ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάθημα Προπτυχιακών Σπουδών:

Επεξεργασία Σημάτων Φωνής και Ήχου

Ακαδημαϊκό Έτος: 2023 - 2024

Εξάμηνο: 8ο

Ατομική Εργασία

Φοιτητής:

Απόστολος Σιαμπάνης Π20173

Πίνακας Περιεχομένων

Εκφώνηση

Εργασία Εαρινού Εξαμήνου 2023-2024

- Ημερομηνία παράδοσης: Ημερομηνία εξέτασης του μαθήματος, ώρα 23:59μμ. Η εργασία συμμετέχει με βάρος 40% στον τελικό βαθμό.
- Η εργασία είναι ατομική και παραδίδεται μέσω της πλατφόρμας e-class. Αποδεκτές γλώσσες είναι οι Matlab και Python. Στα παραδοτέα συμπεριλαμβάνονται:

a) η τεκμηρίωση της εργασίας σε αρχείο pdf, στην πρώτη σελίδα της οποίας αναγράφεται το ονοματεπώνυμο του φοιτητή/φοιτήτριας και ο ΑΜ του/της. Θα μηδενιστούν οι εργασίες που δεν περιέχουν τεκμηρίωση ή στοιχεία φοιτητή/φοιτήτριας.

b) τα αρχεία source code σε ένα συμπιεσμένο αρχείο με όνομα source2023.zip (ή .rar ή άλλη σχετική κατάληξη).

c) οποιαδήποτε άλλα συνοδευτικά αρχεία κρίνετε απαραίτητα σε ένα συμπιεσμένο αρχείο με το όνομα auxiliary2023.zip (ή .rar ή άλλη σχετική κατάληξη).

- Η εργασία θα είναι η ίδια και τον Σεπτέμβριο.
- Η αντιγραφή ή η χρήση generative bots οδηγεί σε μηδενισμό.

Α. Καλείστε να υλοποιήσετε ένα σύστημα που προχωρά στην κατάτμηση μιας πρότασης σε λέξεις, χρησιμοποιώντας **υποχρεωτικά** έναν ταξινομητή background vs foreground της επιλογής σας. Δηλαδή, δοθείσης μιας ηχογράφησης ενός ομιλητή, το σύστημα επιστρέφει τα χρονικά όρια των λέξεων που ειπώθηκαν (σε δευτερόλεπτα). Επίσης, παρέχετε συνοδευτικό πρόγραμμα το οποίο αναπαράγει τις λέξεις που εντοπίστηκαν. Το πλήθος των λέξεων στην πρόταση δεν είναι εκ των προτέρων γνωστό, αλλά μπορείτε να υποθέσετε ότι υπάρχει μικρό διάστημα απουσίας ομιλίας μεταξύ λέξεων.

Θα πρέπει να υλοποιήσετε και να συγκρίνετε τις επιδόσεις των παρακάτω ταξινομητών: Least Squares, SVM, RNN και MLP τριών επιπέδων (προσδιορίστε τον αριθμό νευρώνων ανά επίπεδο). Η σύγκριση θα γίνει όπως προβλέπεται σε δυαδικά συστήματα ταξινόμησης.

Β. Από τις λέξεις που προκύπτουν, υπολογίστε τη μέση θεμελιώδη συχνότητα του ομιλητή.

- Πρέπει να εξηγήσετε ποια δεδομένα χρησιμοποιήσατε κατά τον έλεγχο και την εκπαίδευση του συστήματος. Αν είναι δικά σας, πώς τα δημιουργήσατε και αν είναι open source, πώς αξιοποιούνται.
- Προσπαθήστε να μην εξαρτάται το σύστημα από τα χαρακτηριστικά της φωνής του ομιλητή, αλλά να είναι όσο το δυνατόν ανεξάρτητο ομιλητή.

Προσοχή!!!: Δεν μπορείτε να χρησιμοποιήσετε **συνελικτικά** νευρωνικά δίκτυα. **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.
Καλή επιτυχία!

Ερώτημα Α

Στο αρχείο **classifiers.py** έχουν υλοποιηθεί οι μέθοδοί που αφορούν τα ταξινομητές (classifiers). Ακολουθούν τα βήματα που ακολουθήθηκαν:

Ξεκινώντας έχουμε ως στόχο να δημιουργήσουμε τα μοντέλα των ταξινομητών. Στο κώδικα υπάρχουν υλοποιήσεις για τους ταξηνομητές SVM, MLP (τριών επιπέδων), Least Squares και RNN.

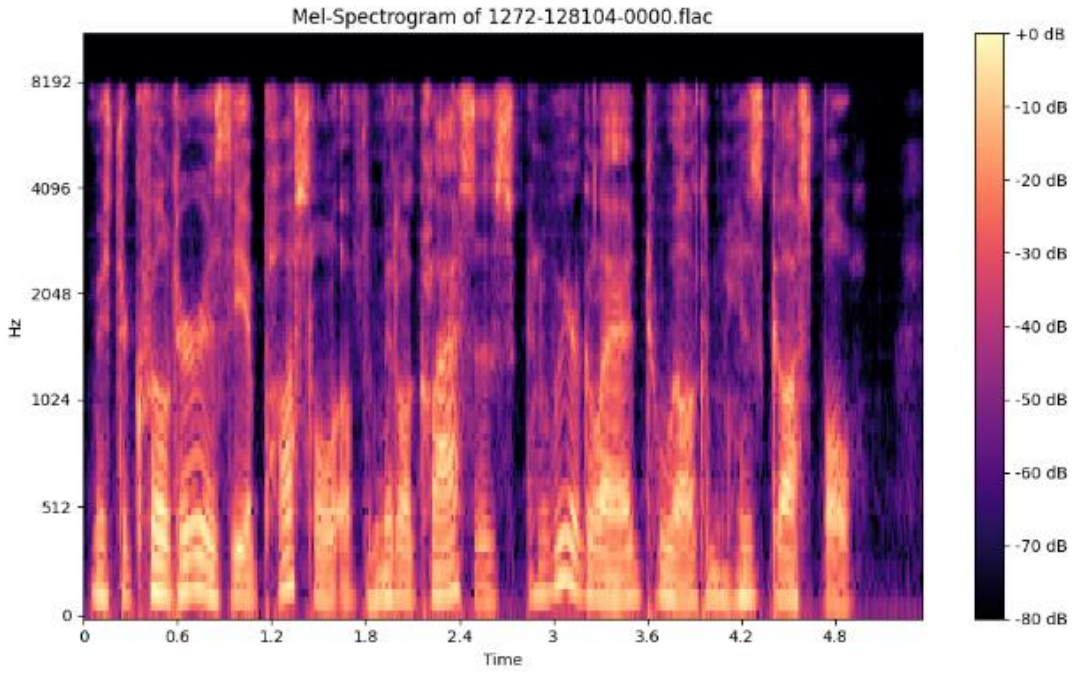
Θα χρησιμοποιηθούν δύο διαφορετικά datasets (LibriSpeech-clean και ESC-50), ένα για τις φωνές πολλαπλών ομιλητών θεωρώντας τα ως ήχους στο προσκήνιο (foreground) και ένα για τους ήχους στο παρασκήνιο, δηλαδή ήχους υποβάθρου (background).

