφάλαιο 3 Τεχνητή Νοημοσύνη και Μηχανική Μάθηση

3.1 Η Τεχνητή Νοημοσύνη

Η Τεχνητή Νοημοσύνη είναι ένας κλάδος της επιστήμης των υπολογιστών που έχει ως στόχο την δημιουργία υπολογιστικών συστημάτων που μπορούν να μιμούνται την ανθρώπινη γνωστική λειτουργία. Ασχολείται δηλαδή με την αυτοματοποίηση της ευφυούς συμπεριφοράς.

Με την μίμηση της ανθρώπινης συμπεριφοράς η τεχνητή νοημοσύνη στοχεύει στην υιοθέτηση κάποιων χαρακτηριστικών ευφυΐας όπως η κατανόηση γλώσσας, η δυνατότητα εκμάθησης, αιτιολογίας και λύσης προβλημάτων. Για να επιτευχθεί αυτός ο στόχος, η δημιουργία ευφυών υπολογιστικών συστημάτων βασίζεται πάνω στις τεχνικές αναπαράστασης της γνώσης, την δυνατότητα της μάθησης, στον καθορισμό των κανόνων βάση των οποίων θα καθοδηγείται η εκμάθηση και η αναζήτηση για καλύτερες λύσεις ή καλύτερα βάρη στην περίπτωση εμφάνισης νευρωνικών δικτύων.

Η Τεχνητή Νοημοσύνη χωρίζεται σε κατευθύνσεις με βάση την εξειδίκευσή τους αλλά και τις τεχνικές που εφαρμόζονται. Οι κατευθύνσεις είναι οι εξής:

- Επεξεργασία Ομιλίας, όπου στόχος είναι η κατανόηση και η παραγωγή του προφορικού λόγου.
- Επεξεργασία Φυσικής Γλώσσας, όπου στοχεύει στην κατανόηση και στην παραγωγή του γραπτού λόγου.
- Σχεδιασμός-Προγραμματισμός, όπου στοχεύει στην επιλογή των ορθών κινήσεων ενός ρομποτικού συστήματος.
- Έμπειρα Συστήματα, όπου βασίζονται στην μίμηση ενός ειδικού στην λήψη αποφάσεων.
- Ασαφή Συστήματα, είναι συστήματα που λειτουργούν με βάση την ασαφή λογική.

- Εξελικτικά Μοντέλα, είναι μοντέλα που βασίζονται στον τρόπο που λειτουργεί ο ανθρώπινος εγκέφαλος αλλά και στον τρόπο με τον οποίο ο άνθρωπος εξελίσσεται
- Μηχανική Όραση και Ρομποτική, όπου στοχεύει στην αναγνώριση αντικειμένων και εικόνων αλλά και στην αυτόνομη εξερεύνηση.
- Μηχανική Μάθηση, όπου στοχεύει στην δημιουργία συστημάτων που μπορούν και μαθαίνουν μόνα τους από τα δεδομένα μετά από κάποια εκπαίδευση.

3.2 Η Μηχανική Μάθηση

Η Μηχανική Μάθηση είναι ένας τομέας που επικεντρώνεται στην κατασκευή συστημάτων που εκτελούν διεργασίες τεχνητής νοημοσύνης. Οι διεργασίες όπως είναι η αναγνώριση, η πρόβλεψη, η διάγνωση, ο προγραμματισμός αλλά και ο έλεγχος ρομπότ είναι οι πιο χαρακτηριστικές. Τα συστήματα αυτά κατασκευάζονται με τέτοιο τρόπο ώστε να μπορούν να εξελίσσουν τον εαυτό τους με βάση προσδιορισμένων αρχικών δομών αλλά και με βάση την εμπειρία που αποκτούν. Έτσι, με την εμπειρία τους και με τις δομές που έχουν οριστεί, τα συστήματα είναι ικανά να προσαρμόζονται και να μαθαίνουν χωρίς να υπάρχει ανθρώπινη παρέμβαση.

Ο τρόπος με τον οποίο θα μάθουν και θα προσαρμοστεί το κάθε σύστημα με Μηχανική Μάθηση διαχωρίζεται σε δύο σημαντικές μεθόδους, στη Supervised Learning ή Επιβλεπόμενη Μάθηση και στη Unsupervised Learning ή Μη επιβλεπόμενη Μάθηση.

Στην περίπτωση του Supervised Learning το σύστημα πρώτα εκπαιδεύεται. Δίνονται δεδομένα που ήδη ανήκουν σε κλάσεις ώστε το σύστημα να μάθει να ξεχωρίζει τις κλάσεις με βάση κάποια χαρακτηριστικά των δεδομένων και να δημιουργήσει ένα μοντέλο. Με αυτόν τον τρόπο, όταν δοθούν δεδομένα που δεν έχουν κατηγοριοποιηθεί σε κάποια κλάση, το σύστημα θα είναι σε θέση να το κατηγοριοποιήσει με την εμπειρία που θα έχει το μοντέλο που θα έχει δημιουργηθεί. Χρησιμοποιείται κυρίως σε προβλήματα που απαιτούν:

- Classification (Ταξινόμηση)
- Prediction (Πρόγνωση)

- Interpretation (Διερμηνεία)

Η μέθοδος Supervised Learning, μπορεί να συνοψιστεί στο εξής μαθηματικό μοντέλο δεδομένων μεταβλητών εισόδου X και μεταβλητών εξόδου Y, με το οποίο ο αλγόριθμος 'μαθαίνει' τον τρόπο να κατηγοριοποιήσει τις X στις Y μέσω μιας άγνωστης συνάρτησης f:

$$Y = f(x)$$

Ο στόχος είναι να βρεθεί η συνάρτηση πρόγνωσης h(x) που για κάθε γνωστή είσοδο X να είναι κοντά στο f(x).

Η συνάρτηση h(x) ονομάζεται συνάρτηση πρόγνωσης και ισχύουν τα εξής για αυτή:

1) Κάθε είσοδος χαρακτηρίζεται ως στιγμιότυπο(instance) και δημιουργεί ένα σύνολο στιγμιότυπων
2) Κάθε είσοδος περιγράφεται με βάση γνωρίσματα (attributes)
3) Οι είσοδοι που θεωρούνται είναι δεδομένα θεωρούνται σύνολο εκπαίδευσης
4) Συνάρτηση στόχου (goal function) είναι μια συνάρτηση που απεικονίζει μια είσοδο από το σύνολο εκπαίδευσης με την γνωστή της έξοδο.
5) Η τιμή της συνάρτησης στόχου ονομάζεται μεταβλητή στόχου (goal variable)
6) Στην επιβλεπόμενη μάθηση, η συμπεριφορά της συνάρτησης πρόγνωσης βελτιώνεται με τη βοήθεια μιας συνάρτησης λάθους (error function) που εντοπίζει την απόκλιση της εξόδου της μεταβλητής στόχου από την επιθυμητή έξοδο. (Γεωργούλη, 2015)

Γνωστοί αλγόριθμοι επιβλεπόμενης επαγωγικής μάθησης – δηλαδή μέσω της απαγωγής, της ικανότητας να εξάγονται συμπεράσματα με βάση κάποιες παρατηρήσεις και ιδιότητες - είναι:

- Decision Trees (Δέντρα απόφασης)
- Explanation – Based Learning (Μάθηση βασισμένη σε επεξηγήσεις)
- Case – Based Learning (Μάθηση βασισμένη σε περιπτώσεις)
- Backpropagation Neural Networks

- Μάθηση κατά Bayes – Μάθηση μέσω στατιστικών Μεθόδων
- Boosting ή αλλιώς μάθηση από ενδυνάμωση (Γεωργούλη, 2015)

Από την άλλη, στο Unsupervised Learning δεν υπάρχει εκπαίδευση με έτοιμα κατηγοριοποιημένα δεδομένα αλλά ούτε και εκπαιδευμένο μοντέλο. Εισάγονται δεδομένα στο σύστημα και με βάση τα χαρακτηριστικά του, το σύστημα δημιουργεί ένα μοντέλο ανακαλύπτοντας μόνο του τις κλάσεις στις οποίες θα τα κατηγοριοποιήσει. Χρησιμοποιείται σε προβλήματα :

- Association Analysis (Ανάλυσης Συσχετισμών): Πρόκειται για μια διαδικασία κατά την οποία διερευνώνται διάφοροι τρόποι που μπορούν τα δεδομένα να συνδεθούν μεταξύ τους. Χαρακτηριστικό παράδειγμα είναι στην ανάλυση δεδομένων συναλλαγών κατά την οποία όταν ένας πελάτης αγοράσει το Α τότε αγοράζει ταυτόχρονα το Β και το Γ. Οι κανόνες συσχέτισης είναι της μορφής {X1,X2…Xn}-> Y και σημαίνει ότι όταν όλα τα προϊόντα X1..Xn αγοραστούν τότε υπάρχει μεγάλη πιθανότητα να βρεθεί και το Y. Για παράδειγμα όποιος αγοράζει καφέ (X1) και ζάχαρη (X2) τότε αγοράζει και αναψυκτικά (Y). Για να αναπαρασταθούν σωστά οι κανόνες συσχετισμών θα πρέπει να ορισθούν και κάποια ποσοτικά μεγέθη όπως η support /coverage (υποστήριξη/κάληψη) που εκφράζει την πιθανότητα να βρεθεί στο καλάθι {X1..Xn,Y} και ισούται με το λόγο των εγγραφών προς το σύνολο των εγγραφών. Ένα άλλο μέγεθος είναι η ακρίβεια (accuracy) που εκφράζει τη πιθανότητα το Y να βρεθεί σε ένα καλάθι που περιέχει τα {X1,X2,..,Xn}. Στην ανάλυση συσχετισμών, συναντάται και ο αλγόριθμος Apriori. (Βλαχάβας Ι, 2020)

