Εργαστηριακή Άσκηση

Μέρος Α΄

ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΟΗΜΟΣΥΝΗ

CEID_NE5218

Στοιχεία Φοιτητή

Υπεύθυνος Καθηγητής: Σ. Λυκοθανάσης, Δ. Κουτσομητρόπουλος

Ακαδημαϊκό Έτος : 10^{Ο} Εξάμηνο 2024-2025

Ονοματεπώνυμο: Ορέστης Αντώνιος Μακρής **Α-Μ** 1084516

Πανεπιστήμιο Πατρών Τμήμα Μηχανικών Η/Υ και Πληροφορική

Table of Contents

Link κώδικα Github:

https://github.com/OrestisMakris/Computational_Intelligence.git

Εισαγωγή

Η εργασία υλοποιήθηκε σε όλο το εύρος της με βάση την εκφώνηση και όλα τα υπό ερωτήματα συν το Bonus βαθύ νευρωνικό δίκτυο.

Περιβάλλον

Για την υλοποίηση του κώδικα και την εκτέλεση των πειραμάτων, χρησιμοποιήθηκε το περιβάλλον ανάπτυξης conda.

Μερικοί λόγοι για τη χρήση του conda περιλαμβάνουν τη δυνατότητα δημιουργίας και διαχείρισης απομονωμένων περιβαλλόντων για κάθε project , αποτρέποντας συγκρούσεις μεταξύ βιβλιοθηκών. Υποστηρίζει εύκολη ενσωμάτωση με CUDA για χρήση GPU, εξασφαλίζει αναπαραγωγιμότητα μέσω του environment.yml, και παρέχει απλοποιημένη εγκατάσταση και διαχείριση πακέτων και εκδόσεων. Επιπλέον, το conda είναι συμβατό με πολλαπλά λειτουργικά συστήματα (Windows, macOS, Linux), διευκολύνοντας τη μεταφορά και την αναπαραγωγή του περιβάλλοντος σε διαφορετικές συσκευές και πλατφόρμες ανάπτυξης. Τέλος, προσφέρει ευκολία στην εγκατάσταση βιβλιοθηκών, ακόμα και εκείνων με εξαρτήσεις σε compiled γλώσσες (torch γραμμένη σε c++ και c), μειώνοντας τα πιθανά σφάλματα εγκατάστασης και επιτρέποντας την ομαλή εκτέλεση του κώδικα σε πολλαπλα στησίματα!

Εγκατάσταση του Conda

Για την εγκαταστήστε το Miniconda ή το Anaconda από την επίσημη ιστοσελίδα:

https://www.anaconda.com/download

Δημιουργία περιβάλλοντος Conda

Δημιουργία του περιβάλλοντος μπορεί να πραγματοποιηθεί μέσα από ένα τερματικό (terminal) και μεταβαίνοντας στον φάκελο που βρίσκεται το αρχείο \cancer-detection\cancer-detection\environment.yml.

Στην συνέχεια εκτελώντας την εντολή: **conda env create -f environment.yml**

θα δημιουργηθεί ένα περιβάλλον με όνομα cancer-detection και θα εγκαταστήσει όλα τα απαραίτητα πακέτα/βιβλιοθήκες.

Αφού δημιουργηθεί το περιβάλλον, ενεργοποιήστε το με την εντολή:

conda activate cancer-detection

Εφόσον το περιβάλλον είναι ενεργό, προχωράμε στην εκτέλεση των αρχείων Python (π.χ. cancer_detection_A1_A2.py) με: **python cancer_detection_A1_A2.py**

Επιλογή Βιβλιοθηκών και Modules

Για την υλοποίηση του προτεινόμενου πλαισίου επεξεργασίας δεδομένων και εκπαίδευσης μοντέλου, επιλέξαμε να χρησιμοποιήσουμε τις εξής βιβλιοθήκες:

Pandas: Η βιβλιοθήκη Pandas (**import pandas as pd**) επιτρέπει την ανάγνωση, διαχείριση και ανάλυση δεδομένων μέσω των δομών τύπου DataFrame, προσφέροντας εύκολες λειτουργίες για φιλτράρισμα, συγχώνευση και μετασχηματισμό δεδομένων.

NumPy: Η NumPy (**import numpy as np**) παρέχει υποστήριξη για πολυδιάστατους πίνακες και αριθμητικούς υπολογισμούς υψηλής απόδοσης, επιτρέποντας αποδοτική διαχείριση και χειρισμό δεδομένων μέσω arrays και μαθηματικών συναρτήσεων..

scikit-learn: Η scikit-learn περιλαμβάνει πολλά modules. Από το s klearn.model_selection, χρησιμοποιήθηκαν τα StratifiedKFold και learning_curve. Το StratifiedKFold εφαρμόζει σταυρωτή επικύρωση διατηρώντας την κατανομή των τάξεων σε κάθε fold, ενώ το learning_curve βοηθά στον εντοπισμό underfitting h overfitting υπολογίζοντας την απόδοση του μοντέλου σε συνάρτηση με το μέγεθος των δεδομένων. Από το sklearn.preprocessing, χρησιμοποιήθηκαν οι StandardScaler, MinMaxScaler, RobustScaler, OneHotEncoder και LabelEncoder. Ο StandardScaler εφαρμόζει κανονικοποίηση ώστε τα χαρακτηριστικά να έχουν μέσο όρο 0 και διασπορά 1, ο MinMaxScaler τα κλιμακώνει σε συγκεκριμένο εύρος τιμών (συνήθως 0 έως 1), ο RobustScaler μειώνει την επίδραση των ακραίων τιμών χρησιμοποιώντας το εύρος τεταρτημορίων, ο OneHotEncoder μετατρέπει τις κατηγορικές μεταβλητές σε δυαδικές binary μορφές και ο LabelEncoder κωδικοποιεί τις κατηγορίες σε ακέραιους αριθμούς. Το ColumnTransformer από το sklearn. compose εφαρμόζει διαφορετικούς μετασχηματισμούς σε διαφορετικές στήλες δεδομένων εντός ενός pipeline.

Pytorch; Η βιβλιοθήκη PyTorch αποτελεί τη βάση για την κατασκευή του νευρονικού μας δικτύου. Το torch προσφέρει τη βασική υποστήριξη για tensors και αριθμητικές πράξεις σε CPU ή GPU. Το torch.nn περιλαμβάνει modules για την κατασκευή του μοντέλου, όπως πλήρως συνδεδεμένα layers (Linear), συναρτήσεις ενεργοποίησης (ReLU, Sigmoid) και συναρτήσεις κόστους (CrossEntropyLoss). Το torch.optim προσφέρει βελτιστοποιητές όπως SGD, Adam και RMSprop, που προσαρμόζουν τα βάρη του δικτύου κατά την εκπαίδευση.

Matplotlib: Η Matplotlib (import matplotlib.pyplot as plt) χρησιμοποιήθηκε για τη δημιουργία γραφημάτων απόδοσης, όπως learning curves και confusion matrices, προσφέροντας ευέλικτη σχεδίαση στατικών και διαδραστικών γραφικών.

Seaborn : Η Seaborn (import seaborn as sns) είναι μια βιβλιοθήκη γραφημάτων βασισμένη στη Matplotlib, η οποία διευκολύνει τη δημιουργία πιο ελκυστικών και στατιστικά πιο περιεκτικών απεικονίσεων, όπως heatmaps, pairplots και boxplots, καθιστώντας την ερμηνεία των δεδομένων πιο άμεση.

Η επιλογή αυτών των βιβλιοθηκών έγινε λαμβάνοντας υπόψη την ευκολία χρήσης, την απόδοση την κοινότητα υποστήριξής τους και την δυνατότητα υποστήριξης από διαφορετικές πλατφόρμες υλοποίησης.

Για την εκπαίδευση του νευρωσικού δικτυού χρησιμοποιήθηκαν τα χαρακτηριστικά Age,Gender,Ethnicity,EducationLevel,BMI,Smoking,AlcoholConsumption,PhysicalActivity, DietQuality,SleepQuality,FamilyHistoryAlzheimers,CardiovascularDisease,Diabetes,Depress

ion,HeadInjury,Hypertension,SystolicBP,DiastolicBP,CholesterolTotal,CholesterolLDL,CholesterolHDL,CholesterolTriglycerides,MMSE,FunctionalAssessment,MemoryComplaints,BehavioralProblems,ADL,Confusion,Disorientation,PersonalityChanges,DifficultyCompletingTasks,Forgetfulness,Diagnosis.

Α1. Προεπεξεργασία και Προετοιμασία δεδομένων

α) Ανάγνωση του συνόλου δεδομένων και πρώτη επεξεργασία:

Η προεπεξεργασία δεδομένων αποτέλεσε σημαντικό βήμα στη διαδικασία ανάπτυξη του μοντέλων μηχανικής μάθησης στα πλαίσιά της εργασίας. Η συνολική διαδικασία της προεπεξεργασίας δεδομένων που ακολουθούμε περιλαμβάνει τα ακόλουθα βήματα:

Η ανάγνωση και η αρχική επεξεργασία των δεδομένων υλοποιούνται μέσω της συνάρτησης load_data(csv_path), η οποία ξεκινά με το pd.read_csv για τη φόρτωση του CSV σε ένα DataFrame, διασφαλίζοντας ότι η μνήμη διαχειρίζεται αποδοτικά ακόμη και για μεγάλα αρχεία. Στη συνέχεια, απορρίπτονται δυναμικά οι μη επιθυμητές στήλες (DoctorInCharge, PatientID) μέσω επαναληπτικής αφαίρεσης, ώστε να διατηρηθεί η γενικότητα της συνάρτησης και να αποφευχθούν λάθη σε διαφορετικές εκδόσεις του dataset. Δεν απαιτητέ η κωδικοποίηση της στήλης Diagnosis σε ακέραιες ετικέτες πχ με τον LabelEncoder, καθώς βρίσκονται είδη κωδικοποιημένα στην μορφή 0 και 1.

Τα χαρακτηριστικά διαχωρίζονται σε τρεις προκαθορισμένες κατηγορίες μη συνεχών: τα ordinal (EducationLevel), όπου η σειρά των τιμών φέρει σημασιολογία, τα nominal (Ethnicity), όπου οι κατηγορίες είναι ασυσχέτιστες μεταξύ τους, και τα binary (π.χ. Gender, Smoking, Hypertension κ.ά.), όπου κάθε στήλη αναπαριστά μια διχοτομημένη μεταβλητή. Αντί να καθοριστούν μεμονωμένα όλες οι υπόλοιπες στήλες, αυτές αναγνωρίζονται αυτόματα ως συνεχείς μέσω διαφοράς συνόλων (all_feature_cols – non_cont_cols), υποστηρίζοντας προσθήκη ή κατάργηση νέων χαρακτηριστικών χωρίς αλλαγές στον κώδικα.

Ο παραπάνω διαχωρισμός εφαρμόστηκε με απώτερο σκοπό να εφαρμοστούν οι κατάλληλοι μετασχηματισμοί για κάθε διαφορετικό είδος δεδομένων χαρακτηριστικών, διότι στο σύνολο του dateset υπάρχουν ποσοτικές τιμές (π.χ. μετρήσεις βιοδεικτών), αλλά και κατηγορικές (π.χ. φύλο, εθνικότητα, κάπνισμα, ύπαρξη συμπτωμάτων κλπ), διατεταγμένες τιμές (επίπεδο εκπαίδευσης, γνωστική κατάσταση, ποιότητα ύπνου κ.α.) , οι διατεταγμένες μπορεί να είναι είτε συνεχής float είτε συνεχής int. Περισσότερα ακολουθούν στην ενότητα γ) Κανονικοποίηση / Μετασχηματισμός δεδομένων.

Για την αξιοπιστία της ροής, εκτελείται ένα απλό “sanity check” που εντοπίζει αμέσως αν κάποια στήλη δεν έχει ταξινομηθεί σε καμία κατηγορία, επιστρέφοντας αναλυτικό μήνυμα λάθους ώστε ο χρήστης να διορθώσει εγκαίρως όποιες παραλείψεις. Τέλος, το DataFrame χωρίζεται σε X (με τα διαδοχικά blocks συνεχών, ordinal, nominal και binary στηλών, διατηρώντας προβλεπόμενη δομή για pipelines) και y (η κωδικοποιημένη διάγνωση σε πίνακα ακέραιων τιμών). Η συνάρτηση επιστρέφει επίσης τις λίστες continuous_cols,

ordinal_cols, nominal_cols και binary_cols, διευκολύνοντας την εφαρμογή περαιτέρω μετασχηματιστών, επιλογή χαρακτηριστικών ή επεκτάσεις σε full ML workflows.

β) Ανάλυση του συνόλου δεδομένων:

Πριν προχωρήσουμε στην κανονικοποίηση των δεδομένων αποφάσισα να πραγματοποιήσω μια μικρή ανάλυση των δεδομένων του συνόλου εκπαίδευσης και επικύρωσής για να δούμε τι κατανομές ακολουθεί το κάθε χαρακτηριστικό τι μέση τιμή έχουν και πως συσχετίζονται μεταξύ τους.

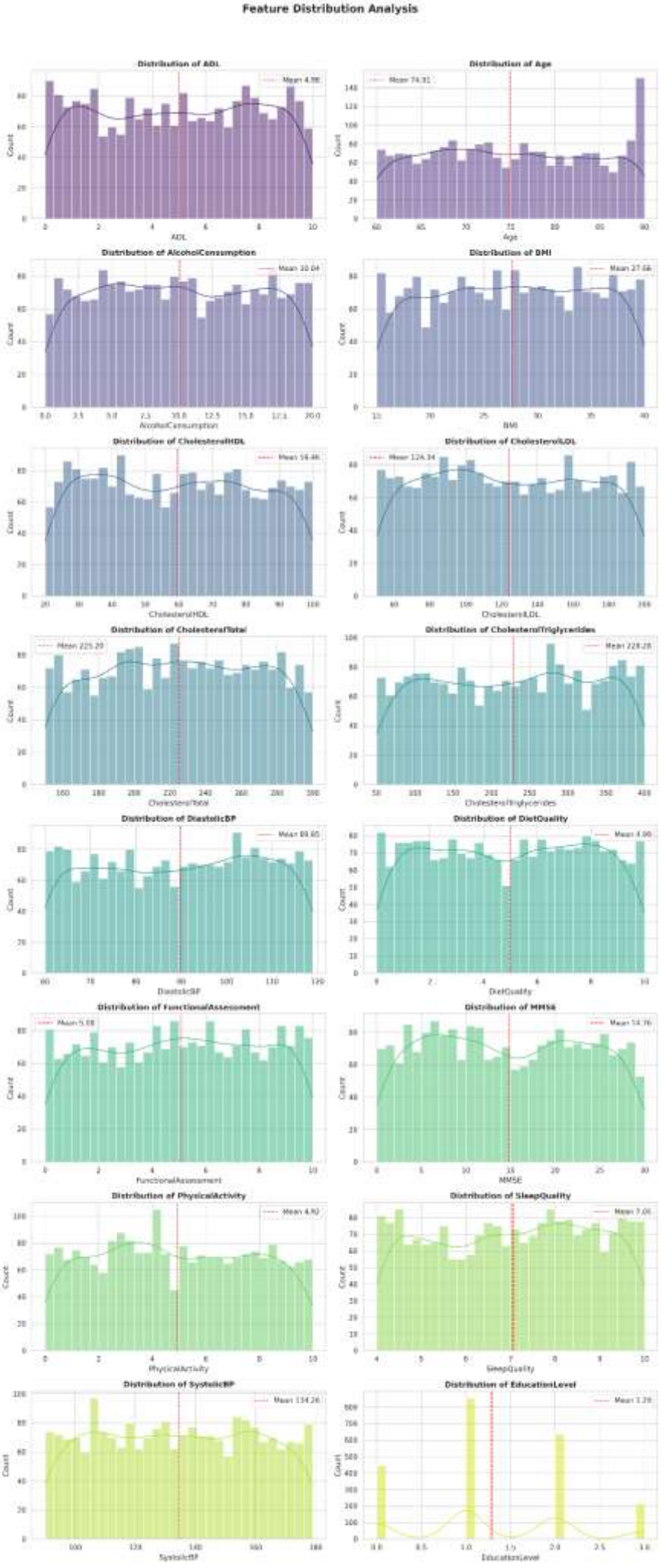
Οπτικοποίηση Κατανομών Συνεχών Χαρακτηριστικών

Η συνάρτηση plot_numeric_distributions_subplot(df, numeric_cols, save_path) έχει σχεδιαστεί για να απεικονίζει ths κατανομές όλων των συνεχών μεταβλητών ενός DataFrame, διευκολύνοντας την κατανόηση της διασποράς, της ασυμμετρίας και πιθανών outliers πριν από την εκπαίδευση του μοντέλου. Επιλέχθηκε αυτή η ανάλυση γιατί η οπτική εκτίμηση των κατανομών αποκαλύπτει κρίσιμες πληροφορίες για το σχήμα των δεδομένων – όπως skewness, βαριές ουρές ή πολλαπλές κορυφές – οι οποίες καθορίζουν ποια μετασχηματιστικά βήματα (π.χ. λογαριθμική κλίμακα, box-cox, robust scaling) θα χρειαστεί να εφαρμόσουμε ώστε να ικανοποιηθούν οι υποθέσεις των αλγορίθμων μας και να βελτιώσουμε τη σταθερότητα της εκπαίδευσης.

Αρχικά, ορίζονται κοινό στυλ και context (paper) στην Seaborn, εξασφαλίζοντας ομοιομορφία και αναγνωσιμότητα σε δημοσιεύσιμες εικόνες. Ο αριθμός των υποπλωκών υπολογίζεται δυναμικά βάσει του πλήθους της λίστας numeric_-cols, με δύο στήλες ανά σειρά, έτσι ώστε το διάγραμα να προσαρμόζεται αυτόματα σε περισσότερες η λιγότερες μεταβλητές.

Για κάθε στήλη, χρησιμοποιείται η histplot με 3 0 κάδους και ενεργοποιημένη γραμμή πυκνότητας (KDE) για λεπτομερή εικόνα των συχνοτήτων, ενώ κάθε ιστογράφημμα χρωματίζεται από παλέτα “viridis” για οπτική συνέπεια. Η κάθετη γραμμή (axvline) στον μέσο όρο επισημαίνει τη θέση του κεντρικού τείχους και βοηθά στον άμεσο εντοπισμό μετατοπίσεων ή ασυμμετριών. Τυχόν επιπλέον άξονες (όταν ο συνολικός αριθμός των subplots υπερβαίνει τον αριθμό των στηλών) αφαιρούνται με fig. delaxes, διατηρώντας καθαρή διάταξη.

Τέλος, το αποτέλεσμα αποθηκεύεται ως αρχείο εικόνας υψηλής ανάλυσης 300 dppi με plt.savefig, εξασφαλίζοντας επαναληψιμότητα και εύκολη ενσωμάτωση σε αναφορές ή παρουσιάσεις χωρίς περαιτέρω χειροκίνητη ρύθμιση. με αυτόν τον τρόπο, η λειτουργία καλύπτει πλήρως την ανάγκη για γρήγορη, επεκτάσιμη και επαναχρησιμοποιήσιμη διερευνητική οπτικοποίηση των συνεχών χαρακτηριστικών, ενώ μας καθοδηγεί στην επιλογή κατάλληλων μετασχηματισμών και scaling πριν την εκπαίδευση των μοντέλων


Feature Distribution Analysis
Distribution of ADL
Distribution of Age
Distribution of AlcoholConsumption
Distribution of BMI
Distribution of CholesterolHDL
Distribution of CholesterolLDL
Distribution of CholesterolTotal
Distribution of CholesterolTriglycerides
Distribution of DiastolicBP
Distribution of DietQuality
Distribution of FunctionalAssessment
Distribution of MMSE
Distribution of PhysicalActivity
Distribution of SleepQuality
Distribution of SystolicBP
Distribution of EducationLevel
Count

Από τα ιστογράμματα των συνεχων χαρακτηριστικών διακρίνουμε τα εξής:

ADL: φαίνεται σχετικά επίπεδη χωρίς έντονη συσσώρευση γύρω από έναν μόνο άξονα, υποδεικνύοντας ομοιόμορφη διασπορά και μικρή ανάγκη μετασχηματισμού.

Age: Ελαφρώς δεξιά ασύμμετρη (right-skewed), με περισσότερα δείγματα στις μεγαλύτερες ηλικίες και ένα “peak” κοντά στα 90–95 έτη, που ίσως χρειάζεται λογαριθμικό ή box-cox μετασχηματισμό για να εξομαλυνθεί η ουρά (τελικά ύστερα από πειραματισμό είδαμε ότι ένας τέτοιος μετασχηματισμός δεν έχει ιδιαίτερη σχεδόν καθόλου βελτιώσει στην απόδοση του μοντέλο).

AlcoholConsumption: η AlcoholConsumption εδω παρατηρούμε πάλη ομοιόμορφη κατανομή.

BMI: Σχετικά κανονική κατανομή γυρω από τον μέσο όρο (~27), χωρίς έντονες ακραίες τιμές, οπότε ο απλός StandardScaler είναι επαρκής.

CholesterolHDL: Ελαφρώς αριστερά ασύμμετρη (left-skewed), με υψηλή πυκνότητα στα 50–70.

CholesterolLDL & CholesterolTotal: Ουδέτερες έως ελαφρώς δεξιά συσχετισμένες κατανομές, κατάλληλες για scaling με Standard ή MinMax.

CholesterolTriglycerides: Ουδέτερες έως ελαφρώς δεξιά συσχετισμένες κατανομές, κατάλληλες για scaling με Standard ή MinMax.

DiastolicBP & SystolicBP: Σχεδόν κανονικές κατανομές με ελαφρά δεξιά μετατόπιση, όπου το StandardScaler θα επαρκεί στις περισσότερες περιπτώσεις.

DietQuality, PhysicalActivity, FunctionalAssessment: Σχεδόν ομοιόμορφες κατανομές σε όλο το εύρος των δυνατών τιμών (0–10), υποδεικνύοντας ότι δεν απαιτούν ειδικούς μετασχηματισμούς πέραν του scaling.

MMSE: Ελαφρώς αριστερά ασύμμετρη, με περισσότερα δείγματα σε υψηλότερους βαθμούς γνωστικών λειτουργιών, όπου ένα power transform μπορεί να βελτιώσει τη συμμετρία.

SleepQuality: Μικρή ασυμμετρία προς τα αριστερά, αλλά χωρίς ακραίες αποκλίσεις· το StandardScaler ή MinMaxScaler είναι κατάλληλα.

EducationLevel: Διακριτές κορυφές σε επίπεδα 0, 1, 2, 3 που αναπαριστούν βαθμίδες εκπαίδευσης· εδώ δεν εφαρμόζουμε continuous scaling, αλλά ενδεχομένως one-hot encoding ή ordinal encoding με awareness της διαβάθμισης.

Οπτικοποίηση Κατανομής Κλάσεων

Η συνάρτηση plot_class_distribution εφαρμόζει την ανάλυση της κατανομής των ετικετών-στόχων (y) με τη μορφή ραβδογράμματος, ώστε να αποτυπώνονται άμεσα τυχόν ανισορροπίες μεταξύ των κλάσεων. Με τιν χρήση τις np.bincount υπολογίζονται οι

συχνότητες εμφάνισης κάθε κλάσης, οι οποίες αποθηκεύονται σε ένα προσωρινό DataFrame (df_counts) με δύο στήλες: "Class" και "Count".

Για την απεικόνιση, χρησιμοποιείται το sns.barplot με παλέτα “viridis” που εγγυάται ευδιάκριτη έγχρωμη κωδικοποίηση και συμβατότητα με τα υπόλοιπα γραφήματα του project. Κάθε μπάρα φέρει ετικέτα με τον ακριβή αριθμό δειγμάτων (μέσω ax.text), τοποθετημένη λίγο πάνω από την κορυφή της, διευκολύνοντας την άμεση ανάγνωση των ποσοτήτων χωρίς να χρειάζεται επιπλέον λεζάντα. Εν τέλη το αποτέλεσμα αποθηκεύεται ως αρχείο εικόνας με plt.savefig, επιτρέποντας απευθείας ενσωμάτωση σε αναφορές ή διαφάνειες.

Παρατηρήσεις & Σημασία για Αντιμετώπιση Ανισορροπίας

Η απεικόνιση της κατανομής των κλάσεων αποκάλυψε σημαντική ανισορροπία: 1 389 δείγματα χωρίς διάγνωση Alzheimer (κλάση 0) έναντι 760 με θετική διάγνωση (κλάση 1). Αρχικά δοκιμάστηκε undersampling της πλειοψηφικής κλάσης ώστε να εξισορροπηθούν οι συλλήψεις, όμως το μοντέλο εμφάνισε σημαντική πτώση στην απόδοση, υποδηλώνοντας απώλεια κρίσιμων πληροφοριών. Επιπλέον, πρόκειται για ιατρικά δεδομένα όπου η μείωση δειγμάτων από την «υγιή» ομάδα μπορεί να καταστήσει επικίνδυνη την εκπαίδευση, καθώς ενδέχεται να χάσουμε υποδείγματα σπάνιων αλλά σημαντικών υποκατηγοριών.

Για τους λόγους αυτούς, αποφασίστηκε να διατηρηθεί η πραγματική κατανομή των κλάσεων και αντί για undersampling να χρησιμοποιηθούν τεχνικές weighting στον ταξινομητή και εξειδικευμένες μετρικές (π.χ. F1-score) κατά την εκπαιδευση και αξιολόγηση. Με αυτόν τον τρόπο διασφαλίζεται ότι το μοντέλο δεν θα μεροληπτεί υπέρ της πλειοψηφίας, ενώ παράλληλα διατηρούμε το πλήρες σύνολο των διαθέσιμων ιατρικών δειγμάτων για πιο αξιόπιστη και γενικεύσιμη πρόβλεψη.

Heatmap

Η συνάρτηση plot_correlation_heatmap(df, numeric_cols, save_patth) υπολογίζει τη μητρόπολη συσχέτισης (Pearson) μεταξύ όλων των συνεχών χαρακτηριστικών (numeric_cols) και την απεικονίζει σε ένα heatmap για εύκολη και γρήγορη εντύπωση των διασυνδέσεων. Πρώτα εξάγουμε τον πίνακα συσχέτισης με το df[numeric_cols].corr(). Έπειτα δημιουργούμε ένα figure μεγέθους 14 * 12 ιντσών και ορίζουμε μια diverging παλέτα ("diverging_palette") που τονίζει τις θετικές (προς το μπλε) και τις αρνητικές (προς το κόκκινο) συσχετίσεις, με κλίμακα από –1 έως +1. Με το sns.heatmap εμφανίζουμε την πλήρη μήτρα (χωρίς mask), προσθέτοντας annot μεταξύ των κελιών (με δύο δεκαδικά) και χωρίζοντας τα τετράγωνα με λεπτές γραμμές (linewidths=.5). Τέλος, η εικόνα αποθηκεύεται σε αρχείο υψηλής ανάλυσης (300 dpi) με plt.savefig και κλείνει για να απελευθερωθεί η μνήμη.

Παρατηρήσεις από το Heatmap

Από το heatmap συσχέτισης (εικόνα) διαπιστώνουμε ότι καμία μεταβλητή δεν εμφανίζει σημαντική γραμμοσκιασμένη συσχέτιση με άλλη εκτός της ίδιας (όλες οι off-diagonal τιμές βρίσκονται κάτω του r = 0.06). Συγκεκριμένα:

Ο δείκτης Age έχει τη μεγαλύτερη (αρνητική) συσχέτιση με το EducationLevel (r≈–0. 06), υποδηλώνοντας ότι οι υψίλότερης ηλικίας συμετεχοντες τηνουν να έχουν χαμηλότερα επίπεδα τυπικής εκπαίδευσης.

