ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

ΟΝΟΜΑΤΕΠΩΝΥΜΟ : ΛΟΪΖΙΔΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΑΜ: Π20007

ΠΕΡΙΕΧΟΜΕΝΑ

Contents

ΤΕΚΜΗΡΙΩΣΗ ΚΩΔΙΚΑ

Για την ολοκλήρωση εργασίας δημιούργησα δεκατέσσερα python αρχεία,τα οποία θα εξηγώ τον σκοπό τους και παράλληλα θα εξηγώ και τον αντίστοιχο κώδικα τους.

load.py

Περιέχει τρείς μεθόθους τις οποίες χρησιμοποιώ σε άλλα python αρχεία.Αυτες οι μεθόδοι είναι οι:

load_audio_files, **extract_mel_spectrogram** και η **extract_single_mel_spectrogram**.

Η **load_audio_files** δέχεται ως είσοδο έναν κατάλογο (directory) και επιστρέφει μια λίστα από αρχεία ήχου.Μετά αρχικοποιεί μια κενή λίστα audio_files για να αποθηκεύση τα αρχεία ήχου που θα φορτωθούν.Χρησιμοποιά τη συνάρτηση os.walk για να περιηγηθεί αναδρομικά σε όλους τους υποκαταλόγους και τα αρχεία του συγκεκριμένου καταλόγου directory.Για κάθε αρχείο που βρίσκεται, συνθέτει την πλήρη διαδρομή του αρχείου χρησιμοποιώντας τη os.path.join.Στη συνέχεια, φορτώνει το αρχείο ήχου χρησιμοποιώντας τη librosa.load, όπου το αρχείο φορτώνεται για διάρκεια 3 δευτερολέπτων. Η μεταβλητή audio περιέχει τα δεδομένα του ήχου, ενώ το _ αγνοεί την τιμή του δείγματος συχνότητας.Προσθέτει στη συνέχεια τα δεδομένα του ήχου στη λίστα audio_files.Τελος, επιστρέφει τη λίστα audio_files που περιέχει όλα τα αρχεία ήχου που φορτώθηκαν από τον κατάλογο.

Η **extract_mel_spectrogram** δέχεται ως είσοδο μια λίστα audio_data που περιέχει αρχεία ήχου, και επιστρέφει ένα πίνακα με τα Mel Spectrograms αυτών των αρχείων. Οι παράμετροι

n_mels, hop_length, και n_fft είναι οι ρυθμίσεις για την εξαγωγή του Mel Spectrogram.Αρχικοποιεί μια κενή λίστα features για να αποθηκεύσει τα Mel Spectrograms που θα εξάγει.

Στη συνέχεια διατρέχει κάθε αρχείο ήχου στη λίστα audio_data.Αν κάποιο αρχείο ήχου είναι κενό (δηλαδή έχει μήκος 0), το παραλείπει.Χρησιμοποιά τη librosa.feature.melspectrogram για να εξάγει το Mel Spectrogram του αρχείου ήχου. Οι παράμετροι n_mels, hop_length, και n_fft καθορίζουν τον αριθμό των Mel συχνοτήτων, το μήκος του hop και το μήκος του FFT αντίστοιχα.Μετατρέπει το Mel Spectrogram σε λογαριθμική κλίμακα χρησιμοποιώντας τη librosa.power_to_db, με αναφορά την μέγιστη τιμή του spectrogram.Μετά προσθέτει το λογαριθμικό Mel Spectrogram στη λίστα features.Τέλος, επιστρέφει τη λίστα features ως έναν πίνακα NumPy.

Η **extract_single_mel_spectrogram** δέχεται μόνο ένα ένα αρχείο ήχου, εξάγει το Mel Spectrogram αυτού του αρχείου, το μετατρέπει σε λογαριθμική κλίμακα, και επιστρέφει το λογαριθμικό Mel Spectrogram

least_squares_training.py

Ουσιαστικά φορτώνει δεδομένα ήχου, εξάγει χαρακτηριστικά, δημιουργεί ετικέτες, αρχικοποιεί και εκπαιδεύει ένα μοντέλο γραμμικής παλινδρόμησης, το οποίο αποθηκεύει σε ένα αρχείο.Αρχικά εισαγάγει τις βιβλιοθήκες ,**LinearRegression** από το **sklearn.linear_model** για την αρχικοποίηση του μοντέλου και χρησιμοποιεί την **fit** για την εκπαίδευση του μοντέλου.Μετα την **numpy** για τη διαχείριση των αριθμητικών δεδομένων.Πιο μετα φορτώνει **load_audio_files** και **extract_mel_spectrogram** από το load για τη φόρτωση και εξαγωγή χαρακτηριστικών από τα αρχεία ήχου.Εισάγει την **joblib** για την αποθήκευση του εκπαιδευμένου μοντέλου σε αρχείο.Φορτώνει τα αρχεία ήχου από τους αντίστοιχους φακέλους για το foreground και το