Υλοποίηση Ταξινομητών

Αρχικά θέλουμε να ταξινομήσουμε κάθε frame σε foreground και background, για γίνει αυτό θα σαρώσουμε το σήμα του ήχου με την τεχνική του κινούμενου παραθύρου. Σε κάθε παράθυρο θα εξάγουμε κάποιες μετρήσεις ώστε να μπορέσουμε να καταλήξουμε σε ταξινόμηση, αυτή η εξαγωγή μετρήσεων (feature extraction) βρίσκεται στο αρχείο **feature_extraction.py**. Από την λογική που υιοθετεί ο κάθε ταξινομητής για την εκπαίδευση του, μπορούμε να εξάγουμε το συμπέρασμα πως δεν απαιτείται η διάρκεια του κάθε αρχείου ήχου να είναι ίδια για τους SVM, MLP και Least Squares. Αντίθετα για τον ταξινομητής RNN απαιτείται η ίδια διάρκεια αρχείων ήχου. Έτσι, η διαδικασία κατά την ανάγνωση των αρχείων διαφέρει λίγο για τον συγκεκριμένο ταξινομητή.

Πιο συγκεκριμένα για την διαδικασία του feature extraction, διαβάζουμε τα αρχεία από τα datasets με την χρήση της βιβλιοθήκης Librosa και για τους τρις πρώτους ταξινομητές όπως αναφέραμε παραπάνω θα φορτώσουμε (librosa.load) ολόκληρα τα αρχεία με ένα offset ίσο με 0.5 ώστε να εξαλείψουμε το μικρό κενό στην αρχή κάθε ηχητικού. Θα χρησιμοποιήσουμε παράθυρο μήκους 512, με βήμα μήκους 256, έτσι διατηρούμε μια υπερκάλυψη 50%. Παράλληλα, γίνεται χρήση φίλτρων mel πλήθους 96. Έτσι, μπορούμε να εξάγουμε τα χαρακτηριστικά για κάθε παράθυρο ήχου πλέον (librosa.feature.melspectrogram) επιστάφοντας τα σε διανύσματα και θα μετατρέψουμε τις τιμές τους σε decibel (librosa.power_to_db). Με παρόμοιο τρόπο εξάγουμε και τα χαρακτηριστικά των παραθύρων για τον ταξινομητή RNN, με την διαφορά ότι χρησιμοποιούμε ίσο μήκος ήχων, 2 δευτερόλεπτα, μόνο για τα αρχεία που έχουν διάρκεια μεγαλύτερη των 4 δευτερολέπτων. Στο σημείο αυτό, ανάλογα το αρχείο ταξινομούμε όλα τα frames με την ετικέτα (label) είτε 0 αν είναι αρχείο ήχου υποβάθρου είτε 1 αν είναι αρχείο ομιλίας, ώστε να γνωρίζουν οι ταξινομητές τις αντιστοιχίες μεταξύ των

ετικετών και των διανυσμάτων. Στο σημείο αυτό αν θελήσουμε μπορούμε να χρησιμοποιήσουμε την μέθοδο plot_spectrogram, η οποία υλοποιείτε στο αρχείο **audio.py**.



Έτσι, μπορούμε να περάσουμε στο στάδιο της εκπαίδευσης του κάθε ταξινομητή.

Ταξινομητής SVM

Θα ξεκινήσουμε με την υλοποίηση του ταξινομητή SVM. Έχοντας, τα διανύσματα και τις ετικέτες, μπορούμε να τα ομαδοποιήσουμε και να εισάγουμε 2 μεγάλους πίνακες αντίστοιχα με τα διανύσματα και τις ετικέτες όλων των αρχείων που έχουμε βρει τα χαρακτηριστικά τους. Δημιουργούμε ένα μοντέλο τύπου LinearSVC, με την παράμετρο random_state ίση με 1 ώστε να ελέγξουμε την ψευδοτυχαία γεννήτρια αριθμών και του εισάγουμε των 2 πίνακες ώστε να αρχίσει την εκπαίδευση του με την .fit μέθοδο. Όταν ολοκληρωθεί η εκπαίδευση αποθηκεύουμε το μοντέλο σε ένα αρχείο με την ονομασία “svm_classifier.joblib”, με την βοήθεια της βιβλιοθήκης joblib, καθώς αυτή αναλαμβάνει την αποθήκευση των αρχείων μας.