Μικρές θετικές συσχετίσεις (r=0.05) εντοπίζονται μεταξύ ADL και FunctionalAssessment, καθώς και μεταξύ SleepQuality και Age, υποδεικνύοντας ελαφρά συνάφεια στην καθημερινή λειτουργικότητα.

Οι δείκτες λιπιδίων (CholesterolHDL, CholesterolLDL, CholesterolTotal, CholesterolTriglycerides) εμφανίζουν σχεδόν μηδενικές συσχετίσεις μεταξύ τους, δείγμα ότι παρά τη βιολογική τους σύνδεση τα επίπεδα στα δεδομένα μας παρουσιάζουν ανεξάρτητες διακυμάνσεις.

Η απουσία ισχυρών (>0.8) συσχετίσεων σημαίνει ότι δεν υφίσταται σοβαρή πολυ-συνάφεια ανάμεσα στα συνεχή χαρακτηριστικά, οπότε αρα αυτό σημαίνει δεν απαιτειται άμεση εφαρμογή μεθόδων μείωσης διαστατικότητας (π.χ. PCA) αποκλειστικά για λόγους αποφυγής πολυ-συνάφειας.

Επιπλέον, λογο ότι πρόκειται για ιατρικά δεδομένα, η διατήρηση όλων των δεικτών ενισχύει την ερμηνευσιμότητα και την κλινική αξιοπιστία των προβλέψεων.

Με βάση τα παραπάνω, ο σχεδιασμός του feature-engineering συνεχίζει με μετασχηματισμούς για όλα τα numeric χαρακτηριστικά, χωρίς να απαιτούνται ειδικοί μετασχηματισμοί λόγω multicollinearity.

γ) Κανονικοποίηση / Μετασχηματισμός δεδομένων:

Ανάλυση μετασχηματισμών

Παρακάτω παρουσιάζεται μια αναλυτική σύγκριση των κυριότερων μεθόδων προεπεξεργασίας δεδομένων που μας δόθηκαν από την εκφώνηση.

Κεντράρισμα (Centering)

Το κεντράρισμα αφαιρεί από κάθε τιμή του χαρακτηριστικού τον μέσο όρο του συγκεκριμένου χαρακτηριστικού.

Μαθηματικός μετασχηματισμός:

$$xi' = xi - \bar{x}$$

όπου xi είναι κάθε τιμή του χαρακτηριστικού και $\bar{x}$ ο μέσος όρος αυτών.

Συχνά χρησιμοποιείται ως προ-επεξεργασία πριν εφαρμοστεί άλλη κλιμάκωση ή μετασχηματισμός, διότι βοηθά στη μείωση της προκατάληψης (bias) των χαρακτηριστικών. Ενισχύει τη σύγκλιση σε βελτιστοποιητικές διαδικασίες (π.χ. gradient descent)..

Συνήθως το κεντράρισμα εφαρμόζεται πριν από άλλες μεθόδους κλιμάκωσης όπως η τυποποίηση ή η κανονικοποίηση. Επομένως, είναι συνήθως συμβατό με άλλες μεθόδους, όπως και είναι στην περίπτωση που πραγματοποιήθηκε μέσο άλλων μεθόδων.

Κανονικοποίηση (Normalization / Min-Max Scaling)

Η κανονικοποίηση αναδιαμορφώνει τα δεδομένα ώστε οι τιμές κάθε χαρακτηριστικού να μετακινηθούν μέσα σε ένα προκαθορισμένο διάστημα, συχνά [0, 1] ή [-1, 1].

Μαθηματικός μετασχηματισμός:

$$xi' = \frac{xi - min(x)}{\max(x) - \min(x)}$$

Η παραπάνω μαθηματική έκφραση αφορά το διάστημα [0, 1] σε περίπτωση που επιθυμούμε άλλο διάστημα, προστίθεται η αντίστοιχη μετατόπιση και κλιμάκωση.

Η Κανονικοποίηση ιδανική για νευρωνικά δίκτυα και αλγορίθμους που απαιτούν εισόδους εντός συγκεκριμένου διαστήματος. Εξασφαλίζει ότι όλες οι μεταβλητές έχουν το ίδιο εύρος τιμών, μειώνοντας την πιθανότητα κάποιας μεταβλητής να υπερεκτιμηθεί. Επίσης ο μετασχηματισμός διατηρεί την κατανομή των δεδομένων.

Τυποποίηση (Standardization / Z-score)

Η τυποποίηση αναδιαμορφώνει τα δεδομένα ώστε κάθε χαρακτηριστικό να έχει μέση τιμή 0 και διακύμανση 1.

$$xi' = \frac{xi - \mu}{\sigma}$$

όπου μ είναι ο μέσος όρος του χαρακτηριστικού και σ η τυπική του απόκλιση.

Εξασφαλίζει ισορροπημένη συνεισφορά όλων των χαρακτηριστικών ανεξαρτήτως της αρχικής τους κλίμακας. Συμβάλλει σε ταχύτερη και πιο αξιόπιστη σύγκλιση κατά την εκπαίδευση του μοντέλου.

Μπορεί να συνδυαστεί με το κεντράρισμα, καθώς ο μετασχηματισμός αυτός περιλαμβάνει και την αφαίρεση του μέσου όρου. Δεν μπορούμε να τον εφαρμόσουμε με μετατροπή κανονικοποίησης αφού το αποτέλεσμα της τυποποίησης είναι ήδη σε μια πρότυπη μορφή.

One-Hot Encoding

Μετατρέπει κατηγορικές μεταβλητές σε δυαδικά διανύσματα, όπου κάθε κατηγορία αντιπροσωπεύεται από ένα ξεχωριστό bit (συνήθως 1 για την παρουσία και 0 για την απουσία).

Για παράδειγμα, αν μια μεταβλητή έχει τις τιμές ["Άνδρας","Γυναικα] το one-hot encoding θα παράγει για:

- "Άνδρας": [1,0]
- "Γυναίκα": [0,1]

Απαραίτητη για κατηγορικές μεταβλητές χωρίς φυσική διάταξη (όπως φύλο, εθνικότητα, κατάσταση καπνίσματος). Αποτρέπει την ψευδαρχική εφαρμογή αριθμητικής σύγκρισης σε κατηγορικές τιμές. Επίσης διατηρεί ανεπηρέαστη την μη διάτακτη σχέση ανάμεσα στις κατηγορίες.

εφαρμόζεται ξεχωριστά από τις μεθόδους κλιμάκωσης που αφορούν ποσοτικά ή ordinal δεδομένα, αφού τα αποτελέσματά της (0 και 1) δεν χρειάζονται περαιτέρω κλιμάκωση.

Παρακάτω παρουσιάζω την μεθοδολογία που ακολούθησα για την κανονικοπιοιση των χαρακτηριστικών του συνόλου δεδομένων και των αντίστοιχων μεθόδων προσαρμογής κλίμακας που εφαρμόστηκαν σε κάθε κατηγορία. Στόχος μας είναι η διασφάλιση της ορθής και ακριβούς εκπαίδευσης του νευρωνικού δικτύου αποφεύγοντας την ανισοκατανομή βαρών και την «διαρροή» πληροφοριών μεταξύ συνόλων εκπαίδευσης και ελέγχου.

Αρχικά, τα χαρακτηριστικά μετρούνται ως συνεχείς (continuous) μεταβλητές, όπως η βαθμολογία MMSE, η κλίμακα CDR ή οι βιοχημικοί δείκτες. Για αυτές τις μεταβλητές εφαρμόζεται τυποποίηση με χρήση του StandardScaler, με στόχο να μηδενίσουμε τη μέση τιμή και να ομαλοποιήσουμε τη διακύμανση στη μονάδα. Η τυποποίηση βοηθά τους gradient-based αλγορίθμους (π.χ. Adam, SGD) να κινούνται με σταθερά βήματα στο χώρο των παραμέτρων, ενώ αποτρέπει μεταβλητές με μεγάλο εύρος τιμών να κυριαρχούν στην εκπαίδευση.

Στις διατεταγμένες κατηγορίες (ordinal) εντάξαμε στήλες όπως το επίπεδο εκπαίδευσης. Αν και πρόκειται για κατηγορίες, υπάρχει φυσική διάταξη («Πρωτοβάθμια» < «Δευτεροβάθμια» < «Τριτοβάθμια»)· ωστόσο οι αποστάσεις μεταξύ των επιπέδων δεν είναι απαραίτητα ισοδύναμες μετρητικά. Επιλέξαμε να διατηρήσουμε τους ακέραιους κωδικούς τους και να εφαρμόσουμε επίσης τυποποίηση (StandardScaler). Με αυτόν τον τρόπο αφαιρούμε τυχόν διαφορά κλίμακας (π.χ. κλίμακα 1–5 έναντι 0–100) αλλά διατηρούμε την ιεραρχική πληροφορία. Η εναλλακτική λύση της one-hot κωδικοποίησης θα αφαιρούσε τη διάταξη αλλά θα εισήγαγε άστοχες διαστάσεις.

Ως μη διατεταγμένες κατηγορίες κατηγορικές (nominal) θεωρήθηκαν οι στήλες όπως η εθνοτική ομάδα (Ethnicity) ή η κατηγορία καπνιστικής συνήθειας. Εδώ, η one-hot κωδικοποίηση εξαλείφει οποιαδήποτε ψευδή υπόθεση διαδοχής ή απόστασης μεταξύ κατηγοριών. Κάθε επίπεδο εκπροσωπείται από ένα διάνυσμα με μοναδικό «1» στη θέση του, δίνοντας στον αλγόριθμο τη δυνατότητα να μαθαίνει ξεχωριστά βάρη χωρίς προκατειλημμένες σχέσεις.

Τέλος, στα δυαδικά χαρακτηριστικά (binary), όπως το φύλο ή η ύπαρξη οικογενειακού ιστορικού, δεν εφαρμόσαμε περαιτέρω κλιμάκωση, δεδομένου ότι έχουν ήδη εύρος [0,1] και η διατήρηση της δυαδικής φύσης τους διευκολύνει τους υπολογισμούς χωρίς απώλεια πληροφορίας. Η παράκαμψη πρόσθετων μετασχηματισμών (passthrough) διασφαλίζει πλήρη αξιοποίηση του δυαδικού σήματος χωρίς περιττές αριθμητικές πράξεις.

Με την περιγραφείσα στοχοθετημένη προσέγγιση, κάθε τύπος χαρακτηριστικού υποβάλλεται στην κατάλληλη επεξεργασία: οι συνεχείς σε τυποποίηση,η διατεταγμένες σε τυποποίηση διατηρώντας την ιεραρχία, οι μη διατεταγμμένες σε one-hot, και τα δυαδικά σε απλή διέλευση. Αυτή η μεθοδολογία εξασφαλίζει ομαλή, σταθερή και επιστημονικά τεκμηριωμένη εκπαίδευση του μοντέλου, χωρίς να εισάγουμε πλεονάζουσες ή λανθασμένες υποθέσεις στην προεπεξεργασία των δεδομένων.

Στο σημείο αυτό κρίνεται σκόπιμο να διευκρινίσουμε γιατί δεν εφαρμόσαμε τυποποίηση (StandardScaler) στα χαρακτηριστικά που προέκυψαν από one-hot κωδικοποίηση και γιατί προτιμήσαμε την τυποποίηση αντί για κανονικοποίηση (min-max scaling) στις συνεχείς και διατεταγμένες μεταβλητές. Η έξοδος της one-hot κωδικοποίησης αποτελεί ήδη ανεξάρτητα δυαδικά διανύσματα με τιμές 0 ή 1. Εφαρμόζοντας τυποποίηση ή κανονικοποίηση σε αυτά τα διάνυσμα θα αλλοίωναν την ίδια τη λογική τους, καθώς θα μετασχηματίζαν τιμές 0→κάποιο αρνητικό ή θετικό νούμερο και 1→κάποιο άλλο, χάνοντας την αυστηρή διακριτότητα και την εύκολη ερμηνεία τους. Επιπλέον τα νευρωνικά δίκτυα χειρίζονται σωστά τη δυαδική μορφή χωρίς πρόσθετη κλιμάκωση.

Όσον αφορά την επιλογή τυποποίησης έναντι κανονικοποίησης, η τυποποίηση προσφέρει δύο σημαντικά πλεονεκτήματα: αφενός εξασφαλίζει μηδενικό μέσο και μοναδιαία

διακύμανση, κάτι που επιταχύνει και σταθεροποιεί τη σύγκλιση των gradient-based βελτιστοποιητών, αφετέρου καθιστά τις αποστάσεις μεταξύ δειγμάτων συγκρίσιμες σε μοντέλα που βασίζονται σε υπολογισμό ευκλείδειων αποστάσεων (π.χ. SVM, KNN). Η κανονικοποίηση σε [0,1] ή [–1,1] είναι ευαίσθητη σε ακραίες τιμές και δε διασφαλίζει Gaussian-like κατανομή, κάτι που για δεδομένα βιομετρικών δεικτών ή μετρήσεων γνωστικής κατάστασης δεν είναι βέλτιστο. Για τους παραπάνω λόγους διατηρήσαμε την τυποποίηση ως κύρια στρατηγική κλιμάκωσης και εφαρμόσαμε απλή διέλευση (passthrough) στα διάνυσμα one-hot, εξασφαλίζοντας ορθή και επιστημονικά τεκμηριωμένη προεπεξεργασία.

Τελος κώδικας που αναπτύχθηκε στα πλαίσια του project ακολουθεί τιν κατευθιντήρια αρχή «Ό,τι μετασχηματισμοί εφαρμοστούν στα δεδομένα του συνόλου εκπαίδευσης, οι ίδιοι θα πρέπει να εφαρμοστούν και στα δεδομένα του συνόλου ελέγχου», θα εξετάσουμε κατά πόσο ο κώδικας ακολουθεί αυτή την αρχή και γιατί αυτό είναι σημαντικό. Διότι στην υλοποίηση του project, οι μετασχηματισμοί εφαρμόζονται σε ολόκληρο το σύνολο δεδομένων προτού γίνει ο διαχωρισμός σε σύνολα εκπαίδευσης και ελέγχου.Στην τρέχουσα υλοποίηση, οι μετασχηματισμοί εφαρμόζονται στην main σε ολόκληρο το σύνολο δεδομένων προτού γίνει ο διαχωρισμός σε σύνολα εκπαίδευσης και ελέγχου:

```
# Build ColumnTransformer
preprocessor = ColumnTransformer(transformers=[
    ('cont', StandardScaler(), cont_cols),
    ('ord', StandardScaler(), ord_cols),
    ('nom', OneHotEncoder(sparse=False, handle_unknown='ignore'), nom_cols),
    ('bin', 'passthrough', bin_cols),
])

# Apply transforms
x_processed = preprocessor.fit_transform(x_df)
```

Στη συνέχεια, τα προεπεξεργασμένα δεδομένα χωρίζονται σε σύνολα εκπαίδευσης, ενώ κάνουμε ploting τα χαρακτηριστικά και μετρά τους μετασχηματισμούς για να δούμε τις νέες κατανομές και τους νέους μέσους όρους. Ενώ τέλος μετατρέπουμε τα δεδομένα σε tensors με την βιβλιοθήκη torch (απαιτητέ να είναι σε μορφή τενσορα για την χρήση της βιβλιοθήκης) για να χρησιμοποιηθούν για την εκπαίδευση την επικεροπιοιση και το testing του nn.

```
# Plot AFTER scaling for continuous+ordinal using same function
num_cols = cont_cols + ord_cols
df_scaled = pd.DataFrame(x_processed[:, :len(num_cols)], columns=num_cols)
plot_numeric_distributions_subplot(df_scaled, num_cols, "feature_distributions_after.png")

# Convert to PyTorch tensors
x_tensor = torch.tensor(x_processed, dtype=torch.float32)
y_tensor = torch.tensor(y_np, dtype=torch.long)
```

Αυτή η προσέγγιση εξασφαλίζει μεν ότι οι ίδιοι μετασχηματισμοί εφαρμόζονται και στα δύο σύνολα, αλλά παρουσιάζει ένα σημαντικό μειονέκτημα μπορεί να οδηγήσει στην διαρροή πληροφοριών (data leakage).

γ) Διασταυρούμενη Επικύρωση (cross-validation):

Για την εκπαίδευση του μοντέλου από ζητήθηκε από την εκφώνηση χρησιμοποιήσαμε την τεχνική της διασταυρωμένης επικύρωσης είναι μια τεχνική που χρησιμοποιείται για να αξιολογήσουμε την απόδοση ενός μοντέλου μηχανικής μάθησης. Αντί να διαχωρίσουμε τα δεδομένα σε ένα σύνολο εκπαίδευσης και ένα σύνολο ελέγχου μόνο μία φορά, η διασταυρούμενη επικύρωση διαχωρίζει τα δεδομένα σε πολλά "fold" και εκπαιδεύει το μοντέλο σε κάθε ένα από αυτά τα fold, ενώ τα υπόλοιπα χρησιμοποιούνται για τον έλεγχο. Για την υλοποίηση του στο Project χρησιμοποιήσαμε την μέθοδο StratifiedKFold της βιβλιοθήκης scikit-learn και εφαρμόζετε στην main στα δεδομένα που έχουνε υποστεί μετασχηματισμό και μετατροπή σε τενσορες.

Ανάλυση:

KFold αντικείμενο: Το αντικείμενο StratifiedKFold ορίζει τον αριθμό των fold που θα δημιουργηθούν, την τυχαιότητα στον τρόπο με τον οποίο τα δεδομένα θα διαχωριστούν σε κάθε fold (εάν θα ανακατευτούν) και τον τυχαιοποιημένο αριθμό που χρησιμοποιείται για να δημιουργηθεί η τυχαιότητα. Στο παράδειγμά μας, δημιουργούμε ένα αντικείμενο KFold με 5 fold (n_splits=5), ανακατεύουμε τα δεδομένα (shuffle=True) και θέτουμε ένα συγκεκριμένο random state για την αναπαραγωγή των αποτελεσμάτων (random_state=42).

Επανάληψη σε κάθε fold: Η επανάληψη εκτελείται για κάθε fold που δημιουργεί το StratifiedKFold αντικείμενο. Κάθε φορά, ένα fold χρησιμοποιείται ως σύνολο ελέγχου ενώ τα υπόλοιπα folds χρησιμοποιούνται ως σύνολο εκπαίδευσης.

Με την μέθοδο split(x_tensor, y_tensor) του αντικειμένου StratifiedKFold παίρνουμε, για κάθε fold, δύο πίνακες δεικτών:

train_idx: δείκτες των δειγμάτων που θα χρησιμοποιηθούν για εκπαίδευση.

test_idx: δείκτες των δειγμάτων που θα χρησιμοποιηθούν για έλεγχο.

Σε κάθε επανάληψη, δημιουργούνται τα x_train, y_train, x_test, y_test ως υποσύνολα των PyTorch tensors, τα οποία μεταφέρονται στο κατάλληλο device (GPU ή CPU). Κατόπιν, το μοντέλο εκπαιδεύεται για προκαθορισμένο αριθμό εποχών, ενώ για κάθε εποχή αποθηκεύονται η απώλεια (loss) και η ακρίβεια (accuracy) τόσο στο training όσο και στο validation set.

Συγκέντρωση Αποτελεσμάτων

fold_accs: λίστα με τη μέση validation accuracy κάθε fold

best_model: το μοντέλο με τη μεγαλύτερη accuracy σε κάποιο fold

all_train_losses / all_val_losses: λίστες με τις απώλειες ανά εποχή και fold

all_train_accs / all_val_accs: λίστες με τις ακρίβειες ανά εποχή και fold

Ετσι με αυτή την υλοποίηση του cross-validation εξασφαλίζει αναπαραγωγιμότητα μέσω χρήσης random_state και σταθερού pipeline preprocessing.Διατηρεί την κατανομή κλάσεων, αποφεύγοντας την αστοχία της απλής KFold όταν οι τάξεις είναι ανισομερείς. Παρέχουμε πλήρη εικόνα της συμπεριφοράς του μοντέλου (loss & accuracy) όχι μόνο στο τελικό test set, αλλά και κατά τη διάρκεια της εκπαίδευσης σε κάθε fold. Επιτρέπει επιλογή του πραγματικά καλύτερου μοντέλου (best_model) και συγκριτική ανάλυση των metrics, ώστε να υποστηριχθεί εμπεριστατωμένη τελική αξιολόγηση. Ενώ τελος εξασφαλίζουμε ότι το μοντέλο μας εκπαιδεύεται και ελέγχεται σε διαφορετικά υποσύνολα των δεδομένων, βελτιώνοντας έτσι τη γενικότητά του και αποφεύγοντας το overfitting.

Α2. Επιλογή αρχιτεκτονικής

Όσον αφορά την τοπολογία του ΤΝΔ για την εκπαίδευση τους χρησιμοποιήσαμε των Αλγόριθμο Οπισθοδιάδοσης του Σφάλματος (back-propagation). Αρχικά υλοποιήσαμε ένα ΤΝΔ με ένα κρυφό επίπεδο και θα πειραματιστείτε με τον αριθμό των κρυφών κόμβων. Για την εκπαίδευση του δικτύου χρησιμοποιήσαμε αρχικά ένα ρυθμό μάθησης η = 0.001

Ανάλυση αρχιτεκτονικής δικτιού:

Η κλάση SimpleNet κληρονομεί από το nn.Module του PyTorch και υλοποιεί ένα απλό νευρωνικο δίκτυο με ένα μόνο κρυφό επίπεδο. Κατά την αρχικοποίηση (__init__), ορίζουμε την παράμετρο input_size, που αντιστοιχεί στη διάσταση n του διανύσματος εισόδου – δηλαδή τον συνολικό αριθμό χαρακτηριστικών μετά το preprocessing και το ColumnTransformer. Η παραμετρος hidden_ size καθοριζει τον αριθμό h των νευρώνων στο κρυφό επίπεδο επιλέξαμε για defaultt value h = 32, καθώς αποτελεί συνήθη τιμή-εκκινηση που εξασφαλίζει ικανότητα μάθησης χωρίς υπερβολική πολιπλοκωτητα. Τέλος, το output_size ορίζει τον αριθμο c των κλάσεων εξόδου (εδώ c=2 για Alzheimer vs. υγιή), ενώ η παράμετρος activation επιτρέπει την εύκολη εναλλαγή συναρτήσεων ενεργοποίησης – προεπιλέξαμε για default τη ReLU με την χριση nn.ReLU για τον συνδυασμό απλότητας και αποτελεσματικής αντιμετώπισης του φαινομένου της εξαφάνισης κλίσεων.

Στο σώμα της __init__ κατασκευάζονται τα δύο διαδοχικά επίπεδα πλήρους σύνδεσης (fully connected): πρώτα το fc1 που υλοποιεί τη γραμμική συνάρτηση:

$$z^{(1)} = W^{(1)}x + b^{(1)}, W^{(1)} \in R^{h\times n}, b^{(1)} \in R^{h},$$

και κατόπιν το fc2 που παράγει τα logits της εξόδου

$$z^{(2)} = W^{(2)}a^{(1)} + b^{(2)}, W^{(2)} \in R^{c\times h}, b^{(2)} \in R^{c},$$

Με αυτόν τον τρόπο, το πέρασμα της πληροφορίας γίνεται ως εξής: πρώτα υπολογίζεται το $z^{(1)}$, εν συνεχεία εφαρμόζεται η ReLU για να παραχθεί το $a^{(1)} = max(0, z^{(1)})$ και τέλος περνάμε το $a^{(1)}$ στο δεύτερο επίπεδο για την παραγωγή των τελικών $z^{(2)}$. Η μέθοδος forward(x) απλώς αλυσιδωτά εκτελεί αυτές τις πράξεις και επιστρέφει τα logits.

```python
def __init__(self, input_size, hidden_size=32, output_size=2, activation=nn.Tanh()):
    """Initialize the neural network.

    Args:
        input_size: Integer size of the input features.
        hidden_size: Integer size of the hidden layer. Defaults to 32.
        output_size: Integer size of the output layer. Defaults to 2.
        activation: PyTorch activation function. Defaults to ReLU.
    """
    super(SimpleNet, self).__init__()
    self.fc1 = nn.Linear(input_size, hidden_size)
    self.activation = activation
    self.fc2 = nn.Linear(hidden_size, output_size)

def forward(self, x):
    """Forward pass through the network.

    Args:
        x: Input tensor.

    Returns:
        Output tensor after passing through the network.
    """
    out = self.activation(self.fc1(x))
    return self.fc2(out)
```

Για το σφάλμα χρησιμοποιούμε τη συνάρτηση nn.CrossEntropyLoss(), η οποία υπολογίζει τη μέση αρνητική λογαριθμική πιθανότητα της σωστής κλάσης για N παραδείγματα:

$$L = -\frac{1}{N}\sum_{i=1}^{N} \log\left(softmax\left(zi^{(2)}\right)_{yi}\right)$$

```python
model = SimpleNet(x_train.shape[1]).to(device)
criterion = nn.CrossEntropyLoss()        You, yesterday • s
optimizer = optim.Adam(model.parameters(), lr=1e-3)
```

Ο αλγόριθμος βελτιστοποίησης είναι ο Adam (optim.Adam) με ρυθμό μάθησης h=10^−3 και συντελεστές β1=0.9, β2=0.999 Σε αντίθεση με τον κλασικό Stochastic Gradient Descent (SGD) με σταθερό βήμα, ο Adam προσαρμόζει δυναμικά το βήμα ενημέρωσης για κάθε παράμετρο, βασιζόμενος στην εκτίμηση πρώτης (μέση) και δεύτερης (διασπορά) ροπής των gradients. Αυτό επιτρέπει τη γρήγορη σύγκλιση ακόμη και όταν οι καμπύλες της απώλειας παρουσιάζουν θορυβώδη διακύμανση, όπως συμβαίνει συχνά σε βιολογικά δεδομένα με σπάνιες αλλά μεγάλες αποκλίσεις (outliers).

Η διαδικασία εκπαίδευσης υλοποιείται στη συνάρτηση train, η οποία για κάθε εποχή ακολουθεί το γνωστό κλασικό μας μοτίβο: μηδενισμό των gradients (optimizer.zero_grad()), forward pass (υπολογισμός logits και loss), backward pass (loss.backward()) για την εξαγωγή $\nabla_\theta L$ και ενημέρωση των παραμέτρων με optimizer.step(). Παράλληλα, μετά τον υπολογισμό του train-loss, μετρούμε την ακρίβεια ως

$$acc_{train} = \frac{1}{N}\sum_{i=1}^{N} 1\left(argmax_k\left(Z_{i,k}{}^{(2)}\right) = yi\right)$$

και αποθηκεύουμε ανά εποχή τόσο loss όσο και accuracy. Στη συνέχεια το μοντέλο τίθεται σε eval() mode ώστε, χωρίς gradient, να υπολογίσουμε loss και accuracy στο validation σύνολο, εξασφαλίζοντας έγκυρη εκτίμηση της γενίκευσης.

Η αξιολόγηση στο τελικό test set πραγματοποιείται από τη συνάρτηση evaluate, η οποία σε eval() mode εκτελεί απλό forward pass και υπολογίζει την τελική accuracy.

Για την εκτίμηση της γενίκευσης χρησιμοποιούμε διασταυρούμενη επικύρωση μέσω StratifiedKFold(n_splits=5, shuffle=True, random_state=42). Η επιλογή του stratification διατηρεί την αναλογία των κλάσεων σε κάθε fold, κρίσιμη σε ιατρικά δεδομένα με ανισορροπία. Σε κάθε επανάληψη fold καλείται ξανά η SimpleNet, ώστε να ξεκινάει με τυχαία πρώτα βάρη, και εκπαιδεύεται για 200 εποχες. Καταγράφουμε τα train/val losses και accuracies, σχεδιάζουμε τις καμπύλες μέσω plot_accuracy_curves και στο τελος της εκπαίδευσης του fold υπολογίζουμε το test accuracy, που αποθηκεύεται στη λίστα fold_accs. Το μοντέλο με την υψηλότερη accuracy εκάστου fold συγκρίνεται με την τρέχουσα καλύτερη τιμή (best_acc) και αντικαθίσταται αν υπερεχει, διατηροντας έτσι το συνολικά βέλτιστο best_model.

