Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

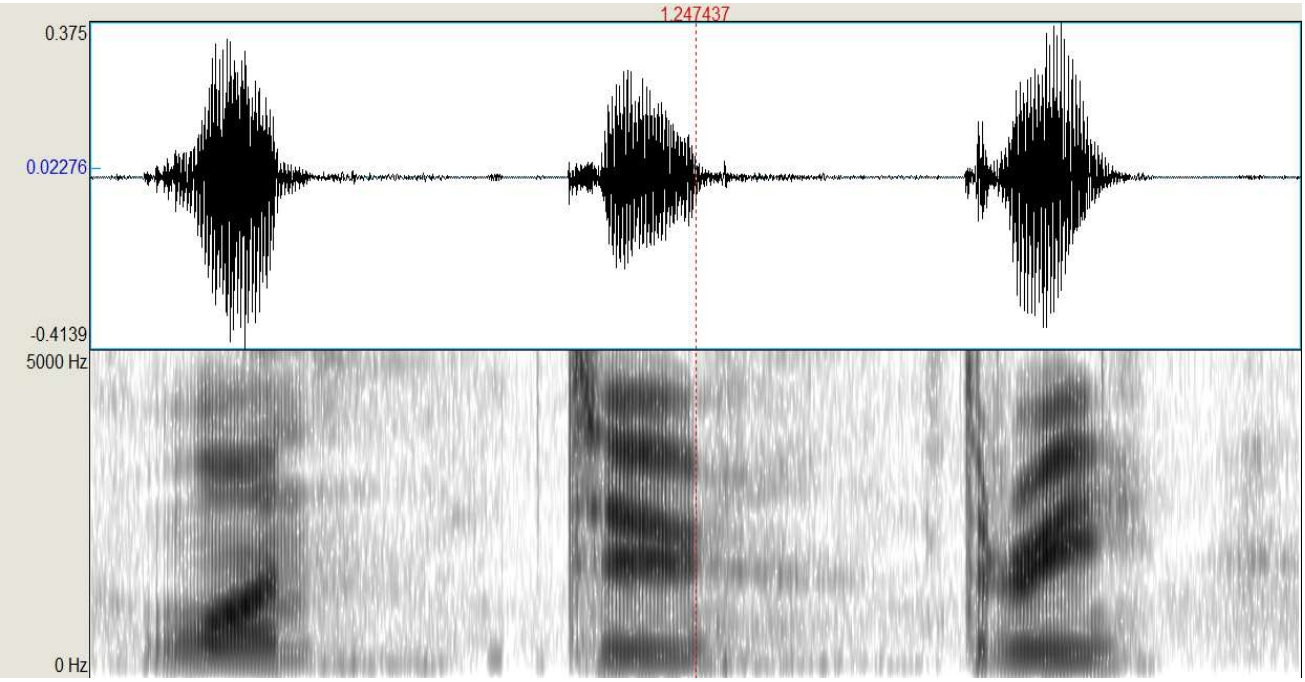
Ροή Σ: Αναγνώριση Προτύπων (7ο Εξάμηνο)
Εργαστηριακή Άσκηση 2

Αλέξης Μάρας 03118074
Δημήτρης Μπακάλης 03118163

Βήμα 1

Άνδρας – Ομιλητής 1

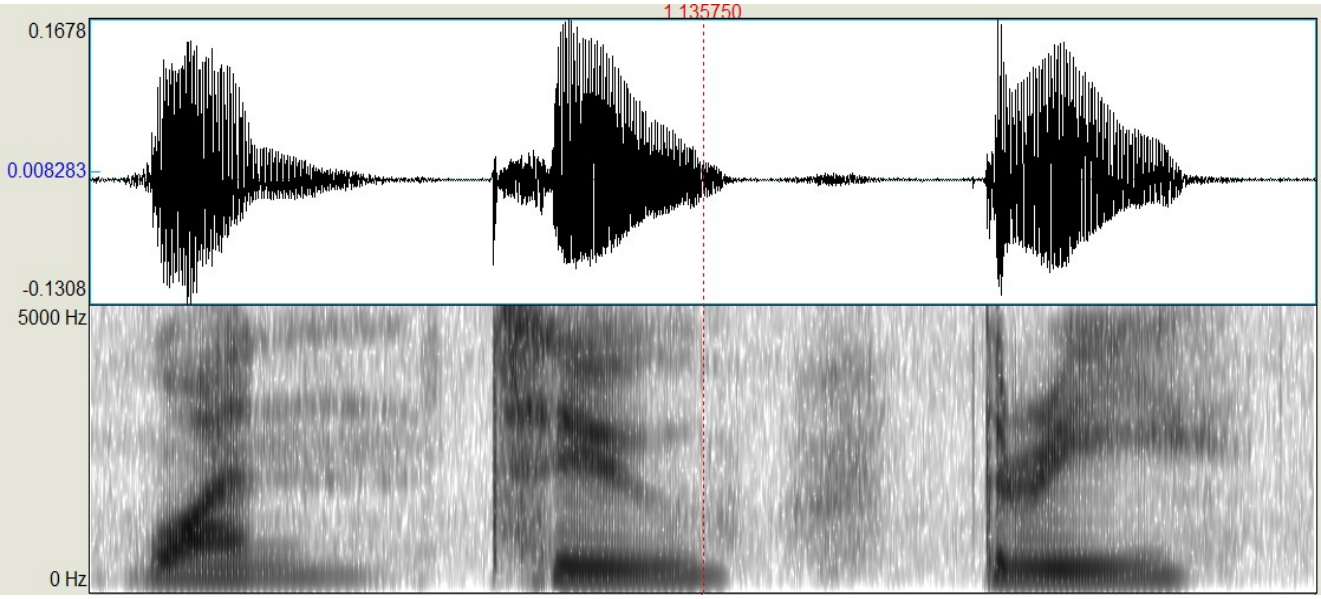
Η κυματομορφή και το σπεκτογράφημα του ηχητικού αρχείου του άντρα είναι τα εξής:



	Pitch(Hz)	First Formant (Hz)	Second Formant (Hz)	Third Formant (Hz)
“ α ”	134.348	788.670	1070.780	2335.707
“ ου ”	128.609	340.438	1779.100	2296.511
“ ι ”	130.161	379.918	1985.937	2456.853

Γυναίκα – Ομιλήτρια 8

Η κυματομορφή και το σπεκτογράφημα του ηχητικού αρχείου του άντρα είναι τα εξής:



	Pitch	First Formant	Second Formant	Third Formant
“ α ”	177.074	905.138	1648.337	3098.374
“ ου ”	183.161	333.992	1637.166	2645.057
“ ι ”	174.943	336.599	2097.328	2898.747

Από τους παραπάνω πίνακες, παρατηρούμε ότι για υπάρχει μία διαφορά στο pitch μεταξύ του άνδρα (γύρω στα 130 Hz) και της γυναίκας (γύρω στα 180 Hz). Οπότε για την διαφοροποίηση του φύλου του ομιλητή η μέση τιμή του pitch αρκεί. Από την άλλη, μπορούμε να χρησιμοποιήσουμε τα formants, για να διαχωρίσουμε τα φωνήεντα. Το ‘‘α’’, έχει για παράδειγμα, υψηλή τιμή 1^{ου} Formant, οπότε μπορούμε έτσι να το διαχωρίσουμε από τα άλλα. Από την άλλη, η διαφοροποίηση των άλλων 2 φωνηέντων είναι πιο δύσκολη, με βάση τις τιμές των 3 πρώτων Formants.

Βήμα 2

Σε πρώτη φάση, θα εισάγουμε τα δεδομένα μας – εκφωνήσεις των ψηφίων. Για κάθε μία από τις εκφωνήσεις, θα διαχωρίσουμε το ψηφίο που εκφωνήθηκε καθώς και τον αντίστοιχο ομιλητή. Για το διάβασμα των wav αρχείων θα χρησιμοποιήσουμε την συνάρτηση load της librosa.

