ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΙΧΝΕΥΣΗ ΜΑΘΗΣΙΑΚΟΥ ΕΠΙΠΕΔΟΥ ΜΕ ΤΗ ΧΡΗΣΗ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

ΒΑΣΙΛΕΒΑ ΜΑΡΙΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κωνσταντίνος Κολομβάτσος
Επίκουρος Καθηγητής

ΣΥΝΕΠΙΒΛΕΠΩΝ

Νικόλαος Ζυγούρης
Καθηγητής

Λαμία 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΙΧΝΕΥΣΗ ΜΑΘΗΣΙΑΚΟΥ ΕΠΙΠΕΔΟΥ ΜΕ ΤΗ ΧΡΗΣΗ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

ΒΑΣΙΛΕΒΑ ΜΑΡΙΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κωνσταντίνος Κολομβάτσος
Επίκουρος Καθηγητής

ΣΥΝΕΠΙΒΛΕΠΩΝ

Νικόλαος Ζυγούρης
Καθηγητής

Λαμία 2025

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE &
TELECOMMUNICATIONS

LEARNING LEVEL DETECTION USING ARTIFICIAL INTELLIGENCE

VASILEVA MARIA

FINAL THESIS

ADVISOR

Konstantinos Kolomvatsos
Assistant Professor

CO ADVISOR

Nikolaos Zygouris
Assistant Professor

Lamia 2025

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

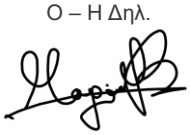
*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: 20/9/2025

Ο – Η Δηλ.



(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση

του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων

σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Η τεχνητή νοημοσύνη αναπτύσσεται ραγδαία και ενσωματώνεται σε ολοένα και περισσότερους τομείς στην καθημερινότητα μας. Ένας από αυτούς, που αξιοποιώντας την μπορεί να επωφεληθεί σημαντικά είναι ο τομέας της εκπαίδευσης. Σκοπός της παρούσας εργασίας, είναι να διερευνήσει το πως η ανάλυση δεδομένων μαθητή μπορεί να αποτελέσει ένα ισχυρό παιδαγωγικό εργαλείο.

Στο πλαίσιο της εργασίας, αναπτύχθηκε εφαρμογή για υπολογιστή η οποία λειτουργεί ως βοηθός ανίχνευσης μαθησιακού επιπέδου ενός μαθητή πρωτοβάθμιας εκπαίδευσης (ηλικία 6 έως και 14) στην αγγλική γλώσσα. Η εφαρμογή περιλαμβάνει τέσσερις ενότητες αξιολόγησης: λεξιλογίου και γραμματικής, οπτικοχωρικής αντίληψης, οπτικής μνήμης και συνδυασμό προσοχής και αντίδρασης σε συγκεκριμένες εικόνες. Από κάθε ενότητα συλλέγονται συγκεκριμένα δεδομένα, όπως χρόνος αντίδρασης, ορθότητα απάντησης και αναλύονται από εκπαιδευμένα μοντέλα «Μηχανών Υποστήριξης Διανυσμάτων» (SVM) κατηγοριοποίησης, τα οποία δίνουν αποτέλεσμα για κάθε ενότητα ξεχωριστά. Έπειτα, τα αποτελέσματα από τα SVM εισάγονται σε μοντέλο «Δάση Τυχαίων Αποφάσεων» (Random Forest) το οποίο παράγει την τελική απάντηση για το επίπεδο του μαθητή.

ABSTRACT

Artificial Intelligence has been developing rapidly and is being integrated gradually more into many areas of our daily lives. One of those areas that can benefit significantly from utilizing it is the process of education. The purpose of this paper is to investigate how student data analysis may become a powerful pedagogical tool.

In the context of this thesis, a desktop application was developed with the purpose of detecting the learning level of primary school students (aged 6 to 14) in English language. The application consists of four main sections: vocabulary and grammar, visual pattern recognition, visual memory and response inhibition. From each section specific data is being collected such as reaction time, answer accuracy and they are analyzed by specially trained Support Vector Machines (SVM) classifier, which provides an initial performance classification for each distinct area. Afterwards, these individual results are fed into another Random Forest classification model, which generates a final decision of the student's overall learning level.

Table of Contents

ΠΕΡΙΛΗΨΗ I
ABSTRACT III

ΚΕΦΑΛΑΙΟ 1 ΕΙΣΑΓΩΓΗ 4

1.1 ΣΚΟΠΟΣ ΤΗΣ ΠΤΥΧΙΑΚΗΣ 4
1.2 ΔΟΜΗ ΠΤΥΧΙΑΚΗΣ 4

2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ ΕΦΑΡΜΟΓΗΣ 6

2.1 Η ΑΓΓΛΙΚΗ ΓΛΩΣΣΑ 6
2.1.1 ΕΙΣΑΓΩΓΗ ΚΑΙ ΙΣΤΟΡΙΚΑ ΑΓΓΛΙΚΗΣ ΓΛΩΣΣΑΣ 6
2.1.2 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΜΟΝΤΕΡΝΑΣ ΑΓΓΛΙΚΗΣ ΓΛΩΣΣΑΣ 6
2.1.2.1 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΛΕΞΙΛΟΓΙΟΥ 7
2.1.2.2 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΓΡΑΜΜΑΤΙΚΗΣ 7
2.1.2.3 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΟΡΘΟΓΡΑΦΙΑΣ ΚΑΙ ΦΩΝΟΛΟΓΙΑΣ 8
2.2 Η ΕΛΛΗΝΙΚΗ ΓΛΩΣΣΑ 8
2.2.1 ΕΙΣΑΓΩΓΗ ΚΑΙ ΙΣΤΟΡΙΚΑ ΕΛΛΗΝΙΚΗΣ ΓΛΩΣΣΑΣ 8
2.2.2 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΕΛΛΗΝΙΚΗΣ ΓΛΩΣΣΑΣ ΚΑΙ Η ΔΙΑΦΟΡΑ ΤΗΣ ΑΠΟ ΤΗΝ ΑΓΓΛΙΚΗ 9
2.2.2.1 ΕΙΣΑΓΩΓΗ ΣΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ ΕΛΛΗΝΙΚΗΣ 9
2.2.2.2 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΛΕΞΙΛΟΓΙΟΥ ΚΑΙ ΔΙΑΦΟΡΕΣ 9
2.2.2.3 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΓΡΑΜΜΑΤΙΚΗΣ ΚΑΙ ΔΙΑΦΟΡΕΣ 9
2.2.2.4 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΟΡΘΟΓΡΑΦΙΑΣ ΚΑΙ ΦΩΝΟΛΟΓΙΑΣ ΚΑΙ ΔΙΑΦΟΡΕΣ 10
2.3 ΓΝΩΣΤΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ 10
2.3.1 ΕΡΓΑΖΟΜΕΝΗ ΜΝΗΜΗ (WORKING MEMORY) 10
2.3.2 ΟΠΤΙΚΗ ΕΡΓΑΖΟΜΕΝΗ ΜΝΗΜΗ (VISUAL WORKING MEMORY) 11
2.3.3 ΟΠΤΙΚΟΧΩΡΙΚΗ ΑΝΤΙΛΗΨΗ (VISUOSPATIAL FUNCTION) 11
2.3.4 INHIBITION (GO/NOGO) 11
2.3.5 ΔΙΑΦΟΡΕΣ ΓΝΩΣΤΙΚΩΝ ΛΕΙΤΟΥΡΓΙΩΝ ΠΟΥ ΜΕΛΕΤΗΘΗΚΑΝ 11
2.3.6 ΓΝΩΣΤΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΚΑΙ ΕΚΜΑΘΗΣΗ ΞΕΝΩΝ ΓΛΩΣΣΩΝ 12
2.3.6.1 ΈΡΕΥΝΑ: ΕΡΓΑΖΟΜΕΝΗ ΜΝΗΜΗ ΚΑΙ ΑΝΑΣΤΟΛΗ 12
2.3.6.2 ΈΡΕΥΝΑ: ΟΠΤΙΚΗ ΕΡΓΑΖΟΜΕΝΗ ΜΝΗΜΗ ΚΑΙ ΟΠΤΙΚΟΧΩΡΙΚΗ ΑΝΤΙΛΗΨΗ 12
2.4 ΛΟΓΙΣΜΙΚΑ ΕΚΜΑΘΗΣΗΣ ΞΕΝΩΝ ΓΛΩΣΣΩΝ 13
2.4.1 ΕΚΜΑΘΗΣΗ ΓΛΩΣΣΑΣ ΜΕ ΧΡΗΣΗ ΒΙΟΜΕΤΡΙΚΩΝ ΔΕΔΟΜΕΝΩΝ 13
2.4.2 ΠΡΟΒΛΕΨΗ ΕΠΙΠΕΔΟΥ ΜΑΘΗΤΩΝ ΣΤΑ ΑΓΓΛΙΚΑ 13
2.4.3 ΛΟΓΙΣΜΙΚΟ SUPERMEMO 13
2.4.4 ΛΟΓΙΣΜΙΚΟ XEROPAN 14
2.4.5 ΛΟΓΙΣΜΙΚΟ LOQU8 14
2.4.6 ΛΟΓΙΣΜΙΚΟ NEUROCHAT 14
2.4.7 ΑΞΙΟΛΟΓΗΣΗ ΛΕΞΙΛΟΓΙΟΥ ΜΕ ΦΩΝΗΤΙΚΑ ΔΕΔΟΜΕΝΑ 15

3. ΤΕΧΝΟΛΟΓΙΚΟ ΥΠΟΒΑΘΡΟ ΕΦΑΡΜΟΓΗΣ 16

3.1 Τεχνητη Νοημοσυνη 16
3.1.1 Τι ειναι η τεχνητη νοημοσυνη; 16
3.1.2 Ιστορικη Αναδρομη Τεχνητης Νοημοσυνης 16
3.2 Μηχανικη Μαθηση 17
3.2.1 Εισαγωγη και ορισμος της Μηχανικης Μαθησης 17
3.2.2 Κατηγοριες Μηχανικης Μαθησης 17
3.2.2.1 Επιβλεπομενη Μαθηση (Supervised Learning) 18
3.2.2.2 Μη Επιβλεπομενη Μαθηση (Unsupervised Learning) 18
3.2.2.3 Ενισχυμενη Μαθηση (Reinforcement Learning) 19
3.2.2.4 Ημι-επιβλεπομενη Μαθηση (Semi-Supervised Learning) 19
3.2.3 Επιβλεπομενη Μαθηση σε βαθος - Κατηγοριοποιηση (Classification) 20
3.2.3.1 Δεδομενα Μοντελου 20
3.2.3.2 Επιλογη Καταλληλου Αλγοριθμου 21
3.2.4 Μεθοδοι Κατηγοριοποιησης - Classification Algorithms 22
3.2.4.1 Δεντρα Αποφασης – Decision trees 22
3.2.4.2 Τυχαια Δαση - Random Forest 23
3.2.4.3 Λογιστικη Παλινδρομηση – Logistic regression 23
3.2.4.4 K-Πλησιεστεροι Γειτονες (K-Nearest Neighbors - KNN) 25
3.2.4.5 Μηχανες Υποστηριξης Διανυσματων (Support Vector Machine - SVM) 26
3.2.5 Αξιολογηση Μοντελου Κατηγοριοποιησης 27
3.2.5.1 Ακριβεια (Accuracy) 27
3.2.5.2 Πινακας Συγχυσης (Confusion matrix) 27
3.2.5.3 Πιθανολογικη Κατηγοριοποιηση - Probabilistic Classification 28
3.3 Tkinter 28
3.3.1 Η Python ως γλωσσα προγραμματισμου 28
3.3.2 Tkinter και Python 28
3.3.3 Βασικα στοιχεια (Widgets) 28
3.3.4 Διαχειριση Διαταξης (Geometry managers) 29
3.3.5 Event - Προγραμματισμος 30
3.3.5.1 Ποια ειναι τα γεγονοτα (events); 30
3.3.5.2 Πως συνδεουμε ενα γεγονος με καποια ενεργεια; 30
3.3.5.3 Πως παραμενει ενεργη η εφαρμογη; 30
3.4 Αλγοριθμος Johnson – Trotter 31
3.4.1 Ορισμος 31
3.4.2 Παραδειγμα 31
3.4.3 Αποδοτικοτητα Αλγοριθμου 33

4.ΕΦΑΡΜΟΓΗ ΚΑΙ ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ 34

4.1 Εισαγωγη 34
4.2 Δεδομενα 34
4.2.1 Ερωτησεις Δοκιμασιων – questions.json 35
4.2.2 Στοιχεια Μαθητων–student_results.json 36

4.2.3 Βασεις Δεδομενων για καθε δοκιμασια 36
4.2.3.1 Λεξικολογικα Δεδομενα – Vocabulary 37
4.2.3.2 Δεδομενα Μνημης – Memory 39
4.2.3.3 Δεδομενα Οπτικων Μοτιβων – Visual Pattern 40
4.2.3.4 Δεδομενα Go/No-Go 42
4.2.3.5 Τελικη Κατηγοριοποιηση Overall Classification 43
4.2.3.6 Χρηση Encoder για json αρχεια 44
4.3 Έναρξη Εφαρμογης 44
4.3.1 Βιβλιοθηκες που χρησιμοποιουνται 45
4.3.2 Προετοιμασια των Widget 45
4.4 Λειτουργια Μαθητη 47
4.4.1 Εισαγωγη στοιχειων 47
4.5 Έναρξη δοκιμασιων 49
4.5.1 Λειτουργια Παραδειγματων 50
4.5.2 Λειτουργια της δοκιμασια Λεξιλογιου – Vocabulary 51
4.5.3 Λειτουργια της δοκιμασια Μνημης – Memory 53
4.5.3.1 Δημιουργια βασικων στοιχειων Widget 54
4.5.3.2 Εμφανιση Ακολουθιας Memory 54
4.5.3.3 Διαχειριση απαντησης Memory 55
4.5.4 Λειτουργια της δοκιμασια Οπτικα Μοτιβα – Visual Patterns 60
4.5.5 Λειτουργια της δοκιμασια Go/No-Go 60
4.6 Μοντελα Κατηγοριοποιησης - Classifiers 63
4.6.1 Μοντελα Δοκιμασιων 63
4.6.1.1 Αποτελεσματικοτητα Κατηγοριοποιητων 65
4.6.2 Τελικος Κατηγοριοποιητης 67
4.6.2.1 Αποτελεσματικοτητα Τελικου Κατηγοριοποιητη 67
4.7 Αποθηκευση Στοιχειων Μαθητη 68
4.8 Επιπεδο Εκπαιδευτικου 69
4.8.1 Πινακας Ελεγχου Αποτελεσματων 70
4.8.1.1 Αριστερο Πανελ 70
4.8.1.2 Κεντρικο Πανελ 71
4.8.1.3 Δεξι Πανελ 72

5.ΕΚΤΕΛΕΣΗ ΕΦΑΡΜΟΓΗΣ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ 75

5.1 Εκτελεση Εφαρμογης 75
5.1.1 Αρχικη Οθονη 75
5.1.2 Επιπεδο Μαθητη 75
5.1.3 Παραδειγμα Λεξιλογιου 76
5.1.4 Ερωτησεις Λεξιλογιου 76
5.1.5 Τελος Ερωτησεων Λεξιλογιου 77
5.1.6 Παραδειγμα Μνημης 77
5.1.7 Ερωτησεις Μνημης 78
5.1.8 Επομενο Επιπεδο σε Δοκιμασια Μνημης 79
5.1.9 Τελος δοκιμασιας Μνημης 79
5.1.10 Παραδειγμα Οπτικων Μοτιβων 80
5.1.11 Ερωτησεις Οπτικων Μοτιβων 80

5.1.12 Τελος Ερωτησεων Οπτικων Μοτιβων 81
5.1.13 Παραδειγμα Go/No-Go 81
5.1.14 Δοκιμασια Go/No-Go 82
5.1.15 Τελος δοκιμασιας Go/No-Go 82
5.1.16 Τελος δοκιμασιων 83
5.1.17 Επιπεδο Εκπαιδευτικου 83
5.1.18 Πινακας Αποτελεσματων 84
5.2 Μελλοντικες Προεκτασεις 88

ΒΙΒΛΙΟΓΡΑΦΙΑ 89

ΕΙΚΟΝΕΣ 91

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

1.1 Σκοπός της Πτυχιακής

Η παρούσα εργασία έχει ως σκοπό να μελετήσει εργαλεία λογισμικού που βασίζονται στις ανθρώπινες γνωστικές λειτουργίες για την ανίχνευση της ποιότητας της μαθησιακής διαδικασίας. Για αρχή η εργασία εμβαθύνει στο θεωρητικό υπόβαθρο, αναλύοντας τις βασικές γνωστικές λειτουργίες που εμπλέκονται στη μάθηση και παρουσιάζονται σύγχρονες εφαρμογές και λογισμικά που βασίζονται σε αυτές τόσο για την μάθηση γενικώς όσο και για την εκμάθηση μιας ξένης γλώσσας ειδικότερα. Επιπλέον, διερευνάται και αναλύεται η έννοια της Τεχνητής Νοημοσύνης και ακόμη περισσότερο της Μηχανικής μάθησης, καθώς πλέον πέρα από το ότι υπάρχουν παντού, χρησιμοποιούνται και σε αρκετές εφαρμογές μαθησιακού περιεχομένου.

Στο πλαίσιο της εργασίας, αναπτύχθηκε λογισμικό για την αξιολόγηση του επιπέδου εκμάθησης της αγγλικής γλώσσας, το οποίο περιέχει τέσσερεις δοκιμασίες που η καθεμία εστιάζει στην αξιολόγηση διαφορετικής γνωστικής λειτουργία. Συγκεκριμένα, υπάρχουν ερωτήσεις λεξιλογικού και γραμματικού περιεχομένου, οι οποίες ελέγχουν τις γλωσσικές δεξιότητές του μαθητή. Έπειτα ακολουθούν ερωτήσεις συμπλήρωσης οπτικών μοτίβων που εξετάζουν την Οπτικοχωρική Αντίληψη του μαθητή. Επιπλέον, εξετάζεται η Οπτική Εργαζόμενη Μνήμη μέσω δοκιμασίας απομνημόνευσης ακολουθιών αριθμών. Τέλος, ακολουθεί δοκιμασία η οποία εξετάζει τον ανασταλτικό έλεγχο του μαθητή μέσω ενός Go/No-Go τεστ. Η εφαρμογή με βάση τις απαντήσεις του μαθητή κάθε φορά και με χρήση Τεχνητής Νοημοσύνης παρέχει ένα τελικό αποτέλεσμα τόσο για την κάθε δοκιμασία ξεχωριστά όσο και για την τελική του απόδοση κι έπειτα κατηγοριοποιεί τον μαθητή ως «Κάτω του Μέσου Όρου» (Below Average) , «Εντός του Μέσου Όρου» (Average) ή «Άνω του Μέσου Όρου» (Above Average). Είναι σημαντικό να τονιστεί ότι η εφαρμογή δεν αποτελεί διαγνωστικό εργαλείο, αλλά σχεδιάστηκε για να λειτουργεί ως βοηθητικό μέσο για τον εκπαιδευτικό, προσφέροντας μια βαθύτερη κατανόηση του γνωστικού προφίλ του κάθε μαθητή.

1.2 Δομή πτυχιακής

- **Κεφάλαιο 1:** Αποτελεί το εισαγωγικό μέρος της εργασίας, όπου αναφέρεται ο βασικός της σκοπός της πτυχιακής, και περιγράφεται εν συντομία η λειτουργία της εφαρμογής που αναπτύχθηκε στο πλαίσιο της πτυχιακής.

- **Κεφάλαιο 2:** Αναφέρεται ολοκληρωτικά στο θεωρητικό υπόβαθρο πάνω στο οποίο έχει βασιστεί η εφαρμογή. Συγκεκριμένα, περιγράφονται η αγγλική γλώσσα και τα χαρακτηριστικά της (**2.1**), η ελληνική και οι διαφορές τις από τα αγγλικά (**2.2**), οι γνωστικές λειτουργίες και πως αυτές βοηθούν στην εκμάθηση μιας νέας γλώσσας (**2.3**) και τέλος, αναφέρονται παραδείγματα από άλλες μελέτες ή εφαρμογές που έχουν παρόμοια λειτουργικότητα. Δηλαδή, εφαρμογές που βασίζονται στη χρήση των γνωστικών λειτουργιών για την εκμάθηση γενικώς (**2.4**).

- **Κεφάλαιο 3**: Επικεντρώνεται στο τεχνολογικό υπόβαθρο στο οποίο βασίζεται η εφαρμογή, καθώς αναφέρει βασικούς ορισμούς και λειτουργίες της Τεχνητής Νοημοσύνης **(3.1)**, εμβαθύνει περισσότερο στην έννοια της Μηχανικής Μάθησης **(3.2)** κι έπειτα αναφέρεται στις επιμέρους κατηγορίες της. Αναφέρεται στη δομή και τη λειτουργία της βιβλιοθήκης Tkinter η οποία χρησιμοποιείται στην εφαρμογή **(3.3)** και

επίσης γίνεται μια εισαγωγή στον αλγόριθμο Johnson-Trotter και εκτελείται ένα παράδειγμα του **(3.4).**

- **Κεφάλαιο 4**: Στο κεφάλαιο αυτό γίνεται εκτενής εισαγωγή στη γενική λειτουργία της εφαρμογής **(4.1)** κι έπειτα στα δεδομένα που χρησιμοποιεί η εφαρμογή **(4.2).** Σε όλο το υπόλοιπο κεφάλαιο αναλύεται ο κώδικας από την έναρξη της εφαρμογής **(4.3)**, στην λειτουργία του μαθητή **(4.4)**, στις δοκιμασίες που υπάρχουν **(4.5)**, τα μοντέλα κατηγοριοποίησης **(4.6)**, τον τρόπο αποθήκευσης των δεδομένων **(4.7)** μέχρι και το επίπεδο του εκπαιδευτικού **(4.8)**.

- **Κεφάλαιο 5:** Επιδεικνύεται μια τυπική εκτέλεση της εφαρμογής με εικόνες **(5.1)** και αναφέρονται μελλοντικές επεκτάσεις που θα μπορούσαν να πραγματοποιηθούν **(5.2)**.

2. Θεωρητικό Υπόβαθρο Εφαρμογής

2.1 Η Αγγλική Γλώσσα

2.1.1 Εισαγωγή και Ιστορικά Αγγλικής Γλώσσας

Στις ημέρες μας η αγγλική γλώσσα, πέρα από απλώς μια ξένη γλώσσα μπορεί να χαρακτηριστεί και ως παγκόσμια καθομιλουμένη. Η κυριαρχία της βασίζεται στην ενσωμάτωση της από τομείς πολιτικής, οικονομίας, καθώς και επικοινωνίας, ενώ υπάρχουν και τομείς όπως αυτός της Πληροφορικής που βασίζεται αποκλειστικά σε αυτήν. Επιπλέον, η εκπαίδευση, τα μέσα μαζικής ενημέρωσης καθώς και τα κοινωνικά μέσα δικτύωσης, επιτρέπουν ακόμη περισσότερο στη αγγλική γλώσσα να ανθίσει, επιτρέποντας της να λειτουργήσει ως γέφυρα συνεννόησης μεταξύ διαφορετικών πληθυσμιακών ομάδων. Επομένως, σήμερα η εκμάθηση και η διδασκαλία τις καθίσταται κρίσιμο και αναπόσπαστο κομμάτι της σύγχρονης κοινωνίας. (Crystal, English as a global language, 2003)

Η αγγλική γλώσσα έχει έντονες γερμανικές ρίζες από τον 5ο αιώνα και έχει δεχτεί ορισμένες επιρροές από προϋπάρχουσες κελτικές διαλέκτους της Βρετάνιας. Σήμερα, η γλώσσα αυτή δεν θυμίζει καθόλου τα γερμανικά, καθώς ποσοστό ύψους 85% των λέξεων της Παλαιάς Αγγλικής γλώσσας πλέον δεν χρησιμοποιείται. Με μία πρώτη ματιά, τα γράμματα της τότε, ήταν αρκετά όμοια με αυτά που γνωρίζουμε σήμερα, ωστόσο σήμερα έχουν καταρριφθεί κάποια πιο περίεργα σύμβολα, κάνοντας την να μοιάζει λιγότερο σύνθετη. Το λεξιλόγιο της, ήταν απλό και βόλευε στη λειτουργία σύνθετων λέξεων οι οποίες ήταν εύκολο να κατανοηθούν όταν διαχωρίζονταν τα συνθετικά τους. Τέλος η γραμματική της Παλαιάς Αγγλικής γλώσσας, φαίνεται να ήταν κλιτή, δηλαδή οι καταλήξεις των όρων που χρησιμοποιούσαν αλλάζαν αναλόγως με τη λειτουργία τους μέσα στη πρόταση, κάτι που θυμίσει έντονα τα Αρχαία Ελληνικά. (Crystal, The Cambridge encyclopedia of the English language, 2018)

2.1.2 Χαρακτηριστικά Μοντέρνας Αγγλικής Γλώσσας

Η Αγγλική γλώσσα που γνωρίζουμε σήμερα, αν και έχει ρίζες γερμανικές όπως προαναφέρθηκε, το λεξιλόγιο της έχει διαμορφωθεί σε μεγάλο βαθμό από γλώσσες, όπως τα Λατινικά και τα Γαλλικά, ενώ ταυτόχρονα κυριαρχούν έντονα τα Ελληνικά σε επιστημονικούς και τεχνολογικούς όρους της. Επειδή έχει επηρεαστεί τόσο έντονα από άλλες γλώσσες, έχει ενστερνιστεί και ξένους όρους αυξάνοντας το λεξιλογικό της πλούτο με πολλαπλούς συνώνυμους όρους (π.χ. ask, inquire). Ως παγκόσμια γλώσσα, δίνει επιτυχημένα ιδιαίτερη έμφαση στην επικοινωνιακή της αποτελεσματικότητα. Γι' αυτό συχνά απλοποιείται ώστε να είναι κατανοητή από όλους, λειτουργώντας ως ένα εξαιρετικά πρακτικό εργαλείο επικοινωνίας μεταξύ ανθρώπων με διαφορετικό γλωσσικό υπόβαθρο. Ακόμη, τα αγγλικά θα μπορούσαν να θεωρηθούν ιδιαίτερα δημιουργικά και ευέλικτα, καθώς επιτρέπουν τη μετατροπή ενός μέρους του λόγου, σε κάποιο άλλο χωρίς την προσθήκη επιθέματος. Για παράδειγμα, το ουσιαστικό 'house' γίνεται ρήμα «to house a cat», ενώ το ρήμα 'laugh' γίνεται ουσιαστικό «have a good laugh». Ωστόσο η ευελιξία της, μπορεί να αποφέρει και κάποιες πολυπλοκότητες. (Crystal, D., Potter, S., 2025)

2.1.2.1 Χαρακτηριστικά Λεξιλογίου

Ενώ η αγγλική γλώσσα λόγω της προέλευσης της θεωρείται πλούσια σε λεξιλόγιο, αυτό προκαλεί αρκετές πολυπλοκότητες στους αναγνώστες της. Αυτή η ποικιλία δημιουργεί συγχύσεις στους μαθητές της, καθώς καλούνται να διαχειριστούν λέξεις με πολλαπλές σημασίες και λεπτές εννοιολογικές διαφορές (π.χ. transformation, metamorphosis). Μια ιδιαιτερότητα κατά την εκμάθηση της γλώσσας είναι τα ομόφωνα και τα ομόγραφα. Τα ομόφωνα, είναι οι όροι που προφέρονται με τον ίδιο τρόπο, αλλά έχουν διαφορετική γραφή και σημασία (πχ. right, write), ενώ τα ομόγραφα οι λέξεις που γράφονται το με τον ίδιο τρόπο, αλλά έχουν διαφορετική προφορά ή σημασία (πχ. lead = ηγούμαι, μόλυβδος). Οι έννοιες αυτές εμποδίζουν την κατάκτηση προχωρημένου λεξιλογίου καθώς απαιτούν βαθιά κατανόηση του εκάστοτε γλωσσικού πλαισίου για την ορθή χρήση της λέξης. (Haqnawaz, H., Naeem, N., & Khan, S, 2024)

Μια ακόμη βασική δυσκολία της γλώσσας αποτελεί η πληθώρα και εκτεταμένη χρήση φραστικών ρημάτων (π.χ. give up). Πέρα από το ότι συχνά υπάρχουν πολλά όμοια φραστικά ρήματα που μπορεί να προκαλέσουν δυσκολία (πχ. take out, take off, take in, take away), η σημασία τους είναι συχνά ιδιωματική και δεν μπορεί να κατανοηθεί εάν η φράση διασπαστεί στα επιμέρους μέρη τους (πχ. take off, απογείωση).

Η δυσκολία αυτή ωστόσο , δεν αποτελεί μόνο λεξιλογικό πρόβλημα, αλλά βασίζεται στη στενή σχέση μεταξύ λεξιλογίου και γραμματικής που έχει η γλώσσα, προσδίδοντας της ένα από τα πιο έντονα της χαρακτηριστικά, τον έντονο «λεξικογραμματικό» της χαρακτήρα. Σύμφωνα με την έννοια αυτή, η γραμματική δεν είναι ένα απλό σύνολο κανόνων που εφαρμόζεται πάνω στις λέξεις. Αντίθετα, ορισμένες από αυτές είναι άρρηκτα συνδεδεμένες με συγκεκριμένες γραμματικές δομές, δημιουργώντας λέξεις με δική τους αυτόνομη σημασία. (Douglas Biber, Stig Johansson, Geoffrey Leech, Susan Conrad, & Edward Finegan, 1999)

2.1.2.2 Χαρακτηριστικά Γραμματικής

Το γραμματικό επίπεδο της Αγγλικής γλώσσας θα μπορούσε να θεωρηθεί κατά μια έννοια απλό, καθώς δεν είναι κλιτή, αλλά περισσότερο αναλυτική. Βασίζεται, δηλαδη, στη σειρά που είναι τοποθετημένες οι λέξεις (Υποκείμενο - Ρήμα - Αντικείμενο) και στη χρήση προθέσεων και βοηθητικών ρημάτων. Σε αντίθεση με το παρελθόν της, όπου βασιζόταν κατά πολύ περισσότερο στη γερμανική γλώσσα, και ήταν κλιτή με περισσότερες πτώσεις, πλέον κατέχει μόνο δυο βασικές πτώσεις, μία κοινή χωρίς κάποιον δείκτη και μια γενική με δείκτη (πχ. the girl's hair).

Παρά την απλότητα αυτή όμως, δεν παύει να χαρακτηρίζεται από πολυπλοκότητες. Μια από αυτές, που συχνά παραβλέπεται, είναι η συστηματική διαφορά μεταξύ της γραπτής και της προφορικής γραμματικής. Για παράδειγμα, ο γραπτός λόγος σε ένα συγγραφικό έργο ή άρθρο είναι δομικά πολύπλοκος και περιέχει περίτεχνες τεχνικές γραμματικής, ενώ ο καθημερινός προφορικός λόγος είναι δομικά πιο απλός, με μικρότερες προτάσεις όντας πιο «κατακερματισμένος». (Douglas Biber, Stig Johansson, Geoffrey Leech, Susan Conrad, & Edward Finegan, 1999)

Ενώ φαινομενικά η Αγγλική γλώσσα φαίνεται να ορίζεται από κάποιους συγκεκριμένους κανόνες γραμματικής, παρατηρούμε ότι αυτοί καταρρίπτονται σε πολλαπλές περιπτώσεις από ανωμαλίες και εξαιρέσεις, οι οποίες βασίζονται στις πολυάριθμες της ιστορικές επιρροές. Χαρακτηριστικό παράδειγμα αποτελούν τα ανώμαλα ρήματα, όπου ο αόριστος και η μετοχή δεν ακολουθούν έναν ορισμένο κανόνα (πχ. be, was/were, been), καθώς και ο πληθυντικός ορισμένων ουσιαστικών που δεν ακολουθούν τον κανόνα του τελικού «-s» (πχ. tooth/ teeth). Επιπλέον, η γραμματική περιπλέκεται από την χρήση ιδιωματικών εκφράσεων, οι οποίες δεν υπακούν σε καθιερωμένους γραμματικούς κανόνες, κάνοντας την εκμάθηση αλλά και την κατανόηση τους ιδιαίτερα απαιτητική (πχ. Get out of hand). (Haqnawaz, H., Naeem, N., & Khan, S, 2024)

2.1.2.3 Χαρακτηριστικά Ορθογραφίας και Φωνολογίας

Ίσως η πιο αναγνωρίσιμη πολυπλοκότητα της αγγλικής γλώσσας είναι η «βαθιά ορθογραφία» της, η οποία δημιουργεί μια έλλειψη φωνητικής συνέπειας μεταξύ γραπτού και προφορικού λόγου. Αυτό συμβαίνει διότι η αγγλική ορθογραφία δεν ακολουθεί πιστά τη φωνημική αρχή, δηλαδή ένας ήχος δίνει συγκεκριμένο σύμβολο, όπως για παράδειγμα στην Ελληνική γλώσσα όπου το «π» ακούγεται το ίδιο σε κάθε λέξη που συναντάται. Αντιθέτως, η Αγγλική δίνει προτεραιότητα στην μορφολογική αρχή, δηλαδή μπορεί δύο λέξεις να έχουν κοινή ρίζα, και παρόμοια σημασία αλλά η προφορά τους ενδέχεται να αλλάξει, δίνοντας έτσι προτεραιότητα στη διατήρηση της οπτικής ρίζας της λέξης (πχ. magic, magician). Αυτή η αρχή της αγγλικής γλώσσας καθιστά ναι μεν την ορθογραφία της ιστορική και ετυμολογική, αλλά ταυτόχρονα «αδιάφανη» για τον μαθητή. (Douglas Biber, Stig Johansson, Geoffrey Leech, Susan Conrad, & Edward Finegan, 1999)

Ένα παράδειγμα, ακόμη, της διαφοράς προφορικού και γραπτού λόγου είναι η χρήση συστολή του λόγου (contractions). Γλωσσικοί όροι όπως «I'm» αντί για «I am» ή «don't» αντί για «do not» είναι εξαιρετικά συνηθισμένοι στην καθομιλουμένη, αλλά αποφεύγονται αυστηρά στον επίσημο ακαδημαϊκό και επιστημονικό λόγο, αποδεικνύοντας ξανά ότι ο γραπτός και ο προφορικός λόγος διαφέρουν. (Douglas Biber, Stig Johansson, Geoffrey Leech, Susan Conrad, & Edward Finegan, 1999)

Πολλές φορές ακόμη, δυσκολία για έναν μαθητευόμενο αποτελεί και η πληθώρα διαλέκτων και προφορών των αγγλικών. Ακριβώς επειδή ομιλείται από πολλούς, υπάρχουν και πολλές διάλεκτοι (Βρετανική, Αμερικάνικη, Αυστραλιανή) και πολλές φορές υπάρχουν μικρές διαφορές στο λεξιλόγιο και την ορθογραφία στις διαλέκτους αυτές (πχ. vacation /holidays, colour/ color). (Haqnawaz, H., Naeem, N., & Khan, S, 2024) (Crystal, D., Potter, S., 2025).

2.2 Η Ελληνική Γλώσσα

2.2.1 Εισαγωγή και Ιστορικά Ελληνικής Γλώσσας

Η Ελληνική γλώσσα κατέχει μια ξεχωριστή θέση μεταξύ των γλωσσών της Ευρώπης, καθώς διαθέτει την μεγαλύτερη καταγεγραμμένη ιστορία της ηπείρου, ενώ ταυτόχρονα, υπήρξε από τις πρώτες γλώσσες που μελετήθηκαν επιστημονικά ως προς την ιστορία της. Μαθαίνοντας και κατανοώντας την ελληνική γλώσσα σε βάθος, μπορεί κανείς να έρθει πιο κοντά με την πλούσια ιστορία και τον πολιτισμό της ευρύτερης περιοχής της Ευρώπης, αφού η ίδια έχει διαδραματίσει σημαντικό ρόλο στη σύνθεση του. Αξίζει να αναφερθεί ακόμη, ότι η ελληνική έχει συνεισφέρει δανείζοντας μεγάλο εύρος όρων σε άλλες γλώσσες και κυρίως σε επιστημονικούς τομείς όπως αυτός της Βιολογίας, της Ιατρικής, και της Τεχνολογίας. (Βελούδης, 2001)

Η ιστορία της Ελληνικής γλώσσας ξεκινά με την Γραμμική Β', ένα συλλαβικό σύστημα που χρησιμοποιήθηκε κατά τη Μυκηναϊκή εποχή. Μετά την κατάρρευση του μυκηναϊκού πολιτισμού ακολούθησαν αιώνες γνωστοί ως «Σκοτεινοί Αιώνες» χωρίς γραπτά στοιχεία. Η ιστορία της σύγχρονης γραφής επισήμως ξεκινά τον 8ο αιώνα, όταν οι Έλληνες ενστερνίστηκαν το σύστημα γραφής των Φοινίκων. Το έως τότε σύστημα των Φοινίκων, περιείχε μόνο σύμφωνα κι οι Έλληνες προχώρησαν στην επαναστατική για την εποχή εκείνη αλλαγή, την προσθήκη φωνήεντων στο αλφάβητο τους δημιουργώντας το πρώτο πλήρες και φωνητικό αλφάβητο ιστορικά. Το αλφάβητο τότε διακρινόταν σε δυο βασικές παραλλαγές, την Ανατολική εκδοχή και την Δυτική εκδοχή, η οποία θυμίζει περισσότερο τα λατινικά. Αργότερα υιοθετήθηκε επίσημα η Ανατολική μορφή η οποία αποτελεί βάση του σημερινού Ελληνικού αλφαβήτου. (Malikouti-Drachman, 2025)

2.2.2 Χαρακτηριστικά Ελληνικής Γλώσσας και η διαφορά της από την Αγγλική

2.2.2.1 Εισαγωγή στα χαρακτηριστικά της Ελληνικής

Η Νέα Ελληνική γλώσσα, όντας μια από τις πρώτες γλώσσες έχει μεγάλη ιστορία και ιδιαίτερα χαρακτηριστικά που ορίζουν τη δομή της. Ένα βασικό στοιχείο της δομής της είναι η έντονα κλιτή της φύση ως γλώσσα, αποδίδοντας μεγαλύτερη σημασία στην γραμματική της (πχ. κλίσεις και γένη ουσιαστικών). Η κλιτή της δομή προσφέρει ευελιξία στην σειρά των λέξεων μέσα σε μια πρόταση. Επιπλέον, η γλώσσα χαρακτηρίζεται από εξαιρετικά έντονη παραγωγικότητα νεών όρων από προϋπάρχουσες ρίζες, εμπλουτίζοντας έτσι το λεξιλόγιο της. Τέλος, αναμφίβολά η γλώσσα έχει διατηρήσει την ιστορική της ορθογραφία, που αντικατοπτρίζει την πλούσια ετυμολογία των λέξεων. (Holton, 2004)

2.2.2.2 Χαρακτηριστικά Λεξιλογίου και διαφορές

Το ελληνικό λεξιλόγιο χαρακτηρίζεται από την ικανότητα του να παράγει πολλαπλούς νέους όρους, σε αντίθεση με το αγγλικό που βασίζεται κυρίως στη γερμανική γλώσσα και έχει δανειστεί πολλούς όρους από τα γαλλικά και τα λατινικά, όπως είδαμε. Στην ελληνική αντικρίζουμε πολλές φορές παράγωγες και σύνθετες λέξεις. Οι πρώτες σχηματίζονται με τη προσθήκη προθεμάτων και επιθεμάτων (πχ. φεύγω, αποφεύγω) και οι δεύτερες προκύπτουν από την ένωση δυο ή περισσότερων λέξεων δημιουργώντας έναν νέο όρο που διατηρεί τη λογική του νοημάτων των συνθετικών του (πχ. μαχαιροπίρουνο). Αυτό το χαρακτηριστικό της Ελληνικής προσδίδει στο λεξιλόγιο της μια καθαρή και κατανοητή δομή, εξασφαλίζοντας παράλληλα ότι η σημασία των νέων λέξεων προκύπτει με μια λογική και παραμένει διακριτή για τον μαθητευόμενο. Αντιθέτως με την αγγλική που οι περισσότεροι της όροι απαιτούν βαθιά κατανόηση για να διαφοροποιηθούν (πχ transformation, metamorphosis). (Holton, 2004)

Για να τονίσουμε την διαφορά των δυο γλωσσών σε επίπεδο παραγωγικότητας, αξίζει να αναφέρουμε τα υποκοριστικά και τα μεγεθυντικά που διαθέτει η ελληνική. Ενώ η αγγλική για να χαρακτηρίσει μεγέθη προσθέτει μεμονωμένα επίθετα στα ουσιαστικά της (πχ little house, enormous town) η ελληνική μπορεί να τροποποιήσει την κατάληξη ουσιαστικών και επιθέτων της για να εκφράσει το μέγεθος αλλά και συναισθηματικό δεσμό προς τον όρο (πχ. αδερφάκι, ομαδάρα). Το συγκεκριμένο χαρακτηριστικό της ελληνικής της προσδίδει μια ιδιαίτερη εκφραστική αμεσότητα, σε αντίθεση με της αγγλικής.

2.2.2.3 Χαρακτηριστικά Γραμματικής και διαφορές

Η ελληνική γλώσσα όπως προαναφέραμε, είναι μια κλιτή γλώσσα, αντιθέτως με την αγγλική η οποία είναι κυρίως αναλυτική. Στα αγγλικά η γραμματική λειτουργία καθορίζεται κυρίως από τη θέση της κάθε λέξης στην πρόταση (π.χ. Υποκείμενο-Ρήμα-Αντικείμενο), ενώ στα ελληνικά οι λέξεις μπορούν να αλλάξουν μορφή πιο ευκολά, λόγω των πολλαπλών πτώσεων της. Συγκεκριμένα, η ελληνική διαθέτει τέσσερεις πτώσεις κι επειδή οι καταλήξεις δηλώνουν με σαφήνεια τον γραμματικό ρόλο κάθε λέξης, αυτή κατέχει μεγάλη ευελιξία στη σειρά των όρων της, επιτρέποντας την αλλαγή θέσης, χωρίς την αλλοίωση του νοήματος της πρότασης (πχ. Ο Κώστας έφαγε την σαλάτα, Την σαλάτα την έφαγε ο Κώστας). Επομένως, η Ελληνική διαθέτει περισσότερες πτώσεις που καθορίζουν με μεγαλύτερη ακρίβεια τη λειτουργία του κάθε όρου, ενώ η αγγλική βασίζεται μονάχα σε δυο, διατηρώντας την απλότητα της.

Περαιτέρω, η κλιτή δομή της ελληνικής εκφράζεται και μέσω των γενών της. Συγκεκριμένα, διαθέτει τρία γραμματικά γένη (αρσενικό, θηλυκό, ουδέτερο), τα οποία είναι πιο αυθαίρετα ορισμένα από τα αντίστοιχα τρία γένη της αγγλική γλώσσας. Δηλαδή, στα αγγλικά χρήση ουδέτερου γένους εφαρμόζεται σε αντικείμενα και ζώα, ενώ το αρσενικό και το θηλυκό όταν πρόκειται για αρσενικά ή θηλυκά όντα αντίστοιχα, ενώ κάτι παρόμοιο δεν ισχύει στα ελληνικά (πχ. The door, η πόρτα). Αυτό θεωρητικά είναι μια πολυπλοκότητα της

ελληνικής γλώσσας, καθώς δεν είναι τόσο ξεκάθαροι οι κανόνες από τους οποίους ορίζεται το γένος ενός όρου. (Holton, 2004)

Μια ακόμη λεπτή διαφορά ανάμεσα στις δύο γλώσσες, είναι η παρουσία κλιτικού συστήματος των ρημάτων σε σύγκριση της αγγλικής που υστερεί. Για παράδειγμα, στα ελληνικά για κάθε πρόσωπο και αριθμό του ρήματος η κατάληξη διαφέρει, επιτρέποντας μας έτσι να παραλείψουμε το υποκείμενο που αφορά το ρήμα (πχ. Εγώ γράφω, γράφω.). Ενώ στα αγγλικά, κάτι όμοιο δεν είναι εφικτό, καθώς η γλώσσα δεν διαθέτει τόσο ξεκάθαρους διαχωρισμούς μεταξύ των καταλήξεων της (πχ. I/you/we/they write) και η δήλωση του υποκειμένου ρήματος είναι κάθε φορά υποχρεωτική. Αξίζει να σημειωθεί ακόμη, ότι η θέση του υποκειμένου είναι πιο ευέλικτη συγκριτικά με τα αγγλικά, αφού αυτά ορίζουν αυστηρά τη θέση του υποκειμένου πριν το ρήμα, ενώ τα ελληνικά επιτρέπουν την ύπαρξη του και μετά το βασικό ρήμα (πχ.). (Daskalaki, 2019)

2.2.2.4 Χαρακτηριστικά Ορθογραφίας και Φωνολογίας και διαφορές

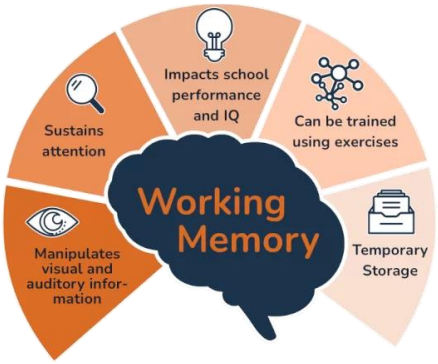
Ένα κεντρικό χαρακτηριστικό της Ελληνικής είναι η ιστορική της ορθογραφία, δηλαδή η γραφή δεν αντιστοιχεί πάντα στην προφορά. Συγκεκριμένα, όταν πρόκειται για την ανάγνωση, η αναγνώριση των όρων δεν αποτελεί δυσκολία, δηλαδη οποιοδήποτε από τα «η», «ι», «υ», «οι», «ει», «υι» διαβάζονται πάντα με τον ίδιο τρόπο. Η δυσκολία της Ελληνικής όμως, έρχεται όταν απαιτείται η γραφή των όρων αυτών, ακριβώς επειδή ένα φωνήεν που ακούγεται ίδιο σε κάθε λέξη, μπορεί να γραφτεί με πολλαπλούς τρόπους. Στα αγγλικά, από την άλλη, τόσο η γραφή όσο και η ανάγνωση διαθέτουν πολυπλοκότητα. Συγκεκριμένα, κατά την ανάγνωση ενός όρου, αυτός συχνά έχει διαφορετική προφορά από λέξη σε λέξη (πχ. rough, through). Το ίδιο πρόβλημα συνεχίζει να υπάρχει και κατά την γραφή, καθώς πολλές φορές η προφορά μια λέξης δεν συνάδει με την γραπτή της μορφή (πχ queue). Συνεπώς, η ελληνική είναι περισσότερο διαφανή συγκριτικά με την αγγλική, καθώς δεν υπάρχει πολυπλοκότητα κατά την ανάγνωση.

