ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΡΟΗΓΜΕΝΕΣ ΤΕΧΝΟΛΟΓΙΕΣ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Μεταπτυχιακή Διπλωματική Εργασία

«Μελέτη και υλοποίηση Νευρωνικών Δικτύων με τη Γλώσσα Προγραμματισμού Python με έμφαση στη Μηχανική Μάθηση και την Επιστήμη των Δεδομένων»

Συγγραφέας

Του Επαμεινώνδα Μουμούρη

ΑΜ: 21015

Επιβλέπων Καθηγητής:

Φοίβος Μυλωνάς

Αθήνα,Φεβρουάριος,2023

UNIVERSITY OF WEST ATTICA

SCHOOL OF ENGINEERING

DEPARTMENT OF INFORMATICS AND COMPUTER ENGINEERING

ADVANCED COMPUTING SYSTEMS

Diploma Thesis

“Study and Implementation of Neural Networks with the Python Programming Language with emphasis on Machine Learning and Data Science “

Student Name and Surname: Epameinondas Moumouris

Registration Number: 21015

Supervisor Name and Surname:

Mylonas Foivos

Athens, February,2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΡΟΗΓΜΕΝΕΣ ΤΕΧΝΟΛΟΓΙΕΣ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

«Μελέτη και υλοποίηση Νευρωνικών Δικτύων με τη Γλώσσα Προγραμματισμού Python με έμφαση στη Μηχανική Μάθηση και την Επιστήμη των Δεδομένων»

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η μεταπτυχιακή διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1.	**Μυλωνάς Φοίβος**	Αναπληρωτής Καθηγητής	Φοίβος-Αποστολος Μυλωνάς - Ψηφιακά υπογεγραμμένο 2023.03.03 21:05:23 +02'00'
2.	**Χρήστος Τρούσσας**	Επίκουρος Καθηγητής	Christos Troussas Digitally signed by Christos Troussas Date: 2023.03.06 08:42:04 +02'00'
3.	**Πάρις Μαστοροκώστας**	Καθηγητής	Paris Mastorokostas Digitally signed by Paris Mastorokostas Date: 2023.03.04 09:43:13 +02'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/η κάτωθι υπογεγραμμένος Μουμούρης Επαμεινώνδας του Γεωργίου με αριθμό μητρώου 21015 φοιτητής του Προγράμματος Μεταπτυχιακών Σπουδών Προηγμένες Τεχνολογίες Υπολογιστικών Συστημάτων του Τμήματος Μηχανικών Πληροφορικής καθ υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

**Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι …… και έπειτα από αίτηση μου στη Βιβλιοθήκη και έγκριση του επιβλέποντα καθηγητή*

Ο/Η Δηλών/ούσα

*** Ονοματεπώνυμο /Ιδιότητα**
(Υπογραφή)

Φοίβος-Απόστολος Μυλωνάς Ψηφιακά υπογεγραμμένο
2023.03.03 21:05:41 +02'00'

Ψηφιακή Υπογραφή Επιβλέποντα

**** Σε εξαιρετικές περιπτώσεις και μετά από αιτιολόγηση και έγκριση του επιβλέποντα, προβλέπεται χρονικός περιορισμός πρόσβασης (embargo) 6-12 μήνες. Στην περίπτωση αυτή θα πρέπει να υπογράψει ψηφιακά ο/η επιβλέπων/ουσα καθηγητής/τρια, για να γνωστοποιεί ότι είναι ενημερωμένος/η και συναινεί. Οι λόγοι χρονικού αποκλεισμού πρόσβασης περιγράφονται αναλυτικά στις πολιτικές του Ι.Α. (σελ. 6):***

https://www.uniwa.gr/wp-content/uploads/2021/01/%CE%A0%CE%BF%CE%BB%CE%B9%CF%84%CE%B9%CE%BA%CE%B5%CC%81%CF%82_%CE%99%CE%B4%CF%81%CF%85%CE%BC%CE%B1%CF%84%CE%B9%CE%BA%CE%BF%CF%85%CC%81_%CE%91%CF%80%CE%BF%CE%B8%CE%B5%CF%84%CE%B7%CF%81%CE%B9%CC%81%CE%BF%CF%85_final.pdf).

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον Καθηγητή κύριο Φοίβο Μυλωνά για την καθοδήγηση και την υποστήριξη του καθ' όλη την διάρκεια της ακαδημαϊκής μου πορείας καθώς και στην παρούσα Μεταπτυχιακή Διπλωματική Εργασία.

Περίληψη

Τα Νευρωνικά Δίκτυα, γνωστά και ως Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ), είναι μια κατηγορία μοντέλων μηχανικής μάθησης που είναι εμπνευσμένη από την δομή και λειτουργία του ανθρώπινου εγκεφάλου. Η βασική ιδέα πίσω από τα Νευρωνικά Δίκτυα είναι η προσομοίωση της συμπεριφοράς του ανθρώπινου εγκεφάλου με τη δημιουργία ενός δικτύου διασυνδεδεμένων κόμβων ή νευρώνων, το οποίο μπορεί να μαθαίνει από δεδομένα και να κάνει προβλέψεις ή αποφάσεις. Τα Νευρωνικά Δίκτυα έχουν γίνει όλο και πιο δημοφιλή τα τελευταία χρόνια λόγω της ικανότητας τους να μαθαίνουν πολύπλοκα μοτίβα και να κάνουν ακριβείς προβλέψεις σε διάφορους τομείς, όπως η αναγνώριση εικόνας και ομιλίας, η επεξεργασία της φυσικής γλώσσας και η πρόβλεψη χρονοσειρών. Έχουν επίσης χρησιμοποιηθεί σε ποικίλες εφαρμογές, όπως αυτοοδηγούμενα αυτοκίνητα, συστήματα συστάσεων και ιατρική διάγνωση.

Σε αυτή τη διπλωματική, θα παράσχουμε μια επισκόπηση των βασικών αρχών των Νευρωνικών Δικτύων, συμπεριλαμβανομένης της αρχιτεκτονικής , της διαδικασίας εκπαίδευσης και των δημοφιλών αλγόριθμων βελτιστοποίησης καθώς και κάποια από τα συμπεράσματα στα οποία καταλήξαμε μέσα από την έρευνα μας. Για την υλοποίηση των μοντέλων μας χρησιμοποιήσαμε σαν κύρια γλώσσα προγραμματισμού την Python καθώς και όλα τα απαραίτητα Frameworks για να μπορέσουμε να έχουμε την ιδανική λειτουργία τους, κάποια από αυτά είναι: Keras, TensorFlow , Pandas και NumPy.

Τα συμπεράσματα στα οποία καταλήγουμε είναι ότι τα Νευρωνικά Δίκτυα μπορούν να μας βοηθήσουν σε πολλούς κλάδους και ιδιαίτερα στα πλαίσια της υγείας για να μπορούμε να έχουμε πιο ακριβή αποτελέσματα και να μπορούμε να προλαμβάνουμε τυχόν προβλήματα που μπορεί να προκύπτουν έγκαιρα. Συνολικά, τα Νευρωνικά Δίκτυα έχουν την δυνατότητα να φέρουν επανάσταση στον κλάδο της υγειονομικής περίθαλψης , βελτιώνοντας την διάγνωση ασθενειών, την ανακάλυψη φαρμάκων και την εξατομικευμένη θεραπεία.

Σαν αποτέλεσμα το πόσο μεγάλη επιρροή έχουν πλέον τα Νευρωνικά Δίκτυα στον τομέα της υγείας το αποδεικνύουμε και από το μοντέλο της Στεφανιαίας Νόσου που έχουμε δημιουργήσει καθώς έχουμε ποσοστά επιτυχίας **92%.**

Abstract

Neural Networks, also known as Artificial Neural Networks (ANNs), are a class of machine learning models inspired by the structure and function of the human brain. The basic idea behind Neural Networks is to simulate the behaviors of the human brain by creating a network of interconnected nodes or neurons that can learn from data and make predictions or decisions. Neural Networks have become increasingly popular in recent years due to their ability to learn complex patterns and make accurate predictions in various fields such as image and speech recognition, natural language processing and time series prediction. They have also been used in a variety of applications such as self-driving cars, recommender systems and medical diagnostics.

In this thesis, we will provide an overview of the fundamentals of Neural Networks, including the architecture, the training process and popular optimization algorithms as well as some of the conclusions we have reached through our research. For the implementation of our models, we used Python as the main programming language and all the necessary Frameworks to enable us to get the ideal functionality, some of them are Keras, TensorFlow, Pandas and NumPy.

The conclusions we come to is that Neural Networks can help us in many disciplines and especially in the context of health to be able to get more accurate results and to be able to prevent any problems that may arise in time. Overall, Neural Networks have the potential to revolutionize the healthcare industry, improving disease diagnosis, drug discovery and personalized treatment.

As a result, how influential Neural Networks are now in the healthcare sector is demonstrated by the Coronary Disease model we have created as we have a 92% success rate.

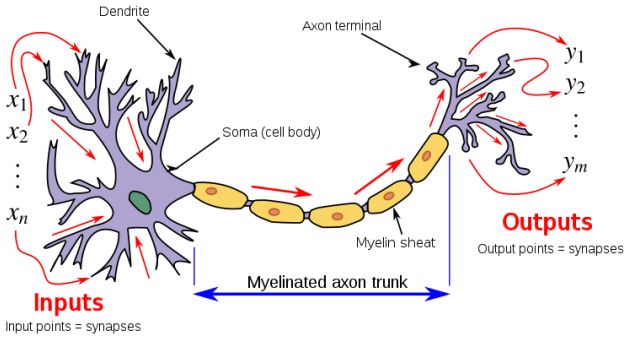
ΠΕΡΙΕΧΟΜΕΝΑ

Παράρτημα Εικόνων

Κεφάλαιο 1 Νευρωνικά Δίκτυα και Ανάλυση

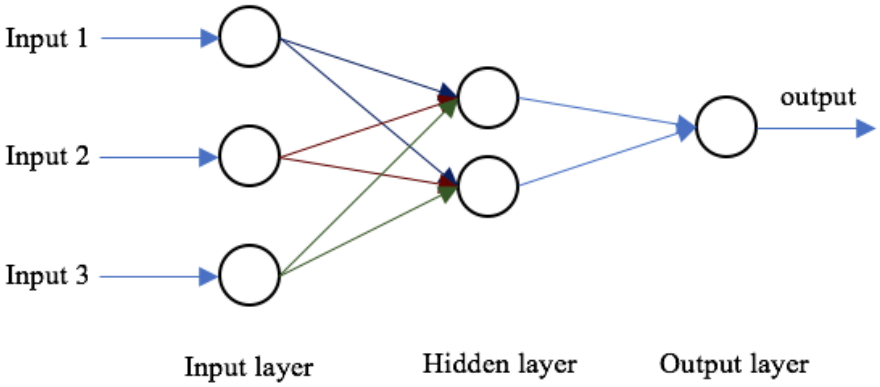
1.1 Εισαγωγή στα Νευρωνικά Δίκτυα

Από βιολογική άποψη, οι νευρώνες αποτελούν μέρος του κεντρικού νευρικού συστήματος και του ανθρώπινου εγκεφάλου. Εκτός από τον ζωντανό κόσμο, στη σφαίρα των Τεχνητών Νευρωνικών Δικτύων της Επιστήμης των Υπολογιστών, ένας νευρώνας είναι μια συλλογή από ένα σύνολο εισόδων, ένα σύνολο βαρών και μια συνάρτηση ενεργοποίησης. Μεταφράζει αυτές τις εισόδους σε μια ενιαία έξοδο. Ένα άλλο στρώμα νευρώνων επιλέγει αυτή την έξοδο ως είσοδο. Στην ουσία, μπορούμε να πούμε ότι κάθε νευρώνας είναι μια μαθηματική συνάρτηση που προσομοιάζει στενά τη λειτουργία ενός βιολογικού νευρώνα.



Εικόνα 1 Βιολογικό Νευρωνικό Δίκτυο

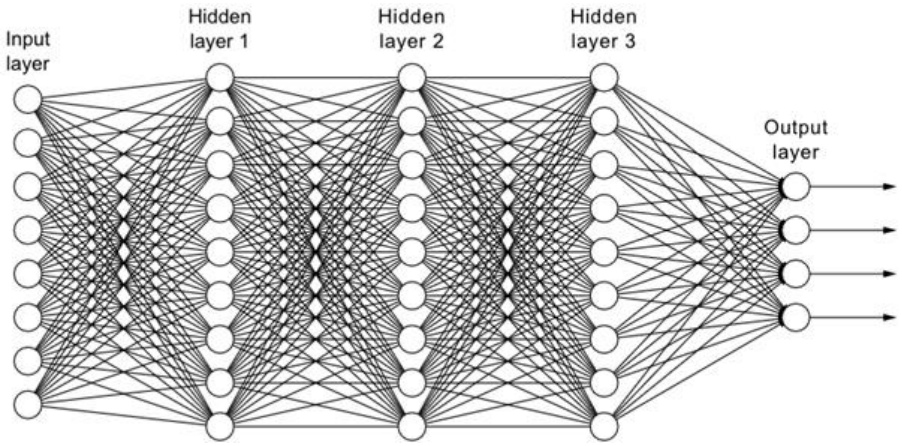
Οι υπολογιστικές μονάδες συνδέονται μεταξύ τους μέσω βαρών, τα οποία έχουν τον ίδιο ρόλο με τις δυνάμεις των συναπτικών συνδέσεων σε βιολογικούς οργανισμούς. Κάθε είσοδος σε έναν νευρώνα κλιμακώνεται με ένα βάρος, το οποίο επηρεάζει τη συνάρτηση που υπολογίζεται σε αυτή τη μονάδα.



Εικόνα 2 Τεχνητό Νευρωνικό Δίκτυο

Ένα Τεχνητό Νευρωνικό Δίκτυο εκτελεί μια λειτουργία μεταδίδοντας τις υπολογισμένες τιμές από τους νευρώνες εισόδου στους νευρώνες εξόδου χρησιμοποιώντας βάρη ως μεσαίες παραμέτρους. Τα βάρη που συσχετίζονται με τους νευρώνες μπορούν να ρυθμιστούν για να διευκολύνουν την διαδικασία της μάθησης. Στα Τεχνητά Νευρωνικά Δίκτυα, τα δεδομένα εκπαίδευσης τα περιέχουν ζεύγη εισόδου-εξόδου της προς εκμάθησης συνάρτησης, χρησιμεύουν ως εξωτερικό ερέθισμα. Για παράδειγμα, τα δεδομένα εκμάθησης μπορεί να περιλαμβάνουν αναπαραστάσεις στοιχείων Pixel εικόνων ως είσοδο και τις σχολιασμένες ετικέτες τους (πχ. «Καρότο», «Μπανάνα») ως έξοδο. Αυτά τα ζεύγη εισόδου εξόδου παρουσιάζονται στο Νευρωνικό Δίκτυο χρησιμοποιώντας τις αναπαραστάσεις εισόδου για να κάνει προβλέψεις σχετικά με τις ετικέτες εξόδου. Η ανατροφοδότηση παρέχεται στο Νευρωνικό Δίκτυο μέσω των δεδομένων εκπαίδευσης, ανάλογα με το πόσο καλά η προβλεπόμενη έξοδος ταιριάζει με την ετικέτα εξόδου που έχει σχολιαστεί. Το Νευρωνικό Δίκτυο προσαρμόζει τα βάρη μεταξύ των νευρώνων για να ελαχιστοποιήσει τα σφάλματα πρόβλεψης, παρόμοια με τον τρόπο με τον οποίο ένας βιολογικός οργανισμός προσαρμόζει τις συναπτικές δυνάμεις σε απόκριση ανάλογα με τα δεδομένα για να αποφύγει κακή ανατροφοδότηση. Στόχος της προσαρμογής των βαρών είναι η βελτίωση της ακρίβειας των μελλοντικών προβλέψεων. Μέσω διαδοχικών προσαρμογών των βαρών, η ικανότητα του Νευρωνικού Δικτύου να υπολογίζει συναρτήσεις βελτιώνεται με την πάροδο του χρόνου, οδηγώντας σε ακριβέστερες προβλέψεις. Αυτή η ικανότητα του Νευρωνικού Δικτύου να υπολογίζει με ακρίβεια αόρατες συναρτήσεις εισόδου μέσω της εκπαίδευσης σε ένα πεπερασμένο σύνολο ζευγών εισόδου-εξόδου αναφέρεται κιόλας ως Γενίκευση Μοντέλου. Το κύριο όφελος των μοντέλων Μηχανικής Μάθησης είναι η ικανότητα τους να γενικεύουν τη μάθηση τους από τα προβλεπόμενα δεδομένα εκπαίδευσης σε παραδείγματα.