Εικόνα 7 Παράδειγμα Ανάλυσης Συσχετισμών Μη Επιβλεπόμενης Μάθησης

- Clustering (Ομαδοποίησης): Πρόκειται για προβλήματα κατηγοριοποίησης κατά τα οποία το μόνο που είναι γνωστό είναι τα δείγματα δεδομένων $\{x_i\}_{i=1}^{N}$. Σκοπός της ομαδοποίησης είναι να δημιουργούνται υποσύνολα από ένα σύνολο παρατηρήσεων τα οποία έχουν μεγάλη πυκνότητα πιθανότητας ή με άλλα λόγια παρουσιάζουν όμοιες παρατηρήσεις με βάση κάποιο κριτήριο. Συνήθως, αυτό το κριτήριο είναι ο υπολογισμός της απόστασης μεταξύ των δεδομένων. Συνηθίζεται να υπολογίζεται η απόσταση Μανχάταν και η Ευκλείδεια απόσταση. Έτσι, λοιπόν, δεδομένου ένα σύνολο δεδομένων D και δύο μεμονωμένα δεδομένα x και y που περιγράφονται από κάποια m χαρακτηριστικά και είναι τα εξής: $(x_1,x_2,\ldots x_m)$ και $(y_1,y_2,..y_m)$, η απόσταση Μανχάταν και η Ευκλείδεια απόσταση παρουσιάζεται από τους εξής τύπους αντίστοιχα:

  - Απόσταση Μανχάταν:

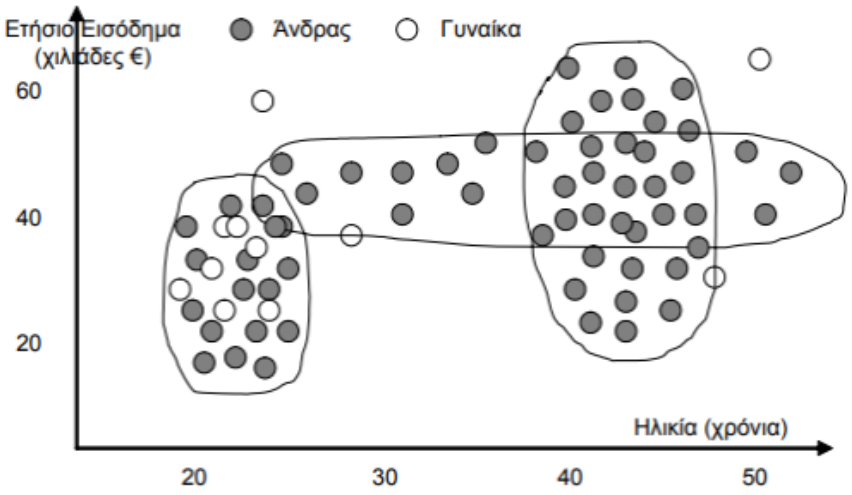
$$d(x,y) = \sum_i |x_i - y_i|$$

- Ευκλείδια απόσταση:

$$d(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$$

Στην ομαδοποίηση συναντώνται και εξής αλγόριθμοι:

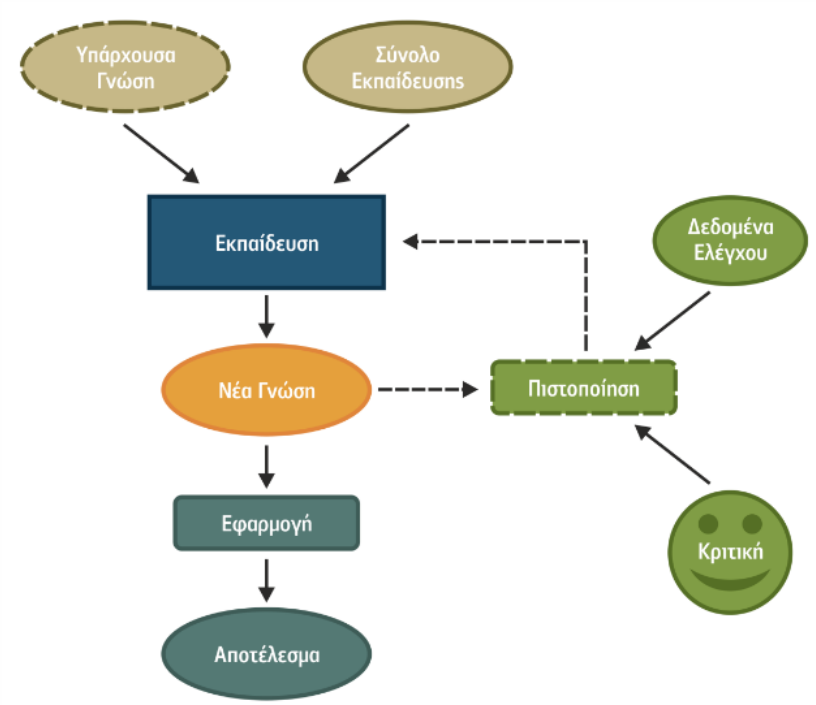
- Αλγόριθμοι που βασίζονται σε διαχωρισμούς (partition based)
- Ιεραρχικοί αλγόριθμοι (hierarchical)
- Πιθανοκρατικοί αλγόριθμοι (probabilistic algorithms)



Εικόνα 8 Παράδειγμα Ομαδοποίησης δεδομένων σε αγοραστές σπορ αυτοκινήτων με βάση ηλικία, ετήσιο εισόδημα και φύλο

Εδώ αξίζει να αναφερθεί ότι υπάρχει και η Ενισχυτική Μάθηση - Reinforcement Learning κατά την οποία ο αλγόριθμος μαθαίνει μέσα από ενέργειες που αλληλοεπιδρούν με το άμεσο περιβάλλον. Η εκμάθηση γίνεται μέσω μιας συνάρτησης η οποία δέχεται ως είσοδο το διάνυσμα μιας κατάστασης σαν δεδομένο εισόδου και εξάγει το βέλτιστο αποτέλεσμα. Χρησιμοποιείται σε προβλήματα Σχεδιασμού (Planning) όπως αυτά που αφορούν στον έλεγχο κινήσεων ρομπότ κυρίως σε βιομηχανικές εγκαταστάσεις. (Γεωργούλη, 2015)

Η Μηχανική Μάθηση, λοιπόν, επικεντρώνεται σε αλγορίθμους που σκοπό έχουν να εκπαιδεύσουν το εκάστοτε σύστημα με βάση τα δεδομένα ενός προβλήματος. Η εκπαίδευση αυτή λαμβάνει χώρα με την είσοδο ενός συνόλου δεδομένων εκπαίδευσης (training set) μέσω της οποίας το σύστημα θα παράγει γνώση. Η διαδικασία που ακολουθείται παρουσιάζεται στην παρακάτω εικόνα:



Εικόνα 9 Η λογική σειρά για την Μηχανική Μάθηση (Γεωργούλη, 2015)

Η Μηχανική Μάθηση βρίσκει εφαρμογή σε πολλά αντικείμενα της Τεχνητής Νοημοσύνης δίνοντας λύση σε προβλήματα που προκύπτουν, όπως επίσης ανακαλύπτοντας βέλτιστες μεθόδους. Οι εφαρμογές της μπορούν να εντοπιστούν σε περιπτώσεις μηχανικής αντίληψης για την εύρεση χαρακτηριστικών και ταξινόμησης, σε περιπτώσεις διαχείρισης πληροφοριών μεγάλης κλίμακας και ανάλυσης δεδομένων ώστε να βρίσκονται μοτίβα και σχέσεις μεταξύ δεδομένων ευκολότερα και γρηγορότερα, στο φιλτράρισμα και στην ανάκτηση πληροφοριών βοηθώντας στην διαχείριση και εκμετάλλευση μεγάλου όγκου δεδομένων, στον έλεγχο και βελτιστοποίηση, στην αναγνώριση φωνής αλλά και στην μηχανική όραση για την αναγνώριση

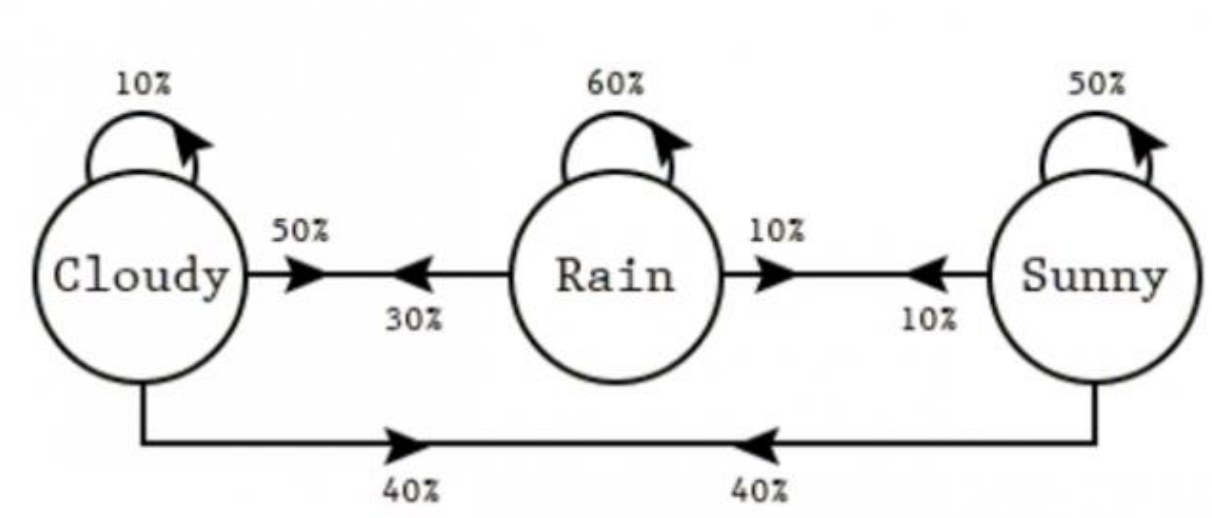
ανθρώπων είτε από την φωνή, είτε από χαρακτηριστικά προσώπου, είτε ακόμα και από τον γραφικό χαρακτήρα και τα δαχτυλικά αποτυπώματα. Τέλος, εφαρμογή της μηχανικής μάθησης μπορούμε να συναντήσουμε στην παραγωγή επιστημονικών μοντέλων σε πολλούς επιστημονικούς κλάδους όπως η ιατρική και η διαστημική έρευνα, και στην δημιουργία υπολογιστών που είναι πιο εύχρηστοι με βάση τις προτιμήσεις και τις απαιτήσεις των χρηστών.

Κεφάλαιο 4 Αλγόριθμοι που χρησιμοποιήθηκαν

4.1 Hidden Markov Models (HMM)

Τα Κρυφά Μοντέλα Markov είναι πιθανολογικά γραφικά μοντέλα που μας βοηθούν να προβλέψουμε μία ακολουθία αγνώστων μεταβλητών από ένα σύνολο παρατηρούμενων μεταβλητών. Οι άγνωστες μεταβλητές ονομάζονται και ως κρυφές καθώς δεν μπορούμε να τις δούμε. Τα HMM βασίζονται στις Markov αλυσίδες που είναι μοντέλα που δείχνουν τις πιθανότητες ακολουθιών τυχαίων καταστάσεων από ένα σύνολο. Αυτό το σύνολο μπορεί να είναι λέξεις ή σύμβολα που αντιπροσωπεύουν δεδομένα και οι καταστάσεις του είναι διακριτές.