2.3 Γνωστικές Λειτουργίες

Η εκμάθηση μιας ξένης γλώσσας δεν αρκείται μόνο στην ανάπτυξη πλούσιων γλωσσικών δεξιοτήτων, απαιτεί και την χρήση των γνωστικών μας λειτουργιών. Οι λειτουργίες αυτές βρίσκονται κυρίως στο μπροστινό μέρος του εγκεφάλου, συγκεκριμένα στον μετωπιαίο και στον προμετωπιαίο λοβό και είναι υπεύθυνες για την ανάπτυξη της ικανότητας να μαθαίνουμε νέα στοιχεία και να εξελισσόμαστε με βάση αυτά. Οι ίδιες είναι υπεύθυνες για την εξέλιξη και πρόοδο μας μέσω της μάθησης καθ' όλη τη διάρκεια της ανάπτυξης μας ως όντα. Παρακάτω, θα επικεντρωθούμε σε ορισμένες γνωστικές λειτουργίες. (Puchta, 2020)

2.3.1 Εργαζόμενη Μνήμη (Working Memory)

Η εργαζόμενη μνήμη αποτελεί μια από τις βασικότερες εκτελεστικές λειτουργίες, καθώς μας επιτρέπει να κρατάμε ενεργές τις πληροφορίες στο μυαλό μας και ταυτόχρονα να τις επεξεργαζόμαστε όταν χρειάζεται. Αναλόγως με το είδος της πληροφορίας, διακρίνεται σε λεκτική και οπτικοχωρική μνήμη. Η σημασία της είναι καθοριστική για την αντίληψη κάθε διαδικασίας που μας συμβαίνει. Για παράδειγμα, η λειτουργία της κατανόησης λόγου, η οποία ανάγεται στη λεκτική μνήμη, μας επιτρέπει να συνδέουμε τα προηγούμενα που είδαμε και μάθαμε με τα επόμενα που ακολουθούν. Αξίζει να τονιστεί ότι η εργαζόμενη μνήμη είναι σημαντική για την ανάπτυξη της λογικής σκέψης και της δημιουργικότητας, καθώς μας επιτρέπει να εντοπίζουμε σχέσεις μεταξύ διαφορετικών εννοιών. (Diamond, 2013)



Εικόνα 2.3: Εργαζόμενη Μνήμη

2.3.2 Οπτική Εργαζόμενη Μνήμη (Visual Working Memory)

Η οπτική εργαζομένη μνήμη είναι το κομμάτι οπτικοχωρικής μνήμης της εργαζόμενης μνήμης και αποτελεί έναν νοητικό μηχανισμός με περιορισμένες δυνατότητες ο οποίος επιτρέπει στο άτομο να διατηρεί ζωντανή την εικόνα την αντικειμένων που τον ενδιαφέρουν κάθετη στιγμή στο μυαλό του. Δεν διατηρεί μόνο θέση ενός αντικειμένου, αλλά το σχήμα, το χρώμα καθώς και πληροφορίες για την υφή του. Η χωρητικότητα της είναι αρκετά μικρή, καθώς οι περισσότεροι άνθρωποι με τυπική ανάπτυξη μπορούν να συγκρατήσουν ταυτόχρονα και με ικανοποιητική λεπτομέρειά την οπτική πληροφορία για τρία έως και τέσσερα διαφορετικά αντικείμενα το πολύ. (Ware, 2013)

2.3.3 Οπτικοχωρική Αντίληψη (Visuospatial function)

Η οπτικοχωρική αντίληψη περιγράφει τις λειτουργίες που επιτρέπουν σε ένα άτομο να ερμηνεύει και να κατανοεί τον οπτικό κόσμο γύρω του. Πιο συγκεκριμένα, αφορά την ικανότητα του εγκεφάλου να αναγνωρίζει σχήματα, να αντιλαμβάνεται τις μεταξύ τους δομές και λεπτομέρειες. Η συγκεκριμένη, είναι σημαντική για τις βασικές δραστηριότητες ενός ατόμου, καθώς βοηθά στην ικανότητα του προσανατολισμού. Εάν η οπτικοχωρική αντίληψη ενός ατόμου είναι μειωμένη, αυτό δύναται να φέρει πρακτικές δυσκολίες, όπως αδεξιότητα στις κινήσεις. Τέλος, η οπτικοχωρική επεξεργασία είναι η ευρύτερη ικανότητα να επεξεργάζεται κανείς νοητικά οπτικά ερεθίσματα, συνθέτοντας, αναλύοντας ακόμα και μεταμορφώνοντας τα ίδια τα ερεθίσματα στο μυαλό του. (Wikipedia contributors., 2024)

2.3.4 Inhibition (Go/Nogo)

Ο ανασταλτικός έλεγχος (Inhibition) ορίζεται ως η νοητική ικανότητα να φιλτράρει και να αγνοήσει κανείς ερεθίσματα που στοχεύουν να μας αποπλανήσουν από τον τρέχοντα μας στόχο. Πρόκειται ουσιαστικά για δυο βασικές δεξιότητες, συγκεκριμένα την διατήρηση της συγκέντρωσης κατά την παρουσία των περισπασμών, αλλά και τη ρύθμιση των συναισθημάτων και των πράξεων μας απέναντι σε εξωτερικά ερεθίσματα, με σκοπό την ολοκλήρωση μιας δραστηριότητας. (Puchta, 2020)

2.3.5 Διαφορές Γνωστικών Λειτουργιών που μελετήθηκαν

Οι όροι Οπτική Εργαζόμενη Μνήμη, Οπτικοχωρική Αντίληψη και Αναστολή που είδαμε παραπάνω θα μπορούσαν να προκαλέσουν σύγχυση σε κάποιο που τα μελετά για πρώτη φορά, οπότε παρακάτω αναλύονται οι διαφορές μεταξύ τους. Η Οπτική Εργαζόμενη Μνήμη, αφορά κυρίως την προσωρινή αποθήκευση και επεξεργασία οπτικών ερεθισμάτων για άμεση χρήση, ενώ η Οπτικοχωρική Αντίληψη σχετίζεται με την ικανότητα κατανόησης της θέσης και των σχέσεων των αντικειμένων στο χώρο. Η Αναστολή, από την άλλη, δεν σχετίζεται τόσο με την αναπαράσταση ή αποθήκευση πληροφορίας, αλλά με τον έλεγχο.

Συγκεκριμένα, αναφέρεται στην ικανότητα να αγνοεί κανείς ερεθίσματα που παρεμποδίζουν την εργασία του τη παρούσα στιγμή. Συνεπώς, οι δύο πρώτες λειτουργίες εστιάζουν στην αποκωδικοποίηση και επεξεργασία πληροφοριών, ενώ η τρίτη αφορά τη ρύθμιση συμπεριφοράς κατά την ύπαρξη «εμποδίων».

2.3.6 Γνωστικές λειτουργίες και εκμάθηση ξένων γλωσσών

Αρκετές έρευνες, όπως θα αναφερθεί και παρακάτω, έχουν αποδείξει ότι οι γνωστικές λειτουργίες, δεν προορίζονται μόνο στη γενική νοητική ικανότητα του ανθρώπου, αλλά συνδέονται και για να βοηθούν την διαδικασίας εκμάθησης μιας ξένης Γλώσσας. Η εργαζόμενη μνήμη είναι απαραίτητη για την επεξεργασίας νέων πληροφορίων και συντελεί στη κατανόηση λεξιλογίου και γραμματικής. Η οπτικοχωρική αντίληψη διευκολύνει την αναγνώριση και την απομνημόνευση γραμμάτων. Τέλος, η ικανότητα αναστολής ενισχύει την εστίαση στη νέα γλώσσα παρεμποδίζοντας περισπασμούς η συγχύσεις από τη μητρική.

2.3.6.1 Έρευνα: Εργαζόμενη μνήμη και Αναστολή

Έρευνα που πραγματοποιήθηκε από Linck, Jared & Weiss, Daniel το 2015, εξέτασε εάν οι γνωστικές λειτουργίες της Εργαζόμενης μνήμης και της Αναστολής μπορούν να προβλέψουν την επιτυχία στην εκμάθηση μιας ξένης γλώσσα. Τα αποτελέσματα των ερευνών, έδειξαν ότι η εργαζόμενη μνήμη λειτούργησε επιτυχώς ως δείκτης πρόγνωσης, καθώς οι εκπαιδευόμενοι φοιτητές με καλύτερα αποτελέσματα σε δοκιμασίας της εργαζόμενης τους μνήμης, παρουσίασαν μεγαλύτερη μαθησιακή πρόοδο. Ο ανασταλτικός έλεγχος, από την άλλη, ο οποίος έχει ως βασική λειτουργία την «αναστολή» της μητρικής γλώσσας και την χρήση της νέας ξένης, δεν αναδείχτηκε ιδιαίτερα ως σημαντικός παράγοντας πρόβλεψης. Οι ερευνητές ωστόσο, αναφέρουν ότι αυτό μπορεί να συνέβη, διότι οι μαθητές μάθαιναν την νέα γλώσσα σε αρχάριο στάδιο, και δεν χρειάστηκε να γίνει καταστολή της μητρικής τους γλώσσας. Επομένως, αναμφίβολά η εργαζόμενη μνήμη αποτελεί σημαντικό παράγοντα, και ενδεχομένως να την ακολουθούσε η αναστολή ελέγχου εάν είχαν εμβαθύνει περεταίρω στην εκμάθηση της νέας γλώσσας. (Linck J. &., 2015)

Παράλληλα, μια ακόμη μελέτη που συνθέτει αποτελέσματα 79 ερευνών ανέδειξε εξίσου την σημασία της Εργαζόμενης μνήμης κατά την εκμάθηση μιας δεύτερης γλώσσας. Τα αποτελέσματα έδειξαν ότι ο εκτελεστικός έλεγχος (executive control) βρέθηκε να είναι καθοριστικός σε σύγκριση με την ικανότητα της αποθήκευσης. Επιπλέον, οι λεκτικές μετρήσεις της εργαζόμενης μνήμης φάνηκε να ήταν ισχυρότεροι προγνωστικοί δείκτες σε σύγκριση με τις μη-λεκτικές (οπτικές). (Linck J. A., 2014)

2.3.6.2 Έρευνα: Οπτική Εργαζόμενη Μνήμη και Οπτικοχωρική αντίληψη

Διαφορετική μελέτη των ερευνητών Clare Wright και Jun Wang εξέτασε πως μαθητευόμενοι της Κινέζικης γλώσσας κατανοούν τους οπτικά σύνθετους χαρακτήρες της. Τα αποτελέσματα της έρευνας έδειξαν ότι οι μαθητές θυμόταν πολύ πιο εύκολα τους χαρακτήρες που αντιπροσώπευαν ουσιαστικά, σε σχέση με αυτούς που αντιπροσώπευαν ρήματα. Αυτό δείχνει ότι το νόημα της λέξης ήταν πιο βοηθητικό από το πόσο ευδιάκριτος ήταν οπτικά ο χαρακτήρας. Η έρευνα δεν έδειξε κάποια άμεση σύνδεση μεταξύ της των καλών αποτελεσμάτων οπτικής εργαζόμενης μνήμης και ευκολία εκμάθησης της γλώσσα. Ωστόσο οι ερευνητές δεν την απέρριψαν εντελώς, καθώς θεωρούν ότι η οπτική μνήμη ίσως αρχίζει να παίζει σημαντικό ρόλο αφότου ο μαθητής αποκτήσει μια βασική εξοικείωση με τη γλώσσα. (Wright, C., & Wang, J. , 2023)

2.4 Λογισμικά Εκμάθησης ξένων γλωσσών

Αναμφίβολα, λοιπόν οι βασικές γνωστικές λειτουργίες παίζουν σημαντικό ρόλο στην εκμάθηση μια ξένης γλώσσας, και μπορούμε να δούμε πώς αξιοποιούνται πρακτικά μέσα από σύγχρονα λογισμικά. Τα τελευταία χρόνια έχουν αναπτυχθεί εφαρμογές που στηρίζονται σε αρχές όπως η εργαζόμενη μνήμη, η οπτικοχωρική αντίληψη και ο ανασταλτικός έλεγχος είτε με στόχο να αξιολογήσουν το επίπεδο μάθησης του μαθητή ή ενισχύοντας και διευκολύνοντας την διαδικασία της μάθησης. Ορισμένες από αυτές ενσωματώνουν συχνά και τη χρήση τεχνητής νοημοσύνης για άμεση επιβεβαίωση προόδου. Παρακάτω λοιπόν θα αναπτυχθούμε σε μερικές εφαρμογές που χρησιμοποιούνται για την ανίχνευση της πορείας ενός μαθητή προς την κατάκτηση νέας γλώσσας.

2.4.1 Εκμάθηση Γλώσσας με χρήση Βιομετρικών δεδομένων

Μια ιδιαίτερα ρηξικέλευθη έρευνα στην εκμάθηση γλωσσών (Yuan, 2025), αξιολόγησε την αποτελεσματικότητα μιας πλατφόρμας τεχνητής νοημοσύνης ενισχυμένης με την χρήση βιομετρικών δεδομένων. Η πλατφόρμα σχεδιάστηκε για να βελτιώσει την κατανόηση κειμένου δεύτερης ξένης γλώσσας σε μαθητές με τη χρήση δεδομένων παρακολούθησης βλέμματος (eye-tracking) και των καρδιακών τους παλμών (HRV). Μέσω αυτών, το σύστημα εντόπιζε στιγμές δυσκολίας ή άγχους από τον μαθητή και προσάρμοζε αναλόγως το εκπαιδευτικό του υλικό.

Τα αποτελέσματα της μελέτης έδειξαν σημαντική βελτίωση των συμμετεχόντων στην κατανόηση κειμένου, αλλά και σημαντικά οφέλη σε ψυχολογικό επίπεδο, όπως μειωμένο άγχος και καλύτερη διαχείριση των γνώσεων σε σύγκρισή με την ομάδα μαθητών που δεν χρησιμοποίησαν το συγκεκριμένο λογισμικό. Αξίζει να αναφερθεί ότι τόσο η ανάλυση της HRV όσο και το eye-tracking έδειξαν θετικά αποτελέσματα, καθώς υπήρχε μείωση του στρες, λιγότερες παλινδρομήσεις του βλέμματος και μικρότερη διάρκεια παρακολούθησης άγνωστων λέξεων.

2.4.2 Πρόβλεψη επιπέδου μαθητών στα Αγγλικά

Σε μια ακόμη έρευνα (S. Gkika, 2022), δημιουργήθηκε διαδικτυακή εφαρμογή που ελέγχει το επίπεδο αγγλικών σε μαθητές της τρίτης δημοτικού, με στόχο την κατηγοριοποίηση των μαθητών σε επίπεδο μετρίων και προχωρημένων. Το τεστ περιλάμβανε διάφορες ασκήσεις που εξέταζαν τις γλωσσικές ικανότητες των μαθητών.

Τα αποτελέσματα της ερευνάς έδειξαν ότι το εργαλείο ανέδειξε το επίπεδο των μαθητών με επιτυχία, καθώς όσα θεωρήθηκαν «προχωρημένα» είχαν περισσότερες σωστές απαντήσεις σε όλες τις δοκιμασίες. Η έρευνα έδειξε λοιπόν ότι παρόμοια εκπαιδευτικά εργαλεία είναι πολύ χρήσιμα σε εκπαιδευτικούς, καθώς μπορούν να δώσουν μια αξιόπιστη αξιολόγηση του επιπέδου των μαθητών.

2.4.3 Λογισμικό SuperMemo

Το SuperMemo αποτελεί ένα ακόμη λογισμικό εκμάθησης το οποίο βασίζεται στη λογική των ψηφιακών καρτών, δηλαδή flashcards με ερωτήσεις και απαντήσεις που δημιουργεί ο χρήστης και χρησιμοποιούνται συχνά για την εκμάθηση λεξιλογίου ξένων γλωσσών. Το πρόγραμμα βασίζεται κυρίως στην αραιή επανάληψη, δηλαδή αντί να επαναλαμβάνει μια πληροφορία συνέχεια, μεγαλώνει τα χρονικά διαστήματα ανά τα οποία εμφανίζει επαναληπτικά την πληροφορία. Η εφαρμογή είναι βασισμένη στην ιδέα της καμπύλης της λήθης, η οποία ως έννοια εξηγεί ότι ο εγκέφαλος ξεχνάει πληροφορίες με προβλέψιμο ρυθμό, οπότε το πρόγραμμα προσπαθεί να επαναφέρει την πληροφορία λίγο πριν ξεχαστεί. Η εφαρμογή λοιπόν παρακολουθεί τις απαντήσεις του μαθητή και εάν δείχνει να έχει εμπεδώσει μια γνώση την επαναφέρει μετά από καιρό, ενώ αν παρατηρήσει δυσκολία, την επαναφέρει πιο σύντομα. Ταυτόχρονα η εφαρμογή προσπαθεί να κάνει αυτό-

εξέταση στον μαθητή, δηλαδή προσπαθεί να κάνει τον μαθητή να θυμηθεί μόνος του την σωστή απάντηση, αντί να την διαβάζει επανειλημμένα. (Wikipedia contributors, 2025)

2.4.4 Λογισμικό Xeropan

Μια ακόμη δημοφιλής εφαρμογή εκμάθησής γλωσσών είναι η Xeropan, η οποία βασίζεται στη μάθηση, χρησιμοποιώντας αποσπάσματα από ταινίες και τηλεοπτικές εκπομπές για να διδάξει λεξιλόγιο και γραμματική. Με αυτό τον τρόπο οι χρήστες μαθαίνουν την γλώσσα όπως αυτή χρησιμοποιείται σε καθημερινό επίπεδο. Η μάθηση είναι έντονα παιγνιώδη καθώς οι χρήστες καλούνται να ακολουθήσουν μια ιστορία, συλλέγοντας πόντους και ξεκλειδώνοντας νέα μαθήματα, με τρόπο που παρακινούνται από συστήματα επιβράβευσης, και διατηρούν ενεργή την ενασχόληση τους. Η εφαρμογή εστιάζει στην ανάπτυξη ομιλίας, γραφής, γραμματικής και κατανόησης μέσω των διαδραστικών βίντεο και έναν βοηθό τεχνητής νοημοσύνης που εξηγεί τους κανόνες. (Wikipedia contributors , 2025)

2.4.5 Λογισμικό Loqu8

Ένα ακόμη παράδειγμα τεχνολογίας που στοχεύει στη διευκόλυνση της εκμάθησης είναι το λογισμικό Loqu8. Η βασική λειτουργία της εφαρμογής είναι να παρέχει πληροφορίες ευθύς αμέσως στον χρήστη, την στιγμή που τις χρειάζεται. Στην πράξη, όταν ένας χρήστης διαβάζει ένα κείμενο και τοποθετεί τον κέρσορα του ποντικού πάνω σε μια άγνωστη λέξη εμφανίζεται ένα αναδυόμενο παράθυρο που μεταφράζει και εξηγεί την σημασία της. Η προσέγγιση αυτή ονομάζεται επαυξημένη μάθηση, αλλά δεν αποτελεί εφαρμογή επαυξημένης πραγματικότητας. Ο τρόπος που αυτό βοηθάει τη μάθηση συνδέεται με το πως λειτουργεί ο ανθρώπινος εγκέφαλος. Αντί κάθε φορά που εντοπίζει κάτι άγνωστο να σταματάει για να το ψάξει, η Loqu8 κάνει αυτή τη διαδικασία αυτόματη και χωρίς να αποσπά τη προσοχή του χρήστη. Η εφαρμογή με επιτυχία κάνει τη διαδικασία της αναζήτησης σχεδόν αόρατη και κάνει τη μάθηση πιο άμεση και ομαλή. (Wikipedia contributors, 2025)

2.4.6 Λογισμικό NeuroChat

Σε μια πρόσφατη έρευνα που πραγματοποιήθηκε (Dünya Baradari, 2025) διερευνήθηκε η χρήση της νευροτεχνολογίας και της τεχνητής νοημοσύνης στην εκπαίδευση, αναπτύσσοντας ένα νευρο – προσαρμοστικό σύστημα «NeuroChat». Το σύστημα αυτό λειτουργούσε σε πραγματικό χρόνο χρησιμοποιώντας μια φορητή συσκευή ηλεκτροεγκεφαλογραφρίας (EEG), η οποία μετρούσε συνεχώς το επίπεδο γνωστικής ενασχόλησης του μαθητή. Οι μετρήσεις που λάμβανε η συσκευή αποστέλλονταν σε ειδικό Γλωσσικό Μοντέλο (LLM) το οποίο είχε προγραμματιστεί ώστε να προσαρμόζει τις απαντήσεις του αναλόγως με τη γνωστική κατάσταση του χρήστη. Για παράδειγμα εάν η ενασχόληση του μαθητή ήταν χαμηλή, το σύστημα έδινε απλούστερες εξηγήσεις.

Τα αποτελέσματα της αξιολόγησης, όπου αυτά συγκρίθηκαν με ένα τυπικό AI chatbot έδειξαν ότι ενώ το NeuroChat κατάφερε να αυξήσει την ενασχόληση του χρήστη, αυτό δεν έφερε διαφορά στα μαθησιακά αποτελέσματα σε σύγκριση με το τυπικό chatbot. Συγκεκριμένα, οι συμμετέχοντες της έρευνας αποδείχτηκε από μετρήσεις της συσκευής (EEG) ότι ήταν περισσότερο συγκεντρωμένοι. Ενώ και από τα δύο chatbot δεν υπήρξε σημαντική διαφορά στις επιδόσεις στα τεστ γνώσεων.

2.4.7 Αξιολόγηση Λεξιλογίου με Φωνητικά Δεδομένα

Μια έρευνα των (Wilschut Thomas, 2021) διερεύνησε εάν τα οφέλη που παρατηρούνται στην εκμάθηση λεξιλογίου μέσω πληκτρολόγησης μπορούν να μεταφερθούν και σε ένα περιβάλλον μάθησης που βασίζεται στην ομιλία. Τα περισσότερα λογισμικά συστήματα βασίζονται στην πληκτρολόγηση και χρησιμοποιούν συστήματα υπολογισμού της μνήμης του κάθε μαθητή με βάση τον χρόνο απόκρισής και την ακρίβεια της τυποποιημένης απάντησης.

Το βασικό πόρισμα της μελέτης ήταν ότι η μάθηση λεξιλογίου, λειτουργεί εξίσου αποτελεσματικά όταν βασίζεται και στην ομιλία. Συγκεκριμένα οι συμμετέχοντες που χρησιμοποίησαν το προσαρμοστικό σύστημα ομιλίας πέτυχαν υψηλότερη ακρίβεια κατά 8-10% και ήταν ταχύτεροι στις απαντήσεις σε σύγκριση με μια ομάδα που χρησιμοποίησε μια πιο παραδοσιακή μέθοδο την μέθοδο flashcard (Leitner). Η έρευνα, χρησιμοποίησε τον χρόνο έναρξης της ομιλίας των συμμετεχόντων για να μετρήσει την μνήμη τους και το σύγκρινε με τον αντίστοιχο χρόνο ανταπόκρισης στην πληκτρολόγηση. Τα αποτελέσματα έδειξαν ότι η προφορική εξάσκηση μπορεί να προσφέρει μια πιο βαθιά και εξατομικευμένη μάθηση, εστιάζοντας όχι μόνο στο λεξιλόγιο, αλλά και την προφορά.

Εικόνα 2.4: Online Learning

3. Τεχνολογικό Υπόβαθρο Εφαρμογής

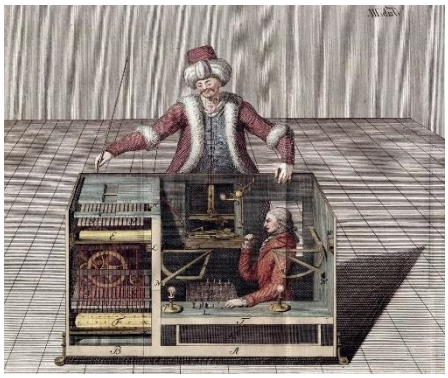
3.1 Τεχνητή Νοημοσύνη

3.1.1 Τι είναι η τεχνητή νοημοσύνη;

Με τον όρο τεχνητή νοημοσύνη, αναφερόμαστε στην επιστημονικό κλάδο που ασχολείται με την δημιουργία έξυπνων υπολογιστικών συστημάτων λογισμικού. Η δημιουργία αυτών έχει εμπνευστεί από τη λειτουργία του ανθρώπινου εγκεφάλου, αλλά δεν έχει στόχο να τις μιμηθεί πλήρως. Πιο συγκεκριμένα, η τεχνητή νοημοσύνη μελετά την ανθρώπινη σκέψη και συμπεριφορά και βασίζεται σε αυτήν για να επιλύσει κάποια προβλήματα, χωρίς όμως αυτό να αποτελεί πιστό κανόνα. Τις περισσότερες φορές καλείται να επιλύσει δύσκολα υπολογιστικά προβλήματα που ξεπερνούν κατά πολύ τις ανθρώπινες δυνατότητες. (McCarthy, 2007)

3.1.2 Ιστορική Αναδρομή Τεχνητής Νοημοσύνης

Η ιδέα για σκεπτόμενες μηχανές έχει ρίζες από τον 17ο αιώνα, όπου στοχαστές όπως ο Leibniz και ο Pascal οραματίστηκαν τις πρώτες μηχανικές συσκευές που θα μπορούσαν να αυτοματοποιήσουν αριθμητικούς υπολογισμούς. Παράλληλα, μπορούμε να δούμε στη μυθολογία και τη λογοτεχνία παρόμοιες ιδέες από τον Όμηρο που έκανε λόγο για μηχανικούς «τρίποδες». Ωστόσο, η επιθυμία για τη δημιουργία μια μηχανής με νοημοσύνη υπήρξε αρκετά έντονη κατά το 18ο αιώνα, όταν ήταν αρκετά δημοφιλή το σκάκι. Τότε πήρε μορφή για πρώτη φορά μια «πανέξυπνη» μηχανή σκακιού *(Εικόνα 3.1)*, η οποία νικούσε σχεδόν κάθε αντίπαλο της, ακόμα και κάποιους διάσημους της εποχής της. Αν και αργότερα αποκαλύφθηκε ότι πίσω από την μηχανή υπήρχε άνθρωπος που την καθοδηγούσε, η τεράστια της φήμη, κατέδειξε το πόσο έντονο ήταν το ενδιαφέρον για τη δημιουργία μηχανής που να επιδεικνύει νοημοσύνη. (Buchanan, 2005)



Εικόνα 3.1: 'The Turk' chess machine

Η μετάβαση από την φαντασία στην επιστήμη, πραγματοποιήθηκε λίγο μετά τον Β' Παγκόσμιο Πόλεμο, με την εμφάνιση των πρώτων σύγχρονων υπολογιστών. Σημαντική επίδραση σε όλο αυτό είχε ο επιστήμονας Alan Turing, μέσω του άρθρου «Computing Machinery and Intelligence», όπου παρουσίασε για πρώτη φορά την διάσημη μέχρι και σήμερα δοκιμασία «Turing Test». Πιο αναλυτικά, κατά την δοκιμασία ένας άνθρωπος «εξεταστής» καλείται να συνομιλήσει γραπτώς ταυτόχρονα με μια μηχανή και με έναν άνθρωπο χωρίς να γνωρίζει την ταυτότητα και των δυο. Εάν ο εξεταστής δεν μπορεί να διακρίνει την μηχανή, τότε αυτή θεωρείται ευφυής. (French, 2000) Από το 1956 και μετά, ωστόσο, η Τεχνητή Νοημοσύνη πλέον υπάρχει και ως επίσημος όρος, χάρη στην διάσκεψη του Dartmouth η οποία θεωρείται η επίσημη αφετηρία του πεδίου. (Buchanan, 2005)

3.2 Μηχανική Μάθηση

3.2.1 Εισαγωγή και ορισμός της Μηχανικής Μάθησης

Αφότου ξεκίνησε να υπάρχει ως όρος η Τεχνητή Νοημοσύνη κι έπειτα, η έννοια της μάθησης θεωρήθηκε ολοένα και πιο σημαντική για την επίτευξη της ευφυΐας των μηχανών. Το πιο χαρακτηριστικό και πρωτοποριακό παράδειγμα, αποτελεί το πρόγραμμα "Samuel's Checkers Player" του Arthur Samuel τη δεκαετία του 1950. Το συγκεκριμένο πρόγραμμα ήταν σχεδιασμένο να μαθαίνει μέσω της εμπειρίας του και να βελτιώνεται κάθε φορά, είτε είχε αντίπαλο άνθρωπο είτε κάποιον άλλον υπολογιστή. Αυτό αποτελεί ένα από τα πρώτα και πιο διάσημα παραδείγματα Μηχανικής Μάθησης. (Buchanan, 2005)

Η παραπάνω προσέγγιση πλέον αποτελεί βασικό συστατικό της σύγχρονης Τεχνητής Νοημοσύνης. Η Μηχανική Μάθηση, λοιπόν ως υποκλάδος της Τεχνητής Νοημοσύνης, διερευνά σε βάθος το πώς μπορούν να δημιουργηθούν υπολογιστικά προγράμματα που μαθαίνουν να βελτιώνονται μέσω εμπειριών. Πιο επίσημα, ένα πρόβλημα μάθησης μπορεί να οριστεί ως η διαδικασία κατά την οποία προσπαθούμε να βελτιστοποιήσουμε την απόδοση μας σε μια συγκεκριμένη διαδικασία με τη βοήθεια προϋπάρχουσας εμπειρίας. (Jordan & Michell 2015)

Η πρόοδος της Μηχανικής Μάθησης τα τελευταία χρόνια έχει πολλά εμφανή αποτελέσματα. Χάρη στην ίδια έχουν υπάρξει πολλές καινοτόμες εφευρέσεις όπως αναγνώριση εικόνας, αναγνώριση ομιλίας, τεχνικές επεξεργασίας φυσικής γλώσσας. Αυτά είναι μερικά μόνο παραδείγματα, τα οποία έχουν υπάρξει τα τελευταία χρόνια και τα περισσότερα από αυτά μας είναι πλέον γνώριμα από την καθημερινότητα μας. Η μεγάλη αλλαγή που έφερε για τους προγραμματιστές είναι η διευκόλυνση της εκπαίδευσης των μοντέλων μηχανικής μάθησης, καθώς οι ερευνητές πλέον δεν χρειάζεται να εκπαιδεύουν το μοντέλο χειροκίνητα, αλλά μπορούν να του παρέχουν πιο κατανοητά παραδείγματα εισόδου και εξόδου, αφήνοντας το να μάθει τους κανόνες μόνο του. (Jordan & Michell 2015)

Το αντίκτυπο της Μηχανικής Μάθησης, σαφώς δεν περιορίζεται μόνο στον τομέα της πληροφορικής, αλλά επιφέρει μια επανάσταση σε όλες τις επιστήμες, από την βιολογία μέχρι και τις κοινωνικές επιστήμες. Η ραγδαία εξέλιξη της, πυροδοτήθηκε από δυο βασικούς παράγοντες. Πρώτον την υψηλή υπολογιστική ισχύ σε χαμηλό κόστος που απαιτεί η εκπαίδευση ενός μοντέλου που μας επιτρέπει την σύνθεση ολοένα και πιο περίπλοκων μοντέλων. Δεύτερον και κυριότερο, η «έκρηξη» δεδομένων γνωστή και ως «Big Data», δηλαδή η τεράστια ποσότητα δεδομένων που συλλέγονται καθημερινά από το διαδίκτυο, αποτελούν βάση εκμάθησης νέων αλγορίθμων και τεχνικών μηχανικής μάθησης. Για την αξιοποίηση αυτών των δεδομένων, ο κλάδος της Μηχανικής Μάθησης έχει αναπτύξει διαφορετικές προσεγγίσεις αναλόγως τη φύση του προβλήματος και το είδος των διαθέσιμων δεδομένων για αυτό. (Jordan & Michell 2015)

3.2.2 Κατηγορίες Μηχανικής Μάθησης

Υπάρχουν αρκετών ειδών αλγόριθμοι Μηχανικής Μάθησης όπου κάθε ένα είδος από αυτά επιτελεί και διαφορετικό σκοπό. Για παράδειγμα υπάρχουν ορισμένοι που εκτελούν διαδικασίες κατηγοριοποίησης, άλλοι είναι ικανοί να προβλέψουν μια «κίνηση», ενώ άλλοι χρησιμοποιούνται για ανίχνευση μοτίβων. Αναλόγως τη διαδικασία που χρειάζεται να γίνει κάθε φορά επάνω στα δεδομένα που υπάρχουν, οι αλγόριθμοι αυτοί διακρίνονται σε τρείς βασικούς τύπους: της επιβλεπόμενης μάθησης (supervised), της μη επιβλεπόμενης μάθησης (unsupervised), και της ενισχυτικής μάθησης (reinforcement). Υπάρχουν και προσεγγίσεις υβριδικές οι οποίες συνδυάσουν τους παραπάνω τύπους. (Sah, 2020)

3.2.2.1 Επιβλεπόμενη Μάθηση (Supervised Learning)

Η Επιβλεπόμενη Μάθηση είναι η κατηγορία της μηχανικής μάθησης όπου ένας αλγόριθμος εκπαιδεύεται χρησιμοποιώντας ένα σύνολο δεδομένων που είναι πλήρως επισημασμένο. Αυτό σημαίνει ότι κάθε σημείο δεδομένων εισόδου συνοδεύεται από την αντίστοιχη έξοδο, λειτουργώντας ως ένας «δάσκαλος» που παρέχει τις απαντήσεις. Στόχος του αλγορίθμου είναι να μάθει τη μεταξύ τους σχέση, δηλαδή την συνάρτηση που αντιστοιχίζει τις εισόδους στις εξόδους, ώστε να μπορεί να γενικεύει και να προβλέπει με ακρίβεια το αποτέλεσμα για νέα, άγνωστα δεδομένα. Τα προβλήματα αυτής της μάθησης διακρίνονται κυρίως σε προβλήματα Κατηγοριοποίησης (Classification) και Παλινδρόμησης (Regression). Τα πρώτα έχουν ως στόχο της πρόβλεψη μιας διακριτής κατηγόριας, ενώ τα δεύτερα στοχεύουν στην πρόβλεψη μιας τιμής. Και τα δύο καταλήγουν σε μια τελική απόφαση έχοντας αναλύσει πλήρως τα αρχικά δεδομένα που τους έχουν δοθεί. (Ayodele, 2010)

Για να είναι πιο κατανοητή η επιβλεπόμενη μάθηση ας θεωρήσουμε ως παράδειγμα ένα σύστημα που προβλέπει την έγκριση ενός τραπεζικού δανείου, όπου ο αλγόριθμος θα τροφοδοτείται με δεδομένα από προηγούμενους αιτούντες. Τα δεδομένα εισόδου είναι τα χαρακτηριστικά του κάθε ατόμου, όπως η ηλικία και το ετήσιο εισόδημα, ενώ τα το δεδομένο εξόδου θα είναι η τελική απόφαση, δηλαδή εάν το δάνειο δόθηκε ή απορρίφθηκε. Κατά την εκπαίδευση του το μοντέλο θα δημιουργήσει σχέσεις και κανόνες μεταξύ των δεδομένων εισόδου, όπως για παράδειγμα συνδέοντας ένα μεγαλύτερο εισόδημα με μεγαλύτερη πιθανότητα έγκρισης. Όταν θα δοθεί στον αλγόριθμο μια νέα εισαγωγή, αυτό θα είναι σε θέση να χρησιμοποιήσει τους κανόνες που ανέπτυξε για να κατηγοριοποιήσει την νέα αίτηση ως «δικαιούχος» ή «μη δικαιούχος».

3.2.2.2 Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)

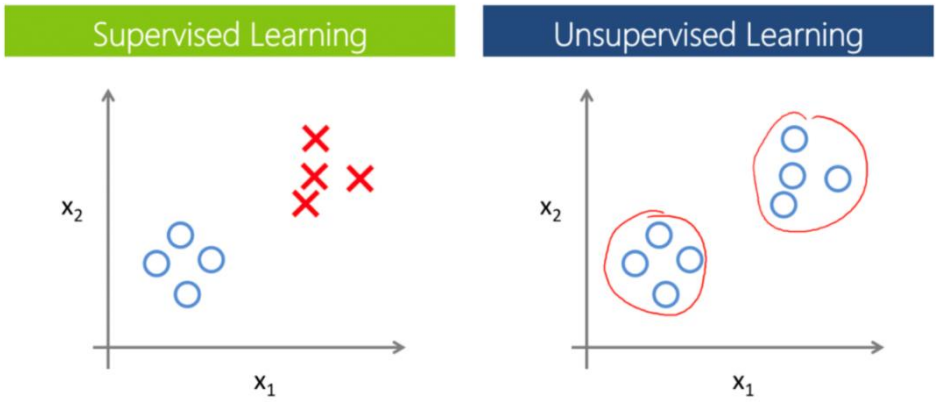
Η Μη Επιβλεπόμενη Μάθηση είναι η προσέγγιση κατά την οποία ο αλγόριθμός εκπαιδεύεται χρησιμοποιώντας δεδομένα που δεν έχουν κάποια ετικέτα. Αυτό σημαίνει ότι το σύνολο τον δεδομένων που χρησιμοποιεί αποτελείται μόνο από εισόδους και δεν υπάρχουν προκαθορισμένες απαντήσεις εξόδου για καθοδήγηση. Στόχος στους συγκεκριμένους αλγορίθμους δεν είναι η πρόβλεψη μιας τιμής, αλλά να ανακαλύψει κρυμμένες δομές και μοτίβα μέσα στα ίδια τα δεδομένα. Ουσιαστικά ο αλγόριθμος προσπαθεί να οργανώσει τα δεδομένα, χωρίζοντας τα σε ομάδες (clusters) με βάση τις ομοιότητες τους. (Ayodele, 2010)

Η πιο συνηθισμένη εφαρμογή αυτής της μεθόδου είναι η ομαδοποίηση (clustering). Μέσω αυτής ο αλγόριθμος δημιουργεί ομάδες μέσα στο σύνολο δεδομένων και έπειτα αυτές μπορούν να χρησιμοποιηθούν για να κατηγοριοποιηθεί ένα νέο άγνωστο στοιχείο σε κάποια από αυτές τις ομάδες. Χαρακτηριστικό παράδειγμα αποτελεί η ανάλυση αγοραστικής συμπεριφοράς των πελατών, για να δίνονται πιο επιτυχημένες προσφορές. Στο συγκεκριμένο παράδειγμα η εταιρία διαθέτει δεδομένα για πολλούς πελάτες, όπως συχνότητα αγορών και κατηγορίες προϊόντων που προτιμούν, μέσω των οποίων ο αλγόριθμος αναγνωρίζει και ομαδοποιεί πελάτες με παρόμοια συμπεριφορά. Έτσι όταν ένας νέος πελάτης προβεί σε αγορά, το σύστημα με βάση τις ομάδες που έχουν δημιουργηθεί, τον εντάσσει στην πιο πιθανή ομάδα, προκειμένου να του αποστέλλονται πιο εξατομικευμένες προσφορές. (Sah, 2020)

3.2.2.3 Ενισχυμένη Μάθηση (Reinforcement Learning)

Η ενισχυμένη μάθηση είναι μια διαφορετική προσέγγιση που εφαρμόζεται σε προβλήματα όπου ένας αυτόνομος πράκτορας πρέπει να μάθει να παίρνει μια ακολουθία αποφάσεων μέσα σε ένα περιβάλλον που μεταβάλλεται. Αντί να εκπαιδεύεται από ένα αμετάβλητο σύνολο δεδομένων, ο πράκτοράς μαθαίνει μέσω της μεθόδου «δοκιμής και σφάλματος». Για κάθε ενέργεια που εκτελεί, λαμβάνει ανατροφοδότηση από το περιβάλλον με την μορφή ανταμοιβής εάν η ενέργεια ήταν ορθή ή ποινής εάν ήταν εσφαλμένη. Στόχος του πράκτορα δεν είναι η άμεση ανταμοιβή, αλλά να αναπτύξει τρόπο με τον οποίο θα μεγιστοποιεί την συνολική του ανταμοιβή σε βάθος χρόνου. Χρησιμοποιούνται πολύ συχνά για την εκπαίδευση ρομπότ καθώς και σε πολλά παιχνίδια. (Sah, 2020)

Για παράδειγμα σε ένα παιχνίδι σκακιού, ο πράκτορας είναι το πρόγραμμα του υπολογιστή που μαθαίνει να παίζει και το περιβάλλον στην προκειμένη περίπτωση είναι η σκακιέρα και οι κανόνες που υπάρχουν για την κίνηση των πιονιών κάθε στιγμή. Με κάθε παιχνίδι που παίζεται, η νίκη λειτουργεί ως ανταμοιβή, ενώ η ήττα ως ποινή και σε κάθε νέο παιχνίδι που ακολουθεί, σταδιακά ο πράκτορας αρχίζει να μαθαίνει σχέσεις μεταξύ των κινήσεων και αρχίζει να αναγνωρίσει τις καλές και τις κακές ακολουθίες κινήσεων. Έπειτα από αρκετά παιχνίδια, ο πράκτορας θα έχει μάθει να παίζει πιο ορθά, βασιζόμενος σε καλύτερες ακολουθίες κινήσεων που του έχουν δώσει νίκες προηγουμένως.



Εικόνα 3.2.3: Επιβλεπόμενη και μη Επιβλεπόμενη Μάθηση

3.2.2.4 Ημι-επιβλεπόμενη Μάθηση (Semi-Supervised Learning)

Η ημι-επιβλεπόμενη μάθηση, όπως υποδηλώνει και το όνομα της αποτελεί συνδυασμός των δύο προηγουμένων που αναλύθηκαν, δηλαδή επιβλεπόμενης και μη επιβλεπόμενης μάθησης. Η βασική της αρχή είναι η εκπαίδευση αλγορίθμων με συνδυασμούς δεδομένων με ετικέτα, και χωρίς ετικέτα. Αυτή η τεχνική είναι ιδιαίτερα χρήσιμη σε σενάρια όπου υπάρχουν λίγα δεδομένα με ετικέτες και πολλά δεδομένα χωρίς καμία ετικέτα. Στην περίπτωση αυτή ένας αλγόριθμος μη επιβλεπόμενης μάθησης (unsupervised) περνάει και εντοπίζει ομάδες μέσα στα δεδομένα κι έπειτα οι υπάρχουσες ετικέτες χρησιμοποιούνται για να χαρακτηρίσουν τις ολόκληρες ομάδες που δημιουργήθηκαν, προσδίδοντας έτσι ετικέτες και στα υπόλοιπα δεδομένα, εκτελώντας αλγόριθμο επίβλεψης (supervised).

Η εφαρμογή ημι-επιβλεπόμενης μάθησης θα μπορούσε να εφαρμοστεί σε σύστημα αναγνώρισης διάσημων προσώπων. Συγκεκριμένα, εάν υπάρχουν φωτογραφίες με λίγες επισημάνσεις ονομάτων των διάσημων, και πολλές φωτογραφίες των διάσημων αυτών αλλά χωρία καμία επισήμανση, τότε πρώτα εφαρμόζεται ένας μη επιβλεπόμενος (unsupervised) αλγόριθμος. Αυτός, ομαδοποιεί όλες τι φωτογραφίες με βάση τις ομοιότητές των προσώπων δημιουργώντας ομάδες για κάθε άτομο ξεχωριστά. Έπειτα εφαρμόζεται επιβλεπόμενος (supervised) αλγόριθμος, ο οποίος με βάση τις γνωστές ετικέτες δίνει την αντίστοιχη σε κάθε ομάδα που έχει δημιουργηθεί, δημιουργώντας έτσι ένα σύστημα το οποίο αναγνωρίζει το κάθε πρόσωπο από πολλές νέες εικόνες. (Geeks for Geeks, 2025)

3.2.3 Επιβλεπόμενη Μάθηση σε βάθος - Κατηγοριοποίηση (Classification)

Η βασική αρχή της επιβλεπόμενης μάθησης βασίζεται στην έννοια της επαγωγικής μάθησης, όταν δηλαδή ένας αλγόριθμος εξάγει ένα σύνολο κανόνων από γνώσεις που έχει αντλήσει από παραδείγματα που περιέχονται σε ένα σύνολο δεδομένων που προορίζονται για την εκπαίδευση του μοντέλου (training set). Σκοπός του είναι η κατασκευή ενός μοντέλου με ισχυρή ικανότητα πρόβλεψης, ώστε να μπορεί να εφαρμοστεί σε νέες ροές δομές δεδομένων που θα δοθούν. Η τυπική ροή για την εφαρμογή της επιβλεπόμενης μάθησης σε ένα μοντέλο είναι συνήθως η παρακάτω ακολουθία βημάτων *(βλ. 3.2.3.1 Δεδομένα μοντέλου)*. (Kotsiantis, 2007)

Η Κατηγοριοποίηση (Classification) είναι μια από τις δύο πιο συχνές λειτουργείες που επιτελεί η Επιβλεπόμενη Μάθηση. Στόχος της μέθοδού είναι η ένταξη δεδομένων σε προκαθορισμένες κλάσεις, οι οποίες είναι γνωστές ως στόχος (target), ετικέτα (label), ή κατηγορία (category). Μέσω της εκπαίδευσης ο αλγόριθμος μαθαίνει από παραδείγματα και αποκτά την ικανότητα να προβλέψει και να αναθέσει σωστή κατηγορία σε ένα νέο άγνωστο δείγμα δεδομένων. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)

Οι βασικές κατηγορίες αυτών των τύπων προβλημάτων είναι τέσσερις, αναλόγως την φύση των δεδομένων και τον στόχων. Η δυαδική κατηγοριοποίηση (Binary Classification), είναι η απλούστερη μορφή και χρησιμοποιείται σε προβλήματα με δύο τελικές κατηγορίες, όπως η απόφαση εάν δικαιούται κάποιος δάνειο (ναι ή όχι). Σειρά έχει η Κατηγοριοποίηση με πολλαπλές κλάσεις (Multi-class Classification) συναντάται, όταν υπάρχουν περισσότερες από δύο πιθανές τελικές κατηγορίες, αλλά κάθε δείγμα μπορεί να ανήκει σε μια μόνο από αυτές, δηλαδή εάν είχαμε μοντέλο κατηγοριοποίησης ειδήσεων σε Αθλητικά, Πολιτικά και Επιστημονικά γεγονότα, κάθε είδηση θα άνηκε σε μια μόνο κλάση από αυτές. Υπάρχει και η Ανισόρροπη Κατηγοριοποίηση (Imbalanced Classification), όπου ο αριθμός δειγμάτων ανά κατηγορία είναι εξαιρετικά ανισόρροπος, για παράδειγμα σε ένα σύστημα εντοπισμού απάτης, υπάρχουν πολλά «φυσιολογικά» δεδομένα και λιγότερα δεδομένα «απάτης». Τέλος, η Κατηγοριοποίηση με πολλαπλές ετικέτες (Multi-label Classification) παρατηρείται όταν υπάρχουν περισσότερες από δύο πιθανές κατηγορίες και ένα δείγμα δεδομένων μπορεί να ανήκει ταυτόχρονα σε πολλές από αυτές. Για παράδειγμα σε σύστημα κατηγοριοποίησης προτιμήσεων διαφόρων ειδών ταινιών, μπορεί κάποιος να ανήκει στις κατηγορίες Δράση, Κωμωδία και Κινούμενα σχέδια ταυτόχρονα. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)

3.2.3.1 Δεδομένα Μοντέλου

1. Το πρώτο βήμα αποτελεί η συλλογή του συνόλου δεδομένων.
2. Το δεύτερο βήμα είναι η προετοιμασία και προεπεξεργασία των δεδομένων. Εδώ αντιμετωπίζονται προβλήματα όπως ελλιπείς τιμές και οι ακραίες τιμές ή θόρυβος. Για κάθε ένα από αυτά υπάρχουν εξειδικευμένες τεχνικές επίλυσης.
3. Το τρίτο βήμα αφορά τη διαχείριση και βελτιστοποίηση των χαρακτηριστικών των δεδομένων. Σε αυτό το σημείο γίνεται η επιλογή των πιο σημαντικών και αφαιρούνται όσα κρίνονται περιττά. Κι αυτό συμβαίνει, διότι σκοπός του μοντέλου είναι να υπάρχουν όσο το δυνατόν πιο καθαρά και ουσιώδη χαρακτηριστικά, ώστε το τελικό μοντέλο να λειτουργεί πιο γρήγορα και να πετυχαίνει μεγαλύτερη ακρίβεια. (Kotsiantis, 2007)

Στο παράδειγμα με το δάνειο, τα βήματα που αναφέρθηκαν θα λειτουργούσαν ως εξής:

1. Συλλογή δεδομένων

ID	Ηλικία	Μηνιαίο Εισόδημα	Ετήσιο Εισόδημα	Αποταμίευση	Απόφαση
1	19	900	11.000	500	Ναι
2	-	1.200	13.000	2500	Όχι
3	42	1.500	35.000	1000	Ναι

2. Προεπεξεργασία Δεδομένων

Εδώ παρατηρείται ότι λείπει η ηλικία από την δεύτερη εισαγωγή, οπότε συμπληρώνεται από τον μέσο όρο των υπολοίπων. Ενώ ταυτόχρονα φαίνεται να υπάρχει ακραίο ποσό στην τρίτη εισαγωγή στο ετήσιο εισόδημα όπου αυτό θα έπρεπε να είναι 12*1500 = 18.000 αντί για 35.000.