background χρησιμοποιώντας τη συνάρτηση **load_audio_files**.Εξάγει τα χαρακτηριστικά Mel Spectrogram από τα δεδομένα ήχου χρησιμοποιώντας τη συνάρτηση **extract_mel_spectrogram**.Δημιουργεί ετικέτες για τα δεδομένα ήχου:1 για το foreground και 0 για το background.Συνδυάζει τα χαρακτηριστικά και τις ετικέτες χρησιμοποιώντας **np.concatenate**.Αναδιαμορφώνει τα χαρακτηριστικά για χρήση σε Least Squares, χρησιμοποιώντας **reshape**.Μετά ορίζει το εκπαιδευτικό σύνολο X_train και y_train.Αρχικοποιεί και εκπαιδεύει το μοντέλο γραμμικής παλινδρόμησης χρησιμοποιώντας **LinearRegression** και **fit**.Τέλος,αποθηκεύει το εκπαιδευμένο μοντέλο σε ένα αρχείο χρησιμοποιώντας joblib.dump.

least_squares.py

Φορτώνει τα χαρακτηριστικά και τις ετικέτες του audio για testing, φορτώνει ένα εκπαιδευμένο μοντέλο γραμμικής παλινδρόμησης, κάνει προβλέψεις, εφαρμόζει φίλτρο μέσης τιμής και αξιολογεί την ακρίβεια του μοντέλου.Εισάγουμε την **joblib** για την φόρτωση και αποθήκευση μοντέλων.Την **numpy** για την διαχείριση αριθμητικών δεδομένων,την **medfilt** από **scipy.signal** για την εφαρμογή φίλτρου μέσης τιμής και την **accuracy_score** από **sklearn.metrics** για την αξιολόγηση της ακρίβειας του μοντέλου.Φορτώνει τα χαρακτηριστικά (combined_features.npy) και τις ετικέτες (combined_labels.npy) του audio προς testing .Αναδιαμορφώνει τα χαρακτηριστικά και τις ετικέτες ώστε να είναι κατάλληλα για την χρήση στο μοντέλο γραμμικής παλινδρόμησης.Ορίζει τα χαρακτηριστικά του audio προς testing (X_test) και τις ετικέτες του (y_test).Φορτώνει το αποθηκευμένο μοντέλο γραμμικής παλινδρόμησης από το αρχείο linear_regression_model.joblib.Κάνει προβλέψεις

χρησιμοποιώντας τα χαρακτηριστικά του audio προς testing.Εφαρμόζει κατώφλι (threshold) στις προβλέψεις για να πάρει δυαδικές ετικέτες (0 ή 1).Εφαρμόζει φίλτρο μέσης τιμής (medfilt) στις δυαδικές προβλέψεις για να βελτιώσει την ομαλότητα των προβλέψεων.Μετα αποθηκεύει τις φιλτραρισμένες προβλέψεις σε ένα αρχείο

(y_pred_filtered_least_squares.npy).Τέλος, υπολογίζει και εκτυπώνει την ακρίβεια του μοντέλου χρησιμοποιώντας την **accuracy_score**.

svm_training.py

Κάνει την ίδια δουλειά με το **least_squares_training.py** όμως αντι να χρησιμοποιεί **LinearRegression** ,χρησιμοποιεί **LinearSVC** απο την **sklearn.svm** όπου σαν παραμέτρους της έχω random_state=42 που χρησιμοποιείται για την αναπαραγωγικότητα των αποτελεσμάτων, η max_iter=1000 καθορίζει το μέγιστο αριθμό επαναλήψεων για τον αλγόριθμο εκπαίδευσης και η dual=False ουσιαστικά χρησιμοποιήται διότι ο αριθμός των δειγμάτων (samples) είναι μεγαλύτερος από τον αριθμό των

χαρακτηριστικών (features).

svm.py

Κάνει την ίδια δουλειά με το **least_squares.py** όμως δεν χρεαιάζεται εδώ να χρησιμοποιήσουμε κάποιο κατώφλι διότι παίρνουμε αμέσως απο την **predict** μόνο 0 και 1.Επίσης τα predictions τα κάνει save με διαφορετικό όνομα (y_pred_filtered_svm.npy).

