ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

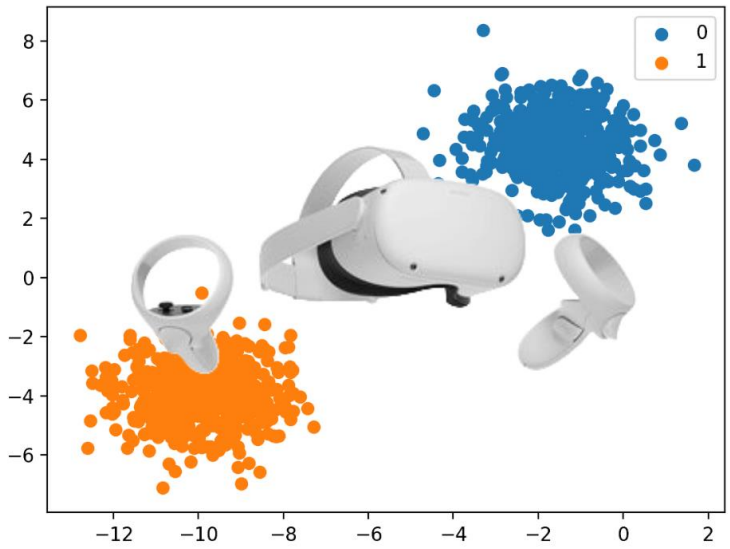
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΕΥΦΥΕΙΣ
ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEB INTELLIGENCE

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μοντέλο Μηχανικής Μάθησης για την κίνηση χειριστηρίων σε παιχνίδι Εικονικής Πραγματικότητας



Της
Σταυρουλάκη Στυλιανής
Αρ. Μητρώου: 13/2022

Επιβλέπων
Κεραμόπουλος Ευκλείδης
Καθηγητής, ΔΙ.ΠΑ.Ε.

Θεσσαλονίκη, Φεβρουάριος 2024

Μοντέλο Μηχανικής Μάθησης για την κίνηση χειριστηρίων σε παιχνίδι Εικονικής Πραγματικότητας

Κωδικός Δ.Ε. 23293

Ονοματεπώνυμο φοιτήτριας: Σταυρουλάκη Στυλιανή

Ονοματεπώνυμο εισηγητή: Κεραμόπουλος Ευκλείδης

Ημερομηνία ανάληψης Δ.Ε.: 30/10/2023

Ημερομηνία περάτωσης Δ.Ε. 17/02/2024

Βεβαιώνω ότι είμαι η συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Η Διπλωματική εργασία αυτή επιλέχθηκε με σκοπό την περεταίρω εξοικείωση με τους αλγορίθμους Μηχανικής Μάθησης και των τεχνολογιών Εικονικής Πραγματικότητας. Η επαγγελματική κλίση της ερευνήτριας είναι η ανάπτυξη βίντεο-παιχνιδιών και εικονικών εμπειριών, για κονσόλες παιχνιδιών και εικονικής/επαυξημένης πραγματικότητας. Κατά τη διάρκεια του μεταπτυχιακού ηγέρθη μεγάλο ενδιαφέρον για τις τεχνολογίες Μηχανικής Μάθησης και εκπαίδευσης Νευρωνικών Δικτύων. Η εργασία επιλέχθηκε για να ενώσει τους δύο αυτούς τομείς ενδιαφέροντος, και να συμβάλλει στην περεταίρω εκπαίδευση της ερευνήτριας πάνω σε αυτές τις τεχνολογίες.

Περίληψη

Οι αλγόριθμοι Μηχανικής Μάθησης (ΜΜ) και οι τεχνολογίες Εικονικής Πραγματικότητας (VR) λαμβάνουν αυξανόμενη προσοχή τα τελευταία χρόνια στον τομέα της Πληροφορικής. Η Μηχανική Μάθηση προσφέρει λύσεις σε περίπλοκα προβλήματα που απαιτούν μεγάλη υπολογιστική ισχύ. Χρησιμοποιείται ευρεία στην πρόβλεψη αποτελεσμάτων, δοσμένων κάποιων μεγάλων σετ δεδομένων. Η Εικονική Πραγματικότητα συμβάλλει τόσο στην ψυχαγωγία, όσο και σε άλλους τομείς όπως η βιομηχανία και η εκπαίδευση.

Στην παρούσα εργασία αναλύονται τεχνικές που αφορούν τις παραπάνω τεχνολογίες, και αναπτύσσονται μοντέλα ΜΜ για την πρόβλεψη κινήσεων με χειριστήρια συσκευών VR σε ένα παιχνίδι. Συγκρίνονται αλγόριθμοι ΜΜ αλλά και Νευρωνικών Δικτύων, δημιουργούνται μοντέλα για κάθε ένα από αυτά, και συγκρίνεται η απόδοσή τους όσον αφορά το πρόβλημα της δυαδικής ταξινόμησης.

Για τις ανάγκες της εργασίας αναπτύχθηκε ένα παιχνίδι στη μηχανή Unity με C#, το οποίο χρησιμοποιήθηκε τόσο για τη συλλογή των δεδομένων εκπαίδευσης των μοντέλων, όσο και για την επίδειξη των δυνατοτήτων του μετά την εκπαίδευση. Η εκπαίδευση πραγματοποιήθηκε με τη γλώσσα Python και αρκετές βιβλιοθήκες της, συμπεριλαμβανομένης της Keras, για Νευρωνικά Δίκτυα.

Το αποτέλεσμα της εργασίας ήταν πολλά εκπαιδευμένα μοντέλα με υψηλή ακρίβεια απόδοσης (μέχρι και 92.5% σε νέα δεδομένα), και ένα παιχνίδι στο Unity για VR κονσόλες.

Λέξεις κλειδιά: Αλγόριθμοι Μηχανικής Μάθησης, Εικονική Πραγματικότητα, Νευρωνικά Δίκτυα, Εκπαίδευση μοντέλων, Δυαδική ταξινόμηση, Unity.

«Machine Learning model for controller movement in a Virtual Reality game»

«Stella Stavroulaki»

Abstract

Machine Learning (ML) algorithms and Virtual Reality (VR) technologies have received increasing attention in recent years in the field of IT. Machine Learning offers solutions to complex problems that require large computational power. It is widely used in predicting outcomes given some large data sets. Virtual Reality contributes to both entertainment and other fields such as industry and education.

In this paper, we analyze techniques related to these technologies, and develop MM models for predicting movements with VR device controllers in a game. Both MM and Neural Network algorithms are compared, models are created for each of them, and their performance with respect to the binary classification problem is compared.

For the purposes of the paper, a game was developed in the Unity engine using C#, which was used both to collect the model training data and to demonstrate its capabilities after training. The training was performed using the Python language and several of its libraries, including Keras, for Neural Networks.

The result of the project was several trained models with high performance accuracy (up to 92.5% on new data), and a game in Unity for VR consoles.

Key words: Machine Learning algorithms (ML), Virtual Reality (VR), Neural Networks, Model training, Binary classification, Unity engine.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους κοντινούς μου ανθρώπους που με έχουν στηρίξει καθολικά σε όλη τη διάρκεια των σπουδών μου, και πιο συγκεκριμένα τη μητέρα μου και τον σύντροφό μου. Τους καθηγητές μου για την υποστήριξή τους, και πιο συγκεκριμένα τον επιβλέπων μου Ευκλείδη Κεραμόπουλο, για την εμπιστοσύνη, τη στήριξη, και τη συμβολή του. Τέλος, τις κοινότητες ανοικτών δεδομένων και πληροφορίας του διαδικτύου που συμβάλλουν καθημερινά στην ανάπτυξη και προώθηση ελεύθερης γνώσης.

Πίνακας Περιεχομένων

Συντομογραφίες

ΜΜ	Μηχανική Μάθηση
ΝΔ	Νευρωνικό Δίκτυο
ANN	Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks)
VR	Εικονική Πραγματικότητα (Virtual Reality)
AR	Επαυξημένη Πραγματικότητα (Augmented Reality)
XR	Εκτεταμένη Πραγματικότητα (Extended Reality)
MLP	Perceptron Πολλαπλών στρωμάτων (Multi-Layer Perceptron)
SVM	Support Vector Machines
KNN	Κ εγγύτεροι γείτονες (K-nearest neighbors)
LSTM	Δίκτυα Μακράς Βραχύχρονης Μνήμης (Long Short-term Memory)
ReLU	Rectified Linear Unit
CNN	Συνελικτικό Νευρωνικό Δίκτυο (Convolutional Neural Network)
RNN	Αναδρομικό Νευρωνικό Δίκτυο (Recurrent Neural Network)
LOO	Leave-One-Out

1

Εισαγωγή

Οι τεχνολογίες Εικονικής Πραγματικότητας (Virtual Reality – VR) και οι αλγόριθμοι Μηχανικής Μάθησης (ΜΜ) αναπτύσσονται ραγδαία κατά τα τελευταία χρόνια. Πρόκειται για δύο έννοιες εν γένει καινοτόμες που αποτελούν την αιχμή του δόρατος στον τομέα της Πληροφορικής και των Ευφυών Συστημάτων. Ο συνδυασμός τους ενισχύει το ενδιαφέρον των ίδιων των θεμάτων και αποτελεί πρόκληση για την επιστημονική κοινότητα, μιας και αυξάνει την ήδη μεγάλη πολυπλοκότητα της ανάπτυξης τέτοιων συστημάτων.

1.1 Ιστορική αναδρομή

Οι τεχνολογίες με κράνη VR (headsets) ξεκίνησαν να αναπτύσσονται το 1962 [1], όμως πήραν τη μορφή με την οποία τα ξέρουμε σήμερα το 1993 με την έκδοση του headset από την εταιρία κονσόλων παιχνιδιού SEGA. Το 2012 ξεκίνησε η ευρείας κατανάλωσης εμπορική παραγωγή των VR Headsets και έκτοτε, μεγάλες εταιρίες ανταγωνίζονται για μια θέση στα σπίτια των καταναλωτών, σχεδιάζοντας και αναπτύσσοντας καλύτερα headsets.

Η ΜΜ ξεκίνησε με την ανάπτυξη του Perceptron το 1957 [2], μια μηχανή που μετέτρεπε αναλογικά σήματα σε διακριτά, χρησιμοποιώντας και αναλογικά και διακριτά σήματα ως εισόδους. Αποτέλεσε το πρότυπο για τα μοντέρνα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks – ANN) και το μοντέλο μάθησης που χρησιμοποιούσε ήταν κοντά σε αυτό που χρησιμοποιούν οι άνθρωποι και τα ζώα. Η έννοια συνέχισε να αναπτύσσεται μέσα στις δεκαετίες, με διάφορες προσεγγίσεις, όντας πάντα αλληλένδετο με τα μαθηματικά, την πληροφορική, και φυσικά την Τεχνητή Νοημοσύνη (Artificial Intelligence – AI). Με το ξεκίνημα του 21^{ου} αιώνα, μαζί με τη ραγδαία άνοδο της τεχνολογίας των υπολογιστών, η ΜΜ έγινε το επίκεντρο ενδιαφέροντος για μεγάλη μερίδα της επιστημονικής κοινότητας, και τα τελευταία χρόνια αποτελεί αναπόσπαστο κομμάτι της πληροφορικής.

1.2 Αντικείμενο διπλωματικής

Η εργασία αυτή ασχολείται με τη μελέτη και εκπαίδευση μοντέλων ΜΜ πάνω σε αριθμητικά δεδομένα, παραγόμενα από χειριστήρια συσκευών VR. Σκοπός είναι να μελετηθεί η ένωση των δύο αυτών τεχνολογιών, να πραγματοποιηθεί σύγκριση μοντέλων και τεχνολογιών, και να παραχθούν κάποιες παρατηρήσεις και συμπεράσματα.

Θα χρησιμοποιηθούν μόνο δωρεάν λογισμικά και λογισμικά ανοιχτού κώδικα, όπως και ανοιχτές πηγές γνώσης, σε πλατφόρμες προσβάσιμες από το ίδρυμα εκπαίδευσης. Η εργασία θα ασχοληθεί σε θεωρητικό επίπεδο με τους αλγορίθμους ΜΜ, τη σύγκριση και μελέτη των διαφορών τους, όσον αφορά την καταλληλότητά τους για το παρόν πρόβλημα. Θα ασχοληθεί επίσης με τις μεθόδους αξιολόγησης πειραμάτων ΜΜ. Όλες οι παραπάνω θεωρητικές αναλύσεις θα συνοδευθούν από την πρακτική τους εφαρμογή πάνω στο παρόν πρόβλημα και θα σημειωθούν οι παρατηρήσεις και τα συμπεράσματα που θα εξαχθούν από αυτήν.

1.2.1 Συνεισφορά

1. Για την υλοποίηση της εργασίας, δημιουργήθηκε ένα παιχνίδι με την πλατφόρμα Unity 3D [3], για τη συλλογή δεδομένων εκπαίδευσης των μοντέλων, αλλά και την αναπαράσταση των πειραμάτων αξιολόγησης των μοντέλων.
2. Μελετήθηκαν και συγκρίθηκαν προ-εκπαιδευμένα (pretrained) μοντέλα ΜΜ. Αρχικά μελετήθηκε η θεωρητική βάση των μοντέλων ώστε να επιλεχθούν και να χρησιμοποιηθούν μοντέλα που αφορούν πιο συγκεκριμένα το πρόβλημα της δυαδικής κατηγοριοποίησης (binary classification), αλλά και αριθμητικά δεδομένα. Σε αντίθεση με μοντέλα που αφορούν κατηγοριοποίηση κειμένου ή εικόνων, τα αριθμητικά δεδομένα έχουν διαφορετική περιπλοκότητα.
3. Τα μοντέλα που επιλέχθηκαν υπέστηκαν επεξεργασία και εφαρμόστηκαν στην πράξη πάνω στα δεδομένα, ώστε να αναδείξουν την καταλληλότητά τους για τη συγκεκριμένη διεργασία. Εκπαιδεύτηκαν 8 μοντέλα με τους αλγορίθμους: logistic regression, decision tree, random forest, K-nearest-neighbor, SVM, Naïve Bayes, LightGBM, και ένα ΝΔ με keras αρχιτεκτονική.
4. Στη συνέχεια μελετήθηκαν τεχνικές υλοποίησης εικονικών περιβαλλόντων για την ανάπτυξη VR παιχνιδιών στο Unity, και εμπλουτίστηκε το περιβάλλον του παιχνιδιού ώστε να γίνει πιο εύχρηστο και να αναδείξει καλύτερα τον σκοπό της εργασίας.
5. Τέλος, τα μοντέλα εφαρμόστηκαν στο παιχνίδι, για να αναδειχθεί η ικανότητά τους στην κατηγοριοποίηση μετά την εκπαίδευσή τους. Παρατίθενται οι μετρικές αξιολόγησής τους, όπως και τα συμπεράσματα της μελέτης.

1.3 Οργάνωση κειμένου

- Στο 2ο κεφάλαιο της εργασίας αναφέρονται σχετικές εργασίες που έχουν ασχοληθεί με την ένωση των δύο θεματικών της εργασίας. Αναλύεται η συνεισφορά τους και η σχετικότητα με την παρούσα εργασία.
- Στο 3ο κεφάλαιο αναλύεται σε βάθος το θεωρητικό υπόβαθρο της εργασίας. Αναλύεται η τεχνολογία VR και η ανάπτυξή της, η έννοια και ιστορία της ΜΜ, η έννοια της Ταξινόμησης και τα μαθηματικά μοντέλα της, τα Νευρωνικά Δίκτυα (ΝΔ), και οι βιβλιοθήκες tensorflow/keras/pytorch.
- Στο 4ο κεφάλαιο περιγράφεται η ανάπτυξη του VR παιχνιδιού και η διαδικασία συλλογής των δεδομένων μέσα στο Unity.
- Στο 5ο κεφάλαιο περιγράφεται η διαδικασία αναζήτησης ελεύθερων δεδομένων, αλλά και η διαδικασία συλλογής και προεπεξεργασίας των δεδομένων.
- Στο 6ο κεφάλαιο αναλύεται η σύγκριση διαφορετικών αλγορίθμων ΜΜ, και το πως μπορούν να αξιοποιηθούν τα ΝΔ για τέτοιες εργασίες.
- Στο 7ο κεφάλαιο περιγράφεται η ανάπτυξη διαφόρων μοντέλων ΜΜ αλλά και ΝΔ , όπως και τα εργαλεία που χρησιμοποιήθηκαν για την εκπαίδευση.
- Στο 8ο κεφάλαιο περιγράφεται η διαδικασία αξιολόγησης των εκπαιδευμένων μοντέλων, αναφέρονται οι παράμετροι αξιολόγησης, και παρατίθενται τα συμπεράσματα της αξιολόγησης.
- Στο 9ο κεφάλαιο περιγράφεται η ενσωμάτωση των εκπαιδευμένων μοντέλων στο Unity, η επίδειξη της ακρίβειάς τους μέσα σε αυτό, όπως και ο εμπλουτισμός της εμπειρίας χρήστη.
- Το 10ο κεφάλαιο είναι ο επίλογος της εργασίας και παρατίθενται η σύνοψη και τα συμπεράσματα αυτής.

2

Σχετικό Έργο

Οι τεχνολογίες με τις οποίες καταπιάνεται η παρούσα εργασία κατέχουν υψηλή δημοτικότητα τα τελευταία χρόνια. Σε όλους τους τομείς, το VR τείνει να χρησιμοποιείται ως επί το πλείστον για την εικονοποίηση δεδομένων. Οι όποιες αναφορές του σε "εκπαίδευση" τείνει να αφορά την εκπαίδευση χρηστών σε διάφορες εργασίες, μέσω κάποιου εικονικού περιβάλλοντος. Εξαιτίας αυτού του φαινομένου, η πλειονότητα των εργασιών που αφορούν την εικονική πραγματικότητα σε συνδυασμό με τη ΜΜ δεν αφορούν την παρούσα εργασία. Παρόλα αυτά, αν και είναι η μειονότητα στην βιβλιογραφική παρουσία τους, έχουν διεξαχθεί αρκετές μελέτες που αξιοποιούν αλγορίθμους ΜΜ για την εκπαίδευση μοντέλων προβλέψεων που αφορούν τις κινήσεις των συσκευών VR, είτε του κράνους είτε των χειριστηρίων, ακόμη και την αναγνώριση χειρονομιών από αυτά. Οι κύριες δύο προσεγγίσεις παρατίθενται παρακάτω, και είναι αυτές της ΜΜ και της Βαθιάς Μάθησης (deep learning - DL).

2.1 Προσέγγιση Μηχανικής Μάθησης

Οι Bahceci et al [4] διεξήγαγαν ένα πείραμα για την αναγνώριση χειρονομιών από τη συσκευή Oculus Quest 2, η οποία χρησιμοποιεί 8 κάμερες inside-out για να ανιχνεύει τις κινήσεις των χεριών και των χειριστηρίων του χρήστη. Χρησιμοποίησαν το πακέτο Oculus Integration και την πλατφόρμα Unity 3D. Για την εκπαίδευση των μοντέλων τους, χρησιμοποίησαν τέσσερις μεθόδους Μάθησης με επίβλεψη: Κ- πλησιέστερος γείτονας (K- Nearest Neighbor, KNN), Random Forest (RF), Gradient Boosting, και Μηχανές διανυσμάτων υποστήριξης (Support Vector Machines, SVMs). Κάποιες από αυτές αναλύονται και παρακάτω στο κεφάλαιο 3.2. Τα δεδομένα τους αποτελούνταν από την απόσταση των άκρων των δακτύλων από το κέντρο της παλάμης του χεριού του χρήστη (Group A), και τη θέση και την περιστροφή των δακτύλων (Group B). Οι χειρονομίες που αποπειράθηκαν να κατηγοριοποιήσουν ήταν έντεκα. Τα αποτελέσματα της κατηγοριοποίησής τους φαίνονται στο Σχήμα 2.1.

Method	Group A Accuracy (%)	Group B Accuracy (%)
KNN	99%	86%
RF	99%	87%
GR	92%	81%
SVM	75%	69%

Σχήμα 2.1: Ακρίβεια κατηγοριοποίησης των τεσσάρων μεθόδων Μάθησης με Επίβλεψη [4]

Οι Moore et al [5] επιχείρησαν να μελετήσουν την αποδοτικότητα μίας εφαρμογής εκπαίδευσης/εξάσκησης μέσω VR (VR Training Application), ως προς το κατά πόσο μαθαίνουν καλύτερα οι χρήστες μέσα στον χρόνο χρήσης της. Το έκαναν αυτό χρησιμοποιώντας την ανίχνευση των κινήσεων του κράνους και των χειριστηρίων τους, και κατηγοριοποιώντας τους σε χρήστες που έμαθαν αρκετά (high-learning - HL), και χρήστες που δεν έμαθαν αρκετά (low-learning - LL) μέσω της εξάσκησης. Κάνοντας ένα τεστ γνώσεων στους χρήστες, τους κατηγοριοποίησαν οι ίδιοι σε HL & LL, και χρησιμοποίησαν Μάθηση με Επίβλεψη με τη μέθοδο SVM δίνοντάς της τα δεδομένα κινήσεων και τις κατηγοριοποιήσεις τους, ώστε να παράγει ένα μοντέλο κατηγοριοποίησης. Η ακρίβεια των πειραμάτων τους ανήλθε στο 85.7% για τα δεδομένα εκπαίδευσης και στο 93.1% για τα δεδομένα testing. Το ζητούμενό τους ήταν να μπορούν να προβλέψουν αν ο χρήστης θα ήταν HL ή LL μόνο από τα δεδομένα των κινήσεών τους. Επικεντρώθηκαν αρκετά στο να κάνουν fit το μοντέλο τους στα VR δεδομένα.

Οι Nier et al [6] κατέχοντας ένα μεγάλο σετ δεδομένων από χρήστες (55.000+) του παιχνιδιού Beatsaber [7], επιχείρησαν να ταυτοποιήσουν τους χρήστες μέσω των βιομετρικών δεδομένων τους από το VR set του καθενός, χρησιμοποιώντας μοντέλα MM. Τα δεδομένα αποτελούνταν από metadata, θέση και προσανατολισμό του κράνους και των χειριστηρίων, το περιβάλλον του παιχνιδιού, και την απόδοση του εκάστοτε συστήματος. Τα χαρακτηριστικά που χρησιμοποίησαν αφορούσαν μέχρι 10 λεπτά παιχνιδιού για τον κάθε χρήστη, και χρησιμοποίησαν το 70% των δεδομένων για την εκπαίδευση, 10% για validation και 20% για testing. Μελέτησαν διάφορα μοντέλα MM αλλά και βαθιάς μάθησης και πειραματίστηκαν σε 500 ίδια δείγματα των δεδομένων για να καταλήξουν σε αυτό με την καλύτερη απόδοση και ακρίβεια για το πείραμά τους (Σχήμα 2.2). Όλα τα μοντέλα που δοκίμασαν έδωσαν χαμηλότερη ακρίβεια από το LightGBM, ένα δέντρο απόφασης κλιμακωτής ενίσχυσης [8], ένα διαδεδομένο μοντέλο MM με μεγάλη απόδοση. Επειδή χρειάζονταν ταξινόμηση 55.000+ κλάσεων, κανένα μοντέλο δεν θα μπορούσε να κάνει αποδοτικά αυτή τη δουλειά από μόνο του, τουλάχιστον χωρίς να αποκλείει το οποιοδήποτε μελλοντικό scalability. Επομένως ανέπτυξαν έναν πολύ-επίπεδο ιεραρχικό ταξινομητή με τρία στρώματα από μικρότερους ταξινομητές, όπου ο καθένας εκπαιδεύτηκε σε μικρότερο δείγμα κλάσεων. Τα πειράματά τους επέφεραν 94,3% ακρίβεια με 50 δείγματα, δηλαδή με

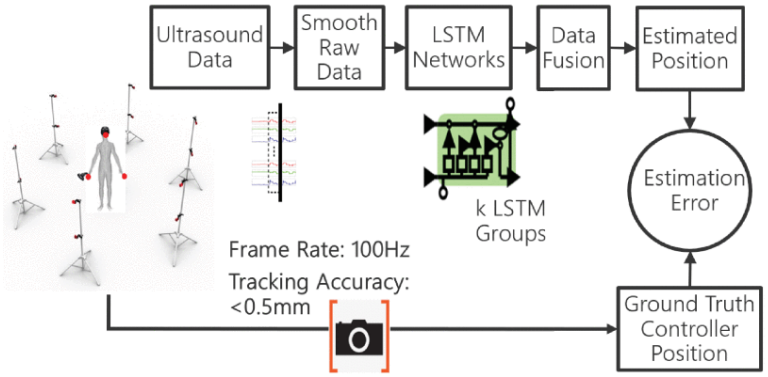
100 δευτερόλεπτα δεδομένων. Σε χρήστες με περισσότερες προσπάθειες του παιχνιδιού, πέτυχαν έως και 99.5% ακρίβεια.

Layer	# of Models	Accuracy (per Model)	Accuracy (per Layer)
Layer 1	10	93.1%	90.2%
Layer 2	10	93.1%	90.2%
Layers 1 & 2	20	93.1%	91.0%
Layer 3	5	84.0%	84.0%
Layers 1, 2, & 3	25	91.3%	94.3%

Σχήμα 2.2 : Ακρίβεια κάθε στρώματος ιεραρχικού μοντέλου ανά μοντέλο και ανά στρώμα, Nier et al [6]

2.2 Προσέγγιση Βαθιάς Μάθησης

Οι C. Song και S. Zarar [9] ανέπτυξαν μία μέθοδο για την εκτίμηση της θέσης των χειριστηρίων VR σε πραγματικό χρόνο, όταν αυτά εξέρχονται από το FOV (πεδίο ορατότητας – Field Of View) των καμερών του κράνους VR. Ενίσχυσαν το VR set τους με αισθητήρες υπερήχων, έναν πομπό υπερήχων στο κράνος και έναν δέκτη υπερήχων στο ένα χειριστήριο. Συγχρονίζοντας τους αισθητήρες μεταξύ τους μέσω εφαρμογής που υλοποίησαν με το πρωτόκολλο UDP, κατάφεραν να παράγουν εκτιμήσεις απόστασης ανάμεσα στο κράνος και το χειριστήριο, σχεδόν σε πραγματικό χρόνο. Στη συνέχεια χρησιμοποίησαν τα ήδη παρεχόμενα δεδομένα από το VR set, μέσω του SteamVR [10] και του Unity, για να εκπαιδεύσουν ένα μοντέλο ΜΜ ώστε να προβλέπει την κίνηση των χειριστηρίων. Επειδή το πείραμά τους εμπλέκει άμεσα τον χρόνο ανάδρασης, επέλεξαν να χρησιμοποιήσουν Δίκτυα Μακράς Βραχύχρονης Μνήμης (Long Short-term Memory - LSTM), τα οποία είναι μορφή τεχνητών Αναδρομικών νευρωνικών δικτύων (Recurrent Neural Networks – RNN), που χρησιμοποιούνται στον τομέα της Βαθιάς Μάθησης [11]. Ένωσαν τα δεδομένα από τους αισθητήρες με αυτά του LSTM μοντέλου για να παράγουν εκτιμήσεις της κίνησης του χειριστηρίου για τα επόμενα 10 δευτερόλεπτα. Κατάφεραν να κρατήσουν το σφάλμα εκτίμησης στα 2 εκατοστά, επεμβαίνοντας περιοδικά στις μετρήσεις με τα δεδομένα των αισθητήρων υπερήχων. Τέλος, χρησιμοποίησαν ένα σύστημα οκτώ καμερών για να καταγράψουν τις κινήσεις των χρηστών του πειράματός τους, ώστε να τα χρησιμοποιήσουν ως μέτρο σύγκρισης στην επιτυχία των προβλέψεών τους. Ισχυρίζονται ότι η μέθοδός τους μπορεί κάλλιστα να λειτουργήσει και για δύο χειριστήρια, όπως και περισσότερους από έναν χρήστες. Οι τεχνικές προδιαγραφές της μεθόδου τους εμφανίζονται στο Σχήμα 2.3.