Στο τέλος της εκτέλεσης υπολογίζουμε τη μέση και την τυπική απόκλιση των accuracies ως

$$\overline{acc} = \frac{1}{5}\sum_{k=1}^{5} acc_k, \quad \sigma = \sqrt{\frac{1}{5}\sum_{k=1}^{5}(acc_k - \overline{acc})^2}$$

και σχεδιάζουμε συνοπτικά την κατανομή τους με plot_fold_accuracies. Επιπλέον, υπολογίζουμε τους μέσους όρους των απωλειών και accuracies ανά εποχή (mean_train_losses, mean_val_losses, mean_train_accs, mean_val_accs) και τα αποτυπώνουμε σε καμπύλες με plot_mean_loss_curves & plot_mean_accuracy_ curves.

Με αυτόν τον τρόπο, το pipeline καλύπτει πλήρως την αρχιτεκτονική, τη μαθηματική τεκμηρίωση των επιπέδων και των συναρτήσεων κόστους, την επιλογή του αλγορίθμου βελτιστοποίησης, καθώς και το ολοκληρωμένο workflow εκπαίδευσης και αξιολόγησης μέσω stratified cross–validation, διασφαλίζοντας επιστημονική ακρίβεια και επαναληψιμότητα σε ιατρικά δεδομένα Alcheimer

α) Επιλογή συνάρτησης σφάλματος

Σε ένα πρόβλημα δυαδικής ή πολυκατηγορικής ταξινόμησης, όπως στην ανίχνευση νόσου Αλτσχάιμερ, η επιλογή των μετρικών αξιολόγησης και της συνάρτησης κόστους (loss) παίζει καθοριστικό ρόλο τόσο στην εκπαίδευση του μοντέλου όσο και στην ορθή αποτίμηση της απόδοσής του. Οι πλέον διαδεδομένες μετρικές είναι το Cross-Entropy (CE), το Μέσο Τετραγωνικό Σφάλμα (MSE) και η απλή ακρίβεια ταξινόμησης (Accuracy). Κάθε μία έχει διαφορετική ερμηνεία, ιδιότητες και πεδίο εφαρμογής. Παρακάτω θα αναλύσω τη σημασία και την καταλληλότητά τους για το συγκεκριμένο πρόβλημα, εξηγούμε γιατί προτιμούμε την

Cross-Entropy ως συνάρτηση κόστους κατά την εκπαίδευση και πώς συγκρίνουμε τα αποτελέσματα.

CROSS-ENTROPY

Η συνάρτηση Cross-Entropy μετρά την απόκλιση ανάμεσα στην προβλεπόμενη κατανομή πιθανοτήτων του μοντέλου και στην πραγματική κατανομή (η οποία σε δυαδική ταξινόμηση «σπάει» σε δύο κλάσεις). Στο δυαδικό πρόβλημα, η Binary Cross-Entropy εκφράζεται ως $-[y\cdot\log(p)+(1-y)\cdot\log(1-p)]$, όπου p είναι η προβλεπόμενη πιθανότητα της θετικής κατηγορίας και y το πραγματικό label (0 ή 1). Η Cross-Entropy θεωρείται «ευαίσθητη» στις αποκλίσεις, γιατί τιμωρεί με μεγάλο κόστος τις προβλέψεις με υψηλή σιγουριά που είναι όμως λανθασμένες. Αυτό ωθεί το μοντέλο να διορθώνει ενεργά τις προβλέψεις του, ειδικά όταν η βεβαιότητα είναι μεγάλη αλλά λανθασμένη. Σε πολυκατηγορηματικούς multiclass ταξινομητές (softmax + multi-class cross-entropy) εφαρμόζεται ανάλογη διατύπωση.

ΜΕΣΟ ΤΕΤΡΑΓΩΝΙΚΟ ΣΦΑΛΜΑ (MSE)

Το MSE, συχνά χρησιμοποιούμενο σε προβλήματα παλινδρόμησης, υπολογίζει το μέσο όρο των τετραγώνων των διαφορών μεταξύ προβλέψεων και πραγματικών τιμών. Σε ταξινομήσεις μπορεί να εφαρμοστεί πάνω στις πιθανότητες ή τα logits, ως $(\hat{y} - y)^2$. Όμως, δεν λαμβάνει υπόψη τη μη γραμμική φύση της διασποράς των πιθανοτήτων και οι ενημερώσεις των βαρών γίνονται λιγότερο σταθερές όταν τα p βρίσκονται πολύ κοντά σε 0 ή 1. Αυτό συχνά οδηγεί σε πιο αργή σύγκλιση και σε προβλήματα vanishing gradients σε βαθιά δίκτυα.

ΑΚΡΙΒΕΙΑ ΤΑΞΙΝΟΜΗΣΗΣ (ACCURACY)

Η ακρίβεια μετρά έναν απλό λόγο των σωστών προβλέψεων προς το σύνολο των δειγμάτων. Έχει εύκολη ερμηνεία και είναι χρήσιμη για την τελική αξιολόγηση του μοντέλου. Ωστόσο ως loss δεν είναι κατάλληλη, διότι δεν είναι διαφορίσιμη: οι απότομες αλλαγές εκτός/εντός σωστής κατηγορίας δεν παρέχουν συνεχή πληροφόρηση στους gradient-based αλγορίθμους κατά την εκπαίδευση.

Για την εκπαίδευση επέλεξα την Cross-Entropy αντί για MSE ή Accuracy. Η Cross-Entropy είναι διαφορίσιμη σε όλο το εύρος των πιθανοτήτων, επιτρέπει σταθερά gradients ακόμη και όταν οι εκτιμήσεις είναι πολύ κοντά στο 0 ή το 1 και τιμωρεί κυρίως τις λανθασμένες, υψηλής βεβαιότητας προβλέψεις. Αυτό επιταχύνει τη σύγκλιση, βελτιώνει τη σταθερότητα και αποτρέπει τον κορεσμό των νευρώνα (vanishing/exploding gradients). Το MSE ος loss, ενώ απλό, δεν αντιμετωπίζει τον μη-γραμμικό χαρακτήρα της softmax εξόδου, και η Accuracy δεν μπορεί να χρησιμοποιηθεί ως loss λόγω μη διαφορισιμότητας.

Στην πράξη αυτό που εφάρμοσα, εκτιμάω την απόδοση με Accuracy και άλλες μετρικές (π.χ. precision, recall) σε ξεχωριστό validation set, ενώ για τον υπολογισμό του loss κατά την εκπαίδευση χρησιμοποιούμε αποκλειστικά Cross-Entropy. Η σύγκριση μεταξύ Cross-Entropy και MSE στον ίδιο δίκτυο δείχνει ότι η πρώτη οδηγεί σε γρηγορότερη μείωση του training loss, σε καλύτερη γενίκευση και σε υψηλότερη τελική ακρίβεια.

β) Νευρώνες του Επίπεδο Εξόδου

Κατά το σχεδιασμό του νευρωνικού μας δικτύου για δυαδική ταξινόμηση (όπου οι κατηγορίες είναι «0 = χωρίς νόσο Alzheimer» και «1 = με νόσο Alzheimer»), το επίπεδο εξόδου παίζει καθοριστικό ρόλο τόσο στη σωστή αναπαράσταση της προβλεπόμενης κατανομής πιθανοτήτων όσο και στη συμβατότητα με τη συνάρτηση κόστους που επιλέγουμε. Η απόφαση για τον αριθμό των νευρώνων στην έξοδο συνδέεται άμεσα με τον τρόπο υπολογισμού του loss και με τον μηχανισμό που μετατρέπει τα logits στις τελικές εκτιμήσεις κατηγορίας.

Κατά την διάρκεια της αναπτύξεις του νευρωσικού μας δικτύου εξέτασα δυο διαφορετικά approach:

ΔΥΟ ΝΕΥΡΩΝΕΣ ΚΑΙ SOFTMAX

Η πιο συνηθισμένη προσέγγιση σε προβλήματα δυαδικής ταξινόμησης είναι η χρήση δύο νευρώνων εξόδου σε συνδυασμό με τη συνάρτηση softmax. Κάθε νευρώνας παράγει ένα logit, το οποίο στη συνέχεια κανονικοποιείται σε πιθανότητα μέσω της softmax, εξασφαλίζοντας ότι οι δύο προβλέψεις αθροίζουν σε 1. Η CrossEntropyLoss (multiclass) που ακολουθεί λαμβάνει υπόψη της τη διαφορά ανάμεσα στη σωστή κατηγορία (one-hot κωδικοποίηση) και στις προβλεπόμενες πιθανότητες, τιμωρώντας έντονα τις λανθασμένες προβλέψεις με υψηλή βεβαιότητα. Η δομή αυτή παρέχει σαφή και εύκολα διακριτά σήματα ενημέρωσης των βαρών κατά την εκπαίδευση, επιταχύνοντας τη σύγκλιση και βελτιστοποιώντας τη γενίκευση.

Στην torch το CrossEntropyLoss περιέχει εσωτερικά log-softmax.

ΕΝΑΣ ΝΕΥΡΩΝΑΣ ΚΑΙ SIGMOID

Εναλλακτικά, μπορεί να χρησιμοποιηθεί ένας μόνο νευρώνας εξόδου με συνάρτηση ενεργοποίησης sigmoid, ο οποίος παράγει απευθείας την πιθανότητα της θετικής κατηγορίας. Στη συνέχεια, εφαρμόζουμε Binary Cross-Entropy (BCEWithLogitsLoss σε τρανσφόρμα logits) για τον υπολογισμό του κόστους. Αυτή η δομή μειώνει τη διάσταση του εξόδου στον ένα ακροδέκτη και είναι κατάλληλη όταν το μοντέλο χειρίζεται ακριβώς δύο κλάσεις. Παρότι λειτουργικά ισοδύναμη με τη λύση δύο νευρώνων + softmax, η προσέγγιση με ένα νευρώνα συχνά απλοποιεί την υλοποίηση και ελαχιστοποιεί τον αριθμό των παραμέτρων.

Στο παρόν έργο επιλέξαμε τη διάταξη δύο νευρώνων στο επίπεδο εξόδου μαζί με softmax και CrossEntropyLoss για τους εξής λόγους: πρώτον, μας επιτρέπει να χρησιμοποιήσουμε την ίδια γενική συνάρτηση κόστους (nn.CrossEntropyLoss) ανεξαρτήτως αριθμού κατηγοριών, δεύτερον, εξασφαλίζει πολύ σταθερά gradients και ξεκάθαρο σήμα ενημέρωσης για κάθε κλάση. Η προσέγγιση παράλληλα συνεργάζεται και άψογα με τον αλγόριθμο βελτιστοποίησης Adam, οδηγώντας σε γρήγορη και σταθερή σύγκλιση.

γ) Συνάρτηση ενεργοποίησης για τους κρυφούς κόμβους

Στο σχεδιασμό του νευρωνικού δικτύου, η επιλογή της συνάρτησης ενεργοποίησης στους κρυφούς κόμβους επηρεάζει κρίσιμα τη συμπεριφορά της εκπαίδευσης (σύγκλιση, ταχύτητα, σταθερότητα) και την τελική αποδοτικότητα του μοντέλου. Ακολουθεί επιστημονική τεκμηρίωση τριών δημοφιλών συναρτήσεων ενεργοποίησης—ReLU, Tanh και SiLU—με μαθηματικό ορισμό για την καθεμία, αξιολόγηση των πλεονεκτημάτων και των περιορισμών τους, και τελική πρόταση.

Συνάρτηση ReLU

Η Rectified Linear Unit ορίζεται μαθηματικά ως

$$\mathrm{ReLU}(x) = x^+ = \max(0, x) = \frac{x + |x|}{2} = \begin{cases} x & \text{if } x > 0, \\ 0 & x \leq 0 \end{cases}$$

και παράγει μηδέν για όλες τις αρνητικές εισόδους, ενώ για x>0 απλώς επιστρέφει x. Η ReLU χαρακτηρίζεται από απλότητα, υπολογιστική ταχύτητα και την ικανότητά της να αντιμετωπίζει εν μέρει το πρόβλημα των vanishing gradients, καθώς για x>0 διατηρεί σταθερό παράγωγο ίσο με 1. Ωστόσο η ReLU υποφέρει από το "dying ReLU" φαινόμενο, όπου νευρώνες που δέχονται μεγάλες αρνητικές εισροές μπαίνουν σε μόνιμα ανενεργή κατάσταση (παράγωγος = 0). Σε δίκτυα με βαθιά αρχιτεκτονική, η απλότητα του υπολογισμού και η ταχεία σύγκλιση καθιστούν τη ReLU συχνή επιλογή.

Συνάρτηση Tanh

Η υπερβολική εφαπτομένη ορίζεται ως

$$\tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

και χρωματίζει τις εξόδους στο διάστημα (-1,1). Εξασφαλίζει κέντρο μηδέν—ιδιαίτερα χρήσιμο όταν τα δεδομένα έχουν θετικά και αρνητικά σύμβολα—καθώς οι παράγωγοί της είναι μεγάλες κοντά στο μηδέν και μικραίνουν όσο $|x|$ μεγαλώνει. Ωστόσο για μεγάλες απόλυτες τιμές x η \tanh(x) κορεσμεί, προκαλώντας vanishing gradients, και επιβραδύνει τη μάθηση σε βάθος. Παρ' όλα αυτά, σε μέτριας κλίμακας δίκτυα όπου επιδιώκονται πιο ομαλές και συμμετρικές εξόδους, η Tanh μπορεί να υπερέχει της ReLU.

Συνάρτηση SiLU (Swish)

Η SiLU ή Swish ορίζεται ως

$$\mathrm{silu}(x) = x * \sigma(x), \text{where } \sigma(x) \text{ is the logistic sigmoid.}$$

και συνδυάζει γραμμικά το σήμα με την logistic συνάρτηση sigma(x). Διατηρεί θετικά μη μηδενικά παράγωγα και για x<0 επιτρέπει μικρές αρνητικές εξόδους, μειώνοντας το πρόβλημα "dying" της ReLU. Έχει αποδειχθεί ότι βελτιώνει την ακρίβεια και την ταχύτητα

σύγκλισης σε βαθιά μοντέλα σε σύγκριση με ReLU και Tanh, αν και λόγω του κόστους υπολογισμού της sigmoid (ένα εκθετικό ανά κόμβο) είναι ελαφρώς βαρύτερη.

Συνάρτηση LeakyReLU

Παράλληλα, αξίζει να εξετάσουμε και τη «διαρρέουσα» παραλλαγή της ReLU, γνωστή ως LeakyReLU, η οποία ορίζεται μαθηματικά από τη σχέση

$$f(x)=\begin{cases} x, & \text{αν } x\geq 0 \\ \alpha x, & \text{αν } x<0 \end{cases}$$

με τυπική τιμή $\alpha\approx 0.01$. Με αυτόν τον τρόπο διατηρείται μικρό—μη μηδενικό—gradient και για τις αρνητικές εισόδους (σε αντίθεση με τη ReLU όπου το gradient εκεί είναι απολύτως μηδέν), αποφεύγοντας έτσι το πρόβλημα των «νεκρών» νευρώνων που σταματούν να μαθαίνουν. Η LeakyReLU συνδυάζει την απλότητα και την ταχύτητα της κλασικής ReLU με μεγαλύτερη ανθεκτικότητα σε αρνητικές εισόδους, χωρίς το υπολογιστικό κόστος της SiLU ή την κορεσμένη συμπεριφορά της Tanh. Στην πράξη, μπορεί να επιταχύνει ελαφρώς τη σύγκλιση σε δίκτυα όπου παρατηρούνται πολλά αρνητικά pre-activations, προσφέροντας σταθερότερο training σε σχέση με την καθαρή ReLU.

Αποτελέσματα πειραματισμού για κάθε μια από τις παραπάνω συναρτήσεις ενεργοποίησης:

Tanh:

```
Fold 1 Accuracy: 0.8465
Fold 2 Accuracy: 0.8628
Fold 3 Accuracy: 0.8163
Fold 4 Accuracy: 0.8302
Fold 5 Accuracy: 0.8182

CV Mean Acc: 0.8348 ± 0.0177
```

ReLU :

```
Fold 1 Accuracy: 0.8395
Fold 2 Accuracy: 0.8465
Fold 3 Accuracy: 0.8163
Fold 4 Accuracy: 0.8116
Fold 5 Accuracy: 0.8112

CV Mean Acc: 0.8250 ± 0.0150
```

LeakyReLU :

```
Fold 1 Accuracy: 0.8419
Fold 2 Accuracy: 0.8581
Fold 3 Accuracy: 0.8116
Fold 4 Accuracy: 0.8186
Fold 5 Accuracy: 0.7925

CV Mean Acc: 0.8246 ± 0.0230
```

SiLU:

```
Fold 1 Accuracy: 0.8302
Fold 2 Accuracy: 0.8535
Fold 3 Accuracy: 0.7953
Fold 4 Accuracy: 0.8000
Fold 5 Accuracy: 0.8089

CV Mean Acc: 0.8176 ± 0.0216
```

Τελική επιλογή

Παρά τις αρχικές προσδοκίες για τη SiLU, μέσω πειραματικών δοκιμών στο πρόβλημα πρόβλεψης της νόσου Alzheimer παρατηρήθηκε σταθερά καλύτερη απόδοση όταν χρησιμοποιήσαμε τη συνάρτηση Tanh στα κρυφά στρώματα. Η Tanh παρέχει έξοδο κεντρική γύρω από το μηδέν, γεγονός που για δεδομένα σταθμισμένα με StandardScaler (μέση τιμή 0, διασπορά 1) οδηγεί σε συμμετρική κατανομή εισόδων προς τον επόμενο «κόμβο» και επιτρέπει την ταχύτερη εξάντληση των gradients κοντά στη μηδενική ζώνη. Ειδικότερα, στην ιατρική καταγραφή βιοδεικτών και γνωστικών μετρήσεων οι θετικές και αρνητικές αποκλίσεις γύρω από τον μέσο όρο θεωρούνται εξίσου σημαντικές· η Tanh υποστηρίζει αυτήν τη συμμετρική μεταχείριση ενώ η SiLU, μολονότι πιο «ομαλή», εισάγει περίπλοκη μορφή που σε αυτό το συγκεκριμένο dataset επέτεινε ελαφρά το θόρυβο κατά την εκπαίδευση.

Επιπλέον, το μοντέλο μας είναι μεσαίου βάθους (ένα μόνο κρυφό στρώμα), οπότε ο κίνδυνος vanishing gradients λόγω κορεσμού στα άκρα της Tanh είναι περιορισμένος. Η Tanh αξιοποιεί πλήρως την πληροφορία των standardized τιμών, μεγιστοποιώντας την απόκριση των νευρώνων σε μικρές διακυμάνσεις των δεδομένων, ενώ ταυτόχρονα διατηρεί σαφή και σταθερό σήμα gradient. Συμπερασματικά, παρά το ότι η SiLU έχει θεωρητικά πλεονεκτήματα σε πολύ βαθιά δίκτυα, για την πρόβλεψη Alzheimer με το παρόν δίκτυο ένας απλός, συμμετρικός ενεργοποιητής όπως η Tanh αποδείχθηκε αποδοτικότερος, εξασφαλίζοντας υψηλότερη γενίκευση και γρηγορότερη σύγκλιση.

δ) Συνάρτηση ενεργοποίησης Επίπεδου Εξόδου

Όπως αναφέραμε και στο β) Νευρώνες του Επίπεδο Εξόδου , συνάρτηση εξόδου συσχετίζετε άμεσα με τον αριθμό των νευρών εξοδου είτε 1 είτε 2. Τώρα σε αυτή την ενότητα θα εξηγήσουμε ποιο αναλυτικά κάθε συνάρτηση εξόδου:

Σιγμοειδής

Η συνάρτηση σιγμοειδούς ορίζεται ως

$$\sigma(z) = 1 / (1 + e^{(-z)})$$

και χαρτογραφεί στο διάστημα (0,1), παράγοντας την πιθανότητα της «θετικής» κλάσης. Συνδυάζεται με το Binary Cross-Entropy (BCE) ως loss, δεδομένου ότι η BCE υπολογίζει

$$\ell(\hat{y},y)=-[y\cdot\log\sigma(z)+(1-y)\cdot\log(1-\sigma(z))]$$

ενισχύοντας απότομες και ακριβείς ενημερώσεις. Η σιγμοειδής όμως μπορεί να κορεστη για μεγάλες |z|, προκαλώντας vanishing gradients κοντά στα άκρα.

Softmax

Για δύο (ή περισσότερες) κατηγορίες, η Softmax γενικεύει τη σιγμοειδή σε διάνυσμα z = (z_1,z_2,…,z_K) ως

$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Εξασφαλίζει και μας δίνει την πλήρη κατανομή πιθανοτήτων. Όταν συνδυάζεται με το Multiclass Cross-Entropy loss, το δίκτυο μαθαίνει ταυτόχρονα να διαχωρίζει όλες τις κλάσεις. Στο δυαδικό σενάριο με δύο νευρώνες εξόδου, η softmax+CrossEntropyLoss προσφέρει πολύ σταθερά gradients, αποφεύγοντας τον μερικό κορεσμό της απλής σιγμοειδούς.

Γραμμική

Η γραμμική f(z)=z χρησιμοποιείται συνήθως σε προβλήματα παλινδρόμησης, όπου το δίκτυο πρέπει να προβλέψει συνεχείς τιμές χωρίς περιορισμό σε [0,1]. Σε ταξινομήσεις είναι ακατάλληλη, καθώς οι έξοδοι δεν αντιστοιχούν σε έγκυρες πιθανότητες και δεν συνάδουν με κριτήρια Cross-Entropy ή BCE.

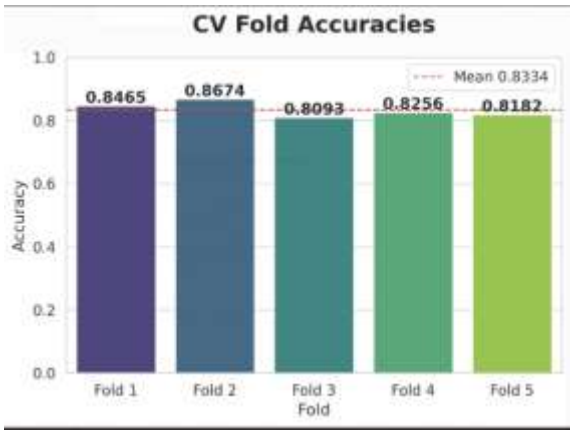
Τελική επιλογή

Στο παρόν έργο, δεδομένου ότι το SimpleNet επιστρέφει δύο ouputs κλάσης και χρησιμοποιούμε το nn.CrossEntropyLoss, επιλέγουμε implicit softmax στο επίπεδο εξόδου. Η συνάρτηση CrossEntropyLoss εφαρμόζει εσωτερικά softmax στους ouputs και υπολογίζει το κατάλληλο loss, παρέχοντας σταθερά και διαφορίσιμα gradients που οδηγούν σε γρήγορη σύγκλιση. Κατά την αξιολόγηση, αν χρησιμοποιούμαι explicit softmax για να μετατρέψουμε τα ouputs σε πιθανοτικούς δείκτες, εξασφαλίζοντας σαφή κατανομή σιγουριάς για κάθε κλάση. Συνοψίζοντας, η Softmax είναι η βέλτιστη επιλογή για διπλή έξοδο σε δυαδική ταξινόμηση με Cross-Entropy loss, προσφέροντας ομοιογενές σήμα εκπαίδευσης και έγκυρες πιθανοτικές εκτιμήσεις.