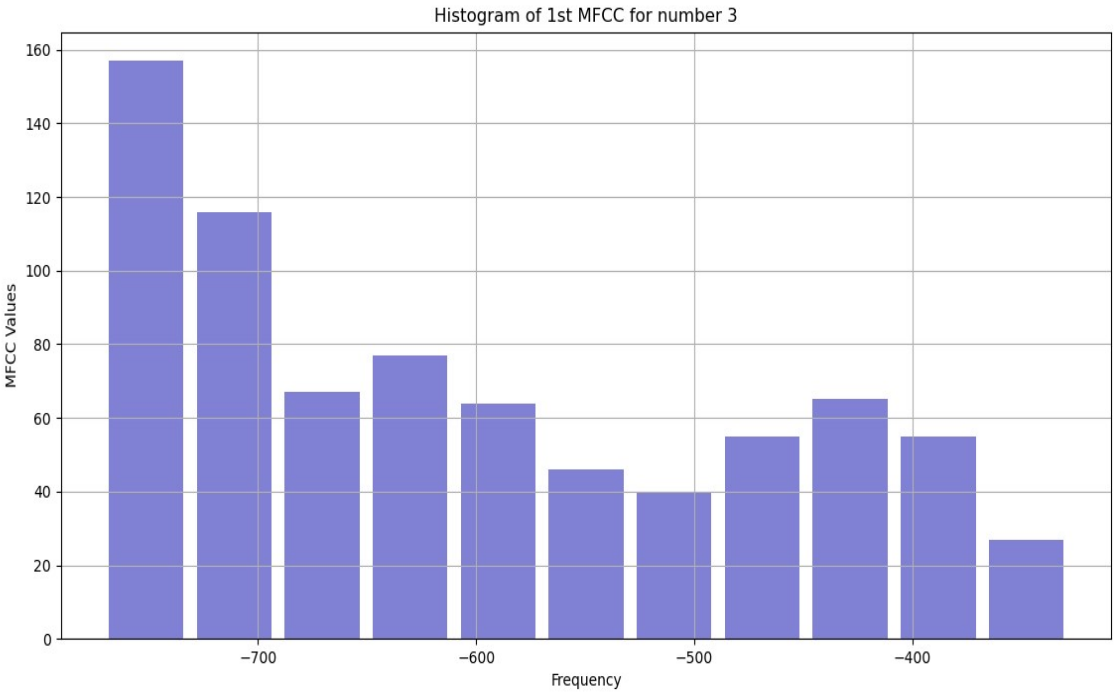
Βήμα 3

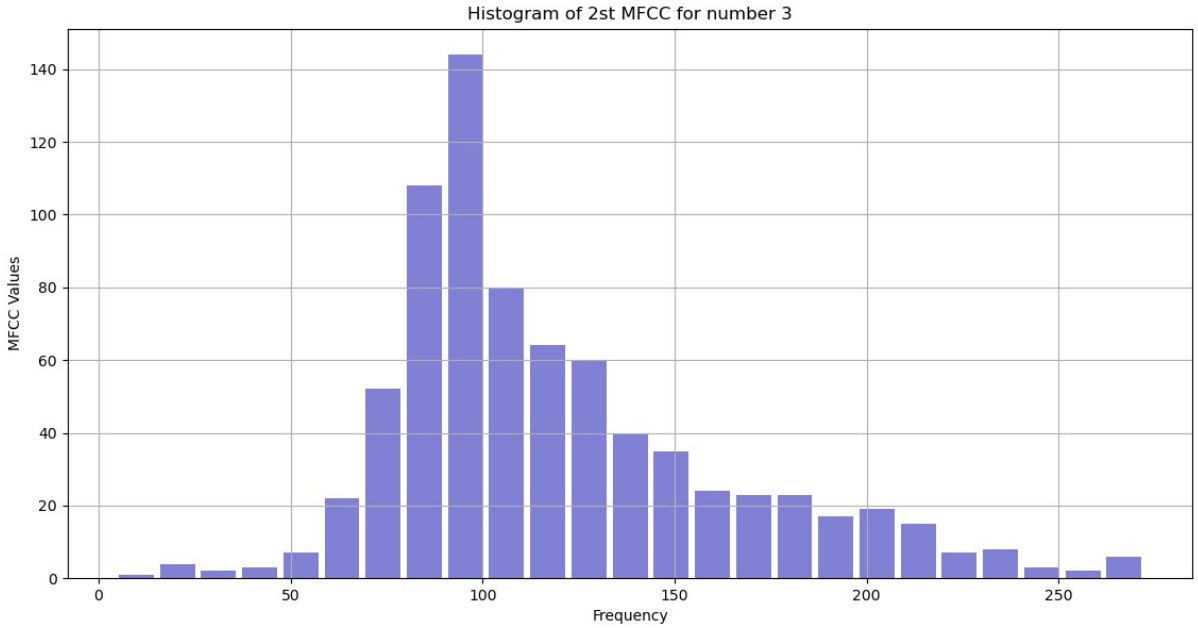
Για κάθε ένα από τα αρχεία ήχου που φορτώσαμε προηγουμένως, θα τα χωρίσουμε σε επικαλυπτόμενα παράθυρα σταθερού μήκους (25 msec) και θα εξάγουμε Mel-Frequency Cepstral Coefficients (MFCCs), χρησιμοποιώντας την συνάρτηση feature.mfcc( ) της librosa. Ύστερα, θα υπολογίσουμε και τις τοπικές παραγώγους $1^{ης}$ και $2^{ης}$ τάξης των παραπάνω χαρακτηριστικών με την βοήθεια της συνάρτησης librosa.delta της librosa με ορίσματα τα mfcc που βρήκαμε πριν και order = 1 ή 2 , αναλόγως ποιες παραγώγους θέλουμε να βρούμε.

Βήμα 4

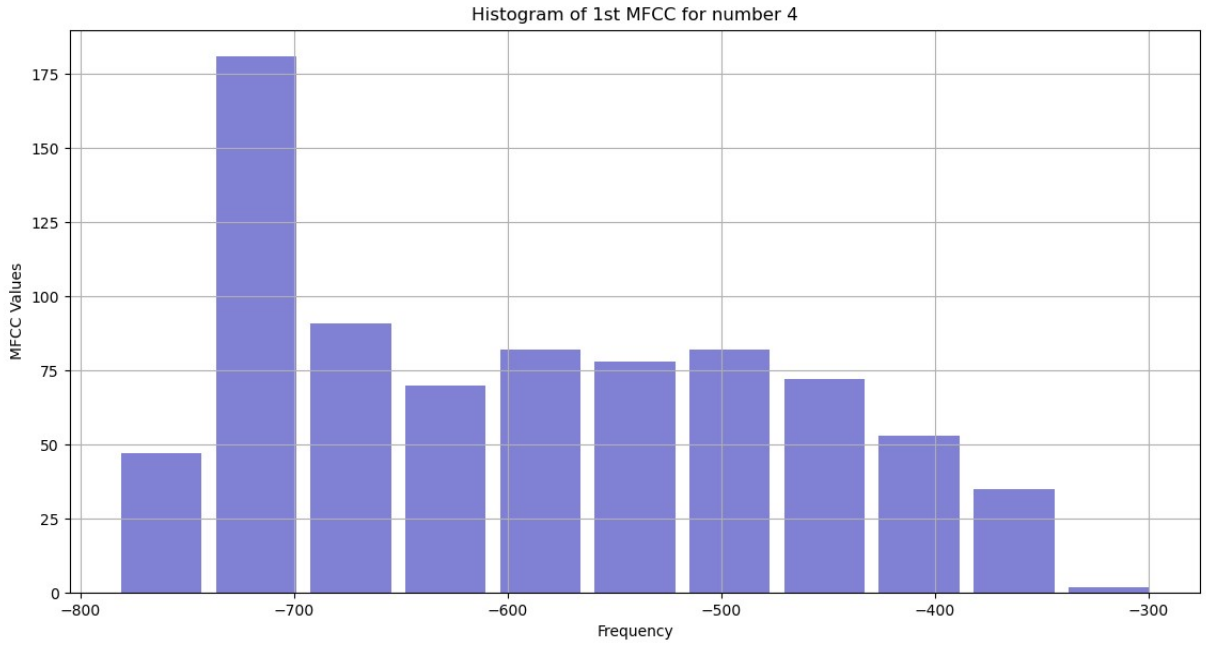
Στο σημείο αυτό θα αναπαραστήσουμε τα ιστογράμματα των $1^{ων}$ και $2^{ων}$ MFCC των ψηφίων 3 και 4.

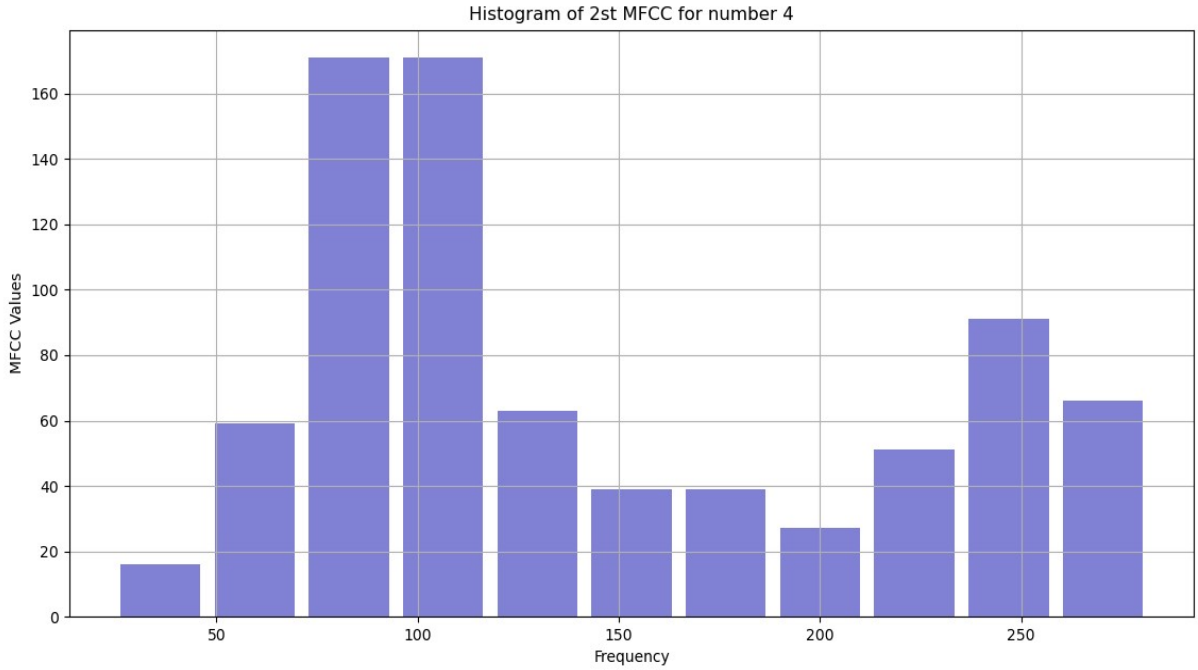
<u>Για το 'Three':</u>




Histogram of 2st MFCC for number 3
MFCC Values
Frequency
0
20
40
60
80
100
120
140
0
50
100
150
200
250