Hardware	
MCU (Cortex M0)	80 MHz, 4 MB Flash, 3.3 V, 478 mW
Ultrasound sensor	42 kHz, 20-100 cm (range), 3.3 V
Software	
MXR toolkit	SteamVR + Unity
Base desktop	Python, C#
Ground-truth	8 Flex3 Camera + Motive 2.0.2

Dataset	
Number of trials	10
Avg. length of trial	12 mins
Ultrasound ranging rate.	4 Hz
OptiTrack sampling freq.	100 Hz
MXR sampling freq.	50-60 Hz

Σχήμα 2.3 : Τεχνικές προδιαγραφές της μεθόδου των C. Song και S. Zarar [9].

Οι Liebers et al [12] ασχολήθηκαν με την αναγνώριση χρηστών μέσω των βιομετρικών στοιχείων τους από την εμπειρία VR. Υλοποίησαν δύο σενάρια για τους χρήστες του πειράματός τους, το ένα με παιχνίδι bowling και το ένα με παιχνίδι τοξοβολίας. Στο πρώτο οι χρήστες καλούνται να ρίξουν τις κορίνες πετώντας τη μπάλα, και στο δεύτερο να πετύχουν το στόχο φορτώνοντας το τόξο και εκτοξεύοντας το βέλος. Αρχικά πήραν μετρήσεις του κάθε χρήστη για το ύψος τους και το μήκος των χεριών τους, μέσα από το VR set. Έπειτα κανονικοποίησαν τα δεδομένα για να είναι συνεπή μεταξύ των χρηστών, χωρίς παράλληλα να μειώνουν την εμπειρία τους και να προκαλείται ίλιγγος. Δημιούργησαν τέσσερα feature sets για την εκπαίδευση, το καθένα με κάπως διαφορετικά χαρακτηριστικά ή διαφορετικά επεξεργασμένα χαρακτηριστικά. Για την ταξινόμηση ανέπτυξαν δύο μοντέλα βαθιάς μάθησης με Python και Tennsorflow/Keras. Το πρώτο είχε τρία στρώματα LSTM και ένα στρώμα εξόδου με τη συνάρτηση ενεργοποίησης softmax [13], όπως και χρησιμοποιήθηκε ο βελτιστοποιητής (optimizer) Adam με βήμα εκπαίδευσης 0.001. Το μοντέλο εκπαιδεύτηκε για 500 εποχές. Το δεύτερο μοντέλο ήταν ένα Perceptron πολλαπλών στρωμάτων (Multilayer Perceptron) με ένα Flatten στρώμα εισόδου, τρία Πυκνά στρώματα όπου στα δύο μπήκε η συνάρτηση ενεργοποίησης ReLU [13] ενώ στο τρίτο η softmax. Ως βελτιστοποιητής χρησιμοποιήθηκε το stochastic gradient descent – SGD με βήμα εκπαίδευσης 0.001, και το μοντέλο εκπαιδεύτηκε για 100 εποχές, μιας και γενικά συγκλίνει γρηγορότερα. Στο Σχήμα 2.4 φαίνονται τα αποτελέσματα της εκπαίδευσης των δύο μοντέλων στα 4 feature sets. Η μελέτη τους έδειξε ότι με λιγότερα δεδομένα τα μοντέλα βαθιάς μάθησης μπορούν να δώσουν μεγαλύτερης ακρίβειας αποτελέσματα. Έδειξε επίσης ότι με την κανονικοποίηση των δεδομένων, η αναγνώριση των χρηστών μπορεί να επιτευχθεί χωρίς τα βιομετρικά τους

χαρακτηριστικά, μόνο από τα συμπεριφορικά τους στοιχεία κατά την καταγραφή. Τέλος έδειξε ότι κατά 90% μπορούν να ταυτοποιήσουν τους χρήστες μέσω ενός VR set κατά τη χρήση εντός μερικών ημερών.

		Acc. WN		**Acc. AN**		**Acc. HN**		**Acc. BN**	
Scenario	**F.-Set**	MLP	RNN	MLP	RNN	MLP	RNN	MLP	RNN
Archery	F0	0.32	0.48	0.56	0.67	0.57	0.63	**0.68**	**0.86**
Bowling	F0	**0.55**	**0.65**	0.41	0.53	**0.55**	0.59	0.46	0.58
Archery	F1	0.38	0.54	0.56	0.63	**0.65**	0.69	0.64	**0.84**
Bowling	F1	0.49	0.59	0.42	0.50	**0.55**	**0.62**	0.48	0.60
Archery	F2	0.37	0.56	0.56	0.65	**0.66**	0.66	0.63	**0.86**
Bowling	F2	0.52	0.58	0.42	0.47	**0.54**	**0.68**	0.45	0.56
Archery	F3	0.61	0.63	0.68	0.70	**0.78**	**0.90**	0.68	0.84
Bowling	F3	0.55	0.56	0.43	0.47	**0.63**	0.66	0.55	**0.67**

Σχήμα 2.4 : Αποτελέσματα εκπαίδευσης μοντέλων των Liebers et al [12].

3

Θεωρητικό Υπόβαθρο

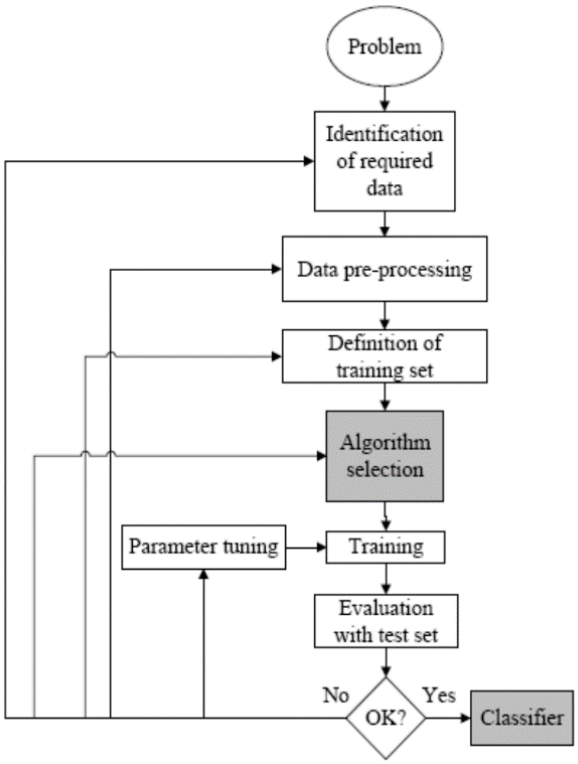
3.1 Μηχανική Μάθηση

Έχουν ειπωθεί πολλοί ορισμοί της ΜΜ, καθώς είναι μία έννοια που υπάρχει από το 1959. Ο K.P. Murphy [14] ορίζει τη Μηχανική Μάθηση ως ένα σετ μεθόδων που μπορούν αυτόματα να ανιχνεύουν μοτίβα σε δεδομένα, και να χρησιμοποιούν αυτά τα μοτίβα για να προβλέψουν μελλοντικά δεδομένα, ή να πάρουν άλλου είδους αποφάσεις, υπό συνθήκες αβεβαιότητας. Χωρίζει τη ΜΜ σε τρεις μεγάλες κατηγορίες:

- Μάθηση με επίβλεψη (predictive/supervised learning). Δίνεται στον αλγόριθμο μία χαρτογράφηση από εισόδους X σε εξόδους Y, με ετικέτες. Η χαρτογράφηση αυτή είναι τα μοτίβα των σχέσεων ανάμεσα στα ζευγάρια εισόδου-εξόδου. Ο αλγόριθμος καλείται να ρυθμιστεί ανάλογα τα μοτίβα αυτά και να μπορεί να προβλέψει τις εξόδους νέων δεδομένων εισόδου. Αν οι έξοδοι είναι διακριτές τιμές έχουμε πρόβλημα ταξινόμησης (classification), ενώ αν είναι συνεχείς τιμές, πρόβλημα παλινδρόμησης (regression).
- Μάθηση χωρίς επίβλεψη (descriptive/unsupervised learning). Τα δεδομένα που δίνονται στον αλγόριθμο είναι δίχως ετικέτες, και αυτός καλείται να βρει «ενδιαφέροντα» μοτίβα στα δεδομένα. Είναι πιο περίπλοκο να αξιολογηθούν τέτοια μοντέλα, εφόσον δεν υπάρχουν από πριν τα μοτίβα για να συγκριθούν τα αποτελέσματα και να παραχθούν μετρικές αξιολόγησης. Ένα σύνηθες παράδειγμα αυτού του τύπου μάθησης είναι η συσταδοποίηση (clustering), όπου τα δεδομένα ομαδοποιούνται σε συστάδες με βάση κοινά τους χαρακτηριστικά.
- Μάθηση με ενίσχυση (reinforcement learning). Μία ανερχόμενα διαδεδομένη κατηγορία, όπου δίνεται στον αλγόριθμο κάποιο σήμα επιβράβευσης ή τιμωρίας για κάθε σωστή ή λάθος πρόβλεψη. Ο αλγόριθμος μαθαίνει μέσω της ενίσχυσης και είναι εξελίξιμος.

3.2 Ταξινόμηση

Όπως αναφέρθηκε, η Ταξινόμηση αποτελεί υποκατηγορία της Μάθησης με επίβλεψη στη ΜΜ. Ο αλγόριθμος καλείται να δημιουργήσει έναν ταξινομητή (classifier) που να μπορεί να γενικεύει αποτελέσματα από νέες εισόδους [15]. Η Γενίκευση ορίζεται ως η ικανότητα να εκτιμάται η σωστή έξοδος t_i για πρότυπα εισόδου x_i που δεν έχουν απαντηθεί κατά την εκπαίδευση. Το σετ δεδομένων εκπαίδευσης ορίζεται ως το training set και το σετ δεδομένων αξιολόγησης ορίζεται ως το test set. Η διαδικασία της ταξινόμησης παρατίθεται στο Σχήμα 3.1, όπου και απεικονίζεται η σημαντικότητα της επιλογής αλγορίθμου για τη δημιουργία του ταξινομητή [16]. Η επιλογή της μεθόδου είναι ένα μεγάλο και σημαντικό κομμάτι στην ανάπτυξη του αλγορίθμου μάθησης και βασίζεται κατά μεγάλο μέρος στο είδος των δεδομένων και στο ζητούμενο του εγχειρήματος.



Σχήμα 3.1: Διαδικασία μάθησης με επίβλεψη/ταξινόμησης [16]

Υπάρχουν αρκετές τεχνικές ταξινόμησης δεδομένων οι οποίες παρατίθενται στο Σχήμα 3.2. Στα Κεφάλαια 6 & 7 αυτής της εργασίας αναλύεται η διαδικασία σύγκρισης διαφόρων προ-εκπαιδευμένων μοντέλων ταξινόμησης σε πρακτικό επίπεδο, για την επιλογή του καταλληλότερου για το πρόβλημα μοντέλου. Οι Aized και Arshad [15] και οι Osisanwo et al [16] έχουν πραγματοποιήσει δύο ενδελεχείς έρευνες και συγκρίσεις πάνω στους αλγορίθμους μάθησης για το πρόβλημα της Ταξινόμησης που συνοψίζονται παρακάτω:

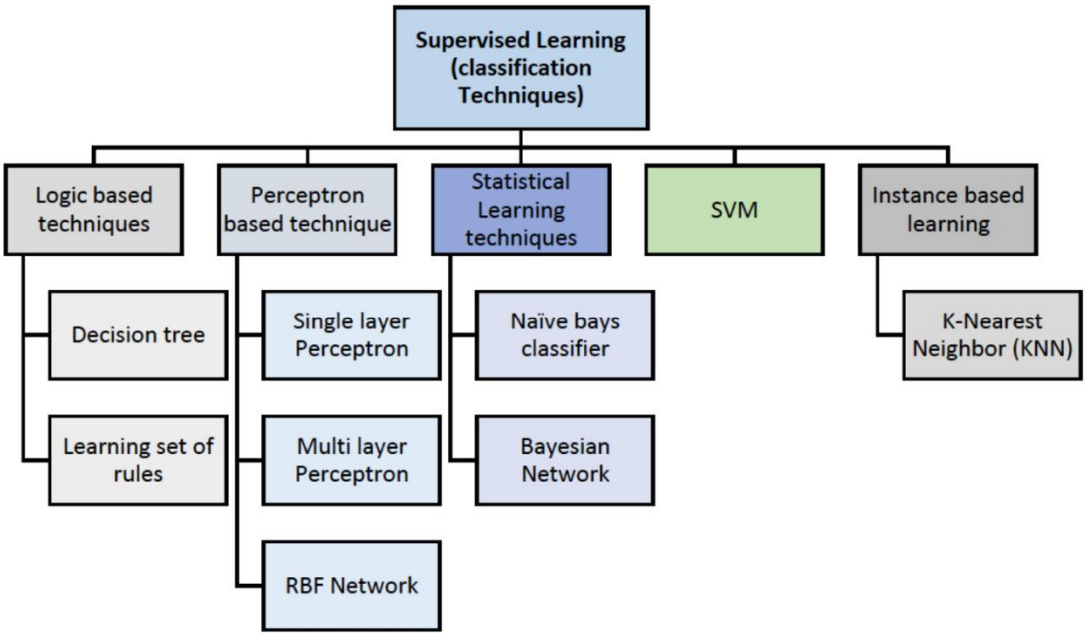
- Δέντρα αποφάσεων (Decision Trees): Κατηγοριοποιούν οντότητες ταξινομώντας με βάση τα γνωρίσματα τους. Ξεκινώντας από πάνω, από τη ρίζα του δέντρου, κάθε φύλλο του δέντρου αναπαριστά ένα γνώρισμα, και κάθε κλαδί μία τιμή που μπορεί να πάρει το γνώρισμα. Συνήθως δημιουργούνται σε δύο στάδια, το αναδρομικό μεγάλωμα του δέντρου και το κλάδεμα του δέντρου. Αξιοσημείωτοι αντιπροσωπευτικοί αλγόριθμοι ο ID3 και ο C4.5.

- Μπεϊζιανά δίκτυα (Bayesian Networks): Γραφικά μοντέλα για την αναπαράσταση πιθανοτικών σχέσεων ανάμεσα σε σετ γνωρισμάτων. Μπορούν να λάβουν υπόψιν παρελθοντικές πληροφορίες που αφορούν το πρόβλημα όσον αφορά τη δομή των σχέσεων ανάμεσα στα γνωρίσματα. Δεν ενδείκνυνται όμως για σετ δεδομένων με πολλά γνωρίσματα.

- K- πλησιέστερος γείτονας (K- Nearest Neighbor, KNN): Βασίζεται στο πόσα στοιχεία πρέπει να είναι κοντινά μεταξύ τους με βάση τα γνωρίσματά τους, ώστε να αποτελούν μία “γειτονιά”. Ενδείκνυται για μεγάλα και θορυβώδη σετ δεδομένων εκπαίδευσης. Απαιτούν πολύ χώρο και χρόνο για την ταξινόμηση και χωλαίνουν στην κατηγοριοποίηση πολυμέσων γι’ αυτό το λόγο.

- Μηχανές διανυσμάτων υποστήριξης (Support Vector Machines, SVMs): Απευθύνονται αποκλειστικά σε δυϊκά προβλήματα ταξινόμησης (όπου οι κλάσεις εξόδου μπορούν να είναι δύο τιμών), και χωρίζουν γραμμικά ή και μη γραμμικά τις δύο κλάσεις. Τα στοιχεία που βρίσκονται πιο κοντά στη διαχωριστική γραμμή των κλάσεων αποτελούν το διάνυσμα υποστήριξης. Έχουν τη δυνατότητα να μειώσουν το αναμενόμενο σφάλμα γενίκευσης απομακρύνοντας αυτά τα στοιχεία από τη διαχωριστική γραμμή.

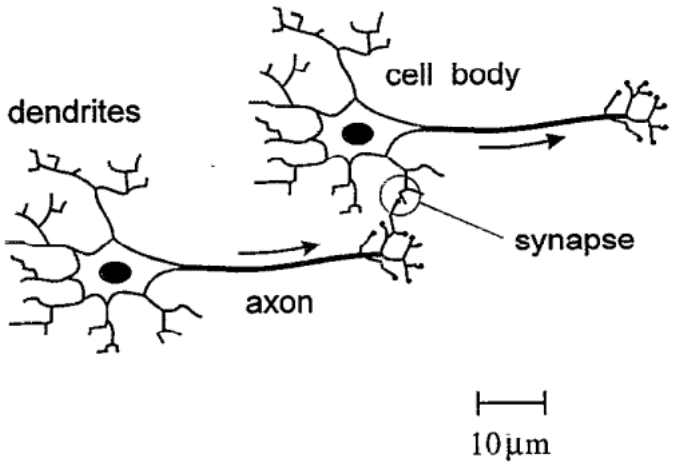
- Perceptron πολλαπλών στρωμάτων (Multilayer Perceptron): Χρησιμοποιεί κρυφά στρώματα νευρώνων. Τα δεδομένα εκπαίδευσης περνάνε επανειλημμένα από τα στρώματα και τους ανατίθενται βάρη, μέχρι να βρεθεί ένα σωστό διάνυσμα πρόβλεψης για όλο το σετ δεδομένων. Όσο πιο πολύπλοκη είναι η συνάρτηση που προσεγγίζεται, τόσο περισσότεροι κρυφοί νευρώνες απαιτούνται.



Σχήμα 3.2: Τεχνικές μάθησης με επίβλεψη/ταξινόμησης [15]

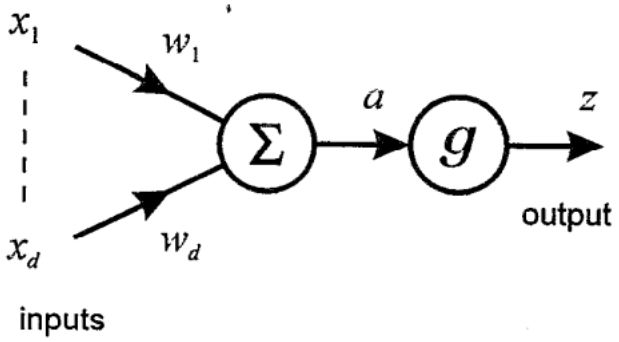
3.3 Νευρωνικά Δίκτυα

Η συμβατική προσέγγιση της πληροφορικής βασίζεται σε ένα ρητό σύνολο προγραμματισμένων εντολών και χρονολογείται από το έργο των Babbage, Turing και von Neumann. Τα νευρωνικά δίκτυα (ΝΔ) αντιπροσωπεύουν ένα εναλλακτικό υπολογιστικό πρότυπο όπου η λύση ενός προβλήματος μαθαίνεται από ένα σύνολο παραδειγμάτων. Η έμπνευση για τα νευρωνικά δίκτυα προέρχεται αρχικά από μελέτες των μηχανισμών επεξεργασίας πληροφοριών στα βιολογικά νευρικά συστήματα (Σχήμα 3.3), ιδίως στον ανθρώπινο εγκέφαλο [17].



Σχήμα 3.3: Αναπαράσταση δύο βιολογικών νευρώνων [17].

Ένα νευρωνικό δίκτυο τροφοδότησης μπορεί να θεωρηθεί ως μια μη γραμμική μαθηματική συνάρτηση η οποία μετασχηματίζει ένα σύνολο εισόδου μεταβλητών σε ένα σύνολο μεταβλητών εξόδου. Η ακριβής μορφή του μετασχηματισμού διέπεται από ένα σύνολο παραμέτρων που ονομάζονται βάρη, οι τιμές των οποίων μπορούν να καθοριστούν με βάση ένα σύνολο παραδειγμάτων του απαιτούμενου mapping. Η διαδικασία προσδιορισμού των τιμών αυτών των παραμέτρων ονομάζεται συχνά μάθηση ή εκπαίδευση, και μπορεί να είναι ένα υπολογιστικά εντατικό εγχείρημα. Μόλις καθοριστούν τα βάρη, ωστόσο, νέα δεδομένα μπορούν να επεξεργαστούν από το δίκτυο πολύ γρήγορα. Μπορούμε να κάνουμε μια αναλογία μεταξύ των τεχνητών νευρωνικών δικτύων και της τυπικής τεχνικής της προσαρμογής καμπυλών με τη χρήση πολυωνυμικών συναρτήσεων. Ένα πολυώνυμο μπορεί να θεωρηθεί ως μια απεικόνιση από μια απλή μεταβλητή εισόδου σε μία μόνο μεταβλητή εξόδου. Οι συντελεστές στο πολυώνυμο είναι ανάλογοι με τα βάρη σε ένα νευρωνικό δίκτυο, και ο προσδιορισμός αυτών των συντελεστών (ελαχιστοποιώντας το σφάλμα του αθροίσματος των τετραγώνων) αντιστοιχεί στη διαδικασία της εκπαίδευσης δικτύου. Στο Σχήμα 3.4 φαίνεται πως ένας νευρώνας δημιουργεί ένα άθροισμα κάποιων εισόδων με βάρη, το μετατρέπει χρησιμοποιώντας μια μη-γραμμική συνάρτηση ενεργοποίησης, και δίνει την τελική έξοδο.



Σχήμα 3.4: Το μοντέλο McCulloch – Pitts ενός νευρώνα [17].

Εκτός από την υψηλή ταχύτητα επεξεργασίας, τα νευρωνικά δίκτυα έχουν τη σημαντική ικανότητα να μαθαίνουν μία γενική λύση ενός προβλήματος από ένα σύνολο συγκεκριμένων παραδειγμάτων. Σε πολλές εφαρμογές αυτό παρακάμπτει την ανάγκη ανάπτυξης ενός μοντέλου πρώτων δοκιμών για τις υποκείμενες φυσικές διεργασίες, το οποίο μπορεί συχνά να αποδειχθεί δύσκολο ή αδύνατο να βρεθεί. Τα κύρια μειονεκτήματα των νευρωνικών δικτύων προέρχονται από την ανάγκη παροχής ενός κατάλληλου συνόλου δεδομένων-παραδειγμάτων για την εκπαίδευση του δικτύου, και τα πιθανά προβλήματα που μπορεί να προκύψουν, αν ένα δίκτυο πρέπει να επεκταθεί σε νέες περιοχές του χώρου εισόδου, οι οποίες διαφέρουν σημαντικά από εκείνες που αντιστοιχούν στα δεδομένα εκπαίδευσης. Σε πολλές πρακτικές εφαρμογές τα προβλήματα αυτά δεν θα είναι σημαντικά, ενώ σε άλλες περιπτώσεις μπορούν να χρησιμοποιηθούν διάφορες τεχνικές για την άμβλυνση αυτών των αρνητικών επιπτώσεων.

3.4 Tensorflow – Keras, Pytorch

Το TensorFlow [18] είναι μια δωρεάν και ανοικτού κώδικα βιβλιοθήκη λογισμικού για μηχανική μάθηση και τεχνητή νοημοσύνη. Μπορεί να χρησιμοποιηθεί σε ένα ευρύ φάσμα εργασιών, αλλά εστιάζει ιδιαίτερα στην εκπαίδευση και την εξαγωγή συμπερασμάτων βαθιών νευρωνικών δικτύων [19]. Αναπτύχθηκε από την ομάδα Google Brain για εσωτερική χρήση της Google στην έρευνα και την παραγωγή. Η είσοδος που τροφοδοτείται στο TensorFlow είναι ένας πολυδιάστατος πίνακας και αυτοί οι πίνακες ονομάζονται τένσορες. Η ροή ενός πολυδιάστατου πίνακα έδωσε το όνομα TensorFlow. Αυτοί οι τένσορες εισέρχονται στο σύστημα ως είσοδος στο ένα άκρο, περνούν από διάφορες πράξεις και τελικά παράγουν έξοδο (Σχήμα 3.5).

Σχήμα 3.5: Αρχιτεκτονική TensorFlow [19].

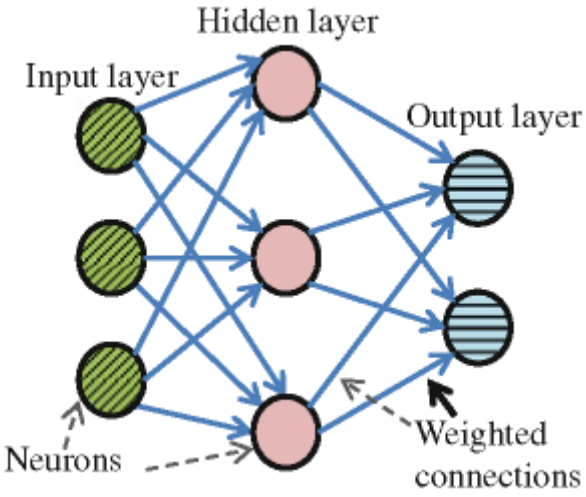
Το Keras είναι ένα API νευρωνικών δικτύων βασισμένο στην Python [20], το οποίο χρησιμοποιείται για την εκτέλεση των TensorFlow, CNTK και Theano. Χρησιμοποιείται με το TensorFlow 2.0 πάνω στο οποίο βρίσκεται η τελευταία έκδοση του Keras. Στην Python, το Keras χρησιμοποιείται με τον συμβολισμό tf.keras. Έχει τέσσερις αρχές λειτουργίας ως εξής:

- Φιλικότητα προς τον χρήστη
- Διαρθρωσιμότητα
- Εύκολη επεκτασιμότητα
- Εργασία με Python

Τα μοντέλα είναι οι πιο σημαντικές δομές δεδομένων στο Keras. Το Sequential είναι ένα μοντέλο που χρησιμοποιείται ευρέως. Αρχικά υπάρχει ένα στρώμα εισόδου, έπειτα ένα ή περισσότερα κρυφά στρώματα, και τέλος ένα στρώμα εξόδου (Σχήμα 3.6).