Ταξινομητής MLP

Θα υλοποιήσουμε έναν ταξινομητή MLP τριών επιπέδων. Έχοντας, τα διανύσματα και τις ετικέτες, μπορούμε να τα ομαδοποιήσουμε και να εισάγουμε 2 μεγάλους πίνακες αντίστοιχα με τα διανύσματα και τις ετικέτες όλων των αρχείων που έχουμε βρει τα χαρακτηριστικά τους. Δημιουργούμε ένα μοντέλο τύπου MLPClassifier, με τις παραμέτρους hidden_layer_size(512, 256, 128) ορίζοντας έτσι των αριθμό των νευρώνων ανά επίπεδο (512 νευρώνες για το πρώτο επίπεδο, 256 για το δεύτερο επίπεδο και 128 για το τρίτο), max_iter ίση με 100 δηλώνοντας το μέγιστο πλήθος επαναλήψεων (100 επαναλήψεις), random_state ίση με 1 ώστε να ελέγξουμε την ψευδοτυχαία γεννήτρια αριθμών και early_stopping ίση με true ώστε να αφήσει το 10% των δεδομένων εισαγωγής για την διαδικασία της επαλήθευσης (Validation). Τώρα αφού ορίσαμε το μοντέλο, μπορούμε να εισάγουμε τους 2 πίνακες ώστε να αρχίσει την εκπαίδευση του με την .fit μέθοδο. Όταν ολοκληρωθεί η εκπαίδευση αποθηκεύουμε το μοντέλο σε ένα αρχείο με την ονομασία “mlp_classifier.joblib”.

Ταξινομητής Least Squares

Για την υλοποίηση ενός ταξινομητή Least Square, έχοντας τα διανύσματα και τις ετικέτες, μπορούμε να τα ομαδοποιήσουμε και να εισάγουμε 2 μεγάλους πίνακες αντίστοιχα με τα διανύσματα και τις ετικέτες όλων των αρχείων που έχουμε βρει τα χαρακτηριστικά τους. Στο πίνακα με τα χαρακτηριστικά μετατοπίζουμε το εσωτερικό του προσθέτοντας μια στήλη από 1, αυτές οι τιμές ονομάζονται bias. Τώρα μετατρέπουμε και τους 2 αυτούς πίνακες σε tensor μορφή - γίνεται χρήση του tensorflow – και χρησιμοποιώντας την μέθοδο tf.linalg.lstsq μπορούμε να εισάγουμε τους 2 πίνακες ώστε να αρχίσει την εκπαίδευση του. Με την ολοκλήρωση της διαδικασίας μας επιστρέφει τις παραμέτρους του μοντέλου με ονομασία theta και τώρα μπορούμε να αποθηκεύσουμε αυτές τις τιμές του μοντέλου σε ένα αρχείο με την ονομασία “ theta_least_squares_classifier.joblib”.

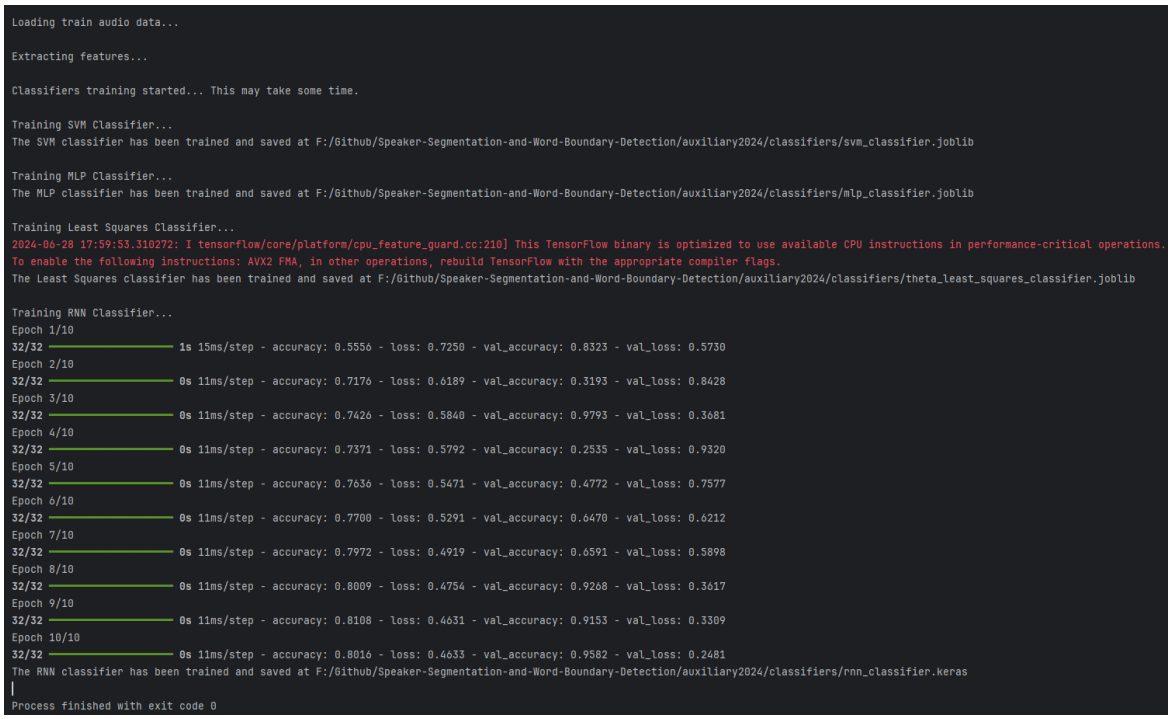
Ταξινομητής RNN

Για την υλοποίηση ενός ταξινομητή RNN, έχοντας τα διανύσματα και τις ετικέτες, τα μετασχηματίζουμε ώστε να έχουμε το κατάλληλο σχήμα εισόδου για το μοντέλο. Τώρα δημιουργούμε το μοντέλο χρησιμοποιώντας αρχικά το tf.keras.models.Sequential, στην συνέχεια προσθέτουμε τα αναδιαμορφωμένα δεδομένα εισόδου και προσθέτουμε ένα layer τύπου SimpleRNN με τις παραμέτρους units=32 ορίζοντας έτσι τις διαστάσεις της εξόδου, activation=’sigmoid’ ώστε οι τιμές που θα επιστρέφει να είναι πιθανότητές και στο όριο μεταξύ 0 και 1, return_sequence=true ώστε να επιστραφούν διανύσματα κατάστασης για κάθε χρονική στιγμή και seed=1 ώστε να ελέγξουμε την ψευδοτυχαία γεννήτρια αριθμών. Ύστερα, προσθέτουμε ένα ακόμα layer τύπου