Ένα παράδειγμα μιας αλυσίδας Markov βρίσκεται στην Εικόνα 10 όπου παρατηρούμε μια αλυσίδα με τρεις καταστάσεις: Cloudy, Rain, Sunny. Η μετάβαση από μία κατάσταση σε μία άλλη χαρακτηρίζεται από ένα ποσοστό πιθανότητας της μετάβασης αυτής.



Εικόνα 10 Καταστάσεις ενός μοντέλου Markov (Kumar, 2021)

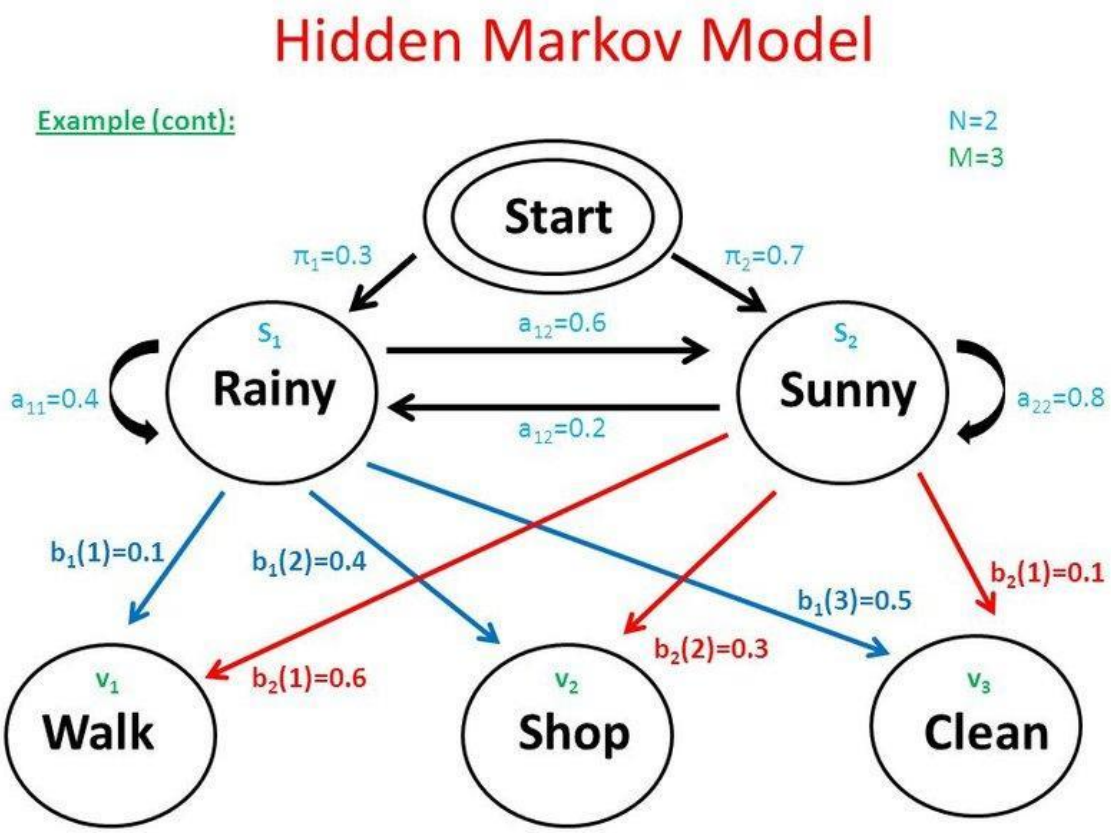
Οι αλυσίδες Markov βασίζονται στην Μαρκοβιανή υπόθεση ότι εάν θέλουμε να προβλέψουμε το μέλλον, μας χρειάζεται μόνο το παρόν και όχι γνώση για το παρελθόν. Με

αυτό σαν βάση, τα μοντέλα Markov χρειάζονται μόνο την κατάσταση την δεδομένη στιγμή για να προβλέψουν την επόμενη ανεξάρτητα με τις καταστάσεις που έχουν προηγηθεί.

$$P(z_t, z_{t-1}, z_{t-2}, \dots z_1) = P(z_t/z_{t-1})$$ (Vivek Vinushanth, 2020)

Ο λόγος για τον οποίο ονομάζονται Hidden Markov Models είναι επειδή κατασκευάζουμε ένα Markov μοντέλο για να υπολογίσουμε μία πιθανότητα για μία ακολουθία διακριτών γεγονότων, αλλά οι καταστάσεις είναι κρυμμένες και δεν μπορούμε να τις παρατηρήσουμε άμεσα. Ορατές είναι μόνο οι τελικές καταστάσεις.

Στην Εικόνα 11 παρουσιάζεται ένα Hidden Markov Model με κρυφες καταστάσεις Rainy και Sunny και με ορατές καταστάσεις τις Walk, Shop και Clean. Οι μεταβάσεις από την μία κατάσταση σε μία άλλη πιθανή κατάσταση, χαρακτηρίζεται από την πιθανότητα μετάβασης. Για παράδειγμα, εάν βρισκόμαστε στην αρχική κατάσταση START, υπάρχει 70% πιθανότητα να ακολουθήσει η κατάσταση Sunny και 30% πιθανότητα να ακολουθήσει η κατάσταση Rainy. Για τις μη ορατές καταστάσεις, υπάρχει 80% πιθανότητα να είναι Sunny μετά από μία Sunny ημέρα και 20% να βρέξει άρα και να είναι Rainy κατάσταση. Αντίστοιχα για τις ορατές καταστάσεις, αν βρισκόμαστε στην κατάσταση Sunny υπάρχει 60% πιθανότητα να παει περίπατο (κατάσταση Walk), 30% για ψώνια (Shop) και 10% για να καθαρίσει (Clean) και αν βρισκόμαστε στην Rainy κατάσταση υπάρχει 50% πιθανότητα να καθαρίσει (Clean), 40% πιθανότητα να πάει για ψώνια (Shop) και μόνο 10% να πάει για περίπατο (Walk).

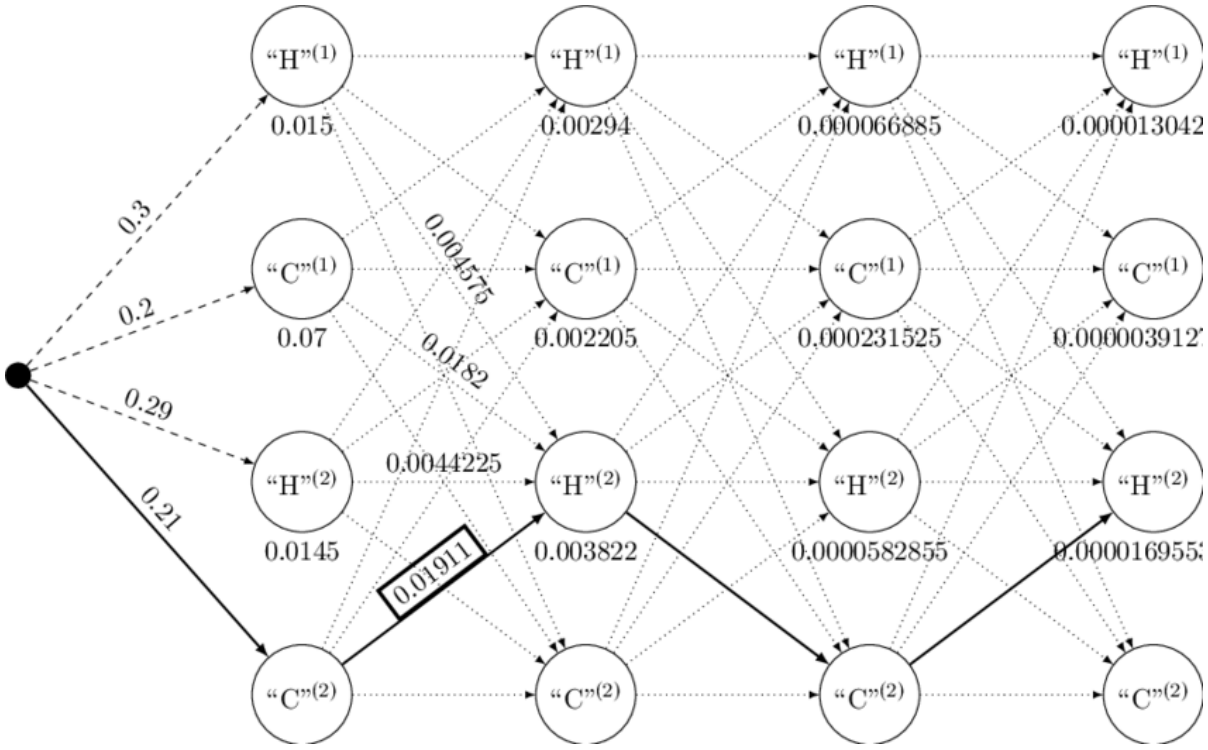


Εικόνα 11 Hidden Markov Model chain (Niall, 2018)

4.1.1 Αλγόριθμος του Viterbi

Όταν μας δίνεται μια σειρά παρατηρήσεων και θέλουμε να βρούμε τις πιο πιθανές ακολουθίες κρυφών καταστάσεων, χρησιμοποιούμε τον Viterbi αλγόριθμο. Ο αλγόριθμος αυτός ουσιαστικά επικεντρώνεται στην εύρεση του πιο πιθανού μονοπατιού που είναι και το συντομότερο ή όπως αλλιώς συναντάται στη βιβλιογραφία ενός trellis, δεδομένου ενός δείγματος από παρατηρήσεις και καταστάσεις. Το συντομότερο μονοπάτι παρουσιάζεται, συνήθως, ως ένα γράφημα με πεπερασμένες καταστάσεις (nodes) που ενώνονται με ακμές (edges) η οποία η κάθε μια είναι αναπαράσταση μιας πιθανής μετάβασης μεταξύ διαδοχικών χρονικών διακριτών διαστημάτων.