Υπάρχουν 11 διαφορετικοί τύποι επιπέδων Keras που είναι διαθέσιμοι για τη διαμόρφωση του νευρωνικού δικτύου βαθιάς μάθησης και παρατίθενται παρακάτω:

1. **Στρώματα πυρήνα**, τα εξής: Dense, Activation, Dropout, Flatten, Reshape, Permute, RepeatVector, Lambda, Activity Regularization.
2. **Συνελικτικά στρώματα:** Όπως και στο TensorFlow, τα στρώματα συνελικτικής ανάλυσης στο Keras αποτελούνται από Conv 1D, Conv 2D, Separable Conv 1D, Separable Conv 2D, Depthwise Conv 1D, Depthwise Conv 2D, Conv 2D Transpose και 3D όλων των παραπάνω.
3. **Pooling Layers:** Τα επίπεδα συγκέντρωσης του Keras είναι τα εξής: max pooling, average pooling, average max pooling, global max pooling, κ.λπ.
4. **Locally Connected Layers**: Τα τοπικά συνδεδεμένα 1D και τοπικά συνδεδεμένα 2D είναι στρώματα που υπάρχουν στα τοπικά συνδεδεμένα στρώματα.
5. **Recurrent Layers**: Τα επαναλαμβανόμενα στρώματα αποτελούνται από εκείνα που λειτουργούν αποκλειστικά με επαναλαμβανόμενα νευρωνικά δίκτυα (RNN), τα στρώματα Gated Recurrent Unit (GRU) και τα στρώματα μακράς βραχυπρόθεσμης μνήμης (LSTM).
6. **Embedding Layers**
7. **Merge Layers**
8. **Σύνθετα στρώματα ενεργοποίησης**: Όπως το LeakyReLU , PReLU , ReLU, softmax, ELU και το κατώφλι ReLU.
9. **Επίπεδα κανονικοποίησης** όπως το Batch Normalization.
10. **Noise Layers**: Το Gaussian noise, το Gaussian dropout και το alpha dropout είναι τα στρώματα θορύβου που χρησιμοποιούνται για τη μείωση του θορύβου στα δεδομένα που παρέχονται.
11. **Layer Wrappers**: Το Time distributed wrapper χρησιμοποιείται για την εφαρμογή στο χρονικό χώρο της εισόδου, ενώ το bidirectional wrapper χρησιμοποιείται με τα RNNs.



Σχήμα 3.6: ΝΔ τροφοδότησης με τα είδη στρωμάτων του.

Το PyTorch [21] είναι ένα framework μηχανικής μάθησης βασισμένο στη βιβλιοθήκη Torch, που χρησιμοποιείται για εφαρμογές όπως η όραση υπολογιστών και η επεξεργασία φυσικής γλώσσας, το οποίο αναπτύχθηκε αρχικά από την Meta AI και τώρα ανήκει στην ομπρέλα του Linux Foundation. Πρόκειται για ελεύθερο λογισμικό ανοικτού κώδικα που κυκλοφορεί υπό την τροποποιημένη άδεια BSD. Στα πλαίσια της εργασίας, χρησιμοποιήθηκε στους πειραματισμούς με προ-εκπαιδευμένα μοντέλα ΝΔ μέσα στο περιβάλλον ανάπτυξης, όμως όπως θα αναλυθεί παρακάτω, δεν επέφερε ωφέλιμα αποτελέσματα. Λειτουργεί και αυτό με τένσορες, αν και προτιμάται το tensorflow για έργα μεγάλης κλίμακας, με σημαντικές απαιτήσεις εγκατάστασης και ανάπτυξης.

3.5 Unity & VR

Για τις ανάγκες αυτής της εργασίας αναπτύχθηκε ένα παιχνίδι στην πλατφόρμα Unity 3D [3]. Μια δωρεάν πλατφόρμα που υποστηρίζει την ανάπτυξη παιχνιδιών με 3D αλλά και 2D γραφικά, που μπορεί να ενσωματώσει λειτουργίες AR και VR, που απευθύνεται σε διάφορες πλατφόρμες λογισμικού (Windows, iOS, Android, Linux, PS4/5, WebGL, κ.α.). Επιπλέον υποστηρίζει την ανάπτυξη διαφόρων media, όπως προσομοιώσεις και ταινίες. Ένας προγραμματιστής μπορεί να εκμεταλλευτεί τις δυνατότητες της μηχανής αυτής μέσω αρχείων κώδικα στη γλώσσα C#. Επιπρόσθετη διευκόλυνση που παρέχεται από τη μηχανή, είναι η αυτόματη σύνδεσή της με το Visual Studio IDE (.NET αρχιτεκτονική), ώστε να γίνεται αμέσως compiled ο κώδικας και το debugging να μπορεί να γίνεται άμεσα μέσω της τερματικής κονσόλας στο Unity.

Η ΜΜ συχνά ενσωματώνεται σε εφαρμογές που αναπτύσσονται με χρήση του Unity και VR. Αυτό επιτρέπει τη δημιουργία εξαιρετικά εξατομικευμένων και εκπληκτικά αληθοφανών εμπειριών. Οι αλγόριθμοι ΜΜ μπορούν να χρησιμοποιηθούν για τη βελτίωση της διαδραστικότητας, την πρόβλεψη της κίνησης του χρήστη, ακόμα και για την προσαρμογή του περιεχομένου σύμφωνα με τη συμπεριφορά του. Με τη συνδυασμένη χρήση του Unity, του VR και της ΜΜ, μπορούν να δημιουργηθούν πραγματικά εκπληκτικές και επαναστατικές εφαρμογές που προσφέρουν μια εντελώς νέα εμπειρία στους χρήστες.

Η συνδυασμένη χρήση του Unity, του VR και της ΜΜ έχει ευρεία εφαρμογή σε πολλούς τομείς, συμπεριλαμβανομένης της εκπαίδευσης, της ψυχολογίας και των παιχνιδιών. Στον τομέα της εκπαίδευσης, δημιουργούνται εκπαιδευτικά περιβάλλοντα μέσω των οποίων οι μαθητές μπορούν να αλληλεπιδρούν με το μάθημα με πιο ενεργό τρόπο, ενθαρρύνοντας τη συμμετοχή και την ενασχόλησή τους. Στην ψυχολογία, εφαρμόζονται ειδικές εφαρμογές που χρησιμοποιούν την εικονική πραγματικότητα για τη θεραπεία συναισθηματικών διαταραχών και την αντιμετώπιση φόβων και φοβιών. Τέλος, στον τομέα των παιχνιδιών, δημιουργούνται εντυπωσιακά και αληθοφανή εικονικά περιβάλλοντα που προσφέρουν στους παίκτες μια μοναδική εμπειρία διασκέδασης και περιπέτειας. Με

την ενσωμάτωση της Μηχανικής Μάθησης, οι δημιουργοί μπορούν να δημιουργήσουν εξατομικευμένες εμπειρίες παιχνιδιού που προσαρμόζονται δυναμικά στις προτιμήσεις και τις ικανότητες του κάθε παίκτη. Με αυτόν τον τρόπο, η συνδυασμένη χρήση αυτών των τεχνολογιών προσφέρει πλούσιες και ενδιαφέρουσες εμπειρίες σε διάφορους τομείς. Στην παρούσα εργασία θα εξεταστούν και οι πρακτικοί τρόποι με τους οποίους υλοποιούνται αυτές οι τεχνολογίες.

4

Ανάπτυξη παιχνιδιού

Το παιχνίδι καλείται να είναι μία πλατφόρμα αλληλεπίδρασης του χρήστη με τις κονσόλες VR που αφορά η εργασία, και ταυτόχρονα το μέσο συλλογής των δεδομένων που χρειάστηκαν για την εκπαίδευση του μοντέλου MM. Η αλληλεπίδραση με τον χρήστη περιλαμβάνει την εκτόξευση σφαιρικών αντικειμένων με κατεύθυνση την θέση του χρήστη, την ενέργειά του ως προς την κίνηση μιας ρακέτας για την απόκρουση του αντικειμένου, και την καταγραφή όλων αυτών των κινήσεων σε μορφή ωφέλιμων δεδομένων.

4.1 Unity Setup

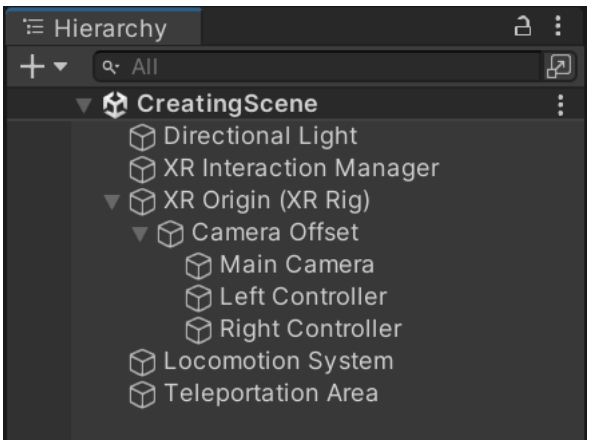
Αρχικά δημιουργήθηκε ένα project μέσω του προγράμματος Unity Hub, με το template 3D project που παρέχει απαραίτητα συστατικά εκκίνησης ενός έργου με τρισδιάστατα γραφικά και μηχανισμούς φυσικής στον τρισδιάστατο χώρο. Η έκδοση Unity που χρησιμοποιήθηκε είναι η Unity 2022.3.9f1, μία από τις τελευταίες LTS (Long-Term Support) εκδόσεις του λογισμικού στην αρχή της εργασίας.

Απαραίτητα για την ενσωμάτωση VR στοιχείων στο παιχνίδι ήταν τα πακέτα/βιβλιοθήκες XR plug-in Management και XR Interaction Toolkit + Default Input Actions. Αυτά περιέχουν ρυθμίσεις και εργαλεία για την σωστή λειτουργία της πλατφόρμας σε συνδυασμό με συσκευές VR, αλλά και scripts και εργαλεία που βοηθούν και εμπλουτίζουν την αρχικοποίηση ενός τέτοιου project, με κάποιες βασικές λειτουργίες. Το XR στο όνομά τους σημαίνει eXtended Reality – Εκτεταμένη Πραγματικότητα, που περιλαμβάνει όλα τα είδη Εικονικής, Επαυξημένης, Μικτής Πραγματικότητας. Αυτές οι ρυθμίσεις υλοποιούν ένα παγκόσμιο πρότυπο, το OpenXR [22], ένα δωρεάν πρότυπο ανοιχτού κώδικα που αφορά την πρόσβαση σε XR πλατφόρμες και συσκευές.

Στη συνέχεια έγιναν κάποιες ρυθμίσεις ώστε να ενσωματωθούν οι επιλογές αλληλεπίδρασης με διάφορα VR Headsets και τα χειριστήριά τους, όμως αυτό που χρησιμοποιήθηκε κατά κύριο λόγο ήταν το HTC Vive Pro 2 [23].

Στην σκηνή του παιχνιδιού, προστέθηκε ένα XR Origin (XR Rig) που αντιπροσωπεύει τον χρήστη και το headset με τα χειριστήρια. Περιέχει την κάμερα του παιχνιδιού, σε first-person view, δηλαδή από τα μάτια του παίκτη, εμπλουτισμένη με έναν Tracked Pose Driver, που λαμβάνει υπόψιν την ύπαρξη δύο οθονών που υπάρχουν στα VR headsets (μία για κάθε μάτι). Περιέχει επίσης ένα αντικείμενο για κάθε χειριστήριο, Left Controller & Right Controller. Προστέθηκε επίσης ένας XR Interaction Manager [24] που είναι απαραίτητος για την αλληλεπίδραση μεταξύ Interactors και Interactables, δηλαδή αντικειμένων που προκαλούν αλληλεπίδραση, και αντικειμένων που δέχονται αλληλεπίδραση. Έτσι τα πρώτα πρέπει να δημιουργούνται με αντίστοιχο script XR Interactor και τα δεύτερα με XR Interactable. Προστέθηκε ένα Locomotion System για να διαχειρίζεται την ομαλή προσομοίωση κίνησης του headset και των controllers. Προστέθηκε ένα Teleportation Area που να δρα ως το πάτωμα στο οποίο στέκεται ο παίκτης, αλλά και το οποίο παρέχει κάποιες λειτουργίες όσον αφορά τη μετακίνηση του παίκτη στο VR περιβάλλον. Τέλος, κάθε νέα σκηνή δημιουργείται αυτόματα με μία τεχνητή πηγή φωτός, το Directional Light, και με μία Κεντρική Κάμερα, η οποία διαγράφηκε.

Όλα τα παραπάνω συστατικά παρέχονται από τις προαναφερθείσες βιβλιοθήκες και είναι απαραίτητα για το ξεκίνημα ενός project που ενσωματώνει VR συσκευές. Υπέστηκαν κάποιες αλλαγές και ρυθμίσεις για να χρησιμεύουν στο παρόν εγχείρημα. όπως η απενεργοποίηση του αριστερού χειριστηρίου, κ.α. Αυτά τα αντικείμενα τοποθετούνται στην Ιεραρχία της σκηνής στο Unity, όπως φαίνεται στην Εικόνα 4.1.



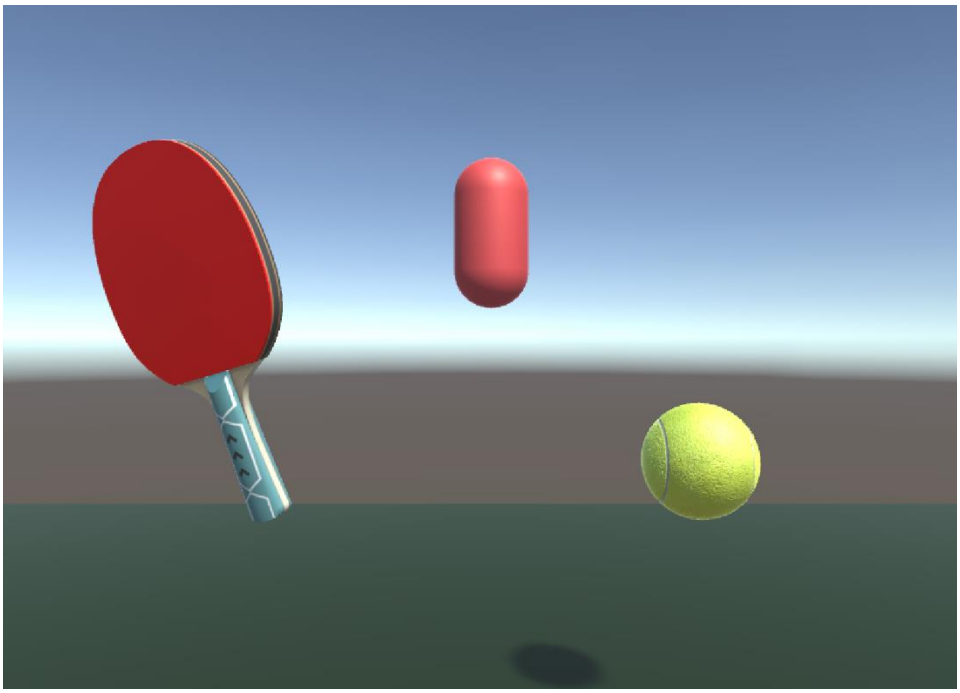
Εικόνα 4.1: Η ιεραρχία της σκηνής με τα βασικά συστατικά XR.

4.2 Περιβάλλον

Το πρώτο στήσιμο του παιχνιδιού ήταν μία υποτυπώδης σκηνή για τις αρχικές δοκιμές των συστατικών. Δημιουργήθηκε ένα κυλινδρικό αντικείμενο από το οποίο θα εκτοξεύεται η μπάλα προς τη θέση του χρήστη. Το αντικείμενο καλύπτεται από ένα material το οποίο θα μπορεί να αλλάζει χρώμα με βάση

την επιτυχία/αποτυχία της απόκρουσης, και αργότερα με βάση την απόδοση στο testing της εκπαίδευσης. Ονομάστηκε shooter και εμφανίζεται στην Εικόνα 4.2.

Επιπλέον προστέθηκε ένα αντικείμενο ρακέτας του ping-pong και ένα αντικείμενο μπάλας του τένις, κατεβασμένα δωρεάν από πλατφόρμες 3D μοντέλων. Η ρακέτα εισήχθη ως παιδί του right controller. Και στα δύο αντικείμενα προστέθηκε ένα συστατικό Rigidbody για τις λειτουργίες της φυσικής υπόστασής τους στο παιχνίδι (βαρύτητα, τριβή, σύγκρουση, κλπ), και ένας collider για την ανίχνευση των συγκρούσεών τους με άλλα αντικείμενα με colliders. Αποθηκεύτηκαν και τα δύο ως prefabs για μελλοντική τους χρήση στο project. Στη ρακέτα προστέθηκε ένα script XR Grab interactable, για να πιάνεται από το χειριστήριο, δηλαδή από το χέρι του παίκτη σαν προέκταση του χειριστηρίου.



Εικόνα 4.2: Η σκηνή demo στο Unity με το Shooter, τη ρακέτα και τη μπάλα.

Για το αντικείμενο Shooter γράφτηκε ένα script (Εικόνα 4.3) που δέχεται ως στοιχεία το αντικείμενο της μπάλας, το transform του ίδιου του shooter (το σώμα του), την ταχύτητα εκτόξευσης της μπάλας, και μια δύναμη που δίνει κυρτότητα στη ρίψη ώστε να μην είναι πολύ απότομη και ευθεία. Το script περιέχει τη μέθοδο ShootBall() η οποία δημιουργεί μια νέα υπόσταση (instance) του αντικειμένου ball με την τοποθεσία και περιστροφή του shooter, βρίσκει την θέση της κάμερας του χρήστη στους άξονες y & z, και δίνει την επιλεγμένη ταχύτητα στη μπάλα. Ταυτόχρονα παράγεται και μια τυχαία float μεταβλητή cameraX μεταξύ των αριθμών -1.00 και 1.00, που χρησιμεύει στο να εκτοξευθεί η μπάλα κάπου κοντά στον χρήστη αλλά όχι πάντα στο ίδιο σημείο στον άξονα x. Αυτός ο μηχανισμός προσθέτει πολυπλοκότητα στο σύστημα, και ευθύνεται για τυχούσες δυσκολίες στην εκπαίδευση, εφόσον

περιλαμβάνει τυχαιότητα, αλλά και εξάρτηση στα δεδομένα. Αυτή η μέθοδος ενεργοποιείται από το άλλο script που γράφτηκε για τη ρακέτα.

```
public void ShootBall()
{
    if (ball != null && GameObject.FindObjectsOfType<Ball>().Length == 0)
    {
        // Instantiate the ball at the starting point
        GameObject newBall = Instantiate(ball, shooter.position, Quaternion.identity);

        // Calculate the direction towards the main camera on y & z
        cameraX = UnityEngine.Random.Range(-xDist, xDist);
        Vector3 xAndZtoCamera = new Vector3(cameraX, Camera.main.transform.position.y, Camera.main.transform.position.z);

        Vector3 directionToCamera = xAndZtoCamera - shooter.position;
        Vector3 upwardComponent = Vector3.up * upwardForce; // Adjust the upward force
        Vector3 combinedDirection = (directionToCamera + upwardComponent).normalized;

        // Set the initial velocity of the ball
        Rigidbody ballRb = newBall.GetComponent<Rigidbody>();
        ballRb.velocity = combinedDirection * speed;

        shots++;
    }
}
```

Εικόνα 4.3: Η μέθοδος ShootBall() του script Shooter.cs

Για το αντικείμενο Racket γράφτηκε ένα script (Εικόνα 4.4) που δέχεται ως στοιχείο το αντικείμενο shooter. Στο ξεκίνημα της εφαρμογής, αποδίδεται μία ενέργεια που ονομάστηκε BallShoot στο συγκεκριμένο κουμπί Grip Button που υπάρχει στα χειριστήρια, και είναι αντιστοιχισμένο στην συσκευή μέσω της βιβλιοθήκης Unity Engine Input System. Με τη μέθοδο OnButtonPress() εκτελείται η μέθοδος ShootBall() του script Shooter.cs. Με αυτόν τον τρόπο, ο χρήστης έχει τον έλεγχο της εκτόξευσης της μπάλας, και έτσι κατά τη διάρκεια της καταγραφής των δεδομένων που θα ακολουθήσει, η διαδικασία θα διεξαχθεί πιο ομαλά, χωρίς περιττές αλληλεπιδράσεις.

```
public Shooter shooter;
private InputAction ballShootAction;

Unity Message | 0 references
private void Start()
{
    // Create a new action for the button press
    ballShootAction = new InputAction("BallShoot", binding:
        "<XRController>{RightHand}/{GripButton}", type: InputActionType.Button);

    // Subscribe to the button press event
    ballShootAction.performed += OnButtonPress;
    ballShootAction.Enable();
}

1 reference
private void OnButtonPress(InputAction.CallbackContext context)
{
    shooter.ShootBall();
}
```

Εικόνα 4.4: Το script Racket.cs

Τέλος, δημιουργήθηκε στη σκηνή ένας αόρατος τοίχος που τοποθετήθηκε ακριβώς πίσω από την κάμερα του χρήστη ώστε να ακολουθεί την κίνησή του. Έχει μόνο έναν collider, και χρησιμεύει στην ανίχνευση συγκρούσεων με τις μπάλες που θα εκτοξευθούν αλλά δεν θα χτυπηθούν από τον χρήστη. Αυτή η λειτουργία ήταν απαραίτητη καθώς οι κινήσεις του χειριστηρίου θα καταγράφονται σε κάθε frame για όσο υπάρχει η μπάλα στη σκηνή, όπως θα αναλυθεί στο επόμενο κεφάλαιο, και θα πρέπει η μπάλα να καταστρέφεται σε περίπτωση αστοχίας.

4.3 Καταγραφή δεδομένων

Τα δεδομένα που απαιτούνται για την εκπαίδευση του μοντέλου MM πρέπει να είναι μεγάλου πλήθους για την όσο το δυνατόν ακριβέστερη εκπαίδευσή του. Όπως θα αναλυθεί και στο Κεφάλαιο 5.1, δεν βρέθηκαν δωρεάν και ανοικτά δεδομένα που να αφορούν το ζητούμενο της εργασίας. Το πεδίο ενδιαφέροντος της εργασίας αφορά την κίνηση ενός χειριστηρίου VR και την ταξινόμησή της ως προς την ορθότητα σε κάποιο πλαίσιο. Οπότε και αναζητήθηκαν αριθμητικά δεδομένα που να αφορούν σε οποιοδήποτε επίπεδο την κίνηση των χειριστηρίων. Για να υπάρχει μια πιο σαφής εικόνα στην αναζήτηση των δεδομένων, ενσωματώθηκε μέσα στο παιχνίδι ένας μηχανισμός που να μετρά και να καταγράφει την κίνηση του χειριστηρίου. Με αυτόν τον τρόπο, εξήχθη μια πληρέστερη εικόνα της μορφής των δεδομένων.

Με βάση τα χαρακτηριστικά στα οποία έχει πρόσβαση η βιβλιοθήκη *UnityEngine.XR*, μπορούν να καταγραφούν διάφορα στοιχεία της κίνησης του χειριστηρίου, όπως η σχετική τοποθεσία, η σχετική περιστροφή, και η ταχύτητα κίνησης. Έτσι μπορεί να καταγραφεί η κίνηση της ρακέτας μέσα στο σενάριο χρήσης. Βεβαίως, δεν θα ήταν ωφέλιμο να καταγράφονται όλα αυτά τα στοιχεία σε χρόνο όπου

η ρακέτα θα ήταν αδρανής και δεν θα υπήρχε κάποια μπάλα για να πετύχει ο χρήστης. Οπότε θεωρήθηκε αποδοτικό να καταγράφονται από το αντικείμενο της ίδιας της μπάλας, καθ' όλη τη διάρκεια της ύπαρξής της στην σκηνή. Δόθηκε λοιπόν η δυνατότητα στον χρήστη να ξεκινά τη διαδικασία εκτόξευσης άρα και καταγραφής, με το πάτημα ενός κουμπιού του χειριστηρίου.

Στην Εικόνα 4.5 παρατίθεται ο κώδικας του αντικειμένου Ball στην μέθοδο LateUpdate(). Η μέθοδος αυτή της βιβλιοθήκης Unity καλείται στο τέλος κάθε frame κατά το οποίο υπάρχει το αντικείμενο στη σκηνή.

```
private void LateUpdate()
{
    if (!_rightController.isValid)
    {
        InitializeInputDevices();
    }

    if (racketInvolved)
    {
        if (_rightController.TryGetFeatureValue(UnityEngine.XR.
            CommonUsages.deviceVelocity, out Vector3 rightVelocity))
        {
            // Get the position and rotation of the controller
            Vector3 controllerPosition = xrController.transform.position;
            Quaternion controllerRotation = xrController.transform.rotation;

            csvWriter.Write(";" + rightVelocity.magnitude + ";");
            csvWriter.Write(controllerPosition.x + ";" + controllerPosition.y +
                ";" + controllerPosition.z + ";");
            csvWriter.Write(controllerRotation.eulerAngles.x + ";" +
                controllerRotation.eulerAngles.y + ";" + controllerRotation.eulerAngles.z);

        }
    }
}
```

Εικόνα 4.5: Η μέθοδος LateUpdate() του script Ball.cs

Το αντικείμενο csvWriter είναι της κλάσης StreamWriter της βιβλιοθήκης System.IO και διαχειρίζεται το άνοιγμα και το κλείσιμο ενός αρχείου .csv, και την καταγραφή των δεδομένων σε αυτό. Όπως συμβαίνει συνήθως με .csv αρχεία, τα δεδομένα χωρίζονται με κάποιο τρόπο μεταξύ τους, και στο παρόν εγχείρημα χωρίζονται από τον χαρακτήρα " ; ". Αποθηκεύονται τα τριών ειδών δεδομένα που αναφέρθηκαν παραπάνω, η σχετική θέση, η σχετική περιστροφή, και η ταχύτητα κίνησης της ρακέτας.

