**­­ A blue and red logo

Description automatically generated**

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΡΟΙΤΟΡ ΚΑΤΑΡΤΖΙΟΥ ΙΩΑΝ Π21077

ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ ΜΑΘΗΜΑΤΟΣ ΕΠΕΞΕΡΓΑΣΙΑ ΦΩΝΗΣ ΚΑΙ ΗΧΟΥ

**ΠΕΙΡΑΙΑΣ**

**Ιούνιος 2025**

**ΠΡΟΛΟΓΟΣ**

Ο στόχος της εργασίας είναι η ανάπτυξη ενός αυτόματου συστήματος που εντοπίζει και χωρίζει το ηχητικό σήμα σε τμήματα ομιλίας (foreground) και υποβάθρου (background). Το σύστημα δέχεται ως είσοδο ένα αρχείο ήχου και επιστρέφει ως έξοδο ένα αρχείο .csv με τα χρονικά όρια κάθε τμήματος, προσδιορίζοντας πότε υπάρχει ομιλία και πότε όχι.

**ΕΚΦΩΝΗΣΗ**

**Θέμα:**

Καλείστε να υλοποιήσετε ένα σύστημα που προχωρά στην κατάτμηση μιας πρότασης σε τμήματα σήματος ομιλίας (foreground) και σήματος υποβάθρου (background), χρησιμοποιώντας υποχρεωτικά έναν ταξινομητή background vs foreground (ανά frame) και εφαρμόζοντας στη συνέχεια μετα-επεξεργασία της ακολουθίας αποφάσεων του ταξινομητή. Τελικώς, δοθείσης μιας ηχογράφησης, το σύστημα επιστρέφει τα χρονικά όρια των τμημάτων σήματος ομιλίας και των τμημάτων σήματος υποβάθρου (σε δευτερόλεπτα).

To format των αποτελεσμάτων είναι ένα αρχείο csv, της μορφής:

Audiofile, start, end, class

File1, 0, 2.2, background

File1, 2.2, 4, foreground

File1, 4,10, background

…

Θα πρέπει να υλοποιήσετε και να σχολιάσετε τις επιδόσεις των παρακάτω ταξινομητών:

**Least Squares** και **MLP τριών επιπέδων** (αποφασίστε το πλήθος νευρώνων ανά στρώμα).

Για την εκπαίδευση, παρέχεται στο link <https://drive.google.com/drive/folders/1A-_ybw6sVtPYjrzkOa1rACvvCyqPqhWn?usp=share_link> ένας κατάλογος αρχείων train, με υποκαταλόγους ομιλίας (speech) και υποβάθρου (noise). Μπορείτε να χρησιμοποιήσετε ένα υποσύνολο των αρχείων αυτών. Για την τελική δοκιμή παρέχεται ένας κατάλογος test με ένα μεικτό αρχείο, καθώς και τα σχετικά transcriptions σε json format (υποκατάλογος transcriptions). Από τα json αρχεία, θα χρειαστεί να διατηρήσετε μόνο τα δεδομένα που αφορούν στο test αρχείο που δίνεται και να αγνοήσετε την υπόλοιπη πληροφορία.

Όλα τα αρχεία προέρχονται από γνωστά, δημόσια σύνολα δεδομένων (https://www.openslr.org, συλλογές MUSAN και CHiME) και ίσως χρειαστεί η τμηματική επεξεργασία τους, αναλόγως του τρόπου ανάλυσης που θα επιλέξετε. Ανεξαρτήτως προσέγγισης, προσπαθήστε να ποσοτικοποιήσετε κατάλληλα τις επιδόσεις των ταξινομητών.

**Σημειώσεις:**

Δεν μπορείτε να χρησιμοποιήσετε άλλου είδους νευρωνικό δίκτυο. Δεν είναι αποδεκτή η χρήση έτοιμων web services ή APIs για speech recognition. Δεν μπορείτε να χρησιμοποιήσετε transfer learning από ήδη εκπαιδευμένα δίκτυα. Οι αντίστοιχες λύσεις μηδενίζονται.

# ΠΕΡΙΕΧΟΜΕΝΑ

[ΠΕΡΙΕΧΟΜΕΝΑ 4](#_Toc200068192)

[1. ΕΙΣΑΓΩΓΗ 5](#_Toc200068193)

[1.1 ΣΤΟΧΟΙ ΕΡΓΑΣΙΑΣ 5](#_Toc200068194)

[2. ΠΕΡΙΓΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΟΣ 5](#_Toc200068195)

[2.1 ΠΑΡΟΥΣΙΑΣΗ ΣΚΕΨΗΣ 5](#_Toc200068196)

[2.2 Αποτελέσματα 6](#_Toc200068197)

[2.2.1 Αποτελέσματα Εκπαίδευσης 6](#_Toc200068198)

[2.2.2 Post-Processing 6](#_Toc200068199)

[2.2.3 Αποτελέσματα σε αρχείο test 7](#_Toc200068200)

[2.2.4 Σύγκριση με Ground Truth 7](#_Toc200068201)

[2.2.5 Ερμηνεία Αποτελεσμάτων 8](#_Toc200068202)

[2.2 ΑΝΑΛΥΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ 9](#_Toc200068203)

[2.2.1 ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ 9](#_Toc200068204)

[2.3.1 ΑΝΑΛΥΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ 9](#_Toc200068205)

[def extract\_features(audio, sr): 9](#_Toc200068206)

[def train\_lsc(features, labels): 10](#_Toc200068207)

[def train\_mlp(features, labels): 10](#_Toc200068208)

[3. ΕΠΙΔΕΙΞΗ ΤΗΣ ΛΥΣΗΣ 14](#_Toc200068209)

[ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ 19](#_Toc200068210)

# 1. ΕΙΣΑΓΩΓΗ

## ΣΤΟΧΟΙ ΕΡΓΑΣΙΑΣ

Στόχος της εργασίας είναι η δημιουργία ενός συστήματος που, με είσοδο ένα αρχείο ήχου, αναλύει το σήμα σε μικρά χρονικά πλαίσια (frames), εξάγει χαρακτηριστικά, ταξινομεί κάθε frame ως ομιλία ή υπόβαθρο χρησιμοποιώντας Least Squares και MLP ταξινομητές, εφαρμόζει μετα-επεξεργασία (median filtering) και τελικά επιστρέφει τις χρονικές περιοχές ομιλίας/υποβάθρου σε αρχείο .csv.

# 2. ΠΕΡΙΓΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

## 2.1 ΠΑΡΟΥΣΙΑΣΗ ΣΚΕΨΗΣ

Η προσέγγιση που ακολουθήθηκε για την τμηματοποίηση του σήματος ομιλίας βασίζεται στην εξαγωγή χαρακτηριστικών από κάθε frame του ηχητικού σήματος και στη συνέχεια στην εκπαίδευση δύο διαφορετικών ταξινομητών για την κατηγοριοποίηση κάθε frame ως speech ή noise. Αρχικά, το ηχητικό σήμα χωρίζεται σε frames διάρκειας περίπου 93ms με hop size 23ms, και για κάθε frame εξάγονται 15 χαρακτηριστικά που αποτελούνται από 13 MFCC συντελεστές, την ενέργεια (RMS) και το Zero Crossing Rate. Τα MFCCs αποτελούν 'αποτύπωμα' του φάσματος συχνοτήτων. Το MFCC 0 αντιπροσωπεύει τη συνολική ενέργεια ενώ τα MFCCs 1-12 καταγράφουν διαφορετικές περιοχές συχνοτήτων που αντιστοιχούν στα σημεία όπου η ανθρώπινη φωνή ακούγεται πιο έντονα, δηλαδή τα πιο χαρακτηριστικά σημεία της ομιλίας. Η ενέργεια του frame δείχνει πόσο δυνατό είναι αυτό το κομμάτι του σήματος, γιατί η ομιλία συνήθως έχει μεγαλύτερη και πιο μεταβαλλόμενη ενέργεια από το θόρυβο, ενώ το Zero Crossing Rate μετρά πόσο συχνά το σήμα διασχίζει το μηδέν, αφού η ομιλία διασχίζει το μηδέν πιο συχνά από τον θόρυβο λόγω των συχνότερων αλλαγών. **Τα χαρακτηριστικά αυτά επιλέχτηκαν μετά από έρευνα στο διαδίκτυο για παρόμοιες υλοποιήσεις που αναφέρονται στην ενότητα βιβλιογραφικές πηγές.**

Για την εκπαίδευση, αντί να χρησιμοποιηθούν ολόκληρα τα αρχεία ομιλίας που είναι μεγάλης διάρκειας το καθένα (σε σχέση με αυτά του θορύβου), εφαρμόστηκε τυχαία δειγματοληψία, παίρνοντας από κάθε αρχείο ομιλίας 4 τυχαία τμήματα των 5 δευτερολέπτων από διαφορετικά σημεία του αρχείου, κάτι που δίνει πολύ μεγαλύτερη ποικιλία στα δεδομένα εκπαίδευσης αντί να παίρνονται μόνο τα πρώτα δευτερόλεπτα από κάθε αρχείο. Συνολικά χρησιμοποιήθηκαν 50 αρχεία ομιλίας και 50 αρχεία ομιλίας από όλο το σύνολο δεδομένων εκπαίδευσης που δόθηκε από το Google Drive. Τα αρχεία θορύβου είναι πολύ μικρότερα οπότε χρησιμοποιούνται ολόκληρα τα αρχεία. Στη συνέχεια γίνεται κανονικοποίηση χαρακτηριστικών με StandardScaler για να έχουν όλα τα χαρακτηριστικά παρόμοια κλίμακα αφού για παράδειγμα η ενέργεια μπορεί να έχει μικρότερες τιμές από τους συντελεστές MFCC, πράγμα που θα οδηγήσει σε ανισορροπίες και επομένως σε κακή εκπαίδευση, καταλήγοντας σε χειρότερο accuracy (~63 vs ~80). Ύστερα, εξισορροπείται το dataset ώστε να υπάρχει ίσος αριθμός frames ομιλίας και θορύβου ώστε η εκπαίδευση να είναι αμερόληπτη, ενώ παράλληλα εφαρμόζεται και shuffle. Όσον αφορά τους ταξινομητές, υλοποιήθηκαν και οι δύο: ο Least Squares Classifier που χρησιμοποιεί γραμμική παλινδρόμηση για να βρει τη βέλτιστη γραμμική σχέση μεταξύ των χαρακτηριστικών και των ετικετών με την πρόβλεψη να γίνεται με τον τύπο πρόβλεψη = w₁×feature₁ + w₂× feature ₂ + ... + bias όπου αν πρόβλεψη > 0.5 τότε έχουμε ομιλία αλλιώς θόρυβο, και ο MLP 3 επιπέδων που είναι νευρωνικό δίκτυο με αρχιτεκτονική (64, 32, 16) νευρώνες στα τρία hidden layers και μπορεί να μάθει πιο σύνθετα μοτίβα και μη-γραμμικές σχέσεις μεταξύ των χαρακτηριστικών, πράγμα που ο γραμμικός ταξινομητής δεν μπορεί να κάνει.