ID	Ηλικία	Μηνιαίο Εισόδημα	Ετήσιο Εισόδημα	Αποταμίευση	Απόφαση
1	19	900	11.000	500	Όχι
2	30	1.200	14.500	2500	Ναι
3	42	1.500	18.000	1000	Ναι

3. Διαχείριση Χαρακτηριστικών

Στο σημείο αυτό, κρατάμε τα χαρακτηριστικά που θεωρούμε σημαντικά για το μοντέλο, οπότε αφαιρούμε το ID, το Μηνιαίο Εισόδημα, καθώς υπάρχει ήδη το ετήσιο και έχουν την ίδια σχέση μεταξύ τους. Έτσι τα τελικά δεδομένα που θα χρησιμοποιηθούν για την δημιουργία του μοντέλου θα έχουν την εξής μορφή:

Ηλικία	Ετήσιο Εισόδημα	Αποταμίευση	Απόφαση
19	11.000	500	Όχι
30	14.500	2500	Ναι
42	18.000	1000	Ναι

3.2.3.2 Επιλογή Κατάλληλου Αλγορίθμου

Η επιλογή του κατάλληλου αλγορίθμου αποτελεί ένα από τα πιο κρίσιμα στάδια της διαδικασίας. Όταν ένα μοντέλο εκπαιδευτεί η απόδοση του αξιολογείται συνήθως με βάση την ακρίβεια των προβλέψεων του (prediction accuracy). Για να μετρηθεί αυτή η ακρίβεια με αξιοπιστία, τα δεδομένα χωρίζονται σε ένα μέρος που χρησιμοποιείται αποκλειστικά για τον έλεγχο της ακρίβειας. Οι πιο γνωστές τεχνικές για αυτόν τον σκοπό είναι ο διαχωρισμός 70% για εκπαίδευση και 30% για έλεγχο. Αν η απόδοση του μοντέλου δεν είναι ικανοποιητική, ενδεχομένως να υπάρχουν ακόμη βελτιώσεις που να χρειάζονται στο σύνολο δεδομένων. (Kotsiantis, 2007)

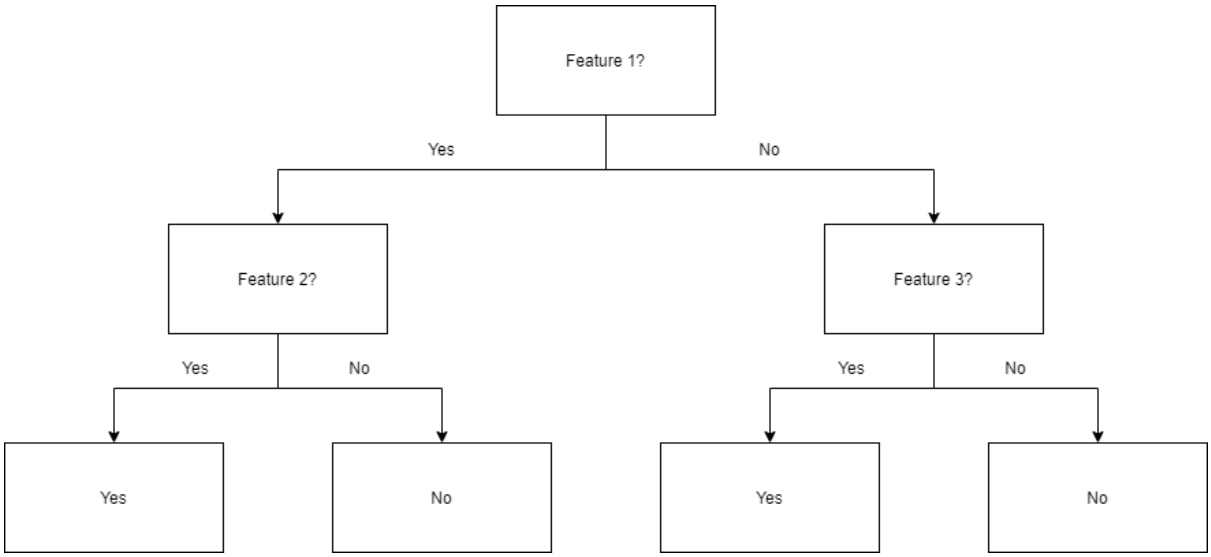
Με βάση τη φύση του συνόλου δεδομένων, ορισμένες φορές κάποιοι συγκεκριμένοι αλγόριθμοι ενδέχεται να είναι πιο αποτελεσματικοί από άλλους. Για παράδειγμα, σε τεράστιου όγκου δομές δεδομένων, οι απλούστεροι τύποι αλγορίθμων είναι συχνά πιο γρήγοροι και ελαφριοί για τη μνήμη. Ο χρόνος εκπαίδευσης, ακόμα, σε μεγάλα δεδομένα μπορεί να είναι μεγαλύτερος. Επομένως υπάρχουν αρκετοί και διαφορετικής φύσης αλγόριθμοι κατηγοριοποίησης.

3.2.4 Μέθοδοι Κατηγοριοποίησης - Classification Algorithms

3.2.4.1 Δέντρα Απόφασης – Decision trees

Τα δέντρα απόφασης είναι από τις παλαιότερες και πιο δημοφιλείς μεθοδολογίες προγνωστικής μοντελοποίησης στην επιβλεπόμενη μάθηση. Είναι ιδιαίτερα αποτελεσματικά για προβλήματα κατηγοριοποίησης και κάθε ένα δέντρο αποτελεί μια γραφική μορφή που αναπαριστά μια σειρά αποφάσεων. Η δομή τους αποτελείται από εσωτερικούς και εξωτερικούς κόμβους (nodes), οι οποίοι συνδέονται μεταξύ τους με κλαδιά (branches), μοντελοποιώντας μια διαδρομή από την αρχική ρίζα του προβλήματος, έως και την τελική απόφαση. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)

Η λειτουργία του δέντρου βασίζεται αποκλειστικά στους κόμβους και τα κλαδιά του. Κάθε εσωτερικός κόμβος (ενδιάμεσος του αρχικού και του τελικού) θέτει μια ερώτηση για ένα συγκεκριμένο χαρακτηριστικό, ενώ τα κλαδιά του συγκεκριμένου κόμβου αντιπροσωπεύουν τις πιθανές απαντήσεις. Η διαδρομή αυτή καταλήγει σε έναν κόμβο-φύλλο, ο οποίος παρέχει την τελική απόφαση, δηλαδή την τελική κλάση. Η κατασκευή του δέντρου ξεκινάει από το επάνω μέρος, την ρίζα, και καταλήγει κάτω, στο τελικό φύλλο. Τα δέντρα αποφάσεων είναι εξαιρετικά διαφανή και εύκολα στην ερμηνεία και χρησιμοποιούνται συχνά σε μοντέλα ιατρικής διάγνωσης. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)



Εικόνα 3.2.4.1: Δέντρα Αποφάσεων – Decision Trees

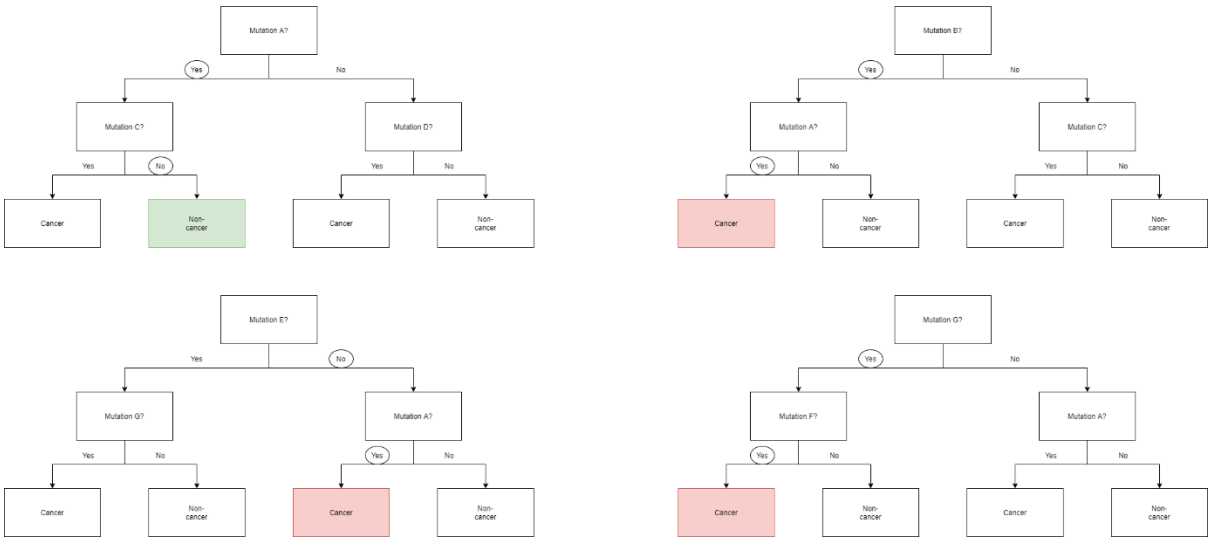
Εάν χρησιμοποιούσαμε το παράδειγμα με το δάνειο και την εικόνα 3.2.4.1 ο κόμβος «Feature 1» θα μπορούσε να είναι η ερώτηση «Ηλικία > 25». Παρομοίως και για τους υπόλοιπους κόμβους θα συγκρινόταν το ετήσιο εισόδημα καθώς και το ποσό αποταμίευσης του καθενός, και τα τελικά φύλλα θα έδιναν την κατηγορία για κάθε πιθανή ροή χαρακτηριστικών.

Η δημοφιλία των δέντρων απόφασης οφείλεται στην απλότητα και την σαφήνεια που προσφέρουν στον χρήστη, ενώ παράλληλα είναι υπολογιστικά ελαφριά και γρήγορα στην εκπαίδευση και εξαγωγή αποτελεσμάτων. Ένας σημαντικός περιορισμός τους, όμως είναι ότι η απόδοση τους συχνά μειώνεται με τη πάροδο του χρόνου, καθώς είναι εξαιρετικά ευαίσθητα στο φαινόμενο της απόκλισης δεδομένων (data drift). Αυτό σημαίνει ότι, καθώς τα μοτίβα στα δεδομένα αλλάζουν, το μοντέλο χρειάζεται επανεκπαίδευση για να παραμένει ακριβές. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)

3.2.4.2 Τυχαία Δάση - Random Forest

Ο αλγόριθμος των Τυχαίων δασών ανήκει στην κατηγορία των αλγορίθμων Συνόλου (Ensemble algorithms) οι οποίοι δεν βασίζονται σε ένα μόνο μοντέλο, αλλά συνδυάζουν τις προβλέψεις από πολλαπλά βασικά μοντέλα με στόχο τη δημιουργία ενός τελικού μοντέλου. Έτσι, ο αλγόριθμος Τυχαίο Δάσος κατασκευάζει πολλαπλά δέντρα απόφασης δημιουργώντας ένα «δάσος». Όταν ένα νέο δεδομένο εισάγεται στο μοντέλο, κάθε δέντρο στο δάσος κάνει την δική του ανεξάρτητη πρόβλεψη και η τελική απόφαση προκύπτει συλλογικά, δηλαδή λαμβάνεται υπόψιν η πλειοψηφία. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)

Το κυριότερο πλεονέκτημα του αλγόριθμου αυτού σε σχέση με ένα μεμονωμένο δέντρο απόφασης είναι η μείωση της υπερπροσαρμογής (overfitting), γεγονός που οδηγεί σε υψηλότερη ακρίβεια. Ακόμη, είναι περισσότερο ανθεκτικό σε θόρυβο και ακραίες τιμές, ενώ μπορεί να διαχειριστεί με ευκολία μεγάλα σύνολα δεδομένων. Από την άλλη όμως, ο αλγόριθμος απαιτεί περισσότερο χρόνο εκπαίδευσης και υπολογιστικούς πόρους σε σύγκριση με άλλες μεθόδους. Επιπλέον, συνήθως θεωρείται πιο αποδοτικός από τα απλά δέντρα, αλλά υστερεί έναντι σε άλλες τεχνικές. Ο συγκεκριμένος αλγόριθμος χρησιμοποιείται συνήθως σε τραπεζικούς τομείς για την εκτίμηση πιθανής απάτης και στο ψηφιακό μάρκετινγκ για την πρόβλεψη απήχησης περιεχομένου των κοινωνικών δικτύων. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)



Εικόνα 3.2.4.2: Τυχαίο Δάσος – Random Forest

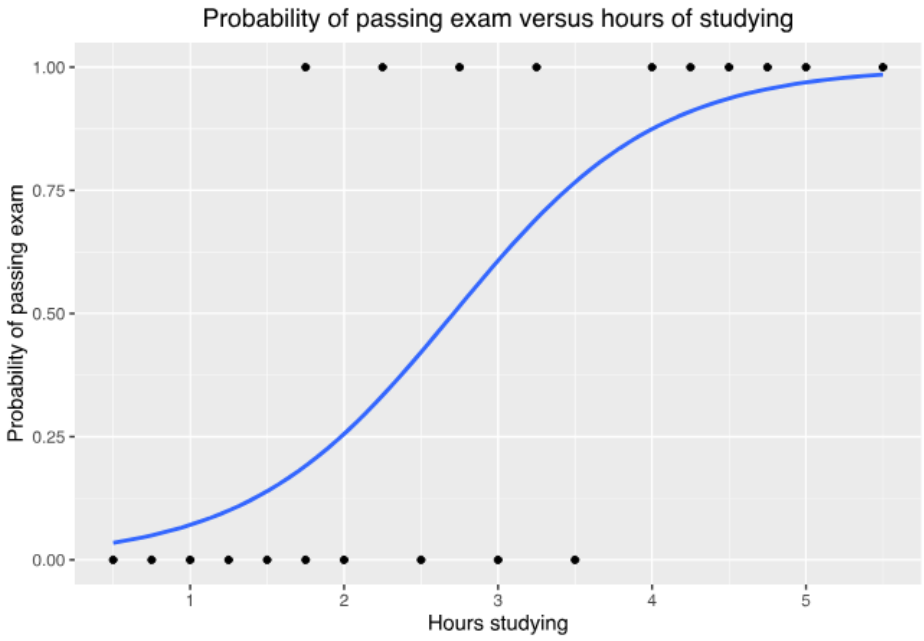
3.2.4.3 Λογιστική Παλινδρόμηση – Logistic regression

Η Λογιστική Παλινδρόμηση είναι μια ισχυρή στατική μέθοδος, που προβλέπει την πιθανότητα να συμβεί ή να μην συμβεί ένα γεγονός, για αυτό και η έξοδος της είναι πάντα μια τιμή μεταξύ του 0 και του 1. Για να καταλήξει στην τελική απόφαση, ο αλγόριθμος ορίζει ένα όριο απόφασης που διαχωρίζει τις κλάσεις. Η πιθανότητα που παράγει το μοντέλο συγκρίνεται με ένα προκαθορισμένο όριο (threshold), το οποίο συνήθως είναι 0.5. Ο συγκεκριμένος αλγόριθμος έχει ως στόχο να δημιουργήσει σχέση ανάμεσα σε μια εξαρτημένη μεταβλητή, η οποία αποτελεί τον στόχο της πρόβλεψης, και ένα σύνολο ανεξάρτητων μεταβλητών, οι οποίες θεωρούνται οι παράγοντες που την επηρεάζουν. Για να πετύχει αυτό ο αλγόριθμος υπολογίζει την πιθανότητα ένα νέο δείγμα να ανήκει σε μια συγκεκριμένη κλάση, μέσω μιας χαρακτηριστικής καμπύλης, της σιγμοειδής καμπύλης

(sigmoid function). Ένα από τα πλεονεκτήματα της είναι ότι διαχειρίζεται αριθμητικά και μη αριθμητικά δεδομένα. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)

Σε γενικές η Λογιστική Παλινδρόμηση είναι ένας γρήγορος, απλός και εύκολος να ερμηνευτεί αλγόριθμος, αλλά η κύρια του αδυναμία είναι ότι μπορεί να λύσει μόνο γραμμικά διαχωρίσιμα προβλήματα. Συγκεκριμένα, έχει απλή υλοποίηση και είναι υπολογιστικά ελαφρύς και η έξοδος της είναι μια πιθανότητα οπότε είναι εύκολη να ερμηνευτεί. Ωστόσο, λειτουργεί μόνο για γραμμικά προβλήματα και είναι αδύναμη όταν τα προβλήματα δεν μπορούν να διαχωριστούν από μια ευθεία γραμμή. Συνήθως χρησιμοποιείται στη Ιατρική για διαγνώσεις, την Μηχανική για προβλέψεις βλάβης, και τις Κοινωνικές Επιστήμες για κατηγοριοποίηση κειμένων ή ανάλυση εκλογικών αποτελεσμάτων. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)

Η εικόνα 3.2.4.3 δείχνει παράδειγμα Logistic Regression όπου οι μαύροι κόμβοι αναπαριστούν πραγματικά δεδομένα, και η μπλε καμπύλη δείχνει τη σχέση πιθανότητας για τα χαρακτηριστικά «Ώρες μελέτης» και «Πιθανότητα επιτυχίας στην εξέταση».



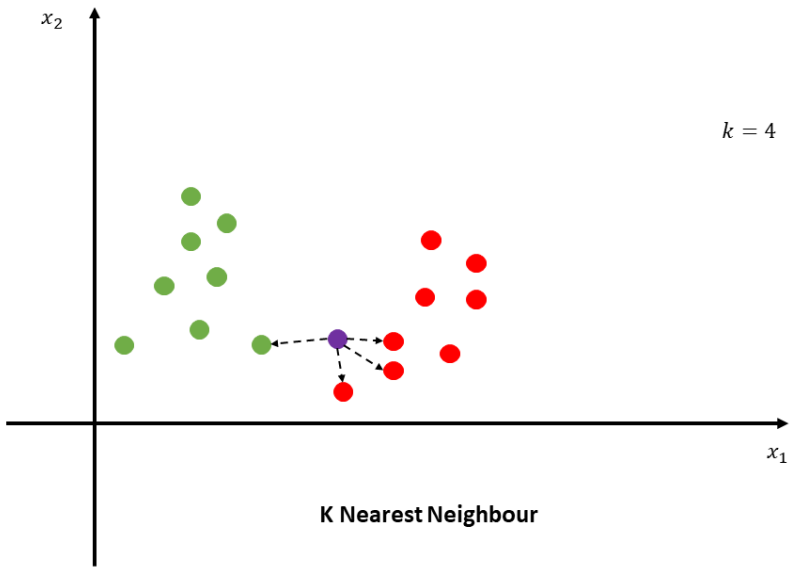
Εικόνα 3.2.4.3: Λογιστική Παλινδρόμηση – Logistic Regression

Για να γίνει πιο κατανοητή η Λογιστική Παλινδρόμηση, θα μπορούσε να εφαρμοστεί το παράδειγμα με την αίτηση δανείου στον αλγόριθμο. Σε αυτή τη περίπτωση ο αλγόριθμος θα εξέταζε τα χαρακτηριστικά του κάθε αιτούντα στη βάση δεδομένων που υπάρχει (ετήσιο εισόδημα, ηλικία, ποσό αποταμιεύσεων) κι έπειτα θα εκπαιδευόταν, προσπαθώντας ταυτόχρονα να «ζυγίσει» τα στοιχεία αναλόγως με το πόσο σημαντικά είναι αυτά κάθε φορά για το τελικό αποτέλεσμα που προκύπτει. Για παράδειγμα μπορεί να θεωρήσει ότι το υψηλό εισόδημα είναι θετικό, η μικρότερη ηλικία είναι πολύ θετική και ότι οι υψηλές αποταμιεύσεις είναι επίσης θετικές. Έτσι για κάθε νέα αίτηση, ο αλγόριθμος θα συνδυάζει τις προηγούμενες του γνώσεις και θα υπολογίζει ένα συνολικό αθροιστικό αποτέλεσμα. Το τελευταίο, θα μετατρέπεται σε πιθανότητα έγκρισης από το 0 έως το 1, κι έτσι κάποιος με υψηλό εισόδημα και υψηλό αθροιστικό αποτέλεσμα μπορεί να έχει πιθανότητα 80%. Εν ολίγοις λοιπόν, ο αλγόριθμος λαμβάνει τα δεδομένα του προβλήματος, υπολογίζει την πιθανότητα για ένα από τα δύο ενδεχόμενα και λαμβάνει μια τελική απόφαση.

3.2.4.4 Κ-Πλησιέστεροι Γείτονες (K-Nearest Neighbors - KNN)

Ο συγκεκριμένος αλγόριθμος είναι ένας από του πιο απλούς αλγορίθμους, αλλά εξίσου ισχυρός. Για να κατηγοριοποιήσει ένα νέο στοιχείο δεδομένων, εξετάσει τα χαρακτηριστικά των πλησιέστερων ήδη κατηγοριοποιημένων γειτόνων του. Κατά την λειτουργία του υπολογίζει την απόσταση του νέου στοιχείου με όλα τα ήδη υπάρχοντα, εντοπίζει τους «Κ» πλησιέστερους, κι αυτοί με την σειρά τους ορίζουν τη πλειοψηφία τους, την κατηγορία του νέου στοιχείου. Η απόδοση του εξαρτάται από την ποιότητα των δεδομένων και η κρισιμότερη του παράμετρος είναι η επιλογή του κατάλληλου αριθμού γειτόνων «Κ», καθώς κάποια πολύ μικρή τιμή ενδέχεται να δώσει εσφαλμένο αποτέλεσμα βασισμένο σε λίγα δεδομένα, ενώ μια πολύ μεγάλη τιμή ενδέχεται και πάλι να διχάσει τον αλγόριθμο από πληθώρα δεδομένων που πρέπει να ληφθούν υπόψιν. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)

Ο KNN αλγόριθμος είναι απλός στην κατανόηση και την εφαρμογή, αλλά ενδέχεται να είναι χρονοβόρος για μεγάλα σύνολα δεδομένων. Επιπλέον, συχνά είναι υπολογιστικά βαρύς καθώς για κάθε νέο εισερχόμενο στοιχείο ο αλγόριθμος πρέπει να το συγκρίνει με κάθε ένα που υπήρχε. Ταυτόχρονα, είναι αρκετά ευαίσθητο ως προς την αποδοτικότητα του, καθώς η χρήση εσφαλμένου αριθμού γειτόνων «Κ» δύναται επηρεάσει σημαντικά το αποτέλεσμα. Ο συγκεκριμένος αλγόριθμος εφαρμόζεται κυρίως σε συστήματα προτάσεων (πχ. ταινιών), στην χρηματοοικονομική και στην ιατρική. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)



Εικόνα 3.2.4.4: Κ-Πλησιέστεροι Γείτονες – KNN

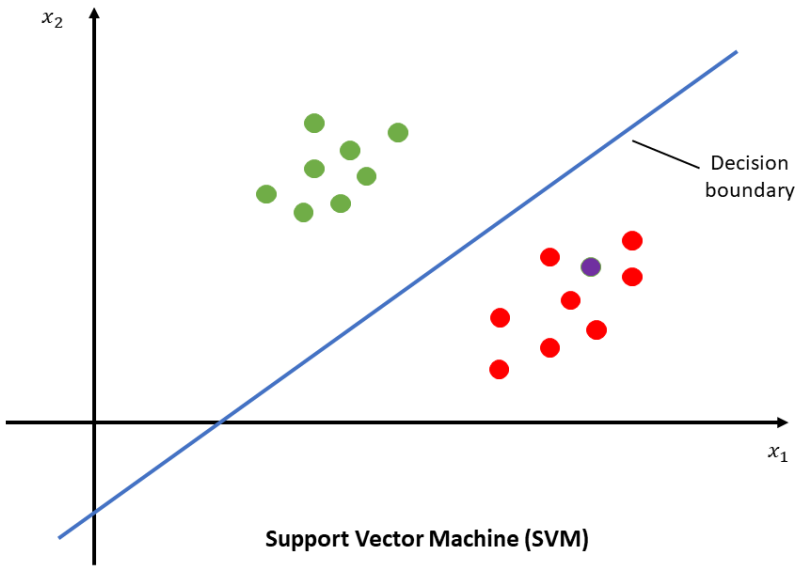
Για παράδειγμα στην εικόνα 3.2.4.4 υπάρχουν δύο τελικές κατηγορίες, κόκκινο και πράσινο, έστω ότι η μία αντιστοιχεί στην «αποδοχή» αίτησης δανείου και η άλλη στην «απόρριψη». Κατά την εισαγωγή ενός νέου στοιχείου, ο μπλε κόμβος, δεδομένου ότι Κ=4, υπολογίζεται η απόσταση του νέου στοιχείου σε σχέση με όλα τα προυπάρχοντα, και ο αλγόριθμος καταλήγει ότι τα τέσσερα κοντινότερα είναι αυτά με τις διακεκομμένες γραμμές. Στη συνέχεια, ελέγχονται οι κατηγορίες που έχουν ήδη δοθεί στους γείτονες, στην προκειμένη περίπτωση η πλειονότητα είναι «απόρριψη». Επομένως και το νέο στοιχείο που δόθηκε θα ανήκει στην κατηγορία «απόρριψη».

3.2.4.5 Μηχανές Υποστήριξης Διανυσμάτων (Support Vector Machine - SVM)

Ο αλγόριθμος SVM είναι μια ισχυρή τεχνική για την επίλυση προβλημάτων ταξινόμησης. Ο βασικός του στόχος είναι να βρει μια βέλτιστή γραμμή σε δύο διαστάσεις, ή ένα επίπεδο σε τρεις, που να διαχωρίζει τα δεδομένα δύο ή περισσότερων κλάσεων με τον πιο σαφή τρόπο. Προσπαθεί δηλαδή, να βρει το μεγαλύτερο δυνατό περιθώριο ανάμεσα στα πλησιέστερα σημεία των κατηγοριών που υπάρχουν. Ο αλγόριθμος προσπαθεί να δημιουργήσει όσο το δυνατόν πιο ευρύ περιθώριο, για να είναι πιο καθαρά τα όρια κάθε κλάσης και να ελαχιστοποιεί τη πιθανότητα σφάλματος. Τα πιο κρίσιμα σημεία είναι τα διανύσματα υποστήριξης (support vectors, τα οποία βρίσκονται πάνω στα όρια του περιθωρίου και ουσιαστικά καθορίζουν την θέση του κάθε επιπέδου. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)

Όταν τα δεδομένα δεν είναι γραμμικά διαχωρίσιμα, τότε ο αλγόριθμος μπορεί να χρησιμοποιήσει την τεχνική του «kernel». Μέσω της χρήσης, δηλαδή, σύνθετων μαθηματικών συναρτήσεων (kernels) ο SVM αλγόριθμος ουσιαστικά προβάλλει τα δεδομένα σε μια υψηλότερη διάσταση, όπου ο γραμμικός διαχωρισμός γίνεται εφικτός. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)

Σε γενικές γραμμές ο αλγόριθμος SVM είναι εξαιρετικά ισχυρός και ακριβής ειδικά σε πολύπλοκα προβλήματα, αλλά συνοδεύεται από πολυπλοκότητα στη χρήση και μειωμένη απόδοση σε πολύ μεγάλα ή «θορυβώδη» σύνολα δεδομένων. Εφαρμόζεται κυρίως σε τομείς πολύπλοκους, όπως Χρηματοοικονομικά, Ιατρικές Διαγνώσεις, και Τραπεζική Ασφάλεια. (Alnuaimi, A. F., & Albaldawi, T. H., 2024)



Εικόνα 3.2.4.5: Μηχανές Υποστήριξης Διανυσμάτων - SVM

Από την εικόνα 3.2.4.5 μπορεί να αναλυθεί πως θα λειτουργούσε ο εξής αλγόριθμος με το παράδειγμα του δάνειου. Ομοίως οι πράσινες κουκκίδες θα ορίζουν «Έγκριση» δανείου και οι κόκκινες αντίστοιχα «Απόρριψη». Κάθε άξονας θα ορίζει κάποιο χαρακτηριστικό, για παράδειγμα το ετήσιο εισόδημα ή την ηλικία. Ο αλγόριθμος κατά την εκπαίδευση του θα δημιουργήσει μια γραμμή – όριο απόφασης, η οποία θα διαχωρίζει τις δυο κατηγορίες με

τον καλύτερο δυνατό τρόπο. Στόχος του είναι να δημιουργήσει το μεγαλύτερο δυνατό εύρος μεταξύ των Εγκρίσεων και τον Απορρίψεων. Κατά την εισαγωγή ενός νέου κόμβου, ο μωβ στην προκειμένη περίπτωση, ο αλγόριθμος ελέγχει σε ποια πλευρά της γραμμής βρίσκεται, και τον κατατάσσει στην κατηγορία «Απόρριψη».

3.2.5 Αξιολόγηση Μοντέλου Κατηγοριοποίησης

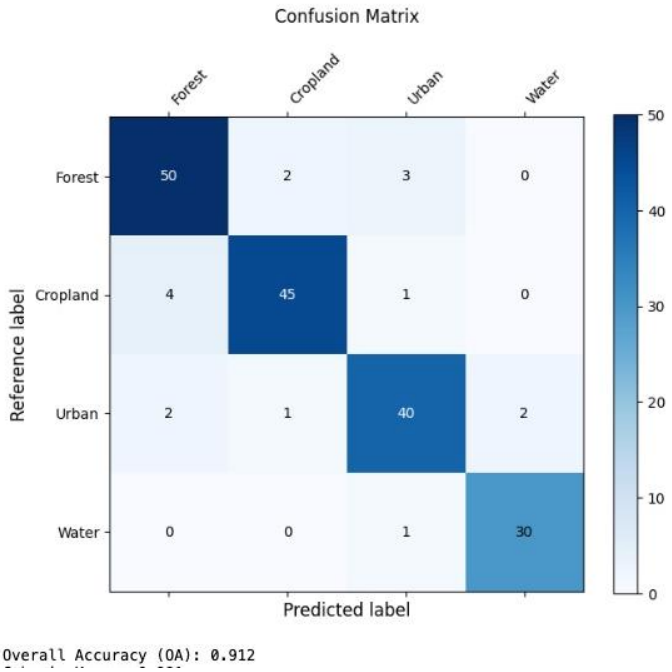
3.2.5.1 Ακρίβεια (Accuracy)

Προκειμένου να αξιολογηθεί η απόδοση ενός μοντέλου μηχανικής μάθησης χρησιμοποιείται συνήθως η έννοια της ακρίβειας (accuracy). Η ακρίβεια δίνει το ποσοστό τον προβλέψεων το οποίο το μοντέλο κατάφερε να εντοπίσει ορθά, συγκεκριμένα η τιμή της δημιουργείται από έναν απλό μαθηματικό τύπο ο οποίος διαιρεί τον αριθμό των σωστών προβλέψεων με τον συνολικό αριθμό των προβλέψεων που πραγματοποιήθηκαν.

Συνήθως κατά τη δημιουργία του μοντέλου, κρατάται ένα ποσοστό από το σύνολο τον δεδομένων που υπήρχε ως «παράδειγμα» και εκπαιδεύεται με αυτό (συνήθως 70%) και το υπόλοιπο ποσοστό (30%), το μοντέλο προσπαθεί να προβλέψει την κατηγορία του. Η τιμή της ακρίβειας προκύπτει αναλόγως με τη ποσότητα των προβλέψεων που ήταν ορθές. Ουσιαστικά η τιμή αυτή εκφράζει με ποσοστό την ακρίβεια της πρόβλεψης.

3.2.5.2 Πίνακας Σύγχυσης (Confusion matrix)

Ο πίνακας σύγχυσης είναι ένα εργαλείο αξιολόγησης απόδοσης κατηγοριοποίησης προσφέροντας μια λεπτομερή εικόνα του που αποτυγχάνει. Ουσιαστικά, είναι ένας πίνακας που συνοψίζει τα αποτελέσματα των προβλέψεων, δείχνοντας τις κατηγορίες που προέβλεψε το μοντέλο συγκριτικά με τις πραγματικές κατηγορίες. Η ονομασία της προέρχεται από το γεγονός ότι αποκαλύπτει που συγχέεται ο αλγόριθμος, δηλαδή ποιες κατηγορίες ενδέχεται να μπερδέψει κάποιες άλλες. (Jacob Murel, 2024)



Εικόνα 3.2.5.2: Confusion Matrix

Στην εικόνα 3.2.5.2, φαίνεται ένας πίνακας σύγχυσης για ένα μοντέλο έχει τέσσερεις κατηγορίες (Δάσος, Καλλιεργήσιμη Γη, Αστική Περιοχή, Νερό). Η κύρια διαγώνιος, η οποία περιέχει τους αριθμούς με έντονο μπλε χρώμα, αποκαλύπτει τις σωστές προβλέψεις, δηλαδή

τις περιπτώσεις όπου η πραγματική κατηγορία ταυτίστηκε με αυτήν που προέβλεψε το μοντέλο. Αντιθέτως, οι αριθμοί εκτός της διαγώνιου δείχνουν τα λάθη του μοντέλου. Για παράδειγμα, το μοντέλο ταξινόμησε λανθασμένα 4 περιοχές που ήταν «Καλλιεργήσιμη Γη» ως «Δάσος», και μπέρδεψε 2 περιοχές «Αστικής Περιοχής» με «Νερό». Η συνολική ακρίβεια του μοντέλου φαίνεται στο τέλος, 91.2% που σημαίνει ότι έκανε σωστές προβλέψεις στο ίδιο ποσοστό ορθότητας στο σύνολο των περιπτώσεων.

3.2.5.3 Πιθανολογική Κατηγοριοποίηση - Probabilistic Classification

Στον χώρο της μηχανικής μάθησης, η έννοια της Πιθανολογικής Κατηγοριοποίησης είναι μια προσέγγιση κατά την οποία ένας αλγόριθμος αντί να αποδίδει κατευθείαν μια απόλυτη κατηγορία σε μια νέα εισαγωγή, υπολογίζει μια κατανομή πιθανοτήτων πάνω σε όλες τις πιθανές κλάσεις. Εν ολίγοις, για ένα δεδομένο στοιχείο, το μοντέλο κατηγοριοποίησης δεν απαντά απευθείας με την απάντηση «Μη δικαιούχος Δανείου», αλλά παρέχει πιο αναλυτική απάντηση όπως «Κατά 70% Μη δικαιούχος δανείου», και «Κατά 30% Δικαιούχος δανείου». Η εκτίμηση αυτή της αβεβαιότητας δίνει μια πολύ καλύτερη εικόνα για την αυτοπεποίθηση του μοντέλου (confidence). Με παρόμοιοι τρόπο λειτουργεί ο αλγόριθμος Logistic Regression, αλλά η μέθοδος αυτή μπορεί να εφαρμοστεί και στους υπολοίπους τύπους που αναφέρθηκαν. (Wikipedia contributors, 2025)

3.3 Tkinter

3.3.1 Η Python ως γλώσσα προγραμματισμού

Η Python είναι μια σύγχρονη γλώσσα προγραμματισμού υψηλού επιπέδου, δηλαδή ως γλώσσα προγραμματισμού είναι πιο κοντά στην ανθρώπινη γλώσσα παρά στην γλώσσα μηχανής που κατανοεί υπολογιστής. Κατά τη δημιουργία της, σκοπός της ήταν να είναι απλή και αναγνωρίσιμη μέσα στον κώδικα. Η φιλοσοφία της επιτρέπει στους προγραμματιστές να υλοποιούν πολύπλοκες λειτουργίες σε σημαντικά λιγότερες γραμμές κώδικα σε σχέση με άλλες γλώσσες. Ως γλώσσα προσφέρει μεγάλη ευελιξία και έχει καθιερωθεί ως ένα από τα πιο δημοφιλή εργαλεία για ένα ευρύ φάσμα εφαρμογών, από την ανάπτυξη ιστοσελίδων, μέχρι και την επιστήμη δεδομένων και την τεχνητή νοημοσύνη. (Βικιπαίδεια, 2025)

3.3.2 Tkinter και Python

Η Tkinter αποτελεί μια καθιερωμένη βιβλιοθήκη της γλώσσας προγραμματισμού Python για την δημιουργία γραφικών διεπαφών χρήστη (GUI) και περιλαμβάνεται σε όλες τις τυπικές εγκαταστάσεις τις γλώσσας. Το όνομα της προέρχεται από την φράση «Tk interface», καθώς ουσιαστικά λειτουργεί ως ένας μεταφραστής επικοινωνίας μεταξύ της Python και του πακέτου εργαλείων Tk. Η ίδια δημιουργήθηκε αρχικά από τον Steen Lumholt και τον Guido van Rossum (ιδρυτής της Python), επιτρέποντας στους προγραμματιστές να γράφουν κώδικα σε Python, ο οποίος στη συνέχεια επιτρέπεται σε εντολές που μπορεί να εκτελέσει το Tk για να σχεδιάσει παράθυρα, κουμπιά, και άλλα οπτικά στοιχεία μιας εφαρμογής. (Wikipedia contributors., 2025)

3.3.3 Βασικά στοιχεία (Widgets)

Τα widgets αποτελούν τα δομικά στοιχεία από τα οποία φτιάχνεται κάθε γραφική διεπαφή (GUI) στη βιβλιοθήκη Tkinter. Υπάρχουν διάφορες και καθένα από αυτά επιτελεί διαφορετικό σκοπό κατά τη κατασκευή του. Παρακάτω θα αναφερθούν τα κυριότερα και τα πιο συχνά στη χρήση.

- **Frame**: Το πλαίσιο είναι βασικό δομικό στοιχείο για την οργάνωση της εφαρμογής και λειτουργεί ως μια απλή ορθογώνια περιοχή, η οποία συνήθως χρησιμοποιείται για ομαδοποίηση και διάταξη άλλων στοιχείων widget. Κάθε πλαίσιο frame μπορεί να έχει περίγραμμα και φόντο.
- **Canvas**: Ο Καμβάς είναι ένα εξαιρετικά ευέλικτη περιοχή σχεδίασης, που παρέχει χώρο δημιουργίας γραφικών στοιχείων. Μπορεί να χρησιμοποιηθεί για γραφήματα, σχέδια, εμφάνιση κειμένου καθώς και εικόνων.
- **Button**: Ρόλος του κουμπιού είναι να ενεργοποιήσει την εκτέλεση μιας συγκεκριμένης λειτουργίας η εντολής όταν ο χρήστης το πατήσει. Στο κουμπί μπορεί να εμφανιστεί κείμενο ή εικόνα.
- **Entry**: Το Πεδίο Εισαγωγής είναι ένα πεδίο κειμένου που επιτρέπει στον χρήστη να πληκτρολογήσει μια μόνο γραμμή κειμένου, χρησιμοποιείται συνήθως όταν απαιτείται η εισαγωγή δεδομένων, όπως ονόματα χρήστη.
- **Label**: Η ετικέτα είναι ένα στοιχείο που χρησιμοποιείται για την εμφάνιση στατικού περιεχομένου, το οποίο μπορεί να είναι είτε κείμενο, είτε μια εικόνα. Σε αντίθεση με το Entry, δεν έχει ως σκοπό την αλληλεπίδραση με τον χρήστη, αλλά μόνο να εμφανίσει πληροφορίες.
- **Scrollbar**: Η Μπάρα Κύλισης, είναι ένα τυπικό στοιχείο ελέγχου, που δίνει τη δυνατότητα στον χρήστη να πλοηγήθει σε περιεχόμενο το οποίο είναι πολύ μεγάλο, για να χωρέσει σε μια περιοχή. Συνήθως χρησιμοποιείται σε συνδυασμό με άλλα στοιχεία όπως ο Καμβάς, η Λίστα, ή το Κείμενο.

Τα περισσότερα στοιχεία διεπαφής στις εφαρμογές που χρησιμοποιούν το Tkinter, περιλαμβάνουν συνδυασμούς από τα παραπάνω στοιχεία. (Lundh, 1999)

3.3.4 Διαχείριση Διάταξης (Geometry managers)

Οι διαχειριστές διάταξης είναι οι μηχανισμοί που χρησιμοποιεί το Tkinter για να καθορίσει τη θέση και το μέγεθος των διαφόρων στοιχείων μέσα σε ένα παράθυρο ή ένα πλαίσιο. Κάθε στοιχείο που προστίθεται σε μια γραφική διεπαφή πρέπει να σημειωθεί με έναν από αυτούς τους διαχειριστές για να γίνει ορατό. Το Tkinter προσφέρει τρεις διαφορετικούς διαχειριστές, ο καθένας εκτελεί διαφορετική λειτουργία. (Lundh, 1999)

- **Grid**: Ο συγκεκριμένος είναι ο πιο ευέλικτος και ισχυρός από τους τρεις. Η κεντρική του ιδέα είναι να οργανώνει τον χώρο ενός παραθύρου σαν έναν πίνακα δύο ή διαστάσεων, και σε κάθε κελί του να τοποθετεί το κάθε στοιχείο με βάση τον αριθμό της γραμμής και της στήλης του. Ο διαχειριστής διάταξης grid είναι ιδανικός για τη δημιουργία σύνθετων διατάξεων ή διατάξεων που ακολουθούν μια λογική σειρά, αλλά μπορεί να εφαρμοστεί και σε πιο απλές δομές επιτελώντας και πάλι σωστά τη λειτουργία του.
- **Pack**: Ο διαχειριστής διάταξης pack, λειτουργεί με πιο απλή λογική, αυτή της στοίβαξης στοιχείων το ένα κάτω από το άλλο, ή το ένα δίπλα από το άλλο. Υπάρχει η επιλογή να «κολλήσει» το στοιχείο στο οποίο αναφέρεται σε κάποια πλευρά (πάνω, κάτω, δεξιά, αριστερά), καθώς και το θα γεμίσει τον χώρο που του παρέχεται ή όχι. Χρησιμοποιείται συνήθως για πιο απλές γραμμικές διατάξεις.
- **Place**: Είναι ο πιο απλός από τους τρείς κι επιτρέπει στον προγραμματιστεί να καθορίσει την ακριβή θέση και το μέγεθος ενός στοιχείου δίνοντας του συγκεκριμένες συντεταγμένες. Γενικώς η χρήση του προτιμάται ιδιαίτερα, καθώς η διάταξη δεν μπορεί να προσαρμοστεί δυναμικά και κατά την αλλαγή του παραθύρου το στοιχείο μένει στην ακριβής θέση που του έχει δοθεί. (Lundh, 1999)

3.3.5 Event - Προγραμματισμός

Η λειτουργία μιας εφαρμογής Tkinter βασίζεται κατά πολύ μεγάλο βαθμό από τα γεγονότα (events). Αυτό σημαίνει ότι αντί να εκτελεί μια σειρά από εντολές με συγκεκριμένη σειρά και έπειτα να τερματίζει, η εφαρμογή κατά το μεγαλύτερο της διάστημα είναι σε διαδικασία αναμονής, αντιδρώντας σε ενέργειες που προκαλούνται από τον χρήστη ή το ίδιο λειτουργικό σύστημα. (Lundh, 1999)

3.3.5.1 Ποια είναι τα γεγονότα (events);

Τα γεγονότα για τα οποία γίνεται λόγος είναι ουσιαστικά «σήματα» που δείχνουν ότι έχει εκτελεστεί μια συγκεκριμένη ενέργεια. Οι πηγές, αυτών είναι διάφορες, αλλά όλες τους σηματοδοτούν ότι υπήρξε αλληλεπίδραση από τον χρήστη. Ορισμένα παραδείγματα παρόμοιων γεγονότων είναι:

- **<Button-1>**: Δείχνει το αριστερό πάτημα του ποντικιού πάνω σε κάποιο widget. Γνωρίζουμε ότι είναι το αριστερό καθώς ο αριθμός 1 είναι για το αριστερό, ο αριθμός 2 για το μεσαίο, και ο αριθμός 3 για το δεξί. Κατά το πάτημα του ποντικιού η εφαρμογή λαμβάνει τις συντεταγμένες του x και y.
- **<Double-Button-1 >**: Αντιπροσωπεύει το διπλό κλικ με το αριστερό κουμπί του ποντικιού.
- **<Enter>:** Ενεργοποιείται όταν ο δείκτης του ποντικιού εισέρχεται στην περιοχή ενός widget. Το συγκεκριμένο γεγονός δεν εμπλέκεται με το πάτημα του κουμπιού «Enter» στο πληκτρολόγιο.
- **<Leave>:** Λειτουργεί αντίθετα από το <Enter>, δηλαδή ενεργοποιείται όταν ο δείκτης του ποντικιού εξέρχεται από την περιοχή του widget.
- **<Return>:** Αντιπροσωπεύει το πάτημα του πλήκτρου Enter στο πληκτρολόγιο, αλλά μπορεί να χρησιμοποιηθεί και για άλλα ειδικά πλήκτρα όπως Escape, Tab, F1,..F12.
- **<Key>:** Ενεργοποιείται με το πάτημα οποιουδήποτε πλήκτρου στο πληκτρολόγιο. Οποιοσδήποτε χαρακτήρας έχει πατηθεί, γίνεται διαθέσιμος στην μεταβλητή.
- **<Configure>:** Δεν προκαλείται από τον χρήστη, αλλά από τον διαχειριστή παραθύρων. Ενεργοποιείται αυτόματα όταν το widget αλλάζει μέγεθος ή θέση. (Lundh, 1999)

3.3.5.2 Πως συνδέουμε ένα γεγονός με κάποια ενέργεια;

Προκειμένου να αποκτήσουν νόημα τα γεγονότα που αναφέρθηκαν, το Tkinter παρέχει έναν ισχυρό μηχανισμό που «συνδέει» (bind) τα γεγονότα αυτά με συγκεκριμένες ενέργειες. Αυτό γίνεται αναθέτοντας σε κάθε γεγονός μια συνάρτηση χειρισμού (handler), η οποία συχνά αναφέρεται και ως «callback». Κάθε φορά μου συμβαίνει ένα γεγονός για το οποίο έχει οριστεί ένας χειριστής, το Tkinter καλεί αυτόματα την αντίστοιχη συνάρτηση. Η πιο απλή και συνηθισμένη μορφή είναι συνήθως μέσω των κουμπιών με τη παράμετρο 'command'. Ορίζεται συνήθως ως 'command= function' και κάθε φορά που ενεργοποιείται το κουμπί με κλικ, εκτελείται ο κώδικας της συνάρτησης 'function'. (Lundh, 1999)

Εκτός από την ανάθεση μιας προκαθορισμένης συνάρτησης, το Tkinter επιτρέπει την χρήση την λεγόμενων συναρτήσεων "lambda" για τη δημιουργία μικρών ανώνυμων συναρτήσεων απευθείας μέσα στη σύνδεση του γεγονότος. Αυτή η τεχνική είναι ιδιαίτερα χρήσιμη για απλές, μονογραμμικές ενέργειες που δεν χρειάζεται να οριστούν ξεχωριστά. Για παράδειγμα η συνάρτηση lambda μέσω της παράμετρού 'command', επιτρέπει το πέρασμα ορισμάτων σε συνάρτηση χειριστή, κάτι που κανονικά δεν είναι εφικτό. Χαρακτηριστικό παράδειγμα αποτελεί η δημιουργία κουμπιών απαντήσεων, όπου η lambda δημιουργεί για προσωρινή συνάρτηση που «θυμάται» την τιμή της επιλογής στην οποία έγινε κλικ. (Lundh, 1999)

3.3.5.3 Πως παραμένει ενεργή η εφαρμογή;

Ο ρόλος της mainloop() είναι να συντονίσει τη διαδικασία που αναφέρθηκε πιο πάνω. Δηλαδή, όταν καλείται αυτή η μέθοδος, συνήθως συμβαίνει στα αρχικά στάδια της εφαρμογής και σταματά να εκτελείται γραμμικά ο κώδικας και πλέον λειτουργεί στην κατάσταση συνεχούς ακρόασης για γεγονότα. Το πρόγραμμα, λοιπόν, περιμένει την εμφάνιση κάποιου γεγονότος και μόλις εντοπίσει κάποιο, ελέγχει εάν υπάρχει συνδεδεμένη συνάρτηση χειριστής σε αυτό και την εκτελεί. Εφόσον η συνάρτηση ολοκληρωθεί, το πρόγραμμα επιστρέφει στην κατάσταση αναμονής για το επόμενο γεγονός. Ο κύκλος αυτός συνεχίζεται αδιάκοπα μέχρι ο χρήστης να τερματίσει το πρόγραμμα, συνήθως μέσω κλείνοντας το κύριο παράθυρο της εφαρμογής. (Lundh, 1999)

Ένα βασικό ακόμη στοιχείο για τη λειτουργία αναμονής χρόνου για κάποιο γεγονός είναι η μέθοδος 'after()', η οποία ανήκει στον βρόχο της mainloop και επιτρέπει τον προγραμματισμό της εκτέλεσης μια συνάρτησης μετά την πάροδο συγκεκριμένου χρονικού διαστήματος. Ο μηχανισμός της είναι σημαντικός για δυναμικά και μη γεγονότα που βασίζονται σε χρονικά όρια για να εκτελεστούν. (Lundh, 1999)

3.4 Αλγόριθμος Johnson – Trotter

3.4.1 Ορισμός

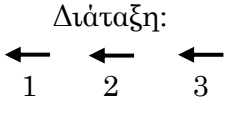
Ο αλγόριθμος Johnson – Trotter είναι μια μέθοδος που χρησιμοποιείται για την παραγωγή όλων των πιθανών αναδιατάξεων (permutations) των ψηφίων ενός αριθμού. Το βασικό του χαρακτηριστικό είναι ότι κάθε νέα διάταξη που παράγεται προκύπτει από την αμέσως προηγούμενη με την αντιμετάθεση δύο γειτονικών στοιχείων. Ο αλγόριθμος λειτουργεί ορίζοντας κατεύθυνση σε κάθε στοιχείο του (αριστερά ή δεξιά) και ορίζει έναν αριθμό ως «κινητό», όταν είναι μεγαλύτερος από τον γειτονικό αριθμό στον οποίο δείχνει η κατεύθυνση του. Η διαδικασία επαναλαμβάνεται μέχρι να μην υπάρχει κανένας «κινητός» αριθμός, που σημαίνει ότι έχουν παραχθεί όλες οι δυνατές διατάξεις. Αξίζει να σημειωθεί ότι ο αλγόριθμος αυτός λειτουργεί για ακολουθίες αριθμών που έχουν την μορφή 1,2.3,…n δηλαδή όπως 123 και 123456. Παρακάτω φαίνεται ο αλγόριθμος γραμμένος σε ψευδογλώσσα. (Bogomolny, 2018)

```
Initialize the first permutation with <1 <2 ... <n
while there exists a mobile integer
  find the largest mobile integer k
  swap k and the adjacent integer it is looking at
  reverse the direction of all integers larger than k
```

3.4.2 Παράδειγμα

Προτού εμβαθύνουμε στο στάδιο εφαρμογής του αλγορίθμου σε κώδικα σε επόμενο κεφάλαιο, αξίζει να αναφερθούμε σε ένα πιο απλό παράδειγμα για να εξηγηθεί ακριβώς πως θα λειτουργούσε. Έστω λοιπόν ότι δοθεί η αρχική διάταξη «1 2 3». Θα αναλυθεί,, πως καταλήγει ο συγκεκριμένος αλγόριθμος να δημιουργήσει όλες τις πιθανές αναδιατάξεις. (Bogomolny, 2018)

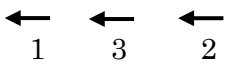
- **Αρχή:**



Στην αρχή οι κατευθύνσεις είναι πάντα στραμμένες προς τα αριστερά. Για να μπορεί πρώτα να κινηθεί το μεγαλύτερο ψηφίο, που βρίσκεται στα δεξιά.