Για το ‘Four’:

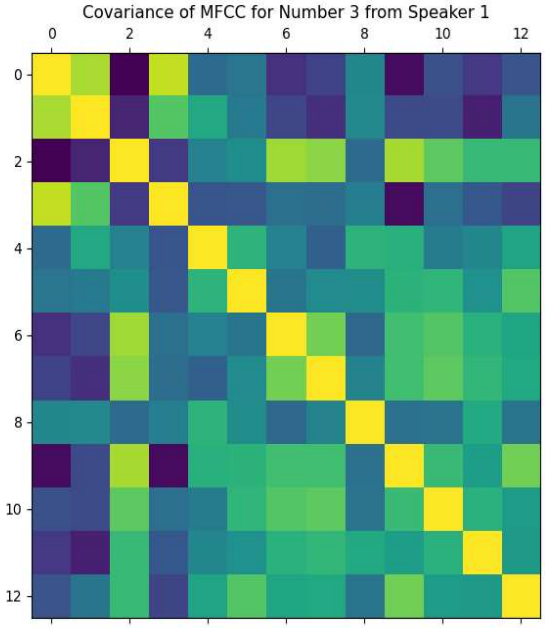

Histogram of 1st MFCC for number 4
MFCC Values
Frequency
0
25
50
75
100
125
150
175
−800
−700
−600
−500
−400
−300

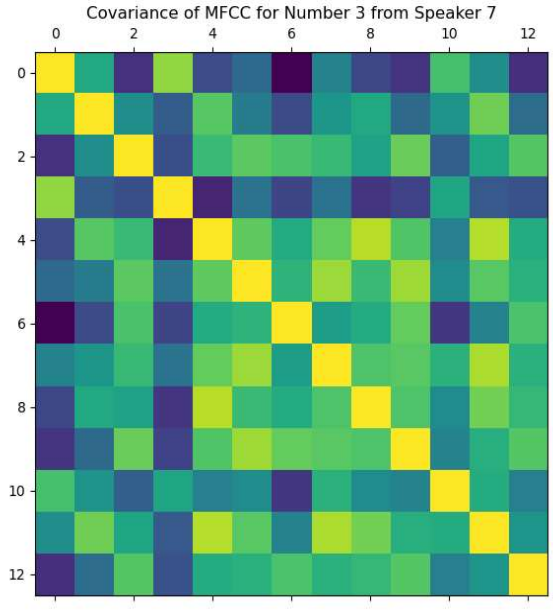


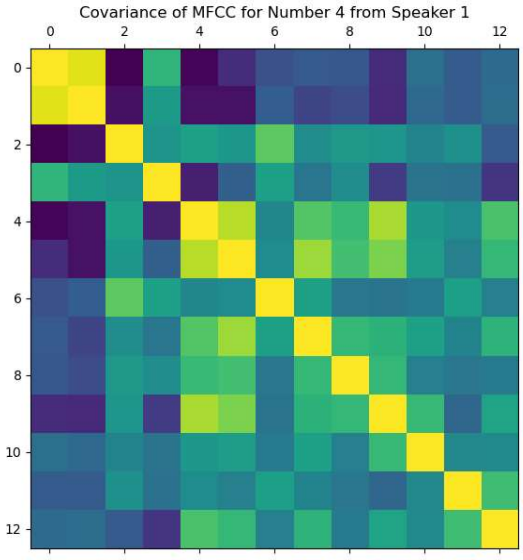
Στην συνέχεια, για τα ψηφία 3 και 4, θα εξάγουμε τα Mel Filterbank Spectral Coefficients, τα οποία προκύπτουν εκτελώντας αντίστροφο μετασχηματισμό DCT στους συντελεστές MFCC. Για να λάβουμε τις τιμές αυτές, μπορούμε να χρησιμοποιήσουμε την συνάρτηση feature.melspectogram της librosa.

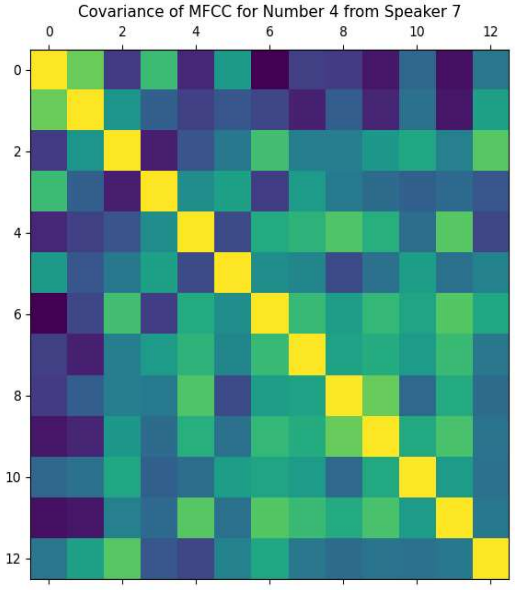
Για κάθε ένα από τα 2 αυτά ψηφία, επιλέγουμε 2 εκφωνήσεις (την $1^{η}$ και $7^{η}$) και απεικονίζουμε γραφικά την συσχέτιση των MFSCs και MFCCs.

<u>Για τα MFCCs:</u>

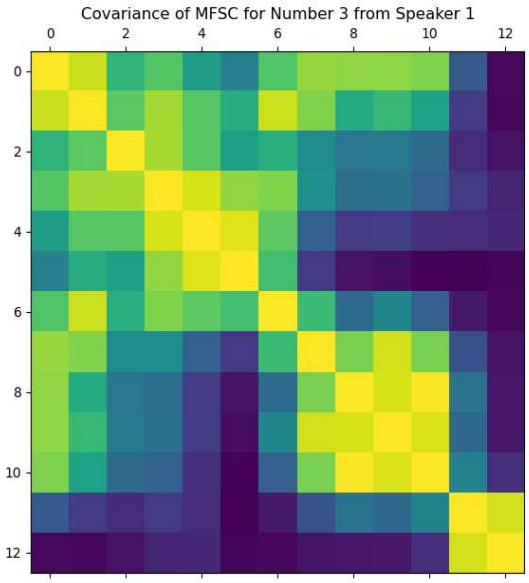

Covariance of MFCC for Number 3 from Speaker 1
0
2
4
6
8
10
12

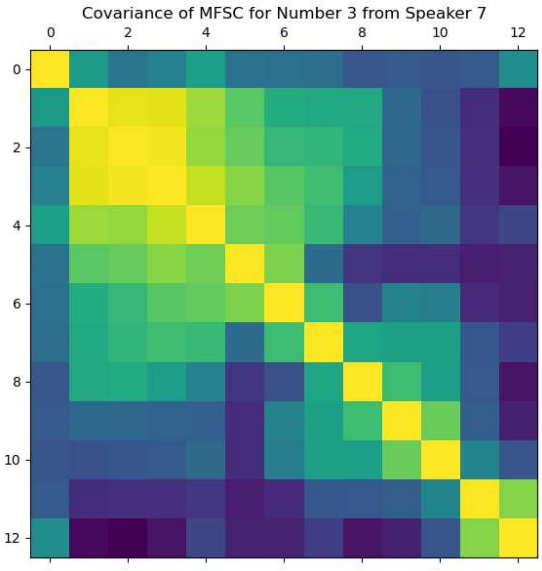

Covariance of MFCC for Number 3 from Speaker 7
0
2
4
6
8
10
12

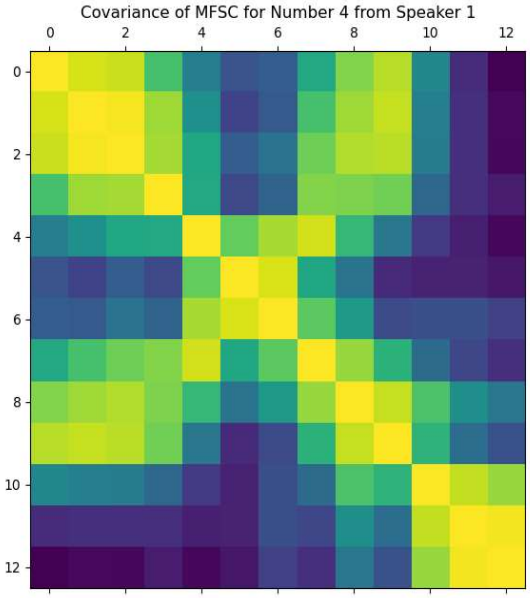

Covariance of MFCC for Number 4 from Speaker 1
0
2
4
6
8
10
12

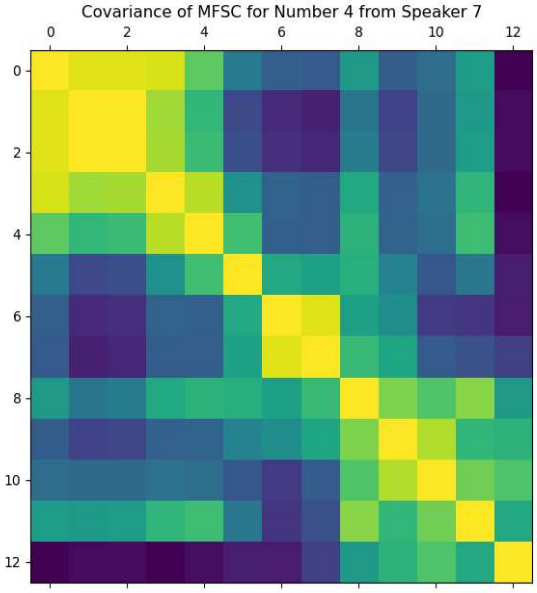

Covariance of MFCC for Number 4 from Speaker 7
0
2
4
6
8
10
12

Για τα MFSCs:


Covariance of MFSC for Number 3 from Speaker 1
0
2
4
6
8
10
12


Covariance of MFSC for Number 3 from Speaker 7
0
2
4
6
8
10
12



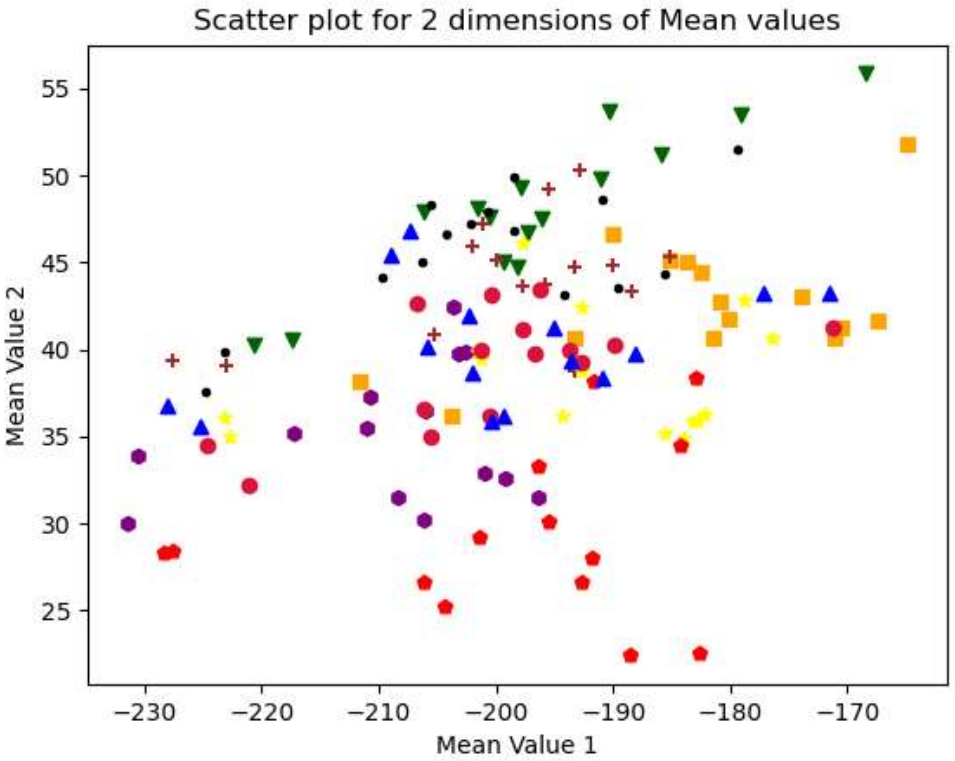


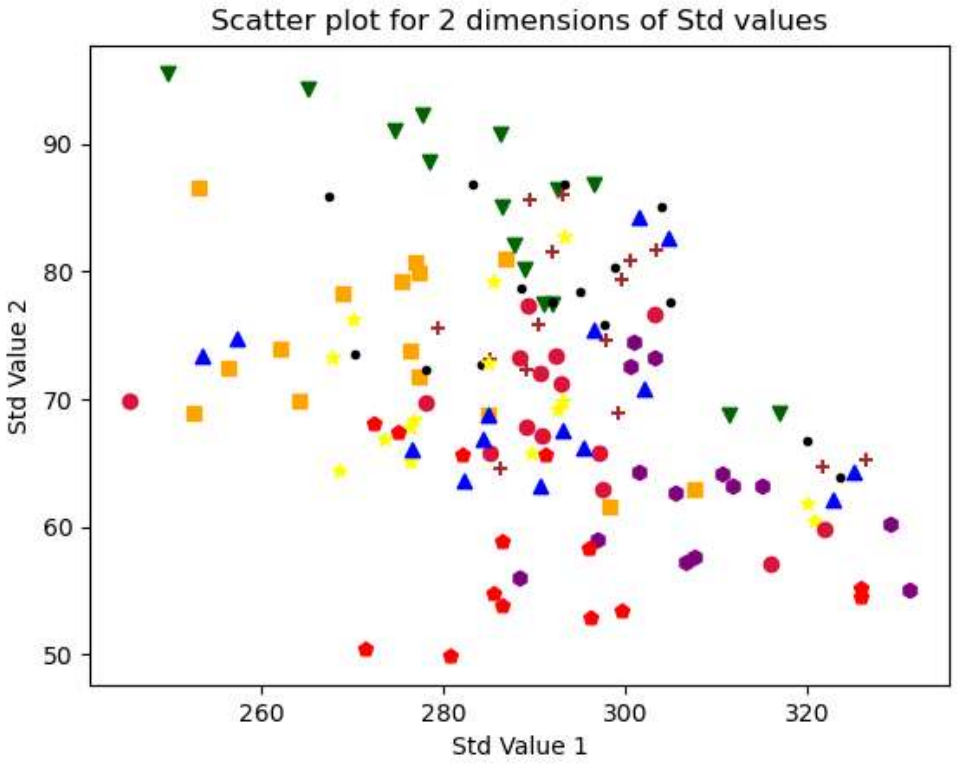
Από τους παραπάνω πίνακες, παρατηρούμε ότι οι τιμές των ετεροσυσχετίσεων στους συντελεστές MFSC είναι αρκετά μεγαλύτερες από ότι των αντίστοιχων MFCC. Αυτό σημαίνει ότι οι MFSC συντελεστές δεν περιέχουν όση πληροφορία όση οι MFCC και αυτός είναι και ο κύριος λόγος για τον οποίο προτιμούν τους MFCC συντελεστές για την ανάλυση των ηχητικών σημάτων.
Σημαντικό ρόλο στο παραπάνω παίζει ο DCT μετασχηματισμός που επιβάλλουμε στους MFSC συντελεστές προκειμένου να πάρουμε τους MFCC.

Βήμα 5

Προκειμένου να μπορέσουμε να κάνουμε ταξινόμηση των ψηφίων, θα εξάγουμε για κάθε εκφώνηση ένα διάνυσμα των μέσων τιμών και ένα διάνυσμα για τις διασπορές των συντελεστών MFCC. Το μέγεθος αυτών των διανυσμάτων είναι προφανώς 13.
Επειδή το πλήθος των παραθύρων κάθε εκφώνησης ποικίλει ανάλογα με την διάρκεια του κάθε ηχητικού αρχείου, αρχικά κάνουμε concatenate στις τιμές των MFCC και των τοπικών παραγώγων τους και στην συνέχεια, θα υπολογίσουμε τις μέσες τιμές και τις διασπορές για κάθε έναν από τους 13 συντελεστές.

Θα αναπαραστήσουμε γραφικά τις τιμές – διανύσματα των 2 πρώτων συντελεστών MFCC για κάθε εκφώνηση, για να δούμε αν είναι δυνατή η κατηγοριοποίηση τους.





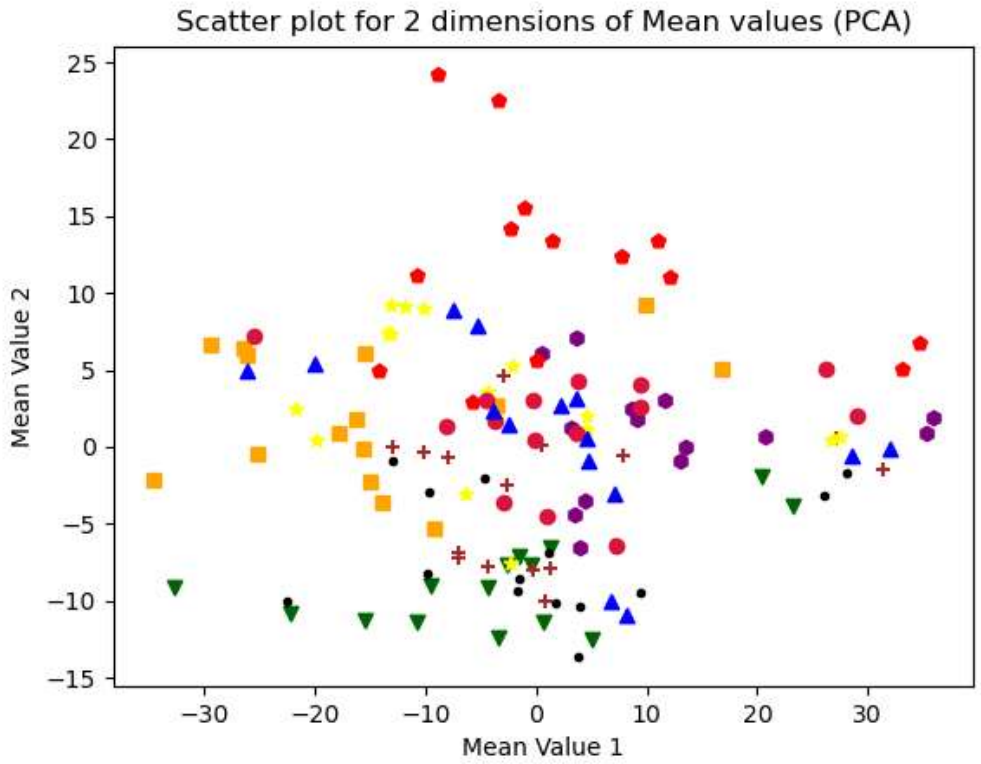
Παρατηρούμε πως παρότι υπάρχει μια μικρή ομαδοποίηση των 9 ψηφίων μεταξύ τους, δεν είναι δυνατή η διάκριση τους με μονάχα τα 2 αυτά χαρακτηριστικά, καθώς υπάρχουν πολλές επικαλύψεις μεταξύ τους.