Στη συνέχεια, παρατηρήθηκε πως η κατάτμηση frame-προς-frame δημιουργεί θόρυβο στις προβλέψεις καθώς το σύστημα μπορεί να αλλάζει πολύ γρήγορα μεταξύ ομιλίας και θορύβου δημιουργώντας πολλά μικρά τμήματα που δεν γίνεται να υπήρχαν στην πραγματικότητα. Για να αντιμετωπιστεί αυτό το πρόβλημα εφαρμόζεται median filtering που είναι είναι τεχνική εξομάλυνσης που αφαιρεί μεμονωμένα λάθη στις προβλέψεις. Αν σε μια ακολουθία από 0 και 1 υπάρχει ένας "θόρυβος" όπως 1, 1, 0, 1, 1, το φίλτρο μεσαίας τιμής (π.χ. με kernel 3) το μετατρέπει σε 1, 1, 1, 1, 1, αγνοώντας το μεμονωμένο 0. Έτσι, μειώνονται οι απότομες και μη πραγματικές αλλαγές στην κατηγοριοποίηση των frames. Μετά από δοκιμές με διάφορα μεγέθη kernel (3, 5, 7, 9) που έδιναν μικρότερη ακρίβεια ή πάρα πολλά τμήματα, επιλέχθηκαν (kernel size 41 για LSC και 81 για MLP) που αφαιρούν τα σφάλματα και δημιουργούν πιο ρεαλιστικά τμήματα που αντιστοιχούν στη φυσική διάρκεια της ομιλίας, καθώς διαπιστώθηκε ότι τα δεδομένα του πραγματικού κόσμου χρειάζονται πιο δυναμική αντιμετώπιση. Τέλος, το σύστημα αξιολογείται σε πραγματικό αρχείο ήχου (unseen) που περιέχει τόσο ομιλία όσο και θόρυβο, με τις προβλέψεις να συγκρίνονται με τα ground truth transcriptions που παρέχονται σε JSON format. Επειδή τα JSON αρχεία περιέχουν σύντομες απομονωμένες περιοχές ομιλίας, εφαρμόζεται συγχώνευση που ενώνει τα τμήματα ομιλίας που είναι κοντά μεταξύ τους (κενό < 0.5s) για να δημιουργηθούν πιο ρεαλιστικά κομμάτια αξιολόγησης. Η τελική έξοδος του συστήματος είναι αρχείο CSV με τη μορφή που απαιτεί η εργασία, περιέχοντας τα χρονικά όρια και την κατηγορία (foreground/background) κάθε τμήματος.

## 2.2 Αποτελέσματα

2.2.1 Αποτελέσματα Εκπαίδευσης

Η εκπαίδευση έγινε σε ισορροπημένο dataset με 86.400 συνολικά samples (43.200 frames ομιλίας και 43.200 frames θορύβου), που αντιστοιχεί σε περίπου 1003 δευτερόλεπτα ήχου για κάθε κατηγορία. Ο Least Squares Classifier είχε ακρίβεια 83.8% στα δεδομένα εκπαίδευσης, ενώ το MLP είχε ακρίβεια 99.0% με training loss 0.0254 μετά από 45 iterations. Η μεγάλη διαφορά στην απόδοση δείχνει την υπεροχή του νευρωνικού δικτύου στο να συλλαμβάνει σύνθετα μοτίβα που ο γραμμικός ταξινομητής δεν μπορεί να εντοπίσει (αφού είναι γραμμικός).

2.2.2 Post-Processing

Το post-processing με median filtering επηρέασε τα αποτελέσματα αφού για τον LSC, η αρχική ακρίβεια 83.8% μειώθηκε καθώς αυξανόταν το μέγεθος του kernel (από 67.1% με kernel 3 έως 59.2% με kernel 9), ενώ παράλληλα μειώθηκε ο αριθμός των τμημάτων (από 21.464 σε 11.700). Το MLP είχε παρόμοια συμπεριφορά, με την ακρίβεια να πέφτει από 74.7% σε 63.5%, αλλά με μείωση των τμημάτων από 21.578 σε 11.738. Αυτή η μείωση της ακρίβειας στα training data δικαιολογείται από το γεγονός ότι τα δεδομένα εκπαίδευσης είναι "καθαρά" (pure speech ή pure noise), ενώ το median filtering προσπαθεί να σταθεροποιήσει τις μεταβάσεις για πραγματικά δεδομένα (που δεν είναι "καθαρά" πάντα).

2.2.3 Αποτελέσματα σε αρχείο test

Στο test αρχείο (S01\_U04.CH4.wav) διάρκειας 9544.2 δευτερολέπτων (περίπου 2.65 ώρες), οι ταξινομητές δεν είχαν την ίδια ακρίβεια. Ο LSC δημιούργησε 1717 τμήματα και εντόπισε 648.3 δευτερόλεπτα ομιλίας (6.8% του συνολικού χρόνου), ενώ το MLP δημιούργησε 384 τμήματα και εντόπισε 9305.6 δευτερόλεπτα ομιλίας (97.5% του συνολικού χρόνου). Η τεράστια διαφορά στον αριθμό τμημάτων (1717 vs 384) δείχνει ότι το MLP με το post-processing με (kernel 81) φτιάχνει πιο αληθινούς διαχωρισμούς.



***Εικόνα 1. Aποτελέσματα***

2.2.4 Σύγκριση με Ground Truth

Το JSON αρχείο (S01.json) είχε 5797 αρχικά τμήματα που συγχωνεύθηκαν σε 2801 ρεαλιστικά τμήματα. Ο LSC είχε F1-score μόλις 0.151, με υψηλό precision (88.9%) αλλά χαμηλό recall (8.2%), δηλαδή όταν προέβλεπε ομιλία είχε σχεδόν πάντα δίκιο, αλλά έχανε το 92% της πραγματικής ομιλίας. Αντίθετα, το MLP είχε F1-score 0.847, με precision 74.2% και recall 98.7%. Αυτό σημαίνει ότι το MLP εντοπίζει σχεδόν όλη την ομιλία (98.7%) αλλά μερικές φορές συμπεριλαμβάνει και κάποιο θόρυβο.