Dense με τις παραμέτρους units=1 ορίζοντας έτσι τις διαστάσεις της εξόδου και activation=’sigmoid’ ώστε οι τιμές που θα επιστρέφει να είναι πιθανότητές. Τέλος, μεταγλωττίζω το μοντέλο με την μέθοδο compile και τις παραμέτρους loss='binary_crossentropy' καθώς προερχόμαστε από ένα νευρώνα - το output του Dense layer – και metrics=[‘accuracy’] για να επιστρέφει την ακρίβεια του μοντέλου κάθε χρονική στιγμή κατά την εκπαίδευση. Τώρα που έχουμε δημιουργήσει το μοντέλο μπορούμε να χρησιμοποιήσουμε την μέθοδο .fit ώστε να εκπαιδεύσουμε τον καταχωρητή έχοντας ως παραμέτρους τους 2 πίνακες με τα χαρακτηριστικά και τις ετικέτες, epochs=10, batch_size=32 και validation_split=0.2 ώστε να αφήσει το 20% των δεδομένων εισαγωγής για την διαδικασία της επαλήθευσης (Validation). Όταν ολοκληρωθεί η εκπαίδευση αποθηκεύουμε το μοντέλο σε ένα αρχείο με την ονομασία “ rnn_classifier.keras”, καθώς το ίδιο το tensorflow παρέχει μέθοδο αποθήκευσης του μοντέλου.

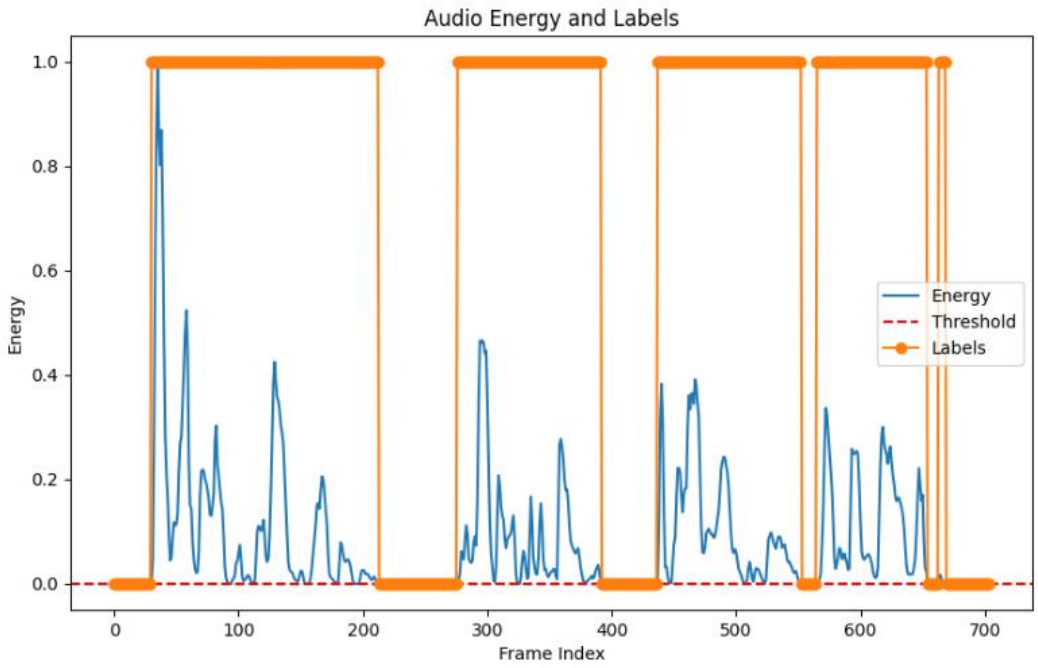
Παράδειγμα Εκτέλεσης classifiers.py

Εκτελώντας το αρχείο **classifiers.py** έχουμε:



Αναγνώριση Λέξεων

Εκτελώντας το αρχείο **word_detector.py** το πρόγραμμα περιμένει από το χρήστη να δώσει σαν είσοδο το πλήρες όνομα αρχείου, μαζί με το είδος του αρχείου. Το αρχείο αυτό θα πρέπει να βρίσκεται στον φάκελο auxiliary2024/dataset/testing. Για το ηχητικό αυτό πρέπει να βρεθεί η πραγματική ακολουθία ετικετών (ground truth) έτσι καλούμε την συνάρτηση label_test_audio, η οποία βρίσκεται στο αρχείο **labeling.py**. Η μέθοδος αυτή με την σειρά της εκτελεί μια συνάρτηση Ενέργειας, η οποία βρίσκεται στο αρχείο energy.py. Χρησιμοποιώντας τις τιμές αυτές και ένα threshold, μπορούμε να βάλουμε τις ετικέτες ανά παράθυρο στο αρχείο ήχου. Με την ολοκλήρωση της διαδικασίας αυτής αν θέλουμε μπορούμε να εμφανίσουμε ένα γράφημα με τις ετικέτες και την ενέργεια του αρχείου.



Στη συνέχεια αφού ο χρήστης επιλέξει το είδος του καταχωρητή που θέλει να χρησιμοποιήσει για την δοκιμή του.