Τα διαγράμματα trellis, όπως στην Εικόνα 12, δείχνουν κάθε πιθανή μετάβαση των καταστάσεων σε μια χρονική στιγμή, με μια επόμενη κατάσταση για το επόμενο βήμα.



Εικόνα 12 Trellis Diagram

Ο αλγόριθμος Viterbi αναζητά την μέγιστη εκτίμηση πιθανότητας της πιο πιθανής ακολουθίας καταστάσεων η οποία ονομάζεται μονοπάτι Viterbi.

4.1.2 Baum-Welch Algorithm

Σε προηγούμενη ενότητα, αναφέρθηκε το HMM και ό,τι χρησιμοποιείται για την πρόβλεψη μελλοντικών καταστάσεων με βάση κάποιες εισόδους. Όλη η φιλοσοφία της μηχανικής μάθησης, βασίζεται στην εξής λογική. Με βάση κάποιες εισόδους να εκπαιδευτεί το σύστημα και να παράγει την κατάλληλη έξοδο σε μελλοντικές διαφορετικές εισόδους. Η ίδια φιλοσοφία αφορά και την εκμάθηση /εκπαίδευση ενός HMM , μόνο που εδώ γίνεται μια διαφοροποίηση, οι καταστάσεις είναι τυχαίες και η εξαγωγή των συμπερασμάτων βασίζεται κυρίως σε μελλοντικές καταστάσεις.

Για την εκπαίδευση ενός HMM χρησιμοποιείται ο αλγόριθμος Baum-Welch. Ο αλγόριθμος αυτός παρατηρεί τα δεδομένα και το σύνολο των πινάκων με τις παραμέτρους που του δίνονται και κάνει μια πρόβλεψη για τις αναμενόμενες κρυφές καταστάσεις. Στην συνέχεια, προσαρμόζονται με τον τρόπο που θα παρουσιαστεί στις επόμενες γραμμές αυτοί οι πίνακες παραμέτρων ώστε να ταιριάζουν καλύτερα στα δεδομένα που παρατηρήθηκαν και στις κρυφές καταστάσεις που αναμένονται. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να συγκλίνουν οι παράμετροι ή το μοντέλο να έχει επιτύχει κάποιον συγκεκριμένο βαθμό ακρίβειας.

Έτσι, λοιπόν, ξεκινώντας από ένα υποθετικό μοντέλο καταστάσεων θ, ο αλγόριθμος επαναλαμβάνεται δίνοντας κάθε φορά δημιουργώντας νέο θ'. Οι παράμετροι π, A, B όπου A και π είναι σύνολα :

$$P(\pi_i = k|x) = \frac{f_k(i)b_k(i)}{P(x)}$$

Για τις μεταβάσεις έχουμε

$$A_{kl} = \sum_{\pi} P(\pi|x,\theta`)A_{kl}(\pi) = \frac{1}{P(x)}\sum_{i} f_k(i)a_{kl}e_l(x_{i+1})\, b_l(i+1)$$

και αντίστοιχα για τις πιθανότητες εκπομπής

$$E_k(b) = \sum_{\pi} P(\pi|x,\theta`)E_k(b,\pi) = \frac{1}{P(x)} \sum_{\{i|x_i = b\}} f_k^j(i)b_k^j(i)$$

όπου $f_k(i)$, $b_k(i)$ οι μεταβλητές που υπολογίζονται από τους αλγόριθμους forward και backward, αντίστοιχα και i, j ∈ [1,N] .

και ορίζοντας την σχέση

$$Q(\theta|\theta`) = \sum_{k=1}\sum_{b} E_k(b)\, log\, e_k(b) + \sum_{k=0}\sum_{l=1} A_{kl}\, log\, a_{kl}$$

4.2 K-means

Ο αλγόριθμος K-means είναι ένας πολύ διάσημος και ισχυρός αλγόριθμος μηχανικής μάθησης. Χρησιμοποιείται για επίλυση πολλών σύνθετων προβλημάτων με μη επιβλεπόμενη μάθηση και προσπαθεί να ομαδοποιήσει παρόμοια δεδομένα με την μορφή cluster. Είναι ενας επαναληπτικός αλγόριθμος και βασίζεται στην απόσταση που έχουν τα δεδομένα μεταξύ τους. Στοχεύει στην ελαχιστοποίηση της απόστασης μεταξύ των σημείων και των κέντρων μέσα σε ένα cluster.

Ο αλγόριθμος ακολουθεί βασικά βήματα κατά την λειτουργία του:

- **Βήμα 1ο :** Ορισμός του αριθμού των K κέντρων που χρειάζονται,
- **Βήμα 2ο :** Αρχικοποίηση των κέντρων,
- **Βήμα 3ο :** Επιλογή cluster και εύρεση μέσης τιμής με βάση την απόσταση των δεδομένων από τα κέντρα που έχουν οριστεί,
- **Βήμα 4ο :** Υπολογισμός των κέντρων με βάση τα διαμορφωμένα cluster,
- **Βήμα 5ο :** Επανάληψη των βημάτων 3 και 4. Εάν δεν γίνεται κάτι καλύτερο πάει στο 6ο,
- **Βήμα 6ο :** Τερματισμός των διαδικασιών. Το μοντέλο είναι έτοιμο.

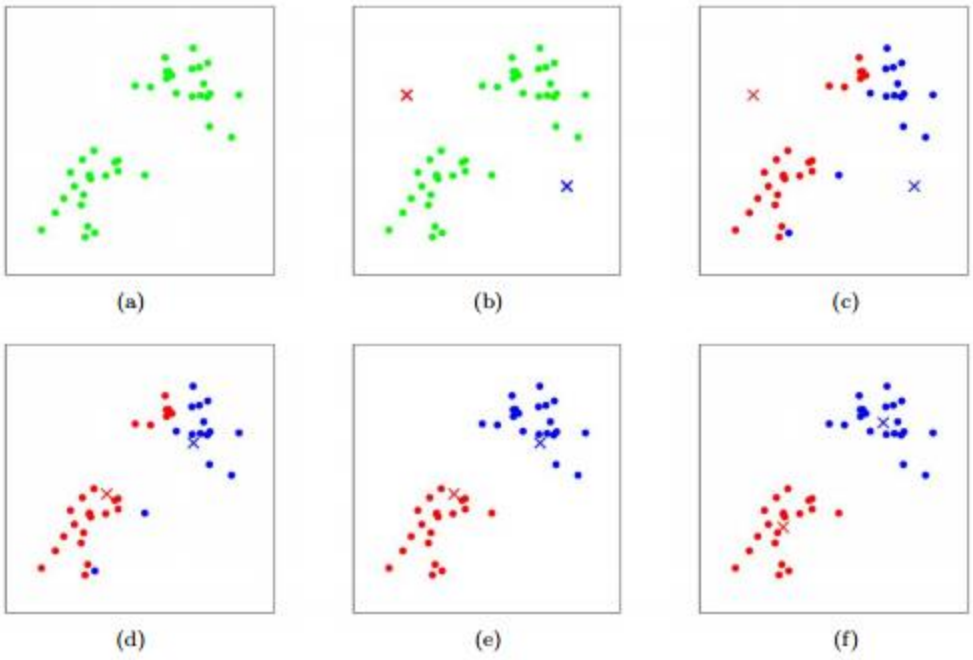
Ο τερματισμός των διαδικασιών του K-means γίνεται με βάση 3 βασικών κριτηρίων:

- Όταν τα κέντρα μένουν ίδια στα νέα clusters
- Όταν τα δεδομένα δεν αλλάζουν ομάδες
- Ή όταν εκπληρωθεί ένας προκαθορισμένος αριθμός επαναλήψεων

Στον K-means το K δηλώνει τον προκαθορισμένο αριθμό των κέντρων που απαιτείται να δημιουργηθούν στην διαδικασία του. Τα κέντρα μπορεί να είναι στοιχεία μέσα από τα ίδια τα δεδομένα ή εντελώς τυχαία σημεία.

Στην Εικόνα 13 Φάσεις κατά την εκτέλεση του K-meansδίνεται ένα παράδειγμα της εκτέλεσης του αλγορίθμου K-means με K=2. Τα δεδομένα στην εικόνα (a) δεν ανήκουν σε

κάποιο cluster και στην εικόνα (b) αρχικοποιούνται δύο κέντρα τυχαία στον χώρο που δεν αποτελούν στοιχεία των δεδομένων. Στις εικόνες (c) (d) (e) και (f) εκτελούνται τα βήματα 3 και 4 και έτσι δημιουργούνται clusters με βάση την απόσταση των σημείων από τα κέντρα και η επανατοποθέτηση των κέντρων στον χώρο για την ελαχιστοποίηση των αποστάσεων. Έχοντας βρει τη καλύτερη απόσταση κέντρων και στοιχείων, ο K-means σταματάει στην εικόνα (f) με δυο clusters.



Εικόνα 13 Φάσεις κατά την εκτέλεση του K-means

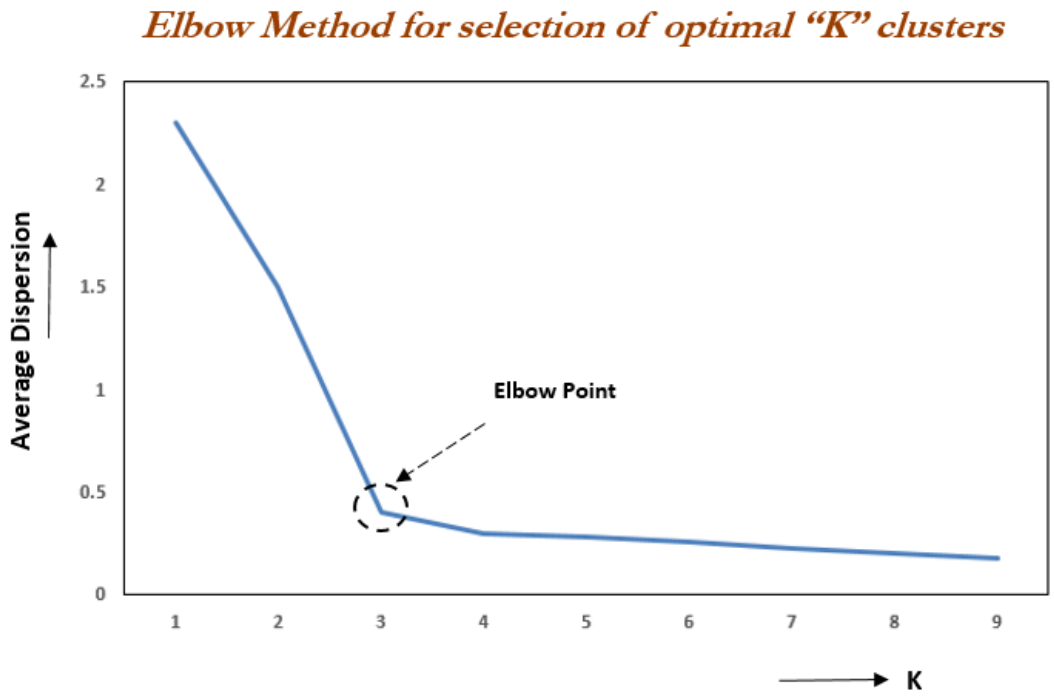
Αφού ο αλγόριθμος K-means είναι μη επιβλεπόμενης μάθησης, αμέσως προκαλείται ένα βασικό πρόβλημα, ο καθορισμός των αριθμών των clusters άρα και των κέντρων. Ο καθορισμός του αριθμού των κέντρων είναι πολύ σημαντικός καθώς εάν οριστεί τυχαία θα έχει αντίθετα και καταστροφικά αποτελέσματα στην ομαδοποίηση των δεδομένων αλλά και στην εξέλιξη της έρευνας που πραγματοποιείται με τον αλγόριθμο. Για να βρεθεί λοιπόν το πλήθος των K κέντρων υπάρχει η μέθοδος Elbow.