Το άνοιγμα του αρχείου συμβαίνει όταν δημιουργηθεί η μπάλα στο περιβάλλον, δηλαδή κατά την εκτόξευσή της, στη μέθοδο Start() του script Ball.cs, και εισάγεται μία νέα καταχώρηση ως γραμμή.

```
void Start()
{
    shooter = GameObject.Find("Shooter");
    xrController = GameObject.Find("Right Controller").GetComponent<XRController>();

    csvWriter = new StreamWriter("data.csv", true);
    csvWriter.Write(Environment.NewLine);
}
```

Εικόνα 4.6: Η μέθοδος Start() του script Ball.cs

Το κλείσιμο του αρχείου συμβαίνει όταν καταστραφεί η μπάλα, είτε από την επιτυχή σύγκρουσή της με τη ρακέτα, είτε από τον αόρατο τοίχο πίσω από τον παίκτη, αν αποτύχει να την χτυπήσει. Κλείνει στη μέθοδο OnCollisionEnter(Collision collision) της βιβλιοθήκης Unity. Η μέθοδος καλείται με τη σύγκρουση της μπάλας με οποιοδήποτε στερεό αντικείμενο που κατέχει κάποιον collider. Εκεί ελέγχεται αν η μπάλα συγκρούστηκε με τη ρακέτα ή με τον τοίχο/πάτωμα, αντίστοιχα χρωματίζεται το αντικείμενο shooter ως ένδειξη επιτυχίας ή αποτυχίας, γράφεται 0 ή 1 στο τελευταίο κελί της σειράς της προσπάθειας στο .csv αρχείο, καταστρέφεται η μπάλα, και κλείνει το αρχείο .csv (Εικόνα 4.7). Τίθεται επιπλέον και μια Boolean μεταβλητή, που απενεργοποιείται σε περίπτωση επαφής της μπάλας με τη ρακέτα, ώστε να μην καταγράφονται αργότερα στην μέθοδο LateUpdate() οι κινήσεις της ρακέτας μετά από την επαφή, καταγράφοντας περιττά δεδομένα.

```
private void OnCollisionEnter(Collision collision)
{
    // Check if the ball collides with a wall
    if (collision.gameObject.CompareTag("Wall") || collision.gameObject.CompareTag("Ground"))
    {
        if (racketInvolved)
        {
            shooter.GetComponent<Renderer>().material = red;
            csvWriter.Write(";0");

            shooterScript.RemovePoints();
            Debug.Log("Ball collided with a wall!");

            csvWriter.Close();
        }
        Destroy(gameObject);
    }

    // Check if the ball collides with the racket
    if (collision.gameObject.CompareTag("Racket"))
    {
        shooter.GetComponent<Renderer>().material = green;
        csvWriter.Write(";1");

        shooterScript.AddPoints();
        Debug.Log("Ball collided with the racket!");

        csvWriter.Close();

        racketInvolved = false;
    }
}
```

Εικόνα 4.7: Η μέθοδος OnCollisionEnter του script Ball.cs

5

Δεδομένα

Το ζητούμενο της εργασίας αφορά την ανάπτυξη κάποιου μοντέλου ΜΜ για την αναγνώριση κινήσεων χειριστηρίων VR, σε οποιοδήποτε φάσμα. Δηλαδή είτε την ταυτοποίηση χρηστών, είτε την αναγνώριση κινήσεων, είτε την ταξινόμηση κινήσεων σε ορθές ή μη. Αναλόγως το ζητούμενο πιο συγκεκριμένα, άλλο είδος δεδομένων θα χρησιμοποιούνταν για την εκπαίδευση, εφόσον άλλο αποτέλεσμα θα ήταν θεμιτό και ωφέλιμο. Στην αρχή της εκπόνησης της εργασίας έγινε μια προκαταρκτική αναζήτηση ελεύθερων δεδομένων από πηγές του διαδικτύου, για την πρώτη εκτίμηση του σκοπού της εργασίας. Αναλόγως του είδους των δεδομένων, απαιτούνται και διαφορετικές υλικές υποδομές. Για την αναγνώριση στάσης και πόζας του χρήστη για παράδειγμα, απαιτούνται επιπλέον εξωτερικοί αισθητήρες και κάμερες. Μία διαδεδομένη μέθοδος για την αναγνώριση κινήσεων χειριστηρίων σε VR είναι η καταγραφή εικόνας από το κράνος των χειριστηρίων, και η εκπαίδευση ενός μοντέλου πάνω σε αυτές τις εικόνες. Όπως αναλύθηκε και στο κεφάλαιο 2, τα χειριστήρια πολλές φορές βγαίνουν εκτός του FOV του κράνους, και άρα δεν μπορεί να καταγραφεί όλο το εύρος των κινήσεων μόνο με τη χρήση των καμερών του κράνους. Μία άλλη προσέγγιση είναι η καταγραφή των δεδομένων θέσης των χειριστηρίων, και η εκπαίδευση πάνω σε αυτά. Αυτή είναι και η προσέγγιση που επιλέχθηκε για το παρόν εγχείρημα, με γνώμονες τη χρήση λιγότερων υλικών μέσων, αλλά και την πρόθεση να παραχθούν ωφέλιμα αποτελέσματα με ελαχιστοποιημένα features.

5.1 Αναζήτηση ελεύθερων δεδομένων

Για την πληρέστερη και μεγαλύτερης ακρίβειας εκπαίδευση μοντέλων ΜΜ, απαιτείται μεγάλος όγκος δεδομένων. Στην προκαταρκτική αναζήτηση ελεύθερων δεδομένων στο διαδίκτυο, αναζητήθηκε κάποιο σετ δεδομένων με μεγάλο πλήθος δειγμάτων, πάνω σε αριθμητικά δεδομένα κινήσεων

χειριστηρίων VR. Η αναζήτηση δεν επέφερε γόνιμα αποτελέσματα, παρόλα αυτά σε αυτό το κεφάλαιο παρατίθενται κάποια σχετικά αποτελέσματα, που θα μπορούσαν να χρησιμοποιηθούν σε μία παρόμοια μελέτη.

Οι Παπαδόπουλος et al [25] συγκέντρωσαν ένα σετ δεδομένων από 7600 κινήσεις χειριστηρίων που αντιστοιχούν σε 17 είδη κινήσεων. Συγκέντρωσαν επίσης δεδομένα για 11 χειρονομίες ενός χεριού και δεδομένα για 2 χειρονομίες δύο χεριών. Τα αρχεία που παρέχονται είναι σε μορφή .bin και δεν βρέθηκε κάποιο documentation ως προς την αξιοποίησή τους. Η μέθοδος που τελικά ακολουθήθηκε για την αξιοποίηση των δεδομένων της παρούσας εργασίας ήταν μια ήδη δοκιμασμένη μέθοδος που δεν αξιοποιούσε αρχεία .bin αλλά δεδομένα στη μορφή αρχείων .csv.

Οι Ponton et al [26] εξέδωσαν μία εργασία στην οποία συγκέντρωσαν δεδομένα πόζας του χρήστη VR, για να δημιουργήσουν animations σε πραγματικό χρόνο, χρησιμοποιώντας μόνο τα δεδομένα από το κράνος και τα χειριστήρια. Το κύριο συστατικό της μεθόδου τους ονομάζεται Motion Matching. Αφορά την πρόβλεψη των κινήσεων του χρήστη με σκοπό την ταυτόχρονη κίνηση του avatar του μέσα στο εικονικό περιβάλλον. Τα δεδομένα τους δεν βρέθηκαν διαθέσιμα.

Οι Pandey et al [27] συγκέντρωσαν περίπου 500.000 ζεύγη εικόνων από το VR headset τους στις οποίες φαίνεται το κάθε χέρι του χρήστη με το χειριστήριο. Τα features με τα οποία επισημαίνεται κάθε φωτογραφία είναι το αν είναι αριστερή ή δεξιά, το timestamp της λήψης της, η χειρονομία που πραγματοποιείται, η θέση και η περιστροφή του χειριστηρίου, τα δεδομένα της κάμερας, και το αν περιέχει errors στην ανίχνευση του χειριστηρίου από την κάμερα. Συμπληρώνοντας μία φόρμα μπορεί κάποιος να ζητήσει το σετ δεδομένων από τους συγγραφείς, όμως για το παρόν εγχείρημα δεν χρησιμεύουν δεδομένα εικόνας, μόνο αριθμητικά δεδομένα που θα δείχνουν την κίνηση των χειριστηρίων.

5.2 Προετοιμασία δεδομένων

Όπως αναλύθηκε στο κεφάλαιο 4.3, τα δεδομένα συλλέγονται μέσω ενός script στο Unity, και για κάθε ρίψη της μπάλας. Τα δεδομένα αποθηκεύονται σε ένα .csv αρχείο, με κάθε του γραμμή να αναπαριστά μία ρίψη. Το μήκος της γραμμής δεν είναι ορισμένο, μιας και η κίνηση καταγράφεται για κάθε frame του παιχνιδιού μέχρι να καταστραφεί η μπάλα, και αυτό μπορεί να συμβεί σε οποιοδήποτε frame, ανάλογα με τις κινήσεις του παίκτη. Στο τελευταίο κελί της κάθε γραμμής αποθηκεύεται μια δυαδική πληροφορία, ‘1’ για την επιτυχή απόκρουση της μπάλας, και ‘0’ για την αποτυχία απόκρουσης. Στο πρώτο κελί της κάθε γραμμής αποθηκεύεται η float μεταβλητή που παράγεται τυχαία σε κάθε ρίψη για το σημείο στον άξονα x όπου θα εκτοξευθεί η μπάλα. Τα στοιχεία που καταγράφονται είναι:

- Η τυχαία μεταβλητή cameraX που παράγεται κατά την εκτόξευση

- Η ταχύτητα της ρακέτας στο συγκεκριμένο frame
- Τρία πεδία για την θέση της ρακέτας στο συγκεκριμένο frame
    - position X
    - position Y
    - position Z
- Τρία πεδία για την στροφή που έχει η ρακέτα στο συγκεκριμένο frame
    - rotation X
    - rotation Y
    - rotation Z
- Η κλάση της ρίψης, που μπορεί να είναι ‘1’ ή ‘0’.

Τα πρωτογενή δεδομένα εμφανίζονται στην Εικόνα 5.1.

	A	B	C	D	E	F	G	H	I	J	K	L
1	0,791759	0,039009	0,091295	0,823645	-5,79676	306,618	13,87055	333,9188	0,060011	0,090306	0,823528	-5,7971
2	0,990544	0,327448	0,241255	1,038731	-5,85665	278,8204	102,5932	245,3826	0,324796	0,239274	1,035447	-5,85778
3	0,914237	0,360159	0,272477	1,065175	-5,89488	284,6675	132,7682	234,692	0,344957	0,268343	1,066686	-5,89336
4	-0,84616	0,86488	0,289272	1,032388	-5,72643	294,4762	20,89602	332,8517	0,817217	0,283271	1,03794	-5,72909
5	0,774854	0,326608	-0,21534	0,943227	-5,65547	327,4941	288,0872	125,9375	0,284546	-0,21405	0,944147	-5,65745
6	0,627442	0,624612	0,226573	1,176661	-6,01683	310,1558	189,8504	193,7238	0,542212	0,22689	1,172895	-6,01822
7	0,883944	0,934584	0,264665	1,07231	-5,78415	287,4043	22,55418	331,2762	0,9131	0,262024	1,073951	-5,79287
8	0,895715	0,736605	0,229582	1,07857	-5,91922	297,688	178,6029	195,226	0,666545	0,226515	1,08005	-5,92407
9	0,307526	1,070321	0,293588	0,912099	-5,60297	322,6825	357,5216	350,306	1,113447	0,290127	0,922113	-5,61218
10	-0,5913	0,968236	0,17332	1,033589	-5,78027	278,653	188,0071	185,9089	0,891188	0,174253	1,035791	-5,78846

Εικόνα 5.1: Μέρος των πρωτογενών δεδομένων.

Για την καλύτερη ανάγνωση, κατανόηση και ανάλυση των δεδομένων, αλλά και για την μελλοντική τους επεξεργασία από το μοντέλο MM, τα δεδομένα χρειάστηκε να υποστούν κάποια προετοιμασία. Αρχικά έπρεπε να διαγραφούν οι κενές γραμμές που ίσως να παράχθηκαν κατά την καταγραφή. Επιπλέον, η κλάση της ρίψης, όντας αποθηκευμένη στο τέλος της γραμμής με αόριστο μήκος, θα αποτελούσε πρόβλημα μελλοντικά στην αναζήτησή της, μιας και είναι το πιο σημαντικό μέρος της καταγραφής. Επομένως έπρεπε να μεταφερθεί μαζικά το τελευταίο κελί στην αρχή του πίνακα, και να αποτελεί την πρώτη στήλη. Τα δεδομένα μετά την πρώτη τους προεπεξεργασία φαίνονται στην Εικόνα 5.2.

	A	B	C	D	E	F	G	H	I	J	K	L
1	1	0,791759	0,039009	0,091295	0,823645	-5,79676	306,618	13,87055	333,9188	0,060011	0,090306	0,823528
2	1	0,990544	0,327448	0,241255	1,038731	-5,85665	278,8204	102,5932	245,3826	0,324796	0,239274	1,035447
3	1	0,914237	0,360159	0,272477	1,065175	-5,89488	284,6675	132,7682	234,692	0,344957	0,268343	1,066686
4	1	-0,84616	0,86488	0,289272	1,032388	-5,72643	294,4762	20,89602	332,8517	0,817217	0,283271	1,03794
5	0	0,774854	0,326608	-0,21534	0,943227	-5,65547	327,4941	288,0872	125,9375	0,284546	-0,21405	0,944147
6	1	0,627442	0,624612	0,226573	1,176661	-6,01683	310,1558	189,8504	193,7238	0,542212	0,22689	1,172895
7	1	0,883944	0,934584	0,264665	1,07231	-5,78415	287,4043	22,55418	331,2762	0,9131	0,262024	1,073951
8	1	0,895715	0,736605	0,229582	1,07857	-5,91922	297,688	178,6029	195,226	0,666545	0,226515	1,08005
9	1	0,307526	1,070321	0,293588	0,912099	-5,60297	322,6825	357,5216	350,306	1,113447	0,290127	0,922113
10	1	-0,5913	0,968236	0,17332	1,033589	-5,78027	278,653	188,0071	185,9089	0,891188	0,174253	1,035791

Εικόνα 5.2: Μέρος των δεδομένων μετά από την πρώτη τους προεπεξεργασία.

Τέλος, σε αυτό το σημείο τα δεδομένα δεν έχουν ετικέτες. Είναι δηλαδή αόριστοι αριθμοί, δυσανάγνωστοι, και με κανένα ωφέλιμο στοιχείο ως προς τη μελλοντική τους επεξεργασία για την εκπαίδευση του μοντέλου MM. Επομένως προστέθηκαν επικεφαλίδες που ονοματίζουν το τι αναπαριστά κάθε στήλη του πίνακα, μέχρι το πέρας των στηλών. Η τελική μορφή των δεδομένων φαίνονται στην Εικόνα 5.3.

	A	B	C	D	E	F	G	H	I	J	K	L
1	class	cameraX	velocity	posX	posY	posZ	rotX	rotY	rotZ	velocity	posX	posY
2	1	0,791759	0,039009	0,091295	0,823645	-5,79676	306,618	13,87055	333,9188	0,060011	0,090306	0,823528
3	1	0,990544	0,327448	0,241255	1,038731	-5,85665	278,8204	102,5932	245,3826	0,324796	0,239274	1,035447
4	1	0,914237	0,360159	0,272477	1,065175	-5,89488	284,6675	132,7682	234,692	0,344957	0,268343	1,066686
5	1	-0,84616	0,86488	0,289272	1,032388	-5,72643	294,4762	20,89602	332,8517	0,817217	0,283271	1,03794
6	0	0,774854	0,326608	-0,21534	0,943227	-5,65547	327,4941	288,0872	125,9375	0,284546	-0,21405	0,944147
7	1	0,627442	0,624612	0,226573	1,176661	-6,01683	310,1558	189,8504	193,7238	0,542212	0,22689	1,172895
8	1	0,883944	0,934584	0,264665	1,07231	-5,78415	287,4043	22,55418	331,2762	0,9131	0,262024	1,073951
9	1	0,895715	0,736605	0,229582	1,07857	-5,91922	297,688	178,6029	195,226	0,666545	0,226515	1,08005
10	1	0,307526	1,070321	0,293588	0,912099	-5,60297	322,6825	357,5216	350,306	1,113447	0,290127	0,922113

Εικόνα 5.3: Μέρος των δεδομένων μετά από τη δεύτερή τους προεπεξεργασία.

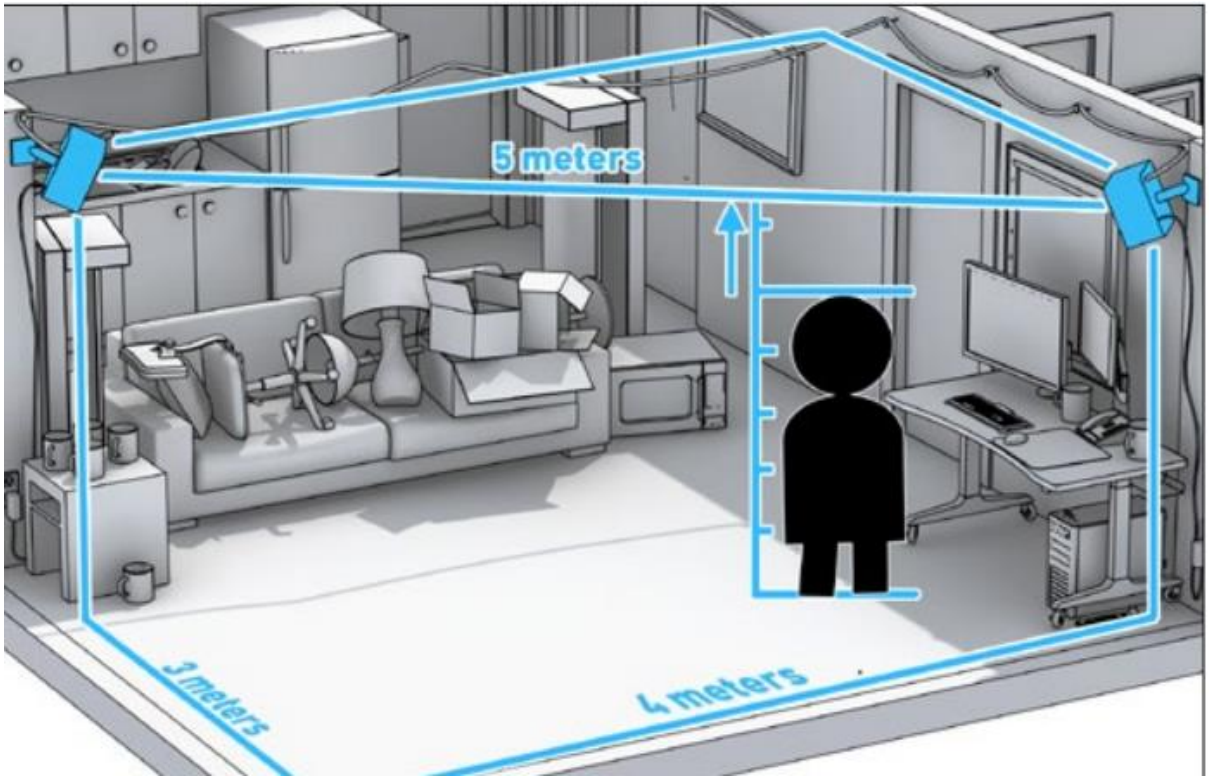
Αυτή η προετοιμασία επετεύχθη με τη συγγραφή δύο scripts στη γλώσσα Python με τον editor Visual Studio Code [28]. Στο Παράρτημα Α βρίσκεται ο κώδικας των 2 scripts.

5.3 Διαδικασία συλλογής

Για τη διαδικασία συλλογής των δεδομένων απαιτούταν το στήσιμο ενός χώρου για τη χρήση συσκευής VR. Οι ασύρματες συσκευές μπορούν να χρησιμοποιηθούν σε οποιοδήποτε σημείο στο χώρο, όμως έχουν χαμηλότερες αποδόσεις λόγω των απωλειών του μέσου διάδοσης της πληροφορίας (αέρας). Επιπλέον, έχοντας μεταβλητά στοιχεία τοποθεσίας, οποιοδήποτε πείραμα που αφορά τις μετρήσεις των κινήσεων του χρήστη, θα εισάγει πολυπλοκότητα στα αποτελέσματα. Για τους παραπάνω λόγους,

χρησιμοποιήθηκε ενσύρματη συσκευή VR, που συνήθως συνοδεύεται από ένα rig με σταθμούς βάσης οι οποίοι ανιχνεύουν την χωροταξία αλλά και την ακριβή θέση του χρήστη και των χειριστηρίων μέσα στο περιβάλλον.

Οι σταθμοί βάσεις τοποθετούνται διαγώνια στο χώρο (αν είναι 2) ή σχηματίζουν ένα τετράγωνο (αν είναι 4), περικλείοντας το χώρο ενέργειας του χρήστη. Το πάτωμα του χώρου πρέπει να είναι άδειο ώστε ο χρήστης να είναι ανεμπόδιστος στις κινήσεις του και το εγχείρημα να είναι ασφαλές. Το κράνος VR συνδέεται με τον υπολογιστή, προσφέροντας μια ομαλή διάδοση της πληροφορίας από την κάρτα γραφικών στους φακούς των γυαλιών. Τα χειριστήρια συνήθως δένονται με λουράκια στα χέρια του χρήστη για λόγους ασφαλείας. Στην Εικόνα 5.5 παρουσιάζεται ένα ενδεικτικό σχεδιάγραμμα για την δομή του χώρου αλληλεπίδρασης με ένα πλήρες VR set.

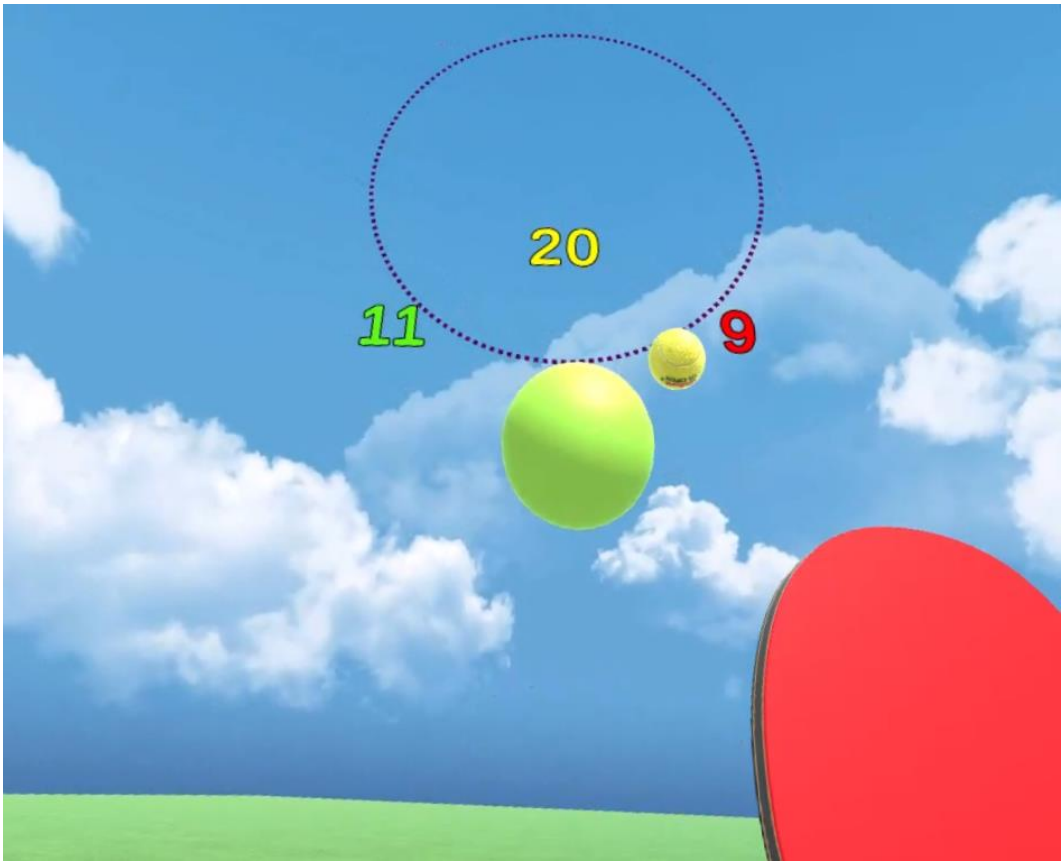


Εικόνα 5.4: Στήσιμο χώρου για VR με σταθμούς βάσης.

Η συλλογή δεδομένων πραγματοποιήθηκε μετά τη δημιουργία του πρώτου στησίματος της σκηνής στο Unity, και ολοκληρώθηκε μετά από αρκετές ρίψεις. Τρεις διαφορετικοί χρήστες πραγματοποίησαν ρίψεις με τη ρακέτα στο εικονικό περιβάλλον, σε διαφορετικούς χρόνους. Τα δεδομένα έπρεπε να έχουν μια συγκεκριμένη δομή αλλά και να περιέχουν μια ποικιλομορφία ώστε να εκπαιδευτεί επιτυχώς το μοντέλο. Το σημείο που στέκεται ο χρήστης έπρεπε να είναι το ίδιο στον άξονα Z (η απόσταση του χρήστη από το shooter) αλλά και στον άξονα X (όχι αριστερά ή δεξιά, στο κέντρο). Το ύψος του κάθε

χρήστη ήταν διαφορετικό, προσδίδοντας διαφορετικότητα στα δεδομένα. Επίσης κάθε ρίψη έπρεπε να πραγματοποιηθεί με το δεξί χέρι του χρήστη.

Για τη διευκόλυνση του πειράματος, μετρούνταν οι επιτυχίες και οι αποτυχίες του χρήστη, ώστε να μπορέσει να δοθεί στο μοντέλο ποικιλία αποτελεσμάτων (επιτυχίας/αποτυχίας). Αν ο χρήστης κατάφερνε πολλές επιτυχίες στις αποκρούσεις, λάμβανε την οδηγία να αποτύχει σε μερικές, προσπαθώντας όμως ακόμη για την επιτυχή απόκρουση. Όπως αναφέρθηκε παραπάνω, ο χρήστης είχε τον έλεγχο της εκκίνησης των ρίψεων, πατώντας το πλαϊνό κουμπί του χειριστηρίου. Τα δεδομένα καταγράφονται από εκείνη τη χρονική στιγμή, μέχρι να ακουμπήσει η μπάλα τη ρακέτα, ή τον τοίχο πίσω από τον χρήστη. Οι συλλογές πραγματοποιούνταν ανά 100 περίπου ρίψεις, και συνολικά συλλέχθηκαν 1000 καταχωρήσεις ρίψεων. Στην Εικόνα 5.5 παρουσιάζεται ένα στιγμιότυπο μέσα από το εικονικό περιβάλλον κατά τη διαδικασία μίας ρίψης από τον παίκτη.



Εικόνα 5.5: Στιγμιότυπο ρίψης μπάλας στο εικονικό περιβάλλον για τη συλλογή δεδομένων.