Ταξινομητής SVM

Επιλέγοντας ο χρήστης τον ταξινομητή SVM, αρχικά φορτώνουμε με την μέθοδο load_svm_classifier – που βρίσκεται στο αρχείο classifiers.py - το μοντέλο του ταξινομητή που είναι αποθηκευμένο στο μονοπάτι (path) auxiliary2024/classifiers/svm_classifier.joblib. Ανακτώντας λοιπόν το μοντέλο μπορούμε πλέον να το εκτελέσουμε ώστε να προβλέψει τις τιμές για το αρχείο ανά παράθυρο, για αυτό χρησιμοποιείται η μέθοδος **predict_audio_labels**. Αρχικά βρίσκουμε τα χαρακτηριστικά (features) του αρχείου με την **extract_features**, στο αρχείο feature_extraction.py, και μετατρέπουμε τις τιμές σε decibel με την μέθοδο db_conversion, στο αρχείο audio.py. Τώρα χρησιμοποιούμε το μοντέλο του ταξινομητή για να προβλέψει τις τιμές ανά παράθυρο. Η διαδικασία αυτή ονομάζεται **Classification per feature extraction** και τώρα μπορούμε να ακολουθεί το επόμενο βήμα που είναι η διόρθωση μερικών λανθασμένων αποφάσεων από τον ταξινομητή. Η διαδικασία αυτή ονομάζεται **post processing** και σε αυτή χρησιμοποιούμε ένα φίλτρο μεσαίας τιμής (median filter) καλώντας την συνάρτηση **apply_median_filter**, η οποία βρίσκεται στο αρχείο post_processing.py και χρησιμοποιώντας την μέθοδο signal.medfilt της βιβλιοθήκης scipy. Η τιμή του kernel_size έχει οριστεί ίση με 5, αυτό σημαίνει ότι μπορούμε να διορθώσουμε έως και 2 ετικέτες ανά ομάδα. Τώρα έχοντας την τελική ακολουθία μπορούμε να την διατρέξουμε και να βρούμε τα όρια των λέξεων που εντόπισε ο ταξινομητής. Αυτό υλοποιείται με την μέθοδο **find_word_boundaries**, όπου βρίσκει τις λέξεις και ύστερα τις εμφανίζουμε στο χρήστη. Ύστερα, με την χρήση της **accuracy_score** από την βιβλιοθήκη sklearn.metrics συγκρίνουμε την ακολουθία που προέβλεψε ο ταξινομητής με την πραγματική ακολουθία (ground truth). Τέλος, με την μέθοδο **play_audio_boundaries** η οποία βρίσκεται στο αρχείο audio_player.py αναπαράγουμε τις λέξεις που βρήκε ο ταξινομητής από το αρχείο ήχου που εισήγαγε ο χρήστης, με την βοήθεια της pydub βιβλιοθήκης και του προγράμματος ffmpeg που αναλαμβάνει την αναπαραγωγή του ηχητικού σήματος.

Ταξινομητής MLP

Επιλέγοντας ο χρήστης τον ταξινομητή MLP τριών επιπέδων, αρχικά φορτώνουμε με την μέθοδο load_mlp_classifier – που βρίσκεται στο αρχείο classifiers.py - το μοντέλο του ταξινομητή που είναι αποθηκευμένο στο μονοπάτι (path) auxiliary2024/classifiers/mlp_classifier.joblib. Ανακτώντας λοιπόν το μοντέλο μπορούμε πλέον να το εκτελέσουμε ώστε να προβλέψει τις τιμές για το αρχείο ανά παράθυρο, για αυτό χρησιμοποιείται η μέθοδος **predict_audio_labels**. Αρχικά βρίσκουμε τα χαρακτηριστικά (features) του αρχείου με την **extract_features**, στο αρχείο feature_extraction.py, και μετατρέπουμε τις τιμές σε decibel με την μέθοδο db_conversion, στο αρχείο audio.py. Τώρα χρησιμοποιούμε το μοντέλο του ταξινομητή για να προβλέψει τις τιμές ανά παράθυρο. Η διαδικασία αυτή ονομάζεται **Classification per feature extraction** και τώρα μπορούμε να ακολουθεί το επόμενο βήμα που είναι η διόρθωση μερικών λανθασμένων αποφάσεων από τον ταξινομητή. Η διαδικασία αυτή ονομάζεται **post processing** και σε αυτή χρησιμοποιούμε ένα φίλτρο μεσαίας τιμής (median filter) καλώντας την συνάρτηση **apply_median_filter**, η οποία βρίσκεται στο αρχείο post_processing.py και χρησιμοποιώντας την μέθοδο signal.medfilt της βιβλιοθήκης scipy. Η τιμή του kernel_size έχει οριστεί ίση με 5, αυτό σημαίνει ότι μπορούμε να διορθώσουμε έως και 2 ετικέτες ανά ομάδα. Τώρα έχοντας την τελική ακολουθία μπορούμε να την διατρέξουμε και να βρούμε τα όρια των λέξεων που εντόπισε ο ταξινομητής. Αυτό υλοποιείται με την μέθοδο **find_word_boundaries**, όπου βρίσκει τις λέξεις και ύστερα τις εμφανίζουμε στο χρήστη. Ύστερα, με την χρήση της **accuracy_score** από την βιβλιοθήκη sklearn.metrics συγκρίνουμε την ακολουθία που προέβλεψε ο ταξινομητής με την πραγματική ακολουθία (ground truth). Τέλος, με την μέθοδο **play_audio_boundaries** η οποία βρίσκεται στο αρχείο audio_player.py αναπαράγουμε τις λέξεις που βρήκε ο ταξινομητής από το αρχείο ήχου που εισήγαγε ο χρήστης, με την βοήθεια της pydub βιβλιοθήκης και του προγράμματος ffmpeg που αναλαμβάνει την αναπαραγωγή του ηχητικού σήματος.