2.2.5 Ερμηνεία Αποτελεσμάτων

Η υπεροχή του MLP (F1: 0.847 vs 0.151) βγάζει νόημα, αφού η ομιλία σε πραγματικές συνθήκες έχει σύνθετα χαρακτηριστικά που ο γραμμικός ταξινομητής δεν μπορεί να συλλάβει, ενώ το MLP φτιάχνει μη-γραμμικές σχέσεις, δημιουργώντας σύνθετα μοτίβα. Επιπλέον, τόσο το scaling όσο και το post-processing, φάνηκαν κρίσιμα για την αύξηση την ακρίβειας.

**Να σημειωθεί ότι για όλα τα παραπάνω υπάρχουν οι αντίστοιχες πηγές στην ενότητα 4, Βιβλιογραφικές αναφορές.**

## 2.2 ΑΝΑΛΥΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Για την ανάπτυξη των προγραμμάτων, χρησιμοποιήθηκε η Pyhton. Αρχικά θα δοθούν κάποιες βασικές πληροφορίες και στη συνέχεια θα αναλυθούν τα κυρίως προγράμματα.

## ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ

ΟΙ βασικές βιβλιοθήκες που χρησιμοποιήθηκαν είναι η librosa για επεξεργασία του ήχου και εξαγωγή χαρακτηριστικών, η NumPy για αριθμητικές πράξεις, η scikit-learn για την υλοποίηση των ταξινομητών, την κανονικοποίηση των χαρακτηριστικών και την αξιολόγηση του μοντέλου, η pandas για αποθήκευση των αποτελεσμάτων σε αρχεία CSV, και η scipy για το φιλτράρισμα με χρήση του median filter.

## ΑΝΑΛΥΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

### def extract\_features(audio, sr):

Η συνάρτηση αυτή εξάγει χαρακτηριστικά από ένα ηχητικό σήμα και αποτελεί τον πυρήνα της επεξεργασίας. Λαμβάνει ως όρισμα το audio signal σε μορφή numpy array και το sample rate (ρυθμός δειγματοληψίας). Η διαδικασία εκτελείται ως εξής:

1. Χρησιμοποιεί τη librosa.feature.mfcc() με παραμέτρους n\_mfcc=13 (αριθμός MFCC συντελεστών), hop\_length=512, n\_fft=2048. Το hop\_length=512 σημαίνει ότι κάθε frame απέχει 23ms από το επόμενο, ενώ το n\_fft=2048 δίνει frames διάρκειας 93ms.
2. Υπολογίζει το RMS (Root Mean Square) energy, με τη librosa, για κάθε frame με ίδιες παραμέτρους, ώστε να είναι συγχρονισμένο με τα MFCCs.
3. Υπολογίζει το Zero Crossing Rate, με τη librosa, που μετρά πόσο συχνά το σήμα αλλάζει πρόσημο, με ίδιες παραμέτρους, ώστε να είναι συγχρονισμένο με τα MFCCs.
4. Χρησιμοποιεί np.vstack() για να ενώσει όλα τα features σε έναν κάθετο πίνακα και κάνει transpose (.T) για να έχει τη μορφή (frames, features).

*Φόρτωση και Προετοιμασία Δεδομένων:*

1. Από κάθε αρχείο ομιλίας μεγάλης διάρκειας, παίρνει 4 τυχαία segments των 5 δευτερολέπτων. Αυτό γίνεται μέσω:
   * librosa.get\_duration() για να βρει τη συνολική διάρκεια
   * np.random.uniform() για τυχαίο offset
   * librosa.load(duration=5, offset=random\_offset) για φόρτωση συγκεκριμένου τμήματος
2. Τα αρχεία θορύβου είναι μικρά, οπότε φορτώνονται εξ ολοκλήρου με librosa.load(noise\_file).
3. Εφαρμόζει StandardScaler για κανονικοποίηση. Χωρίς αυτό, τα χαρακτηριστικά θα είχαν διαφορετική κλίμακα, οδηγώντας σε κακή εκπαίδευση.
4. Shuffle για να βελτιωθεί η γενίκευση.
5. Παίρνει ίσο αριθμό frames από κάθε κλάση χρησιμοποιώντας min\_samples = *min(np.sum(y == 1), np.sum(y == 0))* και ώστε να εξισορροπήσουν οι δύο κλάσεις του συνόλου εκπαίδευσης.

### def train\_lsc(features, labels):

Η συνάρτηση εκπαιδεύει τον Least Squares Classifier με χρήση της LinearRegression() από τη βιβλιοθήκη sklearn. Η εκπαίδευση πραγματοποιείται με την εντολή *model.fit(features, labels)*, για την εύρεση των βέλτιστων βαρών. Ο ταξινομητής κάνει πρόβλεψη με τη γραμμική εξίσωση: prediction = w₁×feature₁ + w₂×feature₂ + … + bias. Η αξιολόγηση ακρίβειας γίνεται ως:

* Αν prediction > 0.5 → Speech (1)
* Αν prediction ≤ 0.5 → Noise (0)

### def train\_mlp(features, labels):

Η συνάρτηση εκπαιδεύει ένα MLP με 3 κρυφά επίπεδα και αρχιτεκτονική (64, 32, 16) νευρώνες, χρησιμοποιεί ReLU ως συνάρτηση ενεργοποίησης και τον βελτιστοποιητή Adam που προσαρμόζει δυναμικά το learning rate. Για να αποφύγει το overfitting, εφαρμόζει early stopping με validation\_fraction 0.15, σταματώντας την εκπαίδευση αν δεν υπάρχει βελτίωση. Η εκπαίδευση διαρκεί το πολύ 100 εποχές ή μέχρι να σταθεροποιηθεί η απόδοση. Επιπροσθέτως, Η predict\_proba() επιστρέφει πιθανότητες για κάθε κλάση — [P(noise), P(speech)]. Με [:, 1] κρατάμε μόνο την πιθανότητα να είναι speech. Έπειτα, εφαρμόζουμε threshold, όμοια με το LSC.