4.2.1 Elbow Method

Στη συγκεκριμένη μέθοδο, ουσιαστικά, αυτό που πραγματοποιείται είναι η διαφοροποίηση του αριθμού των clusters στη περιοχή από 1 έως και 10. Για κάθε μια από αυτές τις τιμές, υπολογίζεται το λεγόμενο WCSS (Within – Cluster Sum of Square). Το προαναφερόμενο WCSS, όπως προκαταβάλλεται από τις λέξεις, είναι το άθροισμα των τετραγώνων των αποστάσεων μεταξύ των σημείων και του κεντρικού σημείου στο cluster.

Όταν αυτά τα παραπάνω σημεία, παρουσιαστούν σε γραφική παράσταση θα παρουσιαστεί ένας αγκώνας (μια γωνία) εξού και το όνομα της μεθόδου.

Εδώ, αξίζει να σημειωθεί, ότι όσο αυξάνεται ο αριθμός των clusters που εισάγονται στο WCSS, το άθροισμα θα αρχίζει να μειώνεται. (Saji, 2021)



Εικόνα 14 Elbow Point

4.3 Bag of Words (BoW)

Πρόκειται για μια μέθοδος που αρχικά προτάθηκε και χρησιμοποιήθηκε για την επίλυση ανάκτησης κειμένου για ανάλυση εγγράφων και στη συνέχεια προσαρμόστηκε για την υπολογιστική όραση και τις διάφορες εφαρμογές της. Παρόλα αυτά, σήμερα, αποτελεί μια από τις πιο υποσχόμενες μεθόδους για την κατηγοριοποίηση εικόνων και συσχετισμού εικόνων. Για τη συγκεκριμένη χρήση, το BoW βασίζεται και αυτό με τη σειρά του στην τεχνική VQ (vector quantization) ομαδοποιώντας (clustering) οπτικά χαρακτηριστικά μιας εικόνας όπως είναι το χρώμα, υφή κ.ο.κ.

Για την εξαγωγή, λοιπόν, με τη μέθοδο BoW περιοχών της εικόνας με συγκεκριμένο οπτικό γνώρισμα, συνήθως ακολουθούνται τα παρακάτω βήματα:

i) Ανίχνευση των περιοχών ενδιαφέροντος

ii) Υπολογισμός των descriptors των περιοχών αυτών

iii) Quantization των παραπάνω υπολογισμών και αποτελεσμάτων και μετατροπή σε λέξεις για την οπτικοποίηση τους

iv) Ανίχνευση των πιθανοτήτων εμφάνισης των παραπάνω ποσοτήτων στην εικόνα έτσι ώστε να προστεθούν στο BoW.

Μαθηματικά, η μέθοδος BoW μπορεί να αποδοθεί ως εξής:

Δεδομένου ενός συνόλου εκπαίδευσης (training dataset) D το οποίο περιέχει n εικόνες και αναπαρίσταται ως εξής:

D =d1 , d2 , . . ., και dn, όπου d είναι το συγκεκριμένο οπτικό χαρακτηριστικό για το οποίο θα εφαρμοστεί η μέθοδος, ένας αλγόριθμος όπως των K-means χρησιμοποιείται για να ομαδοποιήσει το D σύνολο σε βάση έναν αριθμό από οπτικές λέξεις W , όπου W = w1, w2, …wv, όπου V είναι το πλήθος των clusters. Στη συνέχεια, αθροίζοντας τα δεδομένα σε έναν

πίνακα VxN, στον οποίο κάθε στοιχείο του δηλώνει πόσο συχνά η λέξη wi συναντάται στην εικόνα di. (Tsa, 2012)

Η μέθοδος BoW στα NLP προβλήματα λειτουργεί με παρόμοιο τρόπο καθώς δημιουργεί έναν πίνακα όπου παρουσιάζει την συχνότητα εμφάνισης μιας λέξης σε ένα κείμενο. Για να γίνει κατανοητή η εφαρμογή της μεθόδου αυτής στα NLP προβλήματα , παρουσιάζεται ένα απλό παράδειγμα στις επόμενες γραμμές.

Έστω οι δύο προτάσεις:

Sentence1: Will is a nice guy. Will likes nice movies.

Sentence2: William is a nice dude. William like good movies

Για κάθε μια από αυτές τις προτάσεις δημιουργούνται Bow , και είναι ως εξής:

BOW1: {“Will”:2, “is”:1, “a”:1, “nice”:2, “guy”:1, “likes”:1, “movies”:1}

BOW2: {“William”:2, “is”:1, “a”:1, “nice”:1, “dude”:1, “like”:1, “good”: 1, “movies”: 1}

Τα παραπάνω BoWs , θα μετατραπούν σε πίνακα με τη συχνότητα εμφάνισης κάθε λέξης και θα έχει την εξής μορφή:

“Will”	“is”	“a“	“nice”	“guy”	“likes”	“movies”	“William”	“dude”	“like”	“good”
2	1	1	2	1	1	1	0	0	0	0
0	1	1	1	0	0	1	2	1	1	1

Εικόνα 15 BoW στα NLP προβλήματα

4.4 Word2Vec

Η μέθοδος αυτή πρόκειται για μια τελευταία προσθήκη στην επίλυση των NLP. Προτάθηκε από τον Τσέχο Tomas Mikolov και πρόκειται για μια μέθοδος που ήρθε να συμβάλει στην επίλυση των NLP.

Συγκεκριμένα, η μέθοδος αυτή, έχει μια ιδιαίτερη 'ικανότητα', θα μπορούσε να πει κανείς. Μπορεί και ομαδοποιεί vectors παρόμοιων λέξεων (words). Ειδικά εάν της δοθούν είσοδος εύλογου μεγέθους, μπορεί να παράγει με απίστευτη ακρίβεια την σημασία μιας λέξεως απλώς βασιζόμενη στις εμφανίσεις της εντός του κειμένου. (Vatsal, 2021)

Ας δούμε ένα παράδειγμα βασισμένο στα κείμενα από τη Wikipedia. Σκοπός είναι ουσιαστικά να προβλεθεί η λέξη που συσχετίζεται με μια άλλη στο κείμενα , βασιζόμενοι στις εμφανίσεις αυτής της λέξης. Π.χ όταν υπάρχει ως δεδομένο ό,τι άντρας-> βασιλιάς, και υπάρχει η ερώτηση γυναίκα ->?, τότε ο Word2Vec, με τις αντίστοιχες υλοποιήσεις του, παρουσιάζει το εξής αποτέλεσμα:

```
[('βασίλισσα', 0.6089198589324951),
 ('βασιλιά', 0.5953291654586792),
 ('σύζυγο', 0.5802819132804871),
...
```

Η πρώτη απάντηση είναι και η αναμενόμενη και η σωστή. Τα νούμερα δίπλα σε κάθε αποτέλεσμα είναι η πιθανότητα για κάθε αποτέλεσμα που υπολογίζει ο Word2Vec.

Στην ερώτηση μαμά-> μπαμπάς, αγόρι->? , το μοντέλο βγάζει το εξής αποτέλεσμα:

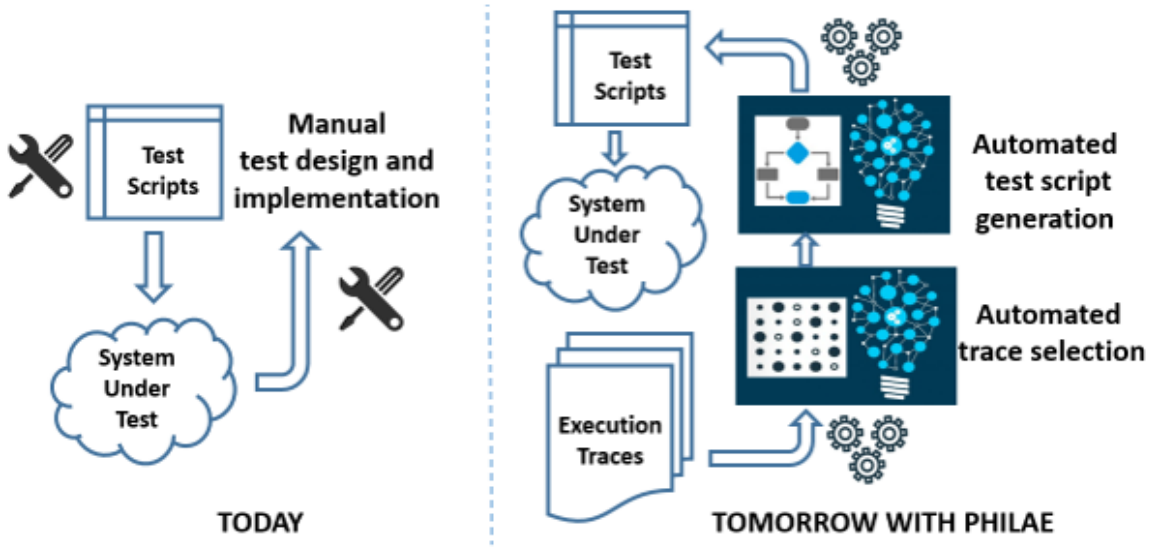
```
[('κορίτσι', 0.7002422213554382),
 ('μωρό', 0.6980640888214111),
 ('κοριτσάκι', 0.6687779426574707),
 ('τρελό', 0.6575441360473633),
 ('σκυλί', 0.6574292182922363),
 ('παιδάκι', 0.6519930362701416),
 ('σκυλάκι', 0.6386491656303406),
```

Και πάλι, η πρώτη απάντηση με τη μεγαλύτερη πιθανότητα να ταιριάζει στα ζητούμενα είναι και η σωστή και η αναμενόμενη.