- **Βήμα 1:**

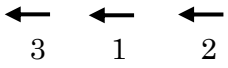
Διάταξη:



Βρίσκουμε τον μεγαλύτερο κινητό αριθμό (μεγαλύτερος από τον γείτονα στον οποίο δείχνει), εδώ είναι το «3» και αλλάζει θέσεις με το 2.

- **Βήμα 2:**

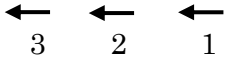
Διάταξη:



Βρίσκουμε τον επόμενο κινητό αριθμός (πάλι το 3) και αλλάζει θέση με τον γείτονά στο οποίο δείχνει (το 1).

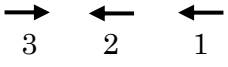
- **Βήμα 3:**

Διάταξη:



Πλέον το 3 δεν είναι κινητό (δεν δείχνει σε μικρότερο). Ο επόμενος μεγαλύτερος είναι το 2, το οποίο δείχνει στο 1, οπότε αλλάζουν θέση.

Διάταξη:



Επιπλέον, επειδή έγινε κίνηση του 2, όλοι οι αριθμοί μεγαλύτεροι του 2, αλλάζουν κατεύθυνση.

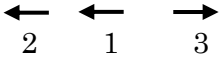
- **Βήμα 4:**

Διάταξη:

Βρίσκουμε τον επόμενο κινητό αριθμό, το 3 που δείχνει προς το μικρότερο του 2 και αλλάζουν θέση μεταξύ τους.

- **Βήμα 5:**

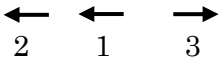
Διάταξη:



Βρίσκουμε τον επόμενο κινητό αριθμό, το 3 που δείχνει προς το μικρότερο του 1 και αλλάζουν θέση μεταξύ τους.

- **Τέλος:**

Διάταξη:



Δεν υπάρχει άλλος κινητός αριθμός οπότε ο αλγόριθμος τερματίζει και έχει δημιουργήσει κάθε πιθανή ακολουθία για το 123.

Τελικές ακολουθίες ψηφίων που δημιουργήθηκαν:

1	2	3	4	5	6
123	132	312	321	231	213

Παρατηρούμε ότι κάθε δυο διαδοχικές διατάξεις διαφέρουν μόνο με την ανταλλαγή δύο γειτονικών στοιχείων. Αυτό σημαίνει ότι οι διατάξεις που είναι κοντά στη σειρά που παράγεται από τον αλγόριθμο έχουν μεγάλες ομοιότητες μεταξύ τους. Μπορεί να φανεί για παράδειγμα στις ακολουθίες 1 και 2, 123 και 132, διαφέρει μόνο η αναστροφή των αριθμών 2 και 3 που βρίσκονται στις δύο τελευταίες θέσεις. Αντίστοιχα, παρατηρείται το ίσιο και για τις ακολουθίες 1 και 6, 123 και 213, όπου διαφέρουν μόνο τα πρώτα δύο ψηφία και το τρίτο παραμένει ίδιο.

3.4.3 Αποδοτικότητα Αλγορίθμου

Ολοκληρώνοντας, ο αλγόριθμος Jonhson – Trotter ξεχωρίζει για τον έξυπνο και αποδοτικό τρόπο λειτουργίας του. Η κεντρική του ιδέα είναι ο κανόνας της ελάχιστης δυνατής αλλαγής, δηλαδή για να βρει την επόμενη διάταξη αλλάζει θέση μόνο σε δύο γειτονικά στοιχεία. Αυτός ο απλός κανόνας είναι εξαιρετικά ισχυρός καθώς εγγυάται ότι ο αλγόριθμος δημιουργεί μια τέλεια διαδρομή, περνώντας από κάθε πιθανή διάταξη ακριβώς μία φορά και κάνοντας πάντα το μικρότερο δυνατό βήμα. Στην πράξη αυτό τον καθιστά πολύ γρήγορο και χρήσιμο. (Wikipedia contributors., 2025)

4.Εφαρμογή και Ανάλυση Κώδικα

4.1 Εισαγωγή

Η εφαρμογή που αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας είναι ένα αυτόνομο, εκπαιδευτικό εργαλείο, σχεδιασμένο για την ανίχνευση μαθησιακού επιπέδου στα αγγλικά χρησιμοποιώντας τις γνωστικές δεξιότητες των μαθητών. Δημιουργήθηκε με τη χρήση της γλώσσας προγραμματισμού Python και της βιβλιοθήκης Tkinter για να την αναπαράσταση του γραφικού περιβάλλοντος Η ρόλος της εφαρμογής διαχωρίζεται σε δύο βασικούς τομείς, αυτός του μαθητή και του Καθηγητή.

Όταν ένας μαθητής εκκινεί την εφαρμογή, αρχικά καταχωρεί τα βασικά του στοιχεία, κι αμέσως μετά οδηγείται σε μια σειρά από τέσσερεις δοκιμασίες, καθεμιά από τις οποίες στοχεύει στην αξιολόγηση διαφορετικής γνωστικής λειτουργίας. Εξετάζεται η λεξιλογική κατανόηση μέσω από ερωτήσεις πολλαπλής επιλογής, με τέσσερις δυνατές απαντήσεις σε κάθε ερώτηση. Σύνολο υπάρχουν 50 ερωτήσεις που συνδυάζουν και γραμματική, ωστόσο εάν ο μαθητής απαντήσει σωστά 9 ερωτήσεις στη σειρά σωστά η δοκιμασία σταματάει με επιτυχία. Έπειτα ακολουθεί η δοκιμασία που εξετάζει την οπτική εργαζόμενη μνήμη, μέσω της παρουσίασης ακολουθιών αριθμών τις οποίες ο μαθητής πρέπει να ανακαλέσει. Η δυσκολία, εδώ κλιμακώνεται δυναμικά σε τρία επίπεδα, αναλόγως με τις επιδόσεις του. Συγκεκριμένα, υπάρχουν 3 επίπεδα με 5 δοκιμασίες το κάθε ένα, και για κάθε 3 συνεχόμενες σωστές απαντήσεις ο χρήστης προχωράει στο επόμενο επίπεδο, ενώ για 3 συνεχόμενες λάθος απαντήσεις η δοκιμασία σταματάει αμέσως και τερματίζει. Για την ανάλυση των απαντήσεων εφαρμόζεται ο αλγόριθμος Johnson-Trotter για τον εντοπισμό μετάθεσης ακολουθίας. Στη συνέχεια, εμφανίζεται δοκιμασία οπτικοχωρικής αντίληψης, η οποία λειτουργεί μέσω συμπλήρωσης 15 οπτικών λογικών μοτίβων. Τέλος, αξιολογείται η ανασταλτική ικανότητα του μαθητή μέσω δοκιμασίας Go/No-Go, στην οποία καλείται από εικόνες που εμφανίζονται να πατάει με το ποντίκι σε συγκεκριμένες μόνο. Υπάρχουν σύνολο 10 διαφορετικές εικόνες και 4 από αυτές αποτελούν εικόνες στόχο.

Όταν ο μαθητής τερματίσει με τη διαδικασία των δοκιμασιών, οι απαντήσεις του, μαζί με κάποιες ακόμη παραμέτρους, διαφορετικές για καθεμιά δοκιμασία, περνούν από συγκεκριμένο μοντέλο κατηγοριοποίησης. Το μοντέλο αναλύει τα δεδομένα και δίνει τελική κατηγορία για τον μαθητή "Κάτω από τον Μέσο Όρο", "Εντός Μέσου όρου", ή "Πάνω από τον Μέσο Όρο" για κάθε μεμονωμένο τεστ από τα τέσσερα, παρέχοντας ταυτόχρονα και ποσοστό βεβαιότητάς (confidence) για την πρόβλεψη του. Στο τέλος, ένα ακόμη μοντέλο κατηγοριοποίησης, λαμβάνει τα τις κατηγορίες που ανέθεσαν τα προηγούμενα μοντέλα για κάθε δοκιμασία και εξάγει τη συνολική ολιστική αξιολόγηση του μαθητή.

Όταν ο ανοίξει την εφαρμογή, ο εκπαιδευτικός, με συγκεκριμένα στοιχεία πρόσβασης μπορεί να δει τα συγκεντρωμένα αποτελέσματα όλων των μαθητών που έχουν ολοκληρώσει το κουίζ. Τα αποτελέσματα στα οποία έχει πρόσβαση είναι η λίστα των μαθητών με τα βασικά τους στοιχεία. Μια πιο συνοπτική λίστα, όπου επιλέγοντας έναν μαθητή εμφανίζεται μια σύνοψη των επιδόσεων του για κάθε τεστ και μια λεπτομερής λίστα ανάλυσης της κάθε δοκιμασίας ξεχωριστά.

4.2 Δεδομένα

Η λειτουργία της εφαρμογής βασίζεται κατά ένα μεγάλο ποσοστό στο σύνολο των δεδομένων που της παρέχεται. Συγκεκριμένα, υπάρχουν αρχεία δεδομένων σε μορφή JSON, η οποία επιτρέπει την αποθήκευση δεδομένων με ιεραρχημένη δομή, με ετικέτες και περιεχόμενα. Η μορφή αυτή χρησιμοποιείται για τις ερωτήσεις που εμφανίζονται σε κάθε δοκιμασία και για το τελικό αρχείο στο οποίο αποθηκεύονται και ανακαλούνται οι απαντήσεις και τα αναλυτικά στοιχεία του κάθε μαθητή. Χρησιμοποιούνται και αρχεία CSV τα οποία λειτουργούν ως απλοί πίνακες δεδομένων με γραμμές και στήλες και για αυτό τον

λόγο χρησιμοποιούνται για την φόρτωση των συνολικών δεδομένων που προορίζονται για εκπαίδευση των μοντέλων τεχνητής νοημοσύνης.

4.2.1 Ερωτήσεις Δοκιμασιών – questions.json

Το αρχείο των ερωτήσεων (questions.json) αποτελεί τη κεντρική ρίζα για ολόκληρη την εφαρμογή. Είναι δομημένο ως ένας πίνακας από αντικείμενα. Κάθε αντικείμενο μέσα στον πίνακα αντιπροσωπεύει μια ξεχωριστή ερώτηση ή δοκιμασία για τον χρήστη. Κάθε ερώτηση ορίζεται από συγκεκριμένους τύπους που την καθορίζουν ως μοναδική. Αυτό καθορίζει τη σειρά εκτέλεσης του κουίζ καθώς δεν είναι τυχαία τοποθετημένες, αλλά σειριακά, δηλαδή πρώτα όλες οι λεξικολογικές, μετά όλες οι μνήμης, οι οπτικοχωρικές και τέλος η Go/No-Go.


```
{
    "type": "vocabulary",
    "question": "My father _____ football every Sunday.",
    "options": [
      "plays",
      "play",
      "is playing",
      "played"
    ],
    "correct_answer": "plays"
  },
  {
    "type": "memory_check",
    "question": "Remember the sequence of numbers",
    "random_sequence": true
  },
 {
    "type": "visual_pattern",
    "question": "Complete the pattern:",
    "question_image": "Images\\optikoxor_antilipsi_im3.png",
    "options": [
      "Images\\im3a.png",
      "Images\\im3b.png",
      "Images\\im3c.png",
      "Images\\im3d.png"
    ],
    "correct_answer": "Images\\im3b.png"
  },
{
    "type": "go_no_go",
    "question": "Click only when you see the CAT image.\nDon't click on other images.",
    "target_image": "Images\\GoNogo\\Cat.png",
    "images": [
      "Images\\GoNogo\\penguin.png",
      "Images\\GoNogo\\duck.png",
      "Images\\GoNogo\\dog.png",
      "Images\\GoNogo\\panda.png"
    ]
  }
```


Από το παραπάνω σχήμα φαίνεται η διαφορετική δομή που υπάρχει για την κάθε ερώτηση, με βασική διαφορά την αναφορά στο κλειδί "type". Η δοκιμασία του Λεξιλογίου και των Οπτικών μοτίβων έχουν όμοια μορφή στο αρχείου και επίσης είναι σύνολο 50 και 15 σε αριθμό αντίστοιχα. Η δοκιμασία της Μνήμης δεν περιέχει επιλογές ή σωστή απάντηση, καθώς αυτές δημιουργούνται κατά τη διάρκεια λειτουργίας της εφαρμογής και είναι σύνολο 15 σε αριθμό, καθώς αυτό είναι το μέγιστο δυνατό πλήθος τους. Η δοκιμασία Go/No-Go ορίζεται από μια εικόνα στόχο και μια λίστα από τις υπόλοιπες εικόνες όχι στόχους. Η τελευταία είναι μια ερώτηση μόνο στο αρχείο, καθώς κατά τη λειτουργία του προγράμματος η συγκεκριμένη δοκιμασία αντιμετωπίζεται ως μια μεμονωμένη συνεχή ερώτηση.

4.2.2 Στοιχεία Μαθητών–student_results.json

Αφού ο μαθητής ολοκληρώσει όλες τις δοκιμασίες, η εφαρμογή διατηρεί τα στοιχεία της απόδοσης του και το αποθηκεύει στο αρχείο "student_results.json". Κάθε φορά που ένας νέος μαθητής ολοκληρώνει το κουίζ, προστίθεται μια νέα, μεγάλη εγγραφή μέσα σε αυτό το αρχείο που περιέχει και τις εγγραφές από προηγούμενους μαθητές. Κάθε τέτοια εφαρμογή αποτελεί έναν ενιαίο φάκελο πληροφοριών χωρισμένο σε διακριτές ενότητες για καλύτερη οργάνωση.


```
"info": {
            "name": "maria",
            "surname": "test",
            "age": "9",
            "class": ""
        },
        "answers": [
            {
                "answer": "is playing",
                "correct": false,
                "type": "vocabulary",
                "time": 30.22
            },
```


Το παραπάνω αποτελεί απόσπασμα από τα δεδομένα που συγκρατεί το συγκεκριμένο αρχείο για τον κάθε μαθητή. Παρακάτω στο αρχείο υπάρχουν κι οι αναλυτικές απαντήσεις όπως μέσος όρος χρόνου για κάθε δοκιμασία, καθώς και τα αποτελέσματα κατηγοριοποίησης για κάθε δοκιμασία μεμονωμένα αλλά και για την τελική κατηγοριοποίηση καθώς και το ποσοστό αυτοπεποίθησης για κάθε ένα από αυτά. Το συγκεκριμένο αρχείο χρησιμοποιείται στη εφαρμογή και όταν πρέπει να αναδείχνουν τα αποτελέσματα από προηγούμενους μαθητές που εκτέλεσαν τη δοκιμασία.

4.2.3 Βάσεις Δεδομένων για κάθε δοκιμασία

Προτού αναφερθούμε στις βάσεις δεδομένων για την κάθε δοκιμασία, αξίζει να αναφερθεί ότι τα δεδομένα είναι τεχνητά και δεν έχουν παρθεί από κάποιο ιστότοπο στο διαδίκτυο. Η λογική με την οποία είναι φτιαγμένα βασίζεται σε μια έρευνα, η οποία εξετάζει την λειτουργικότητα παρόμοιας εφαρμογής επάνω σε παιδία τυπικής ανάπτυξης και παιδία με δυσλεξία. Κατά την διαδικασία της εκτέλεσης της δοκιμασίας της έρευνας η εφαρμογή Askisi-Lexia καταμετρούσε ορθές απαντήσεις και χρόνο απαντήσεων των μαθητών και κατέληξε σε συγκεκριμένους χρόνους και ορθές απαντήσεις που έδωσε ο μέρος όρος της κάθε ομάδας (Zygouris, 2025) .Στην συγκεκριμένη εφαρμογή, έχουμε πάρει τους αντίστοιχους

χρόνους και ορθές απαντήσεις και τα έχουμε παραμετροποιήσει στα δικά μας δεδομένα έτσι ώστε να προκύπτει λογικά η κατηγοριοποίηση για κάθε μαθητή.

4.2.3.1 Λεξικολογικά Δεδομένα – Vocabulary

Στην έρευνα των Zygouris, Nikolaos C., et al. γίνεται λόγος για 15 ερωτήσεις λεξιλογικού τύπου, με τα παιδιά τυπικής ανάπτυξης να απαντούν κατά μέσο όρο 11.33 με μέση διαφορά 4.41. Ενώ για τα παιδιά με δυσλεξία, ο μέσος όρος των σωστών απαντήσεων ανάγεται σε 8.91 με μέση διαφορά 5.40. Στην δική μας εφαρμογή όμως, γίνεται λόγος για 50 ερωτήσεις σύνολο, με δυνατότητα τερματισμού στις 9 συνεχόμενες σωστές. Επομένως, δεν υφίσταται η συγκράτηση των σωστών απαντήσεων ως δεδομένο, αλλά πρέπει να ληφθεί υπόψιν το ποσοστό των σωστών απαντήσεων.

	Askisi-Lexia	**Παρούσα Εφαρμογή**
Ερωτήσεις	16	50
Μαθητές Τυπικής Ανάπτυξης	Σωστά: 11.33 T.A*: 4.41	Σωστά: 75% T.A: 29% (Σε Πουοστά)
Μ.Ο. Χρόνος Απάντησης -	2.13 T.A: 1.1	2.13 T.A: 1.1
Μαθητές με δυσλεξία	Σωστά: 8.91 T.A: 5.40	Σωστά: 59% T.A: 36% (Σε Πουοστά)
Μ.Ο. Χρόνος Απάντησης -	2.18 T.A: 1.06	2.18 T.A: 1.06

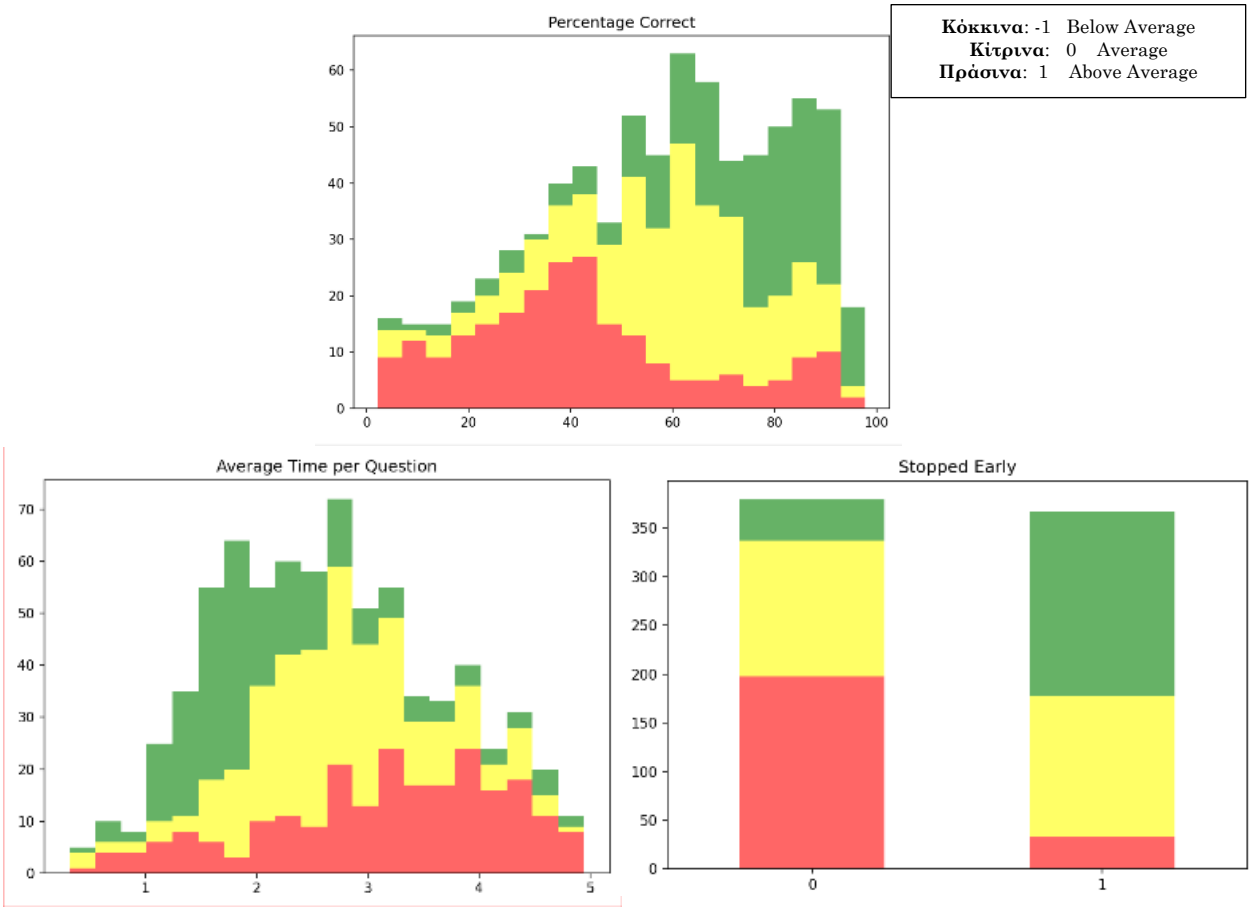
*T.A.= Τυπική Απόκλιση

Στον παραπάνω πίνακα λοιπόν φαίνονται τα αρχικά στοιχεία της εφαρμογής Askisi-Lexia και δίπλα πως έχουν προσαρμοστεί τα δεδομένα σε ποσοστά για να τα χρησιμοποιήσουμε ως βάση, για τη δημιουργία των δεδομένων μας. Ο χρόνος παραμένει σταθερός. Η βάση δεδομένων για τη συγκεκριμένη δοκιμασία περιέχει τα χαρακτηριστικά: «percentage_correct» που αντικατοπτρίζει το ποσοστό των σωστά απαντημένων ερωτήσεων του μαθητή, «stopped_early» που δείχνει το ένα ο μαθητής σταμάτησέ πριν ή μετά της 25η ερώτηση, με ένα ‘0’ για όχι και ‘1’ για ναι, για να γνωρίζουμε να δώσουμε και κάποια παραπάνω βαρύτητα επιτυχίας στον μαθητή που τερμάτισε την δοκιμασία νωρίτερα. Τέλος το χαρακτηριστικό «avg_time_per_question» που δίνει τον μέσο όσο απάντησης ανά ερώτηση. Ο όρος «classification» υποδηλώνει την έξοδο, δηλαδή τη τελική απόφαση που

δόθηκε για τον κάθε μαθητή αναλόγως τις τιμές των χαρακτηριστικών του και παίρνει τιμές ‘-1’, ‘0’ και ‘1’ αναλόγως.

percentage_correct	stopped_early	avg_time_per_question	classification
7.3	0	3.64	-1
95.3	1	2.43	-1
28.87	0	0.54	-1
52.3	0	0.6	-1
2.16	0	0.65	-1
79.9	1	2.4	0
65.9	1	2.4	0
70.6	1	2.4	0
59.9	1	2.41	0
18.54	0	2.43	0
57.8	1	1.61	1
93.0	1	1.61	1
92.5	0	1.63	1
78.3	1	1.64	1
86.3	1	1.64	1

Στην εικόνα παραπάνω φαίνεται ενδεικτικά τι μορφή έχει η βάση δεδομένων του λεξιλογίου και παρακάτω θα δείξουμε τη σχέση του κάθε χαρακτηριστικού ως προς την συχνότητα.



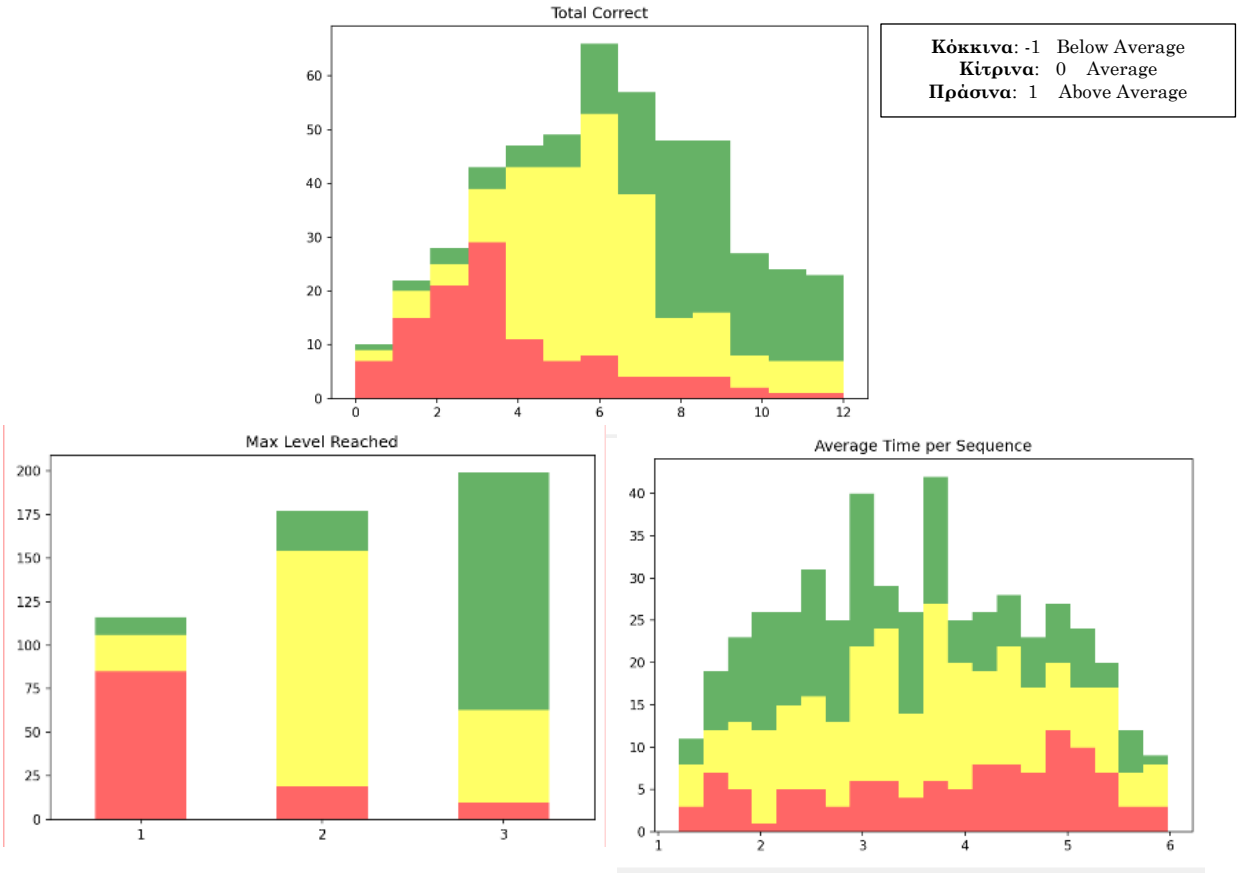
Για παράδειγμα για το χαρακτηριστικό «Percentage Correct» παρατηρούμε ότι όσο χαμηλή είναι η τιμή του 0-45%, αυτό αντιστοιχεί σε περισσότερα κόκκινα σημεία, δηλαδή '-1' «Below Average», αντίστοιχα στις μεσαίες τιμές 50-65% κυριαρχεί το κίτρινο, δηλαδή η κλάση '0' «Average» και στις τιμές 70-100% είναι κυρίως το πράσινο δηλαδή '1' «Above Average». Με όμοιο τρόπο λειτουργούν και τα γραφήματα των υπόλοιπων χαρακτηριστικών.

4.2.3.2 Δεδομένα Μνήμης – Memory

Ομοίως και για αυτή τη δοκιμασία τα στοιχεία στα οποία βασίζεται η βάση δεδομένων καλύπτεται από την εφαρμογή Askisi-Lexia. Στην εφαρμογή της έρευνας καταμετρώνται πόσες επιτυχείς προσπάθειες ανάκλασης είχε ο κάθε μαθητής. Οι τυπικής ανάπτυξης έδειξαν μέσο όρο 3.92 με Μέση Απόκλιση 1.74 και οι μαθητές με δυσλεξία έδειξαν 2.50 με μέση απόκλιση 1.47. Στην δική τους προσέγγιση όμως, ο μαθητής μετά από δύο συνεχόμενα σωστά προχωρούσε σε επόμενο επίπεδο, ενώ η δική μας προχωράει στα 3 συνεχόμενα, αυτό σημαίνει ότι εάν μαθητής της εφαρμογής Askisi-Lexia έκανε 4 σωστά συνεχόμενα θα βρισκόταν στο επίπεδο με ακολουθίες 5 ψηφία, ενώ στη δική μας εφαρμογή για να γίνει κάτι τέτοιο πρέπει ο μαθητής να κάνει σωστά 6 συνεχόμενα. Ο μαθητής ωστόσο με δυσλεξία έδειξε μέσο όρο 2.50, το οποίο σημαίνει ότι δεν ξεπέρασε τις ακολουθίες με 4 ψηφία (δικό μας level 2). Επομένως γνωρίζουμε με σιγουριά ότι μαθητές που δεν ξεπέρασαν το επίπεδο 1 είναι πιο πιθανό να είναι «Κάτω του μέσου όρου». Για το επίπεδο 2, οι μαθητές κατά προσέγγιση θα πρέπει να έχουν 3-5 προσπάθειές για να εισέλθουν σε αυτό, επομένως το λογικό είναι εδώ να ανήκουν μαθητές «Μέσου όρου» που δεν έφτασαν σε ακολουθίες με 5 ψηφία. Και τέλος στο επίπεδο 3 όπου για να φτάσει κάποιος πρέπει να έχει κάνει 6 μέχρι και 10 το πολύ προσπάθειες, είναι πιο λογικό να ανήκουν μαθητές «Άνω του Μέσου όρου»

total_correct	max_level_reached	avg_time_per_sequence	classification
1	1	1.2	0
9	3	1.2	1
2	1	1.21	-1
8	2	1.24	1
0	1	1.25	-1
2	1	1.25	-1
6	3	1.3	0
8	2	1.3	1
4	1	1.38	0
6	2	1.4	0
6	1	1.4	0
2	1	1.45	-1
3	1	1.47	-1
3	1	1.49	-1

Στη δική μας βάση δεδομένων πέρα από τα σωστά ('total_correct'), έχουμε προσθέσει 'max_level_reached' για να αποτελεί βέβαιο κριτήριο βαρύτητα το επίπεδο στο όποιο κατάφερε να φτάσει ο μαθητής, καθώς και 'avg_time_per_sequence' για να υπάρχει ένας μέσος χρόνος που να παίζει και αυτό ρόλο κατά την κατηγοριοποίηση του μαθητή.



Από τα παραπάνω σχήματα, αναλύοντας για αρχή το 'total_correct' μπορούμε να παρατηρήσουμε ότι οι περισσότερες τιμές μεταξύ 0-3 σωστά (δεν έχει ξεπεραστεί το επίπεδο 1) είναι κόκκινες, τιμές μεταξύ 4-7 (έχει ξεπεραστεί ενδεχομένως το επίπεδο 1 και ίσως και το επίπεδο 2) έχει περισσότερες κίτρινες τιμές, ενώ τιμές 8-12 (μέγιστες δυνατές σωστές απαντήσεις είναι 12 και όχι 15) είναι κυρίως πράσινες. Βλέπουμε να υπάρχει και κυριαρχία του κάθε στοιχείου στο επίπεδο του στο 'Max Level Reached' σχεδιάγραμμα. Ενώ στο διάγραμμά του χρόνου, είναι πιο «ελεύθερο», καθώς υπάρχει πιθανότητα μαθητής που έχει φτάσει το επίπεδο 3 να χρειάζεται περισσότερο χρόνο πληκτρολόγησης διότι είναι περισσότερα τα ψηφία. Επομένως, πιο βασικά από τα δύο αυτά είναι τα πρώτα δυο χαρακτηριστικά.

4.2.3.3 Δεδομένα Οπτικών Μοτίβων – Visual Pattern

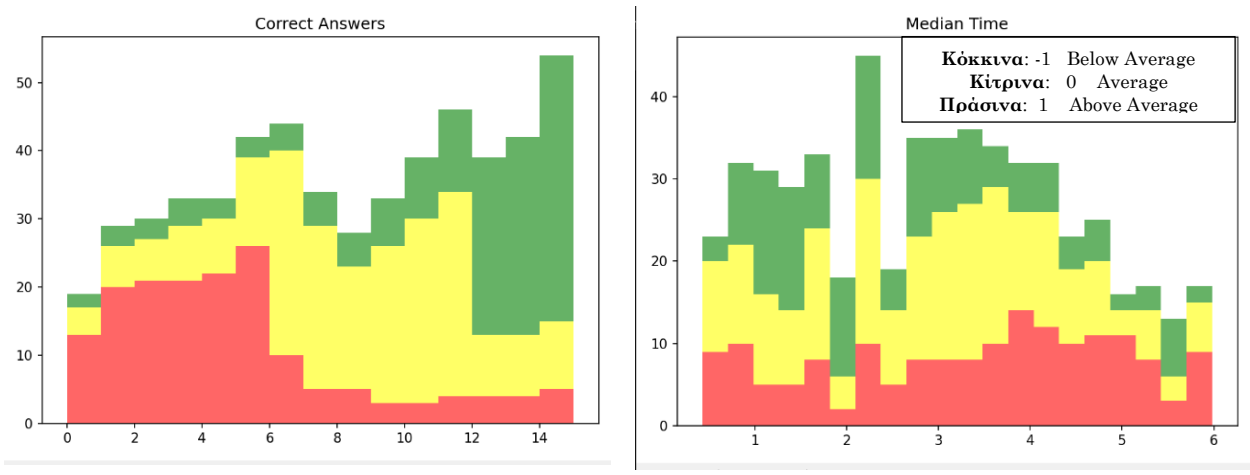
Όπως για τα προηγούμενα, έτσι και εδώ, η δημιουργία της βάσης δεδομένων έχει εμπνευστεί αρκετά από την εφαρμογή Askisi-Lexia. Στην συγκεκριμένη, έχουν χρησιμοποιηθεί 11 ερωτήσεις, ενώ στη δική μας 15. Ταυτόχρονα, στην έρευνα έχει αποδειχθεί ότι παιδιά τυπικής ανάπτυξης είχαν κατά μέσο όρο 7.57 σωστές απαντήσεις με τυπική απόκλιση 2.30, ενώ παιδιά με δυσλεξία είχαν κατά μέσο όρο 6.20 σωστές με απόκλιση 2.59. Ο χρόνος που έκαναν κατά μέσο όρο τα παιδιά με τυπική ανάπτυξης ήταν 2.59 με απόκλιση 1.39, ενώ τα παιδιά με δυσλεξία 2.63 με απόκλιση 1.18. Παρακάτω φαίνεται ο πίνακας αναδιαμόρφωσης δεδομένων για να ταιριάσουν στην εφαρμογή μας.

	Askisi-Lexia		**Παρούσα Εφαρμογή**	
Ερωτήσεις	11		15	
Μαθητές Τυπικής Ανάπτυξης	Σωστά: 7.57	Τ.Α: 2.30	Σωστά: 10.32	Τ.Α: 3.14
Μ.Ο. Χρόνος Απάντησης -	2.59	Τ.Α: 1.39	2.59	Τ.Α: 1.39
Μαθητές με δυσλεξία	Σωστά: 6.20	Τ.Α: 2.59	Σωστά: 8.45	Τ.Α: 3.53
Μ.Ο. Χρόνος Απάντησης -	2.63	Τ.Α: 1.18	2.63	Τ.Α: 1.18

Στον παραπάνω πίνακα, λοιπόν φαίνονται οι μετατροπές που χρειάστηκαν να πραγματοποιηθούν για να ταιριάζουν τα δεδομένα με την εφαρμογή μας, σημαντικό είναι να τονιστεί ότι ο χρόνος παραμένει ίδιος και για τη δική μας εφαρμογή.

correct_answers	median_time	classification
4	0.43	-1
5	0.43	0
6	0.45	-1
11	0.5	0
10	0.51	0
1	0.52	-1
9	0.52	0
12	0.52	1
8	0.53	-1
6	0.54	-1
0	0.57	0
5	0.58	-1
4	2.17	1
14	2.19	1
6	2.2	0
5	2.21	0
9	2.21	0
1	2.23	-1
13	2.23	0
13	2.23	1
15	2.23	1
10	2.25	1
12	2.25	1
9	2.26	0
14	2.26	1

Παραπάνω φαίνεται ένα δείγμα από τη βάση δεδομένων για τη συγκεκριμένη δοκιμασία. Εδώ έχουμε χρησιμοποιήσει δυο χαρακτηριστικά μόνο, τα 'correct_answers' και "median_time" καθώς η φύση της δοκιμασίας δεν μας επιτρέπει να εφαρμόσουμε άλλα. Έχουν συμπεριληφθεί παραδείγματα κατά τα οποία ο μαθητής πατάει πολύ γρήγορα πάνω σε εικόνες με τυχαιότητα. Παρακάτω φαίνονται οι πίνακες ανάλυσης δεδομένων των δυο χαρακτηριστικών.



Από τους πίνακες αυτούς μπορούμε να παρατηρήσουμε για το χαρακτηριστικό ‘correct_answers’, ότι για τις τιμές από μηδέν έως πέντε, κυριαρχεί το κόκκινο ενώ για τις τιμές από 6 έως και 11 κυριαρχεί το κίτρινο και για τις τιμές 12 έως και 15 κυριαρχεί το πράσινο. Αντίστοιχα για το χαρακτηριστικό του χρόνου τα όρια δεν είναι τόσο ξεκάθαρα κάθε φορά αλλά κυρίως για σχετικά γρήγορους χρόνους δίνεται το πράσινο για μέτριος χρόνος όπως 2.5 έως 4 δευτερόλεπτα δίνεται το κίτρινο και το κόκκινο κυριαρχεί κυρίως από τα 4 έως και 6 δευτερόλεπτα.

4.2.3.4 Δεδομένα Go/No-Go

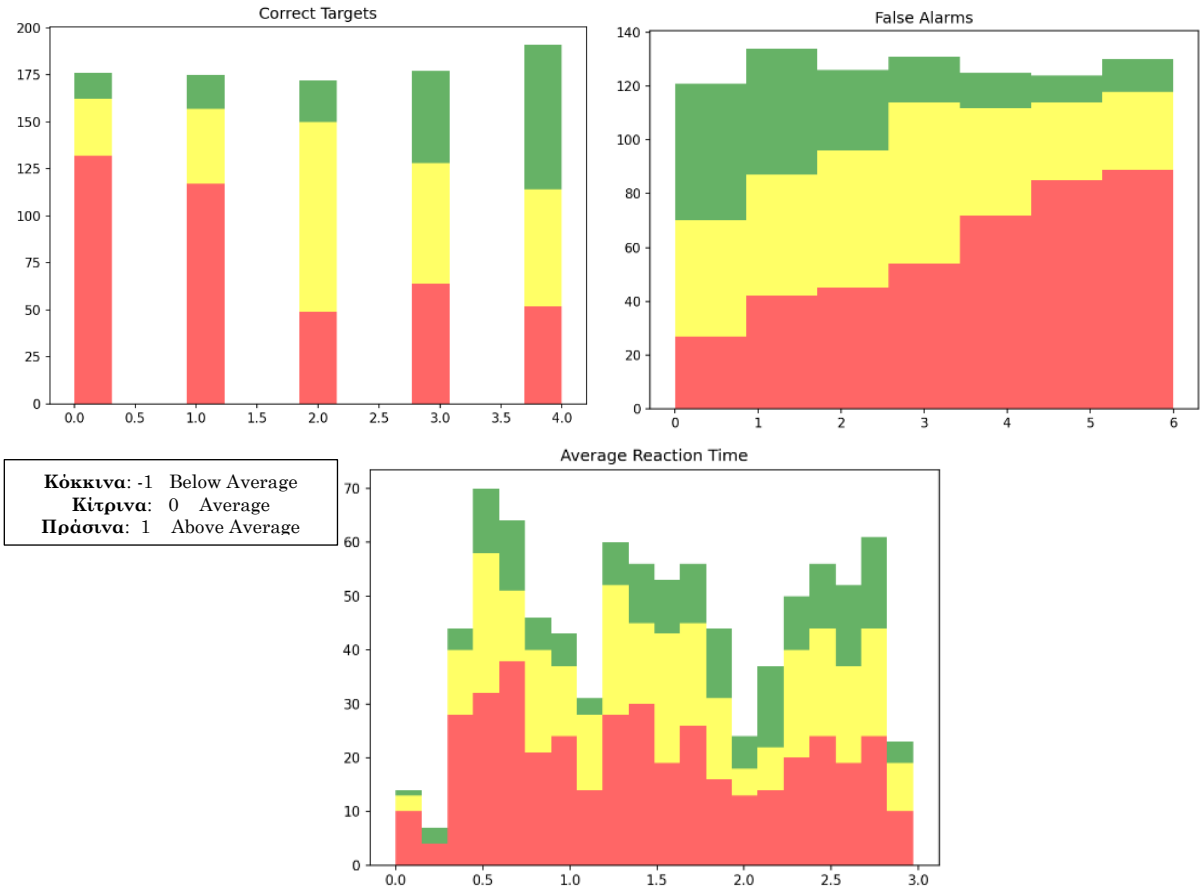
Για τη συγκεκριμένη δοκιμασία έχουμε αντλήσει πάλι δεδομένα από την έρευνα της εφαρμογής Askisi-Lexia. Στην συγκεκριμένη δοκιμασία η έρευνα έδειξε σύνολο από 10 φωτογραφίες στόχους ενώ η δική μας εφαρμογή έχει 10 εικόνες στο σύνολο και 4 εικόνες στόχους. Επιπλέον αναδείχθηκε ότι παιδιά τυπικής ανάπτυξης έφεραν κατά μέσο όρο 7.01 σωστές απαντήσεις με μέση απόκλιση 1.79. Ενώ παιδιά που είχαν δυσλεξία έφεραν 3.05 σωστά με απόκλιση 0.16. Δεν συγκρατήθηκε χρόνος στην εφαρμογή ωστόσο εμείς στη δικιά μας θα κρατήσουμε έναν τυπικό χρόνο. Όπως και προηγουμένως τα δεδομένα αυτά θα πρέπει διαμορφωθούν καθώς δεν αντιστοιχούν στον τρόπο λειτουργίας της δικής μας εφαρμογής.