mlp_training.py

Κάνει την ίδια δουλειά με το **least_squares_training.py** όμως αντι να χρησιμοποιεί **LinearRegression** ,χρησιμοποιεί **MLPClassifier** απο την **sklearn.neural_network** όπου σαν παραμέτρους της έχω random_state=42 που χρησιμοποιείται για την αναπαραγωγικότητα των αποτελεσμάτων, η max_iter=100 καθορίζει το μέγιστο αριθμό επαναλήψεων για τον αλγόριθμο εκπαίδευσης και η hidden_layer_sizes=(128, 64, 32) που ορίζει τρία layers απο νευρώνες που κάθε αριθμός συμβολίζει το πλήθος των νευρόνων που έχει κάθε layer αντίστοιχα.

mlp.py

Κάνει την ίδια δουλειά με το **svm.py** αλλά χρησιμοποιεί window_length = 7 για την **medfilt** για καλύτερα αποτελέσματα επίσης τα predictions τα κάνει save με διαφορετικό όνομα (y_pred_filtered_mlp.npy).

rnn_training.py

Οι ουσιαστικές διαφορές με το **mlp_training.py** είναι πως χρησιμοποιήται η tensorflow.keras για τη δημιουργία και εκπαίδευση νευρωνικών δικτύων.Επίσης τα χαρακτηριστικά μετατρέπονται στη σωστή διάταξη για χρήση σε RNN (από (n_samples, n_features, n_frames) σε (n_samples, n_frames, n_features)).Μετά μετατρέπονται και οι ετικέτες στη σωστή διάταξη (features.shape[0], features.shape[1]) όπου features.shape[0], είναι ο αριθμός των δειγμάτων (samples) στα δεδομένα εισόδου και το features.shape[1] είναι ο αριθμός των χρονικών βημάτων (time steps) για κάθε δείγμα.Για τη δημιουργία του μοντέλου χρησιμοποιεί το **Sequential()** . Μετά προσθέτει το στρώμα εισόδου στο μοντέλο. Το Input στρώμα καθορίζει το

σχήμα των δεδομένων εισόδου,shape=(X_train.shape[1], X_train.shape[2]) καθορίζει ότι η είσοδος θα έχει σχήμα (αριθμός χρονικών βημάτων, αριθμός χαρακτηριστικών).Στη συνέχεια προσθέτει δύο στρώματα RNN στο μοντέλο ,το **SimpleRNN**(50) καθορίζει ότι το RNN στρώμα θα έχει 50 νευρώνες και το return_sequences=True καθορίζει ότι το RNN στρώμα θα επιστρέφει την πλήρη ακολουθία εξόδου για κάθε χρονικό βήμα. Περαιτέρω προσθέτει ένα **TimeDistributed** στρώμα **Dense** στο μοντέλο,το **TimeDistributed**(**Dense**(1)) καθορίζει ότι κάθε χρονικό βήμα της εισόδου θα περάσει από ένα πυκνό στρώμα με 1 νευρώνα.Το activation='sigmoid' χρησιμοποιεί τη συνάρτηση ενεργοποίησης sigmoid, η οποία είναι κατάλληλη για δυαδική ταξινόμηση (εξόδου 0 ή 1).Τέλος, καθορίζουμε τη συνάρτηση κόστους binary_crossentropy, η οποία είναι κατάλληλη για δυαδική ταξινόμηση και συνεχίζουμε με την **fit** που χρησιμοποιεί το 20% των δεδομένων εκπαίδευσης ως σύνολο επικύρωσης.Αυτες ήταν οι βασικές διαφορές τα υπόλοιπα είναι τα ίδια.

rnn.py

Φορτώνει τα χαρακτηριστικά του τεστ από ένα αρχείο .npy.Μεταθέτει τα χαρακτηριστικά για να έχουν το σωστό σχήμα για το RNN. Αντί να έχουν σχήμα (samples, features, time_steps), τώρα έχουν σχήμα (samples, time_steps, features).Φορτώνει τις ετικέτες από ένα αρχείο .npy.Αναδιαμορφώνει τις ετικέτες ώστε να ταιριάζουν με τα δεδομένα εισόδου. Οι ετικέτες θα έχουν τώρα το ίδιο σχήμα με τα χρονικά βήματα και τα δείγματα.Αναθέτει τα δεδομένα εισόδου και τις ετικέτες του τεστ.Φορτώνει το εκπαιδευμένο μοντέλο RNN από το αρχείο.Μετα κάνει προβλέψεις με το μοντέλο. Οι προβλέψεις είναι συνεχείς τιμές, οπότε εφαρμόζει ένα κατώφλι (threshold) 0.5 για να τις μετατρέψει σε δυαδικές τιμές.Αναδιαμορφώνει τις προβλέψεις