6

Αλγόριθμοι

6.1 Σύγκριση αλγορίθμων

Όπως συζητήθηκε και στα προηγούμενα κεφάλαια, υπάρχουν διάφοροι αλγόριθμοι που εμπίπτουν στο παρόν πρόβλημα, είτε αυτοί είναι αλγόριθμοι ΜΜ, είτε βαθιάς μάθησης. Για τα πειράματα δοκιμάστηκαν αρκετοί αλγόριθμοι σε ένα αρχικό μικρό δείγμα των δεδομένων (20 ρίψεις), ώστε να αναδειχθεί αν αυτοί παράγουν μη μηδενικά αποτελέσματα και αν μπορούν να χρησιμοποιηθούν για όλο το πλήθος των δεδομένων που συλλέχθηκαν. Οι αλγόριθμοι ΜΜ που δοκιμάστηκαν ήταν [29]:

- Λογιστική παλινδρόμηση. Ένα στατιστικό μοντέλο που χρησιμοποιείται για την επίλυση προβλημάτων ταξινόμησης στη ΜΜ. Χρησιμοποιεί μία λογιστική συνάρτηση για να εκτιμήσει τις πιθανότητες του αποτελέσματος, κάτι που αναφέρεται και ως η μαθηματικά καθορισμένη σιγμοειδής συνάρτηση (Σχήμα 6.1). Λειτουργεί ομαλά όταν το σετ δεδομένων είναι γραμμικά διαχωρίσιμο, αλλά μπορεί να υπέρ-προσαρμόσει (overfit) σετ δεδομένων υψηλών διαστάσεων. Εκτός από προβλήματα ταξινόμησης, σπανιότερα χρησιμοποιείται και σε προβλήματα παλινδρόμησης. Ένα σημαντικό μειονέκτημα θεωρείται η παραδοχή της γραμμικότητας μεταξύ των εξαρτημένων και ανεξάρτητων μεταβλητών. Στον προκαταρκτικό έλεγχο του μικρού σετ δεδομένων του παρόντος προβλήματος, ο αλγόριθμος έφερε 100% ακρίβεια με εκπαίδευση στο 80% των δεδομένων, και 83.33% ακρίβεια στο 70% των δεδομένων.

$$g(z) = \frac{1}{1 + \exp(-z)}$$

Σχήμα 6.1: Μαθηματικά καθορισμένη σιγμοειδής συνάρτηση

- Δέντρα αποφάσεων. Μια γνωστή μη παραμετρική μέθοδος μάθησης με επίβλεψη. Χρησιμοποιούνται τόσο για ταξινόμηση όσο και για παλινδρόμηση. Οι ID3, C4.5 και CART είναι γνωστοί αλγόριθμοι δέντρων αποφάσεων. Το δέντρο απόφασης ταξινομεί τις περιπτώσεις από τη ρίζα (πάνω) προς τους κόμβους-φύλλα (κάτω). Οι περιπτώσεις ταξινομούνται με τον έλεγχο του χαρακτηριστικού που ορίζεται από τον εν λόγω κόμβο, ξεκινώντας από τον ριζικό κόμβο του δέντρου και στη συνέχεια μετακινούμενοι προς τα κάτω, στο κλαδί του δέντρου που αντιστοιχεί στην τιμή του χαρακτηριστικού. Στον προκαταρκτικό έλεγχο, ο αλγόριθμος έφερε 100% ακρίβεια με εκπαίδευση στο 70% των δεδομένων, όπως και στο 80% των δεδομένων.

- Τυχαίου δάσους (Random Forest). Είναι γνωστή ως μια τεχνική ταξινόμησης ensemble που χρησιμοποιείται στον τομέα της ΜΜ και της επιστήμης των δεδομένων σε διάφορους τομείς εφαρμογών. Η μέθοδος αυτή χρησιμοποιεί την "παράλληλη συνένωση" (parallel ensembling), η οποία προσαρμόζει πολλούς ταξινομητές δέντρων αποφάσεων παράλληλα, σε διαφορετικά υπο-δείγματα σετ δεδομένων και χρησιμοποιεί ψηφοφορία πλειοψηφίας ή μέσους όρους για το τελικό αποτέλεσμα. Ελαχιστοποιεί έτσι το πρόβλημα της υπέρ-προσαρμογής και αυξάνει την ακρίβεια πρόβλεψης και τον έλεγχο. Ως εκ τούτου, το μοντέλο μάθησης RF με πολλαπλά δέντρα αποφάσεων είναι συνήθως πιο ακριβές από ένα μοντέλο που βασίζεται σε ένα μόνο δέντρο αποφάσεων. Για τη δημιουργία μιας σειράς δέντρων απόφασης με ελεγχόμενη διακύμανση, συνδυάζει τη συγκέντρωση bootstrap (bagging) και την τυχαία επιλογή χαρακτηριστικών. Είναι προσαρμόσιμο τόσο σε προβλήματα ταξινόμησης όσο και σε προβλήματα παλινδρόμησης και ταιριάζει καλά τόσο σε κατηγορικές όσο και σε συνεχείς τιμές. Στον προκαταρκτικό έλεγχο, ο αλγόριθμος έφερε 100% ακρίβεια με εκπαίδευση στο 70% των δεδομένων.

- Κ – εγγύτεροι γείτονες (KNN). Είναι ένας αλγόριθμος "μάθησης με βάση το παράδειγμα" ή μη γενικευμένης μάθησης, γνωστός και ως αλγόριθμος "τεμπέλικης μάθησης". Δεν εστιάζει στην κατασκευή ενός γενικού εσωτερικού μοντέλου. Αντίθετα, αποθηκεύει όλες τις περιπτώσεις που αντιστοιχούν σε δεδομένα εκπαίδευσης σε έναν χώρο n διαστάσεων. Ο KNN χρησιμοποιεί δεδομένα και ταξινομεί νέα σημεία δεδομένων με βάση μέτρα ομοιότητας (π.χ. συνάρτηση ευκλείδειας απόστασης). Η ταξινόμηση υπολογίζεται από μια απλή ψηφοφορία πλειοψηφίας των k πλησιέστερων γειτόνων κάθε σημείου. Είναι αρκετά ανθεκτική σε θορυβώδη δεδομένα

εκπαίδευσης και η ακρίβεια εξαρτάται από την ποιότητα των δεδομένων. Το μεγαλύτερο πρόβλημα με το KNN είναι η επιλογή του βέλτιστου αριθμού γειτόνων που πρέπει να ληφθούν υπόψη. Μπορεί να χρησιμοποιηθεί τόσο για ταξινόμηση όσο και για παλινδρόμηση. Στον προκαταρκτικό έλεγχο, ο αλγόριθμος έφερε 100% ακρίβεια με εκπαίδευση στο 70% των δεδομένων.

- Naïve Bayes. Βασίζεται στο θεώρημα του Bayes, με την παραδοχή της ανεξαρτησίας μεταξύ κάθε ζεύγους χαρακτηριστικών. Λειτουργεί καλά και μπορεί να χρησιμοποιηθεί τόσο για δυαδικές όσο και για πολύ-ταξικές κατηγορίες σε πολλές πραγματικές καταστάσεις, όπως η ταξινόμηση εγγράφων ή κειμένων, το φιλτράρισμα ανεπιθύμητης αλληλογραφίας κ.λπ. Καταφέρνει την αποτελεσματική ταξινόμηση των θορυβωδών περιπτώσεων στα δεδομένα και την κατασκευή ενός ισχυρού μοντέλου πρόβλεψης. Το βασικό πλεονέκτημα είναι ότι, σε σύγκριση με πιο εξελιγμένες προσεγγίσεις, χρειάζεται μικρό όγκο δεδομένων εκπαίδευσης για την γρήγορη εκτίμηση των απαραίτητων παραμέτρων. Ωστόσο, η απόδοσή του μπορεί να επηρεαστεί από τις ισχυρές παραδοχές του σχετικά με την ανεξαρτησία των χαρακτηριστικών. Στον προκαταρκτικό έλεγχο, ο αλγόριθμος έφερε 100% ακρίβεια με εκπαίδευση στο 80% των δεδομένων.

- SVM. Σε χώρο υψηλής ή άπειρης διάστασης, μια μηχανή διανυσμάτων υποστήριξης κατασκευάζει ένα υπερεπίπεδο ή ένα σύνολο υπερεπιπέδων. Διαισθητικά, το υπερεπίπεδο, το οποίο έχει τη μεγαλύτερη απόσταση από τα πλησιέστερα σημεία δεδομένων εκπαίδευσης σε κάθε κλάση, επιτυγχάνει ισχυρό διαχωρισμό, καθώς, γενικά, όσο μεγαλύτερο είναι το περιθώριο, τόσο μικρότερο είναι το σφάλμα γενίκευσης του ταξινομητή. Είναι αποτελεσματικό σε χώρους υψηλών διαστάσεων και μπορεί να συμπεριφέρεται διαφορετικά με βάση διαφορετικές μαθηματικές συναρτήσεις, γνωστές ως πυρήνας. Η γραμμική, η πολυωνυμική, η συνάρτηση ακτινικής βάσης (RBF), η σιγμοειδής κ.λπ. είναι οι δημοφιλείς συναρτήσεις πυρήνα που χρησιμοποιούνται στον ταξινομητή SVM. Ωστόσο, όταν το σετ δεδομένων περιέχει περισσότερο θόρυβο, όπως επικαλυπτόμενες κλάσεις-στόχους, το SVM δεν αποδίδει καλά. Στον προκαταρκτικό έλεγχο, ο αλγόριθμος έφερε 83.33% ακρίβεια με εκπαίδευση στο 70% των δεδομένων, και 75% στο 80% των δεδομένων, πολύ χαμηλότερα από τους υπόλοιπους αλγορίθμους.

- Gradient Boosting. Είναι αλγόριθμοι μάθησης ensemble που παράγουν ένα τελικό μοντέλο με βάση μια σειρά από μεμονωμένα μοντέλα, συνήθως δέντρα αποφάσεων. Το gradient χρησιμοποιείται για την ελαχιστοποίηση της συνάρτησης απωλειών, παρόμοια με τον τρόπο που τα νευρωνικά δίκτυα χρησιμοποιούν gradient descent για τη βελτιστοποίηση των βαρών.

Το Extreme Gradient Boosting (XGBoost) αλλά και το LightGBM είναι μορφές gradient boosting που λαμβάνουν υπόψη πιο λεπτομερείς προσεγγίσεις κατά τον προσδιορισμό του καλύτερου μοντέλου. Υπολογίζουν βαθμίδες δεύτερης τάξης της συνάρτησης απώλειας για να ελαχιστοποιήσουν την απώλεια, και προηγμένη κανονικοποίηση, η οποία μειώνει την υπέρ-προσαρμογή και βελτιώνει τη γενίκευση και την απόδοση του μοντέλου. Το XGBoost και το LightGBM είναι γρήγορα στην ερμηνεία τους και μπορούν να χειριστούν καλά τα σετ δεδομένων μεγάλου μεγέθους.

6.2 Αξιοποίηση Νευρωνικών δικτύων

Το παρόν πρόβλημα αφορά Αριθμητικά δεδομένα σε πίνακα (Numerical Tabular data). Η χρήση νευρωνικών δικτύων επιτρέπει την αξιοποίηση προ-εκπαιδευμένων (pre-trained) μοντέλων, τα οποία είναι προσαρμοσμένα σε συγκεκριμένες μορφές δεδομένων, και μπορούν με κάποια επεξεργασία να χρησιμοποιηθούν σε νέα προβλήματα (π.χ. DenseNet, ResNet, BERT) [30]. Η χρήση τους είναι ευρεία σε δεδομένα εικόνας αλλά και κειμένου (NLP). Για αυτό το λόγο, μετά από εκτενή αναζήτηση σε ανοιχτά κανάλια του διαδικτύου όπως το Kaggle [31], δεν βρέθηκε κανένα προ-εκπαιδευμένο μοντέλο που να δέχεται εισόδους αριθμητικών δεδομένων. Έγιναν δοκιμές με μοντέλα για εικόνες μετά από fine-tuning, οι οποίες όμως έφεραν αποτελέσματα παρόμοια με αυτά μιας ρίψης νομίσματος (~50%), όπως επέδειξαν και στασιμότητα κατά την εκπαίδευσή τους. Κάποιες από τις βιβλιοθήκες που δοκιμάστηκαν ήταν οι AutoGluon, TabTransformer, FastAI. Όλες χρησιμοποιούν τις βιβλιοθήκες tensorflow/keras /pytorch.

Μετά την ατελέσφορη απόπειρα χρήσης προ-εκπαιδευμένων μοντέλων, επιχειρήθηκε η δημιουργία ενός απλού νευρωνικού δικτύου με την χρήση των βιβλιοθηκών tensorflow/keras. Οι βιβλιοθήκες που αφορούν την ανάπτυξη τεχνητών ΝΔ περιέχουν έτοιμες μεθόδους οι οποίες μπορούν να συνθέσουν ένα ΝΔ με όσους νευρώνες ή στρώματα νευρώνων χρειάζεται, να είναι συνελικτικά (CNN) ή αναδρομικά (RNN) κ.ο.κ. Η ανάπτυξη του ΝΔ αναλύεται στο υποκεφάλαιο 7.3, όπως και οι απόπειρες εμπλουτισμού του.

Η διαδικασία ανάπτυξης ενός ΝΔ μπορεί να αποδειχθεί αρκετά περίπλοκη, αναλόγως τις απαιτήσεις του προβλήματος. Χρειάζεται ο καθορισμός ποσότητας των στρωμάτων και των νευρώνων, ο ρυθμός και το βήμα εκμάθησης, η επιλογή των συναρτήσεων ενεργοποίησης, και η μέθοδος αξιολόγησης του μοντέλου [32]. Όλα τα παραπάνω ορίζονται ως υπέρ-παράμετροι του μοντέλου (hyperparameters) και χρειάζονται αρκετό fine-tuning ώστε να βρεθεί ο κατάλληλος συνδυασμός που αρμόζει στη λύση του εκάστοτε προβλήματος. Τα ΝΔ αποτελούνται από ένα στρώμα εισόδου, ένα ή περισσότερα κρυφά στρώματα, και ένα στρώμα εξόδου. Αναλόγως την πολυπλοκότητα του δικτύου, το πόσες συναρτήσεις ενεργοποίησης χρησιμοποιεί, και το βάθος του, δηλαδή το πόσα κρυφά στρώματα περιέχει, το ΝΔ υπάγεται στα Ρηχά είτε στα Βαθιά ΝΔ.

Το μοντέλο που αναπτύχθηκε ορίζεται ως ένα Νευρωνικό δίκτυο τροφοδότησης, και πιο συγκεκριμένα Multi-Layer Perceptron (MLP). Είναι ένα σχετικά απλό δίκτυο, με ένα στρώμα εισόδου, δύο κρυφά στρώματα, και ένα στρώμα εξόδου. Κάθε νευρώνας συνδέεται με τον επόμενο και κάθε σύνδεση έχει ένα Βάρος. Περιγράφεται από ένα Sequential μοντέλο του Keras, που επιτρέπει το χτίσιμο του ΝΔ στρώμα με στρώμα (υποκεφάλαιο 7.3).

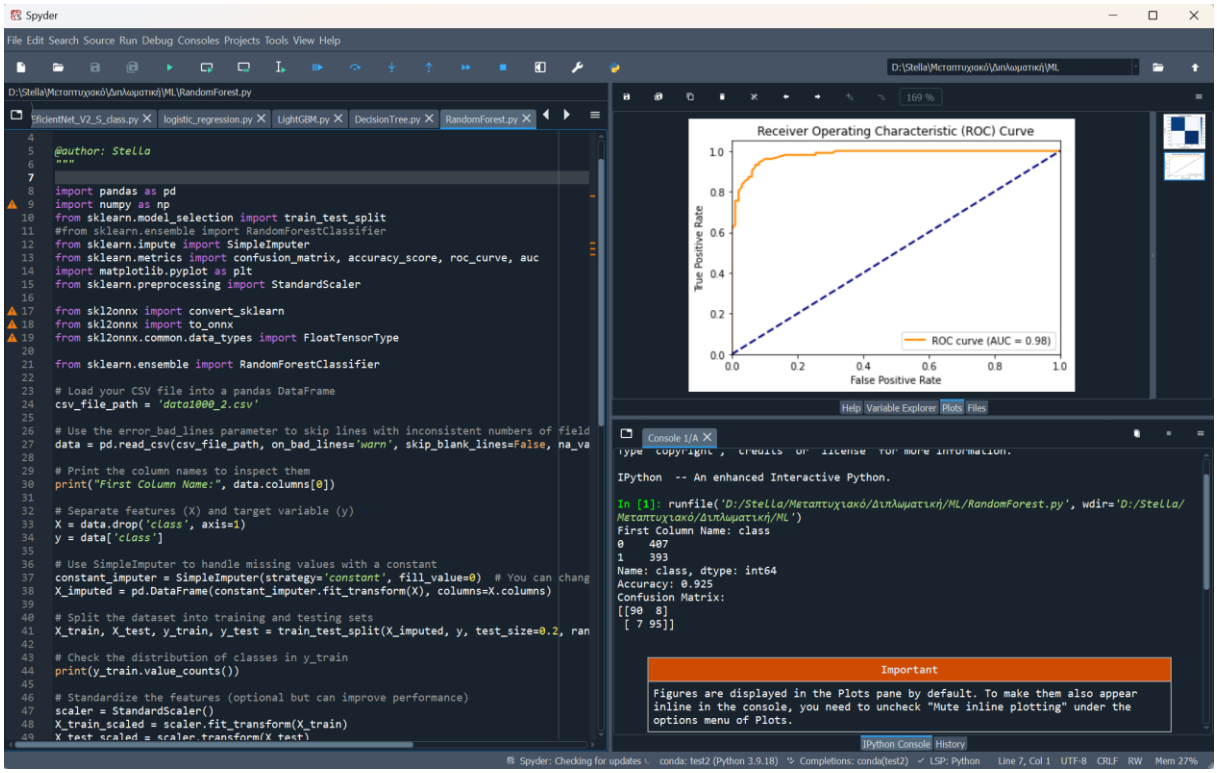
7

Ανάπτυξη Μοντέλου

Σε αυτό το κεφάλαιο παρατίθενται αναλυτικά οι διαδικασίες εκπαίδευσης διαφόρων μοντέλων ΜΜ για την επίτευξη του σκοπού της εργασίας. Πραγματοποιήθηκαν πολλές δοκιμές με διαφορετικούς γνωστούς αλγορίθμους ΜΜ, αλλά και πακέτα με έτοιμες μεθόδους Νευρωνικών δικτύων. Αναλύονται τα εργαλεία ανάπτυξης και παρατίθενται τα αποτελέσματα της εκπαίδευσης πριν ενσωματωθούν στην εφαρμογή παιχνιδιού για την επίδειξη της αξιολόγησής τους. Αυτή η διαδικασία ήταν και η πιο έντονη κατά την ανάπτυξη και συγγραφή, περιείχε πολύ trial-and-error, και απέδωσε τις περισσότερες διαπιστώσεις όσον αφορά τον τομέα αυτόν.

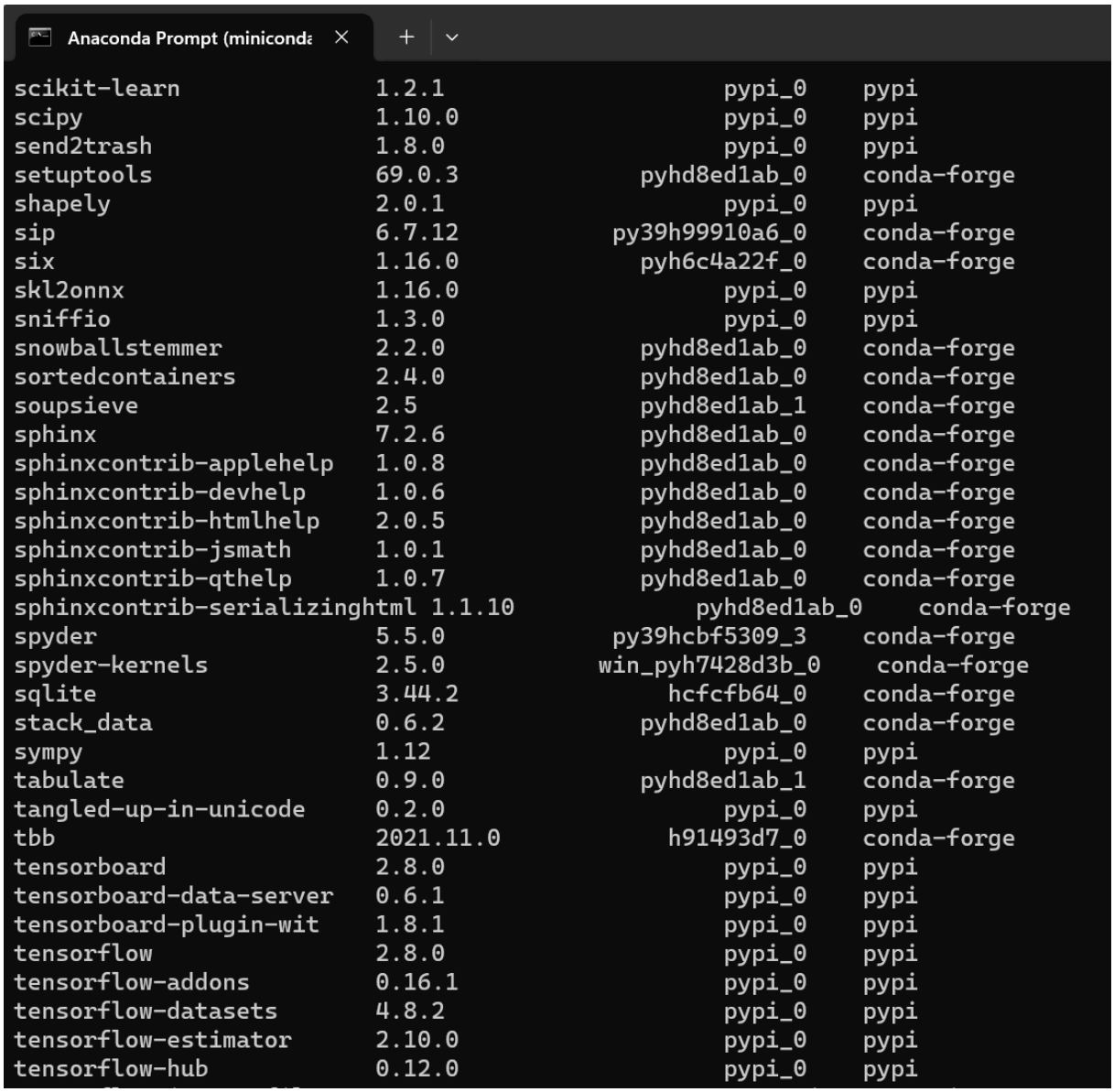
7.1 Εργαλεία εκπαίδευσης

Για τη διαδικασία εκπαίδευσης χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python [20] στο περιβάλλον ανάπτυξης και συγγραφής Spyder IDE [33] (Εικόνα 7.1). Η γλώσσα Python ενδείκνυται για εργασίες ΜΜ, εφόσον ενσωματώνει έναν μεγάλο αριθμό βιβλιοθηκών που αφορούν τέτοιες διεργασίες, όπως τις βιβλιοθήκες tensorflow/keras. Το περιβάλλον ανάπτυξης Spyder υποστηρίζει τη συγγραφή κώδικα με Python, αλλά και χρήση τερματικού που συνδέεται σε περιβάλλον conda το οποίο θα αναλυθεί παρακάτω. Επιπλέον ενσωματώνει λειτουργίες όπως την εμφάνιση διαγραμμάτων, και την εμφάνιση ανάθεσης μεταβλητών στο τρέχον πρόγραμμα.



Εικόνα 7.1: Περιβάλλον Spyder

Το Conda είναι ένας διαχειριστής πακέτων και ένα σύστημα διαχείρισης περιβάλλοντος ανοικτού κώδικα, διαπλατφορμικό, γλωσσικά ανεξάρτητο [34]. Αναπτύχθηκε αρχικά για να λύσει δύσκολες προκλήσεις διαχείρισης πακέτων που αντιμετώπιζαν οι επιστήμονες δεδομένων Python και σήμερα είναι ένας δημοφιλής διαχειριστής πακέτων για Python και R. Στο παρόν εγχείρημα, χρειάστηκε η δημιουργία ενός περιβάλλοντος conda, πιο συγκεκριμένα miniconda, και η εγκατάσταση διαφόρων βιβλιοθηκών μέσα σε αυτό το περιβάλλον, για την υποστήριξη των πειραμάτων. Στην Εικόνα 7.2 φαίνονται κάποια από τα περιεχόμενα του περιβάλλοντος conda, στην τερματική κονσόλα miniconda.



Εικόνα 7.2: Τμήμα περιεχομένων περιβάλλοντος conda, στο τερματικό miniconda

Η εκπαίδευση και η ανάπτυξη πραγματοποιήθηκαν σε ένα αρκετά ισχυρό υπολογιστικό σύστημα, οπότε αξιοποιήθηκε μία ισχυρή κάρτα γραφικών για τους υπολογισμούς. Αυτός ο μετασχηματισμός καθίσταται δυνατός μέσω της εργαλειοθήκης CUDA [35] για κάρτες γραφικών nVidia [36]. Το CUDA είναι μια ιδιόκτητη και κλειστού κώδικα πλατφόρμα παράλληλου υπολογισμού, και διεπαφή προγραμματισμού εφαρμογών, που επιτρέπει στο λογισμικό να χρησιμοποιεί ορισμένους τύπους μονάδων επεξεργασίας γραφικών για επεξεργασία γενικού σκοπού, μια προσέγγιση που ονομάζεται υπολογισμός γενικού σκοπού σε κάρτες γραφικών (general-purpose computing on GPUs). Η έκδοση CUDA πρέπει να συνάδει με την εκάστοτε κάρτα γραφικών, και ρυθμίσεις πρέπει να γίνουν στο περιβάλλον ανάπτυξης (conda) για την υποστήριξη της εργαλειοθήκης.

7.2 Αλγόριθμοι ΜΜ

Εφόσον είχε συλλεχθεί ένα ικανοποιητικό μέγεθος δεδομένων, αξιοποιήθηκαν όλες οι μέθοδοι των κεφαλαίων 6.1 & 6.2 για να εκπαιδευτεί ένα μοντέλο πάνω σε αυτά. Για όλα τα μοντέλα τα δεδομένα υπέστηκαν προ-επεξεργασία. Σε όλα τα μοντέλα επίσης, εφαρμόστηκαν τεχνικές αξιολόγησης, που θα αναλυθούν στο επόμενο κεφάλαιο.