	Askisi-Lexia		**Παρούσα Εφαρμογή**	
Ερωτήσεις - στόχοι	10		4	
Μαθητές Τυπικής Ανάπτυξης	Σωστά: 7.07	Τ.Α: 1.79	Σωστά: 2.8	Τ.Α: 0.72
Μαθητές με δυσλεξία	Σωστά: 3.05	Τ.Α: 0.16	Σωστά: 1.22	Τ.Α: 0.64

Παρακάτω φαίνεται ενδεικτικά παράδειγμα από τη βάση δεδομένων που έχει χρησιμοποιηθεί για τη συγκεκριμένη δοκιμασία. Έχει προστεθεί πέρα από τις σωστές απαντήσεις και τον χρόνο αντίδρασης, το ‘false_allarms’, το οποίο σηματοδοτεί πόσα εσφαλμένα κλικ έχει κάνει ο μαθητής επάνω σε εικόνες μη στόχους, για να μπορεί να υπάρχει βαρύτητα και στις λάθος επιλογές του μαθητή πέρα από τις σωστές.

correct_ta	false_alarr	avg_reacti	classificat
0	0	0	-1
3	5	1.10	-1
2	3	1.11	-1
2	4	1.11	0
3	3	1.11	0
4	1	1.11	1
1	5	1.12	-1
1	6	1.12	-1
2	5	1.12	-1
0	3	1.13	-1

4	2	2.24	1
4	6	2.24	0
0	2	2.25	-1
1	3	2.25	-1
1	6	2.25	-1
2	2	2.25	1
3	5	2.25	-1
1	1	2.26	-1
4	3	2.26	0
0	1	2.27	-1



Στους πίνακες δεδομένων, συγκεκριμένα για το χαρακτηριστικό 'correct_targets' μπορούμε να παρατηρήσουμε ότι για τις τιμές 0 και 1 είναι ξεκάθαρα πιο έντονο το κόκκινο χρώμα, καθώς και με βάση τα στοιχεία που αλλάξαμε από την έρευνα ανήκαν ήδη στην κατηγορία αυτήν. Για τιμές με 2 σωστά κυριαρχεί κατά λίγο περισσότερο το κίτρινο, ενώ για σωστές απαντήσεις 3 και 4 παρατηρούμε να υπάρχει διχασμός, κι αυτό καθώς οι τιμές αυτές κανονικά πρέπει να θεωρούνται κατά βάση κίτρινες και πράσινες, ωστόσο, υπάρχουν περιπτώσεις στις οποίες υπάρχουν 3-4 σωστά αλλά ταυτόχρονα 5-6 λάθος πατημένοι στόχοι. Επομένως από τα συγκεκριμένα παραδείγματα, παρατηρούμε να «κοκκινίζει» περισσότερο η συγκεκριμένη περιοχή. Για τα δεδομένα 'false alarms' υπάρχει μια πιο ισοσταθμισμένη εικόνα καθώς αυτά δεν επηρεάζονται τόσο έντονα από άλλα χαρακτηριστικά. Ταυτόχρονα ο χρόνος, παρατηρούμε ότι δεν ακολουθεί κάποιο συγκεκριμένο μοτίβο, κατά κύριο λόγο είναι αρκετά μοιρασμένο, καθώς δεν θέλουμε να λαμβάνεται τόσο σημαντικά ως παράγοντα.

4.2.3.5 Τελική Κατηγοριοποίηση Overall Classification

Η βάση για το τελευταίο και σημαντικότερο στοιχείο δεδομένων δεν έχει παρθεί από κάπου, είναι και πάλι τεχνητό με βάση τα χαρακτηριστικά των υπολοίπων βάσεων δεδομένων. Το συγκεκριμένο έχει ως έργο να συλλέγει τη τελική κατηγοριοποίηση για κάθε δοκιμασία που έχει περαστεί καθώς και την αυτοπεποίθηση που έχει παράξει κάθε φορά το μοντέλο μηχανικής μάθησης για την απόφαση του, και με βάση τα δεδομένα αυτά παράγει την τελική απόφαση για την κατηγορία στην οποία ανήκει ο μαθητής. Συμπεραίνει το τελικό αποτέλεσμα για τον μαθητή αναλαμβάνοντας διάφορους παράγοντες, για παράδειγμα εάν για τα 4 τεστ έχουν εξαχθεί τα αποτελέσματα : 1, 0, 1, -1 τότε το αποτέλεσμα θα πάει πλειοψηφικά και θα είναι 1,ενώ εάν τα αποτελέσματα είναι -1, 0, -1, 0 τότε κυριαρχεί το αποτέλεσμα με τα υψηλότερα ποσοστά αυτοπεποίθησης (confidence).

vocab_pred	vocab_con	vocab_con	vocab_cor	mem_pred	mem_conf	mem_conf	mem_conf	visual_pred	visual_con	visual_con	visual_con	gonogo_pred	gonogo_cc	gonogo_cc	gonogo_cc	overall_classification
1	0.32	0.13	0.55	-1	0.66	0.04	0.3	0	0.33	0.6	0.07	-1	0.89	0.04	0.07	-1
1	0.05	0.23	0.72	0	0.36	0.58	0.06	-1	0.64	0.23	0.13	-1	0.73	0.22	0.05	0
-1	0.55	0.08	0.37	1	0.12	0.14	0.74	0	0.05	0.61	0.34	-1	0.87	0.02	0.11	-1
1	0.01	0.10	0.89	-1	0.52	0.27	0.21	-1	0.50	0.01	0.49	1	0.13	0.01	0.86	1

Ο παραπάνω πίνακας περιέχει τιμές – παραδείγματα από τη βάση δεδομένων. Μπορούμε να δούμε ότι ακολουθεί τη λογική της πλειοψηφίας, και αλλά και των τιμών confidence όταν πρόκειται για ισοψηφία. Η αναπαράσταση των δεδομένων της συγκεκριμένης δομής όπως κάναμε και για τις προηγούμενες, δεν είναι ιδιαίτερα εφικτή, καθώς περιέχονται πολλαπλά χαρακτηριστικά τα οποία δεν μπορούν να συνδεθούν κατά κάποιο τρόπο με τις κατηγοριοποιήσεις μας (-1,0,1). Σε επόμενο υποκεφάλαιο, ωστόσο θα αναλυθεί αναλυτικά η επιτυχία του κατηγοριοποίησή με accuracy και confidence.

4.2.3.6 Χρήση Encoder για json αρχεία

Αξίζει να σημειωθεί ότι ο κώδικας περιέχει κλάση 'Encoder' σκοπός της οποίας είναι η μετάφραση ορισμένων στοιχείων JSON. Αυτό συμβαίνει διότι η βιβλιοθήκη μηχανικής μάθησης scikit-learn δεν επιστρέφει αριθμούς αναγνωρίσιμους από την Python, αλλά τους επιστρέφει σε μορφή τύπου NumPy. Αυτό συμβαίνει για τις προβλέψεις που υπολογίζει καθώς και της πιθανότητες τις κάθε απαντήσεις (confidence). Η βιβλιοθήκη της json δεν αναγνωρίζει τα παραγόμενα της και εάν πραγματοποιηθεί άμεση αποθήκευση χωρίς καμία επεξεργασία από πριν, προκαλείται διακοπή προγράμματος. Αυτή τη εργασία εκτελεί ο Encoder λοιπόν, ο οποίος μετατρέπει τα στοιχεία των αποτελεσμάτων και εάν εντοπίσει κάποιο της NumPy, τον μετατρέπει στον αντίστοιχο απλό τύπο της Python. Παρακάτω φαίνεται ενδεικτικά ο κώδικας που επιτελεί το έργο αυτό.

```
class NpEncoder(json.JSONEncoder):
    def default(self, obj):
        """Take the data we made with sklearn etc made to
        np and make it understandable fot json"""
        if isinstance(obj, np.integer): #prediction
            return int(obj)
        elif isinstance(obj, np.floating): #probabilities
            return float(obj)
        elif isinstance (obj, np.ndarray): #array probability
            return obj.tolist()
        elif isinstance(obj, datetime):
            return obj.isoformat()
        return super(NpEncoder, self).default(obj)
```

4.3 Έναρξη Εφαρμογής

Για αρχή η εφαρμογή είναι κατασκευασμένη γύρω από μια κεντρική κλάση "EducationalApp", η οποία λειτουργεί ως το κεντρικό σημείο που διαχειρίζεται τα πάντα. Μέσα στην κλάση αυτή, η λειτουργικότητα της βασίζεται σε πολλαπλές συναρτήσεις, όπου η καθεμιά εκτελεί συγκεκριμένο ρόλο στην εφαρμογή. Θα αναλυθούν παρακάτω λεπτομερώς οι πιο σημαντικές.

Η εκκίνησης του προγράμματος γίνεται στο τέλος του κώδικα, μέσα σε ένα μπλοκ το οποίο δημιουργεί το κύριο παράθυρο και μέσω της εντολής mainloop() η εφαρμογή αρχίζει να λειτουργεί σε αναμονή, επιτρέποντας της να μείνει ανοιχτή και να αντιδρά συνεχώς σε ενέργειες του χρήστη.

```
if __name__ == "__main__":
    root = tk.Tk()
    app = EducationalApp(root)
    root.mainloop()
```

4.3.1 Βιβλιοθήκες που χρησιμοποιούνται

Η εφαρμογή χρησιμοποιεί αρκετές βιβλιοθήκες πέρα από την βασική και απαραίτητη Tkinter. Καθεμιά επιτελεί τον δικό της ξεχωριστό ρόλο. Συγκεκριμένα:

- **Tkinter:** Είναι η βασική και χρησιμοποιείται για τη δημιουργία ολοκλήρου του γραφικού περιβάλλοντος της εφαρμογής. Περιέχει τα απαραίτητα βασικά στοιχεία όπως, πλαίσια (frames), καμβάδες (canvas) και τα κουμπιά.
- **PIL:** Η συγκεκριμένη αναλαμβάνει την επεξεργασία των εικόνων, χρησιμοποιείται για να τις ανοίγει και να αλλάζει το μέγεθος τους, για να μπορούν να εμφανιστούν σωστά στην εφαρμογή.
- **Json:** Χρησιμοποιείται για την ανάγνωση τον ερωτήσεων και την εγγραφή των αποτελεσμάτων στο αρχείο μαθητή.
- **Pandas:** Χρησιμοποιείται για να φορτώνει τα σύνολα δεδομένων εκπαίδευσης από τα αρχεία CSV σε πίνακες Dataframes.
- **NumPy:** Χρησιμοποιείται για τις πιο περίπλοκες πράξεις που συμβαίνουν στην εφαρμογή.
- **Sklearn:** Είναι η βιβλιοθήκη στην οποία βασίζεται η μηχανική μάθηση. Παρέχει έτοιμους τους αλγόριθμους κατηγοριοποίησης SVC και Random Forrest καθώς και εργαλεία για την προετοιμασία δεδομένων (Standard Scaller).
- **Threading:** Χρησιμοποιείται για την φόρτωση των βαριών αρχείων (GIF και ήχοι) στο παρασκήνιο, για να ξεκινάει αμέσως η εφαρμογή χωρίς να καθυστερεί λόγω της φόρτωσης των βαρειών αρχείων.
- **Math:** Χρησιμοποιείται για υπολογισμό παραγοντικού στην συνάρτηση Trotter για να γνωρίζει ακριβώς πόσες παραλλαγές ακολουθίας υπάρχουν.
- **Random:** Για την δημιουργία ακολουθιών στη δοκιμασία μνήμης με τυχαίο τρόπο.
- **Statistics:** Για τον υπολογισμό της διαμέσου των χρόνων απαντήσεων (median). Καθώς είναι καλύτερη τεχνική από το mean που μετρά τον μέσο όρο και είναι επιρρεπή σε ακραίες τιμές.
- **Time:** Χρησιμοποιείται για τη μέτρηση των χρόνων σε κάθε ερώτηση του κουίζ.
- **Datetime:** Καταγράφει την ακριβή ημερομηνία και ώρα που ολοκληρώθηκε το τεστ.
- **Re:** Χρησιμοποιείται για τον έλεγχο δεδομένων που εισαγωγή ο χρήστης στην φόρμα, για να ελέγξει ότι δεν υπάρχουν αριθμοί στο όνομα, ή χαρακτήρες στην ηλικία.
- **Itertools:** Χρησιμοποιείται για την συνεχή επανάληψη των frames των αρχείων GIF.

4.3.2 Προετοιμασία των Widget

Εκκίνηση της εφαρμογής, δίνει σήμα στην μέθοδο "__init__" να ξεκινήσει τη λειτουργία της. Στην συγκεκριμένη συνάρτηση εκτελούνται οι εντολές αρχικοποίησης και προετοιμασίας κυρίως. Δημιουργείται το κύριο παράθυρο (root) κι έπειτα ο κεντρικός καμβάς (cavnas) που φιλοξενεί όλα τα γραφικά στοιχεία. Ορίζονται οι γραμματοσειρές που θα χρησιμοποιηθούν σε όλη την εφαρμογή, και καλείται συνάρτηση που φορτώνει όλες τις

εικόνες φόντου για να αποθηκευτούν σε ένα λεξικό για άμεση πρόσβαση. Ξεκινάει να λειτουργεί το ξεχωριστό νήμα το οποίο καλεί την συνάρτηση που φορτώνει τα απαραίτητα αρχείου ήχου και εικόνας για κάθε παράδειγμα που θα υπάρξει αργότερα στο παρασκήνιο. Μέσα σε αυτήν επίσης ορίζονται τα στοιχεία εισόδου του εκπαιδευτικού στην εφαρμογή. Τέλος καλείται η συνάρτηση "create_widgets" που είναι υπεύθυνη για τη δημιουργία της αρχικής εικόνας της εφαρμογής. Αξίζει να σημειωθεί ότι η συνάρτηση καλείται μετά από λίγο χρόνο διότι, εάν κληθεί αμέσως, το πρόγραμμα παγώνει, καθώς δεν έχει ολοκληρωθεί ακόμη η δημιουργία των γραφικών. Ακολουθεί απόσπασμα της συνάρτησης:

```
self.root = root

self.questions = self.load_questions('questions.json')

self.preload_thread = threading.Thread(target=self.preload_example_gif_and_sound,
daemon=True)
self.preload_thread.start()

self.root.after(100, self.create_widgets)
self.root.bind('<Configure>', self.on_window_resize)
```

Συνέχεια έχει η μέθοδος "create_widgets" η οποία αναλαμβάνει την σύνθεση της αρχικής εικόνας που αντικρίζει ο χρήστης με την εκκίνηση της εφαρμογής. Η συγκεκριμένη συνάρτηση εκτελείται όταν η εφαρμογή τρέχει για πρώτη φορά, όταν ο έχει πατηθεί το κουμπί 'Back to main menu' από τα σημεία όπου ο μαθητής εισάγει τα δεδομένα του, το σημείο όπου ο εκπαιδευτικός πρόκειται να εισάγει τα στοιχεία για να εισέλθει ή εάν ο μαθητής τερματίσει την δοκιμασία και πατήσει 'Start Over'. Επειδή, λοιπόν, εκτελείται πολλαπλές φορές είναι χρήσιμο κάθε φορά να είμαστε σίγουροι ότι θα υπάρχει το σωστό φόντο (intro_background). Έπειτα η συνάρτηση δημιουργεί το κείμενο καλωσορίσματος του χρήστη, όπου δίνεται και οι βασικοί κανόνες και οδηγίες για το τεστ. Δημιουργούνται τα δύο κουμπιά επιλογής «Μαθητής» ή «Εκπαιδευτικός» τα οποία τοποθετούνται κάτω από το συγκεκριμένο κείμενο το καθένα σε δικό του παράθυρο widget. Ακολουθεί μικρό απόσπασμα της συνάρτησης.

```
def create_widgets(self):
        self.student_button=ttk.Button(self.canvas,
            text="Student",
            style='Flat.TButton',
            command=self.show_student_form
        )
        self.student_button_window = self.canvas.create_window(
            x_center - 150,
            y_center + 150,
            window=self.student_button
        )
        self.set_background('intro')
```

Σημαντική είναι η αναφορά της συνάρτησης 'on_window_resize', η οποία είναι υπεύθυνη για την αλλαγή μεγέθους του κάθε παραθύρου. Δηλαδή, εάν ο χρήστης προσπαθήσει να αλλάξει τις διαστάσεις του παραθύρου της εφαρμογής η συνάρτηση αυτή ενεργοποιείται και αυτόματα υπολογίζει εκ νέου τις συντεταγμένες με βάση τις νέες

διαστάσεις του παραθύρου, και τοποθετεί στη σωστή θέση το κάθε στοιχείο. Ακολουθεί ενδεικτικό απόσπασμα της συνάρτησης.

```
def on_window_resize(self, event=None):
      x_center, y_center = self.get_center()

      if hasattr(self, 'introduction_text'):
         self.canvas.coords(self.introduction_text, x_center, y_center - 100)

       if hasattr(self, 'student_button_window'):
         self.canvas.coords(self.student_button_window, x_center - 150,
y_center + 150)
      if hasattr(self, 'background_image'):
            self.canvas.coords(self.background_image, x_center, y_center)
```

4.4 Λειτουργία Μαθητή

Η αρχική οθόνη όπως είπαμε λοιπόν σε αρχικό στάδιο έχει τα πλήκτρα «Μαθητής» και «Εκπαιδευτικός». Παρακάτω θα αναλυθεί το στάδιο του μαθητή με τις δοκιμασίες στις οποίες υποβάλλεται κατά τη διάρκεια της εφαρμογής.

4.4.1 Εισαγωγή στοιχείων

Κατά το πάτημα λοιπόν του κουμπιού «μαθητής», αυτό είναι δεσμευμένο με την ενέργεια κλίσης της συνάρτησης «show_student_form». Επομένως ξεκινάει η λειτουργία της συγκεκριμένης συνάρτησης, της οποίας πρώτη ενέργεια είναι να καθαρίσει τον καμβά από τα στοιχεία της αρχικής οθόνης, δηλαδή το κείμενο καλωσορίσματος και τα κουμπιά του μαθητή και του εκπαιδευτικού αντίστοιχα. Να σημειωθεί ότι στο παρακάτω σχήμα τα στοιχεία έχουν διαφορετικές εντολές διαγραφής (delete, destroy), καθώς στοιχεία που τοποθετούνται στον καμβά διαγράφονται μέσω της εντολής .delete(), ενώ τα widgets διαγράφονται μέσω της εντολής .destroy().

```
        self.canvas.delete(self.introduction_text)
        self.student_button.destroy()
        self.teacher_button.destroy()
```

Αμέσως μετά, δημιουργεί και εμφανίζει ένα νέο σύνολο από γραφικά στοιχεία που αποτελούν την φόρμα εισαγωγής δεδομένων. Δημιουργεί Ετικέτες (Labels) για το όνομα, το επώνυμο, την ηλικία και το επίπεδο αγγλικών και αντίστοιχα πεδία εισαγωγής (Entries) όπου ο μαθητής θα πληκτρολογήσει τα στοιχεία του. Τέλος προσθέτει τα κουμπιά «Continue» με το οποίο ξεκινάει η διαδικασία του τεστ και «Go back» όπου επιστρέφει και ξανά καλείται η συνάρτηση create_widgets() για να δημιουργήσει την αρχική οθόνη. Για να τοποθετηθούν αυτά τα στοιχεία της φόρμας ομαλά στην εφαρμογή χρησιμοποιείται διαχειριστής διάταξης ‘grid()’. Η λογική που ακολουθείται είναι ότι στην στήλη 0 τοποθετούνται οι ετικέτες, δηλαδή τα κείμενα, στην στήλη 1 μπαίνουν τα αντίστοιχα πεδία εισαγωγής όπου ο χρήστης πληκτρολογεί. Για κάθε μια στήλη (column) χρησιμοποιείται και η αντίστοιχη γραμμή (row) για να υπάρχει συνοχή, και στη τελευταία γραμμή τοποθετούνται και τα κουμπιά που αναφερθήκαν προηγουμένως. Παρακάτω υπάρχει ενδεικτικό παράδειγμα για το στοιχείο του ονόματος.

```
name_label = tk.Label(self.student_form_frame, text="Name:",
font=self.answer_font, bg='#E18E6F', anchor='w')
        name_label.grid(row=0, column=0, sticky='w', pady=5, padx=5)

        name_entry = ttk.Entry(self.student_form_frame, width=30,
font=self.answer_font)
        name_entry.grid(row=0, column=1, sticky='e', pady=5, padx=5)
        name_entry.bind("<Return>", self.handle_enter_key_student_form)
        self.student_entries["name"] = name_entry
```

Σε κάθε εισαγωγή, έχει ενεργοποιηθεί λειτουργία που να δέχεται το 'Enter' από το πληκτρολόγιο και να το μεταφράζει ως «Κάνε focus στην ακριβώς από κάτω γραμμή», για πιο άμεση και γρήγορη εισαγωγή στοιχείων.

Η εισαγωγή «Επίπεδο Αγγλικών» δεν είναι απαραίτητη, επομένως έχει φτιαχτεί έτσι ώστε να εμφανίζεται μέσα σε αυτό ο όρος 'Optional' με γκρίζα γράμματα όταν ο χρήστης δεν έχει κάνει κλικ σε αυτό, και όταν κάνει κλικ να εξαφανίζεται και στην περίπτωση που πληκτρολογήσει κάτι, να παραμένει το κείμενο που έχει πληκτρολογήσει, ενώ αν το αφήσει κενό επιστρέφει το γκρίζο κείμενο 'Optional'. Από κάτω φαίνεται το κομμάτι του κώδικα που εκτελεί αυτή τη λειτουργία.

```
# English Level Label
        class_label = tk.Label(self.student_form_frame, text="English Level
(A1-C2):", font=self.answer_font, bg='#E18E6F', anchor='w')
        class_label.grid(row=3, column=0, sticky='w', pady=(0,10), padx=5)

        class_entry = ttk.Entry(self.student_form_frame, width=30,
font=self.answer_font)
        class_entry.insert(0, "Optional") # placeholder
        class_entry.config(foreground='gray')
        class_entry.bind("<FocusIn>", self.handle_focus_in)
        class_entry.bind("<FocusOut>", self.handle_focus_out)
        class_entry.bind("<Return>", self.handle_enter_key_student_form)
        class_entry.grid(row=3, column=1, sticky='e', pady=(5,10), padx=5)
        self.student_entries["class"] = class_entry
```

Μετά από την συνάρτηση αυτή καλείται η 'process_student_form' με το πάτημα του κουμπιού 'Continue' της φόρμας του μαθητή. Η συγκεκριμένη συνάρτηση είναι υπεύθυνη για να ελέγξει την ορθότητα των στοιχείων που εισήχθησαν προηγουμένως. Σε περίπτωση που αντικρίσει κάποια σφάλμα στις εισαγωγές όπως αριθμό στο όνομα ή γράμμα στην ηλικία, εμφανίζει μήνυμα λάθους στον χρήστη και σταματά τη διαδικασία, αναμένοντας τον χρήστη να πληκτρολογήσει τα σωστά στοιχεία. Οι απαιτήσεις που έχει είναι να μην υπάρχει αριθμός στο όνομα και στο επώνυμο, η ηλικία να είναι μεταξύ 6-14 και να μην περιέχει γράμματα και το επίπεδο τον αγγλικών εάν δεν είναι κενό να είναι A1,A2,B1..C2 και τα αντίστοιχα με πεζά γράμματα. Εφόσον όλα τα δεδομένα είναι έγκυρα, τα αποθηκεύει στο λεξικό 'self.student_info' και καταστρέφει τα στοιχεία της φόρμας προετοιμάζοντας την εφαρμογή για την έναρξη της δοκιμασίας, καλώντας την συνάρτηση start_quiz()

```
    if not name:
        messagebox.showerror("Error", "Name cannot be empty. Please enter your name.")
        return

    if re.search(r'\d', name):
        messagebox.showerror("Error", "Name cannot contain numbers.")
        return
```

Το πάτημα του κουμπιού «Go Back» από την άλλη, καλεί την συνάρτηση return_to_main_menu ή οποία καθαρίζει την οθόνη από την φόρμα και καλεί την create_widgets για την προετοιμασία της αρχικής οθόνης.

4.5 Έναρξη δοκιμασιών

Εφόσον έχει ολοκληρωθεί με επιτυχία η εισαγωγή δεδομένων του μαθητή σειρά έχει η εκτέλεση της μεθόδου 'start_quiz'. Η συγκεκριμένη συνάρτηση δεν εμφανίζει την πρώτη ερώτηση, αλλά λειτουργεί ως το κεντρικό σημείο προετοιμασίας για αυτό. Ο ρόλος της διασφαλίζει ότι με την έναρξη κάθε νέου κουίζ όλες οι μεταβλητές βρίσκονται στην αρχική τους κατάσταση, αποτρέποντας τη μεταφορά δεδομένων από προηγούμενες χρήσεις. Επομένως στο σημείο αυτό αρχικοποιούμε μεταβλητές που λειτουργούν ως μετρητές για κάθε μεμονωμένη δοκιμασία, αλλά και μεταβλητές και λίστες οι οποίες πρόκειται να κρατήσουν τις απαντήσεις για όλες τις ερωτήσεις.

Η συνάρτηση επιπλέον δημιουργεί κάποια μόνιμα γραφικά όπως το 'question_text_id', στο οποίο θα εμφανίζεται η κάθε ερώτηση για κάθε κουίζ αλλά και κάποιες φορές θα λειτουργεί ως τίτλος για τη κάθε δοκιμασία όταν δείχνουμε το παράδειγμα εκτέλεσης της. Επίσης δημιουργεί το πλαίσιο 'answer_grid' μέσα στο οποίο θα εμφανίζονται οι επιλογές απαντήσεων για κάθε ερώτηση. Εφόσον ολοκληρωθεί η λειτουργία της, αυτή καλεί την συνάρτηση 'show_next_screen()' η οποία παίζει σημαντικό ρόλο στη λειτουργία του κώδικα.

```
def show_next_screen(self):
        if self.current_question < len(self.questions):
            question = self.questions[self.current_question]

            if question['type'] in self.remaining_explanations:
                # We still need to explain this type
                self.show_explanation_screen(question['type'])
                self.remaining_explanations.remove(question['type'])
            else:
                self.build_question_ui(question)

        else:
            self.save_results_and_finish()
```

Η συνάρτηση αυτή λοιπόν εκτελείται για κάθε ερώτηση για όλες τις δοκιμασίες και αποφασίζει κάθε φορά εάν έχουν τελειώσει οι ερωτήσεις, τερματίζει αποθηκεύοντας τα αποτελέσματα του μαθητή. Διαφορετικά εάν υπάρχουν ακόμη ερωτήσεις που δεν έχουν εμφανιστεί και εάν αυτή που έχουμε είναι νέου τύπου τότε πρέπει να δείξουμε παράδειγμα εκτέλεσης για αυτό τον τύπο ερωτήσεων. Στην αρχή του κώδικα δημιουργείται μια λίστα 'self.remaining_explanations' που περιέχει τα ονόματα όλων των δοκιμασιών και η εξής συνάρτηση κάθε φορά που πρόκειται να εμφανιστεί μια νέα τύπου ερώτηση, διαγράφει τον τύπο της από την λίστα. Έτσι κάθε φορά η λίστα περιέχει μόνο όσες δοκιμασίες δεν έχουν εξηγηθεί έως τώρα. Εάν δεν είναι κάποιος νέος τύπος ερώτησης, τότε συνεχίζουμε κανονικά δείχνοντας απευθείας την ερώτηση χωρίς να προηγηθεί παράδειγμα για αυτήν. Η συνάρτηση καλείται όταν προχωράμε από ερώτηση σε ερώτηση του ίδιου τύπου και όταν έχει τελειώσει ένας τύπος δοκιμασίας και πρέπει να προχωρήσουμε στον επόμενο.

4.5.1 Λειτουργία Παραδειγμάτων

Τυπικά λοιπόν, για την πρώτη φορά που θα εκτελεστεί η συνάρτηση αυτή θα δίνεται νέος τύπος ερώτησης, συνήθως λεξιλογίου καθώς αυτές είναι οι πρώτες ερωτήσεις στο json αρχείο μας. Επομένως θα εκτελεστεί η συνάρτηση 'show_explanation_screen'. Η συνάρτηση αυτή δέχεται ως παράμετρο τον τύπο της ερώτησης (πχ. 'vocabulary') και με βάση τη πληροφορία αυτή επιλέγει το κατάλληλα αρχεία ήχου και GIF τα οποία πρόκειται να δείξει και αλλάζει τον τίτλο 'self.question_text_id' με το κατάλληλο τύπο της δοκιμασία που πρόκειται να δείξει. Αξίζει να αναφερθεί ότι αφού στη δοκιμασία μνήμης έχουμε 3 επίπεδα, υπάρχουν 3 διαφορετικά αρχεία ήχου και GIF για τα συγκεκριμένα παραδείγματα.

Έπειτα δίνει τα ορίσματα ήχου και εικόνας στη συνάρτηση 'play_example' η οποία χρησιμοποιεί τα ονόματα των αρχείων της δόθηκαν για να βρει τα αντίστοιχα δεδομένα στα λεξικά 'self.gif_frames' και 'self.sounds' όπου είχαν φορτωθεί από το νήμα προηγουμένως. Η ίδια συνάρτηση σταματά οποιονδήποτε ήχο παίζει από προηγούμενη εκτέλεση (καθώς ενδέχεται να καλείται ως συνάρτηση συνεχόμενα) και ξεκινά την παραγωγή από την αρχή του δεδομένου ήχου. Η κίνηση του GIF επιτυχάνεται μέσω βοηθητικής εσωτερικής συνάρτησης 'animate_gif' η οποία εμφανίζει το κάθε frame με μια μικρή χρονοκαθυστέρηση για να φαίνεται ως ενιαίο βίντεο GIF επαναληπτικά. Το κομμάτι κώδικα αυτό πρέπει να είναι συνάρτηση, καλώς καλείται μέσω της εντολής after() η οποία επιτρέπει την εμφάνιση των πλαισίων frames με μια χρονική σειρά. Κάτι που μια επαναληπτική ροή δεν θα επέτρεπε να συμβεί. Ταυτόχρονα εμφανίζονται τα κουμπιά 'Explain Again' και 'Okay I got it!', μέσω τον οποίων καλείται η συνάρτηση 'play_example' που ξανά δείχνει το παράδειγμα με ήχο και εικόνα ή καλείται η συνάρτηση 'stop_example_and_continue' η οποία παίρνει ως παράμετρο την ερώτηση στην οποία είμαστε για να συνεχίσει στην επόμενη.

```
def play_example(self, gif_file, sound_file):
        frames_to_play = self.gif_frames.get(gif_file) # Get list of frames

        def animate_gif(frame_index):
            image=frames_to_play[frame_index]

            self.gif_label.config(image=image)

            frame_index +=  1 #go to next frame index, until we are finished
            if frame_index>=len(frames_to_play):
                frame_index=0   #then reset and show gif again
            self.animation_id = self.root.after(90, animate_gif,frame_index)

        animate_gif(0)
         # Play the sound effect
        sound = self.sounds.get(sound_file)
        sound.play()
```

4.5.2 Λειτουργία της δοκιμασία Λεξιλογίου – Vocabulary

Μετά την εμφάνιση του πρώτου παραδείγματος, ο μαθητής συναντά πρώτη τη δοκιμασία του Λεξιλογίου. Όταν τερματίζει λοιπόν η προβολή του παραδείγματος, καθαρίζεται η οθόνη από τα widgets του παραδείγματος, και καλείται ευθύς η μέθοδος που είναι υπεύθυνη για την δημιουργία και την εμφάνιση της κάθε ερώτησης της εφαρμογής μας, η 'build_question_ui'. Η συνάρτηση λαμβάνει ως παράμετρο την ερώτηση και φτιάχνει το αντίστοιχο περιβάλλον για αυτήν. Ξεκινάει, ενημερώνοντας το επάνω μέρος της οθόνης με το 'question_text_id' με την ερώτηση της τρέχουσας ερώτησης.

```
question_text = question['question']
self.canvas.itemconfig(self.question_text_id, text=question_text)
```

Έπειτα η συνάρτηση καταλαβαίνει ότι πρόκειται για ερώτηση λεξιλογίου με την εντολή «self.vocabulary_start_time = time.time()», ξεκινά την καταγραφή του χρόνου, για να κρατήσει τον χρόνο απάντησης του μαθητή και ελέγχει εάν έχουν δημιουργηθεί τα απαραίτητα στοιχεία widgets. Εάν δεν έχουν δημιουργηθεί, την πρώτη φορά δηλαδή που θα αντικρίσουμε ερώτηση λεξιλογίου αυτά ξεκινούν να δημιουργούνται. Φτιάχνονται τα κουμπιά απαντήσεων και τα πλαίσια που τα περιβάλλουν μια φορά και αποθηκεύονται σε λίστες για να ξανά χρησιμοποιηθούν στις επόμενες ερωτήσεις. Παρακάτω φαίνεται το κομμάτι κώδικα που εκτελεί αυτή τη λειτουργία.

```
if not self.vocab_widgets_created:                # create widgets once
    row = 0
    col = 0
    button_index = 0
    for option in options:      # make selection frame for button (not visible yet)
        selection_frame = tk.Frame(self.answer_grid, bg='#E18E6F',
                                   highlightthickness=4, highlightbackground='#E18E6F')
        btn = ttk.Button(       # make buttons inside of se selection frame
              selection_frame, text=option, style='Flat.TButton',
              command=lambda o=option, idx=button_index: self.select_option(o, idx))
        btn.pack(padx=2, pady=2)
                              # have selection frames be a grid with the buttons inside
        selection_frame.grid(row=row, column=col, padx=50, pady=20)

        self.vocab_buttons.append(btn)     # Save button and frame for later use
        self.vocab_selection_frames.append(selection_frame)

        col += 1
        if col > 1:
           col =
           row += 1
        button_index += 1
    self.vocab_widgets_created = True
```

Ο σκοπός του selection frame είναι να λειτουργεί ως ένα οπτικό πλαίσιο γύρω από το κάθε κουμπί απάντησης. Όταν ο μαθητής επιλέγει μια απάντηση, αλλάζει το χρώμα του περιγράμματος (highlightbackground) αυτού του πλαισίου για να δείχνει πιο έντονα η επιλογή του. Ενώ το κομμάτι γραμμών col και row είναι ο μηχανισμός που τακτοποιεί αυτόματα τις επιλογές απαντήσεων σε πινακάκι 2x2. Φροντίζει να τοποθετήσει το πρώτο κουμπί στη θέση (0,0), δηλαδή γραμμή 0, στήλη 0 και το δεύτερο στη θέση (0,1). Μόλις η δεύτερη στήλη γεμίσει αυξάνεται η γραμμή και ξανά μηδενίζεται η στήλη και τα επόμενα κουμπιά μπαίνουν στις θέσεις (1,0) και (1,1).

Για κάθε άλλη ερώτηση λεξιλογίου το παραπάνω μπλοκ εντολών δεν εκτελείται πλέον, αλλά εκτελείται το παρακάτω, που επαναχρησιμοποιεί τα γραφικά στοιχεία της δοκιμασίας. Εν ολίγοις, αντί να καταστρέφει τα παλιά κουμπιά και να δημιουργεί νέα για κάθε ερώτηση, ο κώδικας ανατρέχει στις λίστες όπου είχαν αποθηκευτεί τα αρχικά κουμπιά και τα πλαίσιά τους και ενημερώνει τα κείμενα του κάθε κουμπιού με την νέα απάντηση και επαναφέρει το χρώμα του πλαισίου selection_frame στην αρχική του κατάσταση, για να μην φαίνεται η τελευταία επιλογή του χρήστη από την προηγουμένη ερώτηση.

```
else:               # keep resusing existing widgets
    button_index=0
    for option in options:
        btn = self.vocab_buttons[button_index]
        frame = self.vocab_selection_frames[button_index]

        # Update the button with the new text and command
        btn.config(text=option, command=lambda o=option,
                   idx=button_index:self.select_option(o, idx))

         frame.config(highlightbackground='#E18E6F')   # Reset the border color

        button_index += 1
```

Στην ίδια συνάρτηση, (build_question_ui) φροντίζουμε να δημιουργήσουμε και ένα «Continue» button το οποίο όπως και τα προηγούμενα στοιχεία εξασφαλίζουμε ότι δημιουργείται μια φορά και έπειτα επαναχρησιμοποιείται. Στην αρχή το κουμπί έχει τη λειτουργία του απενεργοποιημένου, καθώς περιμένει από τον μαθητή να πατήσει πάνω σε κουμπί απάντησης, κι έπειτα μπορεί να ενεργοποιηθεί κι αυτό. Το κουμπί αυτό καλεί την συνάρτηση 'submit_selected_response'η οποία διαχειρίζεται την απάντηση του μαθητή.

Η μορφή των ερωτήσεων λεξιλογίου έχει το εξής χαρακτηριστικό, ότι το μόνο με το οποίο μπορεί για αρχή να αλληλοεπιδράσει ο χρήστης είναι τα κουμπιά απαντήσεις. Κάθε φορά που πατιέται αυτό, ενεργοποιείται η συνάρτηση 'select_option'. Αυτή δεν καλείται αυτόματα, αλλά μέσω της 'command' του κουμπιού της κάθε απάντησης, με τη βοήθεια μιας lambda συνάρτησης. Η συνάρτηση αυτή, δεν καλεί την 'select_option' αμέσως, αλλά δημιουργεί μια συνάρτηση μιας χρήσης για κάθε συγκεκριμένο κουμπί. Το σημαντικό είναι ότι σε κάθε κλήση της, θα δώσει ως παράμετρο το εκάστοτε κουμπί που πατήθηκε. Δηλαδή για κάθε κουμπί που πατάει ο μαθητής εκτελείται ξεχωριστή lambda και περνιέται διαφορετική παράμετρος στην συνάρτηση 'select_option'.

Όταν έρθει η σειρά της 'select_option' αυτή είναι υπεύθυνη για τη διαχείριση της επιλογής του χρήστη. Αποθηκεύει την απάντηση του χρήστη για μεταγενέστερη επεξεργασία, φροντίζει να χρωματίσει με διαφορετικό χρώμα την επιλεγμένη απάντηση του χρήστη, και εάν ο χρήστης πατάει πολλαπλές απαντήσεις έχει φροντίσει σε κάθε λειτουργία της να επαναφέρει το χρώμα των πλαισίων στο αρχικό τους χρώμα. Τέλος, ενεργοποιεί τη λειτουργία του 'Continue' για να μπορέσει ο μαθητής να προχωρήσει στην επόμενη ερώτηση ή επόμενο στάδιο λειτουργίας της εφαρμογής.

Στο σημείο αυτό, πλέον το κουμπί 'Continue' είναι ενεργοποιημένο και ο μαθητής μπορεί να το πατήσει εφόσον έχει δια σιγουρέψει την απάντηση του. Μόλις αυτό γίνει καλείται η συνάρτηση υπεύθυνη για το παρόν κουμπί, η 'submit_selected_response', η οποία είναι υπεύθυνη για την καταγραφή της τελικής απάντησης που δόθηκε. Πιο συγκεκριμένα μέσα στη συνάρτηση σταματά ο χρόνος καταγραφής και αποθηκεύεται για την παρούσα ερώτηση, έπειτα συγκρίνεται η σωστή απάντηση με αυτήν που δόθηκε από τον μαθητή και αναλόγως ενημερώνει την βαθμολογία για τη παρούσα δοκιμασία. Τα στοιχεία

αυτά καταγράφονται και αποθηκεύονται στο λεξικό ‘self.answers’ για μεταγενέστερη χρήση.

Αξίζει να σημειωθεί ότι στην ίδια συνάρτηση γίνεται έλεγχος τερματισμού, όχι δηλαδή μόνο εάν έχουν τελειώσει οι ερωτήσεις λεξιλογίου, αλλά εάν ισχύει ότι ο μαθητής έδωσε 9 σωστές απαντήσεις από τις τελευταίες 10 που ερωτήσεις που απάντησε. Στην περίπτωση αυτή, η εφαρμογή σταματάει τη δοκιμασία του λεξιλογίου, καταστρέφει τα στοιχεία που χρησιμοποιούσαμε επανειλημμένα για τις ερωτήσεις λεξιλογίου και καλεί το μοντέλο μηχανικής μάθησης για αξιολόγηση του μαθητή. Σε περίπτωση που δεν έχει τερματίσει, η διαδικασία συνεχίζεται κανονικά με μετάβαση στην επόμενη ερώτηση και επαναφορά στην συνάρτηση show_next_screen η οποία ελέγχει το τι πρέπει να συμβεί έπειτα.

```
def submit_selected_response(self):
    # to make sure continue is not pressed multiple times
    if  self.is_processing == False:
        self.is_processing = True
        answer_entry = {
            "answer": self.selected_option,
            "correct": is_correct,
             "type": question['type'],
             "time": question_time
        }
        self.answers.append(answer_entry)
        if (len(self.vocabulary_consecutive_attempts) >= 10 and
           sum(self.vocabulary_consecutive_attempts)>=9)or
           (self.vocabulary_total_questions == 50):
                if self.vocabulary_total_questions<=25:
                     self.vocabulary_stopped_early=1
                self.classify_vocabulary_performance()
```

Μέσα στον κώδικα της ‘submit_selected_response’ χρησιμοποιείται η μεταβλητή self.is_processing η οποία αποτρέπει σφάλματα που θα μπορούσαν να προκληθούν εάν ο μαθητής πατούσε το κουμπί ‘Continue’ πολλαπλές φορές συνεχόμενα. Η μεταβλητή για αρχή είναι Ψευδής και όσο είναι Ψευδής, μπαίνει στο κομμάτι κώδικα που επεξεργάζεται την απάντηση. Με το που εισαχθεί σε αυτό, γίνεται Αληθής προκειμένου να αποτρέψει τον κώδικα να εκτελέσει το κομμάτι κώδικα δυο ή παραπάνω φορές για πολλαπλά πατήματα στο κουμπί. Η λειτουργία των Κατηγοριοποιητών θα αναλυθεί εκτενέστερα σε παρακάτω υποκεφάλαιο της ενότητας.

4.5.3 Λειτουργία της δοκιμασία Μνήμης – Memory

Όταν τερματίσει τη λειτουργία της η δοκιμασία λεξιλογίου, σειρά έχει η δοκιμασία της Οπτικής Εργαζόμενης Μνήμης, μέσω της ανάκλησης αριθμητικών ακολουθιών με ψηφία από 3 έως και 5. Προτού ξεκινήσει η δοκιμασία, η εφαρμογή δείχνει το σχετικό παράδειγμα για την υλοποίηση της δοκιμασίας με ήχο και εικόνα, κι έπειτα ξεκινάει η δοκιμασία.

4.5.3.1 Δημιουργία βασικών στοιχείων Widget

Η προετοιμασία της δοκιμασία απαιτεί την απαραίτητη δημιουργία των στοιχείων widget αρχικά, μέσω της συνάρτησης build_question_ui. Όπως και στην δοκιμασία του λεξιλογίου, τα στοιχεία δημιουργούνται μια φορά, κι έπειτα επαναχρησιμοποιούνται. Επομένως, η συνάρτηση ελέγχει εάν έχουν δημιουργηθεί τα στοιχεία, και εάν όχι ξεκινάει τη δημιουργία τους. Το κύριο πλαίσιο memory_main_frame είναι το πρώτο στοιχείο που δημιουργείται, το οποίο είναι υπεύθυνο για τη φιλοξενία όλων των υπολοίπων γραφικών στοιχείων της δοκιμασίας. Μέσα σε αυτό, δημιουργείται η ετικέτα που περιλαμβάνει το μήνυμα «Get ready..» που σηματοδοτεί την έναρξη εμφάνισης ψηφίων και ένα ακόμη πλαίσιο το οποίο συγκροτεί όλα τα στοιχεία υπεύθυνα για την εισαγωγή της ακολουθίας που μόλις προβλήθηκε. Δηλαδή περιλαμβάνει την ετικέτα απάντησης που δέχεται τους αριθμούς που πληκτρολογεί ο χρήστης, το κουμπί 'Delete' ακριβώς δίπλα σε αυτό που επιτρέπει να σβηστούν ψηφία που έχουν πληκτρολογηθεί. Κάτω από αυτά, δημιουργείται ένα πληκτρολόγιο κουμπιών που περιέχει μόνο ψηφία. Συγκεκριμένα ορίζουμε μια λίστα που λειτουργεί ως συντεταγμένες και κάθε στοιχείο αντιστοιχεί σε συγκεκριμένη θέση στο πινακάκι που δημιουργούμε μέσω μια επανάληψης και την εντάσσουμε στο πλαίσιο με τη χρήση πίνακα grid().

Στο τέλος, δημιουργείται το κουμπί 'Continue', το οποίο χρησιμοποιείται μόνο μόνο στη συγκεκριμένη συνάρτηση καθώς δεν καλεί συνάρτηση ελέγχου απάντησης από κουμπιά, όπως οι δοκιμασίες Λεξιλογίου και Οπτικών Μοτίβων, αλλά καλεί ειδική συνάρτηση 'check_memory_response' που ελέγχει την είσοδο που δόθηκε από τον χρήστη. Παρακαλώ φαίνεται η υλοποίηση του numpad. Η λίστα button_positions περιέχει τις εξής τιμές
button positions = [(3, 1), (0, 0), (0, 1), (0, 2), (1, 0), (1, 1),(1, 2), (2, 0), (2, 1), (2, 2)]

```
for i in range(10):
    # Get the position for the current number from our map
    row, col = button_positions[i]
    # Create the button
    btn = ttk.Button(
        num_pad_frame,
        text=str(i),
        style='Small.TButton',
        command=lambda number=i: self.memory_on_number_click(number)
    )
    # Place the button at its defined position
    btn.grid(row=row, column=col, padx=2, pady=2)
```

4.5.3.2 Εμφάνιση Ακολουθίας Memory

Αφότου δημιουργηθούν μια φορά, μετά η καλείται η συνάρτηση 'run_memory_trial' και η πορεία του προγράμματος δεν ξανά καλεί την build_ui για τον συγκεκριμένο τύπο ερωτήσεων. Η συνάρτηση αυτή λοιπόν, καλείται κάθε φορά που πρόκειται να ξεκινήσει ένας νέος γύρος της δοκιμασίας μνήμης. Ο ρόλος της είναι να προετοιμάσει το περιβάλλον για τη φάση της απομνημόνευσης, να δημιουργήσει τη νέα ακολουθία αριθμών και να ξεκινήσει την προβολή της.