και τις ετικέτες σε μονοδιάστατους πίνακες για να μπορέσει να τις συγκρίνει.Εφαρμόζει ένα φίλτρο μέσης τιμής με παράθυρο μήκους 5 για να εξομαλύνει τις προβλέψεις.Στη συνέχεια αποθηκεύει τις προβλέψεις σε ένα αρχείο .npy.Τέλος, υπολογίζει την ακρίβεια των προβλέψεων συγκρίνοντας τις με τις πραγματικές ετικέτες και εκτυπώνει το αποτέλεσμα.

create_audio.py

Αποθηκεύει τον συνδυασμένο ήχο σε ένα αρχείο FLAC. Χρησιμοποιεί τη βιβλιοθήκη soundfile για τη δημιουργία του αρχείου ήχου.Φορτώνει αρχεία ήχου από δύο καταλόγους έναν για τον ήχο του προσκηνίου και έναν για τον ήχο του υποβάθρου.Εξάγει τα χαρακτηριστικά melspectrogram από τα αρχεία ήχου του προσκηνίου και του υποβάθρου.

Δημιουργεί ετικέτες για τα χαρακτηριστικά του προσκηνίου (1) και του υποβάθρου (0).Συνδυάζει τα χαρακτηριστικά και τις ετικέτες σε έναν πίνακα.Επιλέγει ένα τυχαίο υποσύνολο από 5 αρχεία ήχου του προσκηνίου και του υποβάθρου.Συνδυάζει τους επιλεγμένους ήχους και τις ετικέτες τους σε έναν νέο πίνακα.Αποθηκεύει τον συνδυασμένο ήχο σε ένα αρχείο FLAC.Έπειτα εξάγει τα χαρακτηριστικά melspectrogram από τον συνδυασμένο ήχο.Τέλος, αποθηκεύει τα εξαγόμενα χαρακτηριστικά και τις ετικέτες σε αρχεία .npy για μελλοντική χρήση.

scanner.py

Χρησιμοποιεί την συνάρτηση **scan_words** ,η συνάρτηση αυτή σαρώνει τις φιλτραρισμένες προβλέψεις (1 για λέξη, 0 για θόρυβο) και προσδιορίζει τα όρια των λέξεων.Οι παράμετροι τις είναι οι y_pred_filtered (np.ndarray),πίνακας φιλτραρισμένων προβλέψεων,hop_length (int) αριθμός δειγμάτων μεταξύ

διαδοχικών frame (default: 512) και sr (int),συχνότητα δειγματοληψίας του ήχου (default: 22050).Σαρώνει τις φιλτραρισμένες προβλέψεις για ανίχνευση ορίων λέξεων.Αρχικοποιεί μια λίστα words για αποθήκευση των ορίων των λέξεων.Χρησιμοποιεί δύο while loops για να παρακάμψει τα κομμάτια θορύβου και να ανιχνεύσει τις λέξεις.Όταν βρει την αρχή μιας λέξης, ορίζει το start και συνεχίζει να ανιχνεύει μέχρι να βρει το τέλος της λέξης (end).Εάν η αρχή και το τέλος της λέξης δεν είναι τα ίδια, αποθηκεύει το ζεύγος (start, end) στη λίστα words.Έπειτα μετατρέπη τα όρια των frame σε δευτερόλεπτα.Χρησιμοποιεί λίστα καταγραφής για να μετατρέψει τα όρια frame σε δευτερόλεπτα και αποθηκεύει τα αποτελέσματα στη μεταβλητή word_times.Φορτώνει τις φιλτραρισμένες προβλέψεις από ένα αρχείο .npy με τη χρήση της συνάρτησης np.load.Καλεί τη συνάρτηση scan_words για να σκανάρει τις λέξεις και να προσδιορίσει τα χρονικά όρια των λέξεων.Μετά εκτυπώνει τα χρονικά όρια των λέξεων σε δευτερόλεπτα.Τέλος, αποθηκεύει τα χρονικά όρια των λέξεων σε ένα αρχείο .npy με τη χρήση της συνάρτησης np.save.