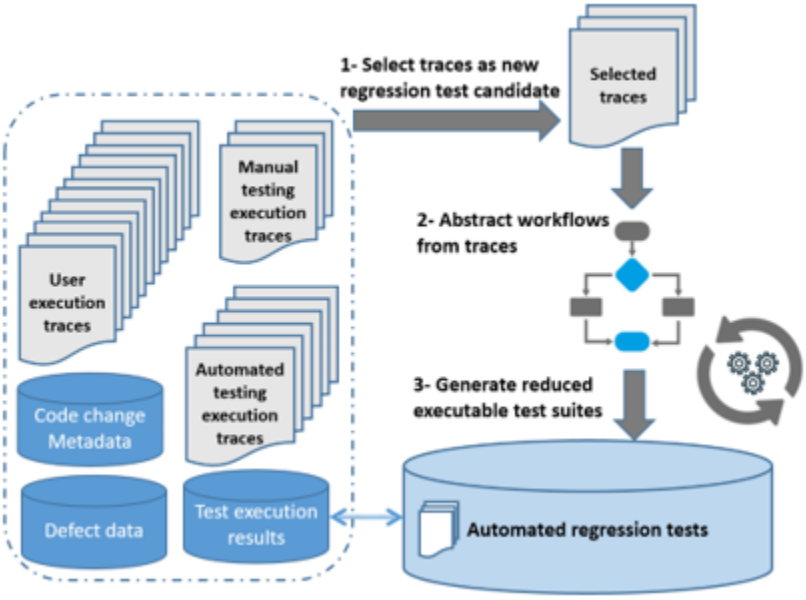
Κεφάλαιο 5 Εφαρμογή αλγορίθμων μηχανικής μάθησης στον έλεγχο λογισμικού

Η υλοποίηση πραγματοποιήθηκε στο πλαίσιο του PHILAE project. Αυτό το project έγινε σε συνεργασία του εργαστηρίου πληροφορικής (LIG) του πανεπιστημίου της Grenoble με το ινστιτούτο FEMTO-ST του πανεπιστημίου της Βουργουνδίας, το τμήμα Lab Services της εταιρίας τηλεπικοινωνιών της Γαλλίας Orange, την εταιρία Smarttesting Solutions & Services, το πανεπιστήμιο της Sunshine Coast και το ερευνητικό εργαστήριο Simula.

Ο στόχος είναι η δημιουργία και η συντήρηση αυτοματοποιημένων παλινδρομικών δοκιμών. Για να επιτευχθεί αυτός ο στόχος γίνεται ανάλυση δεδομένων εκτέλεσης λογισμικού, επιλογή αυτών με τεχνικές μηχανικής μάθησης και συνδυασμός αυτής της ανάλυσης με τα αποτελέσματα του μοντέλου και των αυτοματοποιημένων δοκιμών όπως φαίνεται στις παρακάτω εικόνες.



Εικόνα 16 Δοκιμές πριν και μετά το PHILAE project



Εικόνα 17 Τρόπος λειτουργίας του PHILAE project

Για να αξιολογηθεί η αποτελεσματικότητα της εύρεσης σφαλμάτων των test-suite που θα παραχθούν, χρησιμοποιήθηκε η δοκιμή σε mutation testing.

Στο mutation testing, δημιουργούνται σφάλματα σε κομμάτια κώδικα με τα οποία δοκιμάζεται το test-suite. Κάθε σφάλμα ονομάζεται mutant και για τις δοκιμές των αλγορίθμων χρησιμοποιήθηκαν 49 mutants που δημιουργήθηκαν εσκεμμένα για τις δοκιμές.

Η εφαρμογή των αλγοριθμων έγινε σε Python 3.7.9, σε περιβάλλον Windows 10 και χρησιμοποιήθηκαν οι βιβλιοθήκες Agilkia 0.6.0 και hmmlearn 0.2.2.

5.1 Πρώτη Υλοποίηση

Στην πρώτη υλοποίηση χρησιμοποιήθηκε ένα κρυφό μοντέλο Markov για την ομαδοποίηση και μείωση των δεδομένων από ένα προσομοιωτή καταστήματος supermarket και την χρήση αυτών ως test suite.

5.1.1 Data-set

Τα δεδομένα που είχαμε ήταν αλληλουχίες που προκύπτουν από έναν προσομοιωτή ενός supermarket. Είναι χωρισμένα σε sessions, που κάθε session αφορά τις ενέργειες από έναν «πελάτη», και σε steps που αφορούν κάθε ενέργεια ενός «πελάτη». Για παράδειγμα ο πελάτης ξεκινάει τα ψώνια του με ένα shop-scanner με το οποίο κάνει scan το barcode των προϊόντων. Έτσι, το session ξεκινάει με τον πελάτη να ξεκλειδώνει την συσκευή και ύστερα να προσθέτει ή αφαιρεί προϊόντα από το καλάθι. Όταν τελειώσει τα ψώνια, πάει στο ταμείο όπου μπορεί ο ταμίας να ελέγξει τα προϊόντα στο καλάθι αν συμφωνούν με την λίστα στο scanner ή απλά να πληρώσει και να λήξει το session.

Στην Εικόνα 18 παρουσιάζεται ένα δείγμα από το dataset 1024-steps που χρησιμοποιείται. Από αριστερά προς τα δεξιά, οι στήλες που υπάρχουν παρουσιάζουν τα δεδομένα timestamp, client (sessionID), action, inputs, output.

1584454655792	client0	scan0	debloquer	[]	0
1584454655801	client0	scan0	scanner	[8718309259938]	0
1584454656089	client1	scan1	debloquer	[]	0
1584454656095	client0	scan0	scanner	[3560070976478]	0
1584454656105	client1	scan1	scanner	[3560070048786]	0
1584454656127	client2	scan2	debloquer	[]	0
1584454656146	client0	scan0	scanner	[3560070139675]	0
1584454656152	client1	scan1	scanner	[3270190022534]	0
1584454656157	client2	scan2	scanner	[3020120029030]	0
1584454656192	client3	scan3	debloquer	[]	0
1584454656196	client0	scan0	scanner	[3560070976478]	0
1584454656198	client1	scan1	scanner	[3474377910724]	0
1584454656201	client2	scan2	scanner	[3474377910724]	0
1584454656203	client3	scan3	scanner	[3270190022534]	0
1584454656220	client0	scan0	scanner	[3046920010856]	0
1584454656225	client1	scan1	scanner	[8718309259938]	0
1584454656228	client2	scan2	scanner	[7640164630021]	-2
1584454656240	client3	scan3	scanner	[5410188006711]	0
1584454656253	client0	scan0	scanner	[3046920010856]	0
1584454656256	client1	scan1	scanner	[8715700110622]	0
1584454656258	client2	scan2	scanner	[8715700110622]	0

Εικόνα 18 Δείγμα του dataset που χρησιμοποιήθηκε

Για να είναι πιο ευανάγνωστα και για να μπορούν να συγκριθούν, τα δεδομένα επεξεργάζονται και μετατρέπονται σε ακολουθίες που έχουν ένα γράμμα για την κάθε δράση του όπως φαίνεται στο παρακάτω παράδειγμα:

u……d.tap

που σημαίνει ότι ο πελάτης ξεκλείδωσε το scanner (u) και έκανε scan 6 προϊόντα (.), διέγραψε ένα προϊόν (d), έκανε scan ένα προϊόν (.), μετά πήγε σε ένα ταμείο και μετέφερε (t) τις αγορές του, έκλεισε το scanner (a) και πλήρωσε (p).

Το mapping που ακολουθείται είναι το εξής:

{‘abandon’: ’a’, ‘debloquer’: ’u’, ‘payer’: ‘p’, ‘scanner’: ’.’, ‘transmission’: ‘t’, ‘supprimer’: ‘d’, ‘ajouter’: ‘+’, ‘ouvrirSession’: ‘o’, ‘fermerSession’: ‘c’}

5.1.2 Ομαδοποίηση δεδομένων με Hidden Markov Models

Ο αλγόριθμος HMM χρησιμοποιήθηκε καθώς τα δεδομένα που υπάρχουν είναι αλληλουχίες και είναι επιθυμητό να ομαδοποιούνται με βάση την αρχική τους μορφή. Επίσης τα μοντέλα καταστάσεων αναπαριστούν καλύτερα την συμπεριφορά ενός λογισμικού και μπορούν να χρησιμοποιηθούν ως η βάση για δημιουργία test βασισμένα σε μοντέλα.

Αρχικά για να γίνει η ομαδοποίηση χρειάζεται να υπολογιστεί η απόσταση της κάθε αλληλουχίας με το κέντρο. Η ιδέα της ομαδοποίησης βασίστηκε στον αλγόριθμο K-means με την διαφορά ότι για κέντρα θα χρησιμοποιηθούν HMM και η κάθε αλληλουχία θα έχει απόσταση από τα μοντέλα με βάση την πιθανότητα να παραχθεί από αυτά. Έτσι, ο αλγόριθμος που ακολουθείται είναι ο εξής:

1. Δέχεται K τυχαία HMM για κέντρα.
2. Υπολογίζει την απόσταση μεταξύ μοντέλου και αλληλουχίας.
3. Αναθέτει κάθε αλληλουχία στο μοντέλο που είναι πιο κοντά.
4. Επανέλαβε από το $2^{ο}$ βήμα μέχρι οι αλληλουχίες να σταματήσουν να αλλάζουν clusters.