*def* *post\_processing\_predict & test\_post\_processing(predictions, labels):*

To Post-Processing Median Filter χρησιμοποιείται για να διορθώσει το πρόβλημα της αστάθειας στις προβλέψεις των ταξινομητών. Οι προβλέψεις γίνονται κάθε 23ms (ένα frame), αλλά στην πράξη η ανθρώπινη ομιλία αλλάζει πολύ πιο αργά: τα φωνήματα διαρκούν 100–300ms, οι λέξεις 500–2000ms, και οι παύσεις μεταξύ τους 100–500ms. Αυτό σημαίνει πως χωρίς post-processing, το αποτέλεσμα περιέχει “θόρυβο”, δηλαδή απότομες εναλλαγές 0→1→0 που δεν αντιστοιχούν σε πραγματική αλλαγή ομιλίας.

Συγκεκριμένα, Το medfilt() εφαρμόζει median filtering: σε κάθε frame παίρνει το “μεσαίο” στοιχείο από ένα παράθυρο γειτονικών frames (π.χ. 3 ή 5) και το χρησιμοποιεί για εξομάλυνση. Για να λειτουργήσει, μετατρέπουμε πρώτα τις προβλέψεις σε float. Αφού εφαρμοστεί το φίλτρο, επαναφέρουμε τα δεδομένα σε binary (0 ή 1) με threshold 0.5. Έτσι, διαγράφονται απομονωμένες λανθασμένες προβλέψεις.

Η test\_post\_processing() δοκιμάζει διάφορα kernel sizes (3, 5, 7, 9) και μετρά:

* Accuracy: πόσο βελτιώνεται η πρόβλεψη συνολικά.
* Segments: πόσα διαφορετικά τμήματα ανιχνεύονται αφού λιγότερα σημαίνουν πιο πραγματική συμπεριφορά.

Σε training δεδομένα χρησιμοποιείται μικρό kernel (π.χ. 5 ή 7), ενώ σε testing, χρησιμοποιείται μεγάλο kernel (π.χ. 41 ή 81), αφού αυτό μειώνει τα σφάλματα και κάνει τις προβλέψεις πιο σταθερές. Μάλιστα, πριν το φίλτρο υπήρχαν 234 segments και accuracy 0.823. Μετά το φίλτρο (kernel 5) υπάρχουν στα 45 segments και το accuracy ανεβαίνει στο 0.856.

*def convert\_frames\_to\_segments(predictions, hop\_length=512, sr=22050):*

Η συνάρτηση convert\_frames\_to\_segments() μετατρέπει τις προβλέψεις frame by frameκαρέ σε χρονικά τμήματα με ετικέτες “foreground” ή “background”. Αυτό είναι σημαντικό γιατί αντί να έχουμε μόνο προβλέψεις ανά 23ms, δημιουργούμε κατανοητά και αξιοποιήσιμα χρονικά segments, π.χ. “από 1.2s έως 3.4s είναι ομιλία”.

Ξεκινάμε από το πρώτο frame και παρακολουθούμε πότε αλλάζει η πρόβλεψη από 0 σε 1 ή το αντίστροφο. Κάθε φορά που αλλάζει, καταγράφουμε το χρονικό διάστημα που προηγήθηκε, μετατρέποντας τα frame indices σε χρόνο (σε δευτερόλεπτα), με τον τύπο frame × hop\_length / sample\_rate. Στο τέλος προσθέτουμε και το τελευταίο τμήμα που φτάνει μέχρι το τέλος του αρχείου. Το αποτέλεσμα είναι μια λίστα από segments με τη μορφή (start\_time, end\_time, class\_name).

*def save\_to\_csv(segments, audio\_filename, output\_file):*

Η συνάρτηση save\_to\_csv() αποθηκεύει τα segments που έχουν εξαχθεί σε μορφή CSV. Για κάθε segment (με χρονικά όρια και ετικέτα), δημιουργείται μια γραμμή με τις εξής στήλες: Audiofile, start, end, class.

Η αποθήκευση γίνεται με τη βιβλιοθήκη pandas. Το round(start, 3) και round(end, 3) εξασφαλίζει ακρίβεια 1ms, που είναι απαραίτητη για σωστό συγχρονισμό με το ground truth. Το αποτέλεσμα είναι αρχείο της μορφής:

Audiofile,start,end,class

S01\_U04.CH4,0.000,0.069,foreground

S01\_U04.CH4,0.069,0.116,background

S01\_U04.CH4,0.116,0.162,foreground

*Testing με αρχείο test*

1. Φορτώνεται το test αρχείο και χρησιμοποιείται η librosa.load() με sr=None ώστε να διατηρηθεί το αυθεντικό sample rate του αρχείου. Αυτό διασφαλίζει ότι οι χρονικοί υπολογισμοί (σε δευτερόλεπτα) θα είναι ακριβείς:

2. Τα χαρακτηριστικά εξάγονται με την extract\_features() και στη συνέχεια κανονικοποιούνται με τον ίδιο scaler που εκπαιδεύτηκε στα training δεδομένα.

3. Λαμβάνονται οι πιθανότητες από τους ταξινομητές.

4. Εφαρμόζεται Median Filtering, χρησιμοποιώντας μεγαλύτερο kernel (81 frames ≈ 1.86s), ώστε να εξαλειφθούν μικρά σφάλματα.

5. Οι προβλέψεις μετατρέπονται σε segments με χρονικά όρια, και αποθηκεύονται σε .csv αρχεία

6. Υπολογίζεται το συνολικό μήκος των τμημάτων που έχουν χαρακτηριστεί ως “foreground” και εκφράζεται ως ποσοστό της συνολικής διάρκειας του ήχου.