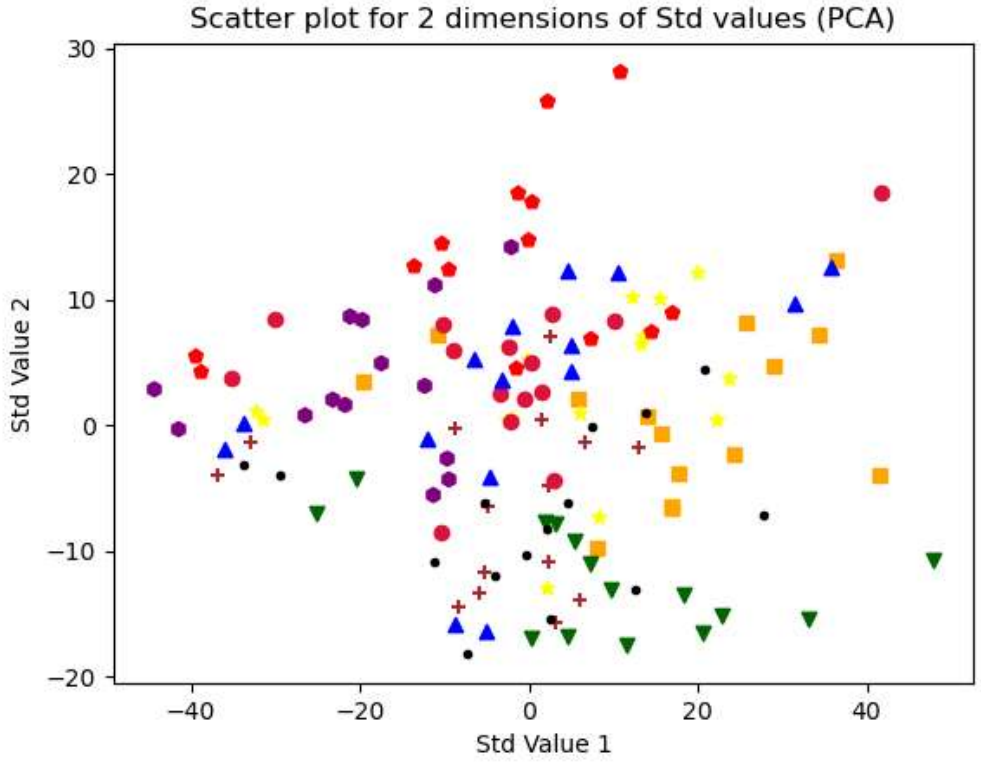
Βήμα 6

Μία καλή μέθοδος για την μείωση των διαστάσεων είναι η εφαρμογή της μεθόδου PCA με την οποία μπορούμε να συμπιέζουμε τα δεδομένα μας αποτελεσματικά, με τις λιγότερες δυνατές απώλειες.
Στο σημείο αυτό, θα εφαρμόσουμε την τεχνική PCA στα παραπάνω διανύσματα των 13 διαστάσεων, προκειμένου να μειώσουμε τις διαστάσεις τους σε 2 (και έπειτα 3) και στο τέλος να τα αναπαριστούμε γραφικά, για να δούμε αν είναι πλέον εφικτός ο διαχωρισμός τους.
Τονίζεται σε αυτό το σημείο ότι για να είναι επιτυχημένη η εφαρμογή της PCA, θα πρέπει να εφαρμοστεί πρώτα μία κανονικοποίηση στα δεδομένα.

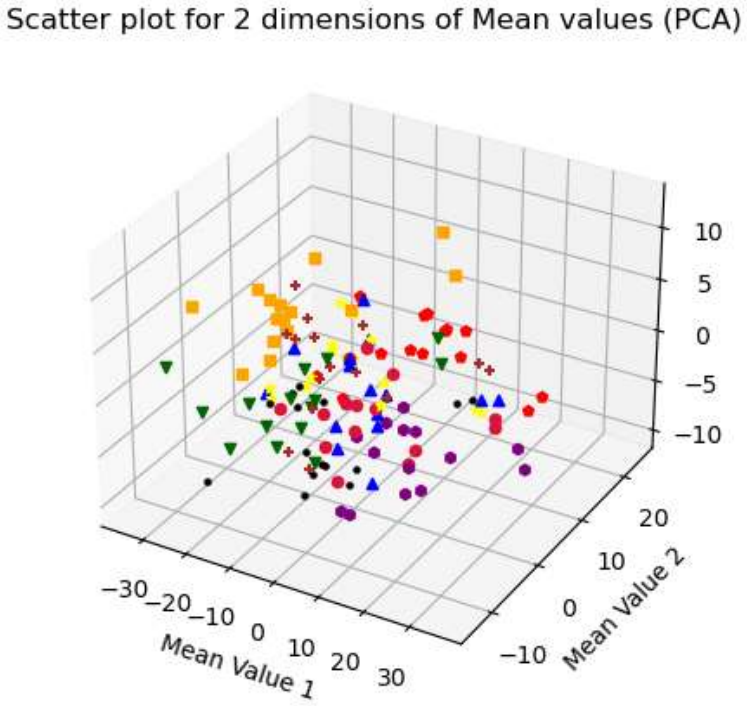
Οι γραφικές παραστάσεις, που προκύπτουν είναι οι εξής:

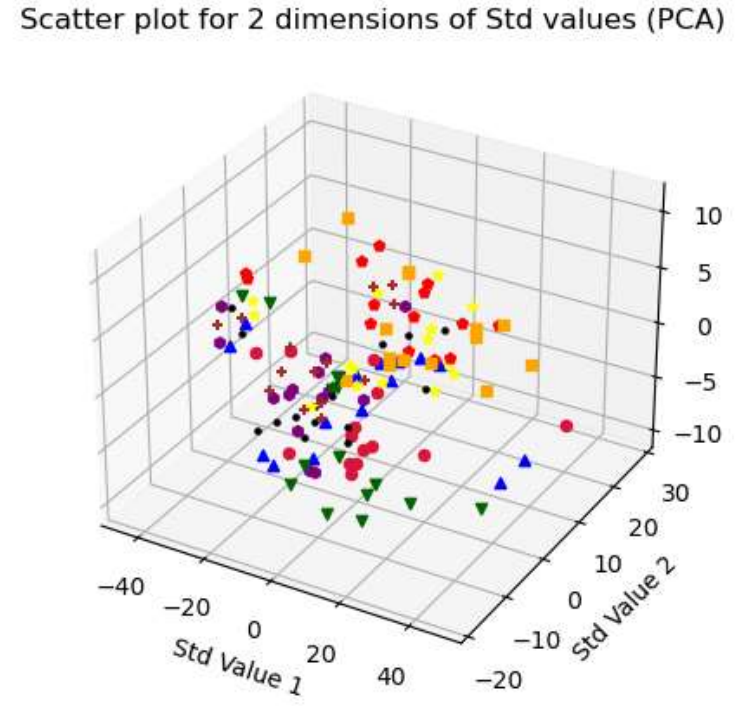
PCA σε 2 διαστάσεις:





PCA σε 3 διαστάσεις:





Η χρήση της PCA μας βοηθάει να διαχωρίσουμε τα δεδομένα μας λίγο καλύτερα, και με αυτόν τον τρόπο το classification, χρησιμοποιώντας μονάχα μερικές από τις διαστάσεις των δεδομένων, θα γίνει πιο αποτελεσματικό.

Για να υπολογίσουμε το ποσοστό της αρχικής διασποράς που διατηρούν οι συνιστώσες που προέκυψαν, τότε χρησιμοποιούμε το object explained_variance_ratio_ της PCA της sklearn.

	Mean	**Std**
2 dimensional	0,82620951	0,86924178
3 dimensional	0,88395442	0,91075584

Παρατηρούμε από τον παραπάνω πίνακα, ότι η εφαρμογή του PCA κρίνεται πετυχημένη, καθώς καταφέρνουμε να διατηρήσουμε ένα μεγάλο ποσοστό της αρχικής πληροφορίας στις 2 και 3 συνιστώσες που ζητήσαμε. Μάλιστα, προσθέτοντας και την $3^{η}$ διάσταση μειώνουμε την απώλεια πληροφορίας περαιτέρω.

Βήμα 7

Θα χωρίσουμε το σύνολο των δεδομένων μας (τα 133 wav αρχεία) σε train και test σε ποσοστό 70 – 30 %, ώστε να εκπαιδεύσουμε και τεστάρουμε τα μοντέλα μας.
Για τον σκοπό αυτό, θα χρησιμοποιήσουμε τα διανύσματα των μέσων τιμών και των διασπορών, που υπολογίσαμε στο βήμα 5.