Ταξινομητής Least Squares

Επιλέγοντας ο χρήστης τον ταξινομητή Least Square, αρχικά φορτώνουμε με την μέθοδο load_theta_least_squares_classifier – που βρίσκεται στο αρχείο classifiers.py - το μοντέλο του ταξινομητή που είναι αποθηκευμένο στο μονοπάτι (path) auxiliary2024/classifiers/theta_least_squares_classifier.joblib. Ανακτώντας λοιπόν τις τιμές theta μπορούμε πλέον να το εκτελέσουμε ώστε να προβλέψει τις τιμές για το αρχείο ανά παράθυρο, για αυτό χρησιμοποιείται η μέθοδος **predict_audio_labels**. Αρχικά βρίσκουμε τα χαρακτηριστικά (features) του αρχείου με την **extract_features**, στο αρχείο feature_extraction.py, και μετατρέπουμε τις τιμές σε decibel με την μέθοδο db_conversion, στο αρχείο audio.py. Στο πίνακα με τα χαρακτηριστικά μετατοπίζουμε το εσωτερικό του προσθέτοντας μια στήλη από 1, δηλαδή το απαιτούμενο **bias** και τον **πολλαπλασιάζουμε με τον πίνακα** που περιέχει τις τιμές theta του ταξινομητή Least Squares. Στη συνέχεια μετατρέπουμε τις τιμές αυτές σε **δυαδικές** με την χρήση της μεθόδου binarize_predictions όπου βρίσκεται στο αρχείο labeling.py χρησιμοποιώντας ένα threshold ίσο με 0,5. Η διαδικασία αυτή ονομάζεται **Classification per feature extraction** και τώρα μπορούμε να ακολουθεί το επόμενο βήμα που είναι η διόρθωση μερικών λανθασμένων αποφάσεων από τον ταξινομητή. Η διαδικασία αυτή ονομάζεται **post processing** και σε αυτή αρχικά θα μετατρέψουμε τον πίνακα σε μία λίστα και ύστερα χρησιμοποιούμε ένα φίλτρο μεσαίας τιμής (median filter) καλώντας την συνάρτηση **apply_median_filter**, η οποία βρίσκεται στο αρχείο post_processing.py και χρησιμοποιώντας την μέθοδο signal.medfilt της βιβλιοθήκης scipy. Η τιμή του kernel_size έχει οριστεί ίση με 5, αυτό σημαίνει ότι μπορούμε να διορθώσουμε έως και 2 ετικέτες ανά ομάδα. Τώρα έχοντας την τελική ακολουθία μπορούμε να την διατρέξουμε και να βρούμε τα όρια των λέξεων που εντόπισε ο ταξινομητής. Αυτό υλοποιείται με την μέθοδο **find_word_boundaries**, όπου βρίσκει τις λέξεις και ύστερα τις εμφανίζουμε στο χρήστη. Ύστερα, με την χρήση της **accuracy_score** από την βιβλιοθήκη sklearn.metrics συγκρίνουμε την ακολουθία που προέβλεψε ο ταξινομητής με την πραγματική ακολουθία (ground truth). Τέλος, με την μέθοδο **play_audio_boundaries** η οποία βρίσκεται στο αρχείο audio_player.py αναπαράγουμε τις λέξεις που βρήκε ο ταξινομητής από το αρχείο ήχου που εισήγαγε ο χρήστης, με την βοήθεια της pydub βιβλιοθήκης και του προγράμματος ffmpeg που αναλαμβάνει την αναπαραγωγή του ηχητικού σήματος.

Ταξινομητής RNN

Επιλέγοντας ο χρήστης τον ταξινομητή RNN, αρχικά φορτώνουμε με την μέθοδο load_rnn_model – που βρίσκεται στο αρχείο classifiers.py - το μοντέλο του ταξινομητή που είναι αποθηκευμένο στο μονοπάτι (path) auxiliary2024/classifiers/rnn_classifier.keras. Ανακτώντας λοιπόν το μοντέλο μπορούμε πλέον να το εκτελέσουμε ώστε να προβλέψει τις τιμές για το αρχείο ανά παράθυρο, για αυτό χρησιμοποιείται η μέθοδος **predict_audio_labels**. Αρχικά βρίσκουμε τα χαρακτηριστικά (features) του αρχείου με την **extract_features**, στο αρχείο feature_extraction.py, και μετατρέπουμε τις τιμές σε decibel με την μέθοδο db_conversion, στο αρχείο audio.py. Στο πίνακα με τα χαρακτηριστικά εφαρμόζουμε έναν μετασχηματισμό ώστε να προσθέσουμε την διάσταση με τα mel φίλτρα – 96 φίλτρα – και τώρα χρησιμοποιούμε το μοντέλο του ταξινομητή για να προβλέψει τις τιμές ανά παράθυρο. Στη συνέχεια μετατρέπουμε τις τιμές αυτές σε **δυαδικές** με την χρήση της μεθόδου binarize_predictions όπου βρίσκεται στο αρχείο labeling.py χρησιμοποιώντας ένα threshold ίσο με 0,5. Η διαδικασία αυτή ονομάζεται **Classification per feature extraction** και τώρα μπορούμε να ακολουθεί το επόμενο βήμα που είναι η διόρθωση μερικών λανθασμένων αποφάσεων από τον ταξινομητή. Η διαδικασία αυτή ονομάζεται **post processing** και σε αυτή αρχικά θα μετατρέψουμε τον πίνακα σε μία λίστα και ύστερα χρησιμοποιούμε ένα φίλτρο μεσαίας τιμής (median filter) καλώντας την συνάρτηση **apply_median_filter**, η οποία βρίσκεται στο αρχείο post_processing.py και χρησιμοποιώντας την μέθοδο signal.medfilt της βιβλιοθήκης scipy. Η τιμή του kernel_size έχει οριστεί ίση με 5, αυτό σημαίνει ότι μπορούμε να διορθώσουμε έως και 2 ετικέτες ανά ομάδα. Τώρα έχοντας την τελική ακολουθία μπορούμε να την διατρέξουμε και να βρούμε τα όρια των λέξεων που εντόπισε ο ταξινομητής. Αυτό υλοποιείται με την μέθοδο **find_word_boundaries**, όπου βρίσκει τις λέξεις και ύστερα τις εμφανίζουμε στο χρήστη. Ύστερα, με την χρήση της **accuracy_score** από την βιβλιοθήκη sklearn.metrics συγκρίνουμε την ακολουθία που προέβλεψε ο ταξινομητής με την πραγματική ακολουθία (ground truth). Τέλος, με την μέθοδο **play_audio_boundaries** η οποία βρίσκεται στο αρχείο audio_player.py αναπαράγουμε τις λέξεις που βρήκε ο ταξινομητής από το αρχείο ήχου που εισήγαγε ο χρήστης, με την βοήθεια της pydub βιβλιοθήκης και του προγράμματος ffmpeg που αναλαμβάνει την αναπαραγωγή του ηχητικού σήματος.