1.2 Αρχιτεκτονική Νευρωνικών Δικτύων



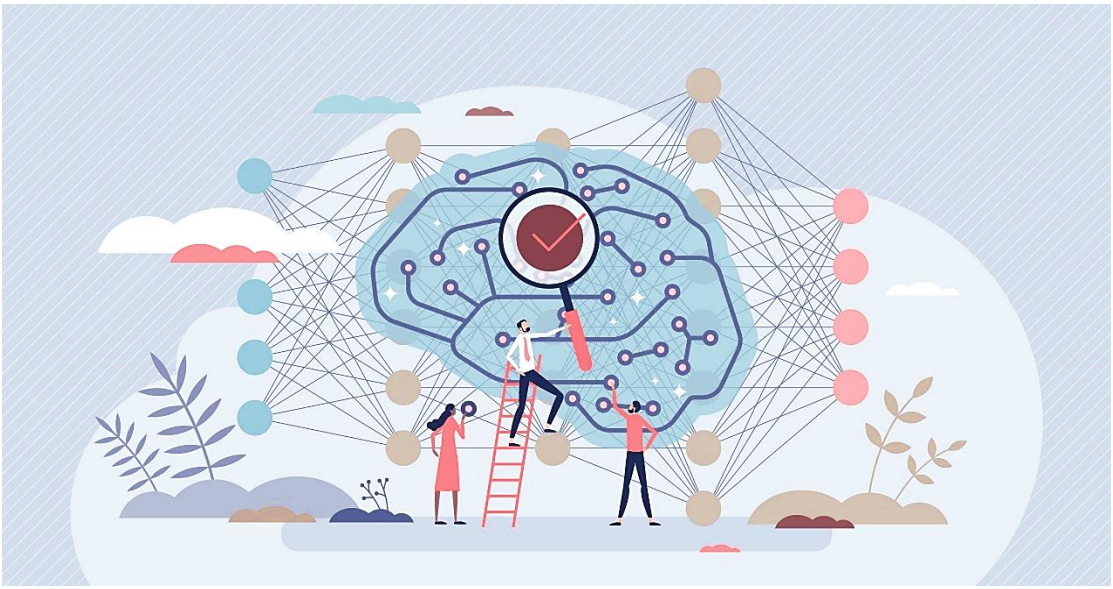
Εικόνα 3 Νευρωνικό Δίκτυο με Πολλαπλούς Κόμβους

Η αρχιτεκτονική ενός Νευρωνικού Δικτύου περιλαμβάνει επίπεδα κόμβων που κατανέμονται σε ένα επίπεδο εισόδου, ένα ή πολλαπλά κρυφά στρώματα και ένα στρώμα εξόδου. Οι κόμβοι είναι οι «Τεχνητοί Νευρώνες» συνδεδεμένοι μεταξύ τους και συνδέονται με ένα συγκεκριμένο βάρος και όριο. Αφού η έξοδος ενός μεμονωμένου κόμβου ξεπεράσει το καθορισμένο όριο, ο συγκεκριμένος κόμβος ενεργοποιείται και τα δεδομένα μεταδίδονται στο επόμενο επίπεδο δικτύου. Εάν δεν ξεπεραστεί η τιμή του ορίου του κόμβου, τα δεδομένα δεν μεταφέρονται στο επόμενο επίπεδο δικτύου.

Σε αντίθεση με τους παραδοσιακούς υπολογιστές, οι οποίοι επεξεργάζονται δεδομένα διαδοχικά, τα Νευρωνικά Δίκτυα έχουν την δυνατότητα να μάθουν και να κάνουν πολλαπλές εργασίες ταυτόχρονα. Με άλλα λόγια, ενώ οι συμβατικοί υπολογιστές ακολουθούν μόνο τις οδηγίες του προγραμματισμού τους, τα Νευρωνικά Δίκτυα εξελίσσονται συνεχώς μέσω προηγμένων αλγορίθμων. Μπορεί να ειπωθεί ότι οι Νευρικοί Υπολογιστές «Προγραμματίζονται» για να εξάγουν λύσεις σε προβλήματα που δεν είχαν δει προηγουμένως. Επιπλέον, οι παραδοσιακοί υπολογιστές λειτουργούν χρησιμοποιώντας λογικές συναρτήσεις και βασίζονται σε ένα συγκεκριμένο σύνολο υπολογισμών και κανόνων. Αντίθετα, οι Νευρωνικοί υπολογιστές μπορούν να επεξεργάζονται λογικές λειτουργίες και ακατέργαστες εισόδους-εξόδους, όπως εικόνες βίντεο και φωνή.

Ενώ οι παραδοσιακοί υπολογιστές είναι έτοιμοι χωρίς καμία παραμετροποίηση, τα Νευρωνικά Δίκτυα πρέπει να «εκπαιδεύονται» με την πάροδο του χρόνου για να αυξάνουν την ακρίβεια και την αποτελεσματικότητα τους. Η ακριβής ρύθμιση αυτών των μηχανών εκμάθησης αποδίδει πολλά οφέλη, δίνοντας στους χρήστες ένα ισχυρό υπολογιστικό εργαλείο σε εφαρμογές τεχνητής νοημοσύνης και επιστήμης υπολογιστών.

Τα Νευρωνικά Δίκτυα είναι ικανά να ταξινομούν και να ομαδοποιούν δεδομένα σε υψηλές ταχύτητες. Αυτό σημαίνει, μεταξύ άλλων, ότι μπορούν να ολοκληρώσουν την αναγνώριση της ομιλίας και των εικόνων μέσα σε λίγα λεπτά αντί για τις ώρες που θα χρειαζόταν όταν πραγματοποιούνταν από κανονικού ανθρώπους. Το πιο συχνά χρησιμοποιούμενο Νευρωνικό Δίκτυο σήμερα είναι οι αλγόριθμοι αναζήτησης Google.



Εικόνα 4 Νευρωνικά Δίκτυα και Ανθρώπινος Εγκέφαλος

1.3 Πως λειτουργεί ένα Νευρωνικό Δίκτυο

Η ικανότητα ενός Νευρωνικού Δικτύου να «σκέφτεται» έχει φέρει επανάσταση στους υπολογιστές όπως τους ξέραμε. Αυτές οι έξυπνες λύσεις είναι ικανές να ερμηνεύσουν

δεδομένα και να αναγνωρίζουν το περιεχόμενο πολλών πραγμάτων και συνδυασμών.

Τέσσερα **κρίσιμα βήματα** που κάνουν τα Νευρωνικά Δίκτυα να λειτουργούν αποτελεσματικά είναι:

- Η **Συσχέτιση** ή η εκπαίδευση επιτρέπει στα Νευρωνικά Δίκτυα να «θυμούνται» μοτίβα. Εάν εμφανιστεί στον υπολογιστή ένα άγνωστο μοτίβο, θα συσχετίσει το μοτίβο με το πλησιέστερο ταίριασμα που υπάρχει στη μνήμη του.

- **Ταξινόμηση** ή οργάνωση δεδομένων ή μοτίβων σε προκαθορισμένες κλάσεις

- **Ομαδοποίηση** ή ο προσδιορισμός μια μοναδικής πτυχής κάθε στιγμιότυπου δεδομένων για την ταξινόμηση του ακόμα και χωρίς κανένα άλλο πλαίσιο.

- **Πρόβλεψη** ή παραγωγή αναμενόμενων αποτελεσμάτων χρησιμοποιώντας μια σχετική είσοδο, ακόμα και όταν δεν παρέχεται εκ των προτέρων όλο το πλαίσιο.

Τα Νευρωνικά Δίκτυα απαιτούν υψηλή απόδοση για να εκτελούν αυτές τις λειτουργίες με ακρίβεια σχεδόν σε πραγματικό χρόνο. Αυτό επιτυγχάνεται με την ανάπτυξη πολλών επεξεργαστών που λειτουργούν παράλληλα μεταξύ τους, οι οποίοι είναι διατεταγμένοι σε επίπεδα.

Η διαδικασία της Νευρωνικής Δικτύωσης ξεκινά με το πρώτο στρώμα, το οποίο λαμβάνει τα ακατέργαστα δεδομένα, παρόμοια με τον τρόπο που τα οπτικά νεύρα λαμβάνουν τις οπτικές εισροές στον άνθρωπο. Στη συνέχεια, κάθε επόμενο στρώμα λαμβάνει τα αποτελέσματα από το προηγούμενο στρώμα, και αυτό συνεχίζεται μέχρι το τελευταίο στρώμα να επεξεργαστεί τις πληροφορίες και να παράγει την έξοδο. Κάθε μεμονωμένος κόμβος επεξεργασίας έχει ήδη δική του βάση δεδομένων, οποία περιλαμβάνει όλες τις προηγούμενες πληροφορίες και κανόνες με τους οποίους είτε είχε αρχικά προγραμματιστεί ή αναπτύχθηκε με την πάροδο του χρόνου.

Μόλις το Νευρωνικό Δίκτυο δομηθεί για μια συγκεκριμένη εφαρμογή, αρχίζει η διαδικασία εκμάθησης, γνωστή και ως εκπαίδευση. Υπάρχουν δυο προσεγγίσεις για την εκπαίδευση :

- Εκπαίδευση με Επίβλεψη
- Εκπαίδευση χωρίς Επίβλεψη

Στην προσέγγιση με επίβλεψη , το δίκτυο εφοδιάζεται με σωστές εξόδους δίδοντας τον επιθυμητό συνδυασμό εισόδου και εξόδου είτε αξιολογώντας χειροκίνητα την απόδοση του δικτύου. Αντίθετα η εκπαίδευση χωρίς επίβλεψη λαμβάνει χώρα όταν τον δίκτυο ερμηνεύει τις εισόδους και παράγει εξόδους εξωτερικές οδηγίες ή υποστήριξη.

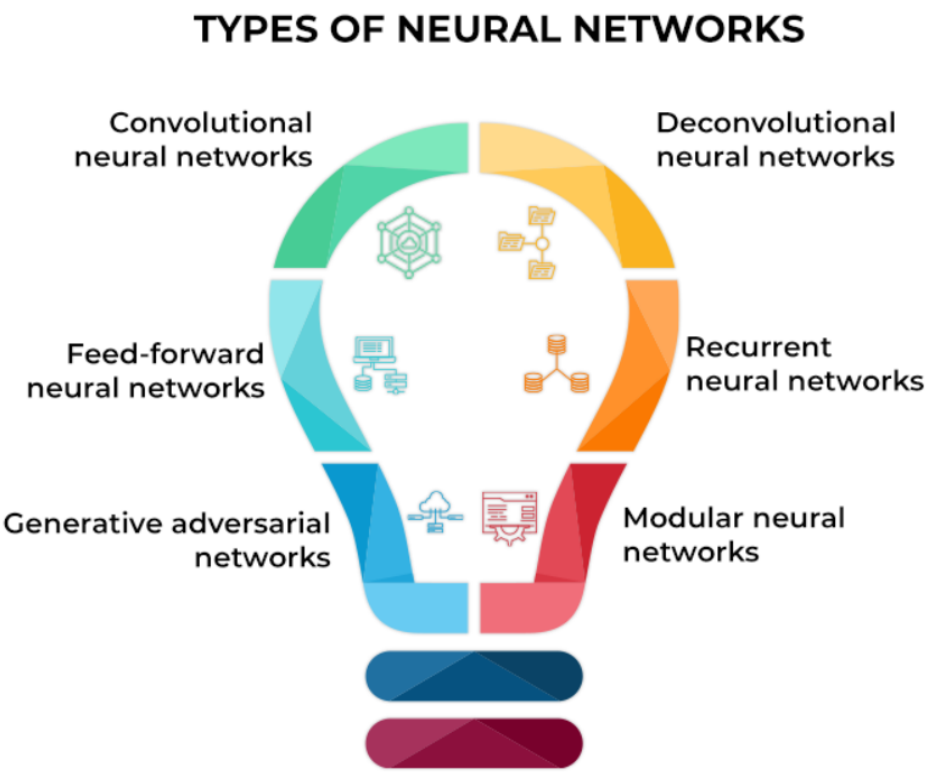
Η προσαρμοστικότητα είναι μία από τις βασικές ιδιότητες ενός Νευρωνικού Δικτύου. Αυτό το χαρακτηριστικό επιτρέπει στους αλγόριθμους μηχανικής μάθησης να τροποποιούνται καθώς μαθαίνουν από την εκπαίδευση τους και τις επόμενες λειτουργίες τους. Τα μοντέλα εκμάθησης επικεντρώνονται βασικά γύρω από το βάρος των ροών εισόδου, όπου κάθε κόμβος εκχωρεί ένα βάρος στα δεδομένα εισόδου που λαμβάνει από τους προηγούμενους κόμβους. Οι εισροές που αποδεικνύονται καθοριστικές για την εξαγωγή των σωστών απαντήσεων έχουν μεγαλύτερη βαρύτητα στις επόμενες διαδικασίες. Εκτός από την προσαρμοστικότητα, τα Νευρωνικά Δίκτυα αξιοποιούν πολλές αρχές για να καθορίσουν τους κανόνες λειτουργίας τους και να κάνουν προσδιορισμούς. Η ασαφής λογική (Fuzzy Logic), η εκπαίδευση που βασίζεται σε κλίση (Gradient-Based Training), οι μέθοδοι Bayes και οι γενετικοί αλγόριθμοι παίζουν όλα ρόλο στη διαδικασία λήψης αποφάσεων σε επίπεδο κόμβου. Αυτό βοηθά στους μεμονωμένους κόμβους να αποφασίσουν τι θα πρέπει να σταλεί στο επόμενο επίπεδο με βάση τις εισόδους που λαμβάνονται από το προηγούμενο επίπεδο. Οι βασικοί κανόνες για τις σχέσεις των αντικειμένων μπορούν επίσης να βοηθήσουν στη διασφάλιση μοντελοποίησης δεδομένων υψηλότερης ποιότητας. Για παράδειγμα, ένα Νευρωνικό Δίκτυο μπορεί να λάβει οδηγίες όπως «τα δόντια βρίσκονται πάντα κάτω από την μύτη» ή «τα αυτιά βρίσκονται σε κάθε πλευρά του προσώπου». Η μη αυτόματη προσθήκη τέτοιων κανόνων μπορεί να βοηθήσει στη μείωση του χρόνου εκπαίδευσης και να βοηθήσει στη δημιουργία ενός πιο αποτελεσματικού μοντέλου Νευρωνικών Δικτύων .

Ωστόσο, η προσθήκη κανόνων δεν είναι πάντα καλό. Κάτι τέτοιο μπορεί επίσης να οδηγήσει σε λανθασμένες υποθέσεις όταν ο αλγόριθμος προσπαθεί να λύσει προβλήματα που δεν σχετίζονται με τους κανόνες. Η προφόρτωση του λανθασμένου συνόλου κανόνων μπορεί να οδηγήσει στη δημιουργία Νευρωνικών Δικτύων που παρέχουν άσχετα, λανθασμένα μη χρήσιμα ή αντιπαραγωγικά αποτελέσματα. Αυτό καθιστά απαραίτητη την προσεκτική επιλογή των κανόνων που προστίθενται στο σύστημα.

Ενώ η Νευρωνική Δικτύωση, και ειδικά η μάθηση χωρίς επίβλεψη, έχουν ακόμα πολύ διαδικασία ανάπτυξης για να μπορούν να φτάσουν στην τελειότητα, μπορεί να είμαστε πιο κοντά στην επίτευξη μιας καθοριστικής ανακάλυψης από όσο νομίζουμε. Είναι γεγονός ότι οι συνδέσεις μέσα σε ένα Νευρωνικό Δίκτυο δεν είναι πουθενά τόσο πολυάριθμες ή αποτελεσματικές όσο αυτές στον ανθρώπινο εγκέφαλο. Ωστόσο, ο νόμος του Moore, ο οποίος δηλώνει ότι η μέση επεξεργαστική ισχύς των υπολογιστών αναμένεται να διπλασιάζεται κάθε δύο χρόνια, εξακολουθεί να ανθίζει. Αυτή η τάση δίνει στις προσδοκίες μας από την Τεχνητή Νοημοσύνη και τα Νευρωνικά Δίκτυα μια οριστική κατεύθυνση.

1.4 Τύποι Νευρωνικών Δικτύων

Τα Νευρωνικά Δίκτυα ταξινομούνται με βάση διάφορους παράγοντες, συμπεριλαμβανομένου του βάθους τους, του αριθμού των κρυφών επιπέδων και των δυνατοτήτων εισόδου/εξόδου κάθε κόμβου που διαθέτουν.