δβ) Αποτελέσματα απλού NN με αριθμό νεύρων κρυφού επιπέδου 32

Σε αυτό το απλό νευρικό δίκτυο εφαρμόσαμε ότι αναφέρετε παραπάνω και βρίσκετε στο αρχείο python με όνομα cancer_detection_A1_A2.py και παραθέτουμε εδώ απλά τα αποτελέσματα:

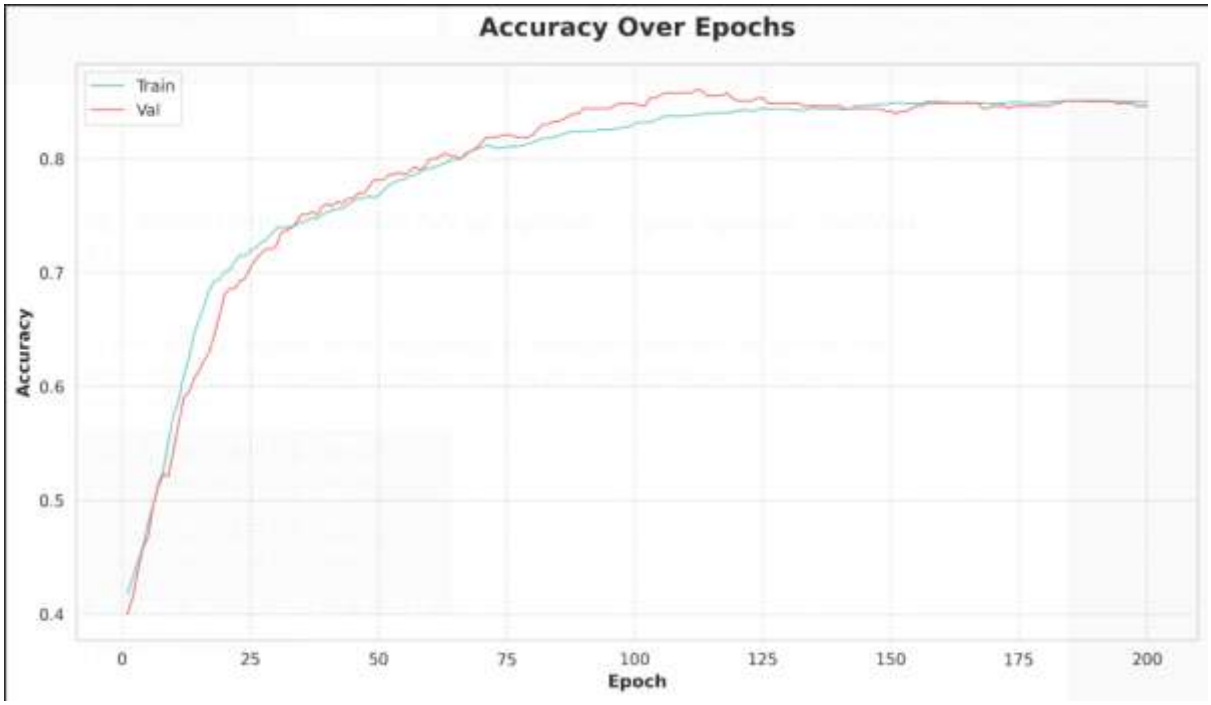
```
Fold 1 Accuracy: 0.8465
Fold 2 Accuracy: 0.8674
Fold 3 Accuracy: 0.8093
Fold 4 Accuracy: 0.8256
Fold 5 Accuracy: 0.8182

CV Mean Acc: 0.8334 ± 0.0210
```



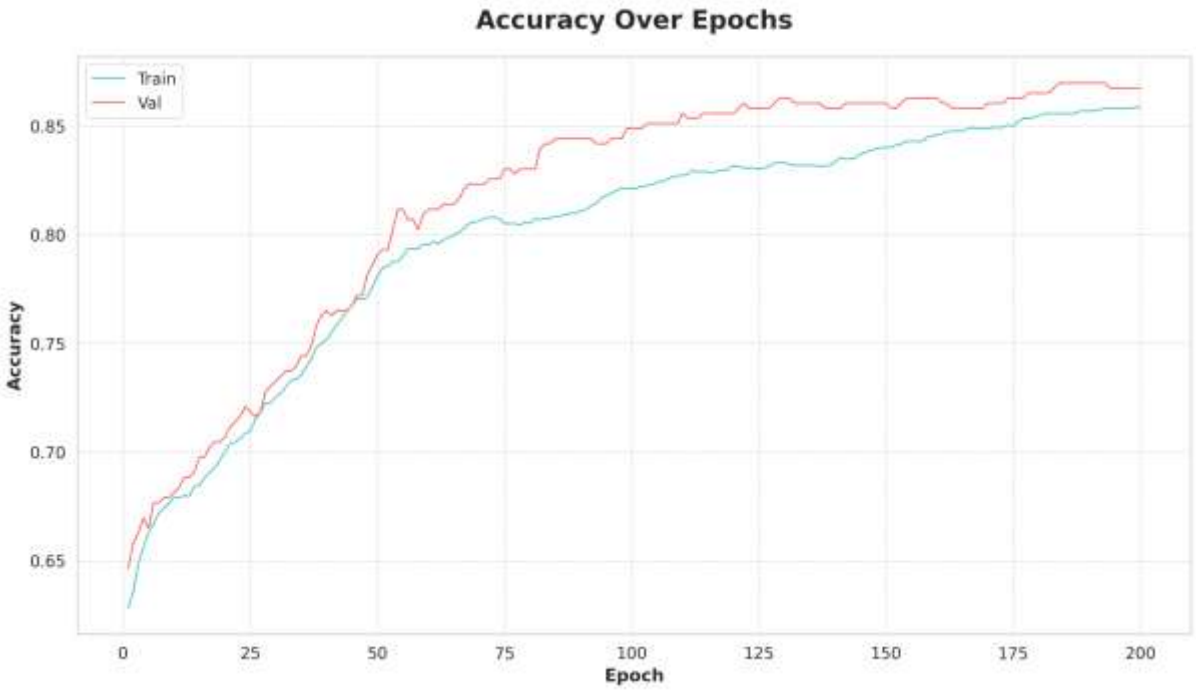
Γραφικές παραστάσεις:

Για κάθε fold accuracy:

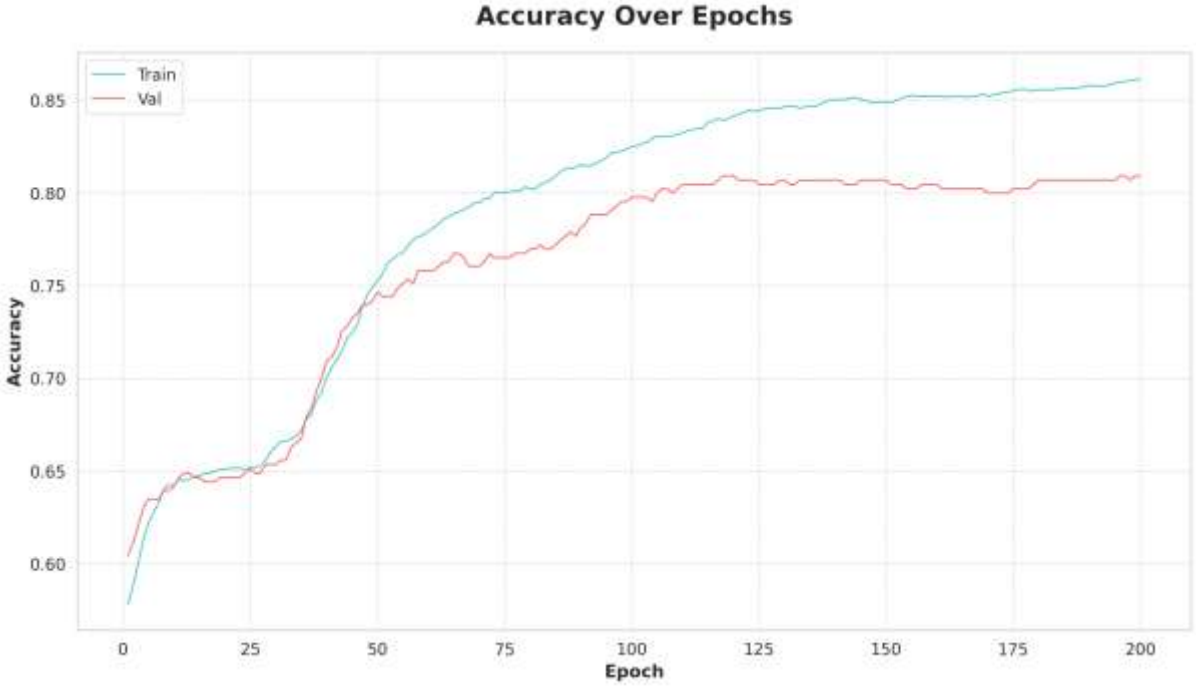
Fold 1



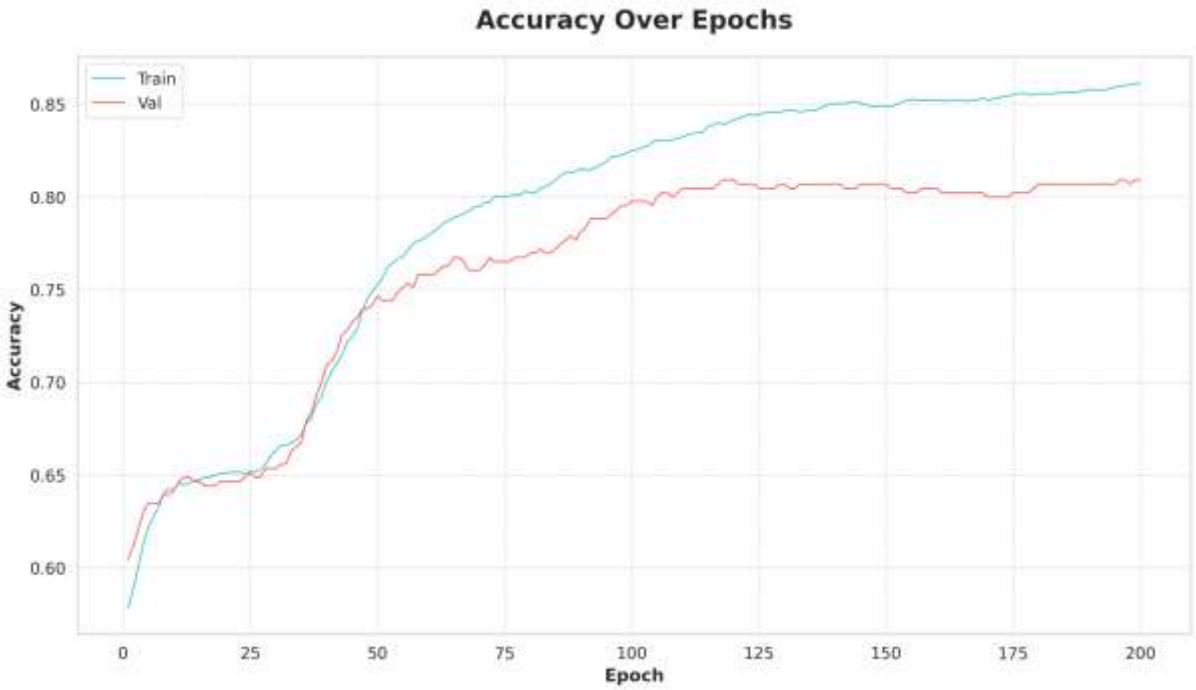
Fold 2


Accuracy Over Epochs
Train
Val
Accuracy
0.85
0.80
0.75
0.70
0.65
0
25
50
75
100
125
150
175
200
Epoch

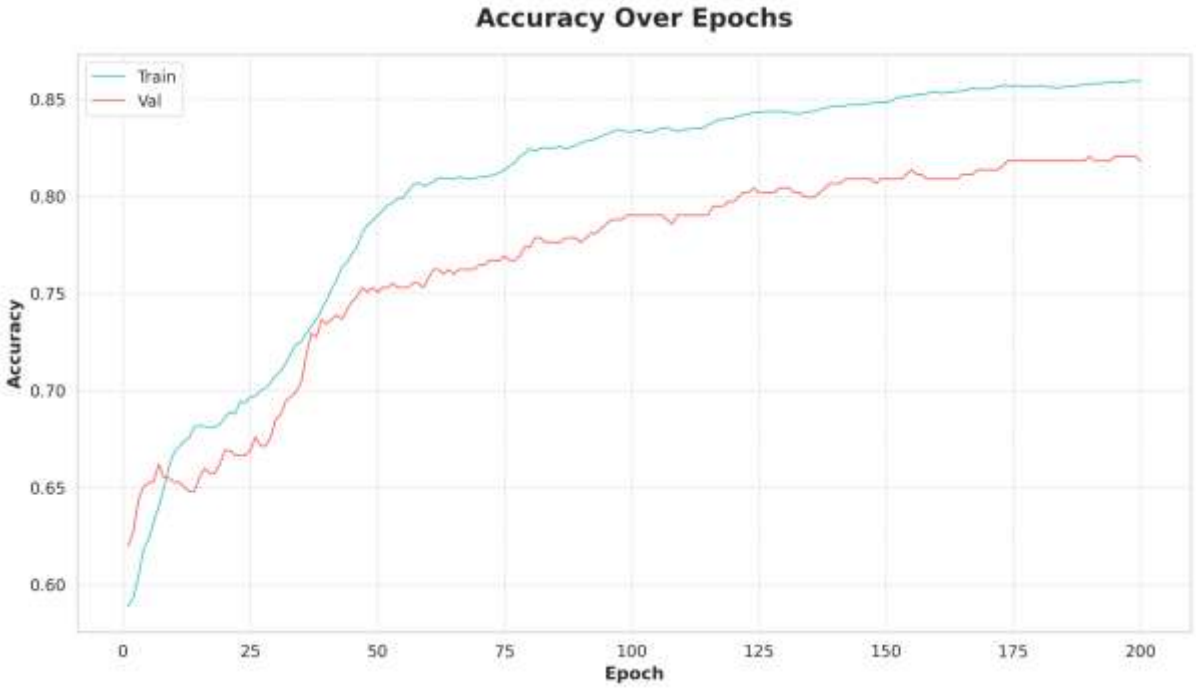
Fold 3


Accuracy Over Epochs
Train
Val
Accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0
25
50
75
100
125
150
175
200
Epoch

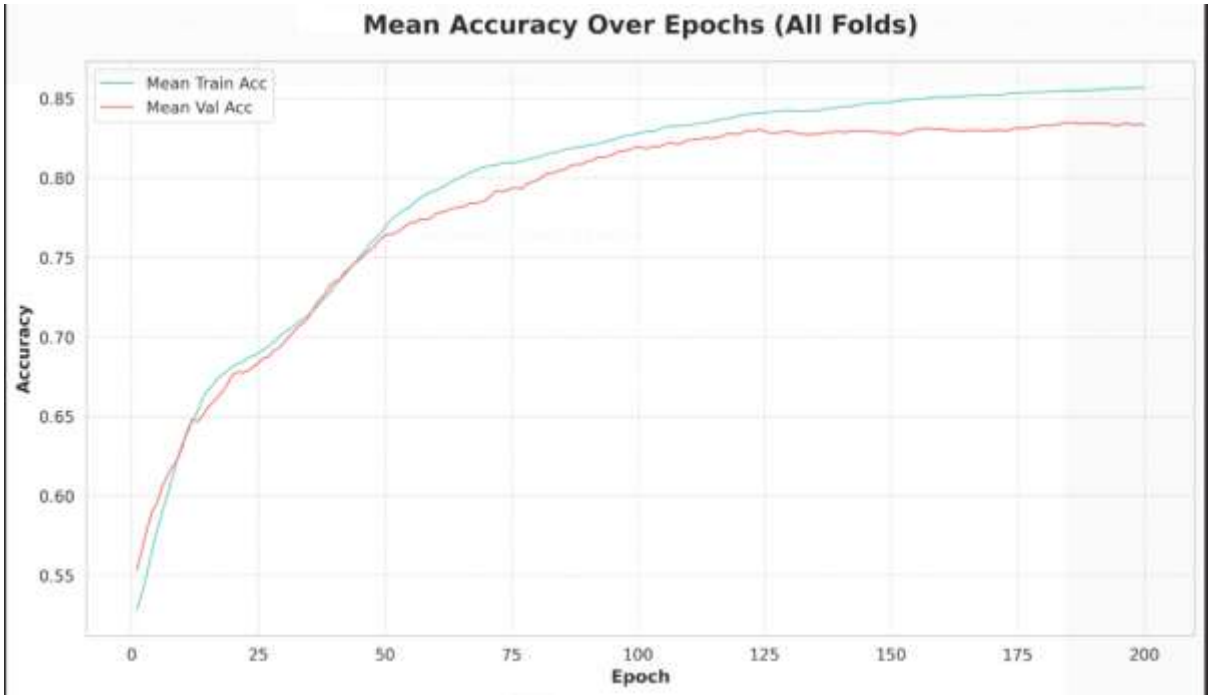
Fold 4



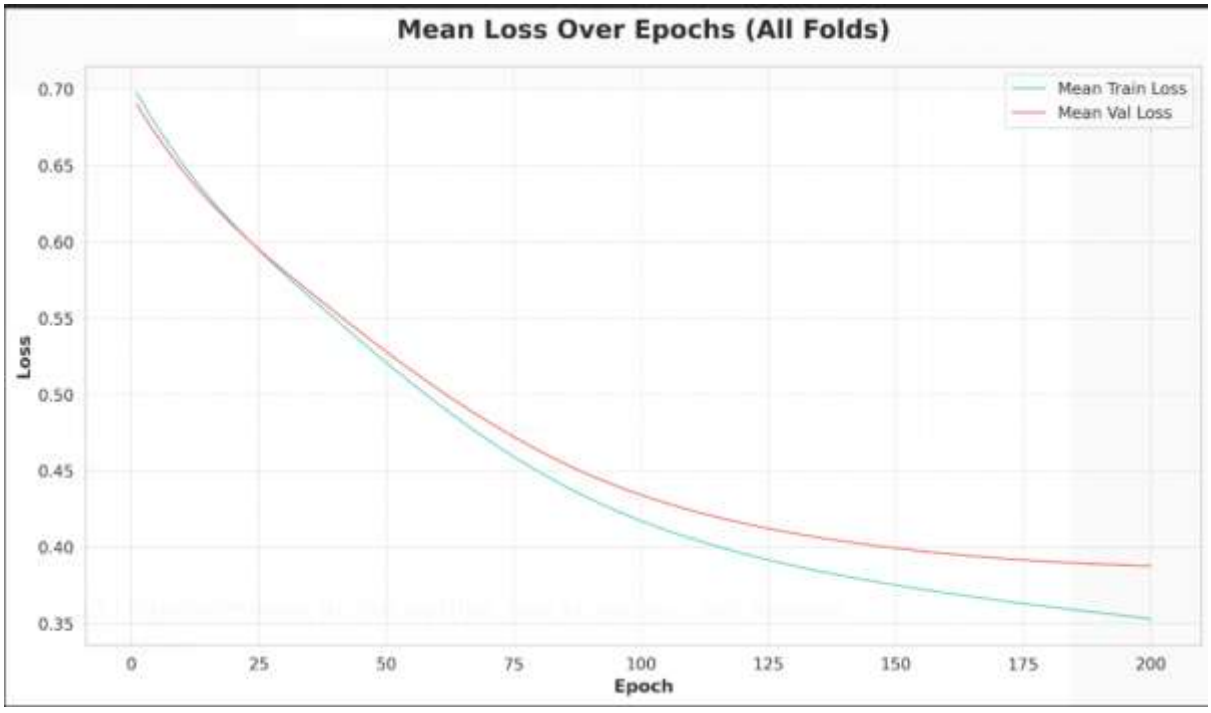
Fold5



Και τέλος τον μέσο όρο αυτόν τον γραφικών παραστάσεων:



Και τον μέσο όρο των Loss για όλα τα folds:



Εξετάζοντας αυτά τα δύο γραφήματα, μπορούμε να δούμε την απόδοση εκπαίδευσης του νευρωνικού δικτύου για το μοντέλο πρόβλεψης της νόσου Alzheimer σε 200 εποχές.

Το γράφημα ακρίβειας δείχνει ότι τόσο η ακρίβεια εκπαίδευσης (γαλαζοπράσινη) όσο και η ακρίβεια επικύρωσης (κόκκινη) βελτιώνονται σταθερά με την πάροδο του χρόνου, με την ακρίβεια εκπαίδευσης να φτάνει περίπου το 85,5% και την ακρίβεια επικύρωσης περίπου το 83,5% μέχρι την εποχή 200. Οι καμπύλες ακολουθούν ένα τυπικό μοτίβο μάθησης - ταχεία

βελτίωση στις πρώτες 75 εποχές ακολουθούμενη από μια πιο σταδιακή αύξηση. Αξιοσημείωτο είναι ότι υπάρχει ένα μικρό αλλά επίμονο κενό μεταξύ της ακρίβειας εκπαίδευσης και επικύρωσης μετά την εποχή 75, υποδεικνύοντας ελαφρά υπερπροσαρμογή.

Το γράφημα απώλειας δείχνει συνεχή μείωση τόσο στην απώλεια εκπαίδευσης όσο και στην απώλεια επικύρωσης, ξεκινώντας από περίπου 0,70 και μειώνοντας σε 0,35 (εκπαίδευση) και 0,39 (επικύρωση). Το σταδιακά διευρυνόμενο χάσμα μεταξύ της απώλειας εκπαίδευσης και επικύρωσης υποδηλώνει επίσης κάποια υπερπροσαρμογή, αν και παραμένει μέτρια.

Η υπερπροσαρμογή (overfitting) που παρατηρείται εκδηλώνεται με δύο τρόπους: Πρώτον, βλέπουμε ότι μετά την εποχή 125 περίπου, το κενό μεταξύ της ακρίβειας εκπαίδευσης και επικύρωσης διευρύνεται σταθερά, με την καμπύλη εκπαίδευσης να συνεχίζει να αυξάνεται ενώ η καμπύλη επικύρωσης σταθεροποιείται. Δεύτερον, στο γράφημα απώλειας, η απόκλιση μεταξύ των δύο καμπυλών γίνεται πιο έντονη μετά την εποχή 100, με την απώλεια εκπαίδευσης να συνεχίζει να μειώνεται ενώ η απώλεια επικύρωσης μειώνεται με βραδύτερο ρυθμό. Αυτό υποδηλώνει ότι το μοντέλο αρχίζει να "απομνημονεύει" τα δεδομένα εκπαίδευσης αντί να μαθαίνει γενικεύσιμα μοτίβα, αν και το φαινόμενο δεν είναι ακόμη σοβαρό.

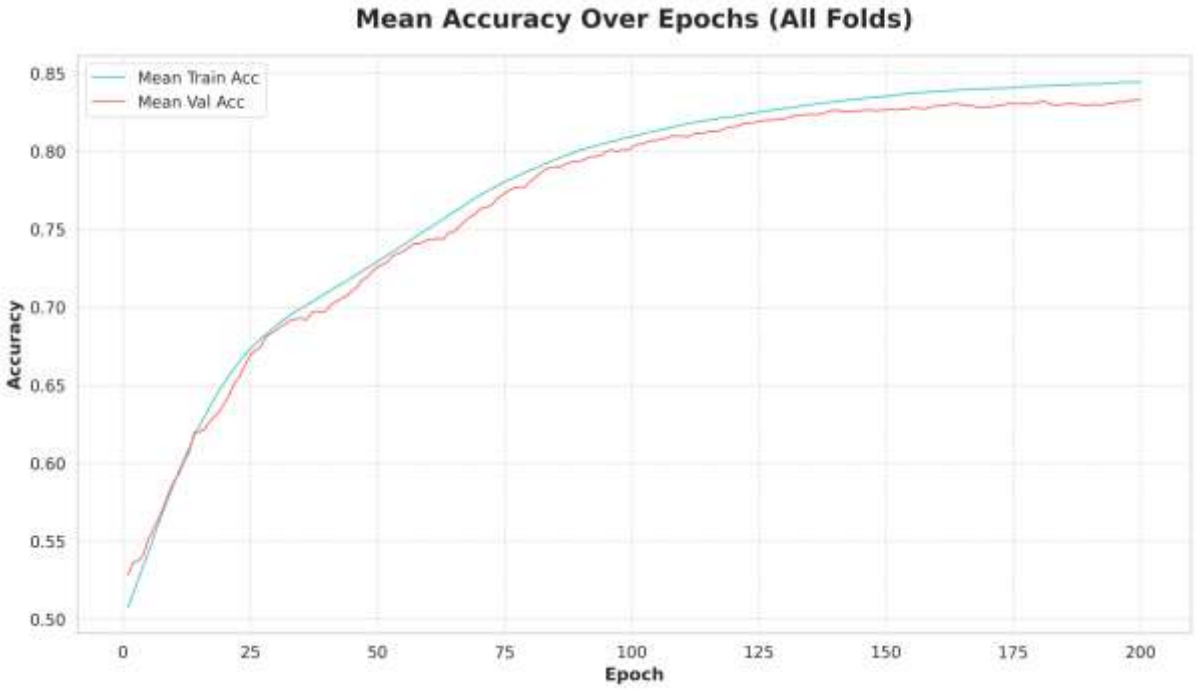
Συνολικά, αυτά τα γραφήματα υποδεικνύουν ένα εύλογα καλά αποδοτικό μοντέλο με καλές ιδιότητες σύγκλισης και αποδεκτή γενίκευση, αν και υπάρχει περιθώριο βελτίωσης μέσω τεχνικών κανονικοποίησης για την αντιμετώπιση της υπερπροσαρμογής που παρατηρείται.

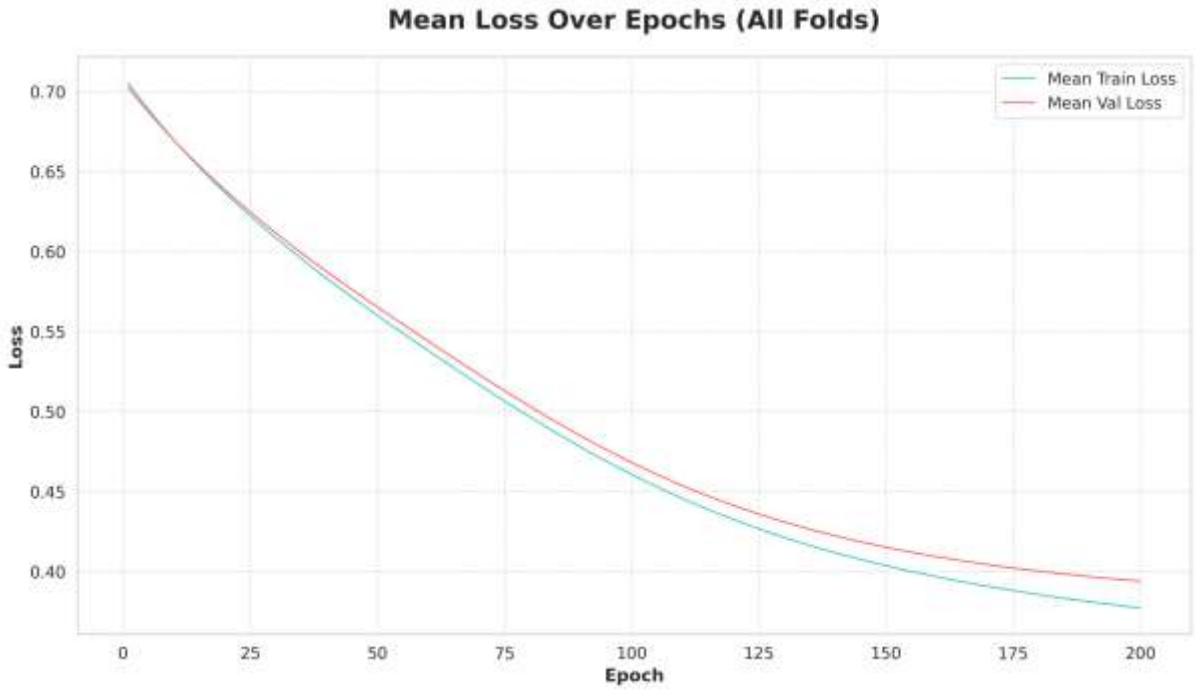
ε) Πειραματισμος με τον αριθμό των νευρώνων του κρυφού επιπέδου

Αριθμός νευρώνων στο κρυφό επίπεδο	CE loss	MSE	Acc
H1 = I/2 =17	0.3995±0.0229	0.1247±0.0098	0.8283±0.0245
H1 = 2I/3 = 23	0.3910±0.0227	0.1215±0.0091	0.8343±0.0175
H1 = I = 35	0.3876±0.0241	0.1198±0.0089	0.8353±0.0182
H1 = 2I =70	0.3916±0.0241	0.1207±0.0083	0.8288±0.013

Plots:

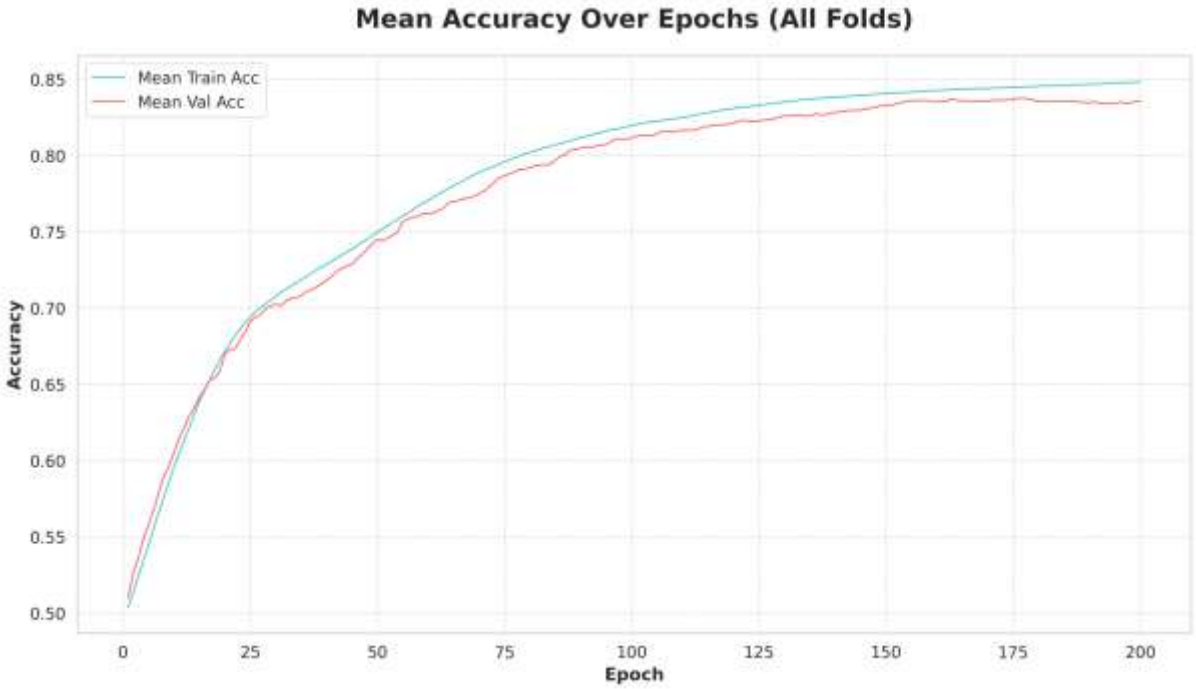
H17:

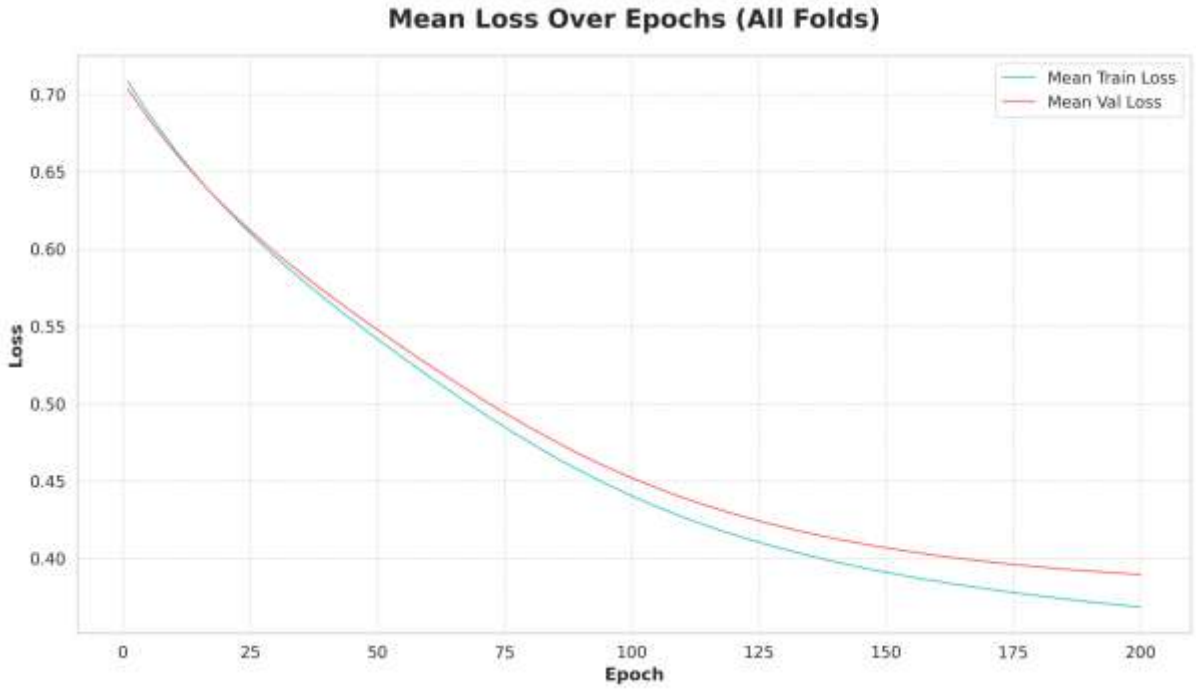

Mean Accuracy Over Epochs (All Folds)
Mean Train Acc
Mean Val Acc
Accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
25
50
75
100
125
150
175
200
Epoch


Mean Loss Over Epochs (All Folds)
Mean Train Loss
Mean Val Loss
Loss
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
25
50
75
100
125
150
175
200
Epoch

Plots:

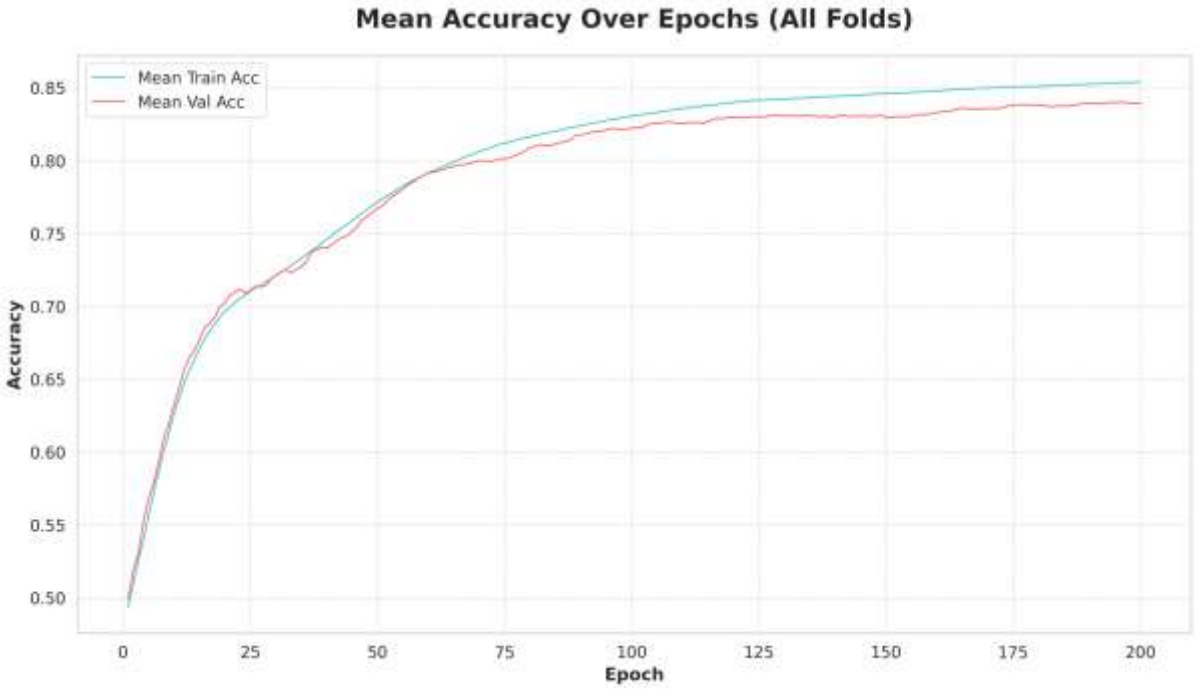
H23:

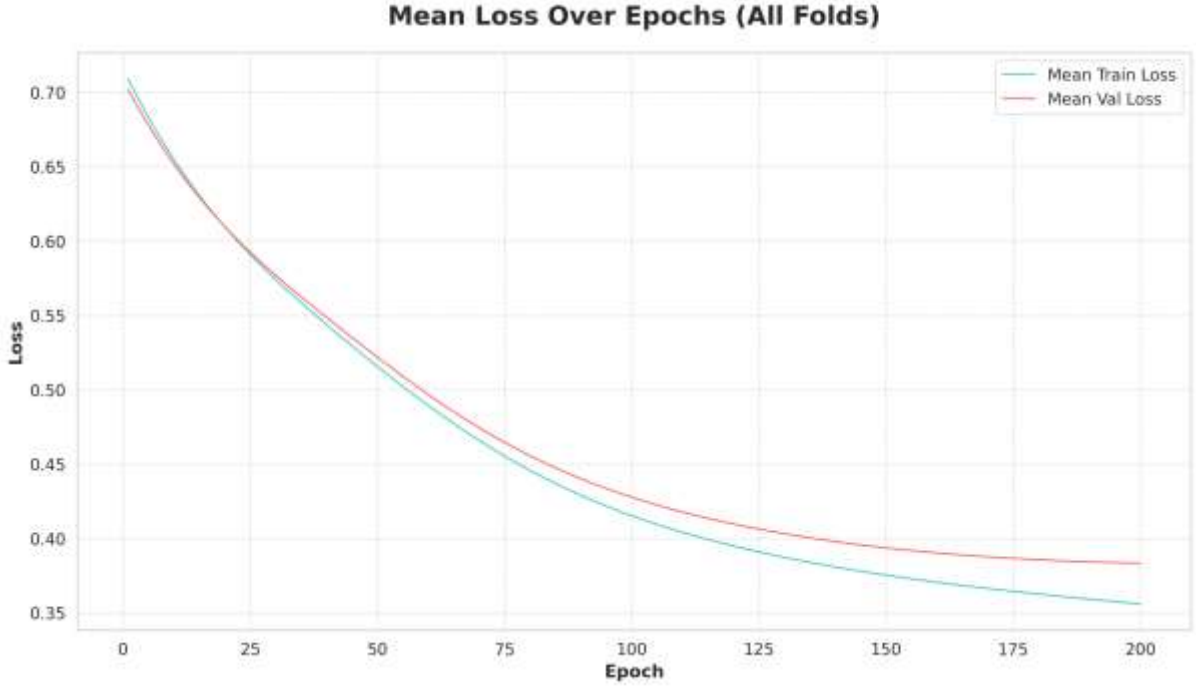

Mean Accuracy Over Epochs (All Folds)
Mean Train Acc
Mean Val Acc
Accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
25
50
75
100
125
150
175
200
Epoch


Mean Loss Over Epochs (All Folds)
Mean Train Loss
Mean Val Loss
Loss
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
25
50
75
100
125
150
175
200
Epoch

Plots:

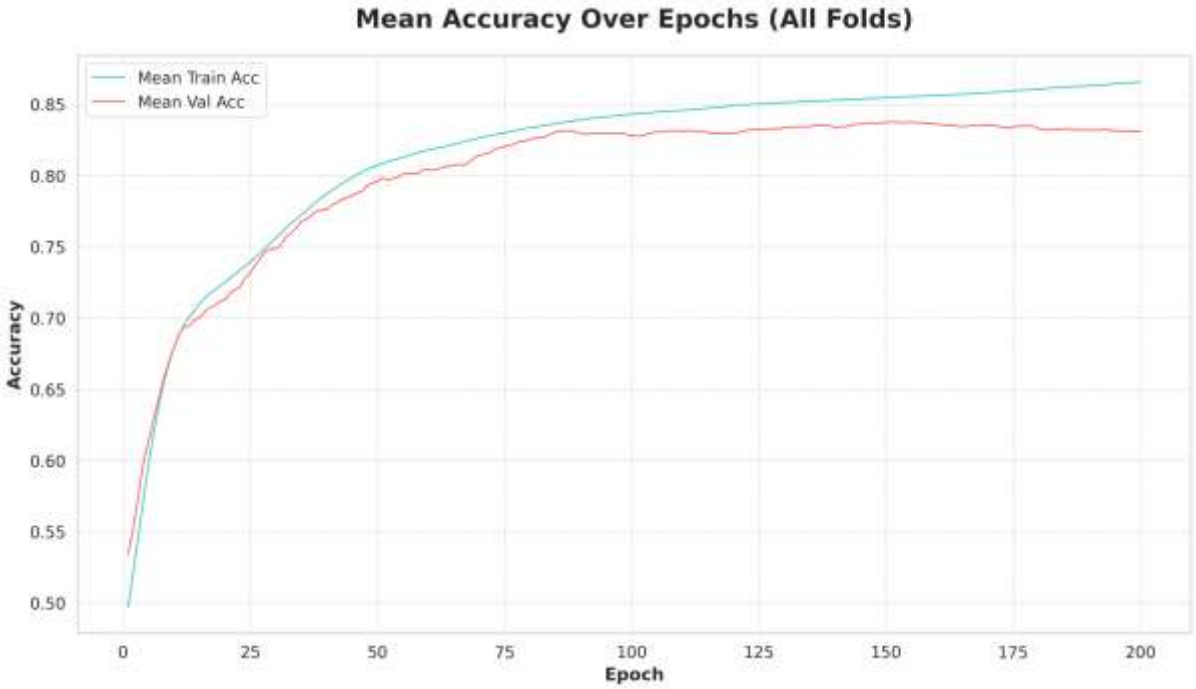
H35

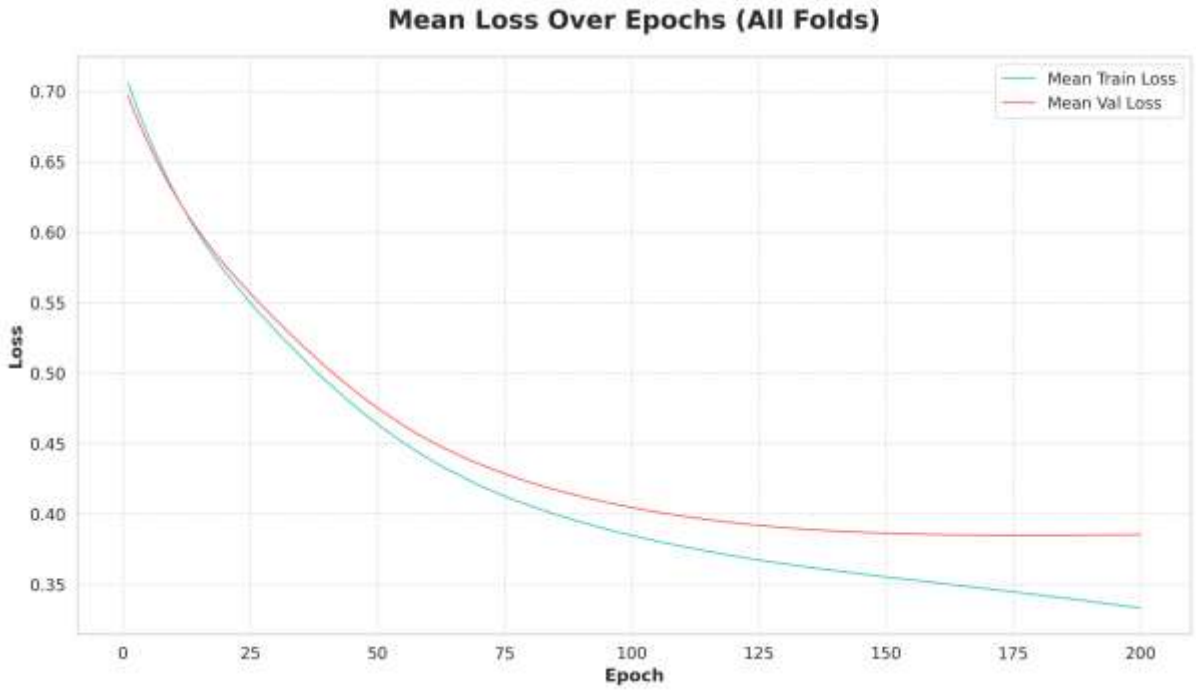

Mean Accuracy Over Epochs (All Folds)
Mean Train Acc
Mean Val Acc
Accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
25
50
75
100
125
150
175
200
Epoch


Mean Loss Over Epochs (All Folds)
Mean Train Loss
Mean Val Loss
Loss
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
25
50
75
100
125
150
175
200
Epoch

Plots:

H70:





Σχολιασμός αποτελεσμάτων

(i) Αριθμός κρυφών κόμβων

Παρατηρούμε ότι καθώς αυξάνουμε τον αριθμό κρυφών κόμβων από 17 σε 35, μειώνεται σταδιακά το CE loss και το MSE, ενώ η ακρίβεια βελτιώνεται έως και περίπου 0.835. Η περαιτέρω αύξηση σε 70 κόμβους δεν προσφέρει επιπλέον βελτίωση· αντιθέτως, το CE αυξάνεται ελαφρώς και η ακρίβεια επιστρέφει σε επίπεδα ~0.829, υποδεικνύοντας αρχόμενη υπερπροσαρμογή ("over-fitting"). Επομένως, εμπειρικά η τιμή H≈I (δηλαδή 35) συνιστά βέλτιστη ισορροπία μεταξύ ικανότητας μοντέλου και γενίκευσης.

(ii) Επιλογή συνάρτησης κόστους

Ως συνάρτηση κόστους κατά την εκπαίδευση χρησιμοποιήσαμε αποκλειστικά τη Cross-Entropy (nn.CrossEntropyLoss), καθώς είναι ειδικά σχεδιασμένη για ταξινομήσεις με softmax εξόδου και εξασφαλίζει δυναμικά gradients ακόμη και όταν οι εκτιμήσεις πλησιάζουν στα άκρα 0 ή 1. Το MSE χρησιμοποιήθηκε μόνον ως επιπλέον μετρική αξιολόγησης πάνω στις softmax-πιθανότητες, παρέχοντας συμπληρωματικό στίγμα σφάλματος αλλά χωρίς συμμετοχή στο backward pass.

(iii) Επιλογή συνάρτησης ενεργοποίησης

Στα κρυφά επίπεδα υιοθετήσαμε τη συνάρτηση Tanh, η οποία παρουσιάζει συμμετρική έξοδο στο διάστημα (–1,1) και κέντρο μηδέν, διευκολύνοντας τη σύγκλιση σε προβλήματα όπου οι είσοδοι έχουν πλέον και αρνητικές τιμές μετά από scaling. Αν και η Tanh τείνει σε κορεσμό για μεγάλες απόλυτες τιμές εισόδου, στα μεσαίου μεγέθους δίκτυά μας παρείχε σταθερή εκπαίδευση· εναλλακτική επιλογή όπως η SiLU θα μπορούσε να μετριάσει το φαινόμενο "dying units" και να επιταχύνει περαιτέρω τη σύγκλιση, όμως εδώ η Tanh απέδωσε ικανοποιητικά.

(iv) Ταχύτητα σύγκλισης

Από την ανάλυση των γραφικών παραστάσεων για τα διαφορετικά μεγέθη κρυφού επιπέδου, παρατηρούμε ότι:

Όλα τα μοντέλα ξεκινούν με παρόμοια τιμή απώλειας (περίπου 0.70) και ακολουθούν φθίνουσα πορεία καθώς αυξάνονται οι εποχές εκπαίδευσης.

Διαφορά μεταξύ καμπυλών train και validation: Σε όλες τις περιπτώσεις, η καμπύλη validation loss (κόκκινη) παραμένει ελαφρώς υψηλότερη από την καμπύλη training loss (πράσινη), ιδιαίτερα μετά την εποχή 50. Αυτό είναι αναμενόμενο και δείχνει ότι τα μοντέλα δεν υπερπροσαρμόζονται σημαντικά στα δεδομένα εκπαίδευσης.

Ρυθμός σύγκλισης: Η πτώση της καμπύλης απώλειας είναι ιδιαίτερα απότομη στις πρώτες 50 εποχές για όλα τα μοντέλα, ενώ μετά την εποχή 100 η μείωση γίνεται πιο σταδιακή.

Όλα τα μοντέλα συγκλίνουν σε σταθερή κατάσταση μετά από περίπου 150-175 εποχές, με το μοντέλο H1 = 35 να επιτυγχάνει τη χαμηλότερη τελική τιμή απώλειας (περίπου 0.38 για validation loss).

Επίσης παρατήρησα ότι τομοντέλο με H1 = 70 (διπλάσιο των εισόδων) εμφανίζει το μεγαλύτερο χάσμα μεταξύ training και validation loss στις τελευταίες εποχές, υποδηλώνοντας μεγαλύτερη τάση προς υπερπροσαρμογή. Αυτό είναι αναμενόμενο λόγω του μεγαλύτερου αριθμού παραμέτρων που επιτρέπει στο μοντέλο να "απομνημονεύσει" καλύτερα τα δεδομένα εκπαίδευσης. Ενο τα μοντέλα με H1 = 35 και H1 = 23 παρουσιάζουν μέτρια διαφορά μεταξύ των καμπυλών, υποδεικνιοντας καλή ισορροπία μεταξύ προσαρμογής και γενίκευσης. Τελος μοντέλο με H1 = 17 εμφανίζει τη μικρότερη διαφορά μεταξύ των καμπυλών αλλά υψηλότερο συνολικό loss, δείχνοντας ότι έχει μικρότερη τάση υπερπροσαρμογής αλλά παράλληλα μειωμένη ικανότητα εκμάθησης των δεδομένων.

Έτσι το καλύτερο μοντέλο από την επιλογή του αριθμού νευρώνων στο κρυφό επίπεδο ίσου με τον αριθμό των εισόδων είναι αθτο με **H1 = I = 35** το οποίο αποδεικνύεται βέλτιστο τόσο ως προς την απόδοση (χαμηλότερο loss, υψηλότερη ακρίβεια) όσο και ως προς τη συμπεριφορά εκπαίδευσης, προσφέροντας καλή ισορροπία μεταξύ υπερπροσαρμογής και υποπροσαρμογής. Η χρήση Cross-Entropy ως συνάρτηση απώλειας σε συνδυασμό με ενεργοποίηση Tanh στα κρυφά επίπεδα προσφέρει ένα αποδοτικό και γενικεύσιμο μοντέλο για την ταξινόμηση της νόσου Alzheimer.

στ) Κριτήριο τερματισμού.

Ο καθορισμός ενός κατάλληλου κριτηρίου τερματισμού της εκπαίδευσης σε κάθε fold του Cross-Validation αποτελεί αναγκαία προϋπόθεση για την αποφυγή υπερπροσαρμογής και τη βελτιστοποίηση του υπολογιστικού κόστους. Χωρίς σαφή όριο, το μοντέλο ενδέχεται να εξελίσσεται πέρα από το σημείο όπου η βελτίωση στη γενίκευση σταματά, ενώ η διατήρηση μηχανισμών τερματισμού εξασφαλίζει ότι η εκπαίδευση διακόπτεται όταν το validation loss δεν παρουσιάζει πλέον ουσιαστικά κέρδη.

Η τεχνική του πρόωρου σταματήματος εφαρμόστηκε στο καλύτερο μοντέλο από το προηγούμενο ερώτημα το **H1 = I = 35.**

Η τεχνική του πρόωρου σταματήματος (early stopping) βασίζεται στην παρακολούθηση μιας επιλεγμένης μετρικής—συνήθως του validation loss—κατά τη διάρκεια των εποχών εκπαίδευσης. Οριζουμε ένα «παράθυρο υπομονής» (patience), δηλαδή έναν αριθμό διαδοχικών εποχών χωρίς βελτίωση, και αν μετά από αυτές τις εποχές το validation loss δεν μειωθεί πέρα από ένα ελάχιστο όριο (min_delta), διακόπτουμε την εκπαίδευση. Με αυτό τον τρόπο, πετυχαίνουμε δθο στόχους: πρώτον, αποφεύγουμε την εκπαίδευση σε φάση στασιμότητας όπου το μοντέλο ενδέχεται να υπερπροσαρμοστεί, και δεύτερον, εξοικονομούμε υπολογιστικούς πόρους διακοπτοντας πρόωρα περιττές εποχές.

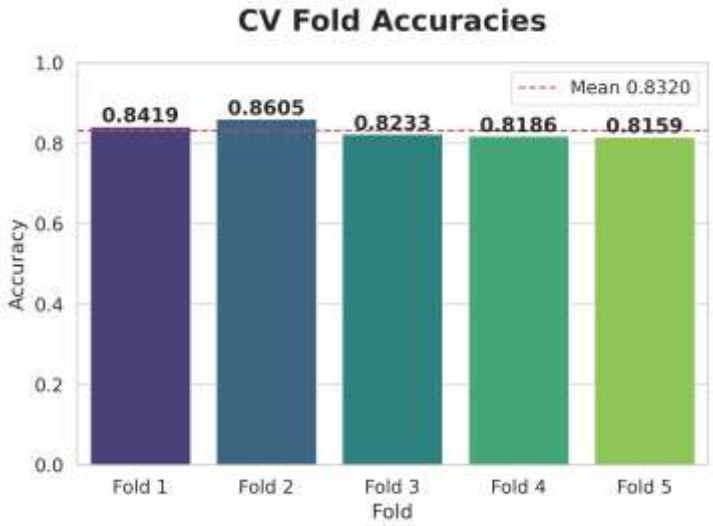
Τεκμηρίωση επιλογής

Ως κριτήριο τερματισμού επιλέγουμε τη μείωση του validation loss με παράθυρο υπομονής 3 εποχών και ελάχιστη βελτίωση min_delta=1e-3. Αυτές οι τιμές προέκυψαν εμπειρικά από τις καμπύλες σύγκλισης, όπου μετά από περίπου 150–180 εποχές οι μεταβολές του validation loss ήταν εξαιρετικά μικρές. Η early stopping εξασφαλίζει ότι αν για 3 συνεχόμενες εποχές δεν επιτευχθεί ουσιαστική πτώση του validation loss, η εκπαίδευση διακόπτεται, διατηρώντας το μοντέλο στο πιο «γενικεύσιμο» σημείο του.

Εφαρμογή σε κάθε fold

Σε κάθε fold της 5-fold διαδικασίας, παρακολουθούμε το validation loss σε κάθε εποχή. Κρατάμε τον δείκτη της καλύτερης εποχής (best_epoch) όπου το validation loss ήταν ελάχιστο και συγκρίνουμε το τρέχον validation loss με αυτήν την τιμή συν το min_delta. Αν ο αριθμός των εποχών χωρίς βελτίωση υπερβεί την παράμετρο patience, σταματάμε τον βρόχο εκπαίδευσης και επαναφέρουμε το μοντέλο στην κατάσταση της καλύτερης εποχής. Με αυτων τον μηχανισμό επιτυγχανεται ο έλεγχος της ποιότητας της μάθησης, αποτρέπεται η εκπαίδευση σε περιττές εποχές και μεγιστοποιείται η απόδοση στο validation set.