***def load\_ground\_truth(json\_file, audio\_filename):***

Αυτή η συνάρτηση αναλαμβάνει να φορτώσει τα δεδομένα (ground truth) από τα JSON transcript αρχεία. Ο στόχος της είναι να εντοπίσει μόνο τις χρονικές περιόδους που περιέχουν ομιλία στο συγκεκριμένο αρχείο ήχου και να αγνοήσει τις υπόλοιπες εγγραφές.

Για να το κάνει αυτό:

* Παίρνει το session ID από το όνομα του αρχείου ήχου (π.χ. "S01" από το "S01\_U04.CH4.wav")
* Διαβάζει το JSON αρχείο και φιλτράρει τις εγγραφές ώστε να κρατήσει μόνο αυτές που ανήκουν στο σωστό session
* Μετατρέπει τα timestamps από μορφή "HH:MM:SS.ms" σε δευτερόλεπτα με χρήση της βοηθητικής συνάρτησης time\_to\_seconds()
* Δημιουργεί λίστα από tuples (start\_time, end\_time, "foreground") με βάση το διάστημα κάθε λέξης που βρέθηκε

***def time\_to\_seconds(time\_str):***

Παίρνει string "Ώρες:Λεπτά:Δευτερόλεπτα.χιλιοστά" και επιστρέφει float με τα συνολικά δευτερόλεπτα. Π.χ. "00:01:20.500" → 80.5 δευτερόλεπτα. Χρησιμοποιεί string split με ':' και float() για να συνδυάσει τα επιμέρους μέρη.

***def merge\_speech\_segments(segments, gap\_threshold=0.5):***

Αυτή η συνάρτηση ενώνει συνεχόμενα segments ομιλίας που έχουν μικρό κενό (<0.5s) μεταξύ τους.

* Κάνει loop από το δεύτερο segment και μετά
* Κρατάει "τρέχον" segment και αν το επόμενο είναι της ίδιας κατηγορίας (foreground) και ξεκινά σύντομα, επεκτείνει το τρέχον
* Αλλιώς, το αποθηκεύει και ξεκινά νέο

Για τα χρονικά διαστήματα μεταξύ των speech segments (δηλαδή τα κενά), αν η απόσταση μεταξύ δύο διαστημάτων είναι μεγαλύτερη από 0.1s, προστίθεται ένα νέο segment τύπου “background”.

* Γίνεται loop με enumerate() πάνω στα συγχωνευμένα segments
* Υπολογίζεται η διαφορά του start του επόμενου με το end του προηγούμενου
* Αν είναι πάνω από 0.1s, προστίθεται νέο tuple (prev\_end, start, "background")

Έτσι εξασφαλίζεται ότι ολόκληρη η διάρκεια καλύπτεται από κάποιο label.

***def segments\_to\_frames(segments, total\_duration, frame\_rate=43.066):***

Αυτή η συνάρτηση μετατρέπει τα ground truth χρονικά διαστήματα σε binary πίνακα frame-level ετικετών.

* Υπολογίζει τον αριθμό frames από int(total\_duration \* frame\_rate)
* Δημιουργεί μηδενικό array με np.zeros(total\_frames)
* Για κάθε segment foreground, μετατρέπει τα timestamps σε index (π.χ. int(start \* frame\_rate)) και θέτει τα αντίστοιχα στοιχεία σε 1

Έτσι παράγει ένα array όπως [0,0,1,1,1,0,...] το οποίο ευθυγραμμίζεται με τα frame-based predictions του μοντέλου.

Το frame\_rate είναι ίδιο με sr/hop\_length (δηλαδή 22050/512 ≈ 43.066).

*def evaluate\_segments(predicted\_segments, ground\_truth\_segments, name):*

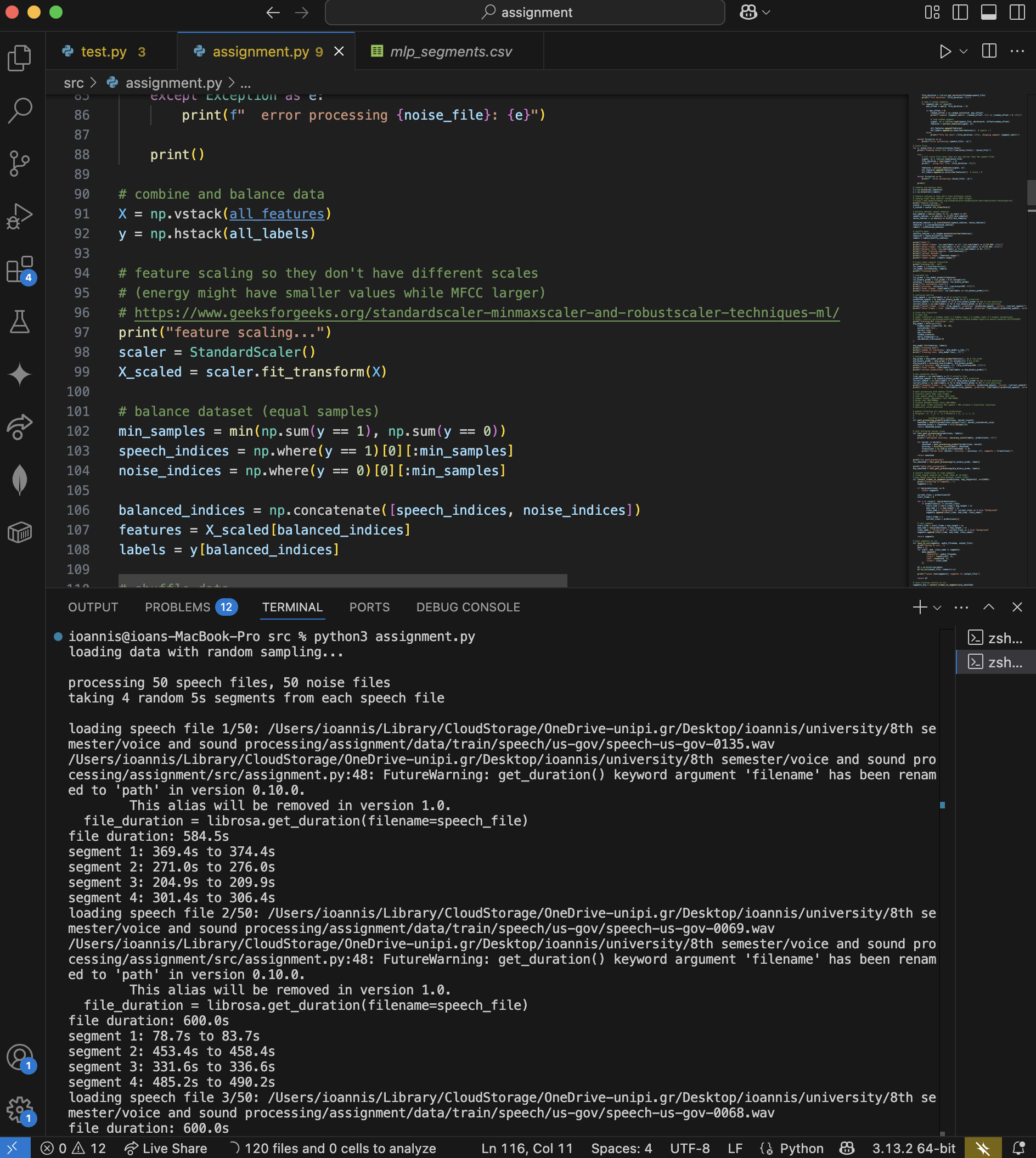
Οι μετρικές που υπολογίζονται είναι οι βασικές:

* **accuracy**: μετρά πόσα frames ταξινομήθηκαν σωστά, δηλαδή (σωστά speech + σωστά noise) / σύνολο.
* **precision**: από όλα τα frames που το μοντέλο είπε ότι είναι speech, πόσα ήταν πραγματικά speech (αποφυγή false positives).
* **recall**: από όλα τα πραγματικά speech frames, πόσα εντόπισε σωστά το μοντέλο. (αποφυγή false negatives).
* **f1-score**: ο αρμονικός μέσος του precision και recall. Δείχνει πόσο καλά ισορροπεί το μοντέλο μεταξύ false positives και false negatives.

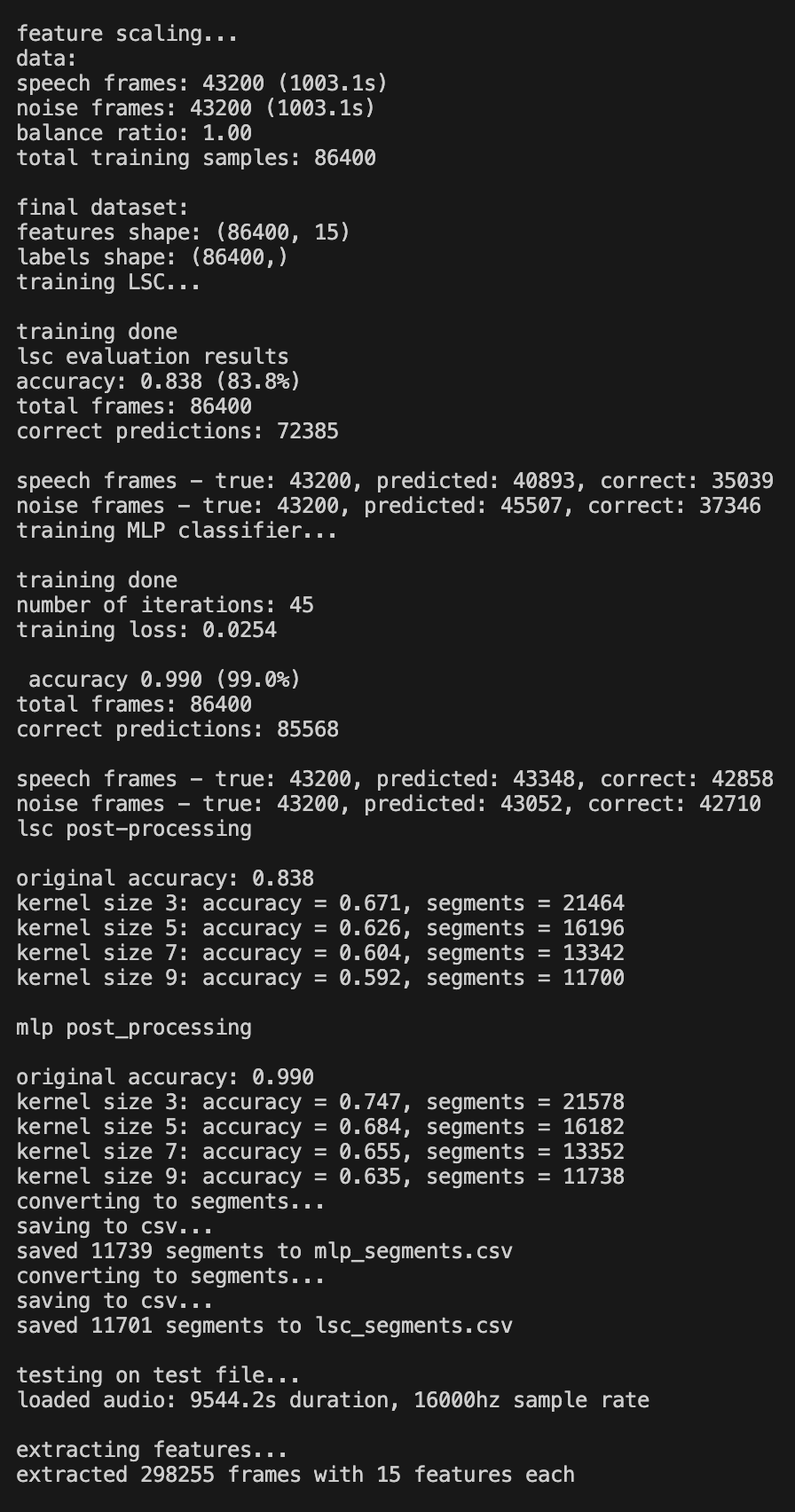
Το correct\_speech είναι ο αριθμός των frames που είναι και predicted και true speech (1 & 1). Το pred\_speech είναι όλα τα frames που το μοντέλο είπε speech, ενώ το true\_speech είναι αυτά που είναι πραγματικά speech στο ground truth.

1. ΕΠΙΔΕΙΞΗ ΤΗΣ ΛΥΣΗΣ

Για να αναδείξουμε την λειτουργία των προγραμμάτων θα τρέξει το προγράμματα *assignment.py.*



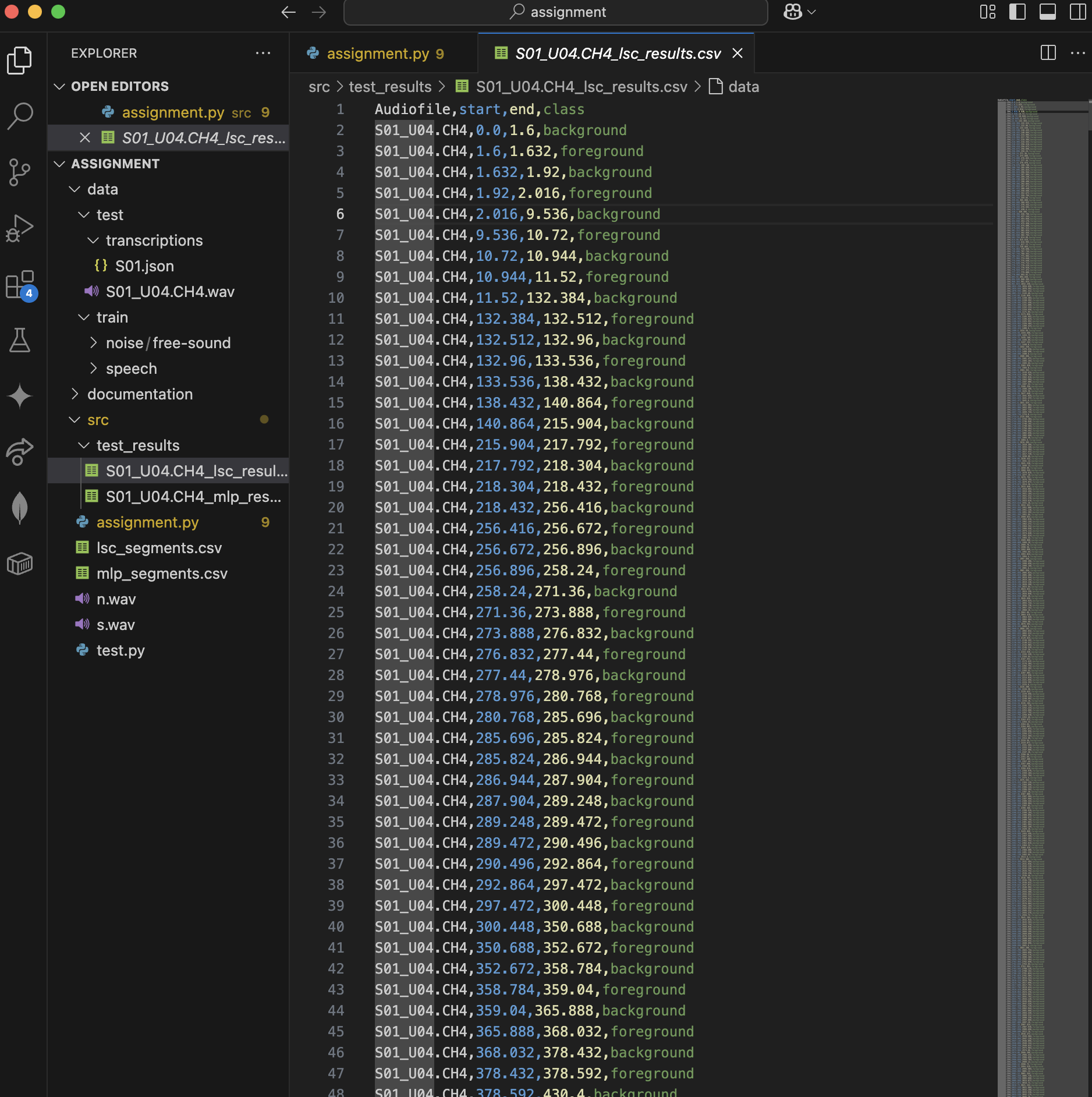
***Εικόνα 2. Έναρξη διαδικασίας***



***Εικόνα 3. Εξέλιξη διαδικασίας***



***Εικόνα 4. Κατάληξη διαδικασίας***



***Εικόνα 5. Τμήμα csv αρχείου της ταξινόμησης του LSC***

A screenshot of a computer

AI-generated content may be incorrect.

***Εικόνα 6. Τμήμα csv αρχείου της ταξινόμησης του MLP***

# ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

1. Σύγγραμμα και Διαφάνειες μαθήματος
2. <https://youtube.com/playlist?list=PL-wATfeyAMNqIee7cH3q1bh4QJFAaeNv0&si=jlicnHc-OeQ2UMt4>
3. <https://en.wikipedia.org/wiki/Mel-frequency_cepstrum>
4. <https://musicinformationretrieval.com/energy.html>
5. <https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.medfilt.html>
6. <https://maelfabien.github.io/machinelearning/Speech4/#rolling-window>
7. <https://www.geeksforgeeks.org/standardscaler-minmaxscaler-and-robustscaler-techniques-ml/>
8. https://www.geeksforgeeks.org/least-square-method/