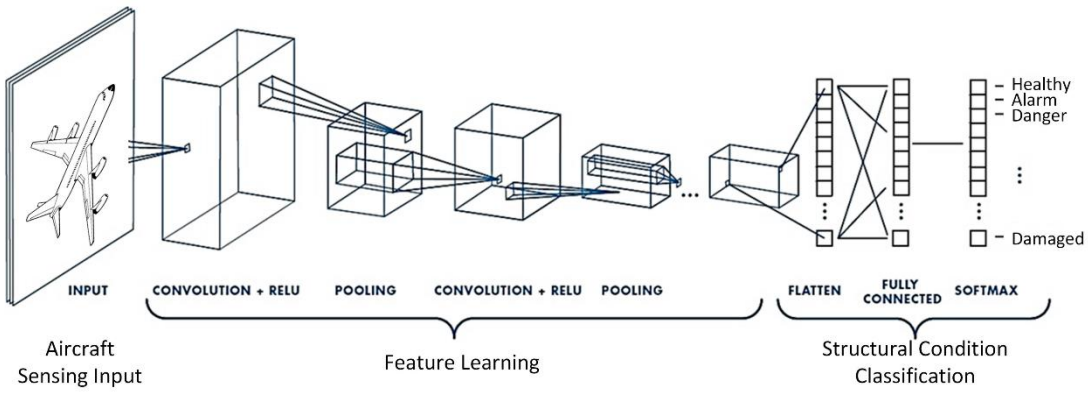


Εικόνα 5 Τύποι Νευρωνικών Δικτύων

Παρακάτω θα δούμε τους διάφορους τύπους Νευρωνικών Δικτύων όπου χρησιμοποιούνται για τις διάφορες προσεγγίσεις που θέλουμε να κάνουμε και με τα αποτελέσματα που θέλουμε να πάρουμε

- **Συνελκτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks)**

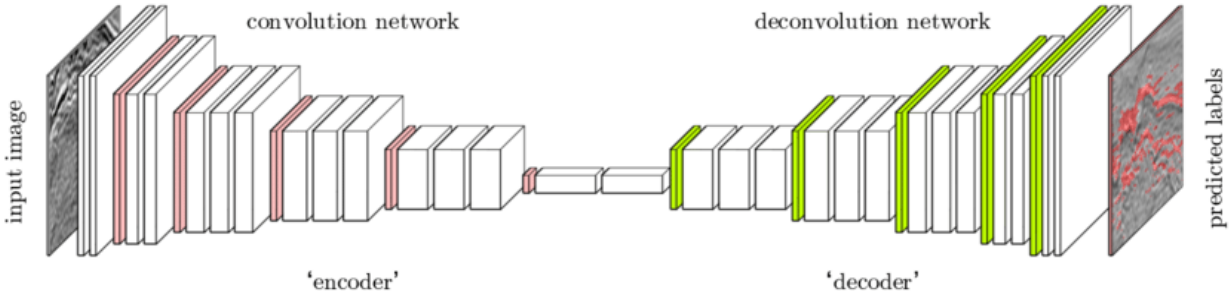
Όντας ένα εξαιρετικά δημοφιλές μοντέλο Νευρωνικής Δικτύωσης, τα Συνελκτικά Νευρωνικά Δίκτυα αξιοποιούν έναν τύπο πολυστρωματικού Perceptron και περιλαμβάνουν ένα ή περισσότερα Συνελκτικά στρώματα. Αυτά τα στρώματα μπορούν να είναι είτε ομαδοποιημένα ή εξ ολοκλήρου συνδεδεμένα. Αυτό το μοντέλο Νευρωνικής Δικτύωσης χρησιμοποιεί αρχές από γραμμική άλγεβρα, ειδικά τον πολλαπλασιασμό πινάκων, για να ανιχνεύσει και να επεξεργαστεί μοτίβα μέσα στις εικόνες. Τα Συνελκτικά επίπεδα σε αυτό το μοντέλο μπορούν χάρτες χαρακτηριστικών που καταγράφουν μια συγκεκριμένη περιοχή μέσα σε μία οπτική είσοδο. Στη συνέχεια, ο ιστότοπος αναλύεται περαιτέρω και αναλύεται για να δημιουργήσει πολύτιμα αποτελέσματα. Τα Συνελκτικά Νευρωνικά Δίκτυα είναι ωφέλιμα για εφαρμογές αναγνώρισης εικόνας που τροφοδοτούνται από AI (Artificial Intelligence). Αυτός ο τύπος Νευρωνικού Δικτύου χρησιμοποιείται συνήθως σε περιπτώσεις προηγμένων χρήσεων όπως αναγνώριση προσώπου, ή επεξεργασία φυσικής γλώσσας (Natural Language Processing – NLP), οπτική αναγνώριση χαρακτήρων (Optical Character Recognition - OCR) και η ταξινόμηση εικόνων. Χρησιμοποιείται επίσης και αναγνώριση παράφρασης και επεξεργασία σήματος.



Εικόνα 6 Συνελκτικό Νευρωνικό Δίκτυο

- **Αποσυνελκτικά Νευρωνικά Δίκτυα (Deconvolutional Neural Networks)**

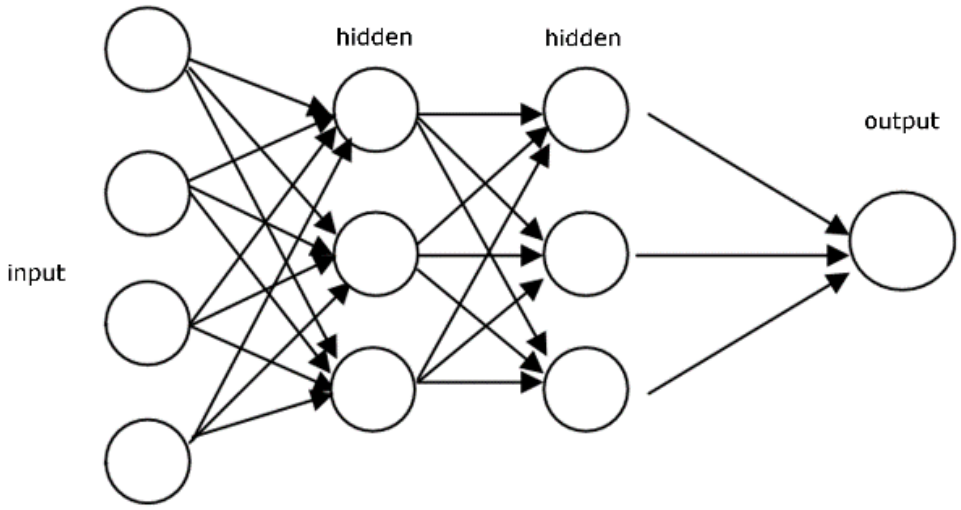
  Τα Αποσυνελκτικά Νευρωνικά Δίκτυα λειτουργούν με τις ίδιες αρχές με τα Συνελκτικά Δίκτυα, εκτός από το αντίστροφο. Αυτή η συγκεκριμένη εφαρμογή της τεχνητής νοημοσύνης στοχεύει στον εντοπισμό χαμένων σημάτων ή χαρακτηριστικών που μπορεί να είχαν προηγουμένως απορριφθεί σαν ασήμαντες πληροφορίες όσο το δίκτυο εκτελούσε την αποστολή που το είχε ανατεθεί. Τα Νευρωνικά Δίκτυα Αποσυνέλξης είναι χρήσιμα για διάφορες εφαρμογές, συμπεριλαμβανομένης της ανάλυσης και της σύνθεσης εικόνας



Εικόνα 7 Αποσυνελκτικό Νευρωνικό Δίκτυο

- **Νευρωνικά Δίκτυα Τροφοδοσίας (Feed-Forward Neural Networks)**

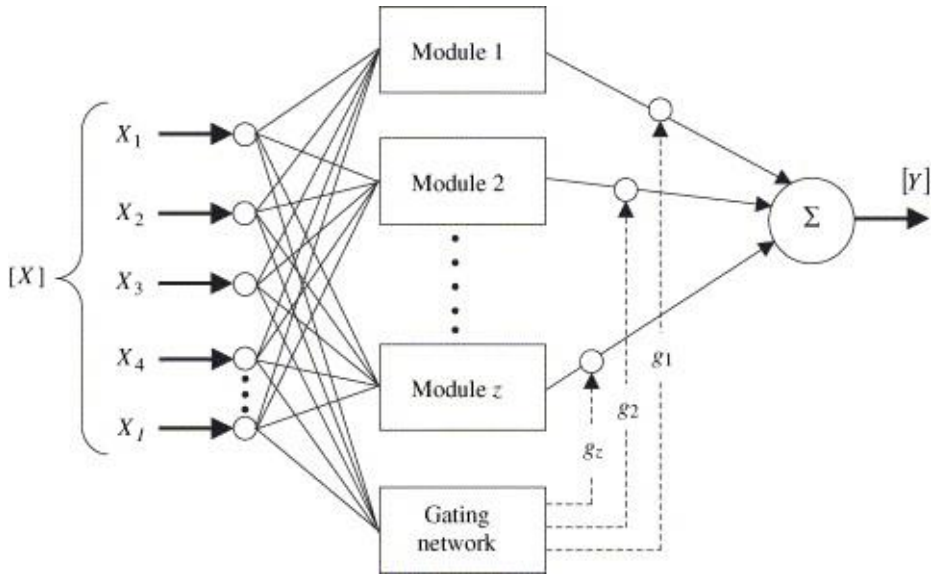
Αυτή η απλή παραλλαγή Νευρωνικού Δικτύου διοχετεύει δεδομένα σε μια μόνο κατεύθυνση μέσω διάφορων κόμβων επεξεργασίας έως ότου τα δεδομένα να φτάσουν στον κόμβο εξόδου. Τα Νευρωνικά Δίκτυα Τροφοδοσίας έχουν σχεδιαστεί για να επεξεργάζονται μεγάλους όγκους «θορυβωδών» δεδομένων και να δημιουργούν «καθαρές» εξόδους. Αυτός ο τύπος των Νευρωνικών Δικτύων είναι επίσης γνωστός ως μοντέλο Πολυστρωματικών Perceptons (Multi-Layer Perceptrons - MLPs) . Μια αρχιτεκτονική Νευρωνικού Δικτύου Τροφοδοσίας περιλαμβάνει το επίπεδο εισόδου, ένα ή περισσότερα κρυφά επίπεδα και το επίπεδο εξόδου. Παρά την ενναλακτική τους ονομασία, αυτά τα μοντέλα αξιοποιούν τους σιγμοειδείς νευρώνες (Sigmoid Neurons) παρά τα Perceptrons, επιτρέποντας τους έτσι να αντιμετωπίσουν μη γραμμικά προβλήματα του πραγματικού κόσμου. Τα Νευρωνικά Δίκτυα Τροφοδοσίας αποτελούν θεμέλιο για την αναγνώριση προσώπου, την επεξεργασία φυσικής γλώσσας, το Computer Vision, και άλλα μοντέλα Νευρωνικών Δικτύων.



Εικόνα 8 Νευρωνικό Δίκτυο Τροφοδοσίας

- **Αρθρωτά Νευρωνικά Δίκτυα (Modular Neural Networks)**

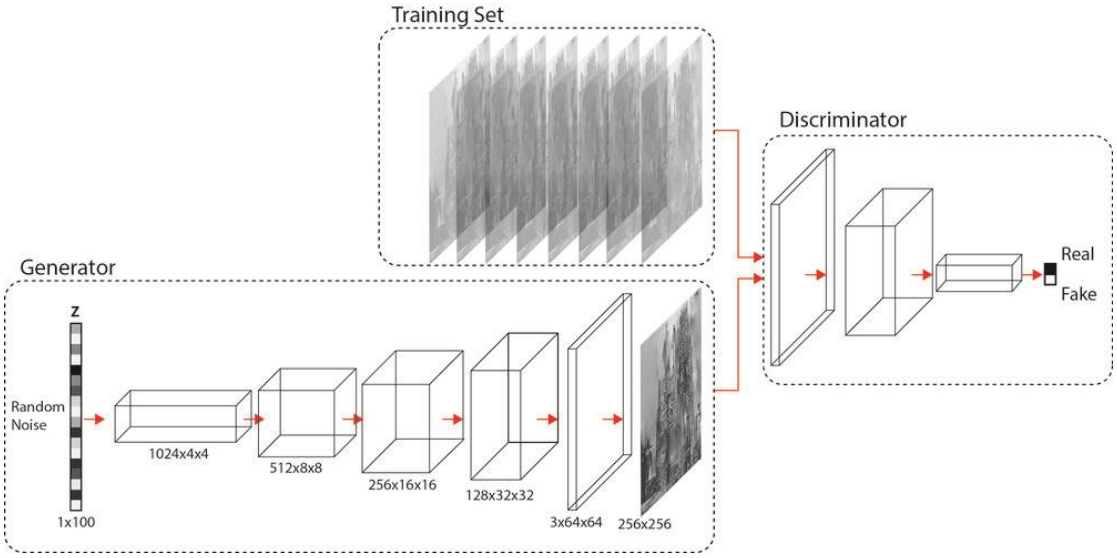
Τα Αρθρωτά Νευρωνικά Δίκτυα διαθέτουν μια σειρά από ανεξάρτητα Νευρωνικά Δίκτυα των οποίων οι λειτουργίες επιβάλλονται από έναν ενδιάμεσο. Κάθε ανεξάρτητο δίκτυο είναι μια «μονάδα» που χρησιμοποιεί διακριτές εισόδους για να ολοκληρώσει ένα συγκεκριμένο μέρος του συνολικού στόχου του μεγαλύτερου δικτύου. Οι μονάδες δεν επικοινωνούν μεταξύ τους ούτε παρεμβαίνουν η μια στις διαδικασίες της άλλης ενώ πραγματοποιείται ο υπολογισμός. Αυτό καθιστά την εκτέλεση εκτεταμένων και πολύπλοκών υπολογιστικών διαδικασιών πιο αποτελεσματικά.



Εικόνα 9 Αρθτωτικό Νευρωνικό Δίκτυο

- **Παραγωγικά Αντιθετικά Δίκτυα (Generative Adversarial Networks)**

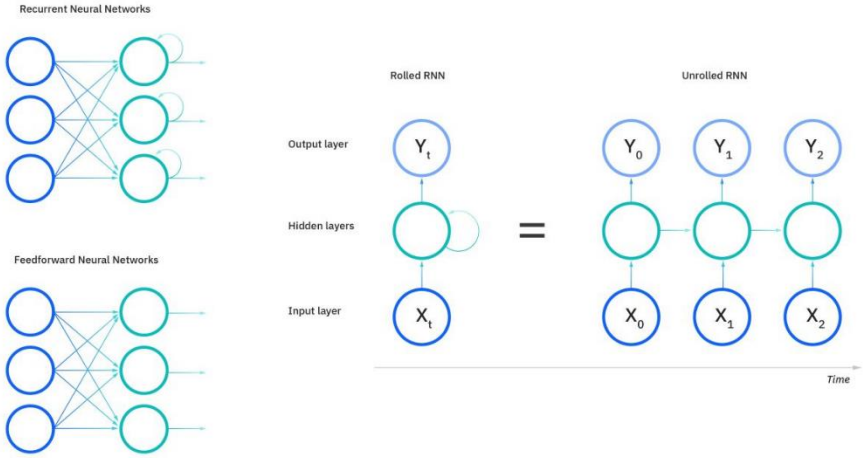
Τα Παραγωγικά Αντιθετικά Δίκτυα είναι μια λύση γενετικής μοντελοποίησης που αξιοποιεί τα Νευρωνικά Δίκτυα Συνελίξεων και άλλες πτυχές της μηχανικής μάθησης για την αυτοματοποίηση της ανακάλυψης μοτίβων στα δεδομένα. Η παραγωγική μοντελοποίηση χρησιμοποιεί μη επιβλεπόμενη μάθηση για την δημιουργία αληθοφανών συμπερασμάτων από ένα αρχικό σύνολο δεδομένων. Τα Παραγωγικά Αντιθετικά Δίκτυα εκπαιδεύουν γεννητικά μοντέλα δημιουργώντας ένα «πρόβλημα μάθησης με επίβλεψη» που περιέχει ένα μοντέλο διάκρισης. Το πρώτο είναι προετοιμασμένο να αναπτύξει συμπεράσματα από την είσοδο. Ταυτόχρονα, το δεύτερο προσπαθεί να χαρακτηρίσει τα παραγόμενα συμπεράσματα είτε ως «αληθινά» (από το σύνολο δεδομένων) είτε ως «ψεύτικα» (παραγόμενα από τον αλγόριθμο). Μόλις το μοντέλο διάκρισης επισημάνει τα παραγόμενα συμπεράσματα λανθασμένα περίπου τις μισές φορές, το κύριο μοντέλο παράγει αληθοφανή συμπεράσματα.



Εικόνα 10 Διάγραμμα Λειτουργίας Παραγωγικών Αντιθετικών Δικτύων

- **Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks)**

Ένα επαναλαμβανόμενο Νευρωνικό Δίκτυο (Recurrent Neural Network) είναι ένας τύπος τεχνητού νευρωνικού δικτύου που χρησιμοποιεί διαδοχικά δεδομένα (Sequential Data) ή δεδομένα χρονοσειράς (Time Series Data). Αυτοί οι αλγόριθμοι βαθιάς μάθησης χρησιμοποιούνται συνήθως για τακτικές ή χρονικές λύσεις , όπως η μετάφραση γλωσσών, η επεξεργασία φυσικής γλώσσας (NLP), η αναγνώριση ομιλίας και η απόδοση λεζάντας σε εικόνες που ενσωματώνονται σε δημοφιλείς εφαρμογές όπως η Siri , η φωνητική αναζήτηση και το Google Translate. Όπως τα νευρωσικά δίκτυα τροφοδότησης και τα Νευρωνικά Δίκτυα Συνελίξεων (CNN), τα επαναλαμβανόμενα Νευρωνικά Δίκτυα χρησιμοποιούν δεδομένα εκπαίδευσης για να μάθουν. Διακρίνονται για την «μνήμη» τους, καθώς λαμβάνουν πληροφορίες από προηγούμενες εισόδους για να επηρεάσουν την τρέχουσα είσοδο έξοδο. Ενώ τα παραδοσιακά βαθιά νευρωνικά δίκτυα υποθέτουν ότι οι είσοδοι και οι έξοδοι είναι ανεξάρτητες μεταξύ τους , η έξοδος των επαναλαμβανομένων νευρωνικών δικτύων εξαρτάτε από τα προηγούμενα στοιχεία εντός της ακολουθίας.



Εικόνα 11 Επαναλαμβανόμενα Νευρωνικά Δίκτυα

Κεφάλαιο 2 Python και βιβλιοθήκες

2.1 Γιατί η γλώσσα Python είναι κατάλληλη για την Μηχανική Μάθηση

Η γλώσσα προγραμματισμού Python είναι μια γλώσσα που υποστηρίζει τη δημιουργία ενός ευρέος φάσματος εφαρμογών. Οι προγραμματιστές τη θεωρούν ως μία εξαιρετική επιλογή για έργα Τεχνητής Νοημοσύνης (AI – Artificial Intelligence), Μηχανικής Μάθησης (ML - Machine Learning) Βαθιάς Μάθησης (DL- Deep Learning).

Κάποιοι από τους λόγους που τείνουμε να χρησιμοποιούμε την γλώσσα Python σε εφαρμογές μηχανικής μάθησης ή στην επιστήμη δεδομένων είναι οι παρακάτω:

- **Απλότητα και Συνοχή**
  Οι αλγόριθμοι Τεχνητής Νοημοσύνης και τα μοντέλα Μηχανικής Μάθησης είναι πολύπλοκες τεχνολογίες πρόβλεψης που η γλώσσα Python μπορεί να υλοποιήσει. Και πως είναι εφικτό αυτό; Με τον ξεκάθαρο κώδικα και τις πολλές βιβλιοθήκες ειδικά για την Μηχανική Μάθηση.