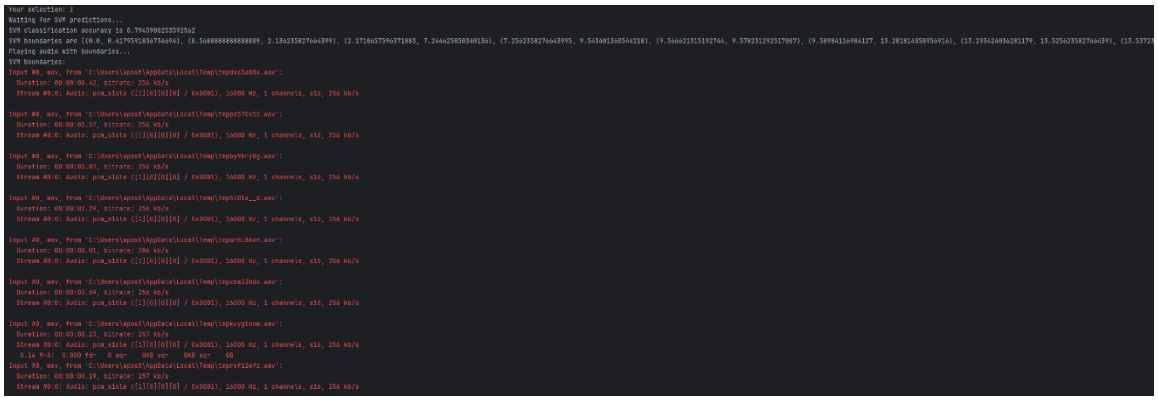
Παράδειγμα Εκτέλεσης word_detector.py

```
Please enter an audio file: 174-84280-0015.flac
```

Ύστερα, εμφανίζοντας ένα κατάλληλο μενού δίνουμε στον χρήστη την επιλογή να διαλέξει τον ταξινομητή οπού θέλει να χρησιμοποιήσει.

```
Please select an option:
1. Predict using SVM
2. Predict using MLP
3. Predict using Least Squares
4. Predict using RNN
5. Exit
Your selection:
```

Επιλέγοντας τον ταξινομητή **SVM**:



Επιλέγοντας τον ταξινομητή **MLP**:

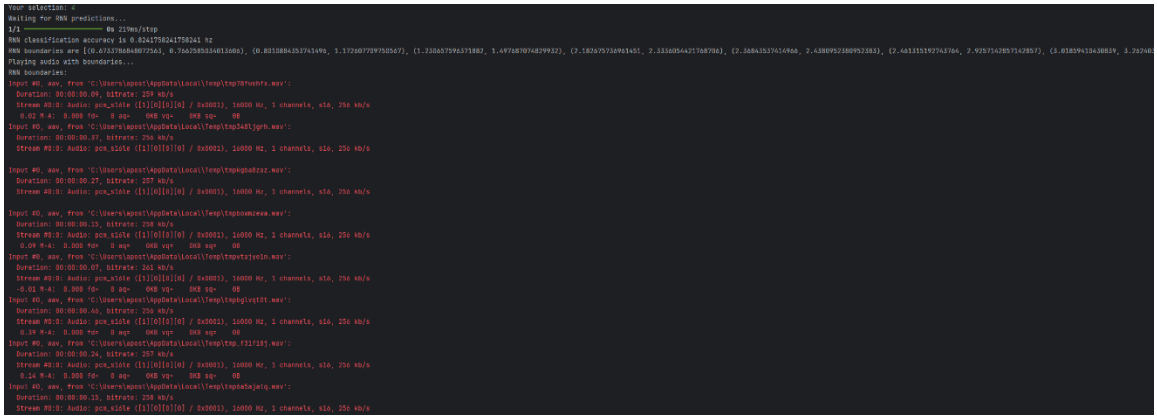
```
Your selection: 2
Waiting for MLP predictions...
MLP classification accuracy is 0.7844463229078613
MLP boundaries are [(0.0, 0.4295691609977324), (0.5688888888888889, 12.945124716553288), (12.979954648526077, 13.722993197278912)]
Playing audio with boundaries...
MLP boundaries:
Input #0, wav, from 'C:\Users\apost\AppData\Local\Temp\tmpe9ywxok5.wav':
  Duration: 00:00:00.43, bitrate: 256 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 16000 Hz, 1 channels, s16, 256 kb/s

Input #0, wav, from 'C:\Users\apost\AppData\Local\Temp\tmpd_bnzrfb.wav':
  Duration: 00:00:12.38, bitrate: 256 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 16000 Hz, 1 channels, s16, 256 kb/s
  12.30 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B
Input #0, wav, from 'C:\Users\apost\AppData\Local\Temp\tmpo0góh162.wav':
  Duration: 00:00:00.74, bitrate: 256 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 16000 Hz, 1 channels, s16, 256 kb/s
```

Επιλέγοντας τον ταξινομητή **Least Squares**:

```
Your selection: 3
Waiting for Least Squares predictions...
2024-06-29 16:20:30.435833: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optim
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate com
Least Squares classification accuracy is 0.7878275570583263
Least Squares boundaries are [(0.0, 0.4295691609977324), (0.5572789115646258, 13.722993197278912)]
Playing audio with boundaries...
Least Squares boundaries:
Input #0, wav, from 'C:\Users\apost\AppData\Local\Temp\tmpd8n1aigo.wav':
  Duration: 00:00:00.43, bitrate: 256 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 16000 Hz, 1 channels, s16, 256 kb/s
   0.34 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B
Input #0, wav, from 'C:\Users\apost\AppData\Local\Temp\tmp3y_95wnx.wav':
  Duration: 00:00:13.17, bitrate: 256 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 16000 Hz, 1 channels, s16, 256 kb/s
  13.05 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B
```

Επιλέγοντας τον ταξινομητή RNN:



Σχεδιαστικές Αποφάσεις

Το μήκους παραθύρου (512) και το μήκος ολίσθησης (256) επιλέχθηκαν ώστε να υπάρχει μία επικάλυψη 50%, καθώς επίσης και το frame rate των ήχων ώστε να μην απαιτούνται να παρθούν πολλά classification decisions.