Τα τυχαία HMM που θα επιλεχθούν, πρέπει να έχουν σχέση με τον τύπο των αλληλουχιών που θα ομαδοποιηθούν. Για να επιτευχθεί αυτό ακολουθούνται οι παρακάτω διαδικασίες:

1. Το πρώτο μοντέλο αρχικοποιείται με μία τυχαία αλληλουχία από τα δεδομένα.
2. Οι αποστάσεις μεταξύ των αλληλουχιών και του μοντέλου υπολογίζεται.
3. Χρησιμοποιώντας τις αποστάσεις, επιλέγεται μια αλληλουχία που βρίσκεται πιο μακριά από το μοντέλο.
4. Αρχικοποιείται ένα καινούριο μοντέλο με βάση αυτή την απομακρυσμένη αλληλουχία.

5. Υπολογίζονται οι αποστάσεις των μοντέλων και των αλληλουχιών και επιστροφή στο 3^{ο} βήμα.

Με αυτή την μεθοδολογία, στη πρώτη υλοποίηση μειώνεται ένα dataset με 61 sessions σε 10 με τον HMM αλγόριθμο να ομαδοποιεί σε 10 clusters τα δεδομένα και ύστερα να επιλέγει από το κάθε cluster μια τυχαία αλληλουχία. Αυτές οι αλληλουχίες που επιλέχθηκαν αποτελούν το μειωμένο dataset που θα χρησιμοποιηθεί ως regression test-suite.

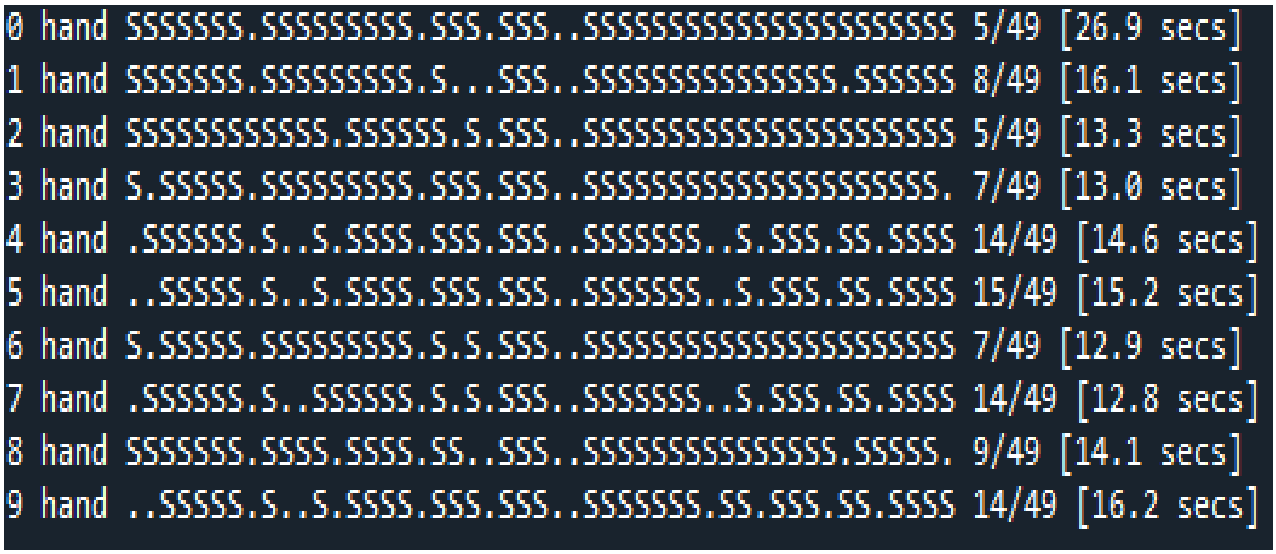
.

5.1.3 Αποτελέσματα

Αρχικά, δοκιμάστηκε ολόκληρο το dataset, χωρίς επεξεργασία, ως test-suite. Τα αποτελέσματα αυτά βοηθούν ώστε να μπορούμε να συγκρίνουμε τα αποτελέσματα από κάθε μέθοδο που εφαρμόζεται.

Μετά την μείωση των δεδομένων από 61 sessions σε 10 και την δοκιμή του test-suite ενάντια σε 49 mutants, προκύπτει ότι με το μειωμένο dataset εντοπίζεται ο ίδιος αριθμός mutant που εντοπίζεται με ολόκληρο το data-set ως test-suite.

Στην Εικόνα 19 εμφανίζονται τα δεδομένα μετά την δοκιμή. Έχουμε 10 επιλεγμένες αλληλουχίες οι οποίες δοκιμάζονται ενάντια σε 49 mutants. Συνολικά εντοπίζονται 19 διαφορετικά σφάλματα, δηλαδή 19 mutants. Για κάθε mutant που δεν εντοπίστηκε, σημειώνεται ένα “S” και ένα “.” για κάθε mutant που εντοπίζεται.



Εικόνα 19 Αποτελέσματα 10 sessions με mutants

Οδηγούμαστε στο αποτέλεσμα δηλαδή της μείωσης στο 1/6 των sessions άρα και των test αλλά με την ίδια αποτελεσματικότητα, εντοπίζοντας τον ίδιο αριθμό mutants.

Η μείωση του dataset είναι σημαντική διότι η δημιουργία test-suite με όλα τα δεδομένα και η δοκιμή τους στο mutation testing είναι χρονοβόρα. Έτσι κρίνεται απαραίτητη η αφαίρεση sessions μέσα από το dataset που δεν είναι αποτελεσματικά ως test suite.

1026-steps.csv	Sessions	Hand Mutants killed (49)
No reduction	61	19
HMM reduction (30 clusters)	30	19
HMM reduction (10 clusters)	10	19

Μπορούμε ακόμα να συγκρίνουμε τα αποτελέσματα από την μείωση δεδομένων με βάση την ομαδοποίηση με ένα HMM με αποτελέσματα από μία άλλη προσέγγιση που είχε υλοποιηθεί στο συγκεκριμένο project στην οποία έγινε επεξεργασία δεδομένων με τον αλγόριθμο Word2Vec και ομαδοποίηση με τον αλγόριθμο K-means.

Algorithm	Clusters (number of sessions)	Killed mutants
Word2Vec + K-means	5	18
	10	19
HMM	5	17
	10	19

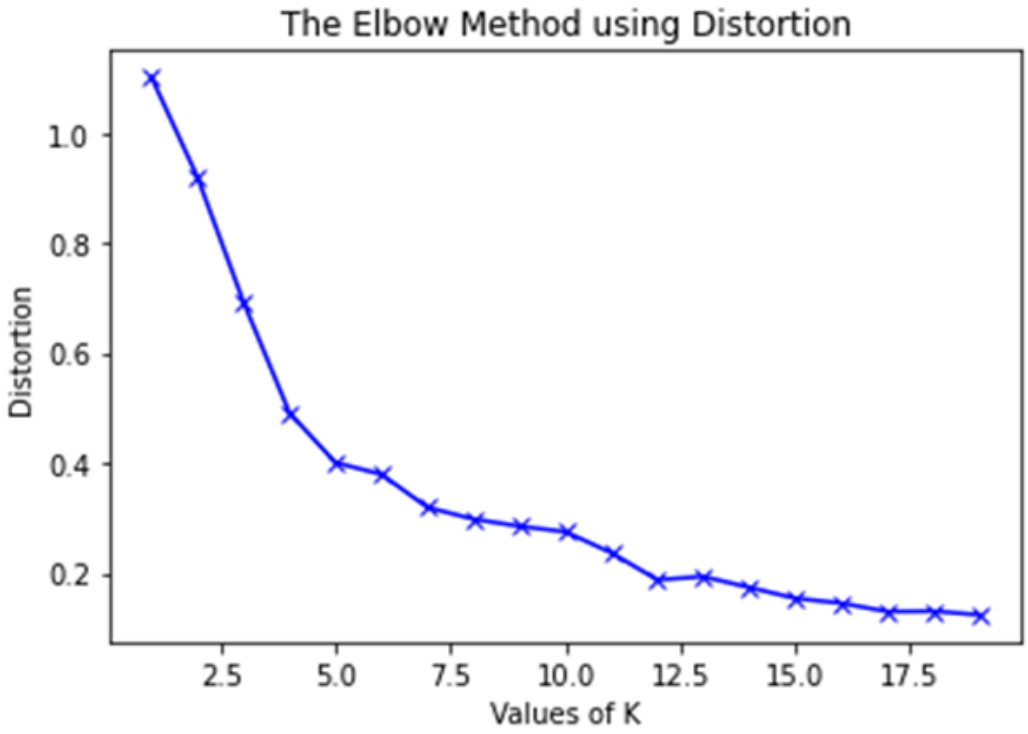
Παρατηρούμε ότι στην περίπτωση των 5 sessions ο αλγόριθμος του HMM υστερεί γιατί εντοπίζει ένα λιγότερο mutant έναντι των αλγορίθμων Word2Vec και K-means. Όμως ο στόχος είναι ο εντοπισμός των ίδιων mutants με βάση των δοκιμών χωρίς κάποια επεξεργασία σε κάθε προσέγγιση. Έτσι καταλήγουμε ότι και στις δυο προσεγγίσεις βρίσκουμε 10 sessions οι οποίες μας εντοπίζουν τον μέγιστο αριθμό mutants (19).

5.2 Δεύτερη Υλοποίηση

Στην δεύτερη υλοποίηση θα γίνει εξωτερική εκτίμηση της ομαδοποίησης με τον αλγόριθμο K-means. Για να γίνει αυτό θα πρέπει να θέσουμε ένα ground truth (βασική αλήθεια) με δεδομένα από το mutation testing και μετά να τα ομαδοποιήσουμε με τον K-means αλγόριθμο αναζητώντας και τον βέλτιστο αριθμό clusters (k). Η αξιολόγηση της ομαδοποίησης με την σύγκριση του ground truth και των δεδομένων από το supermarket γίνεται με το v-measure. Το v-measure είναι ο μέσος όρος ομοιογένειας (homogeneity) και πληρότητας (completeness). Η ομοιογένεια μεγιστοποιείται όταν κάθε cluster περιέχει στοιχεία όμοια όσο το δυνατόν λιγότερων διαφορετικών κατηγοριών και η πληρότητα μεγιστοποιείται όταν κάθε στοιχείο μίας συγκεκριμένης κλάσης ανήκει στο ίδιο cluster. (Rosenberg & Hirschberg, 2007)

5.2.1 Επεξεργασία και Ομαδοποίηση

Το πακέτο δεδομένων από τον προσομοιωτή καταστήματος supermarket με 7079 sessions δοκιμάστηκε ως test-suite χωρίς κάποια επεξεργασία ή μείωση και τα αποτελέσματα που βγήκαν μετατράπηκαν σε αριθμητικά, από “S” – “.” σε 0 και 1 αντίστοιχα. Ύστερα, με την Elbow Method ελέγχθηκε ένα εύρος αριθμών για clusters από k=1 έως k=20 για να βρεθεί ο καλύτερος. Στην Εικόνα 9Εικόνα 20, φαίνεται ότι για k=5 είναι η καλύτερη επιλογή. Έτσι το ground truth είναι τα 5 clusters με τα δεδομένα από το mutation testing.



Εικόνα 20 Αποτελέσματα απο Elbow Method

Για να συγκρίνουμε το clustering του K-means στο data-set από το supermarket θα πρέπει πρώτα να μετατρέψουμε τα δεδομένα σε αριθμητικά. Με την βοήθεια των αλγορίθμων BoW και W2V μπορούμε να μετατρέψουμε τα δεδομένα σε αριθμητικά δεδομένα αφότου έχουμε εφαρμόσει διαφορετικά mappings. Χρησιμοποιούμε την Elbow Method κάθε φορά για να βρούμε τον βέλτιστο αριθμό K και δημιουργούμε τα clusters που θα συγκριθούν με το ground truth.

5.2.2 Αποτελέσματα

Στην Εικόνα 21 παρουσιάζονται να αποτελέσματα με το v-measure να μην ξεπερνάει το 23% στα δεδομένα με mapping μόνο με το όνομα της μεθόδου, με standard scaler και BoW.

Clustering Pipeline	Number of Clusters (100043-steps)	Homogeneity	Completeness	V-measure
M BOW N K	9	0.18	0.13	0.16
MR BOW N K	8	0.15	0.11	0.13
MRP BOW N K	10	0.10	0.07	0.09
MRO BOW N K	9	0.16	0.12	0.14
M W2V N K	9	0.16	0.11	0.13
MR W2V N K	9	0.17	0.12	0.14
MRP W2V N K	8	0.15	0.12	0.13
MRO W2V N K	11	0.17	0.12	0.14
M BOW S K	10	0.26	0.21	0.23
MR BOW S K	7	0.20	0.21	0.20
MRP BOW S K	2	0.0001	0.0003	0.0001
MRO BOW S K	9	0.21	0.20	0.21

Εικόνα 21 Αποτελέσματα

Mappings:

- **M** : Method name. Example of a event : scanner
- **MR** : Method name + return code. Example of event : scanner—0
- **MRP** : Method name + return code + parameter name . Example of event : scanner—0—3270190022534
- **MRO** : Method name + return code + Object name. Example of event : scanner—0—scan0

Encoding:

- **BOW**: Bag Of Words
- **W2V**: Word2Vec

Preprocessing:

- **N**: no Pre-Processing
- **S**: Standard Scaler

Model:

- **K**: K-means

Τα αποτελέσματα της εξωτερικής εκτίμησης της ομαδοποίησης με βάση clusters με τα mutation scores, δείχνουν ότι η καλύτερη περίπτωση ομαδοποίησης του K-means είναι αυτή με το mapping Method name Bag Of Words (23%). Βέβαια το ποσοστό 23% είναι πάλι χαμηλό.

Ταυτοχρόνως εξεταζόταν και μία άλλη προσέγγιση με ground truth με δεδομένα που προέκυπταν από τις γραμμές κώδικα που εντοπίζονταν εσφαλμένες από δοκιμή του test suite. Εκεί τα αποτελέσματα ήταν διαφορετικά όπως θα δούμε στο παρακάτω πίνακα με αυτά που βρήκαμε.

Clustering Pipeline	Number of Clusters (100043-steps)	V-measure (Line coverage Tests)	V-measure (Mutation Tests)
M BOW N K	9	0.45	0.16
MR BOW N K	8	0.45	0.13
MRP BOW N K	10	0.31	0.09
MRO BOW N K	9	0.46	0.14
M W2V N K	9	0.52	0.13
MR W2V N K	9	0.47	0.14
MRP W2V N K	8	0.45	0.13
MRO W2V N K	11	0.52	0.14
M BOW S K	10	0.83	0.23
MR BOW S K	7	0.87	0.20
MRP BOW S K	2	0.63	0.0001
MRO BOW S K	9	0.91	0.21

Κεφάλαιο 6 Συμπεράσματα

Όπως είδαμε στην πρώτη υλοποίηση για την μείωση των δεδομένων, ο αλγόριθμος HMM είναι αρκετά αποτελεσματικός καθώς καταφέρνει να μειώσει δραστικά τα δεδομένα με αποτέλεσμα να δημιουργηθεί μικρό test suite χωρίς όμως να μειωθεί η αποτελεσματικότητα στον εντοπισμό σφαλμάτων. Η μείωση αυτή καθιστά την δημιουργία των δοκιμών πολύ πιο γρήγορη. Συγκριτικά όμως τα αποτελέσματα είναι το ίδιο καλά με τα αποτελέσματα άλλων αλγορίθμων όπως του K-means και ο αριθμός των δεδομένων δεν μπορούν να είναι λιγότερος από 10 sessions και να βρίσκει το μέγιστο δυνατό αριθμό σφαλμάτων.

Συμπερασματικά, με την βοήθεια της μηχανικής μάθησης η παραγωγή αυτοματοποιημένων δοκιμών γίνεται πιο γρήγορα καθώς χρησιμοποιούνται λιγότερα και χρήσιμα δεδομένα τα οποία αποδίδουν το ίδιο στις δοκιμές. Έτσι η χρήση μεθόδων μηχανικής μάθησης στον έλεγχο ορθής λειτουργίας λογισμικού δίνει την λύση στο πρόβλημα που υπάρχει μέχρι τώρα στην πρόβλεψη και στον εντοπισμό σφαλμάτων που βασίζονται στην χρήση του λογισμικού από τους χρήστες του.

6.1 Ανοιχτά ζητήματα

Στις δύο υλοποιήσεις υπάρχουν ανοιχτά ζητήματα είτε για την περεταίρω δοκιμής των αλγορίθμων είτε για την εξέταση της αποτελεσματικότητας των μεθοδολογιών που εφαρμόστηκαν.

6.1.1 Εξέταση αποτελεσματικότητας του HMM με διαφορετικούς τρόπους δοκιμής

Για να αξιολογηθεί η αποτελεσματικότητα της εύρεσης σφαλμάτων του test-suite που έχει παραχθεί, χρησιμοποιήθηκε η δοκιμή σε mutation testing. Στο mutation testing, το test-suite δοκιμαζόταν σε 49 κομμάτια κώδικα που είχαν κατασκευαστεί με σφάλματα. Είναι ενδιαφέρον το test suite να δοκιμαστεί και στα εξής εργαλεία που παράγουν αυτοματοποιημένα mutants:

- Jumble
- PiTest

Τα συγκεκριμένα εργαλεία χρησιμοποιήθηκαν στο PHILAE project για την δοκιμή μερικών αλγορίθμων και του test suite που δημιουργούσαν. Με αυτές τις επιπλέον δοκιμές θα μπορεί ο αλγόριθμος να εξεταστεί αν είναι το ίδιο αποδοτικός και να συγκριθεί με άλλους αλγορίθμους.

6.1.2 Επιπλέον μελέτη της εξωτερικής εκτίμησης της ομαδοποίησης του K-means

Η δεύτερη υλοποίηση ακολουθεί μια μεθοδολογία που είναι ακόμα σε εμπειρικό επίπεδο. Χρειάζεται να εξεταστεί και σε άλλα use-cases όπως επίσης και με άλλους αλγορίθμους ομαδοποίησης για να διαπιστωθεί εάν είναι αποτελεσματική σαν προσέγγιση.

Βιβλιογραφία

Abhijit A. Sawant, Pranit H. Bari, & P. M. Chawan. (2012, May-Jun). Software Testing Techniques and Strategies. *International Journal of Engineering Research and Applications (IJERA)*, σσ. 980-986.

Bertolino, A. (2007). Software Testing Research: Achievements, Challenges, Dreams. *Future of Software Engineering(FOSE'07)*, σσ. 85-103.

Blunsom, P. (2004, August 19). Hidden Markov Models.

Christopher, V. V. (2020, August 18). *towards data science*. Ανάκτηση από https://towardsdatascience.com/markov-and-hidden-markov-model-3eec42298d75

Dondeti, S. N. (2012, June). BLACK BOX AND WHITE BOX TESTING. *International Journal of Embedded Systems and Applications (IJESA)*, σ. 22.

Kumar, A. (2021, September 16). *Data Analytics*. Ανάκτηση από https://vitalflux.com/hidden-markov-models-concepts-explained-with-examples/

McCallum, A. (2004). *HMM -Baum Welch Algorithm.*

Mohd. Ehmer Khan, F. K. (2012). A Comparative Study of White Box, Black Box and. *(IJACSA) International Journal of Advanced Computer Science and Applications*, σ. 4.

Muhammad Abid Jamil, Muhammad Arif, Normi Sham Awang Abubakar, & Akhlaq Ahmad. (2016, November). Software Testing Techniques: A Literature Review. *2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, σσ. 177-182.

Niall Martin Ryan. (2018). *Citation Data-set for Machine Learning Citation Styles and Entity Extraction from Citation Strings.*

Piech, C. (2012). *Stanford CS221*. Ανάκτηση από https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

Rosenberg A., Hirschberg J. (2007, January 1). V-Measure: A Conditional Entropy-Based External Cluster.

Saji, B. (2021, January 20). *In-depth Intuition of K-Means Clustering Algorithm in Machine Learning*. Ανάκτηση από https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/

Syed Roohullah Jan, Syed Tauhid Ullah Shah, Zia Ullah Johar, Yasin Shah, & Fazlullah Khan. (2016, April). An Innovative Approach to Investigate Various Software. *International Journal of Scientific Research in Science, Engineering and Technology IJSRSET*, σσ. 682-689.

Teuvo Kohonen, J. H. (1996, January). SOM PAK The Self-Organizing Map Program Package.

Tsa, C.-F. (2012, September 19). Bag-of-Words Representation in Image Annotation: A Review. σ. 19.

Vatsal. (2021, January 29). *Word2Vec Explained*. Ανάκτηση από https://towardsdatascience.com/word2vec-explained-49c52b4ccb71

Γεωργούλη, Α. (2015). Μηχανική Μάθηση. Στο *ΤΕχνητή Νοημοσύνη*. Καλλίπος, Ανοικτές Ακαδημαϊκές Εκδόσεις.