- **Ποικιλία Βιβλιοθηκών και Frameworks**
  Υπάρχει μια τεράστια βάση δεδομένων με βιβλιοθήκες και Frameworks που χρησιμοποιεί η Python για σκοπούς Μηχανικής Μάθησης, όπως για παράδειγμα:
  - ✓ Η βιβλιοθήκη **NumPy** χρησιμοποιείται με πίνακες και σε ορισμένα τμήματα της γραμμικής άλγεβρας
  - ✓ Το **Keras** , το οποίο είναι ένα API Βαθιάς Μάθησης που εκτελείται στο TensorFlow για να είναι δυνατός και γρήγορος ο χρόνος που τρέχουμε τα πειράματα μας.
  - ✓ Το **TensorFlow**, μια δωρεάν βιβλιοθήκη ανοικτού κώδικα για Machine Learning και Artificial Intelligence που επικεντρώνεται στην εκπαίδευση και τα Βαθιά Νευρωνικά Δίκτυα (Deep Neural Networks – DNN)
  - ✓ Η βιβλιοθήκη **Matplotlib** που επιτρέπει την δημιουργία οπτικοποιήσεων (στατικών, κινούμενων, διακρατικών) στην Python
  - ✓ Η βιβλιοθήκη **Seaborn** που χρησιμοποιείται για την οπτικοποίηση δεδομένων βασισμένη στη Python.
  - ✓ Η βιβλιοθήκη **PyTorch** που χρησιμεύει στη δημιουργία εφαρμογών στην Υπολογιστική Όραση (Computer Vision) και στην επεξεργασία φυσικής γλώσσας.

- **Ανεξαρτησία Πλατφόρμων**

  Οι λύσεις λογισμικού που αναπτύσσονται με την Python μπορούν να κατασκευαστούν και τρέξουν σε πολλαπλές πλατφόρμες λειτουργικών συστημάτων. Για παράδειγμα, Linux, Windows, Mac, Solaris και άλλες. Αυτό κάνει την Python πολύ πιο βολική και πιο εύκολη στη χρήση όταν μιλάμε για εφαρμογές Μηχανικής Μάθησης.

2.1 TensorFlow για την δημιουργία Νευρωνικών Δικτύων

Το TensorFlow είναι μια πλατφόρμα ανοικτού κώδικα για την δημιουργία εφαρμογών Μηχανικής Μάθησης. Πρόκειται για μια συμβολική μαθηματική βιβλιοθήκη που χρησιμοποιεί ροή δεδομένων και διαφοροποιήσιμο προγραμματισμό για την εκτέλεση διάφορων εργασιών που επικεντρώνονται στην εκπαίδευση και την εξαγωγή συμπερασμάτων σε Βαθιά Νευρωνικά Δίκτυα. Επιτρέπει στους προγραμματιστές να δημιουργήσουν εφαρμογές Μηχανικής Μάθησης χρησιμοποιώντας διάφορα εργαλεία, βιβλιοθήκες και πόρους της κοινότητας. Επί του παρόντος, η πιο διάσημη βιβλιοθήκη Βαθιάς Μάθησης στον κόσμο είναι η **TensorFlow** της Google. Το προϊόν της Google χρησιμοποιεί τη μηχανική μάθηση σε όλα τα προϊόντα της για την βελτίωση της μηχανικής αναζήτησης, της μετάφρασης ακόμα και για τις λεζάντες των εικόνων.

2.1.1 Αρχιτεκτονική του TensorFlow

Η αρχιτεκτονική στο TensorFlow λειτουργεί σε τρία μέρη:

- ✓ Προεπεξεργασία των Δεδομένων
- ✓ Δημιουργία του Μοντέλου
- ✓ Εκπαίδευση και εκτίμηση του μοντέλου

Ονομάζεται TensorFlow επειδή δέχεται δεδομένα εισόδου ως πολυδιάστατους πίνακες, γνωστούς και ως Τανυστής (Tensor). Μπορούμε να κατασκευάσουμε ένα είδος διαγράμματος ροής των πράξεων (που ονομάζεται γράφημα) που θέλουμε να εκτελέσουμε σε αυτή την είσοδο. Η είσοδος εισέρχεται από το ένα άκρο, και στη συνέχεια ρέει μέσα σε αυτό το σύστημα πολλαπλών πράξεων και βγαίνει από το άλλο άκρο ως έξοδος.

Αυτός είναι ο λόγος για τον οποίο ονομάζεται TensorFlow, επειδή ο Τανυστής (Tensor) , μπαίνει μέσα, περνάει μέσα από έναν κατάλογο πράξεων και μετά βγαίνει από την άλλη πλευρά.

2.1.2 Χρήση του TensorFlow

Οι απαιτήσεις υλικού και λογισμικού του TensorFlow μπορούν να ταξινομηθούν σε:

- **Φάση Ανάπτυξης**
  Εδώ γίνεται η εκπαίδευση της λειτουργίας μας. Η εκπαίδευση μπορεί να γίνει συνήθως στο Desktop μας ή το Laptop μας.

- **Φάση Εκτέλεσης ή Φάση Εξαγωγής Συμπερασμάτων**
  Μόλις ολοκληρωθεί η εκπαίδευση, το TensorFlow μπορεί να εκτελεστεί σε πολλές διαφορετικές πλατφόρμες όπως:

  - ✓ Desktop με λειτουργικά προγράμματα Windows, MacOS ή Linux
  - ✓ Στο Cloud σαν Web Service
  - ✓ Σε κινητές συσκευές iOS ή Android
  - ✓ Μπορούμε να το εκπαιδεύσουμε σε πολλαπλά μηχανήματα και στη συνέχεια να το εκτελέσουμε σε διαφορετικό μηχάνημα, μόλις έχουμε το εκπαιδευμένο μοντέλο.

Το μοντέλο μας μπορεί να εκπαιδευτεί και να χρησιμοποιηθεί τόσο σε GPU όσο και σε CPU. Οι GPU σχεδιάστηκαν αρχικά και βιντεοπαιχνίδια αλλά στα τέλη του 2010, οι ερευνητές του Stanford διαπίστωσαν ότι οι GPU ήταν πολύ καλές στο να υλοποιούν πράξεις πινάκων στην άλγεβρα, έτσι ώστε να τις καθιστά πολύ γρήγορες για την εκτέλεση τέτοιου είδους υπολογισμών. Η Βαθιά Μάθηση, βασίζεται σε πολλαπλασιασμούς πινάκων. Το TensorFlow είναι πολύ γρήγορο στον υπολογισμό της πράξης αυτής καθώς είναι γραμμένο σε C++. Παρόλο που η υλοποίηση της βιβλιοθήκης έχει γίνει σε C++, μπορεί να προσπελαστεί και να ελεγχθεί από άλλες γλώσσες προγραμματισμού, κυρίως Python.

Τέλος, ένα σημαντικό χαρακτηριστικό του TensorFlow είναι το TensorBoard. Το TensorBoard επιτρέπει τη γραφική και οπτική παρακολούθηση σε ό,τι κάνει το TensorFlow δίνοντας μας έτσι την δυνατότητα να παρακολουθούμε της εργασίες που πραγματοποιούνται.

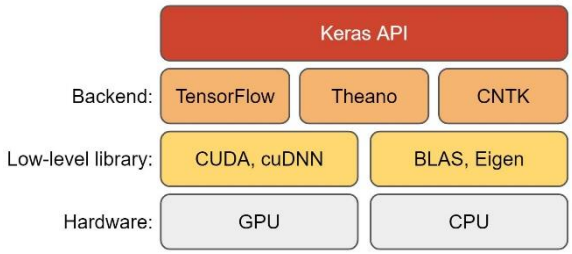
2.2 Τι είναι το Keras

Το Keras είναι ένα υψηλού επιπέδου API Βαθιάς Μάθησης που αναπτύχθηκε από την Google για την υλοποίηση Νευρωνικών Δικτύων. Είναι γραμμένο σε Python και χρησιμοποιείται για να κάνει πιο εύκολη την υλοποίηση Νευρωνικών Δικτύων. Υποστηρίζει επίσης πολλούς Backend υπολογισμούς για τα Νευρωνικά Δίκτυα. Το Keras είναι σχετικά εύκολο στη χρήση και στην εκμάθηση του, καθώς παρέχει ένα Front-End σε Python σε «Υψηλό επίπεδο Αφαίρεσης», ενώ παράλληλα έχει την δυνατότητα να κάνει πολλαπλούς Backend υπολογισμούς. Αυτό καθιστά το Keras πιο αργό από τα άλλα Frameworks που υπάρχουν για την Βαθιά Μάθηση, αλλά εξαιρετικά φιλικό σε αρχάριους προγραμματιστές.

Το Keras μας επιτρέπει να κάνουμε εναλλαγές μεταξύ διάφορων Back-End. Τα Frameworks που υποστηρίζουν το Keras είναι τα παρακάτω:

- TensorFlow
- Theano
- PaidML
- MXNet
- CNTK (Microsoft Cognitive Toolkit)

Το TensorFlow , το οποίο είναι ένα από τα παραπάνω πέντε Frameworks, είναι το μόνο που ενσωματώνει το Keras ως το επίσημο API του υψηλού επιπέδου. Αυτό επιτρέπει την αποτελεσματική Βαθιά Μάθηση, καθώς το Keras διαθέτει προ-κατασκευασμένες ενότητες για διάφορα Νευρωνικά Δίκτυα. Μαζί με αυτό το Core API του TensorFlow μπορεί να χρησιμοποιηθεί για την εκτέλεση υπολογισμών κατά την κατασκευή υπολογιστικών γραφημάτων με τανυστές (Tensors). Αυτό μας παρέχει την ελευθερία και τον έλεγχο να υλοποιήσουμε γρήγορα τις έννοιες και τις ίδιες μας τις ιδέες μας με ακριβή και αποτελεσματικό τρόπο.



Εικόνα 12 Διάγραμμα Keras

2.2.1 Γιατί χρειαζόμαστε το Keras

- Το Keras είναι ένα API που δημιουργήθηκε για να είναι εύκολο στη χρήση και να μπορεί οποιασδήποτε να το μάθει γρήγορα. Προσφέρει σταθερά και απλά API, μειώνει τις ενέργειες που απαιτούνται για την υλοποίηση κοινού κώδικα και εξηγεί με σαφήνεια τα λάθη των χρηστών.

- Ο χρόνος δημιουργίας πρωτοτύπων στο Keras είναι μικρότερος από ότι συνήθως. Αυτό σημαίνει ότι οι εφαρμογές μας μπορούν να υλοποιηθούν σε μικρό χρονικό διάστημα. Επίσης υπάρχει μια ποικιλία επιλογών ανάπτυξης ανάλογα με τις ανάγκες του κάθε χρήστη.

- Γλώσσες υψηλού επιπέδου αφαίρεσης και ενσωματωμένων χαρακτηριστικών είναι αργές και η δημιουργία προσαρμοσμένων χαρακτηριστικών μπορεί να είναι δύσκολη. Το Keras τρέχει πάνω στο TensorFlow και αυτό το κάνει σχετικά γρήγορο. Επίσης είναι ενσωματωμένο σε τέτοιο βάθος ώστε η δημιουργία προσαρμοσμένων ροών εργασίας να δημιουργούνται με ευκολία.

- Η ερευνητική κοινότητά του Keras είναι τεράστια και ιδιαίτερα ανεπτυγμένη. Η τεκμηρίωση και η διαθέσιμη βοήθεια είναι πολύ πιο εκτεταμένη σε σχέση με άλλα Frameworks Βαθιάς Μάθησης.

Εκτός από τα παραπάνω, το Keras κάποια επιπλέον χαρακτηριστικά όπως:

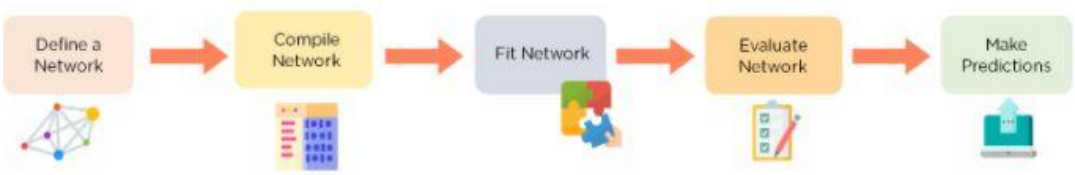
- ✓ Λειτουργεί ομαλά τόσο σε CPU όσο και σε GPU

- ✓ Υποστηρίζει σχεδόν όλα τα μοντέλα Νευρωνικών Δικτύων

- ✓ Έχει παραμετροποιήσιμο χαρακτήρα, γεγονός που το καθιστά ευέλικτο και κατάλληλο για καινοτόμο εύρυνα

2.2.2 Πως υλοποιούμε μοντέλα στο Keras

Παρακάτω θα δούμε ένα διάγραμμα που δείχνει τα βασικά βήματα για να μπορέσουμε να υλοποιήσουμε ένα μοντέλο στο Keras:



Εικόνα 13 Διάγραμμα υλοποίησης μοντέλου στο Keras

- **Ορισμός Δικτύου**
  Σε αυτό το βήμα, ορίζουμε τα διαφορετικά επίπεδα για το μοντέλο μας και τις συνδέσεις μεταξύ τους. Το Keras έχει δύο βασικούς τύπους μοντέλων τα οποία είναι τα Διαδοχικά Μοντέλα και τα Λειτουργικά Μοντέλα. Επιλέγουμε ποιο μοντέλο λειτουργεί καλύτερα για τις ανάγκες μας και στη συνέχεια ορίζουμε την ροή δεδομένων μεταξύ τους

- **Μεταγλώττιση ενός Δικτύου**
  Η διαδικασία προετοιμασίας του κώδικα για να γίνει κατανοητή από την μηχανή μας είναι γνωστή ως Μεταγλώττιση (Compiling). Στο Keras, αυτό επιτυγχάνεται μέσω της χρήσης της μεθόδου model.compile(). Προκειμένου να μεταγλωττίσει το μοντέλο μας, καθορίζουμε την συνάρτηση απώλειας που θα καθορίσει το ποσό των απωλειών που υπάρχουν στο μοντέλο, τον βελτιστοποιητή που θα ελαχιστοποιήσει την απώλεια και τις μετρικές που θα χρησιμοποιηθούν για την αξιολόγηση ακρίβειας του τρόπου μας.

- **Προσαρμογή Δικτύου**
  Μετά την διαδικασία σύνταξης, χρησιμοποιούμε την μέθοδο “Fit” για να εκπαιδεύσουμε το μοντέλο μας στα παρεχόμενα δεδομένα.

- **Αξιολόγηση δικτύου**
  Αφού προσαρμόσουμε το μοντέλο μας, είναι απαραίτητο να εκτιμηθούν τυχόν σφάλματα που υπάρχουν σε αυτό.

- **Προβλέψεις**
  Χρησιμοποιούμε την εντολή **model.predict()** για να κάνουμε προβλέψεις χρησιμοποιώντας το μοντέλο μας σε νέα δεδομένα.

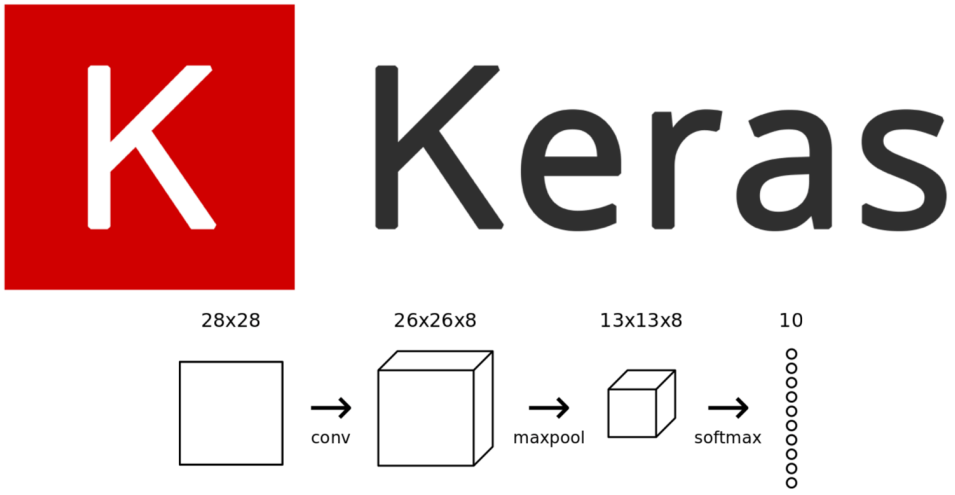
2.2.3 Εφαρμογές του Keras

- ✓ Το Keras χρησιμοποιείται για την δημιουργία «Μοντέλων σε βάθος» που μπορούν να παραχθούν σε smartphone.

- ✓ Χρησιμοποιείται επίσης για κατανεμημένη εκπαίδευση μοντέλων Βαθιάς Μάθησης.

- ✓ Χρησιμοποιείται από μεγάλες εταιρείες όπως το Netflix,Uber κλπ.

- ✓ Χρησιμοποιείται επίσης ευρέως σε διαγωνισμούς Βαθιάς Μάθησης για την δημιουργία και την ανάπτυξη μοντέλων εργασίας, τα οποία γρήγορα σε σύντομο χρονικό διάστημα.



Εικόνα 14 Keras Visual

2.3 Τι είναι το Pandas

Το Pandas είναι ένα πακέτο Python που χρησιμοποιείται ευρέως για την ανάλυση δεομένων, την επιστήμη των δεδομένων και τις λειτουργίες Μηχανικής Μάθησης. Αυτό το πακέτο ανοικτού κώδικα είναι κατασκευασμένο πάνω σε μια άλλη ενότητα που ονομάζεται NumPy και παρέχει τη δυνατότητα εργασίας με πολυδιάστατους πίνακες. Λόγω της ευρείας χρήσης του στην επεξεργασία δεδομένων, το Pandas ενσωματώνεται απρόσκοπτα σε πολλά άλλα πακέτα επιστήμης δεδομένων στο οικοσύστημα της Python και έρχεται προεγκατεστημένο με κάθε διανομή.

2.3.1 Βασικά Χαρακτηριστικά του Pandas

- **Εξαιρετικός χειρισμός δεδομένων**

  Η βιβλιοθήκη Pandas προσφέρει στους χρήστες της τη δυνατότητα να εργάζονται με σειρές και πλαίσια δεδομένων, τα οποία παρέχουν αποδοτικά και γρήγορα μέσα για τον χειρισμό και την διερεύνηση δεδομένων. Αυτές οι δομές μας επιτρέπουν επίσης να αναπαριστούμε αποτελεσματικά τα δεδομένα μας, παρέχοντας μας τη δυνατότητα να τα χειριστούμε με πολλαπλούς τρόπους.

- **Χειρισμός ελλιπών δεδομένων**
  Η αρχική κατάσταση των δεδομένων μπορεί συχνά να είναι περίπλοκη ή δυσνόητη. Ωστόσο, αυτή είναι μόνο η αρχή των προκλήσεων. Τα μη επεξεργασμένα δεδομένα συχνά δημιουργούν προβλήματα, όπως ελλιπείς τιμές και δεδομένα. Η αντιμετώπιση των ελλιπών τιμών είναι ζωτικής σημασίας, καθώς μπορούν να επηρεάσουν το αποτέλεσμα μας. Ευτυχώς, η βιβλιοθήκη Pandas παρέχει ενσωματωμένο μηχανισμό για τον χειρισμό των ελλιπών δεδομένων και την συμπλήρωση τυχόν κενών.

- **Εργαλεία εισαγωγής εξαγωγής**

  Το Pandas προσφέρει μια μεγάλη ποικιλία ενσωματωμένων εργαλείων που μας βοηθούν στην ανάγνωση και την εγγραφή δεδομένων. Όταν προσπαθούμε να κατανοήσουμε τα δεδομένα μας θα πρέπει να τα γράψουμε και σε μία βάση δεδομένων ώστε να έχουμε την δυνατότητα και να τα διαβάζουμε, το Pandas κάνει πλέον αυτή την διαδικασία πολύ απλή για την δική μας ευκολία στο να διαβάζουμε και να εισάγουμε δεδομένα.

- **Καθαρισμός Δεδομένων**

  Η αφθονία των ακατέργαστων δεδομένων μπορεί να καταστεί ιδιαίτερα ένα μεγάλο εμπόδιο στην ακρίβεια της ανάλυσης μας. Αυτό υπογραμμίζει τη σημασία του καθαρισμού των δεδομένων, ο οποίος είναι απαραίτητος για την επίτευξη αξιόπιστων αποτελεσμάτων στην έρευνα και την ανάλυση. Η βιβλιοθήκη Pandas παρέχει εργαλεία για τον καθαρισμό δεδομένων που μας επιτρέπει να λάβουμε τα επιθυμητά αποτελέσματα.

2.3.2 Που χρησιμοποιούμε την βιβλιοθήκη Pandas

Το Pandas είναι ένα ευέλικτο εργαλείο που μπορεί να χειριστεί ένα ευρύ φάσμα εργασιών διαχείρισης δεδομένων. Οι δυνατότητες του είναι τόσο τεράστιες που ίσως είναι πιο πρακτικό να απαριθμήσουμε τους περιορισμούς του παρά τα χαρακτηριστικά του. Ουσιαστικά, το Pandas χρησιμεύει ως κεντρικός κόμβος για όλες τις δραστηριότητες που σχετίζονται με τα δεδομένα, επιτρέποντας μας να εκτελέσουμε με ευκολία μια ποικιλία εργασιών.

Ως παράδειγμα, ας υποθέσουμε ότι έχουμε ένα σύνολο δεδομένων σε ένα αρχείο CSV που θέλουμε να εξερευνήσουμε. Το Pandas, μπορεί να εξάγει τα δεδομένα από το αρχείο CSV και να τα μετατρέψει σε ένα DataFrame, το οποίο είναι ουσιαστικά ένας πίνακας. Αυτό μας επιτρέπει να εκτελέσουμε διάφορες λειτουργίες, όπως:

- Να υπολογίζουμε στατιστικά στοιχεία και να απαντάμε σε ερωτήσεις όπως:
  1. Ποιος είναι ο μέσος όρος, η διάμεσος το μέγιστο ή το ελάχιστο κάθε στήλης;
  2. Ανάλυση συσχετίσεων μεταξύ διαφορετικών στηλών, πχ. Της στήλης Α και της στήλης Β.
  3. Απεικόνιση της κατανομής των δεδομένων στη στήλη Γ.

- Επιπλέον μπορούμε να καθαρίσουμε τα δεδομένα αφαιρώντας τις τιμές που λείπουν και φιλτράροντας γραμμές ή στήλες με βάση συγκεκριμένα κριτήρια.

- Η οπτικοποίηση των δεδομένων είναι εύκολη με τη χρήση του Matplotlib. Μπορούμε να σχεδιάσουμε οτιδήποτε χρειαστούμε για την καλύτερη απεικόνιση τους.

- Αποθηκεύουμε τα καθαρισμένα , μετασχηματισμένα δεδομένα μέσα σε ένα αρχείο CSV, ένα άλλο αρχείου που θέλουμε είτε σε μία βάση δεδομένων για καλύτερη φύλαξη των δεδομένων .

Κεφάλαιο 3 Μοντέλα Νευρωνικών Δικτύων

3.1 Χρήση Βιβλιοθηκών για την υλοποίηση Νευρωνικών Δικτύων

Σε αυτό το κεφάλαιο θα χρησιμοποιήσουμε τις βιβλιοθήκες που έχουμε αναφέρει έως τώρα για να υλοποιήσουμε μοντέλα Νευρωνικών Δικτύων και να κατανοήσουμε την λειτουργία τους σε ένα καλύτερο επίπεδο.

Τα μοντέλα τα οποία θα αναλύσουμε αφορούν δύο κατηγορίες θεμάτων:

- Μοντέλο Στεφανιαίας Νόσου
- Μοντέλο Διαβήτη

Οι βιβλιοθήκες που θα χρησιμοποιήσουμε για την υλοποίηση του μοντέλου μας είναι οι παρακάτω:

- TensorFlow
- NumPy
- Pandas
- IntegerLookup
- Normalization
- StringLookup
- Matplotlib
- Pyplot
- Graphviz

Οι IntegerLookup, Normalization και StringLookup είναι layers εμφωλευμένα μέσα στην βιβλιοθήκη του TensorFlow που μπορούμε να τα χρησιμοποιήσουμε παράλληλα με άλλες βιβλιοθήκες όπως το Pandas.

3.2 Μοντέλο πρόβλεψης Στεφανιαίας Νόσου

Σε αυτή την εφαρμογή θα δούμε πως μπορούμε να κάνουμε ταξινόμηση δομημένων δεδομένων, ξεκινώντας από ένα αρχείο CSV όπου περιέχει τόσο αριθμητικά όσο και κατηγορικά στοιχεία για την υλοποίηση του Νευρωνικού μας Δικτύου. Σε αυτή την περίπτωση θα χρησιμοποιήσουμε στρώματα επεξεργασίας του Keras για να μπορέσουμε να κανονικοποιήσουμε τα αριθμητικά χαρακτηριστικά και να διανυσματοποιήσουμε τα κατηγορικά χαρακτηριστικά.

Το αρχείο το οποίο θα χρησιμοποιήσουμε διαθέτει 300 γραμμές. Κάθε γραμμή περιέχει πληροφορίες για έναν ασθενή(Δείγμα) και κάθε στήλη περιγράφει ένα χαρακτηριστικό του ασθενούς. Θα χρησιμοποιήσουμε τα χαρακτηριστικά αυτά για να προβλέψουμε αν ένας ασθενής έχει Στεφανιαία Νόσο.

Ένα παράδειγμα για κάθε χαρακτηριστικό φαίνεται στο παρακάτω πινακάκι:

Column	Description	Feature Type
Age	Age in years	Numerical
Sex	(1 = male; 0 = female)	Categorical
CP	Chest pain type (0, 1, 2, 3, 4)	Categorical
Trestbpd	Resting blood pressure (in mm Hg on admission)	Numerical
Chol	Serum cholesterol in mg/dl	Numerical
FBS	fasting blood sugar in 120 mg/dl (1 = true; 0 = false)	Categorical
RestECG	Resting electrocardiogram results (0, 1, 2)	Categorical
Thalach	Maximum heart rate achieved	Numerical
Exang	Exercise induced angina (1 = yes; 0 = no)	Categorical
Oldpeak	ST depression induced by exercise relative to rest	Numerical
Slope	Slope of the peak exercise ST segment	Numerical
CA	Number of major vessels (0-3) colored by fluoroscopy	Both numerical & categorical
Thal	3 = normal; 6 = fixed defect; 7 = reversible defect	Categorical
Target	Diagnosis of heart disease (1 = true; 0 = false)	Target

Εικόνα 15 Στοιχεία Πίνακα για εκπαίδευση

Βάση αυτών των δεδομένων που έχουμε θα προχωρήσουμε στην υλοποίηση της ανάλυσης για να δούμε τι ποσοστό επιτυχίας θα έχουμε στις μετρήσεις μας.

3.2.1 Εγκατάσταση Βιβλιοθηκών και Κώδικας

Για να ξεκινήσουμε την εγγραφή του κώδικα χρησιμοποιήσαμε το Anaconda Navigator, ένα framework που μας δίνει την δυνατότητα να γράψουμε τον κώδικα μας με ευκολία και με τις απαραίτητες βιβλιοθήκες που θα χρειαστούμε. Ο κώδικας μας είναι γραμμένος σε Jupiter Notebooks και οι βιβλιοθήκες που θα πρέπει να εισάγουμε για να ξεκινήσουμε είναι οι παρακάτω:

- TensorFlow
- NumPy
- Pandas
- Keras

Τα υπόλοιπα στοιχεία που έχουμε εισάγει, είναι Layers που υπάρχουν είδη στην Βιβλιοθήκη του TensorFlow οπότε απλά τα «καλούμε»

Και η εικόνα που θα έχουμε είναι η εξής:

```
In [13]: import tensorflow as tf
         import numpy as np
         import pandas as pd
         from tensorflow import keras
         from tensorflow.keras import layers
         from tensorflow.keras.layers import IntegerLookup
         from tensorflow.keras.layers import Normalization
         from tensorflow.keras.layers import StringLookup
```

Εικόνα 16 Βιβλιοθήκες Κώδικα

Στη συνέχεια θα πρέπει να εισάγουμε το αρχείο CSV για να μπορέσουμε να αντλήσουμε τις πληροφορίες προκειμένου να ξεκινήσουμε την ανάλυση και να μπορέσουμε να έχουμε μια εικόνα αυτών και θα έχουμε την παρακάτω εικόνα:

```
In [5]: file_url = "http://storage.googleapis.com/download.tensorflow.org/data/heart.csv"
        dataframe = pd.read_csv(file_url)

In [6]: dataframe.shape
Out[6]: (303, 14)

In [45]: dataframe.head()
```

Out[45]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	1	145	233	1	2	150	0	2.3	3	0	fixed	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3	normal	1
2	67	1	4	120	229	0	2	129	1	2.6	2	2	reversible	0
3	37	1	3	130	250	0	0	187	0	3.5	3	0	normal	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0	normal	0

Εικόνα 17 Εισαγωγή CSV και Δεδομένα

Στη τελευταία στήλη με το όνομα Target (στόχος) δείχνει αν ο ασθενείς έχει Στεφανιαία Νόσο με το (1) ή όχι με το (0).

Παρακάτω θα δούμε ένα κομμάτι κώδικα όπου χωρίζει το πλαίσιο δεδομένων εισόδου σε δυο μέρη: Εκπαίδευση και Επικύρωση. Το πλαίσιο δεδομένων χωρίζεται πρώτα σε val_Dataframe και train_Dataframe χρησιμοποιώντας τη μέθοδο Sample, με το 20% των δεδομένων να κατανέμεται στο val_Dataframe και το υπόλοιπο 80% στο train_Dataframe. Το όρισμα random_state ορίζεται σε 1337, το οποίο χρησιμοποιείται ως σπόρος για να εξασφαλιστεί ότι δημιουργείται ο κατάλληλος διαχωρισμός κάθε φορά που εκτελείται ο κώδικας. Αυτό είναι χρήσιμο για την αναπαραγωγή των αποτελεσμάτων. Με την διάσπαση των δεδομένων, ο αριθμός των δειγμάτων σε κάθε σύνολο θα εκτυπώνεται με τη μέθοδο string format. Η συνάρτηση len χρησιμοποιείται για να ληφθεί ο αριθμός των γραμμών σε κάθε πλαίσιο δεδομένων και οι τιμές αυτές εισάγονται στη μέθοδο String Format για να γίνει η τελική παραγωγή.

```
In [9]: val_dataframe = dataframe.sample(frac=0.2, random_state=1337)
        train_dataframe = dataframe.drop(val_dataframe.index)

        print(
            "Using %d samples for training and %d for validation"
            % (len(train_dataframe), len(val_dataframe))
        )

Using 242 samples for training and 61 for validation
```

Εικόνα 18 Διαχωρισμός Δεδομένων

Στη συνέχεια, θα ορίσουμε μια συνάρτηση dataframe_to_dataset όπου δέχεται ένα πλαίσιο δεδομένων Pandas ως όρισμα. Η συνάρτηση αυτή δημιουργεί ένα αντίγραφο πλαισίου δεδομένων και διαχωρίζει τη στήλη Target με ξεχωριστή μεταβλητή (0 ή 1). Η στήλη Target αντιπροσωπεύει την ετικέτα εξόδου που θέλουμε να προβλέψουμε στο μοντέλο που θέλουμε να υλοποιήσουμε. Η συνάρτηση δημιουργεί ένα σύνολο δεδομένων TensorFlow ds τεμαχίζοντας τα χαρακτηριστικά (εξαίροντας την στήλη Target) σε dicts και τη στήλη στόχου σε ξεχωριστό τανυστή. Μετά, το σύνολο δεδομένων ανακατεύεται για να τυχαιοποιηθεί η σειρά των παραδειγμάτων πριν από την επιστροφή του. Τέλος, ο κώδικας δημιουργεί δύο σύνολα δεδομένων από δυο ξεχωριστά πλαίσια τα train_dataframe και val_dataframe χρησιμοποιώντας την συνάρτηση dataframe_to_dataset όπου και αυτά τα σύνολα θα χρησιμοποιηθούν για σκοπούς εκπαίδευσης και επικύρωσης.

```
In [10]: def dataframe_to_dataset(dataframe):
             dataframe = dataframe.copy()
             labels = dataframe.pop("target")
             ds = tf.data.Dataset.from_tensor_slices((dict(dataframe), labels))
             ds = ds.shuffle(buffer_size=len(dataframe))
             return ds


         train_ds = dataframe_to_dataset(train_dataframe)
         val_ds = dataframe_to_dataset(val_dataframe)

In [11]: for x, y in train_ds.take(1):
             print("Input:", x)
             print("Target:", y)

Input: {'age': <tf.Tensor: shape=(), dtype=int64, numpy=29>, 'sex': <tf.Tensor: shape=(), dtype=int64, numpy=1>, 'cp': <tf.Tens
or: shape=(), dtype=int64, numpy=2>, 'trestbps': <tf.Tensor: shape=(), dtype=int64, numpy=130>, 'chol': <tf.Tensor: shape=(), d
type=int64, numpy=204>, 'fbs': <tf.Tensor: shape=(), dtype=int64, numpy=0>, 'restecg': <tf.Tensor: shape=(), dtype=int64, numpy
=2>, 'thalach': <tf.Tensor: shape=(), dtype=int64, numpy=202>, 'exang': <tf.Tensor: shape=(), dtype=int64, numpy=0>, 'oldpeak':
<tf.Tensor: shape=(), dtype=float64, numpy=0.0>, 'slope': <tf.Tensor: shape=(), dtype=int64, numpy=1>, 'ca': <tf.Tensor: shape=
(), dtype=int64, numpy=0>, 'thal': <tf.Tensor: shape=(), dtype=string, numpy=b'normal'>}
Target: tf.Tensor(0, shape=(), dtype=int64)
```

Εικόνα 19 Διαχωρισμός Δεδομένων

3.2.2 Προεπεξεργασία χαρακτηριστικών με Keras Layers

Τα ακόλουθα χαρακτηριστικά που θα χρησιμοποιήσουμε είναι κατηγορικά χαρακτηριστικά κωδικοποιημένα ως ακέραιοι αριθμοί:

- Sex
- CP
- FBS
- Restecg
- Exang
- Ca

Θα κωδικοποιήσουμε τα παρακάτω χαρακτηριστικά χρησιμοποιώντας one-hot encoding ή αλλιώς κωδικοποίηση ενός σημείου. Έχουμε δυο επιλογές που μπορούμε να χρησιμοποιήσουμε:

1. Να χρησιμοποιήσουμε την CategoryEncoding(), η οποία απαιτεί γνώση του εύρους των τιμών εισόδου και θα παρουσιάσει σφάλμα σε είσοδο εκτός εύρους.
2. Να χρησιμοποιήσουμε την IntegerLookup(), η οποία θα δημιουργήσει έναν πίνακα αναζήτησης για εισόδους και θα κρατήσει έναν δείκτη εξόδου για τις άγνωστες τιμές εισόδου.

Στην δική μας περίπτωση θα χρησιμοποιήσουμε την IntegerLookup() καθώς θέλουμε μια απλή λύση που θα χειρίζεται εισόδους εκτός εύρους κατά την εξαγωγή συμπερασμάτων. Έχουμε επίσης ένα κατηγορικό χαρακτηριστικό κωδικοποιημένο ως συμβολοσειρά: thal. Οπότε θα δημιουργήσουμε ένα ευρετήριο όλων των πιθανών χαρακτηριστικών και θα κωδικοποιήσουμε την έξοδο χρησιμοποιώντας το επίπεδο StringLookup().

In [14]:
```
def encode_numerical_feature(feature, name, dataset):
    # Create a Normalization Layer for our feature
    normalizer = Normalization()

    # Prepare a Dataset that only yields our feature
    feature_ds = dataset.map(lambda x, y: x[name])
    feature_ds = feature_ds.map(lambda x: tf.expand_dims(x, -1))

    # Learn the statistics of the data
    normalizer.adapt(feature_ds)

    # Normalize the input feature
    encoded_feature = normalizer(feature)
    return encoded_feature


def encode_categorical_feature(feature, name, dataset, is_string):
    lookup_class = StringLookup if is_string else IntegerLookup
    # Create a lookup layer which will turn strings into integer indices
    lookup = lookup_class(output_mode="binary")

    # Prepare a Dataset that only yields our feature
    feature_ds = dataset.map(lambda x, y: x[name])
    feature_ds = feature_ds.map(lambda x: tf.expand_dims(x, -1))

    # Learn the set of possible string values and assign them a fixed integer index
    lookup.adapt(feature_ds)

    # Turn the string input into integer indices
    encoded_feature = lookup(feature)
    return encoded_feature
```

Εικόνα 20 Κωδικοποίηση Χαρακτηριστικών

Τέλος, τα παρακάτω δεδομένα είναι συνεχή αριθμητικά χαρακτηριστικά :

- Age
- Trestbps
- Chol
- Thalach
- Oldpeak
- Slope

Για κάθε ένα από αυτά τα χαρακτηριστικά, θα χρησιμοποιήσουμε ένα επίπεδο Normalization() για να βεβαιωθούμε ότι ο μέσος όρος κάθε χαρακτηριστικού είναι 0 και η τυπική του απόκλιση του είναι 1.

Παρακάτω ορίζουμε 3 συναρτήσεις χρησιμότητας για να κάνουμε τις λειτουργίες:

- Encode_numerical_feature: για την εφαρμογή της κανονικοποίησης ανά χαρακτηριστικό στα αριθμητικά χαρακτηριστικά.
- Encode_string_categorial_feature: για να μετατρέψει πρώτα τις εισόδους συμβολοσειρών σε ακέραιους δέκτες, και στη συνέχεια να κωδικοποιήσουμε με μια κίνηση αυτούς τους δέκτες.
- Encode_integer_categorial_feature: για την κωδικοποίηση κατηγορικών χαρακτηριστικών ακέραιων αριθμών.

Παρακάτω θα δούμε τον κώδικά που δημιουργεί το μοντέλο μας μέσω του Keras. Ξεκινάμε με τον ορισμό αρκετών τανυστών εισόδου, για κάθε ένα χαρακτηριστικό από τα δεδομένα που έχουμε. Τα χαρακτηριστικά ορίζονται σε δυο κατηγορίες: Κατηγορικά δεδομένα ως ακέραιοι αριθμοί (sex, cp, fbs, restecg, exang, ca) και ένα κατηγορικό χαρακτηριστικό κωδικοποιημένο ως συμβολοσειρά (Thal) και αριθμητικά χαρακτηριστικά (age, tresbps, Chol, Oldpeak, slope). Εν συνεχεία, καλούμε το Encode_categorial_feature για την επεξεργασία των κατηγορηματικών χαρακτηριστικών. Η συγκεκριμένη συνάρτηση λαμβάνει έναν τανυστή εισόδου, το όνομα του χαρακτηριστικού, το σύνολο το δεδομένων εκπαίδευσης και μια Boolean που δείχνει αν το χαρακτηριστικό είναι τύπου συμβολοσειράς (String type) ή όχι και επιστρέφει έναν κωδικοποιημένο τανυστή.

Για τα αριθμητικά στοιχεία, ο κώδικας καλεί την encode_numerical_feature για την επεξεργασία τους. Η συγκεκριμένη συνάρτηση λαμβάνει έναν τανυστή εισόδου, το όνομα του χαρακτηριστικού και το σύνολο δεδομένων εκπαίδευσης και επιστρέφει έναν τανυστή που αναπαριστά το επεξεργασμένο αριθμητικό χαρακτηριστικό.

Τέλος, συνενώνουμε όλους τους κωδικοποιημένους τανυστές και εφαρμόζουμε δυο Dense Layers στους συνενωμένους τανυστές. Το πρώτο Dense Layer έχει 32 μονάδες και χρησιμοποιεί την συνάρτηση ενεργοποίησης “ReLU” και το δεύτερο Dense Layer έχει μία μονάδα και χρησιμοποιεί την συνάρτηση ενεργοποίησης sigmoid. Η τελική έξοδος του μοντέλου είναι ένας δυαδικός ταξινομητής, καθώς η σιγμοειδής συνάρτηση επιστρέφει μια τιμή μεταξύ 0 και 1, η οποία μπορεί να ερμηνευτεί ως πιθανότητα δυαδικής ταξινόμησης. Να αναφέρουμε ότι το μοντέλο κάνει compile χρησιμοποιώντας τα: Adam optimizer, binary cross-entropy loss και την ακρίβεια για μέτρηση.

Adam Optimizer:

Ο Adam Optimizer είναι ένας αλγόριθμος βελτιστοποίησης για την στοχαστική κάθοδο κλίσης (SGD ή Stochastic Gradient Descent), η οποία είναι μια δημοφιλής μέθοδος που χρησιμοποιείται για την εκπαίδευση μοντέλων Βαθιάς Μάθησης. Τα αρχικά Adam σημαίνουν Adaptive Moment Estimation και συνδυάζει τα πλεονεκτήματα άλλων δυο δημοφιλών αλγόριθμων βελτιστοποίησης των:

- ✓ RMSPop
- ✓ Adagrad

Παρακολουθώντας τόσο την πρώτη όσο και την δεύτερη στιγμή κλίσεων.

Με απλά λόγια, ο Adam βελτιστοποιεί τον ρυθμό μάθησης του μοντέλου υπολογίζοντας τον εκθετικό κινητικό μέσο όρο των προηγούμενων τετραγωνικών τιμών την κλίσης και στη συνέχεια χρησιμοποιώντας τον για την ενημέρωση του ρυθμού μάθησης. Αυτό συμβάλει στην αποτροπή του ρυθμού μάθησης από το να γίνει πολύ μικρός ή πολύ μεγάλος και την διάρκεια της εκπαίδευσης, οδηγώντας σε ταχύτερη και σταθερότερη σύγκλιση. Ο Adam χρησιμοποιεί επίσης την ορμή, η οποία συμβάλει στην επιτάχυνση της διαδικασίας βελτιστοποίησης προσθέτοντας ένα κλάσμα της προηγούμενης κλίσης στην τρέχουσα κλίση. Αυτό βοηθά στην εξομάλυνση της διαδικασίας και την αποφυγή της εμπλοκής σε τοπικά ελάχιστα.

Binary Crossentropy:

Η Binary Crossentropy είναι μια συνάρτηση απώλειας που χρησιμοποιείται συνήθως σε προβλήματα δυαδικής ταξινόμησης. Μετρά την διαφορά μεταξύ της προβλεπόμενης κατανομής πιθανότητας και της πραγματικής κατανομής πιθανότητας της μεταβλητής στόχου (Target), η οποία λαμβάνει δυαδική τιμή (0 ή 1). Διαισθητικά, αυτή η συνάρτηση απωλειών ανταμείβει το μοντέλο όταν προβλέπει σωστά την πραγματική ετικέτα και το «τιμωρεί» όταν κάνει λανθασμένες προβλέψεις. Χρησιμοποιείται συνήθως σε προβλήματα δυαδικής ταξινόμησης επειδή είναι εύκολο να βελτιστοποιηθεί και είναι κατάλληλη για προβλήματα με δυο πιθανά αποτελέσματα.

Κατά την διάρκεια της εκπαίδευσης, ο στόχος του μοντέλου είναι να ελαχιστοποιήσει την απώλεια δυαδικής διασταυρούμενης εντροπίας (Binary Crossentropy loss) πράγμα που σημαίνει ότι προσπαθεί να κάνει τις προβλέψεις του να ταιριάζουν όσο το δυνατόν περισσότερο με τις πραγματικές ετικέτες. Αφού το μοντέλο εκπαιδευτεί, μπορεί να χρησιμοποιηθεί για να κάνει προβλέψεις σε νέα δεδομένα χρησιμοποιώντας το τελικό σύνολο βαρών που μαθεύτηκε κατά την διάρκεια της εκπαίδευσης.

Sigmoid Activation Function:

Η Σιγμοειδής Συνάρτηση Ενεργοποίηση ή Sigmoid Activation Function είναι μια ευρέος χρησιμοποιημένη μη γραμμική συνάρτηση ενεργοποίησης στα Νευρωνικά Δίκτυα.

Η Σιγμοειδής Συνάρτηση απεικονίζει οποιονδήποτε αριθμό πραγματικής αξίας σε μια τιμή μεταξύ 0 και 1. Καθώς η είσοδος στη συνάρτηση αυξάνεται, η έξοδος πλησιάζει στο 1, και καθώς η είσοδος μειώνεται , η έξοδος πλησιάζει το 0. Αυτή τη ιδιότητα την καθιστά χρήσιμη για προβλήματα δυαδικής ταξινόμησης που ο στόχος είναι να προβλεφθεί μια δυαδική έξοδος (0 ή 1) με βάση το σύνολο χαρακτηριστικών εισόδου.

Η συνάρτηση αυτή, είναι επίσης διαφορίσιμη, πράγμα που σημαίνει ότι μπορεί να χρησιμοποιηθεί στον αλγόριθμο Backpropagation για την ενημέρωση των βαρών ενός Νευρωνικού Δικτύου κατά την διάρκεια της εκπαίδευσης. Ωστόσο, μπορεί να υποφέρει από το πρόβλημα της εξαφάνισης των κλίσεων, το οποίο μπορεί να δυσχεράνει την σύγκλιση των Νευρωνικών Δικτύων κατά την διάρκεια της εκπαίδευσης.

Λόγω αυτών των περιορισμών, άλλες συναρτήσεις ενεργοποίησης όπως η ReLU και οι παραλλαγές της, χρησιμοποιούνται συνήθως στα σύγχρονα Νευρωνικά Δίκτυα. Σε ορισμένες περιπτώσεις η Σιγμοειδής Συνάρτηση εξακολουθεί να είναι χρήσιμη, όπως το στρώμα εξόδου ενός προβλήματος δυαδικής ταξινόμησης ή ως συνάρτηση πύλης σε Αναδρομικά Νευρωνικά Δίκτυα (RNN ή Recurrent Neural Networks).

ReLU (Rectified Linear Unit):

Η ReLU είναι μια δημοφιλής μη γραμμική συνάρτηση ενεργοποίησης που χρησιμοποιείται στα Νευρωνικά Δίκτυα.

Η συνάρτηση ReLU είναι μια απλή γραμμική συνάρτηση που απεικονίζει κάθε αρνητική είσοδο στο 0 και κάθε θετική είσοδο στον εαυτό της. Αυτό την καθιστά υπολογιστικά αποδοτική για την αξιολόγηση της και επιτρέπει την ταχύτερη εκπαίδευση των Νευρωνικών Δικτύων.

Ένα από τα βασικά πλεονεκτήματα της συνάρτησης ReLU είναι ότι αντιμετωπίζει το πρόβλημα των εξασφαλισμένων κλίσεων, το οποίο μπορεί να εμφανιστεί σε Νευρωνικά Δίκτυα με Σιγμοειδή ή Tanh συναρτήσεις ενεργοποίησης καθώς επιτρέπει στις κλίσεις να ρέουν εύκολα μέσα στο δίκτυο, καθιστώντας ευκολότερη την εκπαίδευση τους. Ένα άλλο πλεονέκτημα της συνάρτησης αυτής, είναι ότι εισάγει αραιότητα στο δίκτυο. Δεδομένου ότι κάθε αρνητική είσοδος αντιστοιχίζεται στο 0, οι νευρώνες ReLU μπορούν να γίνουν ανενεργοί ή και «νεκροί» εάν η είσοδος στον νευρώνα είναι σταθερά αρνητική. Αυτό μπορεί να βοηθήσει στη μείωση της χωρητικότητάς του δικτύου και στην αποφυγή της υπερπροσαρμογής.

Παρά τα πολλά πλεονεκτήματα της, η συνάρτηση ReLU έχει και τους περιορισμούς της. Ένα πιθανό ζήτημα είναι το πρόβλημα των Νευρώνων “Dying ReLU”, το οποίο συμβαίνει όταν η είσοδος σε ένα νευρώνα είναι σταθερά αρνητική και προκαλεί τη μόνιμη αδράνεια του νευρώνα. Αυτό μπορεί να αντιμετωπιστεί με τη χρήση παραλλαγών της συνάρτησης ReLU, όπως η “Leaky ReLU” ή “Parametric ReLU”, η οποίες επιτρέπουν μια μικρή ποσότητα αρνητικής εξόδου. Ένας άλλος περιορισμός είναι ότι η συναρτήσεις ReLU δεν είναι συμμετρικές γύρω από το 0, γεγονός που μπορεί να οδηγήσει σε ασυμμετρία κλίσης σε ορισμένους τύπους δικτύων. Συνολικά, η συνάρτηση ReLU είναι μια δημοφιλής και αποτελεσματική συνάρτηση ενεργοποίησης για Νευρωνικά Δίκτυα, ιδίως σε εφαρμογές βαθιάς μάθησης.

Accuracy Metrics:

Στη Μηχανική Μάθηση, μια μετρική είναι ένα μέτρο που χρησιμοποιείται για την αξιολόγηση της απόδοσης ενός μοντέλου. Οι μετρικές παρέχουν ένα τρόπο ποσοτικοποίησης της απόδοσης ενός μοντέλου σε μια δεδομένη εργασία.

Σε αυτή την περίπτωση, η μετρική που χρησιμοποιείται είναι η ακρίβεια. Η ακρίβεια είναι μια κοινή μετρική αξιολόγηση που χρησιμοποιείται σε προβλήματα ταξινόμησης και μετρά το ποσοστό των σωστά ταξινομημένων δειγμάτων επί του συνόλου των δειγμάτων. Με άλλα λόγια, η ακρίβεια είναι ο λόγος του αριθμού των σωστών προβλέψεων προς τον συνολικό αριθμό προβλέψεων που έκανε το μοντέλο μας.

Για παράδειγμα, αν έχουμε ένα πρόβλημα δυαδικής ταξινόμησης όπου υπάρχουν 100 δείγματα στο σύνολο δοκιμής και το μοντέλο κάνει 80 σωστές προβλέψεις, τότε η ακρίβεια του μοντέλου θα είναι 80/100 ή 80%. Ένα υψηλότερο σκορ ακρίβειας υποδηλώνει καλύτερη απόδοση, ενώ τα χαμηλότερο σκορ ακρίβειας υποδηλώνει χειρότερη απόδοση.

Είναι σημαντικό να σημειωθεί ότι ενώ η ακρίβεια είναι μια ευρέως χρησιμοποιημένη μετρική αξιολόγησης, μπορεί να μην είναι πάντα η καταλληλότερη μετρική που πρέπει να χρησιμοποιηθεί. Σε ορισμένες περιπτώσεις, άλλες μετρικές όπως η ακρίβεια η ανάκληση ή το F1-Score μπορεί να είναι πιο κατάλληλες ανάλογα με τη φύση του προβλήματος ή το επιθυμητό αποτέλεσμα.

Οι παραπάνω συναρτήσεις χρησιμοποιήθηκαν όπως θα δούμε παρακάτω στην εικόνα 20 για να μπορέσουμε να έχουμε μια ομαλή και σωστή παραγωγή αποτελεσμάτων.