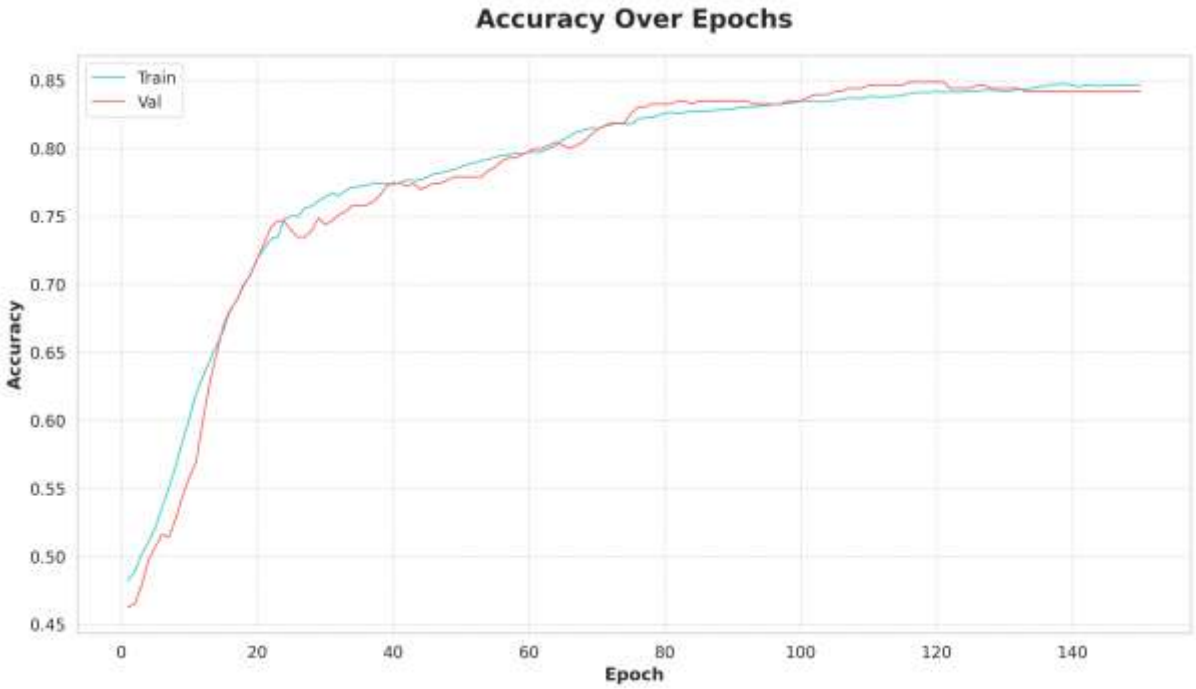
Αποτελεσματα:



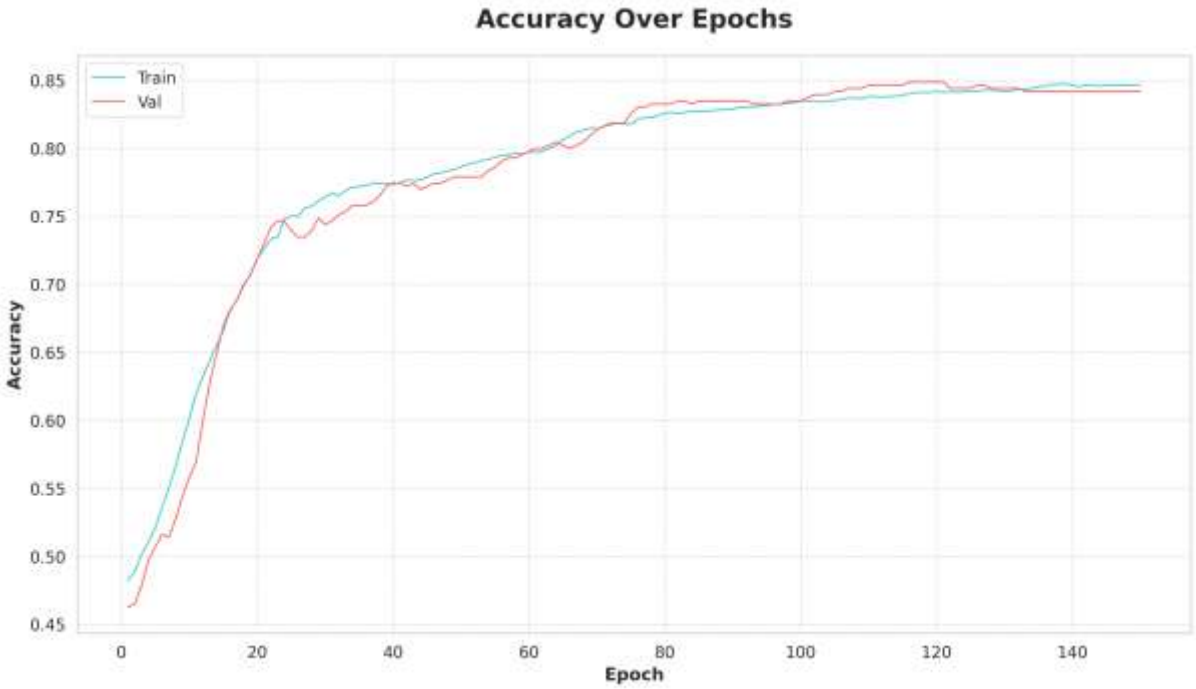

```
Early stopping at epoch 150 (no improvement for 3 epochs)
Fold 1 Accuracy: 0.8419
Early stopping at epoch 154 (no improvement for 3 epochs)
Fold 2 Accuracy: 0.8605
Early stopping at epoch 141 (no improvement for 3 epochs)
Fold 3 Accuracy: 0.8233
Early stopping at epoch 157 (no improvement for 3 epochs)
Fold 4 Accuracy: 0.8186
Early stopping at epoch 154 (no improvement for 3 epochs)
Fold 5 Accuracy: 0.8159

CV Mean Acc: 0.8320 ± 0.0169
```

Plots:

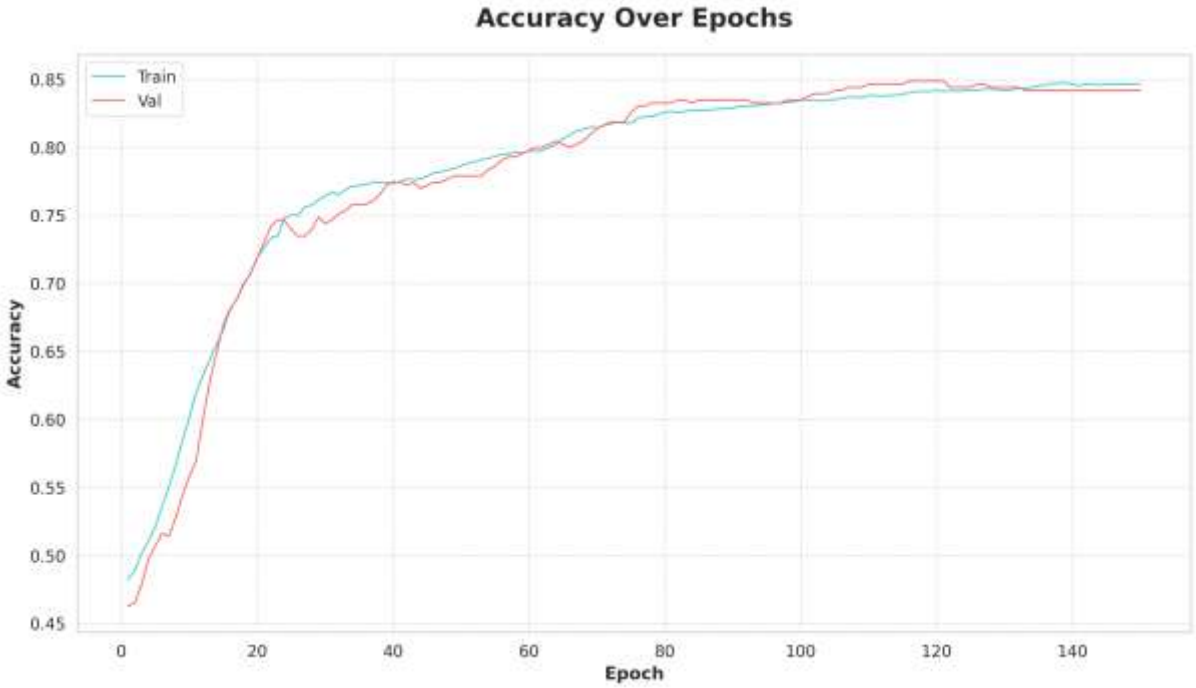
Fold1



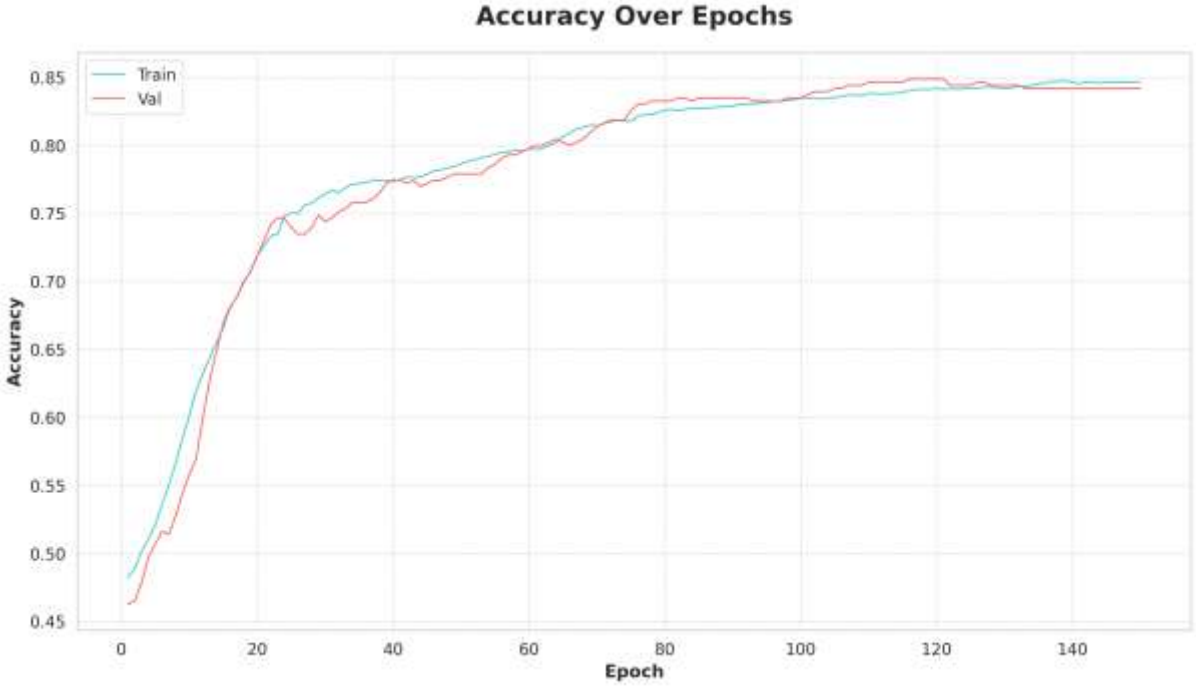
Fold2



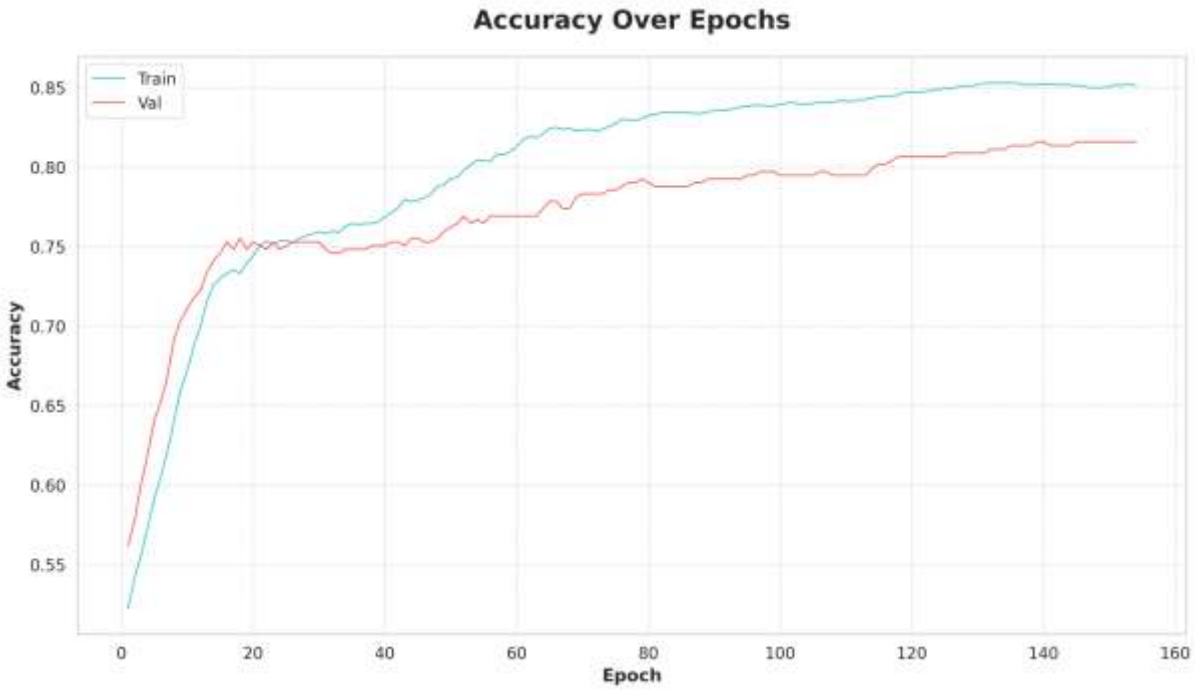
Fold3


Accuracy Over Epochs
Train
Val
Accuracy
Epoch
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
20
40
60
80
100
120
140

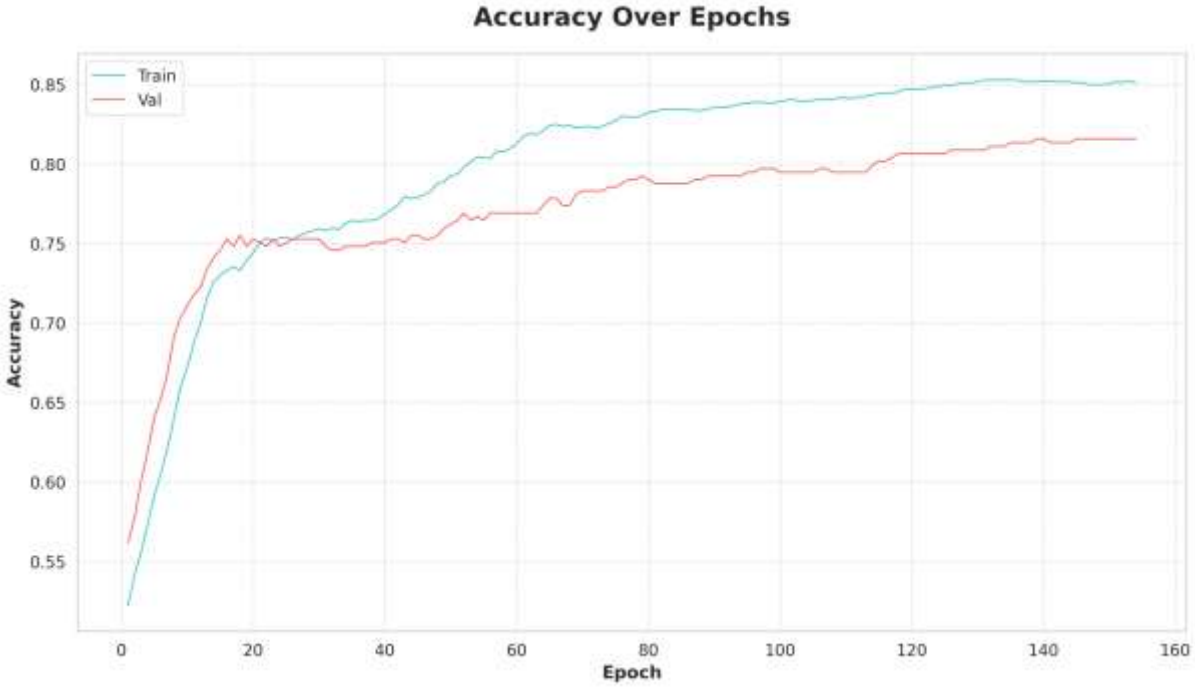
Fold4

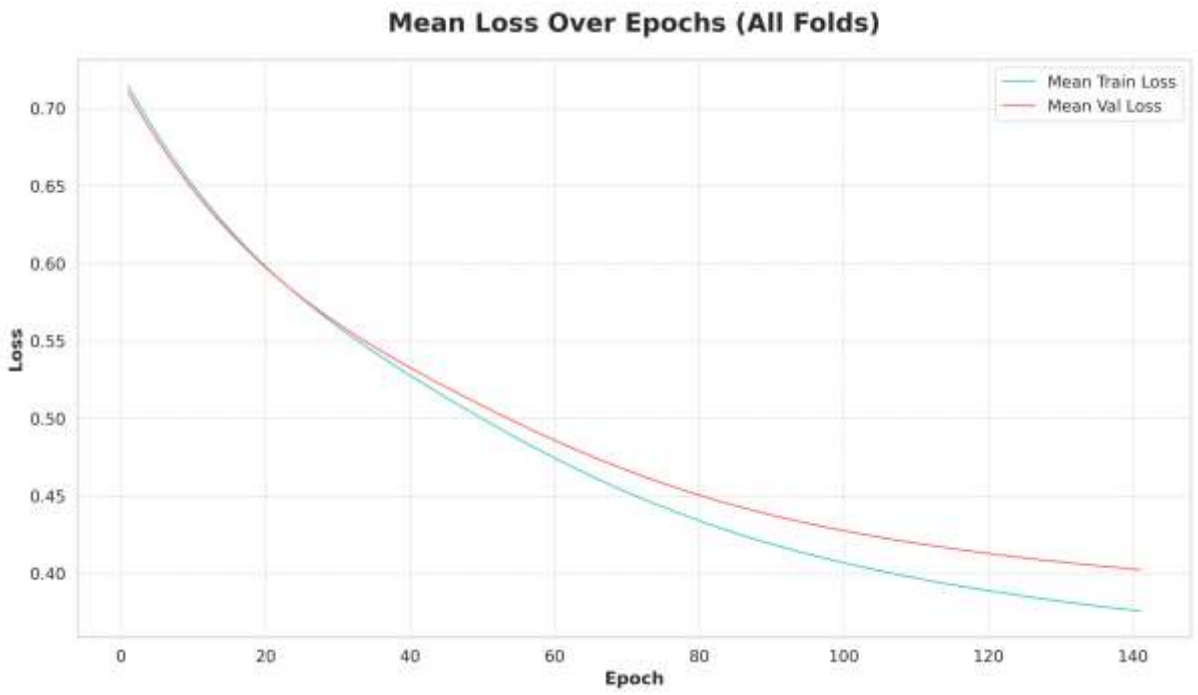

Accuracy Over Epochs
Train
Val
Accuracy
Epoch
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
20
40
60
80
100
120
140

Fold5



Mean Plots Across all plots:





Παρατηρούμε μετρά το early stopping ότι τα nn συγκλίνον πιο γρηγορά και στο mean loss έχουμε μειώσει του κενού μεταξύ vall και train που σημαίνει ότι μειώνετε η πιθανότητα να έχουμε το πρόβλημα του overfiting.

A3. Μεταβολές στον ρυθμό εκπαίδευσης και σταθεράς ορμής

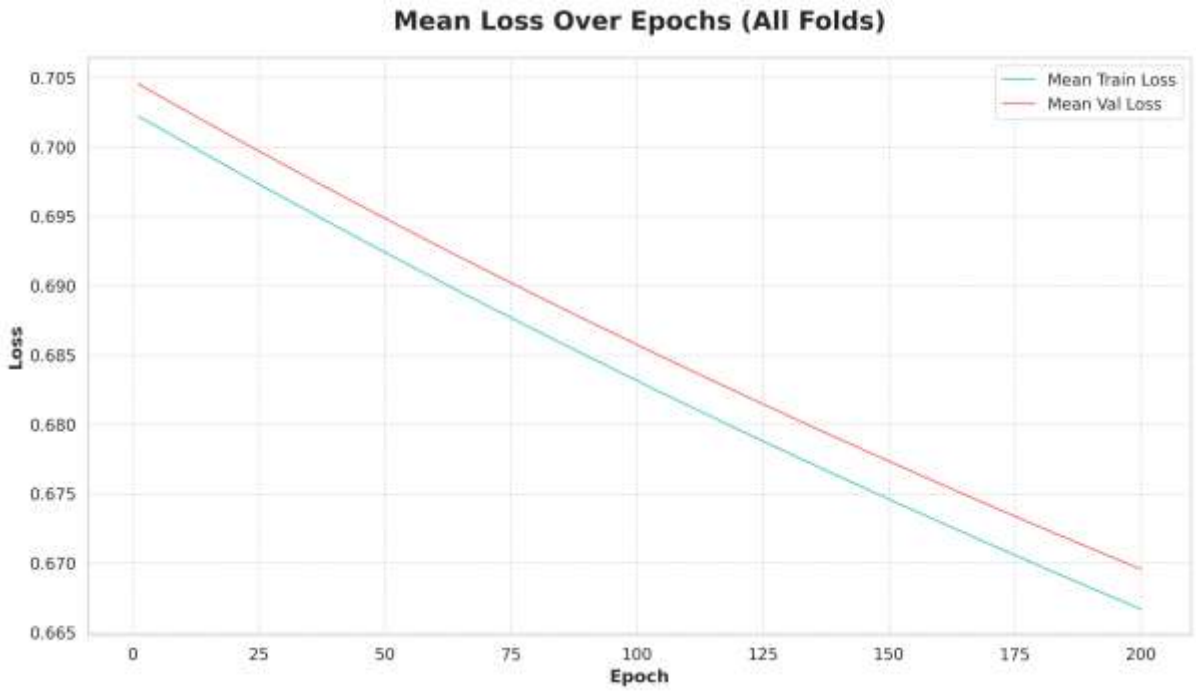
Επιλέγοντας ως τοπολογία το δίκτυο SimpleNet (hidden_size=35, activation=Tanh) που έδωσε το καλύτερο αποτέλεσμα, εφαρμόσαμε ένα απλό grid-search για τον ρυθμό εκπαίδευσης η και τη σταθερά ορμής m. Τα αποτελέσματα παρουσιάζονται στον παρακάτω πίνακα (μέσος ± τυπική απόκλιση):

η	m	CE loss	MSE	Acc
0.001	**0.2**	**0.6696±0.0089**	**0.2383±0.0044**	**0.6017±0.0315**
0.001	**0.6**	**0.6400±0.0212**	**0.2240±0.0102**	**0.6547±0.043**
0.05	**0.6**	**0.3880±0.0233**	**0.1203±0.008**	**0.8343±0.018**
0.1	0.6	**0.3838±0.0246**	**0.1178±0.0089**	**0.8343±0.0128**

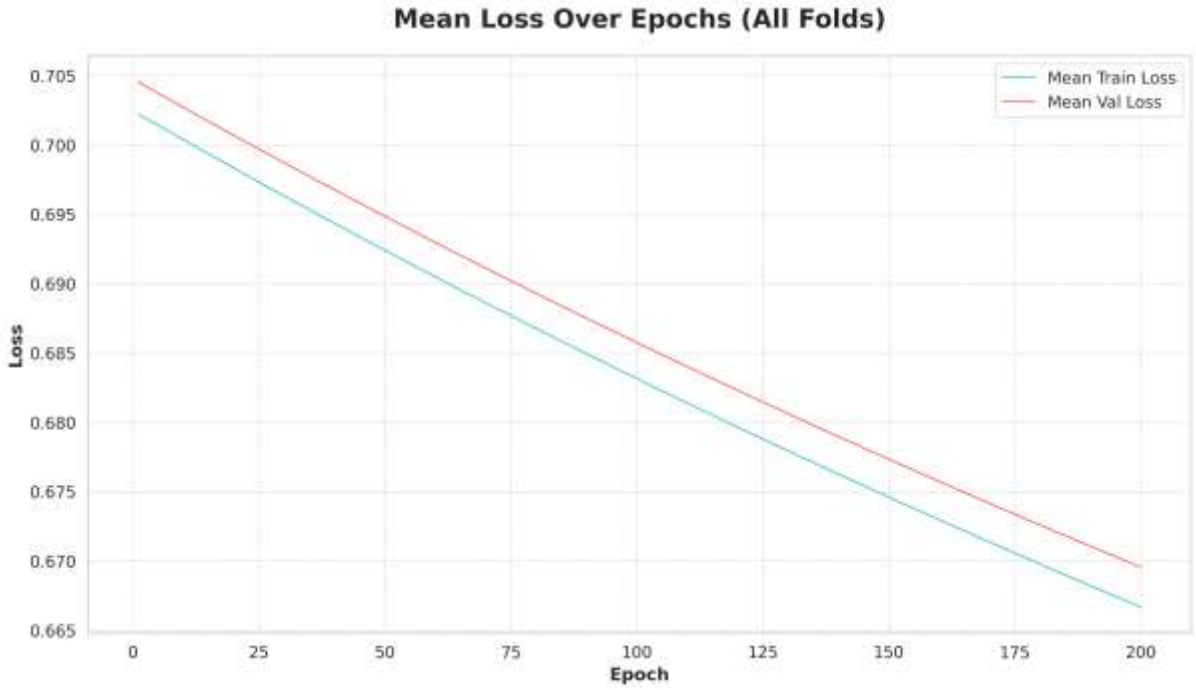
Plots:

η=0.001 , m= 0.2

Loss

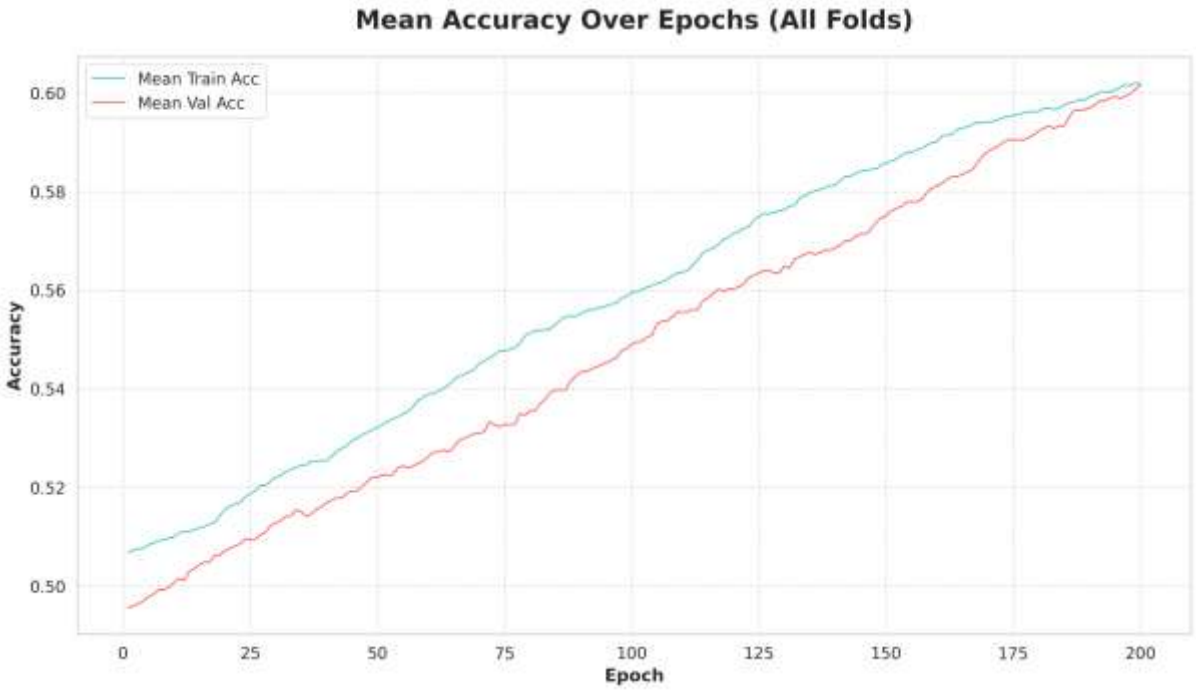


Accuracy

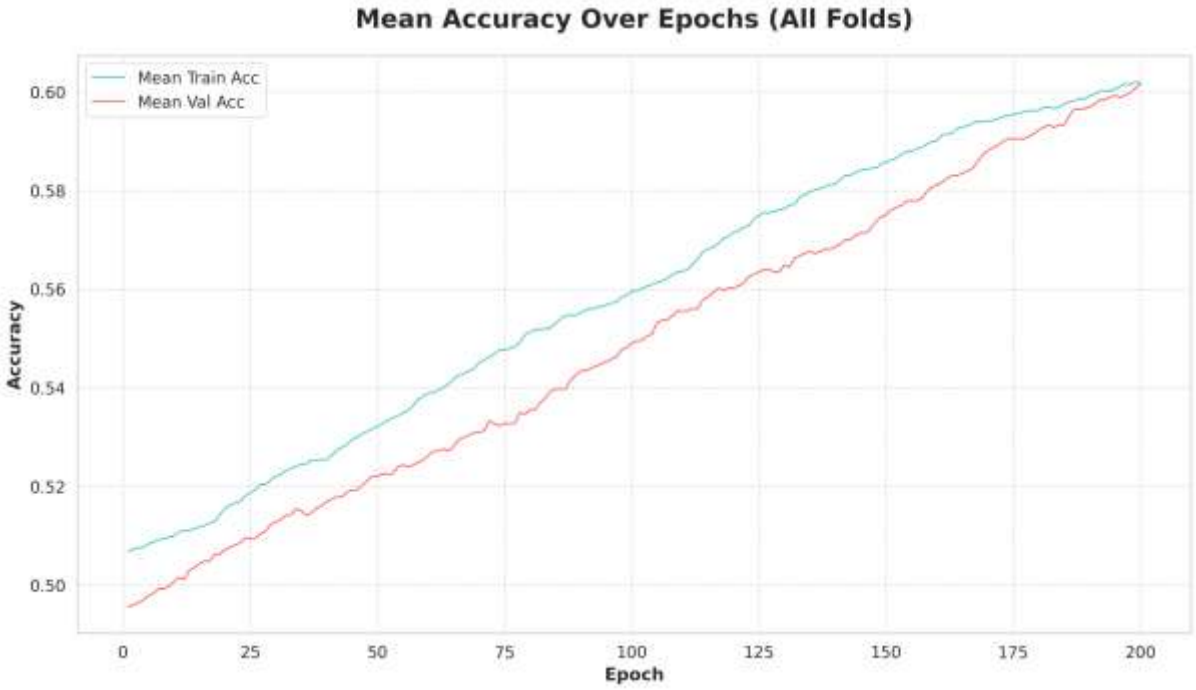


η=0.001, m= 0.6

Loss

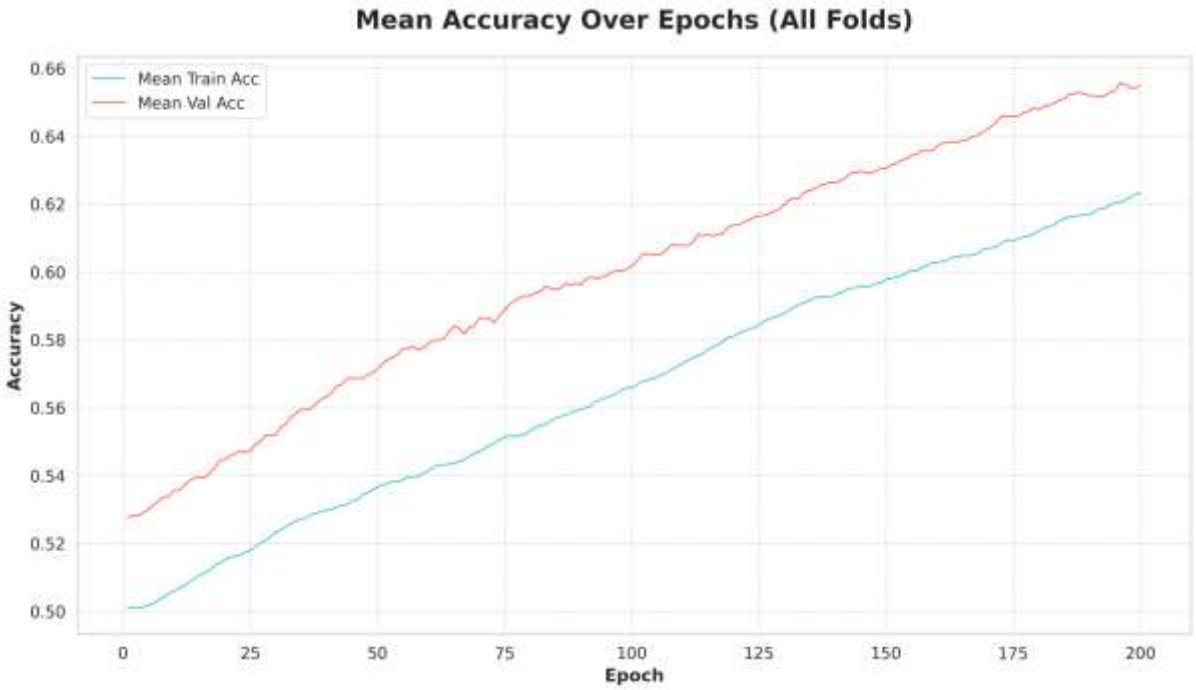


Accuracy

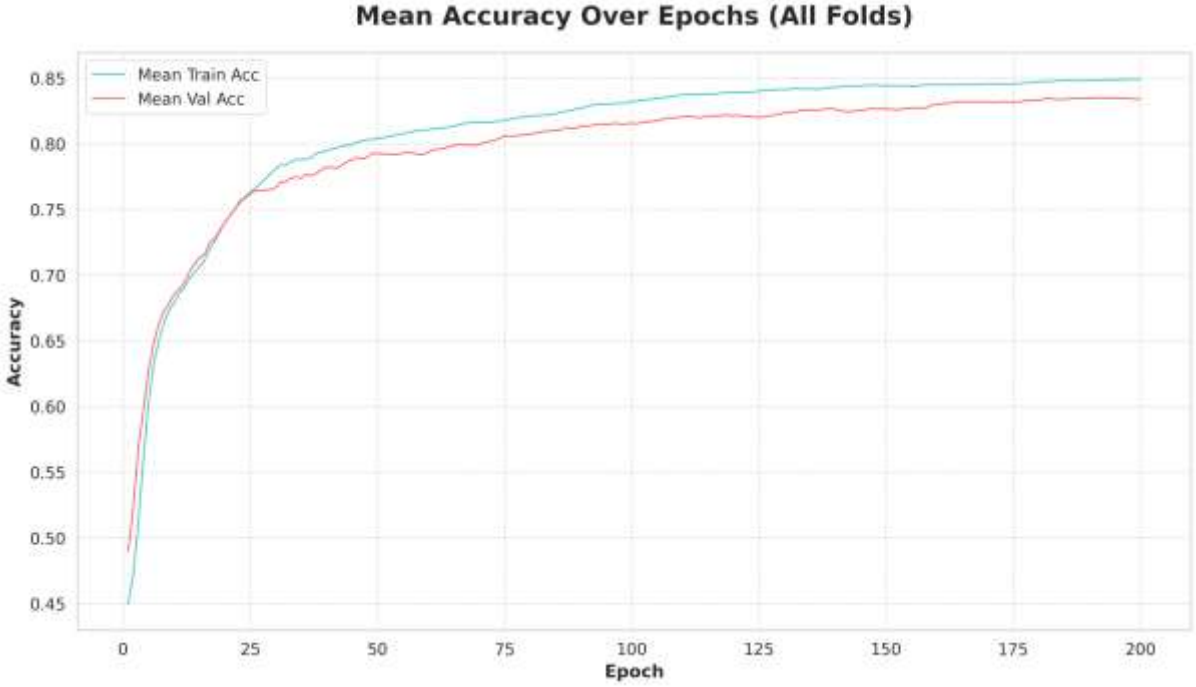


η=0.05, m= 0.6

Loss

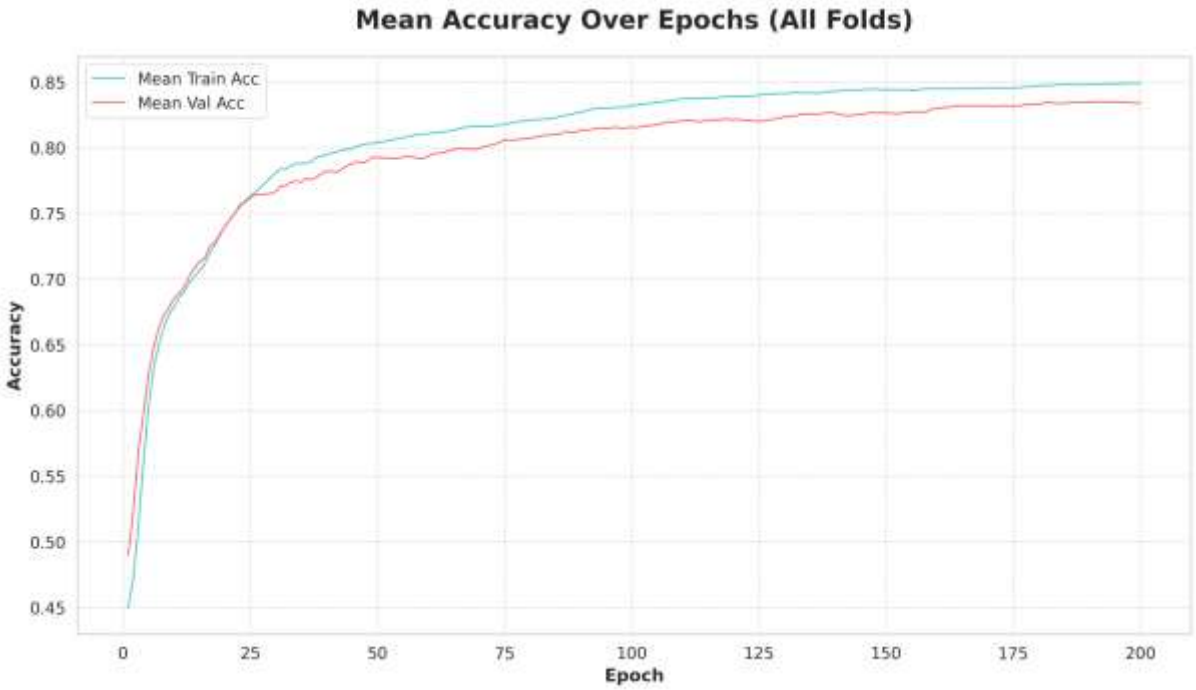


Accuracy

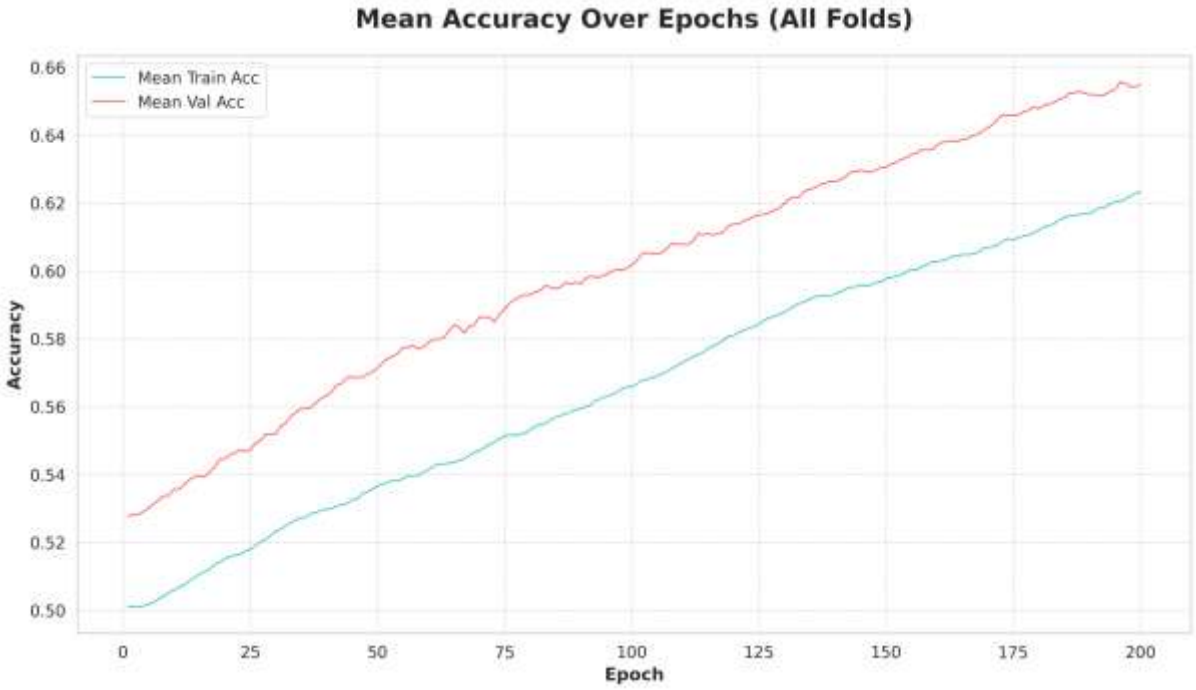


η=0.1, m= 0.6

Loss



Accuracy



Παρουσιάζουμε αναλυτικά τα αποτελέσματα της βελτιστοποίησης των υπερπαραμέτρων ρυθμού εκπαίδευσης (η) και σταθεράς ορμής (m) για το δίκτυο SimpleNet (hidden_size=35, activation = Tanh), όπως προέκυψαν από πενταπλή διασταυρούμενη επικύρωση (5-fold CV). Σε κάθε πείραμα τρέξαμε 200 εποχές με early-stopping (patience = 201) και υπολογίσαμε το μέσο και την τυπική απόκλιση του Cross-Entropy loss, του MSE loss και της ακρίβειας (Accuracy) στο validation set.

Τα γραφήματα σύγκλισης (CE loss και Accuracy μέσων folds) υπάρχουν στα αρχεία

`loss_eta{η}_m{m}.png` και `acc_eta{η}_m{m}.png`, αντίστοιχα μέσα στο φάκελο A3_Plots

Από την ανάλυση του πίνακα διαπιστώνουμε ότι για πολύ μικρό ρυθμό εκπαίδευσης (η = 0.001) η σύγκλιση είναι αργή και το δίκτυο μένει παγιδευμένο σε σχετικά υψηλό κόστος (CE ≈ 0.64–0.67) με χαμηλή τελική ακρίβεια (~60–65 %). Η αύξηση της σταθεράς ορμής από 0.2 σε 0.6 βελτιώνει ελαφρά την απόδοση, υποδεικνύοντας ότι η ορμή βοηθά στην απάλειψη μικρών ταλαντώσεων, αλλά το βήμα εκπαίδευσης παραμένει ανεπαρκές για ταχύτερη σύγκλιση.

Όταν αυξάνουμε το learning rate σε μεσαία τιμή (η = 0.05) με m = 0.6, το δίκτυο επιτυγχάνει δραστική μείωση του CE loss (~0.39) και του MSE loss (~0.12) με ακρίβεια ~83.4 %, πράγμα που σημαίνει ότι οι παράμετροι αυτοί συνδυάζουν γρήγορο βήμα και επαρκή ορμή για σταθερές ενημερώσεις βαρών. Ο περαιτέρω διπλασιασμός του learning rate σε η = 0.1 διατηρεί πρακτικά την ίδια απόδοση (CE ≈ 0.384, MSE ≈ 0.118, Acc ≈ 83.43 %) αλλά επιταχύνει ελαφρώς τη σύγκλιση, καθιστώντας αυτή την επιλογή την προτιμητέα.

Θεωρητικά, η σταθερά ορμής m πρέπει να είναι μικρότερη της μονάδας (0 < m < 1) διότι η ορμή εισάγει ένα κλάσμα της προηγούμενης «ταχύτητας» (κατεύθυνσης) στην τρέχουσα ενημέρωση των βαρών. Αν m ≥ 1, ο συνδυασμός των νέων κλίσεων και της συσσωρευμένης ορμής μπορεί να υπερκεράσει το βέλτιστο βήμα, προκαλώντας εκτροχιασμό της εκπαίδευσης, ΄πηδώντας΄ ένα ολικό η τοπικό ελάχιστο. Τημές m < 1 εξασφαλίζουν ότι το βάρος της μνήμης φθίνει εκθετικά, ελέγχοντας τις ταλαντώσεις και οδηγώντας σε ομαλότερη και πιο αξιόπιστη σύγκλιση.

Συμπερασματικά, το πείραμα έδειξε σαφή υπεροχή των συνδυασμών μεσαίου έως υψηλού learning rate (0.05–0.1) και m=0.6. Η τελική μας σύσταση είναι η χρήση η = 0.1 και m = 0.6, καθώς εξασφαλίζει υψηλή ακρίβεια (~83.4 %), χαμηλό κόστος και ταχύτατη σύγκλιση, χωρίς ανεπιθύμητη αστάθεια.

A4. Ομαλοποίηση

```
Regularization CV Results (L2):
   r      CE_loss±σ      MSE_loss±σ     Acc±σ
Early stopping at epoch 88 (no improvement for 3 epochs)
Early stopping at epoch 81 (no improvement for 3 epochs)
Early stopping at epoch 61 (no improvement for 3 epochs)
0.0001 0.3703±0.0294   0.1135±0.0097   0.8418±0.0152
Early stopping at epoch 88 (no improvement for 3 epochs)
Early stopping at epoch 75 (no improvement for 3 epochs)
Early stopping at epoch 193 (no improvement for 3 epochs)
Early stopping at epoch 60 (no improvement for 3 epochs)
0.001  0.3744±0.0222   0.1142±0.0070   0.8441±0.0136
Early stopping at epoch 94 (no improvement for 3 epochs)
Early stopping at epoch 78 (no improvement for 3 epochs)
Early stopping at epoch 79 (no improvement for 3 epochs)
0.01   0.3796±0.0296   0.1170±0.0106   0.8409±0.0165
```

```
optimizer = optim.SGD(
    model.parameters(),
    lr=0.2, momentum=0.6,
    weight_decay=r
)
```

Βελτιστοποίηση μεθόδου ομαλοποίησης L2 (ridge) στο δίκτυο SimpleNet

Στα πλαίσια της προσπάθειας αποφυγής υπερπροσαρμογής και βελτίωσης της γενικευτικής ικανότητας του νευρωνικού μας δικτύου, επιλέξαμε την L2 regularization (ridge), η οποία προσθέτει στον στόχο εκπαίδευσης όρο proportional στο άθροισμα των τετραγώνων των βαρών. Σε αντίθεση με την L1, που επιτάσσει σπανιότητα στο μοντέλο και μπορεί να μηδενίσει σημαντικές παραμέτρους, η L2 περιορίζει τις ακραίες τιμές των βαρών χωρίς να απωθεί σε αμελητέα πολλούς από τους νευρώνες, διατηρώντας επαρκή χωρητικότητα για την περιγραφή των πολύπλοκων σχεσιακών προτύπων του dataset.

Η δοκιμή πραγματοποιήθηκε με σταθερή τοπολογία (hidden_size=35, activation=Tanh), learning rate η=0.1 και momentum m=0.6, σε 5-fold διασταυρούμενη επικύρωση, για τρεις τιμές του συντελεστή κανονικοποίησης r = weight_decay: 0.0001, 0.001 και 0.01. Σε κάθε πείραμα τρέξαμε έως 200 εποχές με early stopping (patience=3, min_delta=1e-4) και καταγράψαμε το Cross-Entropy (CE) loss, το MSE loss και την ακρίβεια (Accuracy) στο validation set.

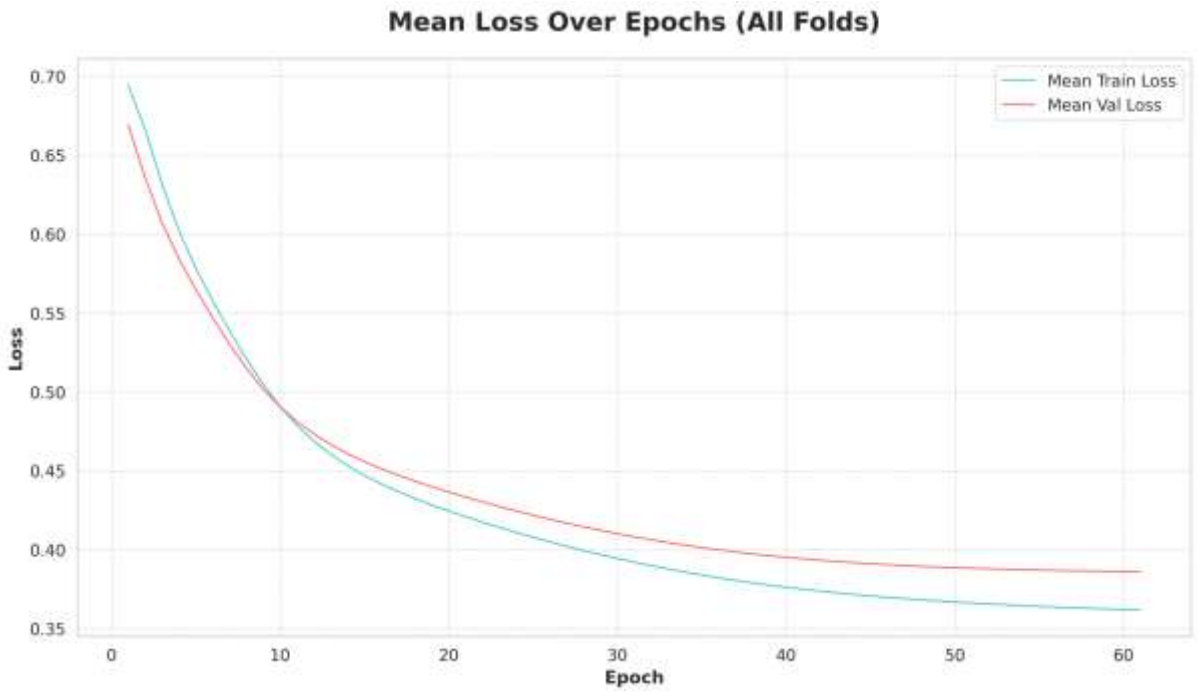
Τα αποτελέσματα συνοψίζονται στον παρακάτω πίνακα

Συντελεστής r	CE loss	MSE	Acc
0.0001	0.3703±0.0294	0.1135±0.0097	0.8418±0.0152
0.001	0.3744±0.0222	0.1142±0.0070	0.8441±0.0136
0.01	0.3796±0.0296	0.1170±0.0106	0.8409±0.0165

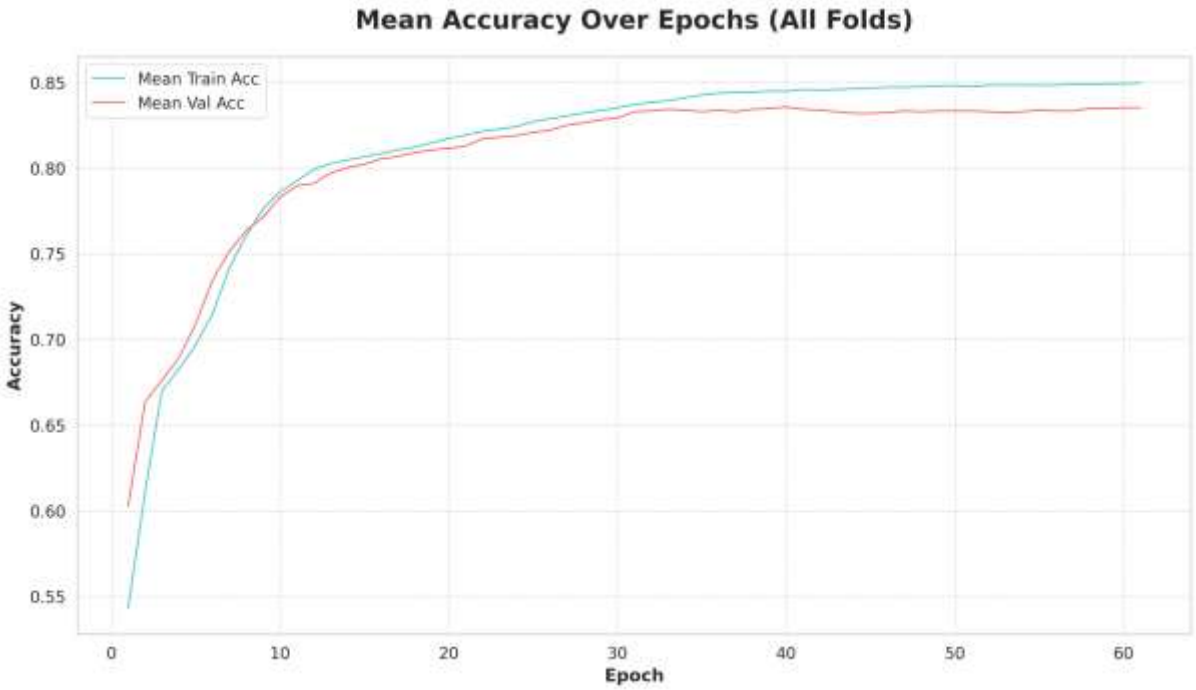
Plots:

r= 0.0001

Loss



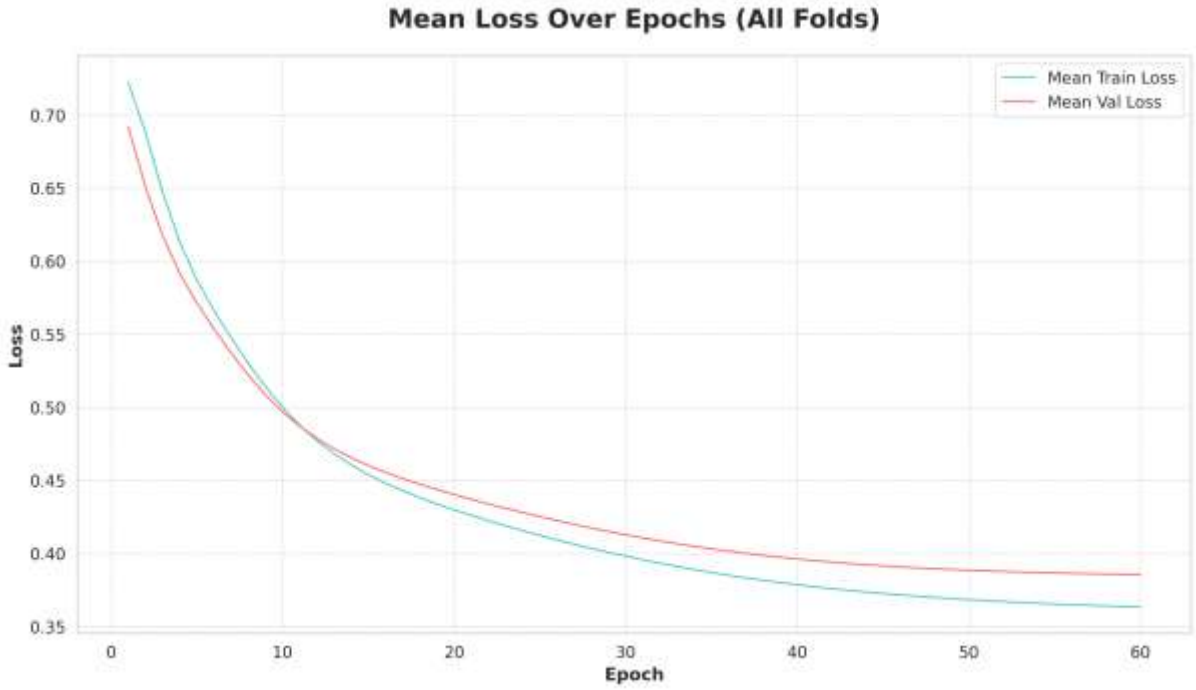
Accuracy



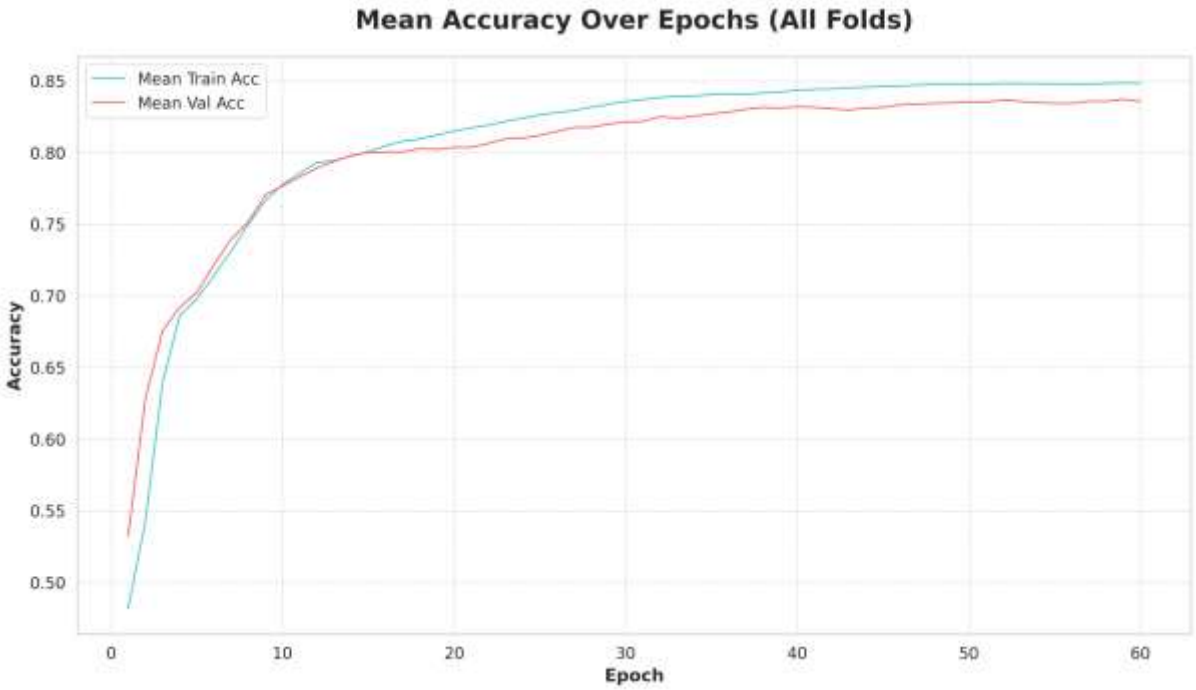
Plots:

r= 0.001

Loss



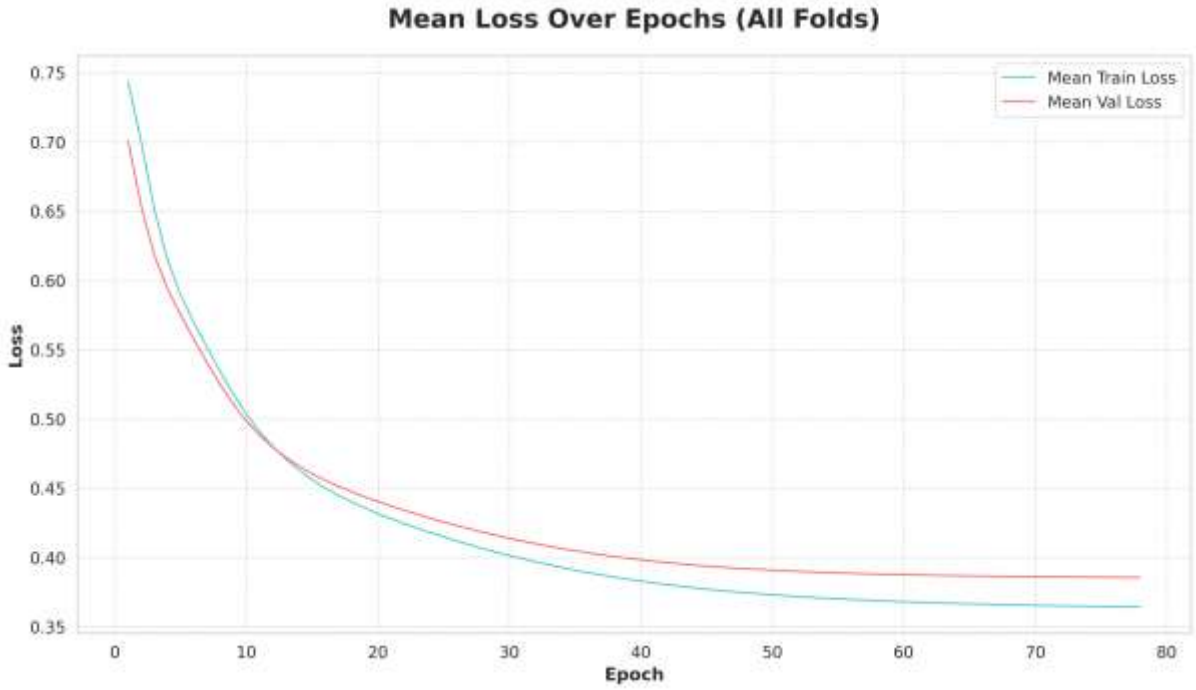
Accuracy



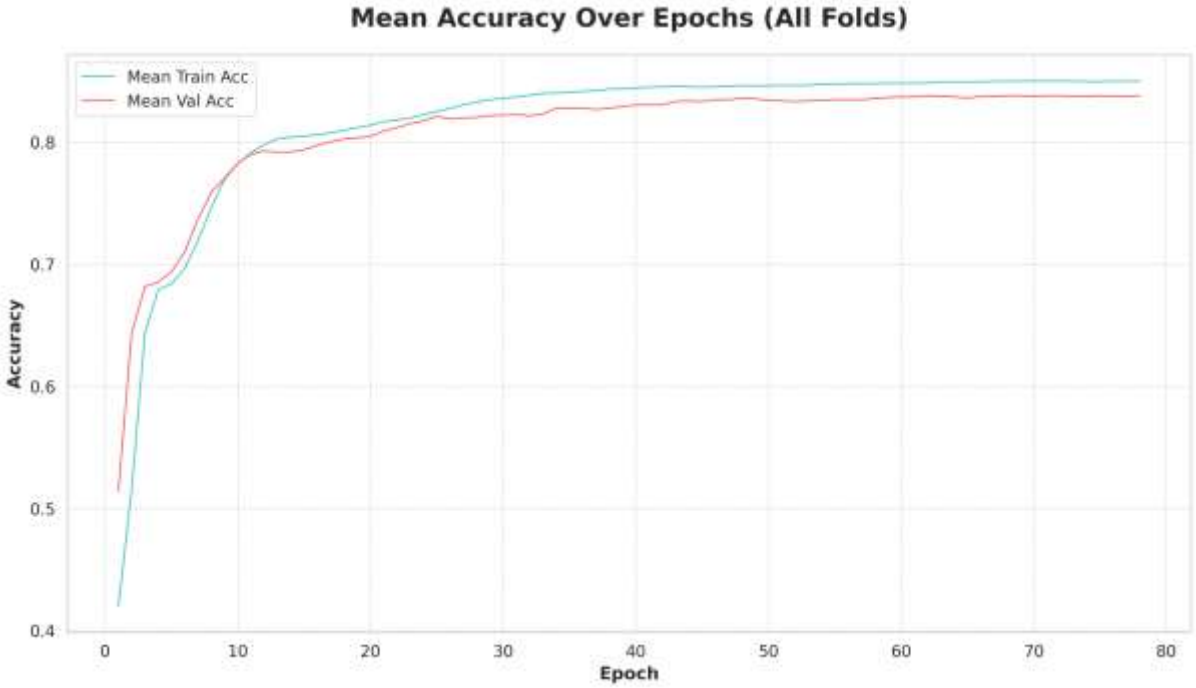
Plots:

r= 0.01

Loss



Accuracy



Σε κάθε πείραμα καταγράφηκε επίσης ο αριθμός της εποχής κατά την οποία ενεργοποιήθηκε το early stopping (ανά fold), γεγονός που αντανακλά την ταχύτητα σύγκλισης υπό την επίδραση του όρου κανονικοποίησης. Τα στοιχεία αυτά απεικονίζονται στις γραφικές παραστάσεις σύγκλισης, αποθηκευμένες ως loss_r{r}.png και acc_r{r}.png, όπου φαίνεται η πορεία του CE loss και της ακρίβειας ανά εποχή (μέσος όρος folds).

Από την εμπειρική ανάλυση προκύπτει ότι η τιμή r = 0.0001 είναι πολύ μικρή για ουσιαστική επίδραση· το δίκτυο εμφανίζει το χαμηλότερο CE loss (~~0.3703) αλλά με σχετικά υψηλή διακύμανση ανά fold (σ≈0.0294), ένδειξη ότι οι παράμετροι ακόμη επιτρέπεται να αυξάνονται σε «θορυβώδη» τιμές. Με r = 0.001, η κανονικοποίηση αποκτά ιδανικό μέγεθος ώστε να συγκρατεί τις υπερβολικές τιμές βαρών χωρίς να περιορίζει την ικανότητα απεικόνισης των δεδομένων. Το validation CE loss διατηρείται σε χαμηλά επίπεδα (~~0.3744) με σημαντικά μικρότερη τυπική απόκλιση (σ≈0.0222) και η ακρίβεια φτάνει το 84.41% με τη χαμηλότερη διακύμανση, υποδεικνύοντας ότι επιτυγχάνεται επιτυχημένος συμβιβασμός bias–variance. Αντίθετα, με r = 0.01 η υπερβολική αυστηρότητα στην τιμωρία των βαρών («underfitting») οδηγεί σε άνοδο του CE loss (~~0.3796) και μείωση της ακρίβειας (~~84.09%), καθώς οι παράμετροι δεν καταφέρνουν να προσαρμοστούν επαρκώς στα δεδομένα.

Συμπερασματικά, για το πρόβλημά μας η L2 regularization με συντελεστή r≈0.001 προσφέρει την καλύτερη ισορροπία ανάμεσα στην αποφυγή υπερπροσαρμογής και τη διατήρηση υψηλής απόδοσης, βελτιώνοντας σημαντικά τη σταθερότητα και τη γενίκευση του μοντέλου.

A5. Βαθύ Νευρωνικό Δίκτυο

Στα προηγούμενα στάδια (A2–A4) εντοπίσαμε τις βέλτιστες υπερπαραμέτρους για το μοντέλο μας: το δίκτυο SimpleNet με ένα κρυφό επίπεδο 35 νευρώνων, learning rate η = 0.1, momentum m = 0.6 και L2 regularization r = 0.001. Βασιζόμενοι σε αυτούς τους «δοκιμασμένους» παράγοντες, επεκτείναμε την αρχιτεκτονική του δικτύου σε βαθύτερα σχήματα, δοκιμάζοντας έως τρία κρυφά επίπεδα, με στόχο να διαπιστώσουμε αν η επιπλέον πολυπλοκότητα αποδίδει στην ταξινόμηση των δεδομένων Alzheimer.

Πιο συγκεκριμένα, δημιουργήσαμε την κλάση DeepNet που χτίζει δυναμικά ένα δίκτυο nn.Sequential από όσα επίπεδα «κρυφών» ορίσουμε. Στη συνάρτηση main ορίσαμε τρεις αρχιτεκτονικές:

1-hidden_35 :ένα επίπεδο 35 νευρώνων
2-hidden_35-25 : δύο επίπεδα σε φθίνουσα διάταξη (35 → 25)
3-hidden_35-70-25 : τρία επίπεδα με αύξουσα-φθίνουσα μορφή (35 70 → 25)

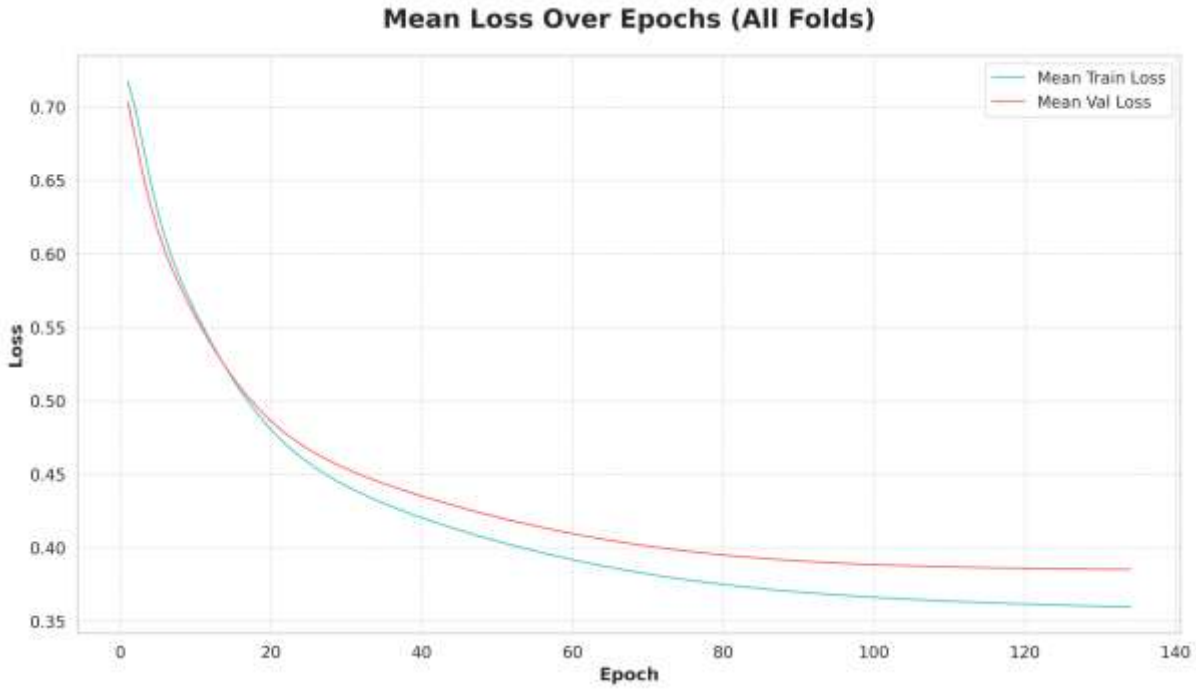
Διατηρήσαμε σταθερό learning rate=0.1, momentmm=0.6, L2 r=0.001 και εφάρμοσα 5-fold CV με early stopping (patience = 20, min_delta = 1e-3), έως 200 εποχες. Οι μέσοι όροι και οι τυπικές αποκλίσεις των μετρικών CE loss, MSE loss και accuracy στο validation set παρουσιάζονται ακολούθως:

Αρχιτεκτονική	CE_loss ± σ	MSE_loss ± σ	Acc ± σ
1-hidden_35	0.3846±0.0252	0.1187±0.0092	0.8399±0.0152
2-hidden_35→25	0.3813±0.0250	0.1179±0.0094	0.8339±0.0155
3-hidden_35→70→25	0.3606±0.0325	0.1095±0.0108	0.8492±0.0170

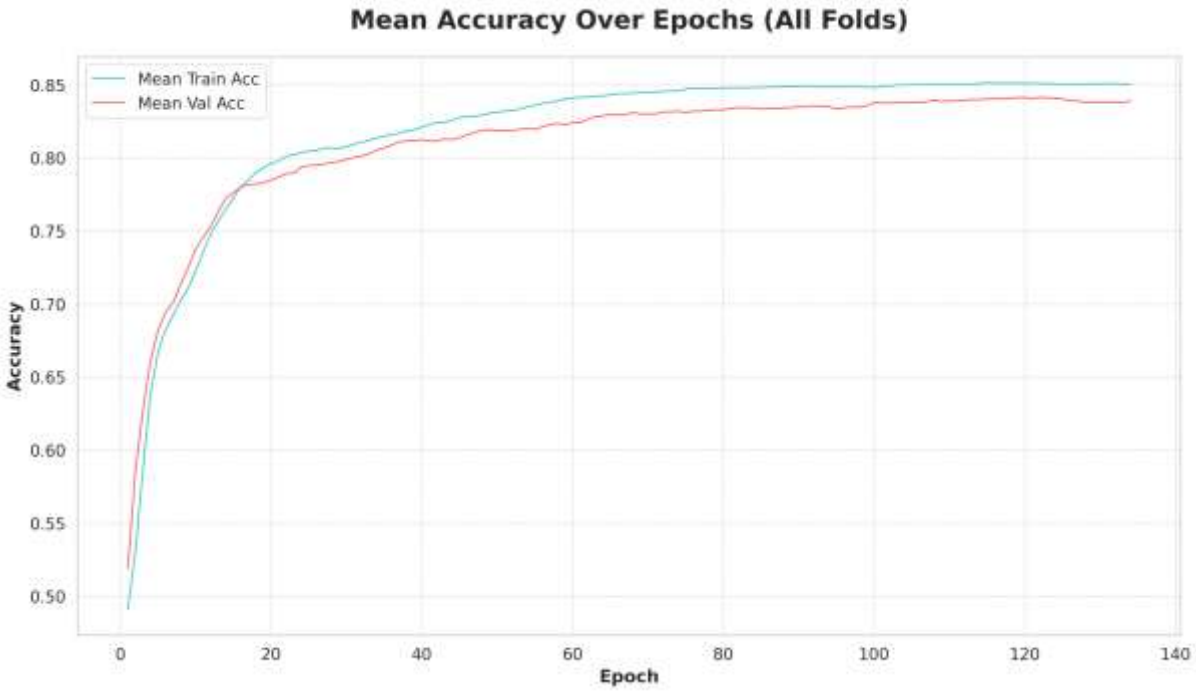
Plots:

1-hidden_35

Loss

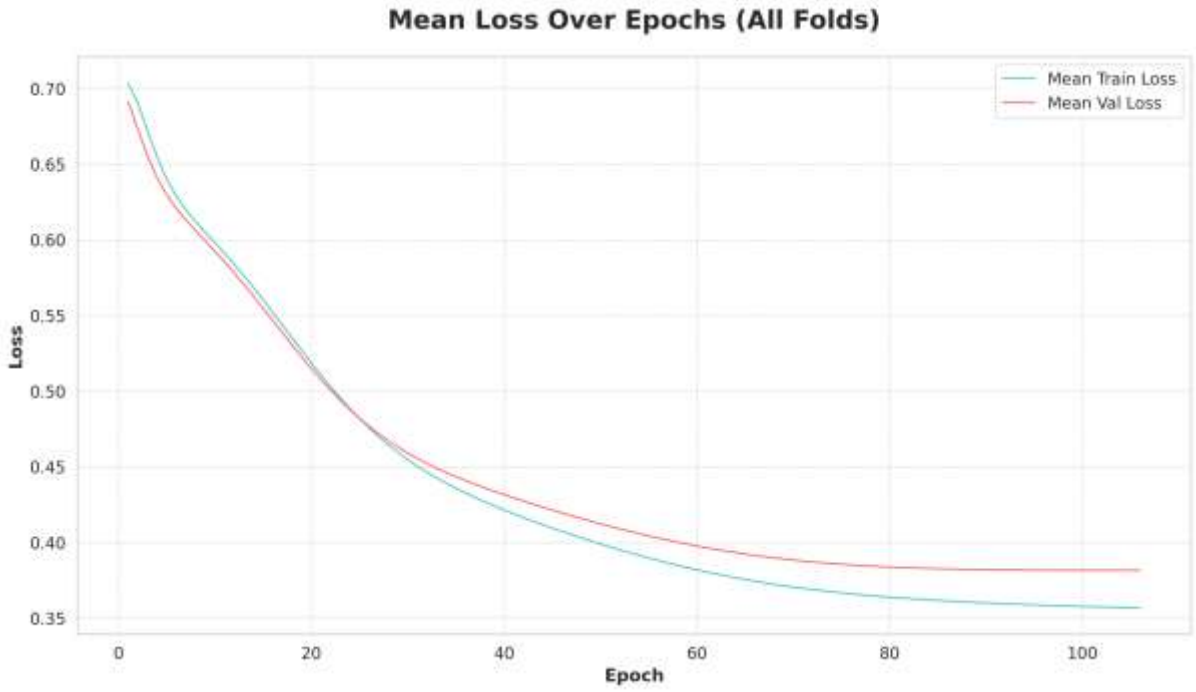


Accuracy

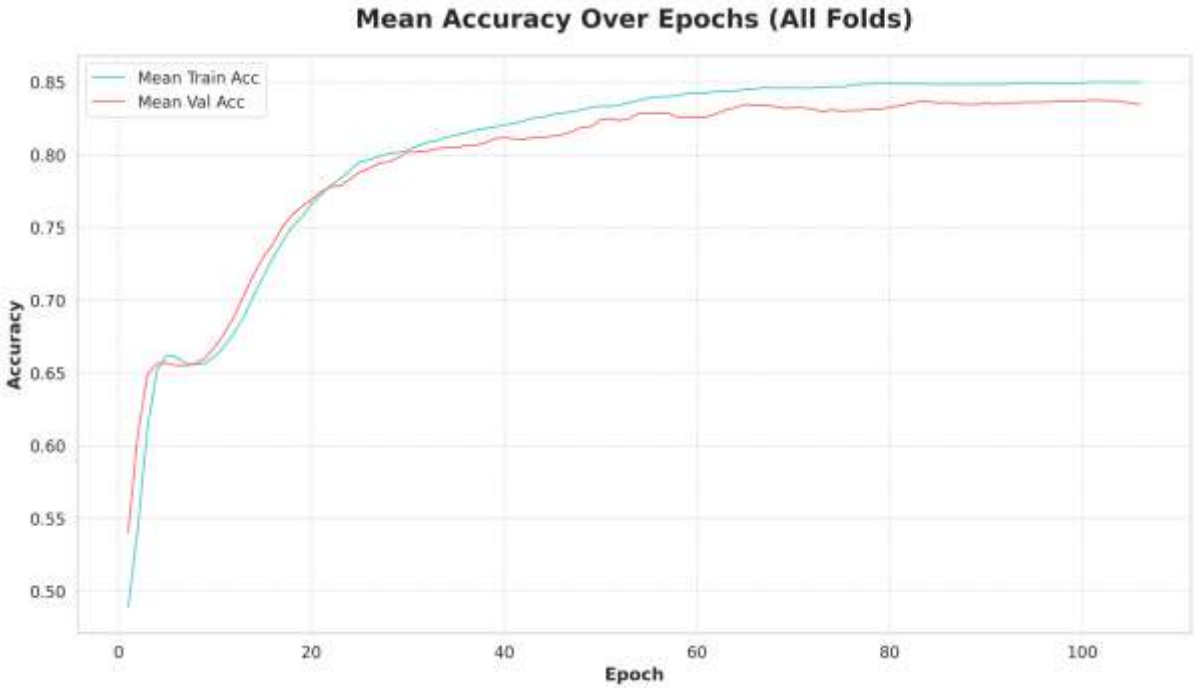


2 hidden_35→25

Loss

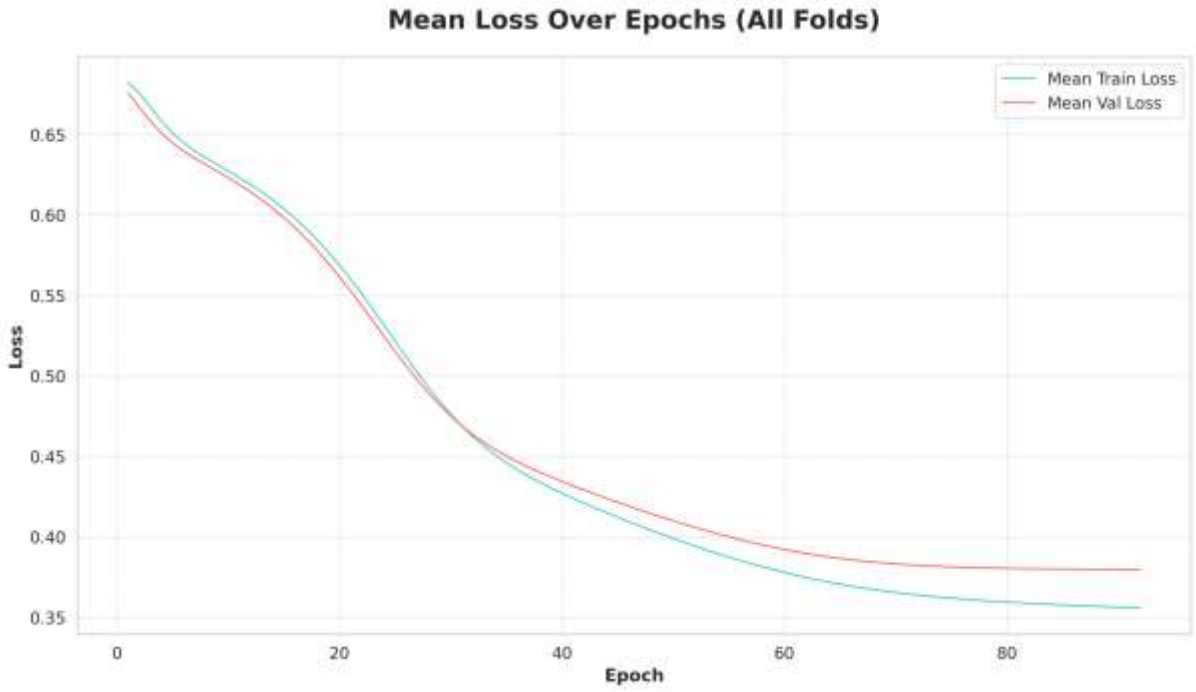


Accuracy

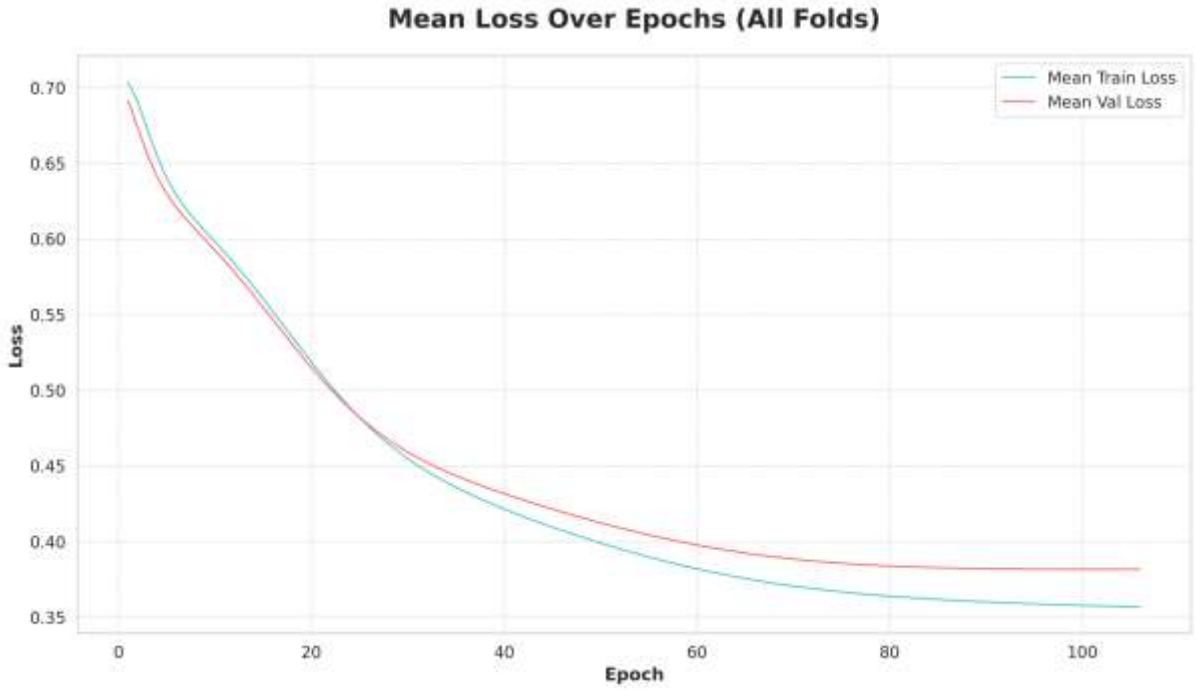


3 hidden_35→70→125

Loss



Accuracy



Σε κάθε case, το early stopping ενεργοποιήθηκε περίπου στο μέσο των εποχών (81–167) χωρίς σημάδια εκτροχιασμού. Από τα γραφήματα σύγκλισης (loss_<Αρχιτεκτονική>.png, acc_<Αρχιτεκτονική>.png) διακρίνεται ότι το τριών επιπέδων δίκτυο πέτυχε την χαμηλότερη τιμή CE loss και την υψηλότερη μέση ακρίβεια (~84.92 %), υποδεικνύοντας ικανότητα να εκμεταλλευτεί την πρόσθετη χωρητικότητα για τη σύλληψη πιο σύνθετων συσχετίσεων στα δεδομένα.

Ωστόσο, η βελτίωση σε σχέση με το μοντέλο ενός επιπέδου (η οποία ήταν οριακή, ~1 %) επιτυγχάνεται με κόστος: πολλαπλασιασμό παραμέτρων, αυξημένη υπολογιστική δαπάνη και κίνδυνο overfitting σε πολύ μικρά σύνολα δεδομένων. Αντίστοιχα, το μεσαίο σχήμα δύο επιπέδων δεν έφερε ουσιαστική υπεροχή έναντι των άλλων.

Συμπερασματικά, ενώ η προσθήκη τρίτου επιπέδου αποδίδει μια μικρή αύξηση της απόδοσης χάρη στην επιπλέον χωρητικότητα, οι αυξημένες απαιτήσεις σε δεδομένα και υπολογιστικό κόστος καθιστούν τη λύση αυτή λιγότερο πρακτική για εφαρμογές με περιορισμένους πόρους. Γενικά, η επιλογή του βάθους πρέπει να ισορροπεί το bias–variance trade-off, και σε αυτό το πρόβλημα το δίκτυο με δύο επίπεδα (35 → 25) ή ακόμη και το απλό με ένα επίπεδο προσφέρει τον βέλτιστο συμβιβασμό ανάμεσα στην απλότητα και την απόδοση.