Αρχικά με τους αλγορίθμους ΜΜ, τα scripts ήταν φαινομενικά ίδια, με τη μόνη διαφορά την ενσωμάτωση του εκάστοτε αλγορίθμου στην εκπαίδευση. Στην Εικόνα 7.3 φαίνεται ο κώδικας για την εκπαίδευση του μοντέλου λογιστικής παλινδρόμησης, και στην επισημαινόμενη γραμμή βρίσκεται η κλήση του αλγορίθμου με τις παραμέτρους της.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.impute import SimpleImputer
from sklearn.metrics import confusion_matrix, accuracy_score, roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

# Load your CSV file into a pandas DataFrame
csv_file_path = 'data1000_2.csv'

# Use the error_bad_lines parameter to skip lines with inconsistent number of fields
data = pd.read_csv(csv_file_path, on_bad_lines='warn', skip_blank_lines=False, na_values='', delimiter=';', decimal=',')

# Print the column names to inspect them
print("First Column Name:", data.columns[0])

# Separate features (X) and target variable (y)
X = data.drop('class', axis=1)
y = data['class']

# Use SimpleImputer to handle missing values with a constant
constant_imputer = SimpleImputer(strategy='constant', fill_value=0)  # You can change fill_value to any constant value
X_imputed = pd.DataFrame(constant_imputer.fit_transform(X), columns=X.columns)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_imputed, y, test_size=0.2, random_state=42)

# Check the distribution of classes in y_train
print(y_train.value_counts())

# Standardize the features (optional but can improve performance)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create a logistic regression model
model = LogisticRegression(class_weight='balanced', max_iter=400)

# Train the model
model.fit(X_train_scaled, y_train)

print(model)

# Make predictions on the test set
y_pred = model.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
```

Εικόνα 7.3: Ο κώδικας του μοντέλου λογιστικής παλινδρόμησης σε python.

Ξεκινώντας την προ-επεξεργασία, χρειάστηκε να διαβαστεί από τον editor το αρχείο csv, αναγνωρίζοντας ότι κάθε καταχώρηση χωρίζεται από την επόμενη με τον ειδικό χαρακτήρα ‘;’ , ότι η

υποδιαστολή επισημαίνεται με τον χαρακτήρα ‘,’ και ότι δεν χρειάζεται να παρακάμπτονται οι κενές γραμμές, μιας και δεν υπάρχουν. Όλα τα περιεχόμενα του αρχείου περάστηκαν στη δομή δεδομένων dataframe, για την ευκολότερη χειραγώγησή τους, μέσω της, πολύ εύχρηστης σε αριθμητικά δεδομένα, βιβλιοθήκης panda.

Έπειτα διαχωρίστηκε η πρώτη στήλη από τις υπόλοιπες, αυτή που περιέχει την κλάση της κάθε καταχώρησης, και τα δεδομένα πλέον βρίσκονται σε δύο ξεχωριστά dataframes (X, y). Όπως έχει αναλυθεί, τα δεδομένα του πειράματος περιέχουν καταχωρήσεις μεταβλητού μήκους, και για αυτό το λόγο, χρειάστηκε να υποστούν καταλογισμό (imputing), ένα είδος κανονικοποίησης στην προ-επεξεργασία τους. Έτσι στα δεδομένα του X, όπου βρέθηκε κενό κελί, συμπληρώθηκε με τον αριθμό ‘0’. Μικρής σημασίας ήταν ο καθορισμός του συγκεκριμένου αυτού αριθμού, εφόσον θα ήταν ίδιος για κάθε κενή καταχώρηση.

Στη συνέχεια τα δεδομένα χωρίστηκαν σε έξι dataframes σε αναλογία 80/20 %, το dataframe X με τα πραγματικά αριθμητικά δεδομένα χωρίστηκε σε X_train & X_test, και το dataframe y με τις κλάσεις των δεδομένων (0 , 1) χωρίστηκε σε y_train & y_test. Από το X_train και y_train, το 20% των καταχωρήσεων χωρίστηκε στο X_val & y_val αντίστοιχα. Έτσι θα εκπαιδευτεί το μοντέλο πάνω στα train dataframes, και θα ελεγχθεί η απόδοσή του πάνω στα validation dataframes, και πάνω στα testing dataframes θα προσπαθήσει να προβλέψει τυφλά τις κλάσεις y για κάθε καταχώρηση X.

Επιπρόσθετα, τα δεδομένα υπόκεινται και σε Standard scaling, όπου οι τιμές κεντράρονται γύρω από τη μέση τιμή με ενιαία τυπική απόκλιση, μια ακόμη τεχνική κανονικοποίησης των δεδομένων που θα χρησιμοποιηθεί και στο επόμενο κεφάλαιο. Αυτό γίνεται για να βρίσκονται τα δεδομένα σε μία κοινή κλίμακα, κάνοντας τους υπολογισμούς με αυτά πιο έγκυρους και αναγνώσιμους.

Το πιο σημαντικό κομμάτι της ανάπτυξης είναι η ίδια η εκπαίδευση, όπου σε αυτά τα μοντέλα γίνεται με τη χρήση της εκάστοτε μεθόδου, της εκάστοτε βιβλιοθήκης. Η καθεμία απαιτεί και την κατάλληλη παραμετροποίηση. Τέλος, το μοντέλο γίνεται ‘fitted’, δηλαδή μετριέται το κατά πόσο καλά γενικεύει το μοντέλο σε νέα δεδομένα, παρόμοια με αυτά στα οποία έχει εκπαιδευτεί. Στη συνέχεια κάνει τις προβλέψεις του πάνω στα δεδομένα ελέγχου (testing) και τυπώνεται η ακρίβεια των προβλέψεων, έχοντας συγκριθεί με τις πραγματικές τιμές του y_test. Μπορούν να αποθηκευτούν τα μοντέλα σε ειδικά αρχεία, όπως και να σχηματιστούν διαγράμματα που να εμφανίζουν τις διαδικασίες τους και την απόδοσή τους. Περισσότερες λεπτομέρειες για αυτές τις διαδικασίες θα αναλυθούν στο κεφάλαιο 7.

7.3 Ανάπτυξη Νευρωνικού δικτύου

Όπως συζητήθηκε στο κεφάλαιο 6.2 δεν ήταν δυνατή η εύρεση κάποιου προ-εκπαιδευμένου μοντέλου που να αφορά το παρόν πρόβλημα και τα παρόντα δεδομένα. Ως εκ τούτου αναπτύχθηκε ένα νέο, σχετικά απλό ΝΔ για την δυαδική κατηγοριοποίηση των δεδομένων.

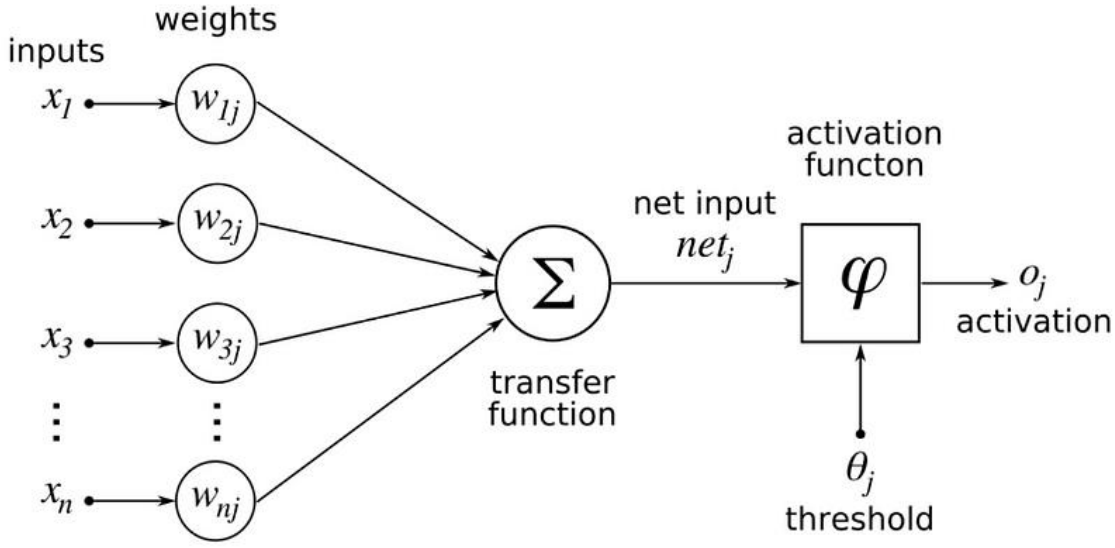
Οι βιβλιοθήκες της python που καθιστούν αυτή τη διαδικασία πραγματοποιήσιμη είναι όπως αναφέρθηκε οι tensorflow/keras. Στην Εικόνα 7.4 φαίνεται το αρχικό script του μοντέλου αυτού.

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Load the dataset
csv_file_path = 'data1000_2.csv'
df = pd.read_csv(csv_file_path, on_bad_lines='warn', skip_blank_lines=False,
                 na_values='', delimiter=';', decimal=',')

# Display the first few rows to understand the data
print(df.head())

# Extract features and labels
X = df.drop('class', axis=1)
y = df['class']

# Use SimpleImputer to handle missing values with a constant
constant_imputer = SimpleImputer(strategy='constant', fill_value=0)
X_imputed = pd.DataFrame(constant_imputer.fit_transform(X), columns=X.columns)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_imputed, y, test_size=0.2, random_state=42)

# Further split the training data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

# Define the model
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_val, y_val))

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Accuracy on the test set: {accuracy}')

# Make predictions
predictions = model.predict(X_test)
```

Εικόνα 7.4: Ο κώδικας ανάπτυξης του απλού Νευρωνικού Δικτύου σε python

Το πρώτο κομμάτι, αυτό της προ-επεξεργασίας αναλύθηκε στο προηγούμενο κεφάλαιο και παραμένει το ίδιο. Ακολουθεί το κομμάτι της σύστασης του ΝΔ, όπου χρησιμοποιούνται έτοιμες μέθοδοι οι οποίες εκτελούνται η μία μετά την άλλη για να δημιουργήσουν τα στρώματα του ΝΔ.

Πρώτα δημιουργείται το ΝΔ με την εντολή model = Sequential(). Δηλαδή δημιουργείται μία γραμμική συστάδα στρωμάτων, πάνω στην οποία εύκολα μπορούν να προστεθούν επιπλέον στρώματα ένα-ένα. Στη συνέχεια, προστίθεται ένα Πυκνό στρώμα με 64 νευρώνες και τη συνάρτηση ενεργοποίησης 'ReLU'. Τα δεδομένα που αναμένονται να εισαχθούν στην είσοδο αυτού του στρώματος, είναι της μορφής X_train.shape[1], δηλαδή τα features των δεδομένων θα είναι όσες είναι οι στήλες του X_train (407 στο πειραμά μας). Αυτό το στρώμα αναπαριστά ένα πλήρως συνδεδεμένο στρώμα, όπου κάθε νευρώνας είναι συνδεδεμένος με κάθε νευρώνα του προηγούμενου στρώματος. Στην Εικόνα 7.5 φαίνεται η δομή ενός μοντέλου με συνάρτηση ενεργοποίησης. Ένα μεγάλο πλήθος νευρώνων αυξάνει την πολυπλοκότητα στη μάθηση του μοντέλου, όμως μπορεί να οδηγήσει σε υπέρ-προσαρμογή (overfitting). Το ReLU (Rectified Linear Unit) είναι μία συνήθης συνάρτηση ενεργοποίηση στα ΝΔ, η οποία εισάγει μη-γραμμικότητα στο δίκτυο επιτρέποντάς του να μαθαίνει σύνθετα μοτίβα (ReLU(x) = max(x, 0)) [37].



Εικόνα 7.5: Δομή μοντέλου με συνάρτηση ενεργοποίησης.

Στη συνέχεια, προστίθεται ακόμα ένα πυκνό στρώμα με 32 νευρώνες και την ίδια συνάρτηση ενεργοποίησης, όπως και το απαραίτητο στρώμα εξόδου. Αυτό αποτελείται από έναν νευρώνα λόγω του ότι το πρόβλημα είναι δυαδικής ταξινόμησης και το αποτέλεσμα πρέπει να είναι ένας αριθμός. Η συνάρτηση ενεργοποίησης εδώ είναι η Σιγμοειδής, η οποία συμπτύσσει το αποτέλεσμα σε έναν πραγματικό αριθμό ανάμεσα στο 0 και στο 1, και μπορεί να ερμηνευθεί και ως ποσοστό/πιθανότητα.

Έπειτα ολόκληρο το μοντέλο συντάσσεται και παραμετροποιείται για την εκπαίδευση. Ορίζεται η συνάρτηση βελτιστοποίησης (adam), η πλέον διαδομένη, που ορίζει το βήμα εκπαίδευσης. Ορίζεται η συνάρτηση απωλειών σε binary crossentropy για προβλήματα δυαδικής ταξινόμησης, που προσμετρά τη διαφορά μεταξύ των προβλέψεων του μοντέλου και των πραγματικών τιμών. Και ορίζεται η μετρική απόδοσης του αποτελέσματος, εδώ η ακρίβεια του ποσοστού σωστών προβλέψεων, κάτι που θα αναλυθεί περεταίρω στο κεφάλαιο 8.

Τέλος, το μοντέλο εκπαιδεύεται στα δεδομένα train για 10 εποχές, μετατρέποντας τα βάρη ώστε να ελαχιστοποιήσει τις απώλειες. Τα βάρη μετατρέπονται κάθε 32 δείγματα στην προκειμένη περίπτωση. Για την επιβεβαίωση των εκμαθημένων πληροφοριών του μοντέλου χρησιμοποιείται το validation set, όπως έχει οριστεί προηγουμένως στο 20% των δεδομένων εκπαίδευσης. Το μοντέλο αξιολογείται πάνω στο test set για να παρουσιαστεί η απόδοσή του στην ακρίβεια και τις απώλειες, κάνοντας τις προβλέψεις του πάνω στο test set.

```
Epoch 1/10
20/20 [==============================] - 0s 11ms/step - loss: 0.5459 - accuracy: 0.7328 - val_loss: 0.4439 - val_accuracy: 0.8062
Epoch 2/10
20/20 [==============================] - 0s 5ms/step - loss: 0.3393 - accuracy: 0.8656 - val_loss: 0.3510 - val_accuracy: 0.8500
Epoch 3/10
20/20 [==============================] - 0s 7ms/step - loss: 0.2830 - accuracy: 0.8938 - val_loss: 0.3243 - val_accuracy: 0.8375
Epoch 4/10
20/20 [==============================] - 0s 11ms/step - loss: 0.2369 - accuracy: 0.9125 - val_loss: 0.3277 - val_accuracy: 0.8188
Epoch 5/10
20/20 [==============================] - 0s 4ms/step - loss: 0.2003 - accuracy: 0.9281 - val_loss: 0.3077 - val_accuracy: 0.8438
Epoch 6/10
20/20 [==============================] - 0s 5ms/step - loss: 0.1828 - accuracy: 0.9359 - val_loss: 0.3485 - val_accuracy: 0.8188
Epoch 7/10
20/20 [==============================] - 0s 7ms/step - loss: 0.1479 - accuracy: 0.9500 - val_loss: 0.3070 - val_accuracy: 0.8438
Epoch 8/10
20/20 [==============================] - 0s 4ms/step - loss: 0.1335 - accuracy: 0.9625 - val_loss: 0.3401 - val_accuracy: 0.8313
Epoch 9/10
20/20 [==============================] - 0s 4ms/step - loss: 0.1120 - accuracy: 0.9656 - val_loss: 0.3007 - val_accuracy: 0.8375
Epoch 10/10
20/20 [==============================] - 0s 4ms/step - loss: 0.0926 - accuracy: 0.9766 - val_loss: 0.3224 - val_accuracy: 0.8625
7/7 [==============================] - 0s 3ms/step - loss: 0.2917 - accuracy: 0.8900
Accuracy on the test set: 0.8899999856948853
```

Εικόνα 7.6: Η επίδοση του μοντέλου ανά εποχή και η ακρίβειά του στο test set.

Στην Εικόνα 7.6 παρουσιάζεται η διαδικασία εκπαίδευσης όπως παράγεται κατά αυτήν και στο τερματικό iPython του Spyder IDE. Εμφανίζονται τα στοιχεία σε κάθε εποχή εκπαίδευσης, και διακρίνεται ότι η ακρίβεια κατά την εκπαίδευση αυξάνεται. Αυτό δείχνει ότι το μοντέλο μαθαίνει με κάθε εποχή και αποδίδει καλύτερα. Το ίδιο δείχνει και η πτώση στην απώλεια, αλλά και η αύξηση στην ακρίβεια του σετ επιβεβαίωσης. Στο τέλος εμφανίζεται και η απώλεια και η ακρίβεια πάνω στο test set. Με τη μέθοδο train/test split, μπορούμε να έχουμε μια καλή εικόνα για την επίδοση του μοντέλου σε νέα δεδομένα, που να μην τα έχει ‘δει’ κατά την εκπαίδευση. Σαφώς και η επίδοσή του είναι υψηλότερη στα γνωστά δεδομένα, όμως όσο πιο κοντά είναι οι δύο ακρίβειες, τόσο πιο δυνατό είναι το μοντέλο.

8

Αξιολόγηση

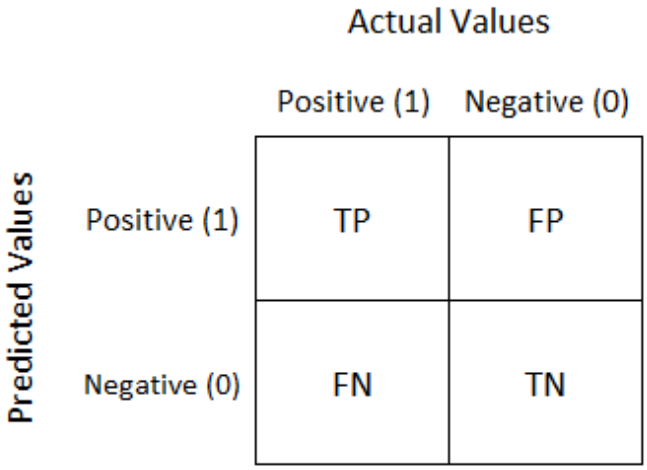
Ένα μεγάλο και σημαντικό κομμάτι της εκπαίδευσης μοντέλων ΜΜ είναι η αξιολόγηση του μοντέλου, και πιο συγκεκριμένα στην παρούσα κατάσταση, του ταξινομητή. Για να επιλεχθεί ο κατάλληλος ταξινομητής για το πρόβλημα, χρειάζεται πρώτα να επιλεχθεί η κατάλληλη μετρική της αξιολόγησής του. Στο παρόν κεφάλαιο αναλύεται η διαδικασία αξιολόγησης των μοντέλων που αναπτύχθηκαν, και οι μετρικές αξιολόγησής τους.

8.1 Παράμετροι αξιολόγησης

Ο Vujovic στην έκθεσή του παρουσιάζει πολλές από τις σημαντικές μετρικές για το πρόβλημα της ταξινόμησης [38]. Εμείς θα επικεντρωθούμε σε συγκεκριμένες μετρικές. Στα αποτελέσματα μιας ταξινόμησης έχουμε τις κατηγορίες:

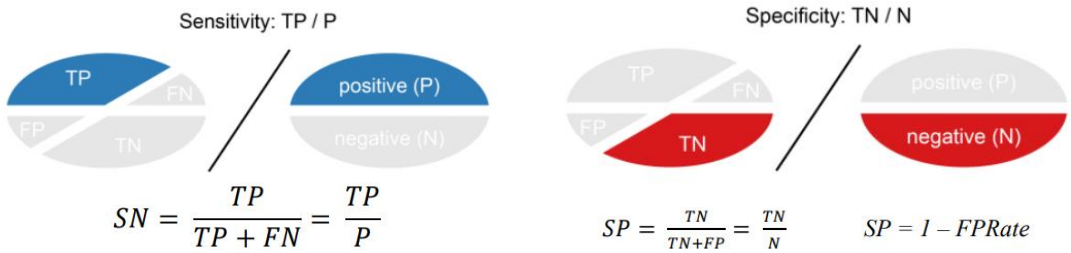
- True Positive (TP): Οι καταχωρήσεις που ταξινομήθηκαν ως θετικές και ήταν θετικές.
- True Negative (TN): Οι καταχωρήσεις που ταξινομήθηκαν ως αρνητικές και ήταν αρνητικές.
- False Positive (FP): Οι καταχωρήσεις που ταξινομήθηκαν ως θετικές και ήταν αρνητικές.
- False Negative (FN): Οι καταχωρήσεις που ταξινομήθηκαν ως αρνητικές και ήταν θετικές.

Στα περισσότερα προβλήματα τέτοιας φύσης, όταν έχουμε δηλαδή δυαδική ταξινόμηση και τα αποτελέσματα μπορεί να είναι 0 ή 1, θεωρείται αυτόματα το 1 ως θετικό αποτέλεσμα, και το 0 ως αρνητικό αποτέλεσμα. Η διαδικασία αυτή εικονοποιείται μέσω του λεγόμενου ‘Πίνακα Σύγχυσης’ (Confusion Matrix), όπου το άθροισμα της κάθε κατηγορίας τοποθετείται σε ένα κελί (Σχήμα 8.1).



Σχήμα 8.1: Πίνακας Σύγχυσης (Confusion Matrix), αξιολόγηση δυαδικής ταξινόμησης.

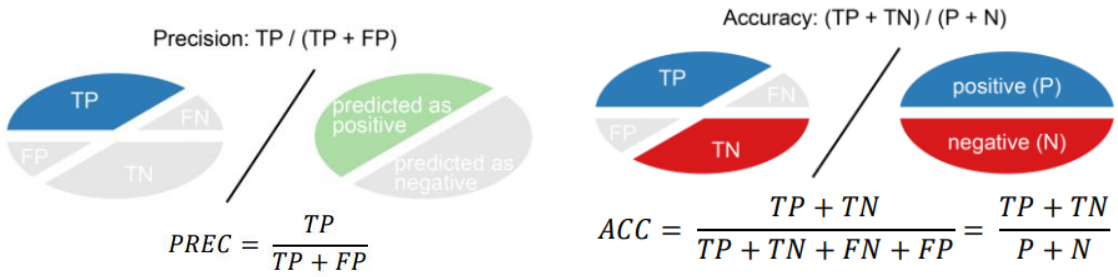
Η Ευαισθησία (Sensitivity) ή Ανάκληση (Recall) είναι το πηλίκο των ορθώς θετικών προβλέψεων με το σύνολο των θετικών πραγματικών δεδομένων. Δείχνει το κατά πόσο το μοντέλο ανιχνεύει τα θετικά παραδείγματα (Βέλτιστο 1.0, Χείριστο 0.0). Η Ειδικότητα (Specificity) είναι το πηλίκο των ορθώς αρνητικών προβλέψεων με το σύνολο των αρνητικών πραγματικών δεδομένων. Δείχνει το κατά πόσο το μοντέλο ανιχνεύει τα αρνητικά παραδείγματα (Βέλτιστο 1.0, Χείριστο 0.0). Αναλόγως τη φύση του προβλήματος, μπορεί να επιθυμείται μεγαλύτερο σκορ στη μία από τις δύο μετρικές. Στο παρόν πρόβλημα δεν υπάρχει κάποιο βάρος στη μία ή την άλλη έκβαση της ταξινόμησης, θέλουμε οι δύο αυτές μετρικές να είναι προσεγγιστικά ίσες. Σε κάποια διάγνωση ασθένειας για παράδειγμα, η μετρική της Ευαισθησίας θα ήταν πιο σημαντική από αυτήν της Ειδικότητας. Και οι δύο μετρικές εμφανίζονται στο Σχήμα 8.2.



Σχήμα 8.2: Ευαισθησία (αριστερά), Ειδικότητα (δεξιά) [38].

Αξίζει να αναφερθεί και η μετρική της Πιστότητας (Precision), όπου είναι το πηλίκο των ορθώς θετικών προβλέψεων με το άθροισμα όλων των θετικών προβλέψεων (Βέλτιστο 1.0, Χείριστο 0.0). Και αυτή μας ενδιαφέρει αν υπάρχει κάποιο βάρος στην έκβαση του αποτελέσματος, συγκεκριμένα των θετικών

αποτελεσμάτων. Η πιο σημαντική μετρική για το παρόν πρόβλημα, είναι αυτή της Ακρίβειας (Accuracy). Η Ακρίβεια ορίζεται ως το άθροισμα των δύο ορθών προβλέψεων, διαιρεμένο με το άθροισμα του συνόλου των πραγματικών αποτελεσμάτων (Βέλτιστο 1.0, Χείριστο 0.0). Και οι δύο αυτές μετρικές εμφανίζονται στο Σχήμα 8.3. Σε προβλήματα με μη ισορροπημένα δεδομένα (όπου οι κλάσεις έχουν διαφορετικές συχνότητες), η χρήση μόνο του Accuracy μπορεί να είναι παραπλανητική. Όμως κατά τη συλλογή αυτών των δεδομένων, αυτό το χαρακτηριστικό λήφθηκε υπόψιν και τα δεδομένα πάρθηκαν όσο το δυνατόν πιο ισορροπημένα.



Σχήμα 8.3: Πιστότητα (αριστερά), Ακρίβεια (δεξιά) [38].

8.2 Αποτίμηση της Ακρίβειας

Για την αποτίμηση της ακρίβειας, σε όλα τα μοντέλα αρχικά χρησιμοποιήθηκε η μέθοδος train/test split με στατικό ποσοστό διαχωρισμού των δεδομένων 80/20. Στη συνέχεια χρησιμοποιήθηκαν και πιο ενδελεχείς έλεγχοι με τις μεθόδους K-fold Cross-validation, Random subsampling, και Leave-one-out (LOO) [39]. Η Leave-one-out χρησιμοποιήθηκε μόνο μία φορά σε κάθε μοντέλο λόγω του τεράστιου χρόνου υπολογισμού της (1000 εκπαιδεύσεις για κάθε μοντέλο, μία για κάθε καταχώρηση του δείγματος).

1. Train/Test Split (Διαίρεση σε σύνολα εκπαίδευσης/ελέγχου 80/20):
   - Διαχωρίζει τα δεδομένα σε δύο σύνολα, ένα για εκπαίδευση (train) και ένα για αξιολόγηση (test), με αναλογία 80/20 στο παράδειγμά μας.
   - Το πλεονέκτημά του είναι η απλότητα και η ταχύτητα υλοποίησης.
   - Η πιθανή αδυναμία είναι η επηρεαστικότητα των αποτελεσμάτων από τον τρόπο διαίρεσης των δεδομένων.
2. k-fold Cross-Validation:
   - Διαχωρίζει τα δεδομένα σε k ομάδες (ή "φωλιές") και εκπαιδεύει το μοντέλο k φορές, χρησιμοποιώντας κάθε φορά μια από τις ομάδες ως σύνολο ελέγχου και τις υπόλοιπες ως σύνολα εκπαίδευσης.

- Πλεονέκτημα του είναι ότι επιτρέπει την αξιολόγηση του μοντέλου σε ολόκληρο το σύνολο δεδομένων και μειώνει την επίδραση της τυχαιότητας στα αποτελέσματα.
- Μειονέκτημα του είναι η αυξημένη υπολογιστική πολυπλοκότητα σε σύγκριση με το train/test split, καθώς εκπαιδεύει το μοντέλο πολλές φορές.

3. Random Subsampling:

- Τυχαία επιλέγει ένα υποσύνολο των δεδομένων και εκπαιδεύει το μοντέλο σε αυτό.
- Πλεονέκτημα είναι η ευελιξία και η δυνατότητα αξιολόγησης της απόδοσης του μοντέλου σε διαφορετικά υποσύνολα δεδομένων.
- Μειονέκτημα είναι ότι η αξιολόγηση μπορεί να είναι λιγότερο σταθερή λόγω της τυχαιότητας της επιλογής του υποσυνόλου.