Οι ταξινομητές που θα χρησιμοποιήσουμε είναι οι εξής:

- Naïve Bayes (Της Sklearn και ο Custom της προηγούμενης άσκησης)
- Nearest Neighbor
- Logistic Regression
- SVM (RBF)

Χωρίς την χρήση επιπλέον χαρακτηριστικών, τα ποσοστά επιτυχίας των παραπάνω ταξινομητών είναι τα εξής (ο custom naive bayes και ο naive bayes της sklearn δίνουν πάντοτε ίδια ποσοστά επιτυχίας):

Classifier	Accuracy
Naive Bayes	0.4
Nearest Neighbor	0.75
Logistic Regression	0.65
SVM (Rbf)	0.55

Μπορούμε να προσθέσουμε επιπλέον χαρακτηριστικά – features, σε κάθε μία από τις εκφωνήσεις, προκειμένου να αυξήσουμε τα ποσοστά επιτυχίας των ταξινομητών. Ένα τέτοιο παράδειγμα, το οποίο θα εφαρμόσουμε στο παράδειγμα μας είναι το zero cross rate. Παρόλα αυτά, επειδή μονάχα η προσθήκη του zero cross rate δεν βελτιώνει ιδιαίτερα την απόδοση των μοντέλων, κρίνεται αναγκαία η προσθήκη και άλλων features της librosa (με προσθήκη του zero cross rate βελτιώνεται μόνο το accuracy του Naive Bayes), αν θέλουμε περαιτέρω αύξηση του accuracy.

Classifier	Accuracy
Naive Bayes	**0.45**
Nearest Neighbor	0.75
Logistic Regression	0.65
SVM (Rbf)	0.55

Βήμα 8

Στο βήμα αυτό, θα εξοικειωθούμε με την δημιουργία ενός αναδρομικού νευρωνικού δικτύου. Παράγουμε 100 ακολουθίες ημιτόνων, 10 δειγμάτων η καθεμία, ίδιας συχνότητας αλλά διαφορετικής αρχικής φάσης. Στόχος μας είναι η εκπαίδευση ενός RNN, το οποίο θα προβλέπει τις τιμές των αντίστοιχων συνημιτόνων (ίδιας συχνότητας και ίδιας αρχικής φάσης). Από τις 100 αυτές τις ακολουθίες, θα χρησιμοποιήσουμε τις 90 για την εκπαίδευση του δικτύου, ενώ τις υπόλοιπες 10 για τον έλεγχο και το testing. Ως loss function χρησιμοποιούμε το μέσο τετραγωνικό σφάλμα MSE.

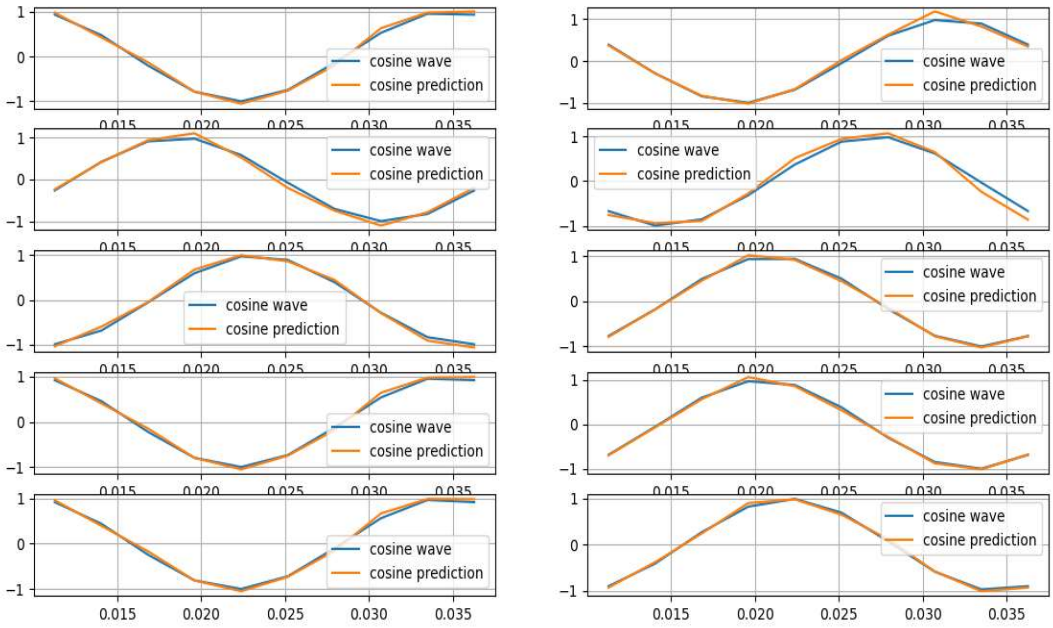
Όπως είναι αναμενόμενο, η εκπαίδευση το δικτύου με πολλές εποχές μειώνει σε μεγάλο βαθμό το συνολικό loss των προβλέψεων των τιμών του συνημιτόνου. Παρόλα αυτά, κατά το τρέξιμο του αλγορίθμου, και ειδικά στην περίπτωση που χρησιμοποιούμε ως optimizer για το δίκτυο

μας τον αλγόριθμο Adam, παρατηρούμε ότι το δίκτυο μαθαίνει με αρκετά γρήγορο ρυθμό. Αυτό σημαίνει ότι ακόμα και μετά από μικρό αριθμό εποχών το loss έχει ελαττωθεί σημαντικά και προσεγγίζουμε αρκετά καλά την επιθυμητή κυματομορφή.
Αντί για χρήση RNN, μπορούν να χρησιμοποιηθούν και 2 άλλες μονάδες: η LSTM και η GRU.
Το πρόβλημα, το οποίο παρατηρείται στα RNN δίκτυα είναι ότι δεν μπορούν να διαχειριστούν, με τον βέλτιστο τρόπο, δεδομένα τα οποία εμφανίζουν μεταξύ τους long term dependencies. Ακριβώς αυτό το πρόβλημα αντιμετωπίζουν τα LSTM και GRU, που στην ουσία αποτελούν παραλλαγές των RNN δικτύων.
Δεδομένου πως η απόσταση μεταξύ των δειγμάτων, τα οποία χρησιμοποιούμε είναι μικρή και επειδή χρησιμοποιούμε σχετικά λίγα δείγματα για την εκπαίδευση του δικτύου, η προσθήκη των LSTM και GRU δεν θα βελτιώσει ιδιαίτερα την επίδοση (εξάλλου το δίκτυο μας προβλέπει με καλή ακρίβεια τις ζητούμενες τιμές.

Για 3 εποχές, έχουμε τα εξής αποτελέσματα, όσον αφορά το loss:

```
Epoch: 0     Batch: 0    Loss 0.4979723393917084
Epoch: 1     Batch: 0    Loss 0.10314254462718964
Epoch: 2     Batch: 0    Loss 0.013856747187674046
```

Για τα test δείγματα, οι κυματομορφές οι οποίες λαμβάνουμε την έξοδο είναι οι εξής:

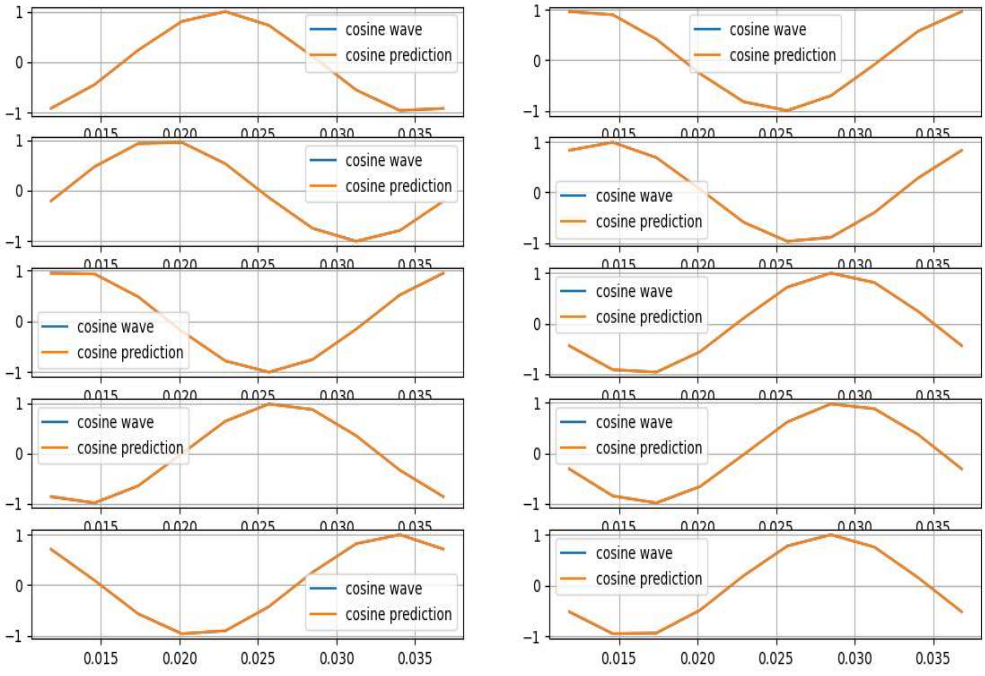


Παρατηρούνται μικρές αποκλίσεις από τις κανονικές τιμές, τις οποίες όμως μπορούμε να τις εξαλείψουμε με αύξηση του αριθμού των εποχών εκπαίδευσης του δικτύου.