In [15]:
```
# Categorical features encoded as integers
sex = keras.Input(shape=(1,), name="sex", dtype="int64")
cp = keras.Input(shape=(1,), name="cp", dtype="int64")
fbs = keras.Input(shape=(1,), name="fbs", dtype="int64")
restecg = keras.Input(shape=(1,), name="restecg", dtype="int64")
exang = keras.Input(shape=(1,), name="exang", dtype="int64")
ca = keras.Input(shape=(1,), name="ca", dtype="int64")

# Categorical feature encoded as string
thal = keras.Input(shape=(1,), name="thal", dtype="string")

# Numerical features
age = keras.Input(shape=(1,), name="age")
trestbps = keras.Input(shape=(1,), name="trestbps")
chol = keras.Input(shape=(1,), name="chol")
thalach = keras.Input(shape=(1,), name="thalach")
oldpeak = keras.Input(shape=(1,), name="oldpeak")
slope = keras.Input(shape=(1,), name="slope")

all_inputs = [
    sex,
    cp,
    fbs,
    restecg,
    exang,
    ca,
    thal,
    age,
    trestbps,
    chol,
    thalach,
    oldpeak,
    slope,
]

# Integer categorical features
sex_encoded = encode_categorical_feature(sex, "sex", train_ds, False)
cp_encoded = encode_categorical_feature(cp, "cp", train_ds, False)
fbs_encoded = encode_categorical_feature(fbs, "fbs", train_ds, False)
restecg_encoded = encode_categorical_feature(restecg, "restecg", train_ds, False)
exang_encoded = encode_categorical_feature(exang, "exang", train_ds, False)
ca_encoded = encode_categorical_feature(ca, "ca", train_ds, False)

# String categorical features
thal_encoded = encode_categorical_feature(thal, "thal", train_ds, True)

# Numerical features
age_encoded = encode_numerical_feature(age, "age", train_ds)
trestbps_encoded = encode_numerical_feature(trestbps, "trestbps", train_ds)
chol_encoded = encode_numerical_feature(chol, "chol", train_ds)
thalach_encoded = encode_numerical_feature(thalach, "thalach", train_ds)
oldpeak_encoded = encode_numerical_feature(oldpeak, "oldpeak", train_ds)
slope_encoded = encode_numerical_feature(slope, "slope", train_ds)

all_features = layers.concatenate(
    [
        sex_encoded,
        cp_encoded,
        fbs_encoded,
        restecg_encoded,
        exang_encoded,
        slope_encoded,
        ca_encoded,
        thal_encoded,
        age_encoded,
        trestbps_encoded,
        chol_encoded,
        thalach_encoded,
        oldpeak_encoded,
    ]
)
x = layers.Dense(32, activation="relu")(all_features)
x = layers.Dropout(0.5)(x)
output = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(all_inputs, output)
model.compile("adam", "binary_crossentropy", metrics=["accuracy"])
```

Εικόνα 21 Μοντέλο και Χαρακτηριστικά

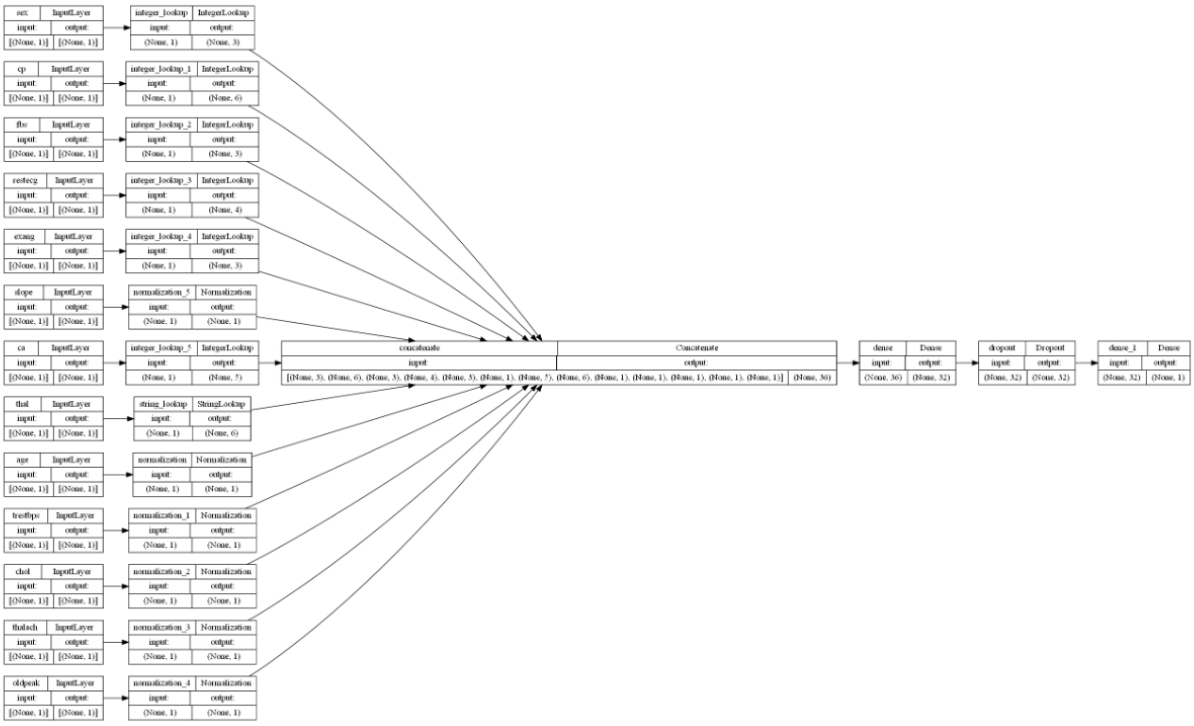
Πριν ξεκινήσουμε την διαδικασία εκμάθησης του Νευρωνικού Δικτύου μας, μπροούμε να να κάνουμε ένα γράφημα με τις τιμές και τις τιμές εισόδου εξόδου όπου έχουμε ορίσει τρέχοντας την παρακάτω συνάρτηση:

```
In [11]: # `rankdir='LR'` is to make the graph horizontal.
         keras.utils.plot_model(model, show_shapes=True, rankdir="LR")
```

Εικόνα 22 Συνάρτηση Διαγράμματος

Και θα έχουμε το παρακάτω σχήμα:

Out[11]:



Εικόνα 23 Διάγραμμα Τιμών

Έπειτα για να ξεκινήσουμε την διαδικασία για την εκμάθηση του μοντέλου μας χρησιμοποιούμε την παρακάτω συνάρτηση και τρέχουμε για δέκα φορές:

```
In [15]: model.fit(train_ds, epochs=10, validation_data=val_ds)

         Epoch 1/10
         8/8 [==============================] - 0s 15ms/step - loss: 0.2290 - accuracy: 0.8802 - val_loss: 0.3
         954 - val_accuracy: 0.8033
         Epoch 2/10
         8/8 [==============================] - 0s 14ms/step - loss: 0.2208 - accuracy: 0.9050 - val_loss: 0.3
         959 - val_accuracy: 0.8033
         Epoch 3/10
         8/8 [==============================] - 0s 14ms/step - loss: 0.2418 - accuracy: 0.8926 - val_loss: 0.3
         964 - val_accuracy: 0.8033
         Epoch 4/10
         8/8 [==============================] - 0s 14ms/step - loss: 0.2195 - accuracy: 0.9132 - val_loss: 0.3
         973 - val_accuracy: 0.8033
         Epoch 5/10
         8/8 [==============================] - 0s 13ms/step - loss: 0.2409 - accuracy: 0.8843 - val_loss: 0.3
         987 - val_accuracy: 0.8033
         Epoch 6/10
         8/8 [==============================] - 0s 15ms/step - loss: 0.2425 - accuracy: 0.9008 - val_loss: 0.3
         994 - val_accuracy: 0.8033
         Epoch 7/10
         8/8 [==============================] - 0s 14ms/step - loss: 0.2502 - accuracy: 0.8884 - val_loss: 0.3
         994 - val_accuracy: 0.8033
         Epoch 8/10
         8/8 [==============================] - 0s 11ms/step - loss: 0.2442 - accuracy: 0.9050 - val_loss: 0.3
         996 - val_accuracy: 0.8033
         Epoch 9/10
         8/8 [==============================] - 0s 16ms/step - loss: 0.2242 - accuracy: 0.9132 - val_loss: 0.3
         989 - val_accuracy: 0.8033
         Epoch 10/10
         8/8 [==============================] - 0s 13ms/step - loss: 0.2478 - accuracy: 0.9008 - val_loss: 0.4
         001 - val_accuracy: 0.8033

Out[15]: <keras.callbacks.History at 0x1da83958c10>
```

Εικόνα 24 Εκμάθηση για 10 Epochs

Βλέπουμε ότι έχουμε ποσοστό επιτυχίας 90%. Θα τρέξουμε την ίδια διαδικασία για είκοσι και πενήντα φορές αντίστοιχα για να δούμε τι απόκλιση θα έχουμε στα αποτελέσματα μας.

- Παρατηρούμε ότι για είκοσι φορές έχουμε 92% επιτυχία.

```
Epoch 20/20
8/8 [==============================] - 0s 15ms/step - loss: 0.1746 - accuracy: 0.9298
466 - val_accuracy: 0.7705

Out[18]: <keras.callbacks.History at 0x1da85dd5af0>
```

Εικόνα 25 Εκμάθηση για 20 Epochs

- Παρατηρούμε ότι για πενήντα φορές έχουμε 91% επιτυχία.

```
Epoch 50/50
8/8 [==============================] - 0s 16ms/step - loss: 0.2106 - accuracy: 0.9132
4320 - val_accuracy: 0.7705

Out[17]: <keras.callbacks.History at 0x1da85dd6fd0>
```

Εικόνα 26 Εκμάθηση για 50 Epochs

Παρατηρούμε λοιπόν ότι οι τιμές που λαμβάνουμε για κάθε μεταβολή είναι αμυδρή και σε μικρά ποσοστά πράγμα που δείχνει ότι το μοντέλο μας λειτουργεί σωστά με τα δεδομένα εκπαίδευσης που του έχουμε ορίσει.

3.3 Μοντέλο πρόβλεψης Διαβήτη

Για να μπορέσουμε να υλοποιήσουμε το παρακάτω μοντέλο θα αξιοποιήσουμε ένα διαθέσιμο σύνολο δεδομένων το όποιο θα μας δώσει την δυνατότητα να κάνουμε τις απαραίτητες προβλέψεις και ενέργειες. Παρακάτω θα δούμε τι βιβλιοθήκες που θα πρέπει να έχουμε ώστε να ξεκινήσουμε την συγγραφή του κώδικα μας

```
In [13]: import pandas as pd
         import numpy as np
         import tensorflow as tf
         from tensorflow.keras.layers import Input, Dense
         from tensorflow.keras.models import Sequential
         from sklearn.model_selection import train_test_split
         import matplotlib.pyplot as plt
         %matplotlib inline
```

Εικόνα 27 Βιβλιοθήκες 2ου Μοντέλου

Είμαστε τώρα έτοιμοι να ξεκινήσουμε την εισαγωγή του συνόλου δεδομένων. Στο επόμενο κομμάτι κώδικα, θα εισάγουμε τα δεδομένα μας και χρησιμοποιούμε τη μέθοδο head() για να μας φέρει τα πρώτα πέντε κορυφαία σημεία δεδομένων και θα έχουμε την παρακάτω εικόνα:

```
In [14]: data = pd.read_csv('diabetes.csv')
         data.head()
```

Out[14]:

	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure	Triceps skin fold thickness	2-Hour serum insulin	Body mass index	Diabetes pedigree function	Age	Class variable
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Εικόνα 28 Εισαγωγή Δεδομένων από το CSV

3.3.1 Ανάλυση Μοντέλου και κώδικας

Στη συνέχεια θα χρησιμοποιήσουμε την εντολή data.isna().sum όπου θα μας επιστρέψει τις στήλες στο πλαίσιο δεδομένων Pandas που έχουμε ορίσει μαζί με τον αριθμό των ελλιπών τιμών που εντοπίστηκαν σε κάθε μία από αυτές, οπότε το 0 σημαίνει ότι δεν υπάρχουν ελλιπείς τιμές και το 1 σημαίνει ότι υπάρχει μόνο μια ελλιπής τιμή που στην περίπτωση μας δεν υπάρχουν.

```
In [15]: data.isna().sum()

Out[15]: Number of times pregnant          0
         Plasma glucose concentration      0
         Diastolic blood pressure          0
         Triceps skin fold thickness       0
         2-Hour serum insulin              0
         Body mass index                   0
         Diabetes pedigree function        0
         Age                               0
         Class variable                    0
         dtype: int64
```

Εικόνα 29 Μέτρηση Ελλιπών Τιμών

Έπειτα θα τρέξουμε την εντολή data.dtypes για να ταξινομήσουμε και να δούμε τι είδους τιμές έχουν οι μεταβλητές μας καθώς και τι σχεσιακές ή λογικές πράξεις μπορούν να εφαρμοστούν χωρίς να υπάρξει κάποιο σφάλμα.

```
In [16]: data.dtypes

Out[16]: Number of times pregnant          int64
         Plasma glucose concentration      int64
         Diastolic blood pressure          int64
         Triceps skin fold thickness       int64
         2-Hour serum insulin              int64
         Body mass index                 float64
         Diabetes pedigree function      float64
         Age                               int64
         Class variable                    int64
         dtype: object
```

Εικόνα 30 Είδος Τιμών

Εν συνεχεία, θα χρησιμοποιήσουμε την συνάρτηση iloc() όπου μας επιτρέπει να επιλέξουμε ένα συγκεκριμένο κελί του συνόλου δεδομένων, δηλαδή μας βοηθάει να επιλέξουμε μια τιμή που ανήκει σε μια συγκεκριμένη γραμμή ή στήλη. Στη δική μας περίπτωση θέλουμε να πιάσουμε όλες τι στήλες και γραμμές από τα δεδομένα μας, οπότε θα έχουμε την εξής διαδικασία:

```
In [17]: x = data.iloc[:,:-1]
         y = data.iloc[:,-1]
```

Εικόνα 31 Επιλογή Κελιών

Μετά από αυτό είμαστε έτοιμοι να δημιουργήσουμε το μοντέλο μας για να ξεκινήσουμε την διαδικασία εκπαίδευσης του μοντέλου μας και να ορίσουμε μια συνάρτηση accuracies ώστε να μπορέσουμε να προβλέψουμε τα αποτελέσματα μας. Θα δημιουργήσουμε μια For Loop όπου θα μπορέσουμε να τρέξουμε 100 epochs για 10 φορές και να δούμε τι αποτελέσματα θα έχουμε.

```
In [22]: accuracies = []
         for i in range (10):
             tf.random.set_seed(i)
             model=Sequential()
             model.add(Dense(1, input_dim=8, activation='sigmoid'))
             model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
             model.fit(x_train, y_train, epochs=100, batch_size=100, verbose=0)
             _,accuracy = model.evaluate(x_test,y_test)
             accuracies.append(accuracy*100)

6/6 [==============================] - 0s 2ms/step - loss: 1.2816 - accuracy: 0.6354
6/6 [==============================] - 0s 1ms/step - loss: 4.4909 - accuracy: 0.5469
6/6 [==============================] - 0s 1ms/step - loss: 6.3778 - accuracy: 0.6510
6/6 [==============================] - 0s 3ms/step - loss: 4.3753 - accuracy: 0.5885
6/6 [==============================] - 0s 2ms/step - loss: 3.0081 - accuracy: 0.4792
6/6 [==============================] - 0s 3ms/step - loss: 3.9440 - accuracy: 0.6094
6/6 [==============================] - 0s 1ms/step - loss: 5.7711 - accuracy: 0.4583
6/6 [==============================] - 0s 1ms/step - loss: 5.9632 - accuracy: 0.4896
6/6 [==============================] - 0s 3ms/step - loss: 6.5113 - accuracy: 0.6458
6/6 [==============================] - 0s 2ms/step - loss: 6.7810 - accuracy: 0.4792
```

Εικόνα 32 Μοντέλο και εκπαίδευση για 100 Epochs και 10 Επαναλήψεις

Εφόσον ολοκληρώσουμε την διαδικασία για 10 φορές θα τρέξουμε την εντολή sum(accuracies)/len(accuracies) να πάρουμε το ποσοστό της.

```
In [23]: sum(accuracies)/len(accuracies)

Out[23]: 55.83333373069763
```

Εικόνα 33 Εντολή Εύρεσης Μέσης Τιμής Αποτελεσμάτων 1ου Αποτελέσματος

✓ Παρατηρούμε ότι έχουμε ποσοστό επιτυχίας 55.83%

- Για 20 φορές με τιμή epochs=150 θα έχουμε τα εξής αποτελέσματα :

```
In [24]: accuracies = []
         for i in range (20):
             tf.random.set_seed(i)
             model=Sequential()
             model.add(Dense(1, input_dim=8, activation='sigmoid'))
             model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
             model.fit(x_train, y_train, epochs=150, batch_size=100, verbose=0)
             _,accuracy = model.evaluate(x_test,y_test)
             accuracies.append(accuracy*100)

6/6 [==============================] - 0s 2ms/step - loss: 2.9396 - accuracy: 0.4010
6/6 [==============================] - 0s 4ms/step - loss: 3.5608 - accuracy: 0.6406
6/6 [==============================] - 0s 4ms/step - loss: 1.3915 - accuracy: 0.5990
6/6 [==============================] - 0s 4ms/step - loss: 0.8952 - accuracy: 0.6250
6/6 [==============================] - 0s 2ms/step - loss: 3.2256 - accuracy: 0.6562
6/6 [==============================] - 0s 1ms/step - loss: 2.9461 - accuracy: 0.4635
6/6 [==============================] - 0s 2ms/step - loss: 5.8160 - accuracy: 0.4688
6/6 [==============================] - 0s 3ms/step - loss: 1.1894 - accuracy: 0.5990
6/6 [==============================] - 0s 4ms/step - loss: 2.2995 - accuracy: 0.6510
6/6 [==============================] - 0s 2ms/step - loss: 4.1071 - accuracy: 0.5000
6/6 [==============================] - 0s 4ms/step - loss: 1.1278 - accuracy: 0.6562
6/6 [==============================] - 0s 3ms/step - loss: 1.9137 - accuracy: 0.4688
6/6 [==============================] - 0s 4ms/step - loss: 2.4247 - accuracy: 0.6302
6/6 [==============================] - 0s 3ms/step - loss: 0.8804 - accuracy: 0.5312
6/6 [==============================] - 0s 1ms/step - loss: 2.0815 - accuracy: 0.5156
6/6 [==============================] - 0s 4ms/step - loss: 1.3358 - accuracy: 0.5625
6/6 [==============================] - 0s 2ms/step - loss: 1.2720 - accuracy: 0.6823
6/6 [==============================] - 0s 1ms/step - loss: 4.8965 - accuracy: 0.4479
6/6 [==============================] - 0s 2ms/step - loss: 3.6234 - accuracy: 0.6354
6/6 [==============================] - 0s 2ms/step - loss: 0.8497 - accuracy: 0.5885
```

Εικόνα 34 Εκμάθηση για 150 epochs και 20 Επαναλήψεις

```
In [10]: sum(accuracies)/len(accuracies)

Out[10]: 55.677083134651184
```