Πιο συγκεκριμένα, η συνάρτηση εξασφαλίζει ότι έχει κρύψει την περιοχή εισαγωγής, χρήσιμο για όταν έχει κληθεί μετά από πληκτρολόγηση προηγούμενης ακολουθίας. τη συνέχεια, η συνάρτηση δημιουργεί τη νέα, τυχαία ακολουθία αριθμών η οποία είναι προσαρμόσιμη, δηλαδή αναλόγως την δυσκολία στην οποία βρισκόμαστε (self.memory_level) δημιουργεί ακολουθία με τρία, τέσσερα ή πέντε ψηφία. Ο τελικός αριθμός μετατρέπεται σε μια λίστα από χαρακτήρες για να μπορεί να είναι έτοιμος για προβολή.

```
# Generate the Sequence based on mem level
        if self.memory_level == 1:
            random_number = random.randint(100, 999)
        elif self.memory_level == 2:
            random_number = random.randint(1000, 9999)
        else:  # This will handle Level 3 and any other case
            random_number = random.randint(10000, 99999)

self.current_memory_sequence = list(str(random_number))
```

Αφότου έχει γίνει αυτό, η συνάρτηση 'run_memory_trial' προετοιμάζει τον χρήστη για τη προβολή, εμφανίζει για αρχή το μήνυμα «Get ready..» κι έπειτα χρησιμοποίει ειδική εντολή για να after() για να προγραμματίσει την εκτέλεση της συνάρτησης που θα προβάλει τους αριθμούς της λίστας με παύση 2 δευτερολέπτων ενδιάμεσα. Μόλις ολοκληρωθεί αυτό, αποκαλύπτεται το κρυμμένο πληκτρολόγιο το οποίο ζητά από τον χρήστη να πληκτρολογήσει την απάντηση του.

```
    def run_step(given_index):
        """If it ends show numpad for user"""
        # Is the given index the last one?
        if given_index >= len(steps):
            #hide question text
            self.canvas.itemconfig(self.question_text_id, text="")
            # Show text for sequence
            self.memory_get_ready_text.config(text="Type the sequence:",font=self.question_font)

            #show numpad, start time, and set keyboard to accept answers
            self.memory_entry_widgets_frame.pack(fill=tk.BOTH, expand=True)
            self.sequence_start_time = time.time()
            self.root.bind('<Key>', self.memory_on_key_press)
            self.root.focus_set()
        # We have still numbers or blanks pace to show
        else:
            #variable for content and its time
            content_to_show, duration = steps[given_index]

            #Show number or blank " "
            self.memory_get_ready_text.config(text=content_to_show, font=self.memory_display_font)

            # run methodfor next index
            self.root.after(duration, run_step, given_index + 1)
```

Η από πάνω συνάρτηση, αναλαμβάνει την οπτική προβολή της ακολουθίας των αριθμών, λειτουργώντας σαν ένας προβολέας. Δημιουργεί μια λίστα αναπαραγωγής, η οποία περιλαμβάνει την εμφάνιση κάθε αριθμού για ένα δευτερόλεπτο ακολουθούμενη από μια παύση με κενή οθόνη. Μέσω μιας εσωτερικής, χρονομετρημένης συνάρτησης που καλεί αναδρομικά τον εαυτό της με τη βοήθεια της root.after(), αυτή η λίστα εκτελείται βήμα-βήμα, δημιουργώντας την ψευδαίσθηση της κίνησης χωρίς να "παγώνει" το περιβάλλον. Μόλις ολοκληρωθεί η προβολή, η συνάρτηση μεταβαίνει στο στάδιο της απάντησης και εμφανίζει το πληκτρολόγιο π0ου είχαμε κρύψει προηγουμένως,, ξεκινά το χρονόμετρο για τον χρήστη και ενεργοποιεί την αναγνώρισή πατημάτων από το πληκτρολόγιο.

4.5.3.3 Διαχείριση απάντησης Memory

Το επόμενο σημαντικό βήμα είναι η διαχείριση της απάντησης του χρήστη. Αφού η ακολουθία προβληθεί και το πληκτρολόγιο εμφανιστεί, η εφαρμογή μπαίνει σε κατάσταση αναμονής, έτοιμη να καταγράψει την απάντηση. Ο χρήστης έχει δύο τρόπους για να εισάγει

την απάντησή του είτε χρησιμοποιώντας το ποντίκι στο ενσωματωμένο πληκτρολόγιο ή πληκτρολογώντας απευθείας από το φυσικό του πληκτρολόγιο. Η εφαρμογή είναι σχεδιασμένη να διαχειρίζεται και τις δύο μεθόδους ταυτόχρονα.

Το ενσωματωμένο πληκτρολόγιο οθόνης, με κάθε πάτημα ενός κουμπιού καλεί τη συνάρτηση 'memory_on_number_click', η οποία προσθέτει το αντίστοιχο ψηφίο στην μεταβλητή 'self.memory_user_input' η οποία ανανεώνει την ετικέτα απαντήσεις για να βλέπει ο χρήστης αυτό που πληκτρολογεί.

```
def memory_on_number_click(self, num):
    """Update text to what user inputed"""
    self.memory_user_input += str(num) #update variable
    self.memory_users_input_display.config(text=self.memory_user_input) #update visible text
```

Με το πάτημα 'Delete' καλείται η συνάρτηση 'memory_on_delete_click' η οποία αφαιρεί τον τελευταίο χαρακτήρα από τη μεταβλητή της απάντησης, και κάνει την αλλαγή εμφανή για τον χρήστη.

```
def memory_on_delete_click(self):
    """Delete from variable last numbers everytime, and update display"""
    self.memory_user_input = self.memory_user_input[:-1] #cuts off last character
    self.memory_users_input_display.config(text=self.memory_user_input)
```

Μετά την προβολή της ακολουθίας ενεργοποιείται η εντολή bind η οποία ενεργοποιεί την λειτουργία του πατήματος του κάθε πλήκτρου. Είναι σημαντικό να γίνει μετα την προβολή της ακολουθίας, καθώς διαφορετικά μπορεί ο μαθητής να πληκτρολογεί την ακολουθία ενώ την βλέπει και να εμφανιστεί 'έτοιμη' μετά. Η συνάρτηση 'memory_on_key_press' ακούει το πληκτρολόγιο λοιπόν, και για κάθε μεμονωμένο πλήκτρο που μας ενδιαφέρει 'backspace', 'Enter' ή ψηφίο, καλεί αντίστοιχες συναρτήσεις για να τις δράσουν αναλόγως. Μόλις ο χρήστης ολοκληρώσει την εισαγωγή της ακολουθίας και πατήσει 'Enter' ή το κουμπί 'Continue' στην οθόνη, το στάδιο της καταγραφής τερματίζει και ξεκινά το επίπεδο αξιολόγησης.

```
def memory_on_key_press(self, event):
    """Handle Keyboard pressing."""
    if event.char.isdigit(): #number pressed
        self.memory_on_number_click(event.char)
    elif event.keysym == 'BackSpace': #backspace key pressed
        self.memory_on_delete_click()
    elif event.keysym == 'Return': #enter pressed
        self.check_memory_response()
```

Η αξιολόγηση της απάντησης πραγματοποιείται μέσω της συνάρτησης 'check_memory_response' η οποία ελέγχει εάν η απάντηση είναι σωστή και την καταγράφει

```
# 1) Advance to the next level or finish the test successfully?
if self.memory_consecutive_correct >= 3:
    if self.memory_level < 3:
        # Advance to the next level
        self.memory_level += 1
        self.memory_trials_in_level = 0
        self.memory_consecutive_correct = 0
        self.memory_consecutive_wrong = 0
        self.memory_main_frame.pack_forget()
        self.show_explanation_screen('memory_check')
    else:
        # Finished all levels successfully
        self.end_memory_test()
```

μαζί με τον χρόνο της, και ελέγχει εάν χρειάζεται να τερματίσουμε την δοκιμασία, να πάμε σε επόμενο επίπεδο ή να μείνουμε στο ίδιο επίπεδο και να δείξουμε όμοια ακολουθία. Μέσα στην ίδια συνάρτηση, καλείται και μια ακόμη σημαντική, η 'analyze_sequence_patterns' η οποία κάνει μια πιο βαθιά ανάλυση πιθανών λαθών, θα επεκταθούμε σε αυτήν παρακάτω. Οι ελέγχου που κάνει η συνάρτηση check_memory_response' φαίνονται στην εικόνα παρακάτω.

Εάν ο χρήστης έχει δώσει 3 συνεχόμενες σωστές απαντήσεις πραγματοποιείται προαγωγή σε επόμενο επίπεδο, και αναλόγως αρχικοποιούνται οι μετρητές για να προετοιμαστούν για το νέο επίπεδο, μόνο όμως εάν δεν βρισκόμαστε στο επίπεδο 3 που είναι και το τελευταίο. Στην περίπτωση αυτήν τερματίζει η δοκιμασία.

```
# 2) Stop the test due to errors?
elif self.memory_consecutive_wrong >= 3 or self.memory_trials_in_level >= 5:
    self.end_memory_test()
```

Στην περίπτωση που ο χρήστης έχει κάνει 3 συνεχόμενά λάθη, ή αν έχει συμπληρώσει 5 προσπάθειες στο τρέχον επίπεδο, τότε η εφαρμογή κρίνει ότι πρέπει να τερματίσει η δοκιμασία μνήμης.

```
# 3) Continue on the same level
else:
    self.root.unbind('<Key>')
    # We need another trial at the current level
    self.run_memory_trial()
```

Αν δεν ισχύει κανένας από τους παραπάνω κανόνες, τότε η συνάρτηση συνεχίζει να δείχνει παράδειγμα του ίδιου επιπέδου, φροντίζοντας να ξανά «κλειδώσει» τη λειτουργία του πληκτρολογίου πριν δείξει την επόμενη ακολουθία. Στην περίπτωση που τερματίσει η δοκιμασία μνήμης, καλείται η συνάρτηση υπεύθυνη για το μοντέλο κατηγοριοποίησης της.

Σε επόμενο στάδιο, λοιπόν, η συνάρτηση ''analyze_sequence_patterns' κάνει μια πιο βαθιά, ποιοτική ανάλυση, ειδικά στην περίπτωση των λανθασμένων απαντήσεων. Σκοπός της είναι να αναλύσει τις απαντήσεις του μαθητή και να δώσει απαντήσεις σχετικά με το «πόσο» λάθος ήταν.

```
# Find closest permutation match
is_exact_permutation = False
trotter_distance = -1  # Will keep the index difference IF FOUND
permutation_path = []  # List of path until index
for i in range(len(all_permutations)):
    if self.memory_user_input==all_permutations[0]:
        #No permutation, we have the right answer, SKIP
        break
    if all_permutations[i] == self.memory_user_input and i>0:
        #found exact permutation, keep index, path and stop
        is_exact_permutation = True
        trotter_distance = i
        #keep list until last index (+1)
        permutation_path = all_permutations[:(i + 1)]
        break
```

Σε πρώτο επίπεδο λοιπόν καταμετρά το πλήθος των σωστών ψηφίων που θυμήθηκε ο μαθητής ανεξάρτητα από τη σειρά. Έπειτα αναλύει ο ποσοστό των ψηφίων που βρίσκονται στην απολύτως σωστή τους θέση. Ελέγχει ακόμη, εάν η απάντηση του χρήστη είναι πλήρης αντιστροφή της σωστής ακολουθίας. Τέλος μέσω του αλγορίθμου Johnson-Trotter παράγει όλες τις πιθανές μεταθέσεις της σωστής ακολουθίας και ελέγχει αν η λανθασμένη απάντηση

του χρήστη ταιριάζει με κάποια από αυτές τις μεταθέσεις. Ένα τέτοιο λάθος υποδηλώνει ότι ο μαθητής θυμήθηκε όλα τα ψηφία, αλλά μπέρδεψε τη σειρά τους. Τελικά, η συνάρτηση επιστρέφει ένα λεξικό με όλες αυτές τις πληροφορίες, το οποίο αποθηκεύεται στα τελικά αποτελέσματα. Παραπάτω φαίνεται ο κώδικας που υλοποιεί τις διαδικασίες αυτές.

Το σημείο αυτό εκτελεί τη λειτουργία του Johnson Trotter, από προηγούμενη συνάρτηση έχει ανακαλέσει όλες τις δυνατές μεταθέσεις για την παρούσα ακολουθία και τις έχει αποθηκεύσει στη λίστα all_permutations. Θα αναλυθεί παρακάτω η συγκεκριμένη συνάετηση. Ελέγχει λοιπόν αν η απάντηση του χρήστη είναι μια ακριβής μετάθεση, δηλαδή αν ανήκει στη λίστα με όλες τις μεταθέσεις και εάν την εντοπίσει καταγράφει την «απόσταση» της μετάθεσης από την αρχική σωστή απάντηση. Καταγράφει ακόμη και τη διαδρομή των μεταθέσεων που οδήγησαν σε αυτήν.

```
# Calculate digit recall (how many correct digits, regardless of position)
count = 0
for digit in correct_sequence:
    if digit in self.memory_user_input:
        count += 1
        #print("FOUND: ",digit)

    digit_recall = count / len(correct_sequence)
    digit_recall = round (digit_recall,2)
```

Αυτός ο κώδικας υπολογίζει την ανάκληση ψηφίων, δηλαδή το ποσοστό των σωστών ψηφίων που ο χρήστης κατάφερε να θυμηθεί. Για να το καταφέρει, περνάει έναν-έναν τους χαρακτήρες της σωστής ακολουθίας και για κάθε έναν, ελέγχει αν αυτός υπάρχει οπουδήποτε μέσα στην απάντηση του χρήστη. Στο τέλος, διαιρεί τον αριθμό των ψηφίων που βρέθηκαν με το συνολικό μήκος της σωστής ακολουθίας για να υπολογίσει το τελικό ποσοστό ανάκλησης.

```
# Calculate position accuracy (how many digits in correct positions)
position_matches = 0
i = 0
while i < len(correct_sequence) and i < len(self.memory_user_input):
    # If the characters at the current position match...
    if correct_sequence[i] == self.memory_user_input[i]:
        position_matches += 1
    i += 1

position_accuracy = position_matches / len(correct_sequence)
position_accuracy = round(digit_recall,2)
```

Αυτός ο κώδικας υπολογίζει την "ακρίβεια θέσης" χρησιμοποιώντας επαναληπτικό βρόχο. Συγκεκριμένα, συγκρίνει την απάντηση του χρήστη με τη σωστή ακολουθία, χαρακτήρα προς χαρακτήρα, στην ίδια ακριβώς θέση και μετράει πόσα ψηφία είναι ταυτόχρονα σωστά και στη σωστή θέση. στο τέλος υπολογίζει το αντίστοιχο ποσοστό ακρίβειας.

Συνάρτηση Johnson-Trotter:

Σε προηγούμενο κεφάλαιο (Τεχνολογικό Υπόβαθρο), αναλύεται η λειτουργία του συγκεκριμένου αλγορίθμου στην θεωρία και δείξαμε ένα παράδειγμα υλοποίησης. Στην πράξη όμως, ο αλγόριθμος αυτός ακόμη λειτουργεί μόνο για ακολουθίες του τύπου '123..n'. Στην εφαρμογή μας, δεν θέλουμε οι ακολουθίες που δίνονται να ακολουθούν αυτό το πρότυπο αλλά να έιναι πιο τυχαία δομημένες. Επομένως, χρειάστηκε να γίνει αναπροσαρμογή του βασικού αλγορίθμου Johnson-Trotter για να είναι ικανό να λειτουργήσει για ακολουθίες που δεν ήταν σε αύξουσα σειρά.

```
# Make sequence into "314" -> ['3','1','4']
original_sequence_list = list(sequence)
# sort seuqnece into ['1','3','4']
sorted_sequence_list = sorted(original_sequence_list)
# lenth of sequence
n = len(original_sequence_list)

id_sequence = []
mapping = {}
for i in range(n):
    #make id list ['1,'2','3']
    id_sequence.append(i+1)

    #make dictionary for id_mapping:
    # "1": "1" , "2": "3" , "3" : "4" from sorted list
    mapping[i+1] = sorted_sequence_list[i]

#Keep starting direcctions
directions = [RIGHT_TO_LEFT for i in range(n)]
all_permutations = []

#Input sorted_sequence first as ['1','3','4'] -> '134'
first_permutation = "".join(sorted_sequence_list)
all_permutations.append(first_permutation)
```

```
#For every possible permutation:
for i in range(1, math.factorial(n)):
    # Find next permutation based on id [1,2..n]
    id_sequence = find_one_permutation(id_sequence, directions)
    # decode given permutation
    current_perm = [mapping[i] for i in id_sequence]
    # add it to list of permutations
    all_permutations.append("".join(current_perm))

#remove dupliates without messing up the order
unique_permutations = list(dict.fromkeys(all_permutations))

#Find the index of our original sequence and split the list in two
start_index = unique_permutations.index(sequence)
part1 = unique_permutations[start_index:]
part2 = unique_permutations[:start_index]

#put them together as a circle, starting from the original sequence
unique_permutations = part1 + part2

return unique_permutations
```

Για την συγκεκριμένη εφαρμογή λοιπόν αντλήσαμε τον βασικό κώδικα για τον αλγόριθμο Johnson-Trotter από ιστοσελίδα διαδικτύου και το αλλάξαμε. Ας θυμηθούμε για αρχή πως λειτουργεί ο αλγόριθμος. Βασική προϋπόθεση για να λειτουργήσει ο αλγόριθμος ορθά έιναι τα ψηφία να βρίσκονται σε αύξουσα σειρά, δηλαδή '136' και ταυτόχρονα να υπάρχει διαφορά μιας μονάδας από κάθε ψηφίου δηλαδή να είναι συνολικά '123' η ακολουθία. Για να λειτουργήσει κάτι τέτοιο για μια τυχαία ακολουθία λοιπόν, για παράδειγμα '195', αυτό που κάνει ο δικός μας αλγόριθμος είναι το εξής. Θα πρέπει για αρχή να μετατρέψει τη τρέχουσα ακολουθία σε αύξουσα, δηλαδή '195' => '159' και δημιουργεί ένα λεξικό μεταφραστή (mapping) το οποίο μας δημιουργεί την ιδανική ακολουθία, δηλαδή '159' => '123' και επάνω σε αυτή την ακολουθία εκτελείται κανονικά η έως τώρα γνωστή διαδικασία του Johnson-Trotter. Για κάθε μετάθεση που παράγεται η συνάρτηση την μετατρέπει κατευθείαν στην πραγματική μετάθεση μέσω αποκωδικοποίησης των μεμονωμένων ψηφίων '1', '2' και '3' στα αντίστοιχα 1', '5' και '9'. Στο τέλος λοιπόν της γνωστής διαδικασίας θα έχουμε μια λίστα μεταθέσεις της ακολουθίας '159'. Αυτή δεν ήταν, όμως, η αρχική μας ακολουθία, αλλά η '195'. Φροντίζουμε λοιπόν μέσα στη λίστα των μεταθέσεων να βρούμε την αρχική μας ακολουθία, και μόλις την βρούμε «σπάμε» την λίστα σε δύο κομμάτια, κι έπειτα θέτουμε πρώτο το κομμάτι που ξεκινάει με την αρχική μας λίστα και ακολουθεί το δεύτερο κομμάτι της λίστας. Έτσι έχουμε κάνει μια αναστροφή της λίστας η οποία όμως δεν επηρεάζει τις τιμές γιατί έχουμε δει στο κεφάλαιο όπου εξηγείται θεωρητικά, ότι το πρώτο και το τελευταίο στοιχείο, συνδέονται.

Επομένως, με το τέλος του κώδικα η συνάρτηση επιστρέφει μια λίστα με τις αναθέσεις όπου στη πρώτη θέση είναι η αρχική μας και όλες οι υπόλοιπες ακολουθούν πιστά τον κανόνα του Johnson-Trotter χωρίς να υπάρχουν ανωμαλίες.

4.5.4 Λειτουργία της δοκιμασία Οπτικά Μοτίβα – Visual Patterns

Μετά την ολοκλήρωση της δοκιμασίας μνήμης σειρά έχει ή δοκιμασία οπτικών μοτίβων. Η λειτουργία αυτής της δοκιμασίας είναι σε μεγάλο βαθμό παρόμοιο με το τεστ λεξιλόγιο καθώς και οι δυο είναι δοκιμασίες πολλαπλής επιλογής και το μόνο που αλλάζει είναι ότι η μία χρησιμοποιεί εικόνες στα κουμπιά της ως απάντηση.

Η διαδικασία δημιουργίας των στοιχείων της είναι πανομοιότυπη με αυτή της δοκιμασίας του λεξιλογίου Δηλαδή δημιουργούνται μια φορά και για κάθε επόμενη φορά επαναχρησιμοποιούνται απλώς αυτή τη φορά αντί για ένα αλλάζουμε το κείμενο κάθε φορά στα κουμπιά αλλάζουμε τις εικόνες. Μαζί με την ερώτηση εμφανίζεται και μια εικόνα στο πάνω μέρος των απαντήσεων και από κάτω εμφανίζονται οι επαναχρησιμοποιούμενες εικόνες απαντήσεις.

Κατά την διαδικασία της επιλογής και της υποβολής η συνάρτηση και πάλι ακολουθεί με τον ίδιο ακριβώς τρόπο όπως το e δοκιμασία του λεξιλογίου και χρησιμοποιεί τις ίδιες συναρτήσεις select_option και submit_selected_response. Όταν ο χρήστης κάνει κλικ σε μια εικόνα-απάντηση του παρέχεται η ίδια οπτική επιβεβαίωση, χρωματίζοντας το περίγραμμα, και ενεργοποιεί το κουμπί 'Continue'. Αντίστοιχα με το πάτημα του κουμπιού 'Continue' καταγράφεται ο χρόνος και ελέγχετε εάν η επιλεγμένη εικόνα είναι η σωστή και αποθηκεύεται το αποτέλεσμα. Η κύρια διαφορά του της δοκιμασίας των οπτικών μοτίβων είναι ότι δεν διαθέτει κανόνα πρόωρο τερματισμό όπως έχει δοκιμασία λεξιλογίου με τις 9 στις 10 σωστές απαντήσεις. Στην προκειμένη περίπτωση ο μαθητής πρέπει να απαντήσει και στις 15 ερωτήσεις για να ολοκληρωθεί η διαδικασία.

```
if not self.visual_pattern_widgets_created:
    #create widgets once
    if question_image: #image has been correctly loaded
        self.visual_pattern_question_image = tk.Label(self.answer_grid, image=question_image, bg='#E18E6F')
        self.visual_pattern_question_image.image = question_image
        self.visual_pattern_question_image.grid(row=0, column=0, columnspan=2, pady=(0, 40))
```

Αυτό το κομμάτι κώδικα για παράδειγμα δεν υπάρχει στο λεξιλόγιο. Δείχνει πώς δημιουργείται το Label που θα φιλοξενήσει την κύρια εικόνα του προβλήματος. Ενώ το παρακάτω δείχνει πως κάθε κουμπί κρατάει εικόνα αντί για κείμενο.

```
btn = ttk.Button(
    selection_frame,
    image=option_image,
    command=lambda o=option, idx=button_index: self.select_option(o, idx)
)
btn.image = option_image
btn.pack(padx=2, pady=2)
```

4.5.5 Λειτουργία της δοκιμασία Go/No-Go

Τελευταία δοκιμασία με τη σειρά της είναι αυτή της αναστολής δηλαδή Go/No-Go. Η συγκεκριμένη δοκιμασία διαφέρει αρκετά από όλες τις υπόλοιπες καθώς δεν έχει να κάνει με ερώτηση. Έχει ως στόχο την ανάδειξη ορισμένων εικόνων και αναμένει τον μαθητή να αντιδράσει με βάση αυτές, Δηλαδή αναμένουμε να κάνει κλικ σε μια συγκεκριμένη εικόνα στόχο και να αγνοήσει όλες τις υπόλοιπες.

Η προετοιμασία της ξεκινά ομοίως από την συνάρτηση 'build_question_ui' Όπου δημιουργείται μια λίστα βημάτων τις οποίες πρέπει να ακολουθήσουμε αυστηρά. F συγκεκριμένα ανακατεύει τις εικόνες στόχους και τις εικόνες όχι στόχους και φτιάχνει μια λίστα που περιλαμβάνει την εμφάνιση μιας εικόνας για 3 δευτερόλεπτα και έπειτα μια παύση με κενή οθόνη για 2 δευτερόλεπτα.

```
elif question['type'] == 'go_no_go':
    self.all_go_nogo_images = []

    for i in range(4): #add 4 targets
        self.all_go_nogo_images.append(question['target_image'])
    #add 4 Non  targets
    self.all_go_nogo_images.extend(question['images'])
    #prepare 2 extra Non target RANDOMLY and add them.
    extra_images = random.sample(question['images'], 2)
    self.all_go_nogo_images.extend(extra_images)

    random.shuffle(self.all_go_nogo_images)

    #list for all of the steps needed
    self.gng_steps = []
    #(what_am_I, content_inside, my_duation)
    self.gng_steps.append(('text', "Get ready...", 2000))

    # Add each image and its blank screen to the list
    for image_path in self.all_go_nogo_images[:-1]:
        self.gng_steps.append(('image', image_path, 3000))
        self.gng_steps.append(('pause', None, 2000))
    #for the last one do not add pause
    self.gng_steps.append(('image', self.all_go_nogo_images[-1], 3000))
```

Ο παραπάνω κώδικας είναι που εκτελεί τη διαδικασία αυτή. Πιο συγκεκριμένα δημιουργεί μια λίστα (self.all_go_nogo_images) η οποία περιέχει συνολικά 10 εικόνες, 4 εικόνες στόχους και 6 εικόνες όχι στόχους. Επειδή έχουμε σύνολο 4 βασικές, χρησιμοποιούμε 2 ακόμη με τυχαίο τρόπο. Έπειτα ανακατεύονται με τυχαίο τρόπο οι εικόνες μέσα σε μια λίστα, για να εμφανιστούν με τυχαίο τρόπο στον μαθητή. Στη συνέχεια δημιουργείται μία λίστα 'self.gng_steps' η οποία περιγράφει βήμα-βήμα τι πρόκειται να εμφανιστεί στην οθόνη. Κάθε βήμα περιγράφεται από το είδος του, το τι περιέχει και τη διάρκειά του. Ο κώδικας φροντίζει προσθέσει ένα εισαγωγικό μήνυμα το οποίο σηματοδοτεί την έναρξη της δοκιμασίας κι έπειτα σε έναν βρόχο επανάληψης προσθέτει επαναληπτικά την κάθε εικόνα και μετά από αυτήν την παύση που πρόκειται να την ακολουθήσει.

Έπειτα καλείται η συνάρτηση ‘run_go_nogo_trial’ για να ξεκινήσει την εκτέλεση του πρώτου βήματος από την λίστα που μόλις δημιουργήθηκε. Η συνάρτηση αυτή λοιπόν παίρνει ως παράμετρο έναν δείκτη που της δείχνει σε ποιο βήμα βρισκόμαστε αντίστοιχα και τι πρέπει να εμφανίσει στην οθόνη. Αν ο δείκτης έχει ξεπεράσει το μήκος της λίστας, αυτό σημαίνει ότι η δοκιμασία έχει ολοκληρωθεί. Τότε, η συνάρτηση καλεί το μοντέλο μηχανικής μάθησης για να αξιολογήσει τον μαθητή και έπειτα καθαρίζει τα γραφικά στοιχεία.

```
def run_go_nogo_trial(self,step_index):
    """ Run a trial of go no go, whether for  ···

    self.is_processing = False
    #Step_index keeps track of what needs to be shown now
    if step_index >= len(self.gng_steps):

        # Run ML classification
        self.classify_go_nogo_performance()
        self.current_question += 1

        # FIXED: Clean up go-no-go components properly
        if hasattr(self, 'go_nogo_container'):
            self.go_nogo_container.destroy()

        self.show_finish_screen("Go/No-Go Task Complete!")
        return
    # Get the current step's details from the playlist
    step_type, step_content, step_duration = self.gng_steps[step_index]
```

Αν η δοκιμασία δεν έχει τελειώσει, η συνάρτηση διαβάζει το βήμα που πρέπει να εκτελέσει και σε περίπτωση που έχει να δείξει εικόνα, ξεκινάει το χρονόμετρο για να μετρήσει τον χρόνο αντίδρασης και δείχνει την αντίστοιχη εικόνα στην οθόνη. Το πιο σημαντικό που κάνει είναι ότι ενεργοποιεί τη δυνατότητα για κλικ, μέσω του bind. Από την στιγμή αυτή κι έπειτα ο χρήστης μπορεί να κάνει κλικ και ενεργοποιείται η συνάρτηση ‘handle_go_nogo_click’. Σε περίπτωση που έχει να δείξει ‘pause’ ή το αρχικό μήνυμα ‘Get Ready..’ αδειάζει την οθόνη και απενεργοποιεί τη δυνατότητα για κλικ μέσω του unbind. Πριν τερματίσει η συνάρτηση ‘run_go_nogo_trial’ χρησιμοποιεί την εντολή ‘self.root.after()’ για να προγραμματίσει την επόμενη εκτέλεση της. Για παράδειγμα, αν μόλις έδειξε εικόνα πρέπει να μένει για 3 δευτερόλεπτα, κι έπειτα καλεί τον εαυτό της για το επόμενο βήμα.

```
# Process the step based on its type
if step_type == 'text':
    self.go_nogo_label.config(text=step_content, font=self.go_no_go_font, image='')

elif step_type == 'image':
    question = self.questions[self.current_question]
    if step_content == question['target_image']:
        self.current_is_target= True
    else:
        self.current_is_target=False

    self.gonogo_start_time = time.time() #In case we need to track wrong ones
    image = self.load_image(step_content, size=(300, 300))
    if image:
        self.go_nogo_label.config(image=image, text='')
        self.go_nogo_label.image = image
        self.go_nogo_label.bind('<Button-1>', self.handle_go_nogo_click)

elif step_type == 'pause':
    self.go_nogo_label.config(image='', text='')
    self.go_nogo_label.unbind('<Button-1>')

# Schedule this function to run again for the NEXT step
self.root.after(step_duration, self.run_go_nogo_trial, step_index + 1)
```

Για τη διαχείριση της λειτουργίας κλικ του μαθητή, έχουμε τη συνάρτηση ‘handle_go_nogo_click’. Η συνάρτηση διαχειρίζεται το κάθε κλικ κατά τη διάρκεια της

δοκιμασίας και ενεργοποιείται μόνο όταν μια εικόνα είναι ορατή στην οθόνη. Έχει ως εσωτερική λειτουργία το self.is_processing για να διασφαλίσει ότι δεν θα καταγραφεί κάποιο διπλό κλικ επάνω στην εικόνα. Η συνάρτηση υπολογίσει τον χρόνο αντίδρασης από το κλικ του μαθητή και το καταγράφει και ελέγχει εάν η εικόνα στην οποία έγινε κλικ αποτελεί εικόνα στόχος ή όχι και αναλόγως αποθηκεύει την απάντηση ως σωστή ή ως εσφαλμένη. Μόλις καταγράψει το αποτέλεσμα ολοκληρώνεται η εκτέλεση της και προχωράει το τεστ στο επόμενο βήμα της συνάρτησης run_go_nogo_trial.

```
def handle_go_nogo_click(self, event):
    #prevents double cllicking
    if self.is_processing == False:
        self.is_processing = True
        reaction_time = round(time.time() -self.gonogo_start_time,2)

        if self.current_is_target:
            self.target_image_pressed += 1
            self.go_nogo_reaction_times.append(reaction_time)
        else:
            self.non_target_image_pressed += 1
```

4.6 Μοντέλα Κατηγοριοποίησης - Classifiers

4.6.1 Μοντέλα Δοκιμασιών

Μετά την ολοκλήρωση της κάθε δοκιμασίας, καλείται για καθεμιά από αυτές η αντίστοιχη συνάρτηση που εκτελεί το μοντέλο κατηγοριοποίησης με βάση τον μαθητή. Η διαδικασία είναι σχεδόν πανομοιότυπη για σχεδόν και τις τέσσερεις δοκιμασίες με μόνη διαφορά να αποτελούν τα χαρακτηριστικά τα οποία χρησιμοποιεί η καθεμιά.

```
def classify_vocabulary_performance(self):

    percentage = (self.vocabulary_correct_answers / self.vocabulary_total_questions) * 100
    percentage = round(percentage,2)
    self.vocab_avg_time =round( statistics.median(self.vocabulary_question_times) ,2)

    df = pd.read_csv('vocabulary_dataset.csv', delimiter=';')
```

Για κάθε μεμονωμένη συνάρτηση λοιπόν, το πρώτο βήμα είναι η φόρτωση του κατάλληλου συνόλου δεδομένων εκπαίδευσης από το αντίστοιχο .csv αρχείο μέσω της βιβλιοθήκης pandas έχουμε δει σε προηγούμενο υποκεφάλαιο το περιεχόμενο των αρχείων αυτών για κάθε δοκιμασία. Στην παραπάνω εικόνα απεικονίζεται η ενέργεια αυτή για το μοντέλο του λεξιλογίου. Στην περίπτωση του λεξιλογίου ακριβώς πριν εκτελεστεί η εκπαίδευση, φροντίζουμε να έχουμε προετοιμάσει τις μεταβλητές του μαθητή, εδώ φτιάχνουμε το ποσοστό των σωστών απαντήσεων και τον μέσο χρόνο ανά ερώτηση, για να τα πάρει το μοντέλο και να εξάγει το συμπεράσματα του με βάση αυτά αργότερα.

```
features = ['percentage_correct', 'stopped_early', 'avg_time_per_question']
target = 'classification'

X = df[features]
y = df[target]
```

Στη συνέχεια ο κώδικας, προετοιμάζει τα δεδομένα, διαχωρίζοντας τη βάση δεδομένων σε δύο μέρη, τα χαρακτηριστικά (Features 'X') και τον Στόχο (Target 'y'). Τα πρώτα είναι οι στήλες που πρόκειται να χρησιμοποιηθούν ως είσοδος και το δεύτερο είναι η στήλη με τελική κατηγοριοποίηση του μαθητή.

```
scaler = StandardScaler()
model = SVC(kernel='rbf', class_weight='balanced', random_state=42, probability=True)
X_scaled = scaler.fit_transform(X)
model.fit(X_scaled, y)
```

Τα χαρακτηριστικά έπειτα περνούν από έναν κανονικοποιητή. Η ενέργεια αυτή είναι απαραίτητη για να μετατραπούν όλες οι τιμές σε μια κοινή κλίμακα και για να μπορεί το μοντέλο να δώσει ίση βαρύτητα σε κάθε χαρακτηριστικό. Είναι σημαντικό να γίνεται σε μοντέλα όπως το SVC (Support Machine Classifier) που βασίζεται σε αποστάσεις. Χωρίς την κανονικοποίηση το μοντέλο θα έδινε μεγαλύτερη βαρύτητα σε μεγαλύτερές αριθμητικές τιμές θεωρώντας τις πιο σημαντικές. Ο αλγόριθμος SVC είναι της φύσης SVM που αναλύθηκαν σε προηγούμενο κεφάλαιο και λειτουργεί με την ίδια φιλοσοφία.

Στη συνέχεια το μοντέλο SVC παίρνει τα κανονικοποιημένα χαρακτηριστικά και τους στόχους για αυτές (classification_class) και ξεκινάει την εκπαίδευση. Η παράμετρος 'rbf' ορίζει ότι το μοντέλο θα χρησιμοποιήσει γραμμικά όρια, όπως στο παράδειγμα που είχαμε δει στο SVM, αλλά καμπύλες, για να διαχωρίσει καλύτερα τα δεδομένα. Η παράμετρος 'balanced' προσαρμόζει την βαρύτητα των στόχων να είναι ισορροπημένη, ενώ η παράμετρος 'random_state=42' ορίζει ότι υπάρχει τυχαιότητα κατά την εκτέλεση του μοντέλου, αλλά η τυχαιότητα είναι ορισμένη με τον παραπάνω αριθμό και θα δίνει ίδια αποτελέσματα για κάθε εκτέλεση. Τέλος, η παράμετρος 'probability=True' δίνει την εντολή στο μοντέλο, να εκτελέσει υπολογισμούς έτσι ώστε να μας δώσει και το ποσοστό βεβαιότητας για κάθε πιθανή κατηγορία.

```
student_df = pd.DataFrame([[percentage, self.vocabulary_stopped_early , self.vocab_avg_time]],
    columns=features)
student_df_scaled = scaler.transform(student_df)

prediction = model.predict(student_df_scaled)[0]
probability = model.predict_proba(student_df_scaled)[0]
```

Αφού το μοντέλο έχει εκπαιδευτεί είναι πλέον έτοιμο να αξιολογήσει την απόδοση του μαθητή. Αρχικά, τα δεδομένα του μαθητή τοποθετούνται σε έναν πίνακα DataFrame της βιβλιοθήκης pandas για να διασφαλίσουμε ότι τα δεδομένα έχουν ακριβώς την ίδια δομή με τα δεδομένα εκπαίδευσης, Δηλαδή στο επόμενο βήμα εφαρμόζουμε κανονικοποίηση στα δεδομένα του μαθητή για να έχουν την ίδια κλίμακα με τα δεδομένα εκπαίδευσης.

Έπειτα, τα δεδομένα του μαθητή δίνονται στο εκπαιδευμένο μοντέλο, για την τελική πρόβλεψη όπου δίνεται η τελική κατηγορίας του μαθητή (-1,0 ή 1). Αμέσως μετά επιστρέφονται οι πιθανότητες που υπολογίζει το μοντέλο, για καθεμιά από τις τρεις πιθανές κατηγορίες, δηλαδή το confidence του μοντέλου.

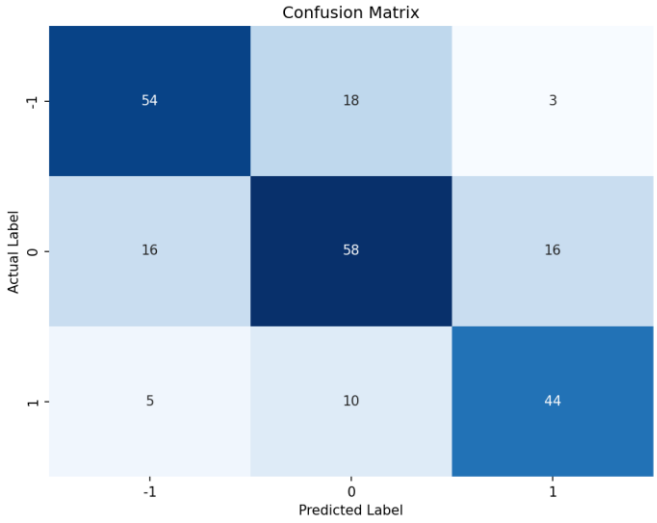
```
# Make probabilities to classes
class_labels = [-1,0,1]
probability_per_class = {}
for i in range(len(class_labels)):
    probability_per_class[i-1] = round(probability[i]* 100,1)

# Store result for Final classifier
self.test_results['vocabulary'] = {
    'prediction': int(prediction),
    'confidence': probability_per_class
}
```

Στο τέλος του κώδικα, η πρόβλεψη και οι πιθανότητες ομαδοποιούνται και αποθηκεύονται σε ένα λεξικό για να χρησιμοποιηθούν στον τελικό κατηγοριοποιητή.

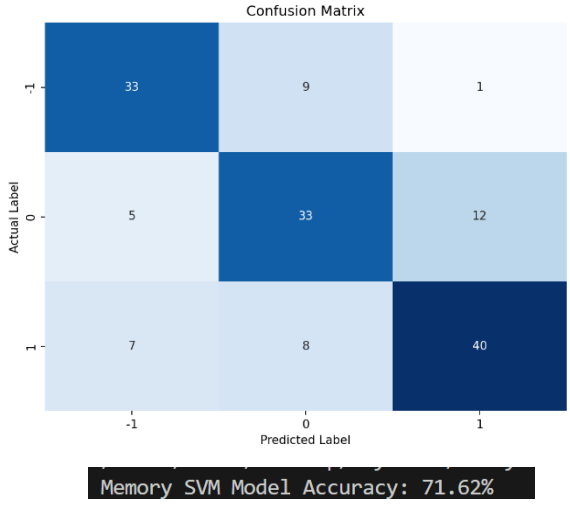
4.6.1.1 Αποτελεσματικότητα Κατηγοριοποιητών

Για κάθε μοντέλο Κατηγοριοποίησης, εκτελέστηκε ξεχωριστά αξιολόγηση μεσω εκπαίδευσης μοντέλου και Πίνακα Απόδοσης. Παρακάτω είναι τα αποτελέσματα που υπήρξαν για κάθε δοκιμασία ξεχωριστά:

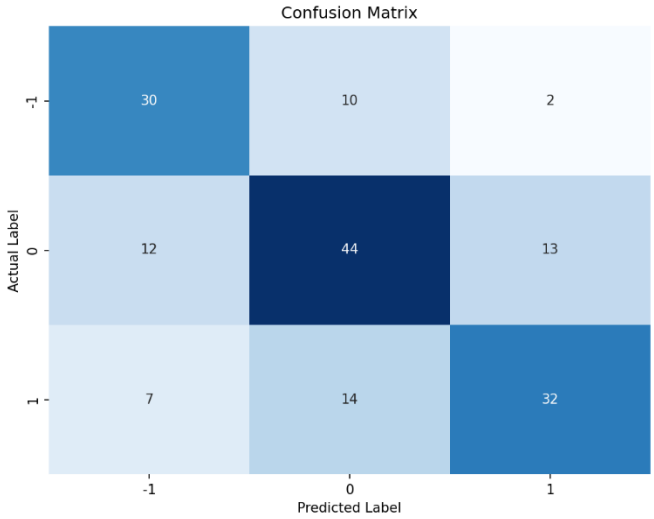
- **Λεξιλόγιο :** Ο Πίνακας απόδοσης δείχνει αρκετά έντονα της διαγώνιο, πράγμα που σημαίνει ότι το μοντέλο έχει προβλέψει σωστά τις περισσότερες φορές το τελικό αποτέλεσμα. Σε γενικές γραμμές είναι αρκετά καλά υποσχόμενα τα αποτελέσματα του, και η συνολική του απόδοση με σχεδόν 70% σωστή πρόβλεψη. Δεν θέλουμε να είναι πολύ υψηλό, αλλά ούτε και πολύ χαμηλό.



- **Οπτική Εργαζόμενη Μνήμη:** Εδώ τα αποτελέσματα δείχνουν πάλι επιτυχή εκπαίδευση και πρόβλεψη, καθώς τα όριά της διαγώνιου είναι καλά θετημένα και βλέπουμε ότι ελάχιστες φορές έχει μπερδέψει κάποια κατηγορία με κάποια εσφαλμένη. Το ποσοστό επιτυχία είναι στα 71% εξίσου ισορροπημένο.

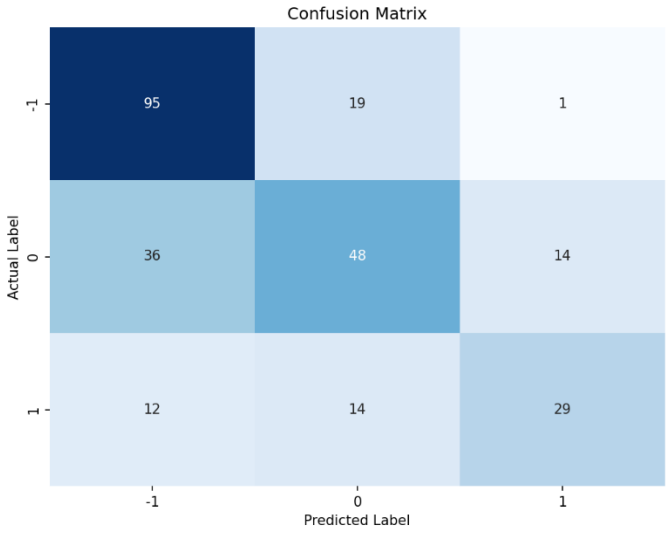


- **Οπτικά Μοτίβα:** Εδώ το μοντέλο δείχνει να συγχέεται περισσότερο, αλλα είναι λογικό, καθώς για το συγκεκριμένο μοντέλο υπήρχαν μόνο δυο διαθέσιμα χαρακτηριστικά για την εκπαίδευση του, οι σωστές απαντήσεις και ο μέσος χρόνος για αυτές. Παρά την σύγχυση όμως, το μοντέλο καταφέρνει να έχει μια σχετικά έντονη διαγώνιο, δηλαδή να μην μπερδεύει σε σημαντικό βαθμό τις κατηγορίες μεταξύ τους.




```
Visual Pattern SVM Model Accuracy: 64.63%
```

- **Go/No-Go:** Το τελευταίο μοντέλο, δείχνει να μην έχει τόσο καθαρή εικόνα στον πίνακα απόδοσης της, αλλά όπως είδαμε προηγουμένως στην ανάλυση δεδομένων, αυτό οφείλεται κυρίως στο ότι μετράμε τις λάθος απαντήσεις και θέλουμε να έχουν και αυτές αντίκτυπο στην κατηγορία που θα δοθεί. Παρόλα αυτά όμως, στη πλειοψηφία του το μοντέλο προβλέπει με επιτυχία τις κατηγορίες και σημειώνει επιτυχία με ποσοστό 64%




```
Go/No-Go SVM Model Accuracy: 64.18%
```

4.6.2 Τελικός Κατηγοριοποιητής

Μετά την εκτέλεση των τεσσάρων κατηγοριοποιητών, το τελευταίο βήμα για την αξιολόγησή του μαθητή είναι η εκτέλεση του τελικού κατηγοριοποιητή. Τα δεδομένα που λαμβάνει αυτός είναι τα αποτελέσματα των προηγούμενων κατηγοριοποιητών και μέσω αυτών βγάζει ένα τελικό ολιστικό συμπέρασμα. Δημιουργεί για αρχή λίστα με τα 16 χαρακτηριστικά που έχει συνολικά, δηλαδή η τελική πρόβλεψη για κάθε δοκιμασίας και ταυτόχρονα η τιμή confidence για κάθε κατηγορία που υπήρξε.

```
df = pd.read_csv('final_classification.csv', delimiter=';')

features = [
    'vocab_pred', 'vocab_conf_-1', 'vocab_conf_0', 'vocab_conf_1',
    'mem_pred', 'mem_conf_-1', 'mem_conf_0', 'mem_conf_1',
    'visual_pred', 'visual_conf_-1', 'visual_conf_0', 'visual_conf_1',
    'gonogo_pred', 'gonogo_conf_-1', 'gonogo_conf_0', 'gonogo_conf_1'
]
target = 'overall_classification'
```

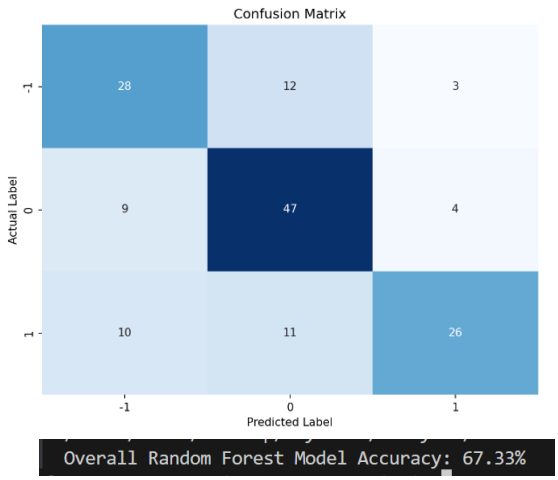
Ακολουθεί η εκπαίδευση του μοντέλου, σε αυτή τη περίπτωση χρησιμοποιείται μοντέλο 'Random Forest' αντί για SVC που χρησιμοποιήσαμε για τις δοκιμασίες, διότι αυτό έφερε καλύτερα αποτελέσματα για τη συγκεκριμένη βάση δεδομένων, από ότι ο SVC. Επομένως, εδώ δεν χρειάζεται κανονικοποίηση δεδομένων.

```
X = df[features]
y = df[target]

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X, y)

student_df = pd.DataFrame([student_features], columns=features)

prediction = model.predict(student_df)[0]
probability = model.predict_proba(student_df)[0]
```

Αφότου εκπαιδευτεί το μοντέλο, δίνονται οι τιμές που έχουμε συλλέξει από το πρόγραμμα και βγάλει το τελικό αποτέλεσμα της κατηγορίας του μαθητή, όπως και το confidence που έχει για την απάντηση του και την αποθηκεύει. Αυτή αργότερα εμφανίζεται στην οθόνη του εκπαιδευτικού. Την οποία θα μελετήσουμε παρακάτω.