4. Leave-one-out:

- Για ένα dataset με n παρατηρήσεις εκπαιδεύει και αξιολογεί το μοντέλο n φορές, κάθε φορά αφήνοντας ένα δείγμα έξω από το σύνολο εκπαίδευσης.
- Πλεονέκτημα είναι το ότι κάθε παρατήρηση χρησιμοποιείται ως σύνολο ελέγχου, εξαλείφοντας την επίδραση της τυχαιότητας στην αξιολόγηση, και χρησιμοποιεί όλα τα δεδομένα διαθέσιμα για εκπαίδευση και αξιολόγηση, μειώνοντας τη δυνατότητα υπερ-εκπαίδευσης.
- Μειονέκτημα είναι η αυξημένη υπολογιστική πολυπλοκότητα, ιδιαίτερα για μεγάλα datasets, καθώς πρέπει να εκπαιδευτεί και να αξιολογηθεί το μοντέλο για κάθε παρατήρηση ξεχωριστά.

Στην Εικόνα 8.4 φαίνεται ο κώδικας που χρησιμοποιήθηκε σε όλα τα μοντέλα για την αποτίμηση της ακρίβειας με όλες τις παραπάνω μεθόδους. Στο επόμενο υπο-κεφάλαιο παρατίθενται τα αποτελέσματα όλων των μοντέλων με όλες τις τεχνικές αξιολόγησης.

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f'Accuracy: {accuracy}')

# k-fold cross-validation
k_fold = KFold(n_splits=5, shuffle=True, random_state=42)
accuracy_k_fold = np.mean(cross_val_score(model, X_imputed, y, cv=k_fold))
print("Accuracy με k-fold cross-validation:", accuracy_k_fold)

# Random subsampling
num_iterations = 10
subsample_size = 0.8
accuracies = []

for _ in range(num_iterations):
    indices = np.random.choice(len(X), int(len(X) * subsample_size), replace=False)
    X_subsample = X_imputed.iloc[indices]
    y_subsample = y.iloc[indices]
    model.fit(X_subsample, y_subsample)
    accuracy = model.score(X_test_scaled, y_test)
    accuracies.append(accuracy)

average_accuracy_subsample = np.mean(accuracies)
print("Average accuracy με random subsampling:", average_accuracy_subsample)

# Leave-one-out
#from sklearn.model_selection import LeaveOneOut

#loo = LeaveOneOut()
#scores = cross_val_score(model, X_imputed, y, cv=loo)
#("Accuracy με leave-one-out:", scores.mean())
```

Εικόνα 8.4: Κώδικας για την αποτίμηση ακρίβειας με τις 4 μεθόδους.

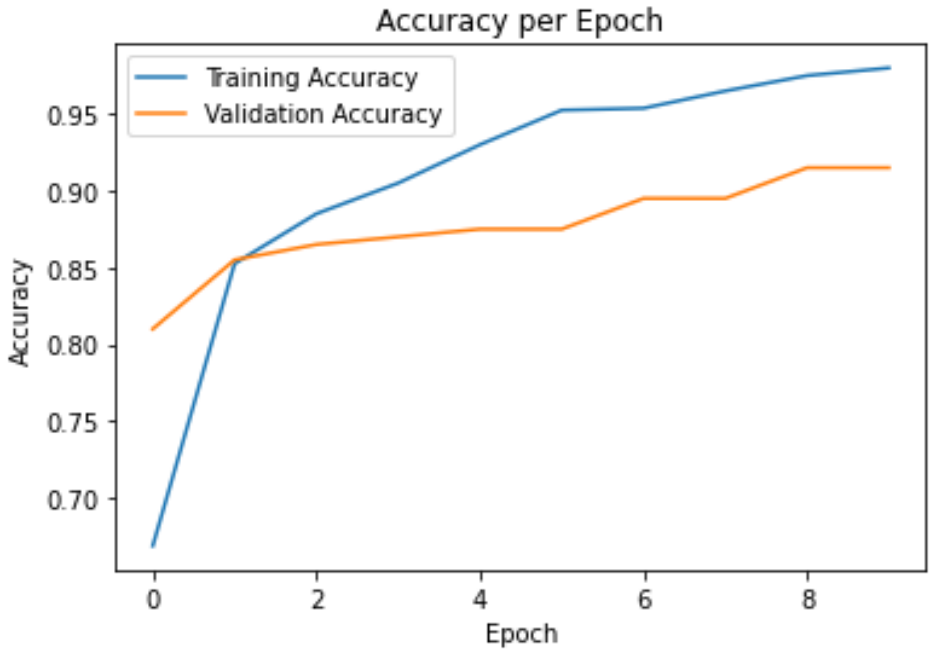
8.3 Αποτελέσματα

Κάποιοι αλγόριθμοι ΜΜ απέδωσαν καλύτερα από το απλό ΝΔ στην εκπαίδευσή τους με βάση το κριτήριο της ακρίβειας (accuracy). Στον Πίνακα 8.5 φαίνονται τα αποτελέσματα του καθενός σε φθίνουσα σειρά, όπως εκπαιδεύτηκαν στο τελικό δείγμα με τις 1000 καταχωρήσεις, και με κάθε τεχνική αξιολόγησης. Φυσικά όπως αναλύθηκε στο προηγούμενο κεφάλαιο, αυτά τα αποτελέσματα αφορούν την απόδοση του κάθε αλγορίθμου πάνω στο test set των δεδομένων, δηλαδή στην εκάστοτε τεχνική, τα δεδομένα για τα οποία δεν έχει δει ο αλγόριθμος τη σωστή τους κλάση.

Αλγόριθμος	80/20	5-fold cv	Random subsample	LOO
Random Forest	**92.5**	90.5	63.55	90.8
LightGBM	**91.5**	89.6	61.55	**92.2**
SVM	87.5	82.3	49	82
Logistic Regression	85.5	76.3	72.75	78.7
Decision Tree	84	85.7	56.9	85
Naïve Bayes	83.5	80.6	53.05	81.3
K-Nearest Neighbors	83	80	49.75	82.1
Keras model	**91**	66.6	84.79	74.5

Πίνακας 8.5: Ακρίβεια αλγορίθμων ΜΜ κατά την εκπαίδευση.

Το Νευρωνικό δίκτυο απέδωσε μέχρι και **91%** ακρίβεια σε νέα δεδομένα με απλό train/test split. Λόγω της καθορισμένης τυχαιότητας στον διαχωρισμό των δεδομένων, αλλά και στον διαχωρισμό των batches κατά την εκπαίδευση, το μοντέλο θα παράγει ελαφρώς διαφορετικά αποτελέσματα με κάθε δοκιμή. Τα αποτελέσματα μίας δοκιμής από αυτές εμφανίζεται στο Σχήμα 8.6. Κατά την εκπαίδευση το μοντέλο απέδωσε μέχρι και **98.12%** ακρίβεια στα ήδη γνωστά δεδομένα. Αυτό είναι και το μοντέλο που θα χρησιμοποιηθεί στην ενσωμάτωση στο Unity, που θα εξηγηθεί στο επόμενο κεφάλαιο.



Σχήμα 8.6: Ένα διάγραμμα που δείχνει την ακρίβεια του μοντέλου κατά την εκπαίδευση, ανά εποχή.

8.4 Συμπεράσματα αξιολόγησης

Κατά την αξιολόγηση των τελικών μοντέλων, επιβεβαιώθηκε η απλότητα του προβλήματος όσον αφορά τη φύση των δεδομένων. Τα αριθμητικά δεδομένα είναι από τις απλούστερες μορφές δεδομένων για την πρόκληση της ΜΜ, τουλάχιστον σε σχέση με αυτήν της κατηγοριοποίησης εικόνων ή κειμένου. Ωστόσο για τον συγκεκριμένο λόγο, αποδείχθηκε πιο δύσκολο να βρεθούν έτοιμα datasets και προ-εκπαιδευμένα μοντέλα για να αξιοποιηθούν στο παρόν εγχείρημα. Η απλότητα του προβλήματος φάνηκε και στην απόδοση της ακρίβειας των μοντέλων, η οποία μετά από ελάχιστο fine-tuning στα μοντέλα με τους αλγορίθμους ΜΜ ήταν κατευθείαν αποδεκτή. Τα μοντέλα μάθαιναν με κάθε εποχή και με κάθε αλγόριθμο ΜΜ, και δεν παρατηρήθηκε υπερ-προσαρμογή (overfitting) αλλά ούτε υπο-προσαρμογή (underfitting) των μοντέλων.

Όσον αφορά το ΝΔ, το μοντέλο που κρατήθηκε και συμπεριλήφθηκε στην εργασία δεν παρουσίασε overfitting & underfitting. Άλλα μοντέλα που δοκιμάστηκαν δεν κατάφεραν να μάθουν, διότι δεν ήταν κατάλληλα για το πρόβλημα της ταξινόμησης αριθμητικών δεδομένων όπως συζητήθηκε σε προηγούμενο κεφάλαιο. Όμως το τελικό Ρηχό ΝΔ απέδωσε πολύ καλά με τα δεδομένα σε κάθε ‘τρέξιμό’ του, και όπως θα αναλυθεί στο επόμενο κεφάλαιο, η δομή της σύστασής του απεδείχθη η πλέον κατάλληλη για την ενσωμάτωση κάποιου από τα μοντέλα στο Unity.

Το σετ δεδομένων δεν ήταν αρκετά μεγάλο, ούτε οι αλγόριθμοι αρκετά περίπλοκοι, για να υπάρξει πρόβλημα με την απαιτούμενη υπολογιστική ισχύ που χρειάστηκε για να ‘τρέξουν’ τα μοντέλα. Εκτός από την αξιολόγηση με LOO, τα μοντέλα παρήγαγαν αποτελέσματα μέσα σε milliseconds για κάθε εποχή.

Συμπερασματικά, για την δυαδική ταξινόμηση των αριθμητικών δεδομένων κίνησης με χειριστήρια VR, η εργασία προτείνει με σειρά απόδοσης τη χρήση του αλγορίθμου Random Forest, του LightGBM, ή ένα ρηχό ΝΔ με χρήση του keras.

9

Ενσωμάτωση στο Unity

Για την καλύτερη επίδειξη της επίδοσης των μοντέλων, χρησιμοποιήθηκε η σκηνή του παιχνιδιού στο Unity. Το βέλτιστο σενάριο αυτής της επίδειξης θα ήταν να μπορεί να εκπαιδεύεται το μοντέλο σε πραγματικό χρόνο κατά την διαδικασία συλλογής των δεδομένων, ενώ ο παίκτης παίζει το παιχνίδι, και να παρουσιάζονται οι προβλέψεις του μοντέλου σε σύγκριση με τα πραγματικά αποτελέσματα. Αυτή η διαδικασία είναι ακριβή σε πόρους και χρονοβόρα, οπότε η εμπειρία δεν θα μπορούσε να είναι ομαλή και χωρίς καθυστερήσεις μέσα στο εικονικό περιβάλλον. Αυτό που αποπειράθηκε να συμβεί, ήταν η δοκιμή του μοντέλου σε πραγματικό χρόνο όσον αφορά τις προβλέψεις του, χρησιμοποιώντας νέα δεδομένα με τη συμμετοχή του χρήστη. Όπως φαινόταν η επιτυχία ή αποτυχία του χρήστη στο να πετύχει την μπάλα στο εικονικό περιβάλλον, έτσι θα φαίνεται σε σύγκριση και η πρόβλεψη του μοντέλου σε κάθε βολή, με απώτερο σκοπό την εικονοποίηση της επίδοσης του μοντέλου με νέα δεδομένα.

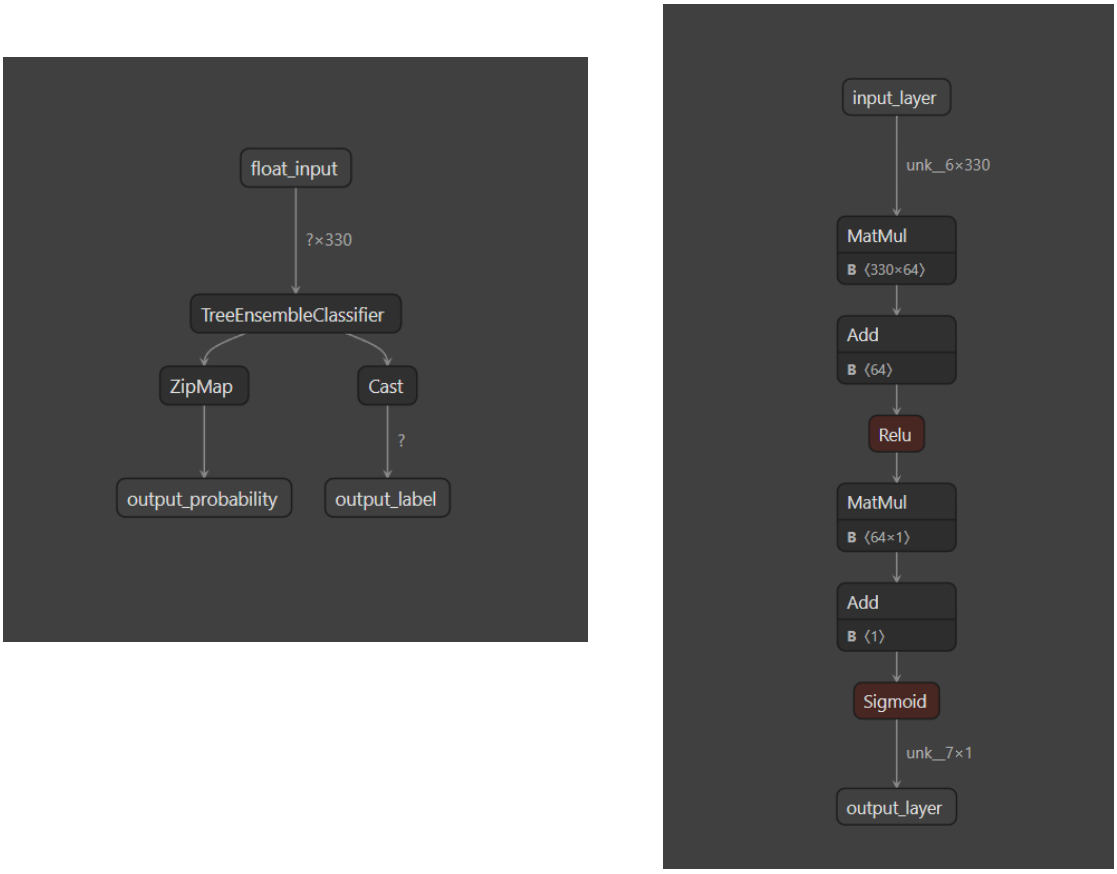
9.1 Συμβατότητα βιβλιοθηκών

Χρειάστηκε να ενσωματωθεί το εκάστοτε μοντέλο στην πλατφόρμα με την προσθήκη πακέτων και βιβλιοθηκών που εξυπηρετούν στην αξιοποίηση ή και στην εκπαίδευση μοντέλων ΜΜ (MLagents [40], Sentis [41]). Αυτή η διαδικασία δεν ήταν άμεση, και αποδείχθηκε περίπλοκη, μιας και χρειάστηκαν αρκετοί πειραματισμοί και δοκιμές για τη συμβατότητα των αρχείων των μοντέλων αλλά και των μεθόδων που υποστηρίζει η κάθε βιβλιοθήκη. Η μηχανή Unity δεν αναγνωρίζει αρχεία μοντέλων της βιβλιοθήκης scikitlearn ούτε της tensorflow. Αντ’ αυτού αναγνωρίζει αρχεία .onnx [42], μιας συλλογής εργαλείων και πρωτοκόλλων ανοιχτού κώδικα, μέσω της βιβλιοθήκης Sentis. Υπάρχει τρόπος να δημιουργήσει και να εκπαιδεύσει κάποιος ένα μοντέλο ΜΜ από την αρχή μέσα στο Unity με το πακέτο MLagents, όμως αυτή η διαδικασία δεν θα συνδύαζε τις μεθόδους που ήταν απαραίτητες για αυτό το πείραμα μέσω python και Conda environments. Η βιβλιοθήκη Sentis του Unity δεν υποστηρίζει τους classifiers που χρησιμοποιούνται από το scikit-learn για την εκπαίδευση μοντέλων, όπως οι Linear

Classifier, Logistic Regression Classifier, Random Forest Classifier, Tree Ensemble Classifier, κλπ [43]. Υποστηρίζει πιο απλοϊκούς ταξινομητές, που χρησιμοποιούνται σε μοντέλα βαθιάς μάθησης, όπως Max Pool, Bernoulli, Sigmoid, Relu, κλπ [44].

Δοκιμάζοντας μετατροπές των αρχείων των μοντέλων σε άλλη μορφή αρχείων, και εισάγοντάς τα στο unity, ήταν φανερό το ποια μοντέλα και ποιοι ταξινομητές αναγνωρίζονταν από το σύστημα. Οι μετατροπές έγιναν με τη βιβλιοθήκη skl2onnx [45] και τις μεθόδους convert_sklearn & to_onnx μέσα στο εκάστοτε script. Οι μετατροπές των αρχείων που δημιουργήθηκαν με το tensorflow/keras έγιναν με τη βιβλιοθήκη tf2onnx [46] και την εντολή tf2onnx.convert στο περιβάλλον conda. Για την ευκολότερη μελέτη των παραγόμενων αρχείων των μοντέλων ώστε να διερευνηθεί η ασυμβατότητά τους, χρησιμοποιήθηκε το πρόγραμμα Netron [47], Εικόνα 9.1.

Αυτό σήμαινε ότι δεν θα μπορούσαν να χρησιμοποιηθούν τα μοντέλα MM μέσα στο Unity για την επίδειξη της απόδοσής τους. Μόνο τα μοντέλα που χρησιμοποιούν τους ταξινομητές που αναγνωρίζονται από το Sentis μπόρεσαν να αναγνωριστούν από το Unity, δηλαδή το ΝΔ που δημιουργήθηκε με το keras.



Εικόνα 9.1: Το μοντέλο Decision Tree με sklearn (αριστερά) και keras (δεξιά) ενσωμάτωση, στο Netron.

9.2 Ενσωμάτωση μοντέλου στο Unity

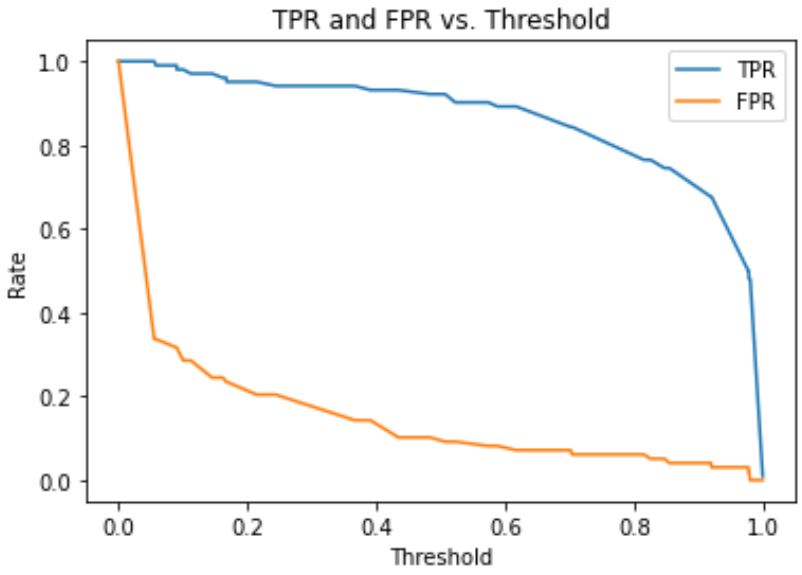
Όπως αναφέρθηκε παραπάνω, χρειάστηκε να εισαχθεί στο project το πακέτο MLagents που περιέχει και τη βιβλιοθήκη Sentis, μέσω του package manager του Unity. Συντάχθηκε ένα νέο script σε C# το οποίο ενσωματώθηκε στο αντικείμενο Game Manager της σκηνής (Εικόνα 9.2). Περιέχει μια Start μέθοδο για την αρχικοποίηση κάποιων μεταβλητών στην εκκίνηση του παιχνιδιού, και μία μέθοδο για την πρόβλεψη του αποτελέσματος, η οποία καλείται από το script Ball.cs μετά το τέλος κάθε ρίψης.

```
void Start()
{
    runtimeModel = ModelLoader.Load(modelAsset);
    worker = WorkerFactory.CreateWorker(BackendType.GPUCompute, runtimeModel);
}

1 reference
public void PredictShotOutcome(float[] inputData)
{
    // Convert the float array to a MathNet.Numerics vector of doubles
    var vector = DenseVector.OfArray(inputData.Select(x => (double)x).ToArray());

    // Perform standard scaling using MathNet.Numerics
    var mean = Statistics.Mean(vector);
    var stdDev = Statistics.StandardDeviation(vector);

    var standardizedVector = (vector - mean) / stdDev;

    // Convert the standardized vector back to float[]
    float[] standardizedArray = standardizedVector.Select(x => (float)x).ToArray();

    // Create a 3D tensor shape with size of inputData
    TensorShape shape = new TensorShape(1, standardizedArray.Length);

    // Create a new tensor from the array
    TensorFloat tensor = new TensorFloat(shape, standardizedArray);

    worker.Execute(tensor);

    foreach (var outputName in runtimeModel.outputs)
    {
        TensorFloat outputTensor = worker.PeekOutput(outputName) as TensorFloat;
        outputTensor.MakeReadable();
        float[] outputData = outputTensor.ToReadOnlyArray();