player.py

Περιέχει τη συνάρτηση **play_audio_segment**,παίζει απλά ένα δεδομένο τμήμα ήχου.Οι παράμετροι της είναι οι,audio_segment (np.ndarray) δεδομένα ήχου που θα αναπαραχθούν και sample_rate (int) συχνότητα δειγματοληψίας των δεδομένων ήχου.Χρησιμοποιεί τη βιβλιοθήκη **sounddevice** για να παίξει τον ήχο και να περιμένει να τελειώσει.Επίσης περιέχει τη συνάρτηση **play_words**,παίζει τα τμήματα ήχου που αντιστοιχούν στις ανιχνευμένες λέξεις.Οι παράμετροι της είναι,audio_file_path (str) διαδρομή προς το αρχείο ήχου και word_times (list) λίστα από ζεύγη (start, end) με τα χρονικά όρια (σε δευτερόλεπτα) των ανιχνευμένων λέξεων.Φορτώνει το αρχείο ήχου χρησιμοποιώντας

τη βιβλιοθήκη soundfile.Για κάθε ζεύγος (start, end) στη λίστα word_times,μετατρέπει τα χρονικά όρια σε δείκτες δειγμάτων.

Εξάγει το τμήμα ήχου για τη λέξη.Παίζει το τμήμα ήχου χρησιμοποιώντας τη συνάρτηση **play_audio_segment**.

Ορίζει τη διαδρομή προς το αρχείο ήχου.Φορτώνει τα χρονικά όρια των λέξεων από ένα αρχείο .npy με τη χρήση της συνάρτησης np.load.Τέλος, καλεί τη συνάρτηση **play_words** για να παίξει τα ανιχνευμένα τμήματα ήχου.

split_audio.py

Περιέχει τη συνάρτηση **split_audio_clip**,διαιρεί ένα αρχείο ήχου σε κλιπ των 4 δευτερολέπτων και τα αποθηκεύει ως αρχεία FLAC 24-bit.Οι παράμετροι είναι,path (str) διαδρομή προς το αρχείο ήχου εισόδου και output_dir (str) κατάλογος όπου θα αποθηκευτούν τα κλιπ εξόδου.Η συνάρτηση χρησιμοποιεί τις βιβλιοθήκες **librosa** και **soundfile** για την επεξεργασία του ήχου.Χρησιμοποιεί τη συνάρτηση **librosa.get_samplerate** για να λάβει τον ρυθμό δειγματοληψίας του αρχείου ήχου.Έπειτα χρησιμοποιεί τη συνάρτηση **librosa.get_duration** για να λάβει τη διάρκεια του αρχείου ήχου σε δευτερόλεπτα.Μετά χρησιμοποιεί ένα βρόχο for για να διατρέξει το αρχείο ήχου σε διαστήματα των 4 δευτερολέπτων.Φορτώνει ένα τμήμα των 4 δευτερολέπτων του αρχείου ήχου ξεκινώντας από το 'i' δευτερόλεπτο χρησιμοποιώντας τη συνάρτηση **librosa.load**.Αποθηκεύει το τμήμα ήχου σε ένα αρχείο FLAC 24-bit χρησιμοποιώντας τη συνάρτηση **sf.write** από τη βιβλιοθήκη **soundfile**.Στη συνέχεια ορίζει τη διαδρομή προς το αρχείο ήχου και τον κατάλογο εξόδου.Τέλος, καλεί τη συνάρτηση **split_audio_clip** για να διαιρέσει το αρχείο ήχου σε κλιπ και να τα αποθηκεύσει.Το έκανα αυτό το αρχείο python όυτως ώστε, να διαιρέσω ένα μεγάλο audio που έιχα κάνει record για καλύτερη διαχείριση του audio.