4.6.2.1 Αποτελεσματικότητα Τελικού Κατηγοριοποιητή

Το μοντέλο από τον πίνακα απόδοσης του δεν φαίνεται να συγχέεται ιδιαίτερα, έχει αρκετά καθαρά τα όρια της διαγώνιου και σχετικά καλή απόδοση ακρίβειας 67%



4.7 Αποθήκευση Στοιχείων Μαθητή

Εφόσον ολοκληρωθεί και το τελευταίο κομμάτι για τον μαθητή, δηλαδή έχει ολοκληρώσει και τις τέσσερεις δοκιμασίες, εκτελείται η συνάρτηση 'save_results_and_finish'. Ο ρόλος της συγκεκριμένης συνάρτησης είναι να συγκεντρώσει, να αποθηκεύσει και να παρουσιάσει την τελική οθόνη της δοκιμασίας. Το πρώτο πράγμα που κάνει η συνάρτηση αυτή είναι να καλέσει τον τελικό κατηγοριοποιητή για να λάβει την τελική αξιολόγηση του μαθητή. Στη συνέχεια δημιουργεί ένα μεγάλο λεξικό 'student' το οποίο λειτουργεί ως τελικό αρχείο του μαθητή και σε αυτό συγκεντρώνονται όλες οι βασικές πληροφορίες του μαθητή. Δηλαδή, κρατάει τα στοιχεία εισαγωγής του (ονοματεπώνυμο, ηλικία), την πλήρη αναλυτική λίστα 'answers' με κάθε απάντηση και τον χρόνο καθεμίας, Τις αναφορές και τα αποτελέσματα από τα μοντέλα μηχανικής μάθησης για καθεμιά δοκιμασία στο 'summary', καθώς και το τελικό αποτέλεσμα του κατηγοριοποίηση στο σημείο 'overall_classification'.

```
student = {
    "info": self.student_info,
    "answers": self.answers,
    "date_time": datetime.datetime.now().strftime("%d/%m/%Y %H:%M:%S"),
    "summary": {
        "vocabulary": {
            "score": f"{self.vocabulary_correct_answers}/{self.vocabulary_total_questions}",
            "total_questions": self.vocabulary_total_questions,
            "avg_time": self.vocab_avg_time,
            "stopped_early":self.vocabulary_stopped_early,
            "ml_result": self.test_results['vocabulary']
        },
        "memory": {
            "score": f"{self.memory_total_correct }/{self.memory_total_attempts}",
            "avg_time": self.memory_avg_time,
            "level_performance": self.level_performance,
            "sequence_analysis_details": self.sequence_analyses,
            "ml_result": self.test_results['memory']
        },
        "visual_pattern": {
            "score": f"{self.visual_pattern_correct_answers}/{15}",
            "avg_time": self.visual_pattern_avg_time,
            "ml_result": self.test_results['visual_pattern']
        },
        "go_nogo": {
            "score": f"{self.target_image_pressed}/4",
            "false_alarms": self.non_target_image_pressed,
            "avg_time": self.go_nogo_avg_time,
            "all_reaction_times": self.go_nogo_reaction_times,
            "ml_result": self.test_results['go_nogo']
        }
    },
    "overall_classification":self.overall_classification
```

Έπειτα ακολουθεί διαδικασία εγγραφής των στοιχείων αυτών στο αρχείο 'student_results.json' για να μπορούν να εμφανιστούν σε επόμενες χρήσεις στην οθόνη του εκπαιδευτικού. Συγκεκριμένα, διαβάζει πρώτα το τρέχον αρχείο, προσθέτει τον νέο στοιχείο του τρέχοντος μαθητή και ξαναγράφει ολόκληρη την ενημερωμένη λίστα χρησιμοποιώντας τον NpEncoder για τη διαχείριση των ειδικών τύπων την Numpy (έχει αναλυθεί σε προηγούμενο κεφάλαιο).

4.8 Επίπεδο Εκπαιδευτικού

Το επίπεδο στο οποίο έχει πρόσβαση ο εκπαιδευτικός είναι κατά βάση ο πίνακας δεδομένων για κάθε μαθητή που έχει ολοκληρώσει τη δοκιμασία έως τώρα. Προτού όμως φτάσει εκεί, χρειάζεται πρώτα να εισαχθεί στο σύστημα με συγκεκριμένα στοιχεία εισαγωγής.

Η διαδικασία αυτή ξεκινάει με τη συνάρτηση 'show_teacher_login' η οποία ενεργοποιείται όταν πατήσει από την αρχική οθόνη το κουμπί 'Teacher'. Η συνάρτηση αυτή καθαρίζει την οθόνη και δημιουργεί μια παθλή φόρμα που ζητά όνομα χρήστη και κωδικό πρόσβασης, η τοποθέτηση της στο χώρο λειτουργεί ακριβώς με την ίδια λογική της φόρμας του μαθητή, δηλαδή με πίνακα.

```
# Create and place the username label and entry
username_label = tk.Label(self.login_form,
    text="Username:",
    font=self.answer_font,
    bg='#E18E6F')
username_label.grid(row=0, column=0, sticky='w', pady=10, padx=10)

self.username_entry = ttk.Entry(self.login_form, width=30, font=self.answer_font)
self.username_entry.grid(row=0, column=1, sticky='ew', pady=10, padx=10)
self.username_entry.bind("<Return>", self.handle_enter_key_login_form)

# Create and place the password label and entry
password_label = tk.Label(self.login_form, text="Password:", font=self.answer_font, bg='#E18E6F')
password_label.grid(row=1, column=0, sticky='w', pady=(10,20), padx=10)

self.password_entry = ttk.Entry(self.login_form, width=30, font=self.answer_font, show="*")
self.password_entry.grid(row=1, column=1, sticky='ew', pady=(10,20), padx=10)
self.password_entry.bind("<Return>", self.handle_enter_key_login_form)
```

Με το πάτημα του κουμπιού 'Login' η συνάρτηση 'process_teacher_login' ελέγχει αν τα στοιχεία που δόθηκαν ταιριάζουν με τα προκαθορισμένα και αποθηκευμένα στοιχεία του καθηγητή. Αν αυτά είναι σωστά, η εφαρμογή οδηγεί στον πίνακα ελέγχου με τα αποτελέσματα. Σε αντίθετη περίπτωση εμφανίζεται μήνυμα λάθους.

```
def process_teacher_login(self):
    """Process the teacher login credentials from direct variables."""
    # Get username and password from the new instance variables
    username = self.username_entry.get().strip()
    password = self.password_entry.get().strip()

    if username == self.teacher_credentials["username"] and password == self.teacher_credentials["password"]:
        #correct login, delete widgets and show student data
        self.login_form.destroy()
        self.canvas.delete(self.login_title_id)
        self.show_student_results()
    else:
        messagebox.showerror("Error", "Invalid username or password.")
```

4.8.1 Πίνακας Ελέγχου Αποτελεσμάτων

Μετά την επιτυχημένη σύνδεση, ο καθηγητής αντικρίζει τον κεντρικό πίνακα ελέγχου, ο οποίος είναι σχεδιασμένος σε μια διάταξη τριών πάνελ.

- **Αριστερό Πάνελ**: Περιέχει λίστα με όλους τους μαθητές που έχουν ολοκληρώσει τη δοκιμασία και για κάθε έναν από αυτούς εμφανίζονται βασικές πληροφορίες όπως όνομα, επώνυμο, ηλικία και ημερομηνία δημιουργίας εισαγωγής.
- **Κεντρικό Πάνελ**: Όταν ο καθηγητής κάνει κλικ σε έναν μαθητή από την αριστερή λίστα, το κεντρικό πάνελ ενημερώνεται δυναμικά. Εμφανίζει "κάρτες" με τη συνοπτική απόδοση του επιλεγμένου μαθητή για κάθε μία από τις τέσσερις δοκιμασίες ξεχωριστά, καθώς και την τελική, συνολική αξιολόγηση. Κάθε κάρτα περιλαμβάνει το σκορ, τον μέσο χρόνο και, το πιο σημαντικό, την ταξινόμηση που έκανε το μοντέλο της Μηχανικής Μάθησης.
- **Δεξί Πάνελ:** Αν ο καθηγητής κάνει κλικ σε μία από τις κάρτες στο κεντρικό πάνελ (π.χ. Τεστ Μνήμης), το δεξί πάνελ ζωντανεύει. Εμφανίζει μια εξαντλητικά λεπτομερή ανάλυση της απόδοσης του μαθητή στη συγκεκριμένη δοκιμασία, δείχνοντας απαντήσεις ανά ερώτηση, χρόνους, συγκεκριμένα λάθη, όπως οι μεταθέσεις στη μνήμη και την αναλυτική διάσπαση των ποσοστών βεβαιότητας του μοντέλου.

4.8.1.1 Αριστερό Πάνελ

Η συνάρτηση που είναι υπεύθυνη για τη δημιουργία αυτού του πάνελ είναι ‘create_left_panel’. Δημιουργεί τα απαραίτητα πλαίσια που θα χρειαστούν για να φανούν τα δεδομένα. Στη συνέχεια ανοίγει και διαβάζει ολόκληρο το αρχείο με τα στοιχεία μαθητών και φορτώνει τις εγγραφές σε μια λίστα. Για να δείξει τα δεδομένα αυτά δημιουργεί ένα στοιχείο τύπου ‘ttk.Treeview’, που τα δείχνει με μορφή πίνακα στήλες και γραμμές. Διατρέχει επαναληπτικά μέσα από τα στοιχεία που ανάκτησε από το αρχείο μαθητών και προσθέτει μια-μια νέα γραμμή στον πίνακα.

```
# Create treeview for student list
columns = ('name', 'surname', 'age', 'class', 'date_time')
self.students_tree = ttk.Treeview(list_container, columns=columns, show='headings', height=21)

# Configure columns
self.students_tree.heading('name', text='Name')
self.students_tree.heading('surname', text='Surname')
self.students_tree.heading('age', text='Age')
self.students_tree.heading('class', text='Class')
self.students_tree.heading('date_time', text='Date')

# Set column widths
self.students_tree.column('name', width=95)
self.students_tree.column('surname', width=95)
self.students_tree.column('age', width=40)
self.students_tree.column('class', width=50)
self.students_tree.column('date_time', width=85)
```

```
for student in student_results:
    info = student['info']
    date_display = student.get("date_time", "N/A")

    self.students_tree.insert('', 'end', iid=str(i), values=(
        info['name'],
        info['surname'],
        info['age'],
        info.get('class') or "N/A",
        date_display
    ))
    i += 1
# Add scrollbar and set it
left_scrollbar = ttk.Scrollbar(list_container, orient=tk.VERTICAL, command=self.students_tree.yview)
self.students_tree.config(yscroll=left_scrollbar.set)

left_scrollbar.pack(side='right', fill='y')
self.students_tree.pack(side='left', fill='both', expand=True)

# If a click is done on a row call method
self.students_tree.bind('<ButtonRelease-1>', self.on_student_selected)
```

Στο τέλος της δεσμεύει το γεγονός του κλικ με τη συνάρτηση ‘on_student_selected’. Αυτή η συνάρτηση είναι υπεύθυνη για τη σύνδεση του αριστερού με το κεντρικό πάνελ. Ο ρόλος της είναι να αναγνωρίσει την επιλογή του μαθητή που πατήθηκε, να βρει την πλήρη εγγραφή για αυτό τον μαθητή στη λίστα με τα στοιχεία όλων των μαθητών κι έπειτα να το δώσει ως παράμετρο στην συνάρτηση ‘display_center_panel_summary’.

4.8.1.2 Κεντρικό Πάνελ

Το κεντρικό πάνελ δημιουργείται από την συνάρτηση ‘create_center_panel’ και δημιουργείται μόνο μία φορά. Ο ρόλος της είναι να δημιουργεί το βασικό πλαίσιο, τίτλο και το σημαντικό προσωρινό μήνυμα που προτρέπει τον χρήστη να επιλέξει έναν μαθητή από την αριστερή λίστα για να ενεργοποιηθεί η συνάρτηση display_center_panel_summary’.

Η συνάρτηση ‘display_center_panel_summary’ είναι αυτή που όπως είπαμε ενεργοποιείται με κάθε κλικ που γίνεται επάνω σε κάποιο μαθητή στο αριστερό πάνελ. Λαμβάνει ως παράμετρο το αρχείο που περιέχει τις πληροφορίες για τον μαθητή που επιλέχθηκε. Κάθε φορά που εκτελείται φροντίζει να καθαρίσει την οθόνη από στοιχεία προηγούμενου μαθητή που είχαν πατηθεί. Στη συνέχεια δημιουργεί «Κάρτες» για κάθε δοκιμασία με δεδομένα που λαμβάνει από τη λίστα summary. Κάθε κάρτα δημιουργεί ετικέτες για σύνολο σωστών απαντήσεων, μέσο χρόνο απάντησης και την κατηγοριοποίηση που έδωσε το μοντέλο μηχανικής μάθησης μαζί με το αντίστοιχο confidence.

```
click_action = lambda event, key=quiz_type: self.display_right_panel_details(key, student_info)
for widget in [card_frame, title_label, info_label, classif_label, confidence_label]:
    #make the whole card clickable
    widget.bind("<Button-1>", click_action)
```

Το παραπάνω κομμάτι κώδικα είναι και το ποιο σημαντικό, καθώς για κάθε κάρτα κου δημιουργείται, ορίζεται μια lambda συνάρτηση η οποία κρατάει τον τύπο της δοκιμασία που πατήθηκε (‘memory’) και τη δεσμεύει μέσω bind στο μήκος όλης της κάρτας, δηλαδή το πλαίσιο, ο τίτλος και οι ετικέτες της.

Μέσα στην ‘display_center_panel_summary’ δημιουργείται ξεχωριστά η κάρτα για τη συνολική αξιολόγηση που εκτέλεσε ο Τελικός Κατηγοριοποιητής. Η συγκεκριμένη κάρτα είναι η μόνη, που με το πάτημα επάνω της δεν ενεργοποιεί κάποιο γεγονός. Κι αυτό, διότι αποτελεί το τελικό συμπέρασμα και δεν υπάρχει κάποια πιο λεπτομερής ανάλυση για να εμφανίσει.

```
#Overall Classification Card
final_classif_frame = tk.Frame(center_scrollable_frame, bg="#f8f4a9", relief='raised', bd=2,
                               width=200, height=190)
final_classif_frame.pack_propagate(False)
final_classif_frame.pack(fill='x', pady=5)

overall_class = student_info['overall_classification']
prediction = overall_class['prediction']
confidence_breakdown = overall_class['confidence_breakdown']
classification_text = pred_meanings[prediction]
main_text = f"Overall Classification:\n {classification_text} ({overall_class['confidence']:.1f}%)"
```

4.8.1.3 Δεξί Πάνελ

Το δεξί πάνελ είναι το τελικό επίπεδο ανάλυσης, παρέχοντας λεπτομερώς δεδομένα για κάθε δοκιμασία του μαθητή. Η δημιουργία των βασικών του στοιχείων πραγματοποιείται από τη συνάρτηση 'create_right_panel' η οποία καλείται μια φορά. Σκοπός της είναι να κατασκευάσει πλαίσιο και να τοποθετήσει σε ένα μήνυμα που να παροτρύνει τον χρήστη να πατήσει επάνω σε κάποια κατηγορία τεστ από το κεντρικό πάνελ για να δει τις λεπτομέρειές. Καθοριστικό ρόλο παίζει η συνάρτηση 'display_right_panel_details', η οποία ενεργοποιείται κάθε φορά που ο χρήστης πατάει σε κάποια καρτέλα δοκιμασίας στο κεντρικό πάνελ και λαμβάνει ως παραμέτρους τον τύπο της δοκιμασίας και τον πλήρη φάκελο του μαθητή.

Η συνάρτηση αυτή αρχικά καθαρίζει από οπουδήποτε προηγούμενο περιεχόμενο μπορεί να υπήρχε και έπειτα δημιουργεί ένα καμβά μαζί με μια μπάρα κύλισης για να διασφαλίσει ότι ο χρήστης θα μπορεί να δει όλες τι πληροφορίες, καθώς είναι αρκετές και δεν χωράνε όλες στην οθόνη. Έπειτα αναλόγως με τη παράμετρο κλειδί που της δόθηκε καλεί μια διαφορετική συνάρτηση υπεύθυνη για την λεπτομερής εμφάνιση του κάθε τύπου ερώτησης.

<u>Λεπτομέρειες Λεξιλογίου</u>:

Η συνάρτηση 'add_vocabulary_details' είναι υπεύθυνη για την παρουσίαση των λεπτομερειών λεξιλογίου. Παίρνει τα απαραίτητα δεδομένα του μαθητή και τα εμφανίζει. Πιο συγκεκριμένα, μέσα σε βρόχο επανάληψης, διατρέχει μια προς μια τις απαντήσεις του μαθητή και δημιουργεί κείμενο το οποίο περιέχει τον αριθμό της ερώτησης, τον χρόνο που χρειάστηκε για αν την απαντήσει, εάν αυτή ήταν σωστή ή λάθος και την τελική απάντηση που έδωσε ο μαθητής.

```
full_text = "Individual Question Performance:\n"
question_number=1
for answer in test_answers:
    time_val = answer['time']
    if answer['correct']:correct_mark ="✓"
    else: correct_mark= "✗"
    student_answer = answer['answer']

    full_text += f"Q{question_number}: time: {time_val}s {correct_mark}  |  Answer given: {student_answer}\n"
    question_number += 1
```

Στη συνέχεια, προσθέτει μια ενότητα με τα συνολικά στατιστικά της δοκιμασίας, από το αρχείο 'summary_data', δείχνοντας το τελικό σκορ, τον μέσο χρόνο αλλά και αν υπήρξε πρόωρος τερματισμός δοκιμασίας. Στο τέλος της, δημιουργεί μια ξεχωριστή ενότητα με τα αποτελέσματα της Μηχανικής Μάθησης, όπου παρουσιάζει την τελική πρόβλεψη του

```
ml_result = summary_data.get('ml_result')
if ml_result:
    confidence_dict = ml_result['confidence']
    prediction = ml_result['prediction']

    pred_meanings = {-1: "Below Average", 0: "Average", 1: "Above Average"}

    ai_text = f"\nAI Prediction for Vocabulary:\n"
    ai_text += f"  {pred_meanings[prediction]} ({prediction})\n"

    #Show confidence for all 3 predictions
    ai_text += "\nConfidence Breakdown:\n"
    ai_text += f"  • Below Average (-1): {confidence_dict['-1'] :.1f}%\n"
    ai_text += f"  • Average (0):        {confidence_dict['0'] :.1f}%\n"
    ai_text += f"  • Above Average (1):  {confidence_dict['1'] :.1f}%\n"

    ai_label = tk.Label(parent, text=ai_text, font=('Arial', 12,'bold'),
                        bg='#f0f8ff', justify='left', fg="#1B3851")
    ai_label.pack(anchor='w',padx=10, pady=(7, 5))
```

μοντέλου, και την αναλυτική αυτοπεποίθηση (confidence) που έδειξε το μοντέλο για τη καθεμια κατηγορία ξεχωριστά.

Λεπτομέρειες Οπτικών Μοτίβων:

Η λειτουργία υπεύθυνη για την εμφάνιση της στο δεξί πάνελ είναι της συνάρτησης 'add_visual_pattern_details' η λειτουργία της είναι σχεδόν πανομοιότυπη με την αντίστοιχη του Λεξιλογίου καθώς και οι δύο δοκιμασίες είναι τύπου πολλαπλής επιλογής. Η μοναδική, αλλά σημαντική διαφορά, έγκειται στον τρόπο που παρουσιάζεται η απάντηση του μαθητή. Επειδή η απάντηση είναι αποθηκευμένη ως η πλήρης διαδρομή προς το αρχείο της εικόνας. Υπό άλλες τις άλλες περιπτώσεις, η συνάρτηση είναι ακριβώς η ίδια.

Λεπτομέρειες Μνήμης:

Η συνάρτηση υπεύθυνη να δείξει τις συγκεκριμένες λεπτομέρειες έιναι η 'add_memory_details_new'. Η συγκεκριμένη είναι πιο περίπλοκη, καθώς η δοκιμασία την μνήμης διατηρεί περισσότερες πληροφορίες για τις ακολουθίες του μαθητή. Αρχικά παρουσιάζει μια γρήγορη σύνοψη της απόδοσης για κάθε επίπεδο δυσκολίας, δηλαδή πόσες σωστές και πόσες εσφαλμένες είχε ανά επίπεδο.

```
# Performance summary
full_text = "Level Performance:\n"
level_performance = summary_data['level_performance']
for level_num in [1,2,3]:
    #performance=level_performance.get(str(level_num))
    performance=level_performance[str(level_num)]
    #"level_performance": {"1": [false,false,false]
    #                      "2": []
    #                      "3": []  }
    correct_count = performance.count(True)
    total_count = len(performance)
    performance_str=""
    for result in performance:
        if result:performance_str+= "✓"
        else: performance_str+= "✗"
    full_text += f"Level {level_num}: {correct_count}/{total_count}  - {performance_str}\n"
```

Στη συνέχεια παρέχει μια λεπτομερή ανάλυση για κάθε μια προσπάθεια του μαθητή, για κάθε μεμονωμένη ακολουθία στην οποία έδωσε απάντηση. Συγκεκριμένα αναγράφεται

```
for i in range(len(performance)):

    answer = test_answers[sequence_counter]
    analysis = analysis_list[sequence_counter]

    if answer['correct']: correct_mark = "✓"
    else: correct_mark= "✗"
    time_val = answer['time']

    full_text += f"\n  Expected: {answer['expected_sequence']}  |  Given: {answer['answer']} {correct_mark}\n"
    full_text += f"  Time taken to answer: {time_val} seconds\n"
    full_text += f"  - Digit Recall: {analysis['digit_recall']*100}%\n"
    full_text += f"  - Position Accuracy: {analysis['position_accuracy']*100}%\n"

    if analysis['is_complete_reversal']:
        full_text += "  - Pattern: Complete sequence reversal\n"

    if analysis['is_exact_permutation']:
        distance = analysis['trotter_distance']
        full_text += f"  - Johnson Trotter:\n     Spotted permutation ({distance} steps away)\n"
        path = analysis['permutation_path']
        full_text += f"      - Path: {path}\n"
    sequence_counter += 1
```

η αναμενόμενη απάντηση του μαθητή και η πραγματική, ο χρόνος που χρειάστηκε για να απαντήσει καθώς και οι μετρήσεις που υπολογίστηκαν από τη συνάρτηση 'analyze_sequence_patterns'.

Στο τέλος της, όπως και για κάθε προηγούμενη δοκιμασία, υπάρχει αναφορά για τα αποτελέσματα του μοντέλου μηχανικής μάθησης, δηλαδή η κατηγορία στην οποία εντάσσει τον μαθητή και το πόσο σίγουρο είναι για αυτό.

Λεπτομέρειες Go/No-Go:

Η συνάρτηση 'add_go_nogo_details' είναι υπεύθυνη για την ανάδειξη των λεπτομερειών στο δεξί πάνελ για τη συγκριμένη δοκιμασία. Στο σημείο αυτό παρέχεται μια επισκόπηση των σωστών κλικ του μαθητή επάνω στις εικόνες στόχους και τα εσφαλμένα κλικ πάνω σε εικόνες όχι στόχους συνοδευόμενο από τον μέσο χρόνο αντίδρασης για τα σωστά κλικ. Στη συνέχεια φαίνονται οι πιο λεπτομερής ανάλυση των σωστών απαντήσεων μαζί με τον χρόνο για κάθε ένα από αυτά. Και στο τέλος, όπως και στις υπόλοιπες δοκιμασίες αναφέρεται η ανάλυση της τελικής κατηγοριοποίησης για τη συγκεκριμένη δοκιμασία.

```
full_text = "Performance Summary:\n"
full_text += f"• Correct Targets: {summary_data['score']}\n"
full_text += f"• False Alarms: {summary_data['false_alarms']}/6\n"
full_text += f"• Average Reaction Time: {summary_data['avg_time']}s\n"

reaction_times = summary_data['all_reaction_times']

full_text += "\nTarget Reaction Times:\n"
target_number = 1
for time_val in reaction_times:
        full_text += f"Target {target_number}: {time_val:.3f}s\n"
        target_number += 1
```

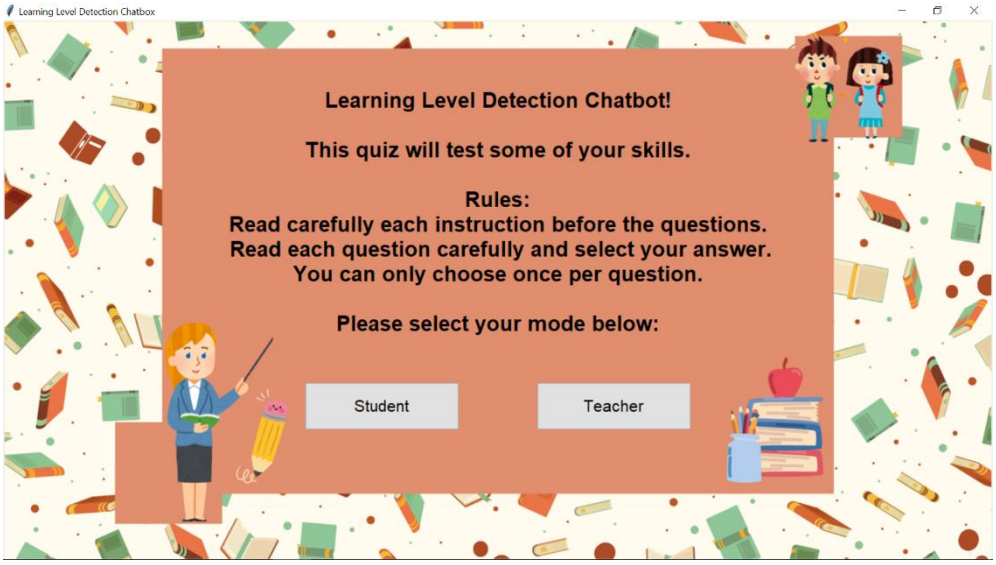
5.Εκτέλεση Εφαρμογής και Συμπεράσματα

5.1 Εκτέλεση Εφαρμογής

Εφόσον εξηγήθηκε η πλήρης λειτουργία της εφαρμογής, μπορούμε πλέον να δείξουμε και πως λειτουργεί, αυτή για τον χρήστη. Θα δούμε αναλυτικά την λειτουργία του μαθητή και του εκπαιδευτικού μέσα από στιγμιότυπα.

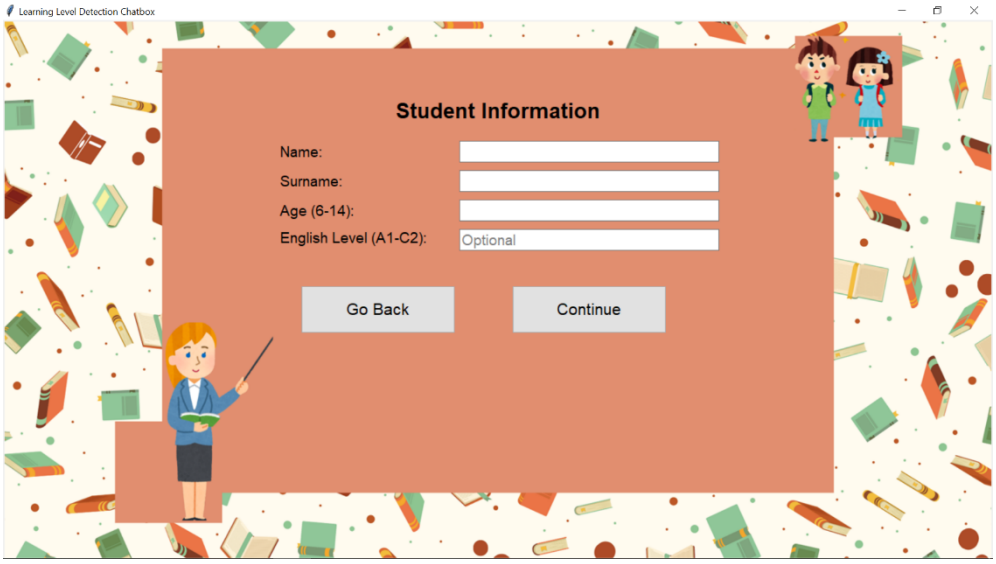
5.1.1 Αρχική Οθόνη

Η αρχική οθόνη, όπως είδαμε και σε επίπεδο κώδικα περιέχει το κείμενο με τις αρχικές οδηγίες και τα κουμπιά «Μαθητή» και «Εκπαιδευτικού»



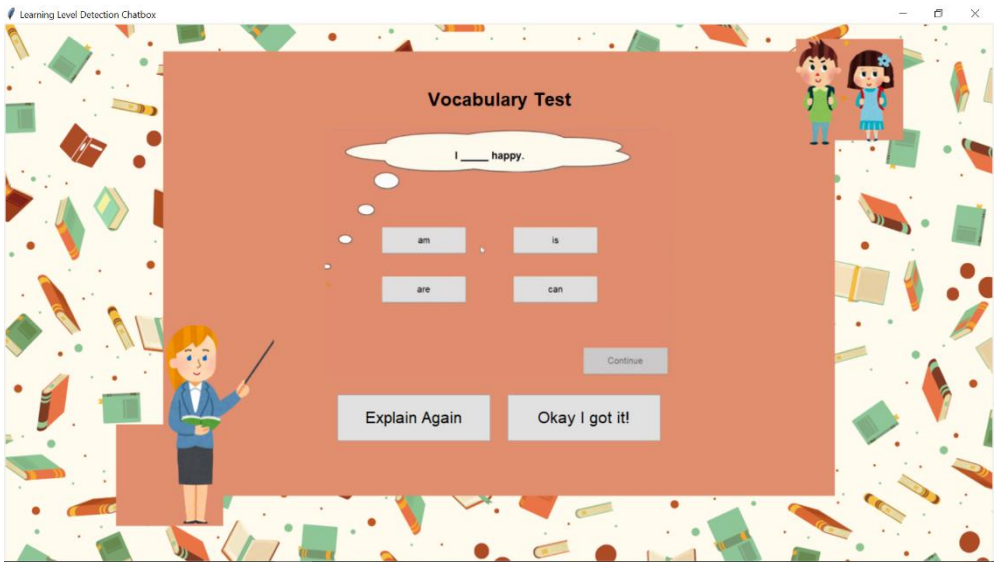
5.1.2 Επίπεδο Μαθητή

Με το πάτημα του κουμπιού «Μαθητής» εμφανίζεται η φόρμα συμπλήρωσης του μαθητή. Στην οποία πρέπει να εισάγει σίγουρα τα στοιχεία του ονόματος, επωνύμου και ηλικίας, σε διαφορετική περίπτωση εμφανίζεται μήνυμα λάθους. Σε περίπτωση που πατήσει το 'Go Back' η εφαρμογή μεταβαίνει στο προηγούμενο στάδιο της Αρχικής Εικόνας.



5.1.3 Παράδειγμα Λεξιλογίου

Αφού συμπληρώσει ορθά τα στοιχεία του, ξεκινάει να εμφανίζεται το πρώτο παράδειγμα για τις ερωτήσεις Λεξιλογικού τύπου με αντίστοιχο ήχο και GIF. Στην περίπτωση που πατηθεί το κουμπί 'Explain Again', τότε ξανά παίζει ο ίδιο ήχος και το GIF ξανά από την αρχή. Ενώ εάν πατηθεί το κουμπί 'Okay I got it!' Η αναπαραγωγή ήχου και GIF σταματάει και ξεκινάνε να εμφανίζονται οι ερωτήσεις λεξιλογίου μια-μια.

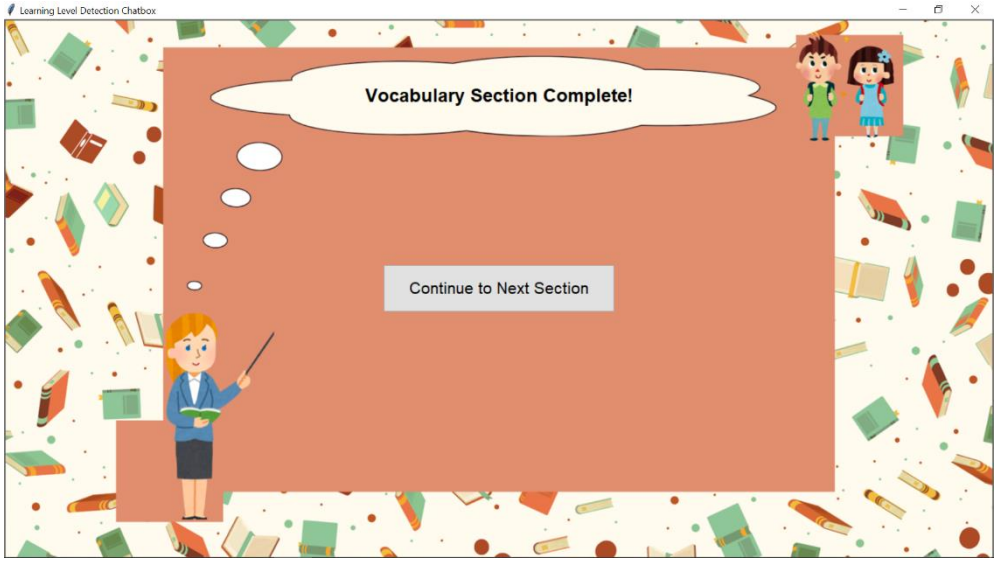


5.1.4 Ερωτήσεις Λεξιλογίου

Αφού ξεκινήσει η δοκιμασία λεξιλογίου, εμφανίζονται οι ερωτήσεις μέσα στο συννεφάκι ομιλίας της δασκάλας και τα κουμπιά που περιέχουν τις δυνατές απαντήσεις. Μέχρι να πατηθεί κάποιο από αυτά η λειτουργία του 'Continue' είναι κλειδωμένη. Με το που πατηθεί κάποια επιλογή, το επιλεγμένο κουμπί αλλάζει περίγραμμα και ενεργοποιείται το κουμπί 'Continue', όπως φαίνεται και την παρακάτω εικόνα.

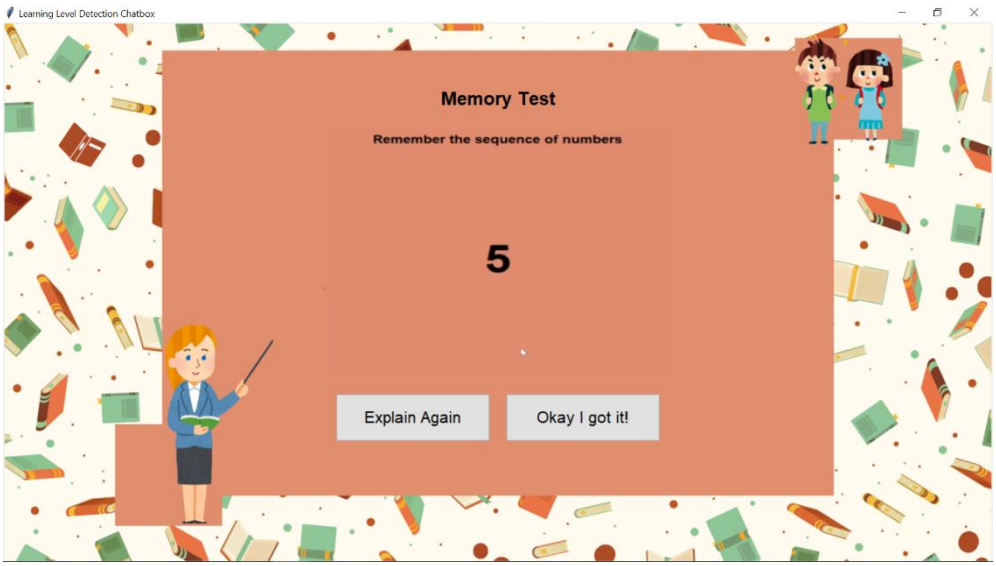
5.1.5 Τέλος Ερωτήσεων Λεξιλογίου

Αφού ολοκληρωθούν οι ερωτήσεις λεξιλογίου, μαθητής βλέπει την παρακάτω εικόνα στην οθόνη του, η οποία σηματοδοτεί το τέλος της συγκεκριμένης διαδικασίας και λειτουργεί ως λογικός διαμεσολαβητής πριν ξεκινήσει το παράδειγμα για την επόμενη δοκιμασία.



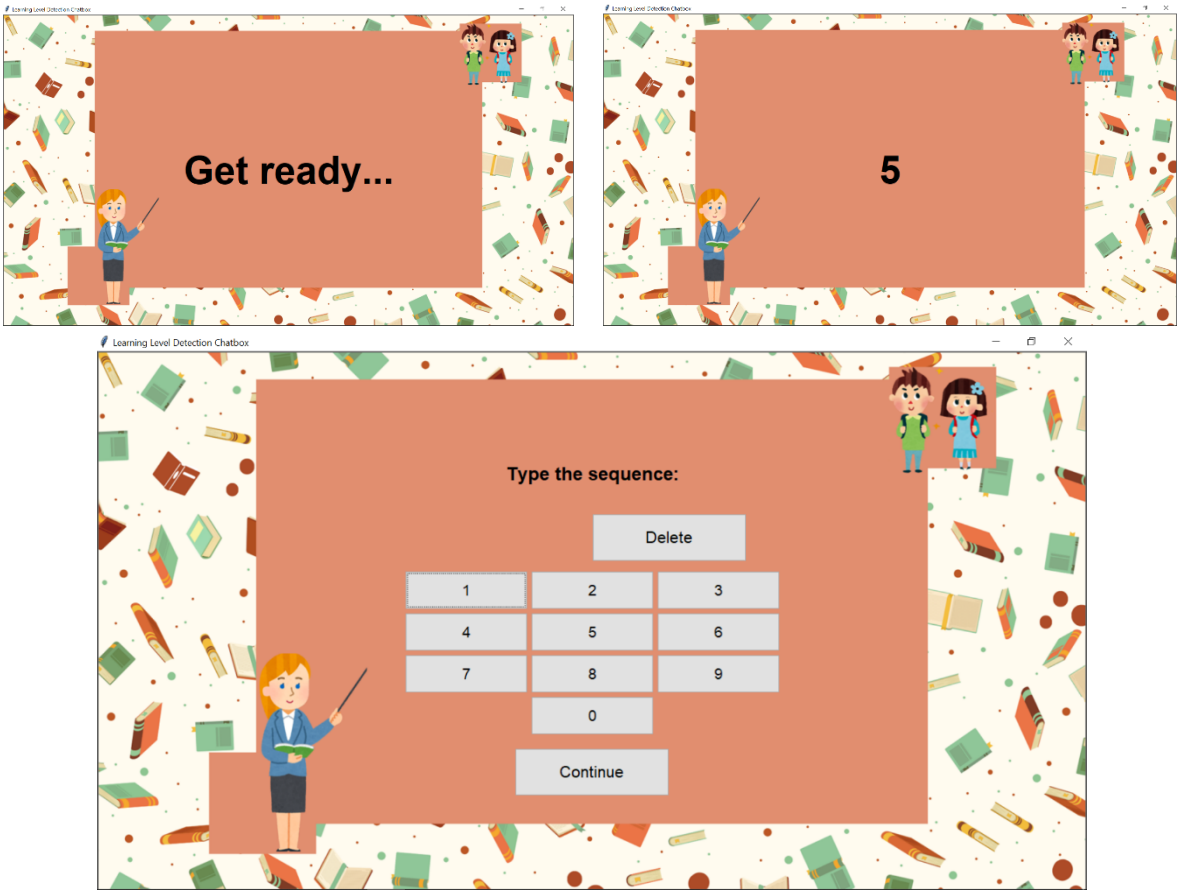
5.1.6 Παράδειγμα Μνήμης

Με το που πατήσει το κουμπί 'Continue to Next Section' ξεκινάει να εκτελείται το παράδειγμα για τη δοκιμασία της μνήμης. Τα κουμπιά της λειτουργούν ακριβώς με τον ίδιο τρόπο όπως και της δοκιμασίας Λεξιλογίου.

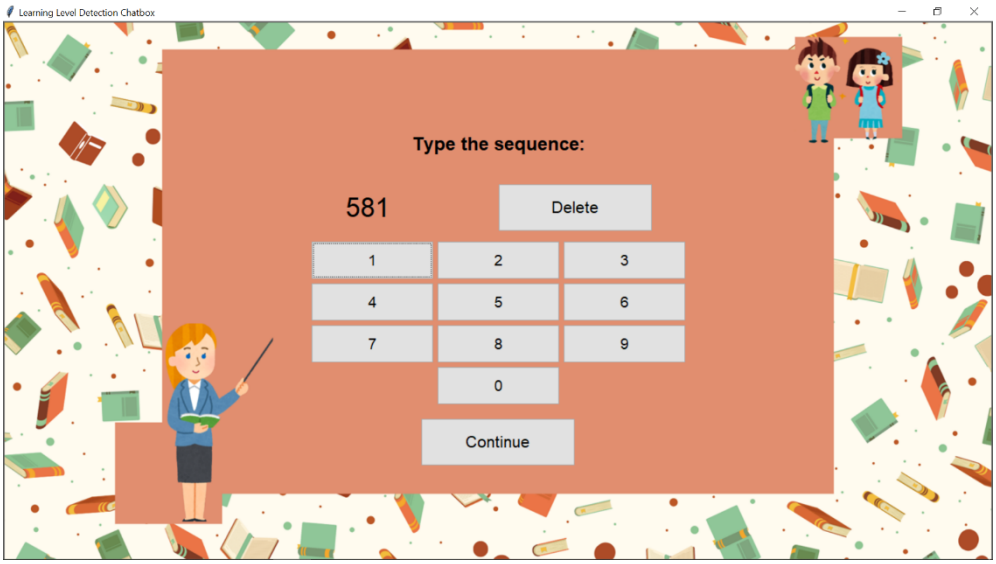


5.1.7 Ερωτήσεις Μνήμης

Με το που πατηθεί το κουμπί 'Okay I got it!' ξεκινάει η λειτουργία του των ερωτήσεων μνήμης όπου αρχικά εμφανίζεται στην οθόνη το κείμενο 'Get Ready..', στη συνέχεια εμφανίζονται ένας-ένας οι αριθμοί της ακολουθίας και μόλις ολοκληρωθεί η εμφάνιση της, ακολουθεί το πινακάκι για την εισαγωγή της ακολουθίας.

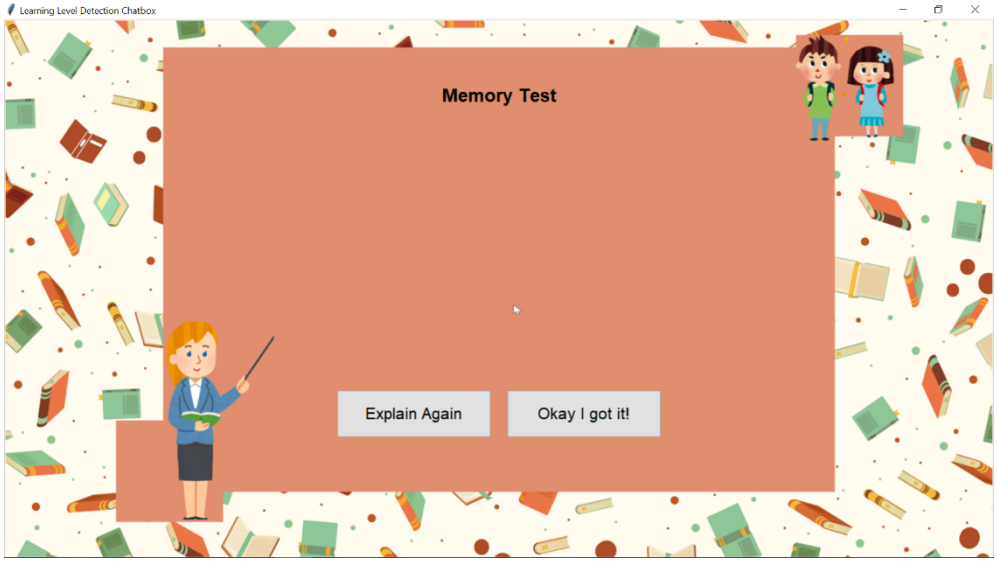


Όταν ο χρήστης πληκτρολογήσει την απάντηση του, η δοκιμασία συνεχίζεται εκ νέου από την αρχή αυτού του βήματος, μέχρις ότου να τερματίσει η διαδικασία.



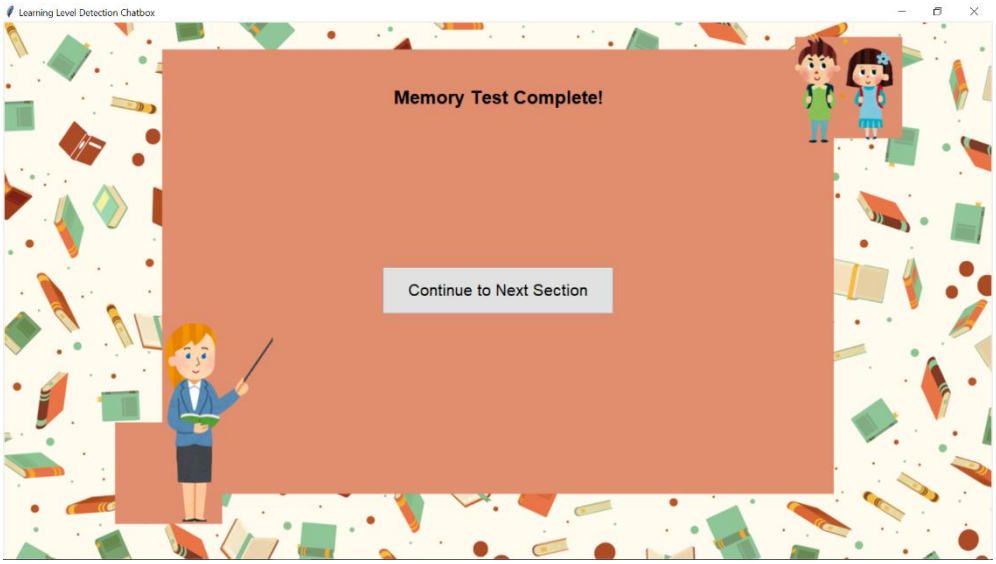
5.1.8 Επόμενο Επίπεδο σε Δοκιμασία Μνήμης

Σε περίπτωση που απαντηθούν 3 σωστές απαντήσεις συνεχόμενά για ένα επίπεδο, τότε η εφαρμογή προχωράει στο επόμενο επίπεδο. Εάν γίνει αυτό, η διαδικασία της δοκιμασίας κάνει μια παύση και δείχνει ενδεικτικό παράδειγμα για το επίπεδο στο οποίο πρόκειται να μπει. Για κάθε επίπεδο ο ήχος και το GIF είναι διαφορετικά καθώς αναπαριστούν παράδειγμα με αντίστοιχα ψηφία ακολουθίας. Στη προκειμένη περίπτωση ο μαθητής βλέπει την παρακάτω εικόνα, κι έπειτα η διαδικασία ξεκινάει από την αρχή, μόνο που αύτη τη φορά οι ακολουθίες είναι κατά ένα ψηφίο μεγαλύτερες.



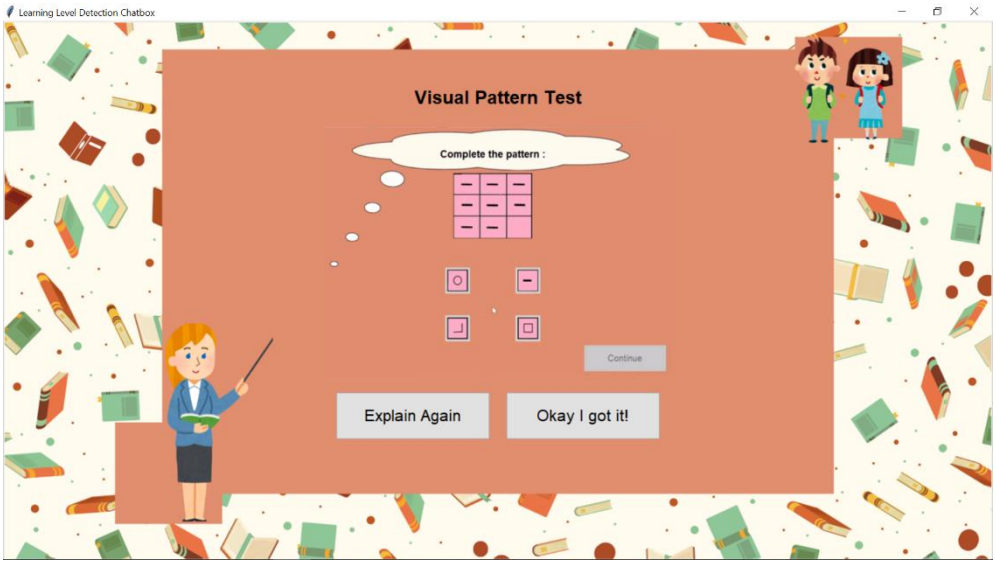
5.1.9 Τέλος δοκιμασίας Μνήμης

Όπως και με τη δοκιμασία του λεξιλογίου έτσι και με της μνήμης με το τέλος της εμφανίζεται σχετικό μήνυμα τερματισμού και κουμπί που προετοιμάζει τον χρήστη για το επόμενο τύπο ερωτήσεων.



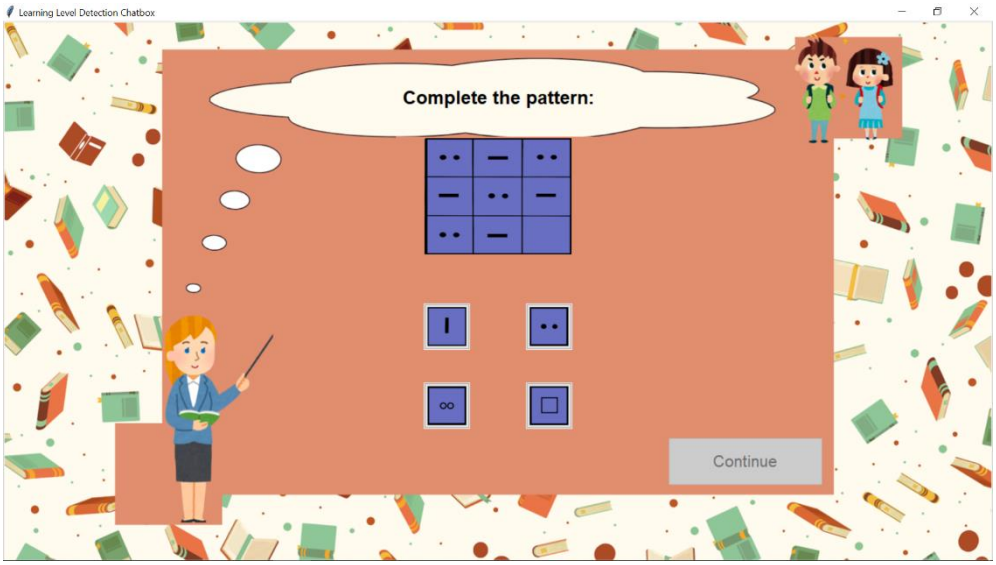
5.1.10 Παράδειγμα Οπτικών Μοτίβων

Προτού ξεκινήσει η διαδικασία των ερωτήσεων με θέμα τα Οπτικά Μοτίβα, εμφανίζεται και πάλι σχετικό παράδειγμα το οποίο εξηγεί στον χρήστη τι απαιτείται να κάνει. Τα κουμπιά λειτουργούν και εδώ ακριβώς τον ίδιο τρόπο όπως και στα προηγούμενα παραδείγματα.



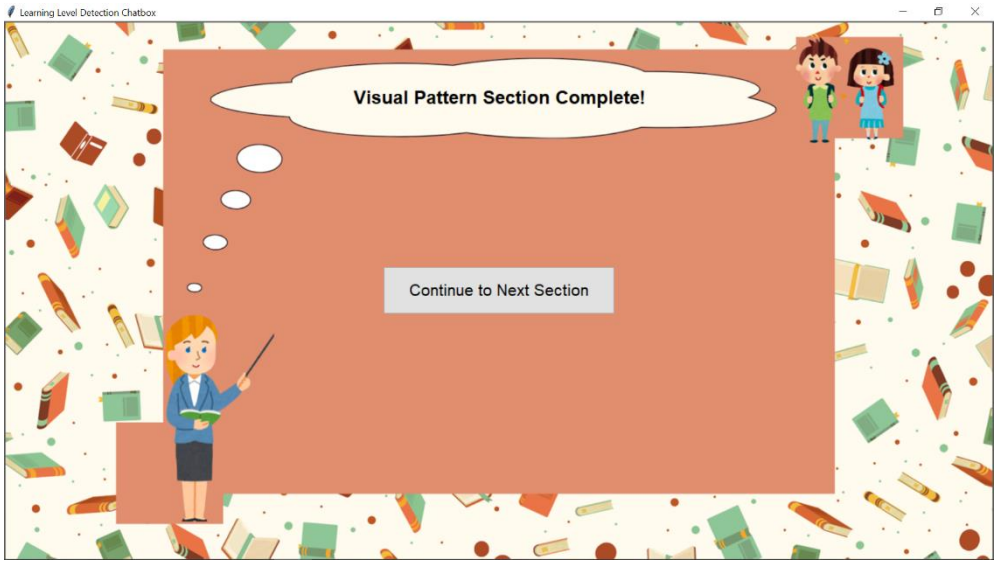
5.1.11 Ερωτήσεις Οπτικών Μοτίβων

Με το πάτημα 'Okay I got it!' Ξεκινούν να εμφανίζονται στην οθόνη μια-μια οι σύνολο 15 ερωτήσεις με οπτικά μοτίβα. Παρακάτω φαίνεται ακριβώς η εμφάνιση της κάθε ερώτησης. Όπως ακριβώς και στις ερωτήσεις λεξιλογίου υπάρχει το κουμπί 'Continue' το οποίο αρχικά είναι απενεργοποιημένο και αναμένει ο χρήστης να πατήσει κάποιο κουμπί απάντησης. Με το που πατήσει κάποιο, αυτό περιβάλλεται από χρωματισμένο πλαίσιο και ενεργοποιείται το κουμπί 'Continue' όπως φαίνεται στις εικόνες.



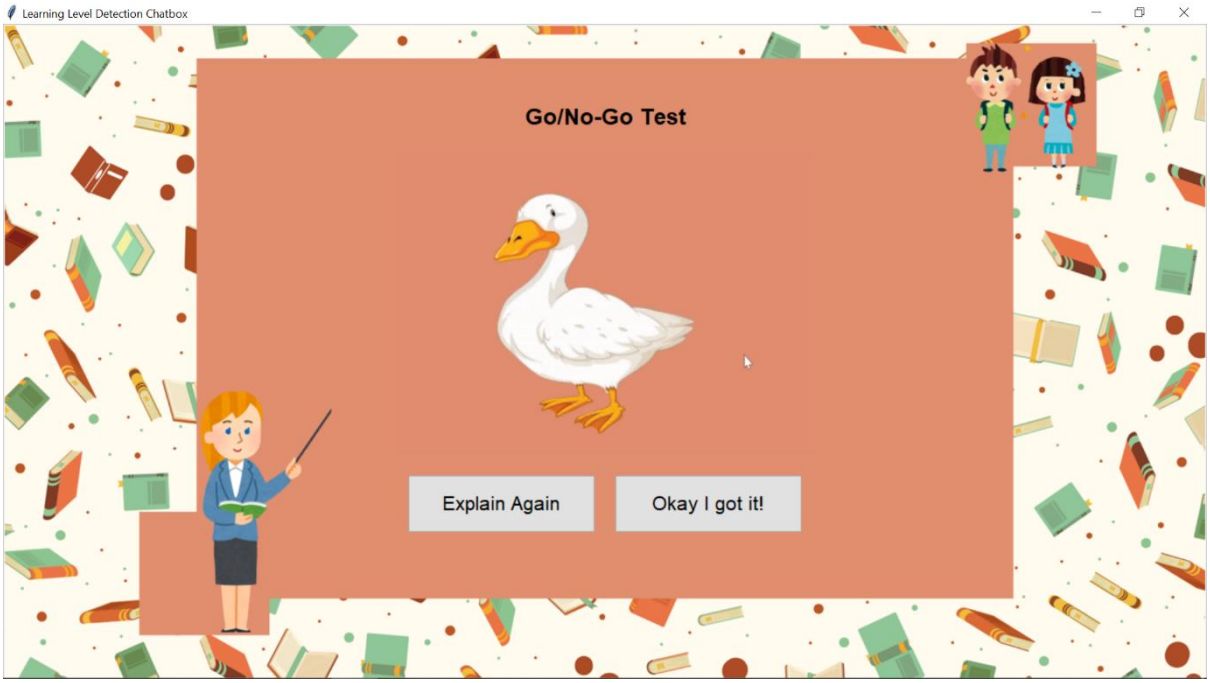
5.1.12 Τέλος Ερωτήσεων Οπτικών Μοτίβων

Μόλις απαντηθούν και οι 15 ερωτήσεις με τα οπτικά μοτίβα από τον μαθητή, εμφανίζεται οθόνη που σηματοδοτεί το τέλος της δοκιμασίας, με το πάτημα του κουμπιού 'Continue to Next Section' ο μαθητής περνάει στην τελευταία δοκιμασία.



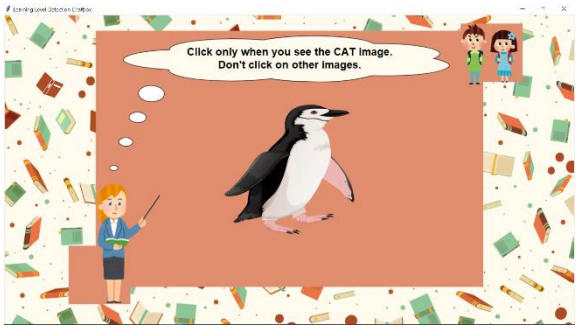
5.1.13 Παράδειγμα Go/No-Go

Προτού, λοιπόν, αρχίζει η δοκιμασία της αναστολής με τη Go/No-Go, εμφανίζεται και πάλι ένα παράδειγμα το οποίο δείχνει πως λειτουργεί η δοκιμασία. Με το πάτημα του κουμπιού 'Okay I got it!' ξεκινάει η διαδικασία της δοκιμασίας.



5.1.14 Δοκιμασία Go/No-Go

Κατά τη δοκιμασία αυτή, δεν εμφανίζονται ερωτήσεις, αλλά με το που πατηθεί το κουμπί για να ξεκινήσει η διαδικασία εμφανίζεται πρώτα ένα μήνυμα 'Get Ready..' κι έπειτα ξεκινάει η προβολή εικόνων από διάφορα ζώα, μέσα από τα οποία ο μαθητής πρέπει να πατήσει επάνω στην γάτα με το ποντίκι του.

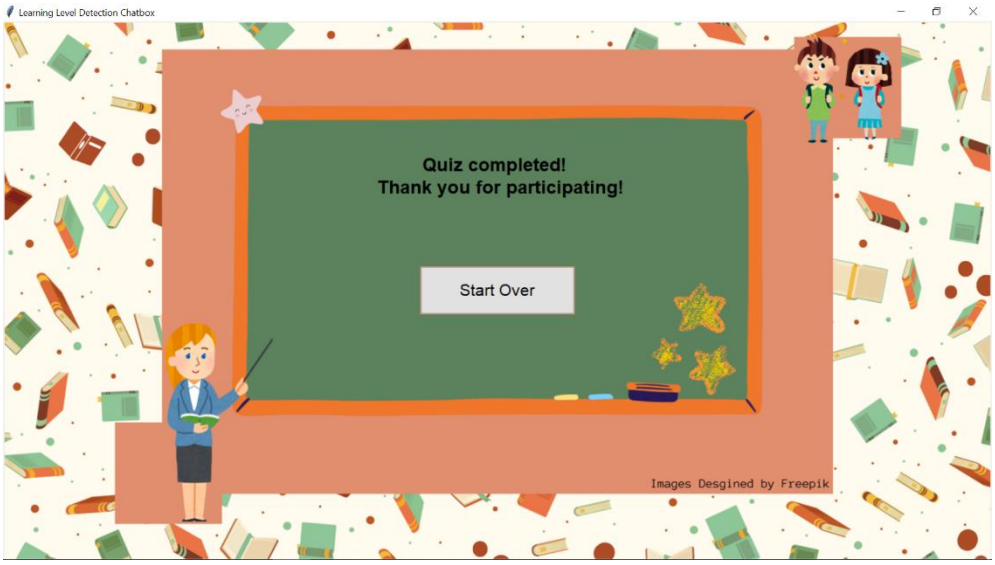


5.1.15 Τέλος δοκιμασίας Go/No-Go

Μόλις τερματίσει η λειτουργία της δοκιμασίας Go/No-Go εμφανίζεται σχετική οθόνη που σηματοδοτεί το τέλος της. Με το πάτημα του κουμπιού ο χρήστης μεταβαίνει στην τελική οθόνη η οποία φαίνεται παρακάτω.

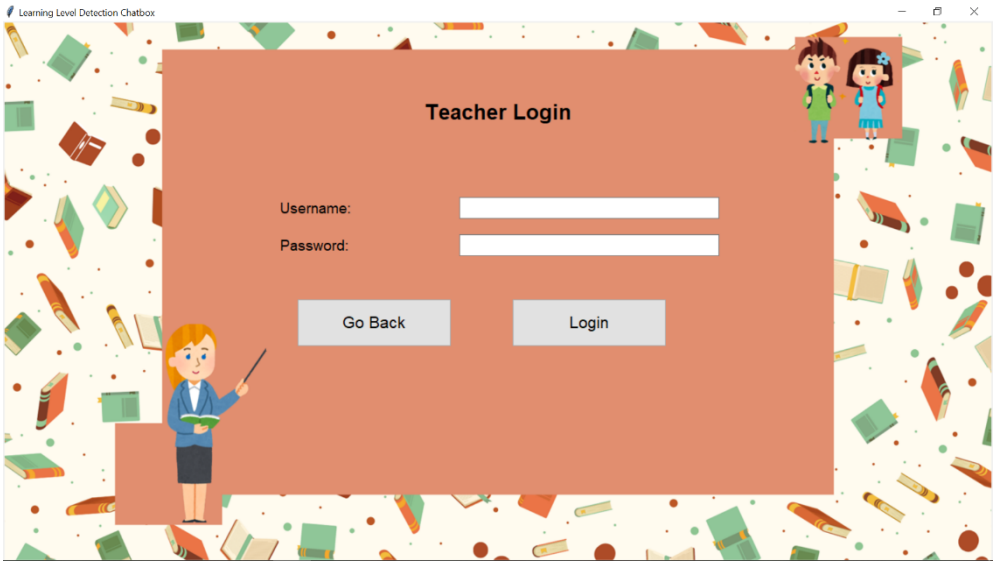
5.1.16 Τέλος δοκιμασιών

Μετά το τέλος των δοκιμασιών, εμφανίζεται η τελική οθόνη της εφαρμογής η οποία εμφανίζει ενδεικτικό μήνυμα στον χρήστη και τον προτρέπει να ξεκινήσει ξανά τη δοκιμασία. Εάν το πατήσει, η ροή επαναφέρεται στην Αρχική οθόνη, από όπου μετά μπορεί είτε να γίνει εισαγωγή για άλλον μαθητή ή να εισέλει ο εκπαιδευτικός για να δει τα αποτελέσματα του μαθητή.



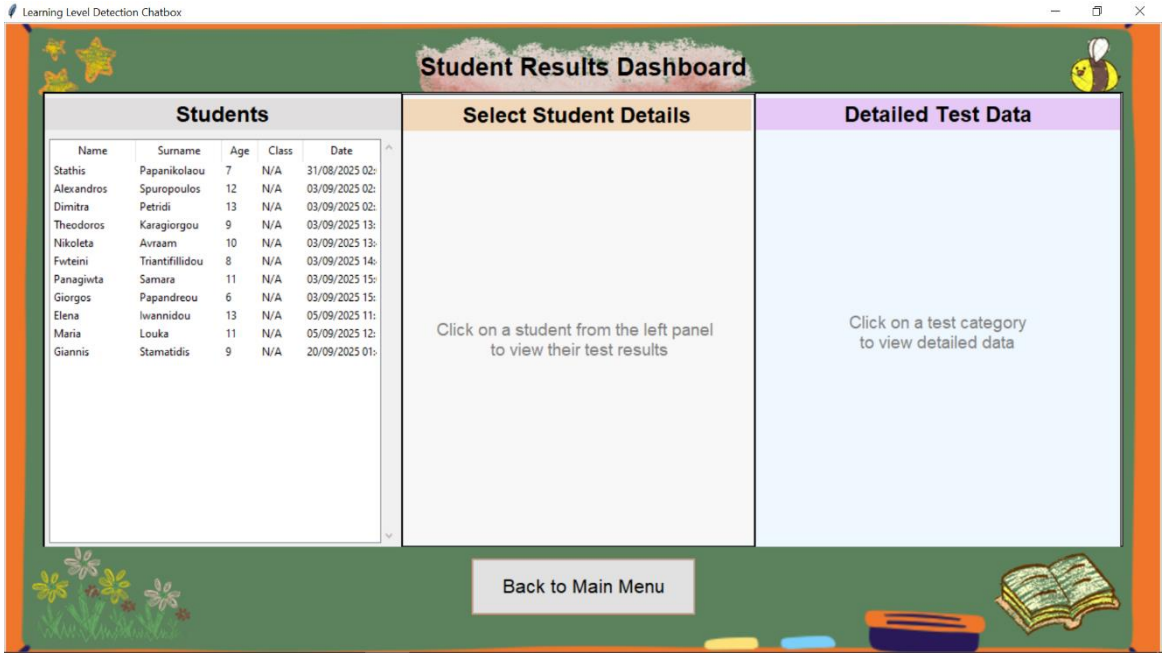
5.1.17 Επίπεδο Εκπαιδευτικού

Από την αρχική οθόνη, εάν πατηθεί το πλήκτρο 'Teacher' ο χρήστης θα έχει την παρακάτω εικόνα, όπου θα πρέπει να βάλει τα στοιχεία εισαγωγής του για να δει τα αποτελέσματα των μαθητών που έχουν δοθεί έως τώρα στην εφαρμογή. Θα πρέπει ο εκπαιδευτικός να δώσει τα σωστά στοιχεία διαφορετικά εμφανίζεται παράθυρο λάθους. Σε περίπτωση που πατηθεί το κουμπί 'Go back' επιστρέφουμε στην αρχική οθόνη της εφαρμογής.

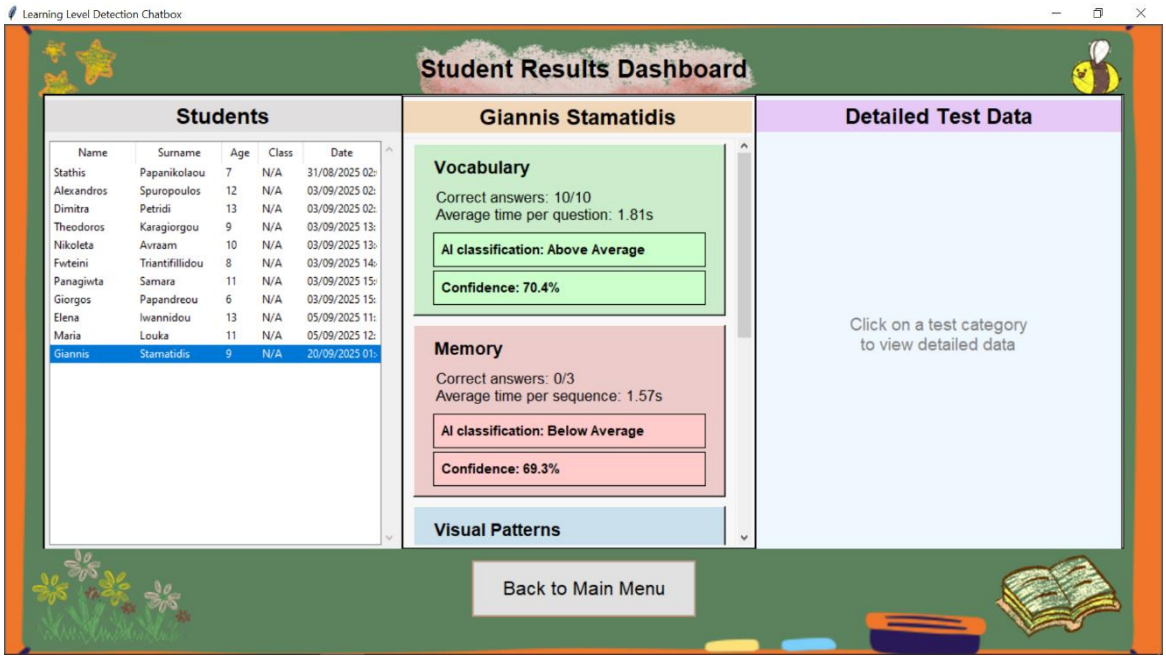


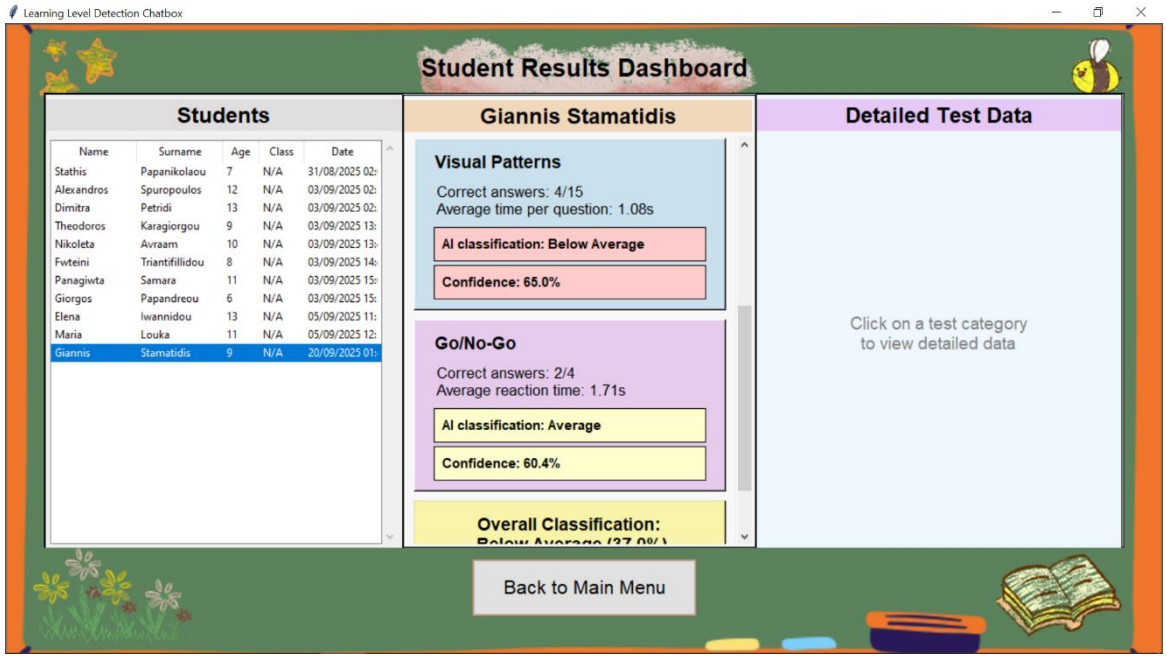
5.1.18 Πίνακας Αποτελεσμάτων

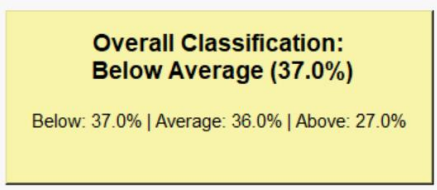
Με το που εισάγει τα σωστά στοιχεία ο εκπαιδευτικός και πατηθεί το κουμπί 'Login' η οθόνη της εφαρμογής έχει την εξής μορφή. Δείχνει ολόκληρο το αριστερό πάνελ με τις έως τώρα εγγραφές μαθητών που έχουν ολοκληρώσει τις δοκιμασίες με τις βασικές τους πληροφορίες δίπλα και το κεντρικό πάνελ δίνει οδηγίες στον χρήστη να πατήσει το πάνελ που βρίσκεται αριστερά.



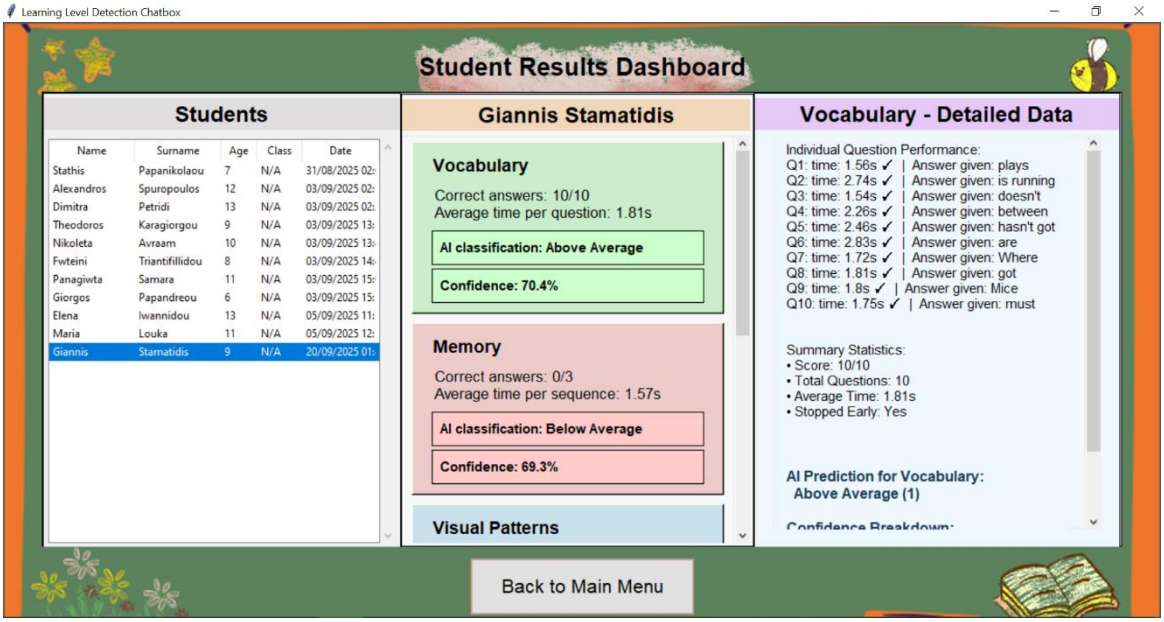
Μόλις πατηθεί κάποια από τις εγγραφές στο αριστερό πάνελ, «ζωντανεύει» η λειτουργία του κεντρικού πάνελ και εμφανίζει τις βασικότερες πληροφορίες για τα αποτελέσματα που έφερε ο μαθητής για κάθε δοκιμασία.



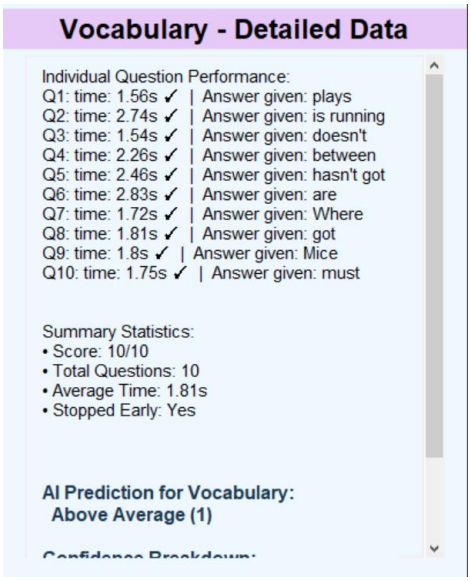


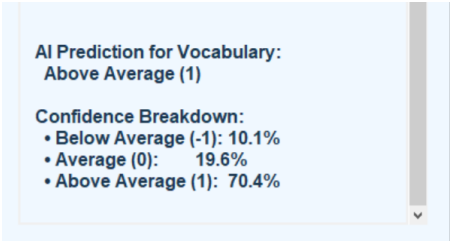


Εάν πατηθεί κάθε ένα από τα χρωματιστά μπλοκ πέρα του κίτρινου που δείχνει το τελικό αποτέλεσμα για τον μαθητή, ενεργοποιείται το δεξί πάνελ που δείχνει μεμονωμένα λεπτομέρειες για το κάθε κουΐζ. Όπως φαίνεται παρακάτω.

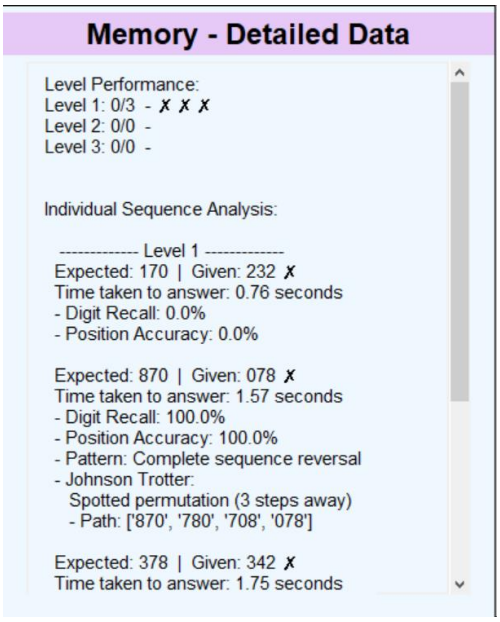


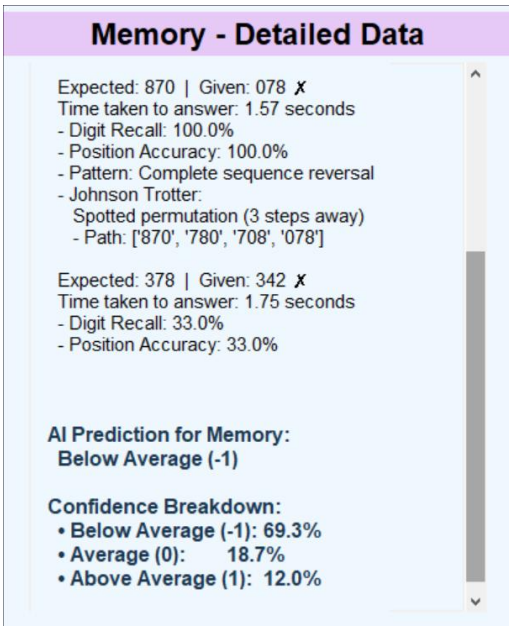
Για την κάθε δοκιμασία μεμονωμένα το δεξί πάνελ λειτουργεί διαφορετικά, για παράδειγμα για το λεξιλόγιο φαίνονται αναλυτικά οι απαντήσεις του μαθητή και κάποια ακόμη στατιστικά στοιχεία και στο τέλος δίνεται το αποτέλεσμα του μοντέλου μηχανικής μάθησης. Παρακάτω φαίνεται ενδεικτικό παράδειγμα.



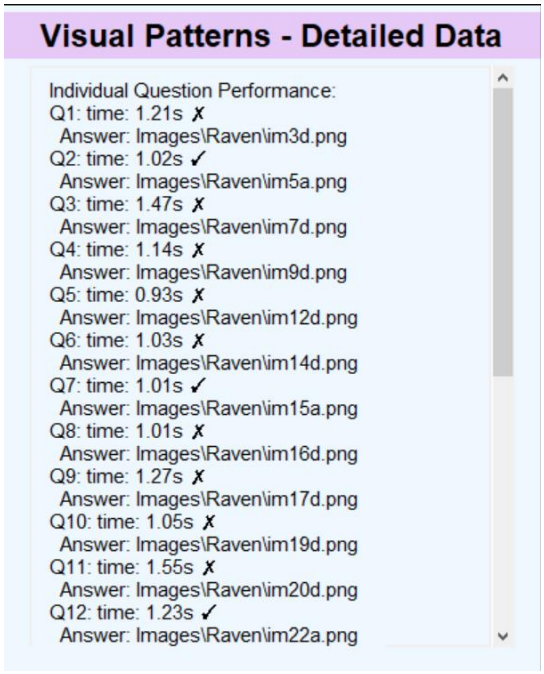


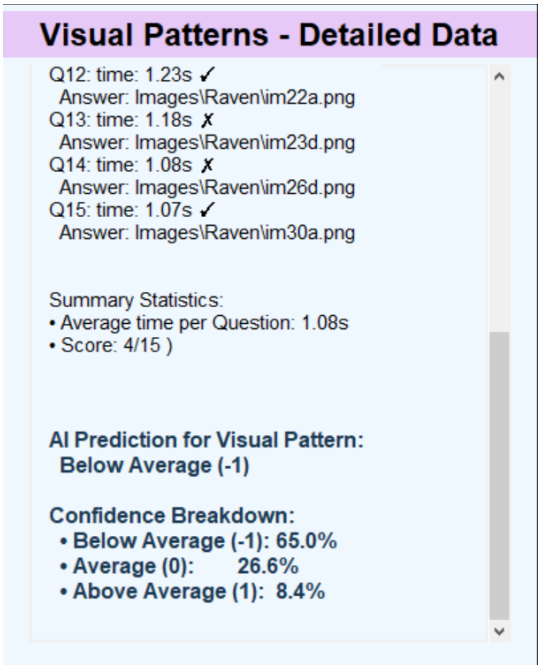
Για δοκιμασίες Μνήμης το πάνελ έχει την παρακάτω μορφή συνήθως. Περιέχει περισσότερα στοιχεία από ότι άλλα πάνελ συνήθως διότι αναλύει κάθε ακολουθία μεμονωμένα για αρκετά χαρακτηριστικά. Βλέπουμε και πως μέσα στην εφαρμογή φαίνεται η εμφάνιση των αποτελεσμάτων που εξάγουμε από τον αλγόριθμο Johnson_Trotter.



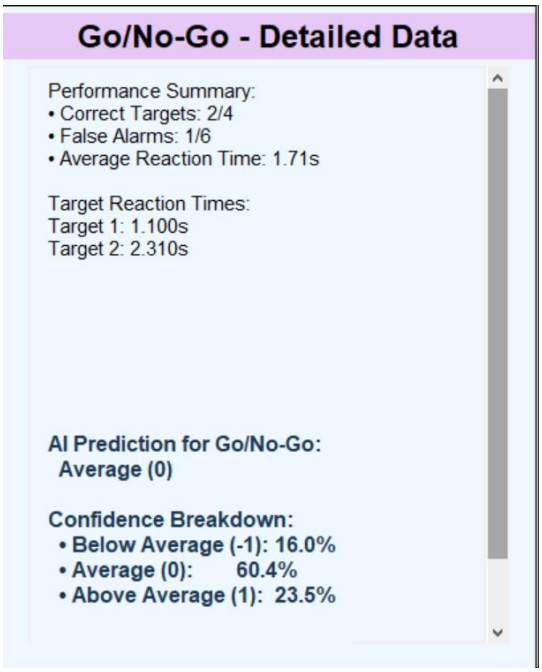


Για δοκιμασίες Οπτικών Μοτίβων το δεξί πάνελ έχει την παρακάτω μορφή συνήθως. Φαίνονται δηλαδή αναλυτικά οι απαντήσεις που έδωσε ο μαθητής αλλά και εάν αυτές ήταν σωστές ή όχι και στο κάτω μέρος αναλύεται πάλι το κομμάτι του κατηγοριοποιητή για τη παρούσα δοκιμασία.





Για τη δοκιμασία της Go/No-Go το πάνελ συνήθως δεν περιέχει πολλές πληροφορίες, καθώς ως δοκιμασία δεν συλλέγει πολλές πληροφορίες και όσες συλλέγει είναι μικρού μεγέθους. Στο κάτω μέρος της, όπως και για τις υπόλοιπες, φαίνονται να αποτελέσματα του κατηγοριοποιητή της.



5.2 Μελλοντικές Προεκτάσεις

Υπάρχουν αρκετές βελτιώσεις που θα μπορούσαν να γίνουν στο πλαίσιο της εφαρμογής. Μια από αυτές θα μπορούσε να ήταν η χρήση πραγματικών δεδομένων για την εκπαίδευση των μοντέλων αντί για τεχνητά, καθώς κάτι τέτοιοι θα προσέδιδε περισσότερη αξιοπιστία στην εφαρμογή. Η ενέργεια αυτή ενδεχομένως να αύξανε και την ακρίβεια των μοντέλων μηχανικής μάθησης που χρησιμοποιούνται. Σε επίπεδο κώδικα, θα μπορούσε να εφαρμοστεί η ιδέα της επαναχρησιμοποίησης στοιχείων σε ακόμη περισσότερα στοιχεία widgets πέρα από αυτά των δοκιμασιών, προκειμένου να δημιουργούνται και να διαγράφονται επανειλημμένα τα ίδια στοιχεία.

Για να εμπλουτιστεί η διαγνωστικότητα του εργαλείου, θα μπορούσε να προστεθεί μια μεγαλύτερη ποικιλία γνωστικών δοκιμασιών που θα αξιολογούν και άλλες λειτουργίες, όπως η ακουστική μνήμη. Θα μπορούσε, ακόμη, θα προστεθεί λειτουργία προσαρμογής δυσκολίας για τον μαθητή, όπου σε πραγματικό χρόνο η εφαρμογή θα ρυθμίσει αυτόματα το επίπεδο των ερωτήσεων αναλόγως με τις επιδόσεις του. Τέλος, για να ενισχυθεί η αίσθηση της παιχνιδοποίησης της εφαρμογής, θα μπορούσε να προστεθούν περισσότερα ηχητικά εφέ, όπως ένας διακριτικός ήχος για το κλικ ή ένας ήχος επιβράβευσης στο τέλος των δοκιμασιών.

Βιβλιογραφία

Wikipedia contributors . (2025, July 29). *Xeropan*. Ανάκτηση από In Wikipedia, The Free Encyclopedia: Retrieved 14:11, September 12, 2025, from https://en.wikipedia.org/w/index.php?title=Xeropan&oldid=1303251543

Alnuaimi, A. F. (2024, EDP Sciences.). An overview of machine learning classification techniques. *In BIO Web of Conferences (Vol. 97, p. 00133)*.

Ayodele, T. O. (2010). Types of machine learning algorithms. *New advances in machine learning, 3(19-48), 5-1*.

Bogomolny, A. (2018). *Johnson-Trotter Algorithm*. Ανάκτηση από Cut The Knot : https://www.cut-the-knot.org/Curriculum/Combinatorics/JohnsonTrotter.shtml?

Buchanan, B. G. (2005). A (Very) Brief History of Artificial Intelligence. *AI Magazine*.

Crystal, D. (2003). *English as a global language*. Cambridge university press.

Crystal, D. (2018). *The Cambridge encyclopedia of the English language*. Cambridge university press.

Crystal, D., Potter, S. (2025, July 31). *English language*. Ανάκτηση από Encyclopedia Britannica: https://www.britannica.com/topic/English-language

Daskalaki, E. C. (2019). Input effects across domains: The case of Greek subjects in child heritage language. *Second Language Research*.

Diamond, A. (2013). *Executive functions. Annual review of psychology, 64, 135–168*. Ανάκτηση από https://doi.org/10.1146/annurev-psych-113011-143750

Douglas Biber, Stig Johansson, Geoffrey Leech, Susan Conrad, & Edward Finegan. (1999). *Grammar of Spoken and Written English*. Longman.

Dünya Baradari, N. K. (2025, March 10). *NeuroChat: A Neuroadaptive AI Chatbot for Customizing Learning Experiences*. Ανάκτηση από Cornell University: https://arxiv.org/abs/2503.07599

French, R. M. (2000). The Turing Test: the first 50 years. *Trends in cognitive sciences*.

Geeks for Geeks, a. (2025). *Semi Supervised Learning Examples*. Ανάκτηση από Geeks for Geeks: https://www.geeksforgeeks.org/machine-learning/semi-supervised-learning-examples/

Halkiopoulos, C. &. (2024). *Leveraging AI in E-Learning: Personalized Learning and Adaptive Assessment through Cognitive Neuropsychology—A Systematic Analysis*. . Ανάκτηση από Electronics, 13(18), 3762. : https://doi.org/10.3390/electronics13183762

Haqnawaz, H., Naeem, N., & Khan, S. (2024). *Complexities of English: A study of grammar, vocabulary, and pronunciation*. Kashf Journal of Multidisciplinary Research.

Holton, D. M.-W. (2004). *Greek: An Essential Grammar of*. Routledge.

Jacob Murel, E. K. (2024, January 19). *What is a confusion matrix?* Ανάκτηση από IBM: https://www.ibm.com/think/topics/confusion-matrix

Jordan, M. I. & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*.

Kotsiantis, S. B. (2007, July 16). Supervised Machine Learning: A Review of Classification Techniques. *Department of Computer Science and Technology*.

Linck, J. &. (2015). Can Working Memory and Inhibitory Control Predict Second Language Learning in the Classroom? *Sage Open*.

Linck, J. A. (2014). Working memory and second language comprehension and production: a meta-analysis. *Psychonomic bulletin & review*.

Lundh, F. (1999). *An Introduction to Tkinter*.

Malikouti-Drachman, A. L. (2025, August 2). *Greek language*. Ανάκτηση από Encyclopedia Britannica: https://www.britannica.com/topic/Greek-language

McCarthy, J. (2007). What is artificial intelligence.

Puchta, H. (2020). Cognitive control functions in language learning. *Part of the Cambridge Papers in ELT seires Cambridge: Cambridge University Press*.
S. Gkika, N. Z. (2022). WEB-ASSESSING FOREIGN LANGUAGE ACQUISITION OF MIXED ABILITY PRIMARY SCHOOL PUPILS TO ENHANCE DIFFERENTIATED FLIPPED BLENDED LEARNING. *ICERI2022 Proceedings*.
Sah, S. (2020). Machine learning: a review of learning types.
Ware, C. (2013). *Information Visualization (Third Edition).*
Wikipedia contributors. (2025, May 3). *Loqu8.* Ανάκτηση από Wikipedia, The Free Encyclopedia.: Retrieved 14:42, September 12, 2025, from https://en.wikipedia.org/w/index.php?title=Loqu8&oldid=1288508161
Wikipedia contributors. (2025, August 19). *Probabilistic classification.* Ανάκτηση από Wikipedia, The Free Encyclopedia: Retrieved 17:20, September 17, 2025, from https://en.wikipedia.org/w/index.php?title=Probabilistic_classification&oldid=1306782816
Wikipedia contributors. (2025, June 12). *SuperMemo.* Ανάκτηση από Wikipedia, The Free Encyclopedia: Retrieved 13:31, September 12, 2025, from https://en.wikipedia.org/w/index.php?title=SuperMemo&oldid=1295252811
Wikipedia contributors. (2024, July 27 Retrieved 10:16, September 10, 2025,). *Visuospatial function.* Ανάκτηση από In Wikipedia, The Free Encyclopedia.: from https://en.wikipedia.org/w/index.php?title=Visuospatial_function&oldid=1236997383
Wikipedia contributors. (2025, May 11). *teinhaus–Johnson–Trotter algorithm.* Ανάκτηση από Wikipedia, The Free Encyclopedia: Retrieved 16:54, September 17, 2025, from https://en.wikipedia.org/w/index.php?title=Steinhaus%E2%80%93Johnson%E2%80%93Trotter_algorithm&oldid=1289873329
Wikipedia contributors. (2025, June 24). *Tkinter* . Ανάκτηση από Wikipedia, The Free Encyclopedia: Retrieved 23:44, September 16, 2025, from https://en.wikipedia.org/w/index.php?title=Tkinter&oldid=1297130115
Wilschut Thomas, S. F. (2021). *Benefits of Adaptive Learning Transfer From Typing-Based Learning to Speech-Based Learning.* Ανάκτηση από Frontiers in Artificial Intelligence: https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2021.780131
Wright, C., & Wang, J. . (2023). he role of visual processing in learning Mandarin characters. *Journal of the European Second Language Association, 7(1), 31–45.*
Yuan, H. (2025). *Artificial intelligence in language learning: biometric feedback and adaptive reading for improved comprehension and reduced anxiety.* . Ανάκτηση από Humanit Soc Sci Commun 12, 556: https://doi.org/10.1057/s41599-025-04878-w
Zygouris, N. C. (2025). Validation of the Askisi-Lexia neuropsychological web-based screener: A neuropsychological battery for screening cognitive and phonological skills of children with dyslexia. *Applied Neuropsychology: Child, 1-17.*
Βελούδης, Γ. (2001). *Ιστορίες της Ελληνικής γλώσσας.* Ανάκτηση από Η πύλη για την ελληνικη γλώσσα: https://share.google/RzjiZc5axkk4OOTNb
Βικιπαίδεια. (2025, Μαρτίου 24). *Python.* Ανάκτηση από Η Ελεύθερη Εγκυκλοπαίδεια: Ανακτήθηκε 23:30, Σεπτεμβρίου 16, 2025 από το //el.wikipedia.org/w/index.php?title=Python&oldid=11023389.

Εικόνες

Εικόνα 2.3: Εργαζόμενη Μνήμη:
https://learnfully.com/the-role-of-working-memory-in-cognitive-development-everyday-learning-and-academic-performance/

Εικόνα 2.4: Online Learning:
https://managedservicesjournal.com/market-trends/education/7-education-trends-that-are-driving-tech-implementations/

Εικόνα 3.1 'The Turk' chess machine:
https://timelessmoon.getarchive.net/amp/media/racknitz-the-turk-3-b4d6d2

Εικόνα 3.2.3: Επιβλεπόμενη και μη Επιβλεπόμενη Μάθηση:
https://commons.wikimedia.org/wiki/File:Machin_learning.png

Εικόνα 3.2.4.1: Δέντρα Αποφάσεων – Decision Trees:
https://commons.wikimedia.org/wiki/File:Decision_Tree_Depth_2.png

Εικόνα 3.2.4.2: Τυχαίο Δάσος – Random Forest:
https://commons.wikimedia.org/wiki/File:Random_Forest_Diagram_with_Four_Trees.png

Εικόνα 3.2.4.3: Λογιστική Παλινδρόμηση – Logistic Regression:
https://commons.wikimedia.org/wiki/File:Exam_pass_logistic_curve.svg

Εικόνα 3.2.4.4: Κ-Πλησιέστεροι Γείτονες – KNN:
https://commons.wikimedia.org/wiki/File:K_nearest_neighbour_explain.png

Εικόνα 3.2.4.5: Μηχανές Υποστήριξης Διανυσμάτων – SVM:
https://commons.wikimedia.org/wiki/File:SVM_explain.png

Εικόνα 3.2.5.2: Confusion Matrix:
https://commons.wikimedia.org/wiki/File:Confusion_matrix_with_accuracy_metrics_for_land_cover.jpg