        float predf = outputData[0];
        pred.text = predf.ToString();
        Debug.Log(predf);
        int prediction = (predf >= 0.01f) ? 1 : 0;
        if (prediction == 1)
            mlResult.GetComponent<Renderer>().material = green;
        else if (prediction == 0)
            mlResult.GetComponent<Renderer>().material = red;
    }
}
```

Εικόνα 9.2: Το script MLprediction.cs

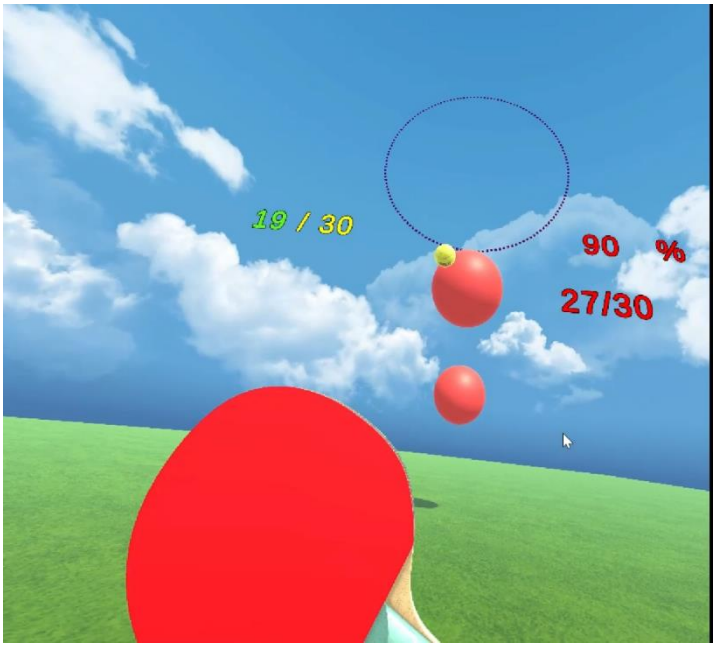
Στην εκκίνηση φορτώνεται το μοντέλο από τα αρχεία του project το οποίο έχει τεθεί στο περιβάλλον Unity Inspector (drag & drop), και δημιουργείται ένας «εργάτης»/πράκτορας που θα εκτελεί τους υπολογισμούς χρησιμοποιώντας την κάρτα γραφικών του υπολογιστικού συστήματος.

Η μέθοδος υπολογισμού (Predict Shot Outcome ()) δημιουργεί πρώτα ένα σχήμα 1 x 407, κάτι το οποίο εξήχθη σαν πληροφορία από την χειρωνακτική επίβλεψη του μοντέλου με το πρόγραμμα Netron. Αυτό περνιέται ως μεταβλητή από το περιβάλλον Unity. Στη συνέχεια δημιουργεί έναν τένσορα με χαρακτηριστικά το σχήμα και τα δεδομένα εισόδου της μεθόδου. Τα δεδομένα εισόδου περνιόνται από το script που καλεί τη μέθοδο, και είναι ένας πίνακας float που περιέχει τα δεδομένα της ρίψης (cameraX, τοποθεσία, περιστροφή, ταχύτητα) για κάθε frame, αφού πρώτα κανονικοποιηθούν για να συμβαδίζουν με το σχήμα του τένσορα. Για αυτή τη διαδικασία, χρειάζεται να επαναληφθεί η κανονικοποίηση που δέχονται τα δεδομένα κατά την εκπαίδευση, δηλαδή να περάσουν από έναν Standard Scaler. Σε αυτό το πλαίσιο, αυτή η διεργασία μπορεί να επιτευχθεί μέσω του πακέτου MathNet.Numerics [48] και των βιβλιοθηκών Statistics & Linear Algebra. Τα δεδομένα μετατρέπονται σε double [] για το scaling, και ξανά πίσω σε float [] για την εκτέλεση του μοντέλου. Στη συνέχεια εκτελείται το μοντέλο από τον πράκτορα με εισόδους τα δεδομένα. Τέλος, γίνεται αναγνώσιμο το αποτέλεσμα εξόδου του μοντέλου, το οποία αποτελείται από έναν πραγματικό αριθμό, και αν είναι κάτω από 0.09 το σχήμα προβολής αποτελέσματος στη σκηνή γίνεται κόκκινο, ενώ αν είναι πάνω από 0.09 γίνεται πράσινο. Η τιμή αυτή υπολογίστηκε σχηματίζοντας το παρακάτω διάγραμμα (Σχήμα 9.3), βλέποντας τις τιμές των True Positive και False Positive rate. Επιλέγεται ποια μετρική θα ‘θυσιαστεί’ στη θέση της άλλης, βρίσκοντας μια ισορροπία ώστε να εξάγεται η μέγιστη δυνατή ακρίβεια.



Σχήμα 9.3: TP rate & FP rate στο threshold 0-1 της εξόδου του ΝΔ.

Όπως και με την εμφάνιση του αποτελέσματος της πραγματικής ρίψης, ο χρήστης μπορεί να δει σε πραγματικό χρόνο αν πέτυχε η βολή του, και αν το μοντέλο προέβλεψε σωστά το αποτέλεσμα αυτής. Ταυτόχρονα βλέπει τη μετρική της Ακρίβειας, πόσες σωστές προβλέψεις έχει κάνει το μοντέλο, αλλά και πόσες βολές έχει πετύχει ο ίδιος (Εικόνα 9.4). Για λόγους διαύγειας αλλά και υπολογιστικής απόδοσης, κατά τη διάρκεια του testing στο περιβάλλον Unity, απενεργοποιείται η διαδικασία καταγραφής νέων δεδομένων, και αντίστροφα. Αυτό γίνεται απλώς με μία Boolean μεταβλητή ελέγχου στο script Shooter.cs, η οποία ενεργοποιείται/απενεργοποιείται μέσα από τον Inspector του Unity πριν 'τρέξει' το παιχνίδι.



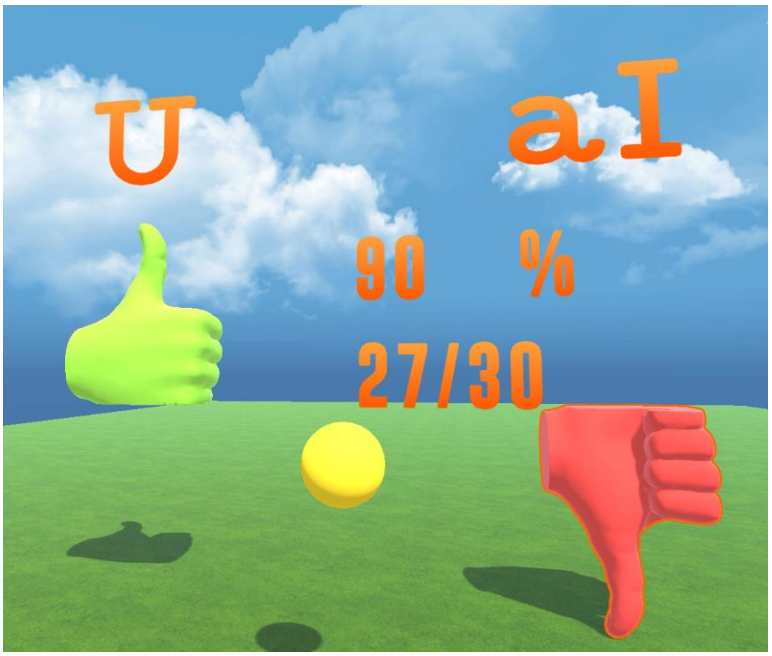
Εικόνα 9.4: Στιγμιότυπο ρίψης κατά το testing στο Unity.

Στο script Ball.cs έγιναν κάποιες προσθήκες για την ενσωμάτωση αυτού του μηχανισμού. Ο κώδικας της κάθε μπάλας που εκτοξεύεται, κρατάει μια μεταβλητή custom κλάσης Float Stack, στην οποία προσθέτει τα δεδομένα της ρίψης αμέσως μετά την καταγραφή τους στο csv αρχείο. Όταν τελειώσει η ρίψη, αφού κλείσει το csv αρχείο, κανονικοποιεί το stack καλώντας την μέθοδο από την custom κλάση Get stack as array, όπου τα δεδομένα από μία λίστα float μετατρέπονται σε πίνακα float, αφού ελεγχθεί το μέγεθος της λίστας. Αν είναι μεγαλύτερο των 407 καταχωρήσεων, κρατώνται μόνο οι πρώτες 407 καταχωρήσεις. Αν είναι μικρότερο, οι θέσεις γεμίζονται με μηδενικά στοιχεία. Αν είναι ακριβώς 407, φυσικά μετατρέπεται αυτόματα η λίστα σε πίνακα, με μία γραμμή και 407 στήλες. Ο κώδικας αυτός βρίσκεται στο Παράρτημα Α.

9.3 Εμπλουτισμός εμπειρίας

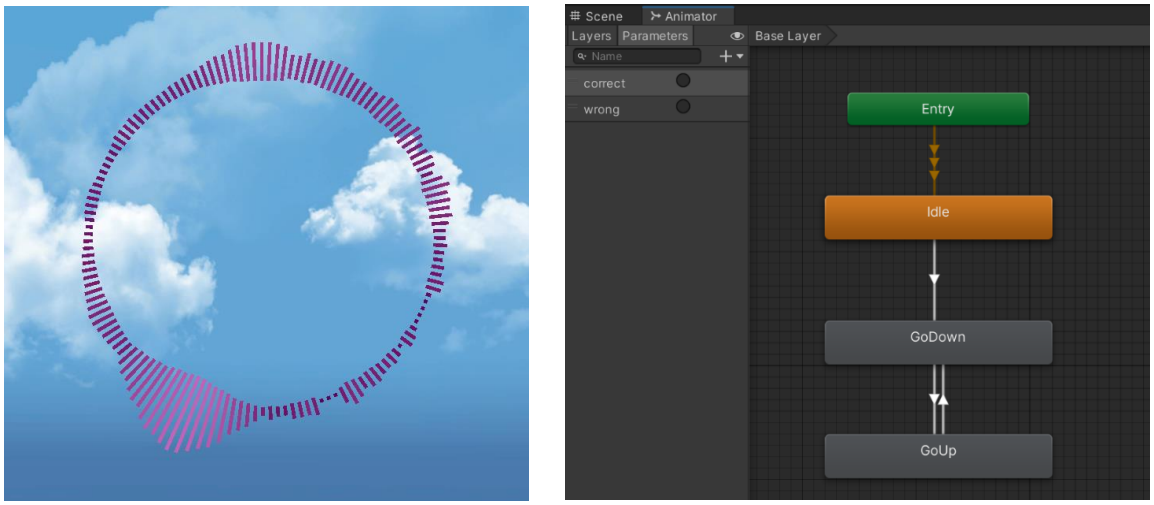
Εφόσον η λειτουργικότητα του παιχνιδιού έχει επιβεβαιωθεί, και το μοντέλο προβλέπει σε πραγματικό χρόνο μέσα στη σκηνή του Unity, μπορούσε το παιχνίδι να εμπλουτιστεί με κάποια στοιχεία ως προς την χρηστικότητα και το παρουσιαστικό του. Μέχρι στιγμής για λόγους αποσφαλμάτωσης, κατά τη διάρκεια του testing, μία σφαίρα εμφανίζεται πράσινη/κόκκινη για το αν πέτυχε ο χρήστης τη μπάλα, και μία σφαίρα κάνει το ίδιο αν το μοντέλο προέβλεψε επιτυχία ή αποτυχία του χρήστη. Εγκαταστάθηκε ένα νέο 3D μοντέλο από μία ανοιχτή και δωρεάν πηγή [49], που να δείχνει το σήμα της επιτυχίας από ένα χέρι. Του προστέθηκαν νέα materials για το χρώμα, αυτά που είχαν οι σφαίρες (πράσινο/κόκκινο). Τοποθετήθηκε ένα αριστερά του Shooter, αυτό για την επιτυχία του χρήστη, και ένα δεξιά, για την πρόβλεψη. Από πάνω τους προστέθηκαν ετικέτες κειμένου σε ένα World Space Canvas, δηλαδή μία επιφάνεια που φιλοξενεί στοιχεία UI αλλά μέσα στις διαστάσεις της σκηνής και όχι της οθόνης. Οι ετικέτες υποδεικνύουν το ποιο χέρι αναπαριστά ποιο αποτέλεσμα. Πιο μακριά στο βάθος της σκηνής προστέθηκαν κι άλλες ετικέτες που δείχνουν το ποσοστό ακρίβειας του μοντέλου και το πόσες ρίψεις έχει προβλέψει σωστά / σύνολο ρίψεων.

Οι ετικέτες υπέστησαν επεξεργασία ως προς την εμφάνιση, το font τους, και το gradient χρώμα τους. Ο κώδικας που υπολογίζει την ακρίβεια προστέθηκε σε διάφορα σημεία του script MLprediction.cs. Έπειτα προγραμματίστηκαν animations που περιστρέφουν τα χέρια προς τα πάνω ή προς τα κάτω αν το αποτέλεσμα ήταν διαφορετικό από το προηγούμενο της κάθε απόφασης (Εικόνα 9.6 δεξιά). Παράλληλα μεταβάλλεται και το χρώμα των χεριών από πράσινο σε κόκκινο και αντίστροφα για ένα δευτερόλεπτο, με κάθε animation, και έπειτα επιστρέφουν στην αρχική τους θέση (οριζόντια) και με κίτρινο χρώμα (Εικόνα 9.5).



Εικόνα 9.5: Τελική εμφάνιση σκηνής Unity.

Στον World Space Canvas προστέθηκε επίσης και ένα άλλο asset ξένης ελεύθερης πηγής [50]. Το συγκεκριμένο asset εικονοποιεί κύματα ήχου από αρχεία ήχου μέσα στη σκηνή. Επιλέχθηκε ένα σχήμα κύκλου με κάθετες μπάρες που να δείχνουν την ένταση μίας φωνής για να προστεθούν αρχεία ήχου στο περιβάλλον (Εικόνα 9.6, αριστερά). Τροποποιήθηκε ως προς το χρώμα του και το ύψος και την απόσταση των ράβδων μεταξύ τους, μέσα από τον Inspector του Unity.



Εικόνα 9.6: Κύκλος με κύματα-μπάρες ήχου (αριστερά), Animator (δεξιά).

Για τη συνάφεια με το θέμα της εργασίας, παράχθηκαν διάφορα ηχητικά clips από μία ιστοσελίδα που μετατρέπει κείμενο σε ήχο από διάφορες αληθινές φωνές, με τεχνικές Επεξεργασίας Φυσικής Γλώσσας (NLP), οι οποίες υλοποιούνται με τεχνικές ΜΜ [51]. Για κάθε ηχητικό κλιπ συντάχθηκε ένα κείμενο με σημεία στίξης και παύσεις, κατεβάστηκαν, και εισήχθησαν στο project. Το script που τα ελέγχει και τα αναπαράγει παρατίθεται στο Παράρτημα Α. Το ύφος τους και ο λογισμός πίσω από αυτά είναι ότι ο αλγόριθμος μιλάει στον παίκτη, τον ενθαρρύνει και τον αποθαρρύνει, τον ενημερώνει για το πλαίσιο της εργασίας και τη διαδικασία της εκπαίδευσης. Καυχιέται αν έχει προβλέψει με μεγάλη ακρίβεια τα αποτελέσματα των ρίψεων, και προκαλεί τον παίκτη να συνεχίσει να προσπαθεί να πετύχει την μπάλα.

10

Επίλογος

10.1 Σύνοψη και συμπεράσματα

Κατά την εκπόνηση αυτής της εργασίας δοκιμάστηκαν πολλές τεχνικές ΜΜ από την τεχνική τους σκοπιά. Όμως μελετήθηκαν και θεωρητικά, και πραγματοποιήθηκε ενδελεχής και βαθιά έρευνα πάνω στην δυαδική κατηγοριοποίηση αριθμητικών δεδομένων, με τεχνικές ΜΜ. Πραγματοποιήθηκε ανάπτυξη ενός εικονικού περιβάλλοντος για τη συλλογή δεδομένων από συσκευές VR. Δοκιμάστηκαν και συγκρίθηκαν αλγόριθμοι ΜΜ όπως και Νευρωνικά δίκτυα, στην απόδοσή τους πάνω στην δυαδική ταξινόμηση, και τα τελευταία ενσωματώθηκαν με επιτυχία στο εικονικό περιβάλλον, για την επίδειξη της απόδοσής τους. Η διαδικασία της αξιολόγησής τους έγινε με επιστημονικές μεθόδους, και καταγράφηκε αποτελεσματικά και διεξοδικά. Η τάση της εργασίας ήταν για τα περισσότερα αν όχι όλα τα εργαλεία και λογισμικά που χρησιμοποιήθηκαν, να ήταν ανοιχτού κώδικα και ελεύθερης χρήσης.

Η κατανόηση της φύσης του προβλήματος οδήγησε στην έγερση κινήτρου για την αποτελεσματικότερη επίλυση του προβλήματος, και ενέτεινε το ενδιαφέρον της ερευνήτριας για τις τεχνολογίες αυτές. Επιπλέον δόθηκε η ευκαιρία να αναπτυχθεί περαιτέρω τριβή με το σχετικό αντικείμενό της, την ανάπτυξη εμπειριών σε περιβάλλοντα ανάπτυξης παιχνιδιών και εικονικών εμπειριών. Αυτός ο συνδυασμός των δύο πολύ επίκαιρων και καινοτόμων τεχνολογιών, αφενός επιβεβαιώθηκε ως πολύ ενδιαφέρων μέσω της εκπόνησης της εργασίας, και αφετέρου προσδοκάται να εμπλουτίστηκε με την ίδια την εκπόνηση της εργασίας. Η ίδια η συγγραφή του κειμένου, έφερε την ερευνήτρια πιο κοντά στον ακαδημαϊκό κλάδο και τρόπο σκέψης, όπως και στα πιο καίρια σημεία της έρευνας πάνω σε αυτούς τους τομείς, στην αιχμή του δόρατος στην τεχνολογία αυτή τη στιγμή.

10.2 Μελλοντικές επεκτάσεις

Ως συνέχιση αυτής της εργασίας, θα αποπειραθεί μια δομημένη συγκεντρωτική έκδοση του κώδικα ως πακέτο για ελεύθερη επαναχρησιμοποίηση στο διαδίκτυο, σε ιστότοπους που φιλοξενούν τέτοιου είδους έργα. Έπειτα θα εμπλουτιστεί περεταίρω το εικονικό περιβάλλον στο Unity, ίσως για να δημιουργηθεί μία πλήρης εμπειρία παιχνιδιού με τις υλοποιημένες τεχνολογίες.

Θα ήταν σκόπιμο στο μέλλον να γίνει βαθύτερη ανάλυση μεθόδων και αλγορίθμων ΜΜ για την επίλυση του προβλήματος, όπως και η ανάπτυξη ενός βαθιού νευρωνικού δικτύου με πιο περίπλοκους μηχανισμούς και πολύ περισσότερα στρώματα. Σίγουρα θα βοηθούσε επίσης στην εκπαίδευση η συλλογή πολλών περισσότερων δεδομένων / ρίψεων. Επιπλέον, θα μπορούσε να επεκταθεί το φάσμα της εργασίας σε περισσότερα είδη κινήσεων με το χειριστήριο, αλλά και βολές από τυχαία και διαφορετικά σημεία.

Όπως αναφέρθηκε και εισαγωγικά, ο απώτερος σκοπός μίας τέτοιας προσπάθειας θα ήταν η αξιοποίηση κάποιου μοντέλου ΜΜ για τη δημιουργία κάποιου ευφυούς πράκτορα, ο οποίος να μπορεί να παίζει στο παιχνίδι αντί του παίκτη ή απέναντί του. Αυτό το εγχείρημα είναι φιλόδοξο και εκτενές, και θα μπορούσε να αποτελέσει μία αρκετά πρόσφορη μελλοντική επέκταση της παρούσας εργασίας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Berkman, M.I. (2018). History of Virtual Reality. In: Lee, N. (eds) Encyclopedia of Computer Graphics and Games. Springer, Cham. https://doi.org/10.1007/978-3-319-08234-9_169-1

[2] A. L. Fradkov, “Early history of machine learning,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1385–1390, 2020. doi:10.1016/j.ifacol.2020.12.1888

[3] Μηχανή Unity 3D: https://unity.com

[4] O. Bahceci, A. Pena-Rios, G. Buckingham and A. Conway, "Supervised Machine Learning Hand Gesture Classification in VR for Immersive Training," 2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), Christchurch, New Zealand, 2022, pp. 748-749, doi: 10.1109/VRW55335.2022.00225.

[5] A. G. Moore, R. P. McMahan, H. Dong and N. Ruozzi, "Extracting Velocity-Based User-Tracking Features to Predict Learning Gains in a Virtual Reality Training Application," 2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Porto de Galinhas, Brazil, 2020, pp. 694-703, doi: 10.1109/ISMAR50242.2020.00099.

[6] Nair, V., Guo, W., Mattern, J., et al. “Unique Identification of 50,000+ Virtual Reality Users from Head & Hand Motion Data.” arXiv.Org. Available at: https://arxiv.org/abs/2302.08927v1.

[7] Παιχνίδι VR Beat Saber: https://beatsaber.com

[8] Z. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). “LightGBM: a highly efficient gradient boosting decision tree”, NIPS’17: Proceedings of the 31st International Conference on Neural Information Processing Systems. Pages 3149-3157 https://dl.acm.org/doi/10.5555/3294996.3295074

[9] C. Song and S. Zarar, "Sensor Fusion for Learning-based Tracking of Controller Movement in Virtual Reality," 2019 27th European Signal Processing Conference (EUSIPCO), A Coruna, Spain, 2019, pp. 1-5, doi: 10.23919/EUSIPCO.2019.8902570.

[10] Βιβλιοθήκη και εφαρμογή SteamVR: https://www.steamvr.com

[11] Sherstinsky, A. “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network,” *Physica D: Nonlinear Phenomena*, V. 404, 2020, p. 132306.

[12] Liebers, J., Abdelaziz, M., Mecke, L., et al. “Understanding User Identification in Virtual Reality Through Behavioral Biometrics and the Effect of Body Normalization,” *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021.

[13] Sharma, S., and Athaiya, A. “ACTIVATION FUNCTIONS IN NEURAL NETWORKS,” *International Journal of Engineering Applied Sciences and Technology*, V. 04, No. 12, 2020, pp. 310–6.

[14] K. P. Murphy, “Types of Machine Learning,” in Machine learning: A probabilistic perspective, Cambridge, MA: MIT Press, 2012, pp. 2–3

[15] Aized Amin Soofi and Arshad Awan, “Classification techniques in Machine Learning: Applications and issues,” *Journal of Basic & Applied Sciences*, vol. 13, pp. 459–465, 2017. doi:10.6000/1927-5129.2017.13.76

[16] Osisanwo F.Y, Aknisola J.E.T, Awodele O., Hinmikaiye J. O, Olakamni O, and Akinjobi J, “Supervised Machine Learning Algorithms: Classification and Comparison,” International Journal of Computer Trends and Technology, vol. 48, no. 3, pp. 128–138, Jun. 2017, doi: 10.14445/22312803/ijctt-v48p126

[17] Bishop, C. M. “Neural networks and their applications,” Review of Scientific Instruments, V. 65, No. 6, 1994, pp. 1803–32

[18] Εργαλειοθήκη TensorFlow: https://www.tensorflow.org

[19] Joseph, F. J. J., Nonsiri, S., and Monsakul, A. “Keras and TensorFlow: A Hands-On Experience,” Advanced Deep Learning for Engineers and Scientists, 2021, pp. 85–111

[20] Γλώσσα Python: https://www.python.org

[21] Εργαλειοθήκη PyTorch: https://pytorch.org

[22] Πρότυπο OpenXR: https://www.khronos.org/openxr/

[23] Συσκευή VR HTC Vive Pro 2: https://www.vive.com/us/product/vive-pro2/specs/

[24] Εργαλειοθήκη XR Interaction Manager: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@ 2.0/manual/xr-interaction-manager.html#:~:text=The%20Interaction%20Manager%20acts%20as,set%20of%20Interactors%20and%20Interactables.

[25] Papadopoulos, G., Doumanoglou, A., & Zarpalas, D. (2023, June 14). *VRGestures: Controller and hand gesture datasets for virtual reality*. Zenodo. https://zenodo.org/records/8027807

[26] Ponton, J. L., Yun, H., Andujar, C., & Pelechano, N. (2022). Combining motion matching and orientation prediction to animate avatars for consumer-grade VR devices. Computer Graphics Forum, 41(8), 107–118. https://doi.org/10.1111/cgf.14628

[27] Pandey, R., Pidlypenskyi, P., Yang, S., & Kaeser-Chen, C. (n.d.). HMD Controller Dataset. https://sites.google.com/view/hmd-controller-dataset

[28] Editor Visual Studio Code: https://code.visualstudio.com

[29] Sarker, I.H. Machine Learning: Algorithms, Real-World Applications and Research Directions. SN COMPUT. SCI. 2, 160 (2021). https://doi.org/10.1007/s42979-021-00592-x

[30] Han, X., Zhang, Z., Ding, N., et al. “Pre-trained models: Past, present and future,” AI Open, V. 2, 2021, pp. 225–50.

[31] Πλατφόρμα Kaggle: https://www.kaggle.com

[32] Janiesch, C., Zschech, P. & Heinrich, K. Machine learning and deep learning. Electron Markets 31, 685–695 (2021). https://doi.org/10.1007/s12525-021-00475-2

[33] Περιβάλλον ανάπτυξης Spyder IDE: https://www.spyder-ide.org

[34] Σύστημα Conda: https://docs.conda.io/en/latest/
[35] Εργαλειοθήκη CUDA: https://developer.nvidia.com/cuda-toolkit
[36] Κάρτες γραφικών nVidia: https://www.nvidia.com/en-eu/
[37] Συνάρτηση ενεργοποίησης ReLU: https://paperswithcode.com/method/relu
[38] Vujovic, Ž. Đ. "Classification Model Evaluation Metrics," International Journal of Advanced Computer Science and Applications, V. 12, No. 6, 2021.
[39] Berrar D., "Cross-validation," Encyclopedia of Bioinformatics and Computational Biology, pp. 542–545, 2019. doi:10.1016/b978-0-12-809633-8.20349-x
[40] Εργαλειοθήκη MLagents: https://github.com/Unity-Technologies/ml-agents
[41] Πακέτο Sentis: https://docs.unity3d.com/Packages/com.unity.sentis@1.0/manual/index.html
[42] Ανοικτή σουίτα πρωτοκόλλων ONNX: https://onnx.ai
[43] Classifiers που χρησιμοποιούνται από το scikit-learn: http://onnx.ai/sklearn-onnx/supported.html
[44] Classifiers που υποστηρίζει το Sentis: https://docs.unity3d.com/Packages/com.unity.sentis@1.0/manual/supported-operators.html#supported-operators
[45] Βιβλιοθήκη skl2onnx: https://pypi.org/project/skl2onnx/
[46] Βιβλιοθήκη tf2onnx https://github.com/onnx/tensorflow-onnx
[47] Εφαρμογή εικονοποίησης μοντέλων Netron: https://github.com/lutzroeder/netron
[48] Πακέτο MathNet.Numerics: https://numerics.mathdotnet.com
[49] Thumbs-up 3D model asset: https://sketchfab.com/3d-models/thumbs-up-20acd8d826d2457d88135f094a161f4f
[50] Sound visualization asset: https://assetstore.unity.com/packages/tools/audio/simplespectrum-free-audio-spectrum-generator-webgl-85294
[51] Ιστοσελίδα text-to-speech: https://ttsmaker.com

ΠΑΡΑΡΤΗΜΑ Α : ΚΩΔΙΚΑΣ

```python
import csv

def move_last_to_first(input_csv, output_csv):
    with open(input_csv, 'r') as infile, open(output_csv, 'w', newline='') as outfile:
        reader = csv.reader(infile, delimiter=';')
        writer = csv.writer(outfile, delimiter=';')


        for row in reader:
            if row:
                # Find the index of the last non-empty cell
                last_non_empty_index = len(row) - 1
                while last_non_empty_index >= 0 and not row[last_non_empty_index]:
                    last_non_empty_index -= 1

                # Move the last non-empty cell to the first position
                row = [row[last_non_empty_index]] + row[:last_non_empty_index] + row[last_non_empty_index+1:]
            else:
                row = []

            # Write the modified row to the output CSV
            writer.writerow(row)

if __name__ == "__main__":
    input_csv_file = "xTest.csv"  # Replace with the path to your input CSV file
    output_csv_file = "xTest1.csv"  # Replace with the desired path for the output CSV file

    move_last_to_first(input_csv_file, output_csv_file)
```

Μεταφορά της τελευταίας στήλης της εκάστοτε γραμμής του αρχείου δεδομένων, στην πρώτη στήλη.

```python
import csv

def add_header_and_modify(input_csv, output_csv):
    with open(input_csv, 'r') as infile, open(output_csv, 'w', newline='') as outfile:
        # Explicitly specify the delimiter (use the correct delimiter for your CSV file)
        reader = csv.reader(infile, delimiter=';')
        writer = csv.writer(outfile, delimiter=';')

        # Determine the number of columns in the input CSV
        num_columns = len(next(reader))

        max_columns = 0
        row_with_most_columns = None

        for row in reader:
            num_columns = len(row)

            if num_columns > max_columns:
                max_columns = num_columns
                row_with_most_columns = row

        infile.seek(0)

        # Read the header row
        header_row = next(reader)

        # Header row
        new_header_row = ["class", "cameraX"]
        for i in range(2, (abs(max_columns // 7) + 2)):
            new_header_row.extend(["velocity", "posX", "posY", "posZ", "rotX", "rotY", "rotZ"])

        # Write the header row to the output CSV
        writer.writerow(new_header_row)
        writer.writerow(header_row)

        for row in reader:
            writer.writerow(row)

if __name__ == "__main__":
    input_csv_file = "xTest1.csv"  # Replace with the path to your input CSV file
    output_csv_file = "xTest2.csv"  # Replace with the desired path for the output CSV file

    add_header_and_modify(input_csv_file, output_csv_file)
```

Ονομασία κάθε στήλης με επικεφαλίδες, μέχρι το πέρας των στηλών του αρχείου δεδομένων.

```
public class FloatStack
{
    public int content;
    string prin;

    private List<float> stack;

    1 reference
    public FloatStack()
    {
        stack = new List<float>();
    }

    1 reference
    public void Push(float value)
    {
        stack.Add(value);
    }

    1 reference
    public float[] GetStackAsArray(int cont)
    {
        List<float> floatList = new List<float>();
        float[] floatArray = new float[cont];

        if (stack.Count < cont)
        {
            for (int i = 0; i < cont; i++)
            {
                if (i < stack.Count)
                {
                    floatArray[i] = stack[i];
                }
                else
                {
                    floatArray[i] = 0f;
                }
            }
        }

        else if (stack.Count > cont)
        {
            for (int i = 0; i < cont; i++)
            {
                floatArray[i] = stack[i];
            }
        }

        else if(stack.Count == cont)
        {
            floatArray = stack.ToArray();
        }

        for (int i = 0; i < 10; i++)
        {
            prin = prin + " " + floatArray[i].ToString();
        }

        return floatArray;
    }

    0 references
    public void PrintFloatStack()
    {
        foreach (float value in stack)
        {
            Debug.Log(value + " ");
        }
    }

    0 references
    public void ClearStack()
    {
        stack.Clear();
    }
}
```

Custom κλάση Float Stack για τον χειρισμό λίστας float, μετατροπή σε πίνακα, και κανονικοποίηση.

```
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class VoiceManager : MonoBehaviour
{
    public AudioSource audioSource;
    public AudioClip testingNow, mistakes, missed, goodJob, accuracy, directions;
    public MLprediction mlpred;
    public Shooter shooterScript;
    public bool testing = true;
    public bool te, mi, miss, go, acc, dir = false;

    Unity Message | 0 references
    void Start()
    {
        audioSource.clip = directions;
        audioSource.Play();
    }

    Unity Message | 0 references
    private void LateUpdate()
    {
        if (!audioSource.isPlaying && !te)
        {
            audioSource.clip = testingNow;
            audioSource.Play();
            te = true;
        }

        if ((shooterScript.missedInARow>2) && !mi)
        {
            audioSource.clip = mistakes;
            audioSource.Play();
            mi = true;
        }

        if ((shooterScript.madeInARow > 4) && !go)
        {
            audioSource.clip = goodJob;
            audioSource.Play();
            go = true;
        }

        if (mlpred.count>3 && mlpred.wrongPred && !miss)
        {
            audioSource.clip = missed;
            audioSource.Play();
            miss = true;
        }

        if (mlpred.perc > 89 && mlpred.count > 10 && !acc)
        {
            audioSource.clip = accuracy;
            audioSource.Play();
            acc = true;
        }
    }
}
```

Script Voice Manager για τη διαχείριση των ηχητικών κλιπ του μοντέλου στο Unity.