compute_frequency.py

Η συνάρτηση **compute_fundamental_frequency** υπολογίζει τη θεμελιώδη συχνότητα ενός δοθέντος ηχητικού τμήματος χρησιμοποιώντας αυτοσυσχέτιση. Ορίζει το εύρος για έγκυρη ανίχνευση συχνοτήτων (50 Hz έως 500 Hz).Υπολογίζει την αυτοσυσχέτιση του ηχητικού σήματος.Κανονικοποιεί την αυτοσυσχέτιση.Βρίσκει τη μέγιστη αυτοσυσχέτιση εντός του καθορισμένου εύρους lag.Αν η μέγιστη αυτοσυσχέτιση είναι κάτω από το κατώφλι, επιστρέφει None.Προσδιορίζει το lag που αντιστοιχεί στη μέγιστη αυτοσυσχέτιση.Υπολογίζει τη θεμελιώδη συχνότητα διαιρώντας το ρυθμό δειγματοληψίας με το lag.Η **average_fundamental_frequency** συνάρτηση υπολογίζει τη μέση θεμελιώδη συχνότητα των λέξεων σε ένα ηχητικό αρχείο. Πιο συγκεκριμένα,φορτώνει το ηχητικό αρχείο.Μετά για κάθε χρονικό διάστημα που περιέχει μια λέξη μετατρέπει τα χρονικά όρια σε δείγματα.Υπολογίζει τη θεμελιώδη συχνότητα για κάθε λέξη χρησιμοποιώντας τη συνάρτηση **compute_fundamental_frequency**.Αν η θεμελιώδης συχνότητα είναι έγκυρη, την προσθέτει σε μια λίστα.Υπολογίζει και επιστρέφει τη μέση θεμελιώδη συχνότητα, αν υπάρχουν έγκυρες συχνότητες.Τελος, ορίζει τη διαδρομή προς το ηχητικό αρχείο, φορτώνει τα χρονικά όρια των λέξεων και υπολογίζει τη μέση θεμελιώδη συχνότητα.Ουσιαστικά θέλαμε απο τις λέξεις που προκύπτουν, να υπολογίσουμε τη μέση θεμελιώδη συχνότητα του ομιλητή με κάποια μέθοδο αυτοσυσχέτησης (sort term auto correlation function) και να παίρναμε το average της τιμης αυτης, δηλαδή θα έτρεχαι αυτο το function για κάθε λέξη που βρήκαμε και στο τελος να πέρναμε το μέσο.Όμως είναι άξιο να σημειωθεί πως δεν μπορούμε να εμπιστευθούμε την απάντηση που θα πάρουμε απο αυτη τη συναρτηση αυτοσισχετισης, οπότε πρέπει να κάνουμε scale.Κάνωντας αυτην την πράξη θα λάβουμε μια

κανονικοποιημενη αυτοσισχετηση στην οποια θα βαλουμε ενα threshold = 0,7 οπου έτσι θα κρατάμε τις εκτιμησεις τις θεμελιωθους συχνωτητας μονον οταν η τιμη αυτοσισχετησης η μεγιστη ειναι μεγαλυτερη απο 0,7 .Αυτο ειναι ενας τροπος να προσθεσουμε εναν διακοπτη του οποιου η δουλεια ειναι να εμπιστευομαστε μονο τα frames στα οποια η μεγιστη αυτοσισχετηση ειναι μεγαλυτερη απο 0,7.

ΔΙΑΧΕΙΡΙΣΗ ΚΑΙ ΧΡΗΣΗ ΔΕΔΟΜΕΝΩΝ

Για τα δεδομένα foreground που χρησιμοποιώ τα πήρα από το https://www.openslr.org/12/, τα οποία ανήκουν στο σύνολο δεδομένων LibriSpeech,όμως σχετικά πήρα εναν αρκετά μικρό αριθμό audio σε σύγκριση με το τι προσέφερε. Αξιοποιούνται στην εργασία για να εκπαιδεύσω και να αξιολογήσω μοντέλα μηχανικής μάθησης στον εντοπισμό ομιλίας και διαχωρισμό ομιλίας από τον θόρυβο. Πιο συγκεκριμένα, χρησιμοποιούνται για να δημιουργήσω τα θετικά παραδείγματα ομιλίας που χρειάζονται για την εκπαίδευση και δοκιμή των μοντέλων μου.Τα δεδομένα background είναι απολύτως δικά μου και τα δημιούργησα μέσω ενός app στο κινητό που με αφήνει να κάνω record τον χώρο γύρο μου.Συγκεκριμένα έκανα ,record σε ένα café που είχε αρκετό θόρυβο γύρο γύρο και έκανα και record στο σπίτι μου που είχε μερικό θόρυβο. Μέσα στον φάκελο LibriSpeech μπορείτε να βρείτε τα LICENCE με README των δεδομένων foreground και τα ίδια τα δεδομένα foreground στον φάκελο def-forground.Επίσης μπορείτε αν θέλετε να δείτε τα background δεδομένα στον φάκελο dev-background και το αρχείο για testing στον φάκελο test.Επίσης στο auxilary2023 εχω ακομα τρείς φακέλους , models που είναι φιλαγμένα μέσα τα μοντέλα μου.Ο Time_limits_of_words που περιέχει τα χρονικά όρια των λέξεων που έχουν ανιχνευτεί και ο Predictions που περιέχει τα φιλτραρισμένα predictions του κάθε ταξινομητή.