Εικόνα 35 Εντολή Εύρεσης Μέσης Τιμής Αποτελεσμάτων 2ου Αποτελέσματος

✓ Παρατηρούμε ότι έχουμε ποσοστό επιτυχίας 55.67%

- Για 30 φορές με epochs=200 θα έχουμε τα εξής αποτελέσματα:

```
In [27]: accuracies = []
         for i in range (30):
             tf.random.set_seed(i)
             model=Sequential()
             model.add(Dense(1, input_dim=8, activation='sigmoid'))
             model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
             model.fit(x_train, y_train, epochs=200, batch_size=100, verbose=0)
             _,accuracy = model.evaluate(x_test,y_test)
             accuracies.append(accuracy*100)
```

```
6/6 [==============================] - 0s 2ms/step - loss: 0.6808 - accuracy: 0.7292
6/6 [==============================] - 0s 3ms/step - loss: 2.1144 - accuracy: 0.6198
6/6 [==============================] - 0s 3ms/step - loss: 0.6646 - accuracy: 0.6615
6/6 [==============================] - 0s 2ms/step - loss: 0.8288 - accuracy: 0.6302
6/6 [==============================] - 0s 4ms/step - loss: 1.9591 - accuracy: 0.6302
6/6 [==============================] - 0s 1ms/step - loss: 0.7765 - accuracy: 0.6458
6/6 [==============================] - 0s 2ms/step - loss: 4.0805 - accuracy: 0.6354
6/6 [==============================] - 0s 1ms/step - loss: 0.6027 - accuracy: 0.7083
6/6 [==============================] - 0s 2ms/step - loss: 3.3007 - accuracy: 0.6406
6/6 [==============================] - 0s 1ms/step - loss: 1.7592 - accuracy: 0.5990
6/6 [==============================] - 0s 2ms/step - loss: 0.9329 - accuracy: 0.7083
6/6 [==============================] - 0s 2ms/step - loss: 1.6327 - accuracy: 0.6302
6/6 [==============================] - 0s 2ms/step - loss: 2.2261 - accuracy: 0.4844
6/6 [==============================] - 0s 4ms/step - loss: 4.4414 - accuracy: 0.5417
6/6 [==============================] - 0s 4ms/step - loss: 1.7773 - accuracy: 0.6771
6/6 [==============================] - 0s 2ms/step - loss: 1.8784 - accuracy: 0.5000
6/6 [==============================] - 0s 2ms/step - loss: 0.5789 - accuracy: 0.7396
6/6 [==============================] - 0s 2ms/step - loss: 2.0259 - accuracy: 0.6979
6/6 [==============================] - 0s 2ms/step - loss: 5.3318 - accuracy: 0.4427
6/6 [==============================] - 0s 2ms/step - loss: 1.4592 - accuracy: 0.6354
6/6 [==============================] - 0s 2ms/step - loss: 3.2908 - accuracy: 0.5417
6/6 [==============================] - 0s 1ms/step - loss: 2.2205 - accuracy: 0.6042
6/6 [==============================] - 0s 3ms/step - loss: 0.7910 - accuracy: 0.6458
6/6 [==============================] - 0s 2ms/step - loss: 1.2424 - accuracy: 0.6458
6/6 [==============================] - 0s 2ms/step - loss: 2.3381 - accuracy: 0.6094
6/6 [==============================] - 0s 2ms/step - loss: 1.4587 - accuracy: 0.5208
6/6 [==============================] - 0s 1ms/step - loss: 1.7774 - accuracy: 0.5573
6/6 [==============================] - 0s 2ms/step - loss: 0.6100 - accuracy: 0.7031
6/6 [==============================] - 0s 1ms/step - loss: 1.3648 - accuracy: 0.5990
6/6 [==============================] - 0s 3ms/step - loss: 3.5673 - accuracy: 0.5417
```

Εικόνα 36 Εκμάθηση για 200 epochs και 30 Επαναλήψεις

```
In [28]: sum(accuracies)/len(accuracies)
Out[28]: 61.753471990426384
```

Εικόνα 37 Εντολή Εύρεσης Μέσης Τιμής Αποτελεσμάτων 3ου Αποτελέσματος

✓ Παρατηρούμε ότι έχουμε ποσοστό επιτυχίας 61.75%

Η παραπάνω συνάρτηση στην εικόνα 36 υπολογίζει τη μέση ακρίβεια ενός συνόλου που έχουμε ορίσει βάση της συνάρτησης For τα οποία εκπαιδεύονται και αξιολογούνται πολλές φορές. Η λίστα accuracies περιέχει την ακρίβεια κάθε εκπαιδευμένου μοντέλου που αξιολογήθηκε στα δεδομένα δοκιμής. Η συνάρτηση αυτή υπολογίζει την μέση ακρίβεια για το μοντέλα που έχουμε ορίσει για την συνάρτηση For αθροίζοντας όλες τις τιμές ακρίβειας στη λίστα διαιρώντας το άθροισμα με το μήκος της λίστας. Αυτό θα μας δώσει μια μέση ακρίβεια των μοντέλων για όσες φορές το τρέξουμε.

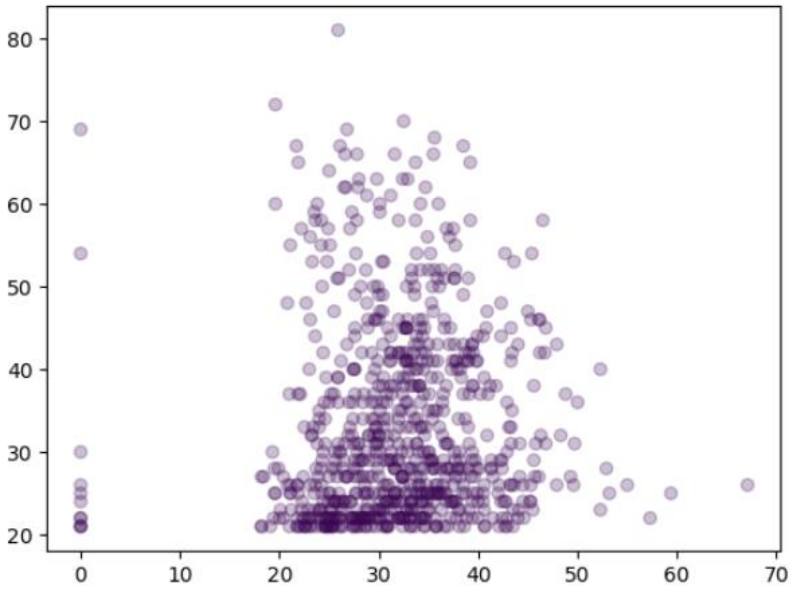
Ολοκληρώνοντας την ανάλυση αυτή παρατηρούμε ότι τα ποσοστά μας δεν είναι και το πιο ιδανικά και αυτό συμβαίνει γιατί έχουμε στην κατοχή μας έχουμε περιορισμένα δεδομένα. Όσο περισσότερα δεδομένα έχουμε να τροφοδοτήσουμε και να εκπαιδεύσουμε το Νευρωνικό Δίκτυο μας τόσο μεγαλύτερη επιτυχία θα υπάρξει και στην πρόβλεψη των αποτελεσμάτων που θα παράγει. Επιπλέον παράγουμε και ένα γράφημα ώστε να αναλύσουμε και να δούμε σε τι ηλικίες και με τι δείκτη μάζας σώματος υπάρχει περίπτωση να έχουν διαβήτη. Τρέχοντας το παρακάτω μπλοκ κώδικα:

```
In [30]: differ = np.abs(y.to_numpy()-y_pred.T)
         fig, ax = plt.subplots()
         ax.scatter(x=x['Body mass index'], y=x['Age'], c=differ, alpha=.25)

Out[30]: <matplotlib.collections.PathCollection at 0x1eb39e22c10>
```

Εικόνα 38 Μπλοκ Κώδικα για Παραγωγή Γραφήματος

Θα έχουμε το εξής γράφημα για ανάλυση:



Εικόνα 39 Γράφημα Κώδικα

Για το μπλοκ κώδικα στην εικόνα 37 έχουμε τα εξής:

1. Differ = np.abs(y.to_numpy()-y_pred.T): Εδώ τα y και y_pred ανήκουν σε Pandas Series ή Dataframe. Η μέθοδος to_numpy() χρησιμοποιείται για την μετατροπή των Series ή Dataframe αντικείμενα σε πίνακες NumPy. Ο τελεστής εκτελεί αφαίρεση κατά στοιχεία μεταξύ δυο πινάκων και η συνάρτηση Np.abs υπολογίζει την απόλυτη τιμή κατά στοιχείο, οπότε η μεταβλητή differ θα περιέχει την απόλυτη διαφορά μεταξύ των πραγματικών τιμών target (y) και των προβλεπόμενων τιμών target (y_pred).

2. Fig, ax = plt.subplots(): Εδώ, η plt.subplots δημιουργεί ένα αντικείμενο σχήματος (fig) και ένα αντικείμενο άξονα (ax) που μπορούν να χρησιμοποιηθούν για την σχεδίαση.

3. ax.scatter(x=x['Body mass index'], y=x['Age'], c=differ, alpha=.25): Ο άξονας x είναι η στήλη “Body mass index” από το Dataframe x, ο άξονας y είναι η στήλη “Age” από το Dataframe x και το χρώμα των δεικτών καθορίζεται από τη μεταβλητή differ. Το όρισμα alpha ελέγχει την διαφάνεια των δεικτών με τιμή 0,25 να σημαίνει ότι οι δείκτες θα είναι μερικώς διαφανείς. Η μέθοδος scatter δημιουργεί ένα διάγραμμα διασποράς με τα καθορισμένα δεδομένα x,y και χρώματος και σχεδιάζεται στο αντικείμενο ax του άξονα.

Το διάγραμμα διασποράς που δημιουργήθηκε από τον κώδικα είναι μια απεικόνιση που μας βοηθά στο να κατανοήσουμε καλύτερα τις σχέσεις μεταξύ δυο μεταβλητών και του πόσο καλά οι προβλέψεις ενός μοντέλου ταιριάζουν με τις πραγματικές τιμές. Έτσι αυτό το διάγραμμα διασποράς απεικονίζει την σχέση μεταξύ των μεταβλητών “Body mass index” και “Age”. Το χρώμα των σημείων αντιπροσωπεύει την διαφορά μεταξύ των πραγματικών τιμών και τον προβλεπόμενων τιμών, οπότε με το πιο σκούρο χρώμα υποδηλώνει την μεγαλύτερη διαφορά που έχουν οι τιμές μεταξύ τους ενώ οι πιο αχνές σε χρώμα δηλώνουν την μικρότερη διαφορά μεταξύ τους .

Κεφάλαιο 4 Κώδικας

4.1 Κώδικας Μοντέλου Στεφανιαίας Νόσου

```
import tensorflow as tf
import numpy as np
import pandas as pd
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import IntegerLookup
from tensorflow.keras.layers import Normalization
from tensorflow.keras.layers import StringLookup

file_url =
"http://storage.googleapis.com/download.tensorflow.org/data/he
art.csv"
dataframe = pd.read_csv(file_url)
dataframe.shape
dataframe.head()
val_dataframe = dataframe.sample(frac=0.2, random_state=1337)
train_dataframe = dataframe.drop(val_dataframe.index)

print(
    "Using %d samples for training and %d for validation"
    % (len(train_dataframe), len(val_dataframe))
)
def dataframe_to_dataset(dataframe):
    dataframe = dataframe.copy()
    labels = dataframe.pop("target")
    ds = tf.data.Dataset.from_tensor_slices((dict(dataframe),
labels))
    ds = ds.shuffle(buffer_size=len(dataframe))
    return ds


train_ds = dataframe_to_dataset(train_dataframe)
val_ds = dataframe_to_dataset(val_dataframe)

for x, y in train_ds.take(1):
    print("Input:", x)
    print("Target:", y)
train_ds = train_ds.batch(32)
val_ds = val_ds.batch(32)
```

```
def encode_numerical_feature(feature, name, dataset):
    # Create a Normalization layer for our feature
    normalizer = Normalization()

    # Prepare a Dataset that only yields our feature
    feature_ds = dataset.map(lambda x, y: x[name])
    feature_ds = feature_ds.map(lambda x: tf.expand_dims(x, -
1))

    # Learn the statistics of the data
    normalizer.adapt(feature_ds)

    # Normalize the input feature
    encoded_feature = normalizer(feature)
    return encoded_feature


def encode_categorical_feature(feature, name, dataset,
is_string):
    lookup_class = StringLookup if is_string else
IntegerLookup
    # Create a lookup layer which will turn strings into
integer indices
    lookup = lookup_class(output_mode="binary")

    # Prepare a Dataset that only yields our feature
    feature_ds = dataset.map(lambda x, y: x[name])
    feature_ds = feature_ds.map(lambda x: tf.expand_dims(x, -
1))

    # Learn the set of possible string values and assign them
a fixed integer index
    lookup.adapt(feature_ds)

    # Turn the string input into integer indices
    encoded_feature = lookup(feature)
    return encoded_feature
```

```
# Categorical features encoded as integers
sex = keras.Input(shape=(1,), name="sex", dtype="int64")
cp = keras.Input(shape=(1,), name="cp", dtype="int64")
fbs = keras.Input(shape=(1,), name="fbs", dtype="int64")
restecg = keras.Input(shape=(1,), name="restecg",
dtype="int64")
exang = keras.Input(shape=(1,), name="exang", dtype="int64")
ca = keras.Input(shape=(1,), name="ca", dtype="int64")

# Categorical feature encoded as string
thal = keras.Input(shape=(1,), name="thal", dtype="string")

# Numerical features
age = keras.Input(shape=(1,), name="age")
trestbps = keras.Input(shape=(1,), name="trestbps")
chol = keras.Input(shape=(1,), name="chol")
thalach = keras.Input(shape=(1,), name="thalach")
oldpeak = keras.Input(shape=(1,), name="oldpeak")
slope = keras.Input(shape=(1,), name="slope")

all_inputs = [
    sex,
    cp,
    fbs,
    restecg,
    exang,
    ca,
    thal,
    age,
    trestbps,
    chol,
    thalach,
    oldpeak,
    slope,
]

# Integer categorical features
sex_encoded = encode_categorical_feature(sex, "sex", train_ds,
False)
cp_encoded = encode_categorical_feature(cp, "cp", train_ds,
False)
fbs_encoded = encode_categorical_feature(fbs, "fbs", train_ds,
False)
restecg_encoded = encode_categorical_feature(restecg,
"restecg", train_ds, False)
exang_encoded = encode_categorical_feature(exang, "exang",
train_ds, False)
ca_encoded = encode_categorical_feature(ca, "ca", train_ds,
False)
```

```
# String categorical features
thal_encoded = encode_categorical_feature(thal, "thal",
train_ds, True)

# Numerical features
age_encoded = encode_numerical_feature(age, "age", train_ds)
trestbps_encoded = encode_numerical_feature(trestbps,
"trestbps", train_ds)
chol_encoded = encode_numerical_feature(chol, "chol",
train_ds)
thalach_encoded = encode_numerical_feature(thalach, "thalach",
train_ds)
oldpeak_encoded = encode_numerical_feature(oldpeak, "oldpeak",
train_ds)
slope_encoded = encode_numerical_feature(slope, "slope",
train_ds)

all_features = layers.concatenate(
    [
        sex_encoded,
        cp_encoded,
        fbs_encoded,
        restecg_encoded,
        exang_encoded,
        slope_encoded,
        ca_encoded,
        thal_encoded,
        age_encoded,
        trestbps_encoded,
        chol_encoded,
        thalach_encoded,
        oldpeak_encoded,
    ]
)
x = layers.Dense(32, activation="relu")(all_features)
x = layers.Dropout(0.5)(x)
output = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(all_inputs, output)
model.compile("adam", "binary_crossentropy",
metrics=["accuracy"])

# `rankdir='LR'` is to make the graph horizontal.
keras.utils.plot_model(model, show_shapes=True, rankdir="LR")

model.fit(train_ds, epochs=20, validation_data=val_ds)
```

4.2 Κώδικας Μοντέλου Διαβήτη

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Sequential
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib inline
data = pd.read_csv('diabetes.csv')
data.head()
data.isna().sum()
data.dtypes
x = data.iloc[:,:-1]
y = data.iloc[:,-1]
x_train, x_test, y_train, y_test = train_test_split(x,y,
random_state=42)
accuracies = []
for i in range (20):
    tf.random.set_seed(i)
    model=Sequential()
    model.add(Dense(1, input_dim=8, activation='sigmoid'))
    model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])
    model.fit(x_train, y_train, epochs=150, batch_size=100,
verbose=0)
    _,accuracy = model.evaluate(x_test,y_test)
    accuracies.append(accuracy*100)
sum(accuracies)/len(accuracies)
y_pred = model.predict(x)
y_pred = np.where(y < .5,0,1 )
differ = np.abs(y.to_numpy()-y_pred.T)
fig, ax = plt.subplots()
ax.scatter(x=x['Body mass index'], y=x['Age'], c=differ,
alpha=.25)
```

Τα δεδομένα για το συγκεκριμένο μοντέλο είναι από το Kaggle.com όπως αναφέρεται και στην βιβλιογραφία στο 5ο κεφάλαιο.

Βιβλιογραφία

Miroslav Kumbat (2018), An Introduction to Machine Learning (Second Edition)

Wolfgang Ertel (2011), Introduction to Artificial Intelligence (Second Edition)

Sandro Skansi (2018), Introduction to Deep Learning (From the Logical Calculus to Artificial Intelligence) (First Edition)

Charu C. Aggarwal (2018), Neural Networks and Deep Learning (First Edition)

https://www.python.org/

https://keras.io/

https://www.tensorflow.org/

https://www.kaggle.com/