Για 20 εποχές, έχουμε τα εξής αποτελέσματα, όσον αφορά το loss:

```
Epoch: 0     Batch: 0    Loss 0.4962712228298187
Epoch: 1     Batch: 0    Loss 0.09043287485837936
Epoch: 2     Batch: 0    Loss 0.027358002960681915
Epoch: 3     Batch: 0    Loss 0.006979621481150389
Epoch: 4     Batch: 0    Loss 0.00601044949144125
Epoch: 5     Batch: 0    Loss 0.0030948526691645384
Epoch: 6     Batch: 0    Loss 0.001121971639804542
Epoch: 7     Batch: 0    Loss 0.0006300051463767886
Epoch: 8     Batch: 0    Loss 0.0005494130309671164
Epoch: 9     Batch: 0    Loss 0.00021062603627797216
Epoch: 10    Batch: 0    Loss 0.0003881672746501863
Epoch: 11    Batch: 0    Loss 0.0001177747908513993
Epoch: 12    Batch: 0    Loss 0.0001395029539708048
Epoch: 13    Batch: 0    Loss 7.85856245784089e-05
Epoch: 14    Batch: 0    Loss 7.282414298970252e-05
Epoch: 15    Batch: 0    Loss 6.171334098326042e-05
Epoch: 16    Batch: 0    Loss 2.606741873023566e-05
Epoch: 17    Batch: 0    Loss 1.0328640200896189e-05
Epoch: 18    Batch: 0    Loss 1.1115525921923108e-05
Epoch: 19    Batch: 0    Loss 8.550078746338846-06
```

Για τα test δείγματα, οι κυματομορφές οι οποίες λαμβάνουμε την έξοδο είναι οι εξής:



Σε αυτή την περίπτωση, παρατηρούμε πλήρη ταύτιση με τις ζητούμενες κυματομορφές.

Βήμα 9

Χρησιμοποιούμε τις 5 συναρτήσεις του αρχείου parser.py, για να αποθηκεύσουμε τα δεδομένα (πληροφορία , ids, εκφωνητές, labels) των ηχητικών σε πίνακες, ώστε να εξάγουμε σε μία λίστα 13 features για κάθε παράθυρο κάθε ηχητικού (όχι σε πίνακα λόγω του μεταβλητού αριθμού παραθύρων), να τα χωρίσουμε σε train-test δεδομένα με αναλογία 90%-10% (τα ids 0-4 είναι τα test δεδομένα και τα ids 5-49 τα train δεδομένα) και να τα κανονικοποιήσουμε, με χρήση του StandardScaler() της sklearn. Στη συνέχεια, χωρίζουμε τα train δεδομένα μας σε training και validation με ποσοστό 80%-20%, ώστε να μπορούμε να κάνουμε evaluate το μοντέλο μας μετά την εκπαίδευσή του. Η παραπάνω διαδικασία πραγματοποιείται με χρήση της συνάρτησης train_test_split με ορίσματα τα δεδομένα και τα labels των, αρχικών train δεδομένων μας, test_size = 0.20, stratify = y_train (θέλουμε stratifiedsplit).

Βήμα 10

Στο συγκεκριμένο βήμα δημιουργούμε 10 GMM-HMM μοντέλα (ένα για κάθε αριθμό). Αρχικά, κάνουμε append στη λίστα data όλα τα train δεδομένα με label τον αντίστοιχο αριθμό. Στον X θέλουμε να είναι όλα τα παράθυρα με τα 13 features στη σειρά (έχει σταθερές διαστάσεις, επομένως μπορεί να είναι και πίνακας), άρα αρκεί να κάνουμε concatenate τον πίνακα data. Στη συνέχεια, επιλέγουμε τον αριθμό των states του HMM και των γκαουσιανών του GMM και δημιουργούμε τα μείγματα για κάθε state, χρησιμοποιώντας την GeneralMixtureModel( ) της pomegranate.

Αρχικοποιούμε τον πίνακα A = $\{\alpha_{ij}\}$, που δηλώνει την πιθανότητα να μεταβούμε από το state i στο state j (n_states*n_states). Γνωρίζουμε, επίσης, ότι το μοντέλο είναι της μορφής left-right (άρα ο A είναι άνω τριγωνικός) και ότι επιτρέπονται μεταβάσεις μόνο μεταξύ διαδοχικών καταστάσεων.

Υποθέτοντας πως έχουμε πχ. 3 states,αρχικοποιούμε τον πίνακα A ως εξής:

$$A = \begin{matrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{matrix}.$$

Οι αρχικές πιθανότητες καταστάσεων (πίνακας starts) είναι [1, 0, 0] και οι τελικές πιθανότητες καταστάσεων (πίνακας ends)είναι [0, 0, 1]. Τέλος, ορίζουμε τα HMMs με τις αρχικοποιήσεις που έχουμε κάνει μέσω της HiddenMarkovModel() της pomegranate.

Βήμα 11

Θα εκπαιδεύσουμε τα μοντέλα μας με χρήση του αλγορίθμου Expectation-Maximizationκαι των δεδομένων, που έχουμε για το κάθε ψηφίο. Επιπλέον, θα ορίσουμε έναν μέγιστο αριθμό επαναλήψεων του ΕΜ, σε περίπτωση που δεν υπάρχει σύγκλιση (πχ. 100). Με αυτά τα δεδομένα, θα κάνουμε fit τα data κάθε ψηφίου στο αντίστοιχο μοντέλο, για να τα εκπαιδεύσουμε.

Βήμα 12

Το τελικό classification θα γίνει με βάση την τιμή του λογαρίθμου της πιθανοφάνειας σε κάθε ένα από τα μοντέλα. Προφανώς, το μοντέλο με τη μεγαλύτερη πιθανοφάνεια θα είναι και η τελική μας πρόβλεψη. Μπορούμε να εξάγουμε την τιμή αυτή μέσω της συνάρτησης Viterbi του HMM.

Μετά από πειράματα σχετικά με τις παραμέτρους της εκπαίδευσης των μοντέλων, states={2, 3, 4}, mixtures = {2, 3, 4}, max_iterations = {20, 50, 100, 200}, καταλήξαμε (στο αναμενόμενο αποτέλεσμα) ότι το καλύτερο accuracy επιτυγχάνεται με 4 states, 4 mixtures και 200 max_iterations.

Αξίζει να σημειωθεί πως η μετρική accuracy με αυτές τις παραμέτρους είχε τιμή σταθερά πάνω από 99%, ενώ υπήρξαν περιπτώσεις που είχε ακόμα και 100%. Συγκρίναμε αυτά τα αποτελέσματα, για να έχουμε την καλύτερη πιθανότητα να προβλέψουμε σωστά τα δεδομένα, καθώς όταν το επαναλάβουμε για τα test δεδομένα, σε κανονικές συνθήκες

δεν θα μπορούμε να έχουμε αποτέλεσμα για την μετρική accuracy(δεν έχουμε τα labels). Επομένως, για να έχουμε όσο το δυνατό μικρότερες αποκλίσεις και υψηλότερες επιδόσεις στα scores των δύο datasets, προσπαθούμε να λάβουμε τις βέλτιστες παραμέτρους.

Βήμα 13

Με τις βέλτιστες παραμέτρους, λοιπόν, παίρνουμε τα ακόλουθα αποτελέσματα σχετικά με το accuracy:

Για το validation data set:

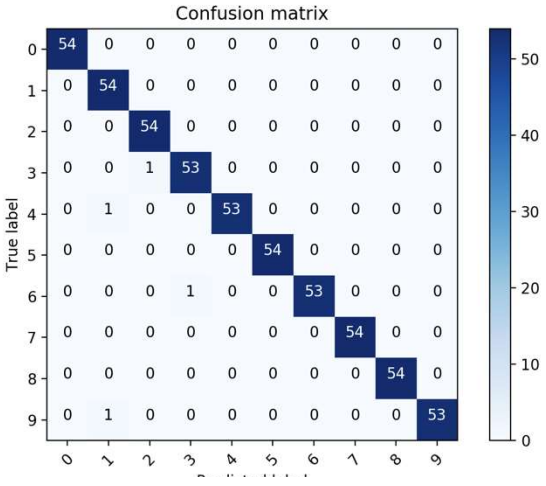
```
Accuracy = : 0.9925925925925926
```

Για το test data set:

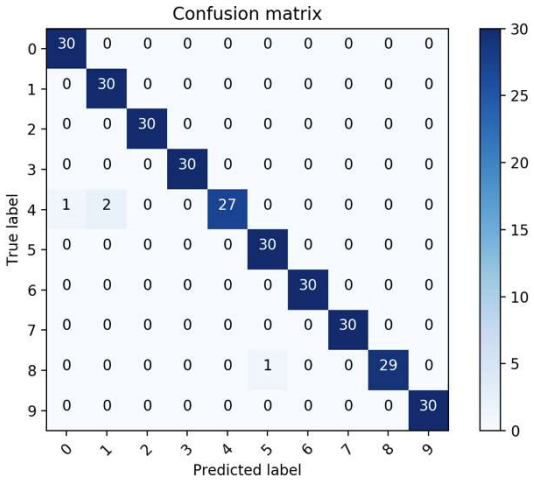
```
Accuracy = : 0.9866666666666667
```

Αντίστοιχα, για τα confusion matricestων δύο sets έχουμε:

Για το validation data set:



Για το test data set:



Βήμα 14

Σε αυτό το σημείο θα εκπαιδεύσουμε ένα LSTM δίκτυο για την αναγνώριση των ψηφίων. Ο λόγος για τον οποίο χρησιμοποιούμε ένα LSTM, αντί για ένα απλό RNN, είναι επειδή, όπως είπαμε προηγουμένως, ικανό να διατηρεί πληροφορίες στην μνήμη, οι οποίες ελήφθησαν πριν από μεγαλύτερο χρονικό διάστημα, λύνοντας με αυτόν τον τρόπο το vanishing gradient problem των RNN.
Το LSTM αποτελείται από 3 πύλες (gates):

- Input Gate: Στο οποίο αποφασίζουμε ποιες τιμές της εισόδου θα χρησιμοποιηθούν και ποίες όχι, ενώ αποφασίζονται και τα βάρη που θα έχουν οι τιμές αυτές.

- Forget Gate: Εξετάζει τι θα χρησιμοποιηθεί και τι όχι από την προηγούμενη κατάσταση.

- Output Gate: Αποφασίζεται ποια θα είναι η τιμή του επόμενου hidden state.

Σε πρώτη φάση, θα υλοποιήσουμε ένα απλό LSTM, για την ταξινόμηση των δειγμάτων, το οποίο θα εκπαιδεύσουμε για 30 εποχές σύνολο με Batch size 20. Συγκεκριμένα, καθώς το πρόβλημα που εξετάζουμε είναι πρόβλημα ταξινόμησης, θα χρησιμοποιήσουμε ως loss function την Cross Entropy Loss, ενώ ως βελτιστοποιητή τον Adam (όπως και στο βήμα 8).

Το training loss κάθε εποχής στην απλή αυτή περίπτωση είναι το εξής:

```
Epoch: 0     Loss 1.6239754347889512
Epoch: 1     Loss 0.6544812701090619
Epoch: 2     Loss 0.392623796645138
Epoch: 3     Loss 0.27302872486136576
Epoch: 4     Loss 0.1904101156871076
Epoch: 5     Loss 0.20038915151316258
Epoch: 6     Loss 0.0978575650240398
Epoch: 7     Loss 0.10423983693881719
Epoch: 8     Loss 0.07142569595965108
Epoch: 9     Loss 0.03648766119214189
Epoch: 10    Loss 0.02489808314540251
Epoch: 11    Loss 0.057601604576188105
Epoch: 12    Loss 0.07292402697796072
Epoch: 13    Loss 0.09925288727282788
Epoch: 14    Loss 0.045130638356096354
Epoch: 15    Loss 0.04210611563111241
Epoch: 16    Loss 0.03378335681756855
Epoch: 17    Loss 0.028296179249149712
Epoch: 18    Loss 0.02737935793724049
Epoch: 19    Loss 0.029265945239415118
Epoch: 20    Loss 0.02610037424623173
Epoch: 21    Loss 0.0075685877177252055
Epoch: 22    Loss 0.0029871800474615563
Epoch: 23    Loss 0.0021639641927322373
Epoch: 24    Loss 0.0016333746224753904
Epoch: 25    Loss 0.0014138509965656947
Epoch: 26    Loss 0.0011181428401161814
Epoch: 27    Loss 0.000978903516456571
Epoch: 28    Loss 0.0008672573005444267
Epoch: 29    Loss 0.0007574647965224425
```

Έχοντας εκπαιδεύσει, λοιπόν το μοντέλο μας, θα κάνουμε αποτίμηση του χρησιμοποιώντας το validation set. Το training accuracy, που προκύπτει είναι το εξής:

```
522
540
0.9666666666666667
```

Μπορούμε να τροποποιήσουμε το LSTM είναι η εφαρμογή ορισμένων τεχνικών κανονικοποίησης, όπως του Dropout και του L2 regulation. Με το Dropout, αφαιρούμε ορισμένους από τους νευρώνες σε κάθε επανάληψη, και επιχειρούμε να προβλέψουμε την έξοδο με τους εναπομείναντες κόμβους του δικτύου. Με αυτόν τον τρόπο αποφεύγουμε τον κίνδυνο να κάνουμε overfit το μοντέλο μας.
Ομοίως, με την L2 κανονικοποίηση προσπαθούμε να ελαχιστοποιήσουμε την πιθανότητα να γίνει κάποιου είδους overfitting. Για να το πετύχει αυτό, μειώνει σε κάθε επανάληψη ένα μικρό ποσοστό των βαρών (με τρόπο ώστε τα βάρη αυτά να μην γίνουν ποτέ μηδέν).

Εφαρμόζοντας τις 2 αυτές τεχνικές στο μοντέλο μας προκύπτει ότι:

Ένας άλλος τρόπος βελτιστοποίησης του δικτύου είναι με την χρήση της Early Stopping τεχνικής. Σε αυτήν την περίπτωση, εξετάζουμε τις τιμές του loss σε κάθε επανάληψη. Εάν μετά από κάποιο σημείο και έπειτα η τιμή του loss δεν μειώνεται πλέον, θα περιμένουμε για ένα χρονικό διάστημα (στην περίπτωση μας για 2 ακόμη εποχές), ώστε να δούμε αν η τιμή του loss θα μειωθεί περαιτέρω. Σε περίπτωση που κάτι τέτοιο δε συμβεί, θα σταματήσουμε το training του μοντέλου και θα φορτώσουμε το μοντέλο με το ελάχιστο loss. Με αυτόν τον τρόπο θα γλυτώσουμε αρκετό χρόνο από την εκπαίδευση του μοντέλου, μειώνοντας βέβαια, στον αντίποδα, την ακρίβεια του (με την χρήση των checkpoints και την εντολής torch.save πετυχαίνουμε όσον το δυνατόν μικρότερη μείωση στο συνολικό accuracy).

```
Epoch: 0     Loss 1.554809993063962
Epoch: 1     Loss 0.6418743708895313
Epoch: 2     Loss 0.3829841235721553
Epoch: 3     Loss 0.2456371822497911
Epoch: 4     Loss 0.22156111789108426
Epoch: 5     Loss 0.1650975671493345
Epoch: 6     Loss 0.10013546598040396
Epoch: 7     Loss 0.12794048878950653
Epoch: 8     Loss 0.10393812761780012
Early Stopping ... Time to exit
Epoch: 9     Loss 0.021517302840948105
Model from Epoch: 7      and Batch: 0
513
540
0.95
```

Θα εξετάσουμε και την περίπτωση ενός Bidirectional LSTM. Στην περίπτωση αυτή, η πληροφορία που αποθηκεύουμε στην μνήμη και την οποία χρησιμοποιούμε για να εκπαιδεύσουμε το μοντέλο μας προέρχονται τόσο από το παρελθόν, όσο και από το μέλλον. Χρησιμοποιώντας αυτές τις επιπρόσθετες πληροφορίες, μπορούμε να βελτιστοποιήσουμε την επίδοση του δικτύου, αν και προσθέτουμε υπολογιστικό φόρτο και αυξάνουμε την πολυπλοκότητα του μοντέλου.

```
Epoch: 0     Loss 1.3594713862295504
Epoch: 1     Loss 0.38632076806216326
Epoch: 2     Loss 0.22680766946049752
Epoch: 3     Loss 0.11663129711868586
Epoch: 4     Loss 0.12848025498290858
Epoch: 5     Loss 0.08058348473126965
Epoch: 6     Loss 0.06101456627517042
Epoch: 7     Loss 0.04733241486677865
Epoch: 8     Loss 0.019834902042661
Epoch: 9     Loss 0.0724858605104533
Epoch: 10    Loss 0.0434259058814927
Epoch: 11    Loss 0.06500393536838668
Epoch: 12    Loss 0.053890231724061
Epoch: 13    Loss 0.015144931571234742
Epoch: 14    Loss 0.008098830385952842
Epoch: 15    Loss 0.04359179222309548
Epoch: 16    Loss 0.03139323053862123
Epoch: 17    Loss 0.012979970642913098
Epoch: 18    Loss 0.007790219057049534
Epoch: 19    Loss 0.004574059085792181
Epoch: 20    Loss 0.003284187004788287
Epoch: 21    Loss 0.002537793116331428
Epoch: 22    Loss 0.002149869776379395
Epoch: 23    Loss 0.0015560233272548499
Epoch: 24    Loss 0.0010956195682983551
Epoch: 25    Loss 0.000610891037097274
Epoch: 26    Loss 0.00046540663451769096
Epoch: 27    Loss 0.000359612027077
Epoch: 28    Loss 0.0003021291109622258
Epoch: 29    Loss 0.0002671120131102119
533
540
0.987037037037037
```

Τέλος, για να μειώσουμε επιπλέον τον υπολογιστικό φόρτο του μοντέλου καθώς και τον χρόνο εκπαίδευσης θα κάνουμε χρήση της συνάρτησης pack_padded_sequence της Pytorch. Ουσιαστικά με αυτόν τον τρόπο, κατά την διάρκεια του training δεν θα λάβουμε υπόψη τις γραμμές, με τις οποίες κάναμε προηγουμένως zero padding τα δεδομένα εκπαίδευσης. Ούτως ή άλλως το γεγονός ότι όλα αυτά τα δεδομένα είναι μηδενικά, σημαίνει ότι οι παράγωγοι (gradients) μηδενίζονται και έτσι δεν επηρεάζει η παρουσία τους την λειτουργικότητα του μοντέλου, οπότε και μπορούμε να τα αγνοήσουμε.
Με αυτόν τον τρόπο, μειώνουμε τον χρόνο εκπαίδευσης του μοντέλου, ενώ παράλληλα η αποτελεσματικότητα του μοντέλου δε μεταβάλλεται.
Για να ελέγξουμε την εγκυρότητα του παραπάνω, θα τρέξουμε τον κώδικα μας για 5 εποχές, χρησιμοποιώντας την συνάρτηση time της βιβλιοθήκης time.

Εφόσον, έχουμε δει όλα τα παραπάνω, μπορούμε να συνδυάσουμε τις παραμέτρους που εξετάσαμε, προκειμένου να βελτιστοποιήσουμε το νευρωνικό LSTM δίκτυο, τόσο ως προς την ταχύτητα, όσο και ως προς την επίδοση του. Αναλυτικότερα, θα χρησιμοποιήσουμε ένα Bidirectional LSTM, με Dropout Layer και L2 regulation, για να αποφύγουμε το overfitting, ενώ επίσης θα χρησιμοποιήσουμε και την Early Stopping τεχνική, για να επιταχύνουμε την διαδικασία εκπαίδευσης του μοντέλου.

Confusion Matrix για το Validation Set