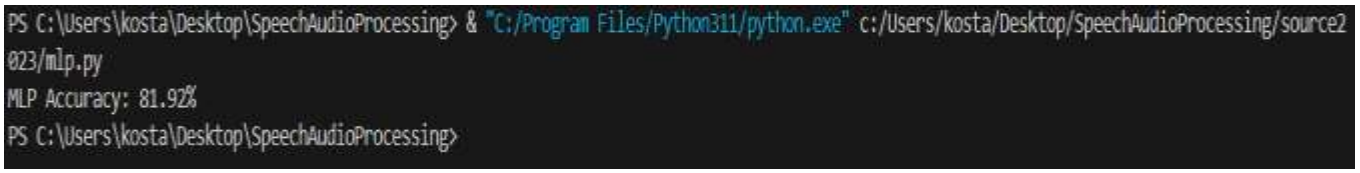
ΠΑΡΑΔΕΙΓΜΑΤΑ ΕΚΤΕΛΕΣΗΣ ΚΩΔΙΚΑ ΓΙΑ ACCURACY

Αυτα είναι τα παραδείγματα εκτέλεσης των αρχείων least_squares.py, svm.py, mlp.py και rnn.py αντίστοιχα.Βλέπουμε πως τα νευρωνικά δίκτυα έχουν μεγαλύτερη επιτυχία στα δεδομένα αυτά ,όμως κάτι παράξενο που παρατήρησα είναι πως ο mlp έχει αρκετά καλύτερο accuracy απο τον rnn.

Least Squares:

SVM:

MLP:



RNN:

ΠΑΡΑΔΕΙΓΜΑΤΑ ΕΚΤΕΛΕΣΗΣ ΚΩΔΙΚΑ SCANNER

Αυτα είναι τα παραδείγματα εκτέλεσης του αρχείου scanner.py όπου την πρώτη φορά τρέχει με τα predictions του least_squares μετά με του svm μετα με mlp και τέλος με του rnn .

Lest Squares:

```
PS C:\Users\kosta\Desktop\SpeechAudioProcessing> & "C:/Program Files/Python311/python.exe" c:/Users/kosta/Desktop/SpeechAudioProcessing/source2
023/scanner.py
Word times (in seconds): [(0.0, 0.9752380952380952), (1.1145578231292517, 2.995374149659864), (4.504671201814059, 5.456689342403628), (5.549569
1609977325, 9.032562358276644), (9.125442176870749, 12.07437641723356), (12.770975056689343, 13.374693877551021), (13.444353741496599, 15.06975
0566893424), (16.46294784580499, 17.507845804988662), (17.647165532879818, 18.11156462585034), (19.17968253968254, 19.249342403628116), (19.365
442176870747, 24.125532879818593), (24.706031746031748, 24.91501133786848), (25.007891156462584, 25.123990929705215), (25.170430839002268, 30.1
62721088435376)]
PS C:\Users\kosta\Desktop\SpeechAudioProcessing>
```

SVM:

```
PS C:\Users\kosta\Desktop\SpeechAudioProcessing> & "C:/Program Files/Python311/python.exe" c:/Users/kosta/Desktop/SpeechAudioProcessing/source2
023/scanner.py
Word times (in seconds): [(0.0, 0.9752380952380952), (1.1145578231292517, 2.995374149659864), (4.504671201814059, 5.456689342403628), (5.549569
1609977325, 9.032562358276644), (9.125442176870749, 12.07437641723356), (12.770975056689343, 13.374693877551021), (13.444353741496599, 15.06975
0566893424), (16.46294784580499, 17.507845804988662), (17.647165532879818, 18.11156462585034), (19.17968253968254, 19.249342403628116), (19.365
442176870747, 24.125532879818593), (24.706031746031748, 24.91501133786848), (25.007891156462584, 25.123990929705215), (25.170430839002268, 30.1
62721088435376)]
PS C:\Users\kosta\Desktop\SpeechAudioProcessing>
```