Στην υλοποίηση του RNN χρησιμοποιήθηκε binary cross-entropy ως loss-function καθώς στην έξοδο υπήρχε ένας νευρώνας. Στην περίπτωση που υπήρχαν δύο νευρώνες με soft-max activation θα χρησιμοποιούταν categorical cross-entropy ως loss-function.
Σε ένα νευρωσικό δίκτυο όπως είναι αυτό του RNN είναι χρήσιμο να έχω ένα μέρος των δεδομένων για validation γιατί κατά την διάρκεια του training μπορούμε να βλέπουμε από πιο σημείο και μετά παύει να συγκλίνει το νευρωνικό. Με άλλα λόγια κοιτώντας το validation error μπορώ να καταλάβω πότε ένα νευρωνικό συγκλίνει.

Γενικές Παρατηρήσεις

Χρησιμοποιήθηκαν δύο datasets, ένα για foreground (LibriSpeech - clean) και ένα για background (ESC-50), όσο περισσότερα αρχεία χρησιμοποιήσουμε από τα datasets τόσο περισσότερο βελτιώνονται τα μοντέλα. Χάριν της εργασίας χρησιμοποιήθηκαν 700 αρχεία ήχου background και 743 αρχεία ήχου για foreground.

Για την φάση των δοκιμών με εξαίρεση αρχεία που βρίσκονται μέσα στο φάκελο auxiliary2024/dataset/testing έχουν δοκιμαστεί αρχεία και από τα dataset Common Voice Delta Segment 18.0 στην Αγγλική αλλά και στην Ελληνική γλώσσα

Από τα αποτελέσματα του παραδείγματος παραπάνω αλλά και γενικότερα, μπορούμε να παρατηρήσουμε πως οι ταξινομητές SVM, MLP και Least Square είναι πιο θορυβώδεις σε σχέση με τον RNN.

Ερώτημα Β

Με την ολοκλήρωσης της εκτέλεσης του αλγορίθμου του ερωτήματος Α, χρησιμοποιώντας τα όρια των λέξεων αναζητούμε την μέση θεμελιώδη συχνότητα του ομιλητή. Έτσι, καλούμε τη μέθοδο mean_fundamental_frequency η οποία βρίσκεται στο αρχείο **fundamental_frequency.py**. Σε αυτή μετατρέπουμε τα όρια των λέξεων που έχει βρει ο ταξινομητής από δευτερόλεπτα σε παράθυρα ώστε να βρούμε τα όρια πάνω στο ίδιο το αρχείο. Με την βοήθεια της μεθόδου αυτοσυσχέτισης **pyin** της βιβλιοθήκης **Librosa** μπορούμε να βρούμε πιθανοτικά με την χρήση του αλγορίθμου yin τις συχνότητες που προκύπτουν από τις λέξεις. Σαν παραμέτρους χρησιμοποιούμε ως y=segment, δηλαδή την λέξη που εντοπίσαμε, fmin=librosa.note_to_hz('C2') που είναι η προτεινόμενη ελάχιστη συχνότητα σε hertz – περίπου 65Hz -, fmax=librosa.note_to_hz('C7') που είναι η προτεινόμενη μέγιστη συχνότητα σε hertz – περίπου 2093Hz – και sr=sample_rate ώστε να έχουμε την δειγματοληψία του αρχείου.

Παράδειγμα εκτέλεσης Mean Fundamental Frequency

```
Mean fundamental frequency of the audio within the boundaries is 164.8 hz
```

Το παραπάνω εμφανίζεται μετά την εκτέλεση του audio_player.py για την προσφώνηση των λέξεων που βρήκε ο ταξινομητής που επιλέχθηκε.
Στο παράδειγμα αυτό παρατηρούμε πως ο ομιλητής είναι άντρας καθώς όπως βλέπουμε η μέση θεμελιώδη συχνότητα του ομιλητή είναι περίπου 165hz.

Γενικές Παρατηρήσεις

Τα αποτελέσματα που η μέση θεμελιώδη συχνότητα του ομιλητή θα είναι περίπου στα 135Hz σημαίνει πως ο ομιλητής αυτός είναι άντρας, ενώ στην περίπτωση που είναι περίπου στα 270Hz σημαίνει πως ο ομιλητής είναι γυναίκα.

Βιβλιογραφία – Δικτυογραφία

1) Rabiner L. (2011), Ψηφιακή Επεξεργασία Φωνής: Θεωρία και Εφαρμογές, 1η Έκδοση, Broken Hill Publishers LTD
2) LibriSpeech-clean Foreground και Testing Dataset, ανάκτηση από https://www.openslr.org/12 στις 19/06/2024
3) ESC-50 Background Dataset, ανάκτηση από https://github.com/karolpiczak/ESC-50?tab=readme-ov-file στις 19/06/2024
4) Librosa, ανάκτηση από https://librosa.org/doc/latest/index.html στις 20/06/2024
5) Joblib, ανάκτηση από https://joblib.readthedocs.io/en/stable/ στις 22/06/2024
6) Save and Load models, ανάκτηση από https://www.tensorflow.org/tutorials/keras/save_and_load στις 22/06/2024
7) LinearSVC, ανάκτηση από https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html στις 20/06/2024
8) MLPClassifier, ανάκτηση από https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html στις 20/06/2024
9) Least Squares, ανάκτηση από https://www.tensorflow.org/api_docs/python/tf/linalg/lstsq στις 20/06/2024
10) RNN, ανάκτηση από https://www.tensorflow.org/api_docs/python/tf/keras/layers/SimpleRNN στις 20/06/2024
11) RNN, ανάκτηση από https://web.stanford.edu/~jurafsky/slp3/9.pdf στις 20/06/2024
12) Media Filter, ανάκτηση από https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.medfilt.html#scipy.signal.medfilt στις 21/06/2024
13) Pydub, ανάκτηση από https://github.com/jiaaro/pydub στις 26/06/2024
14) Λήψη ffmpeg, ανάκτηση από https://www.ffmpeg.org/download.html στις 26/06/2024
15) Common Voice English και Greek έξτρα Datasets, ανάκτηση από https://commonvoice.mozilla.org/en/datasets στις 27/06/2024
16) Fundamental Frequency, ανάκτηση από https://www.studysmarter.co.uk/explanations/english/phonetics/fundamental-frequency/ στις 27/06/2024