MLP:

```
PS C:\Users\kosta\Desktop\SpeechAudioProcessing> & "C:/Program Files/Python311/python.exe" c:/Users/kosta/Desktop/SpeechAudioProcessing/source2
023/scanner.py
Word times (in seconds): [(0.0, 2.995374149659864), (6.0836281179138325, 9.05578231292517), (10.356099773242631, 10.425759637188209), (10.47219
954648526, 12.051156462585034), (12.167256235827665, 15.069750566893424), (17.809705215419502, 18.134784580498867), (18.22766439909297, 21.1069
38775510205), (22.407256235827663, 22.453696145124717), (22.50013605442177, 22.523356009070294), (22.6162358276644, 23.59147392290249 5), (23.70
7573696145126, 27.190566893424037), (28.421224489795918, 29.60544217687075), (29.76798185941043, 30.162721088435376)]
PS C:\Users\kosta\Desktop\SpeechAudioProcessing>
```

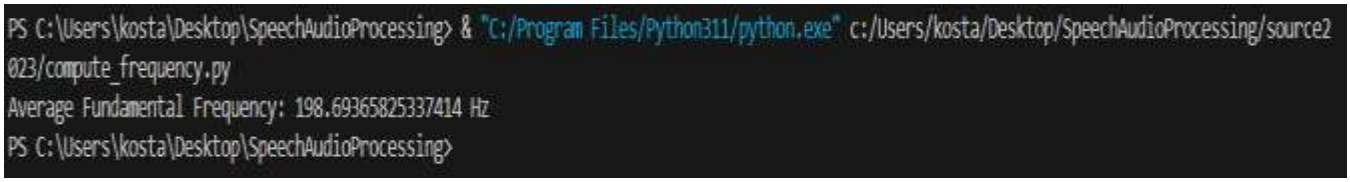
RNN:

```
PS C:\Users\kosta\Desktop\SpeechAudioProcessing> & "C:/Program Files/Python311/python.exe" c:/Users/kosta/Desktop/SpeechAudioProcessing/source2
023/scanner.py
Word times (in seconds): [(0.0, 2.995374149659864), (6.037188208616780, 15.139410430839002), (17.879365079365008, 30.162721088435376)]
PS C:\Users\kosta\Desktop\SpeechAudioProcessing>
```

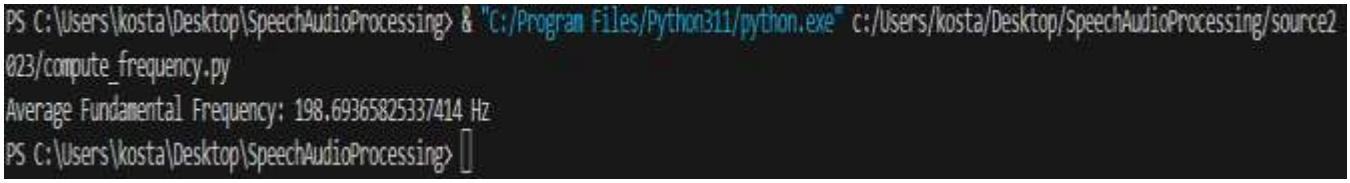
ΠΑΡΑΔΕΙΓΜΑΤΑ ΕΚΤΕΛΕΣΗΣ ΚΩΔΙΚΑ ΓΙΑ ΤΟΝ ΥΠΟΛΟΓΙΣΜΟ ΜΕΣΗΣ ΘΕΜΕΛΙΩΔΗΣ ΣΥΧΝΟΤΗΤΑΣ

Αυτα είναι τα παραδείγματα εκτέλεσης του αρχείου compute_frequency.py όπου την πρώτη φορά τρέχει με τα woed_times του least_squares μετά με του svm μετα με mlp και τέλος με του rnn .

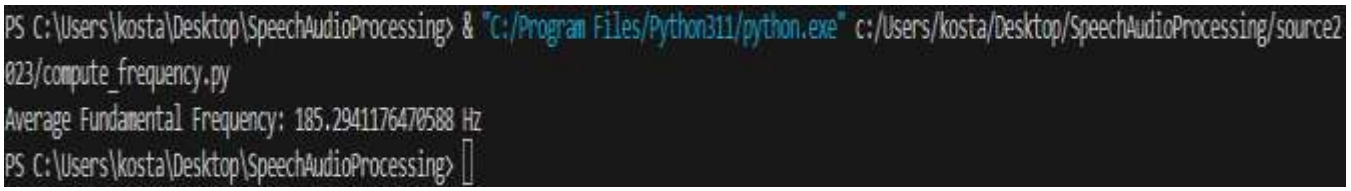
Least Squares:



SVM:



MLP:



RNN: