ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ/ΚΩΝ & ΜΗΧ/ΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΗΜΑΤΩΝ, ΕΛΕΓΧΟΥ ΚΑΙ ΡΟΜΠΟΤΙΚΗΣ

Μάθημα: "Αναγνώριση Προτύπων"

9ο εξάμηνο, Ακαδημαϊκό Έτος 2020-21

2η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

Αναγνώριση φωνής με Κρυφά Μαρκοβιανά Μοντέλα και Αναδρομικά Νευρωνικά Δίκτυα

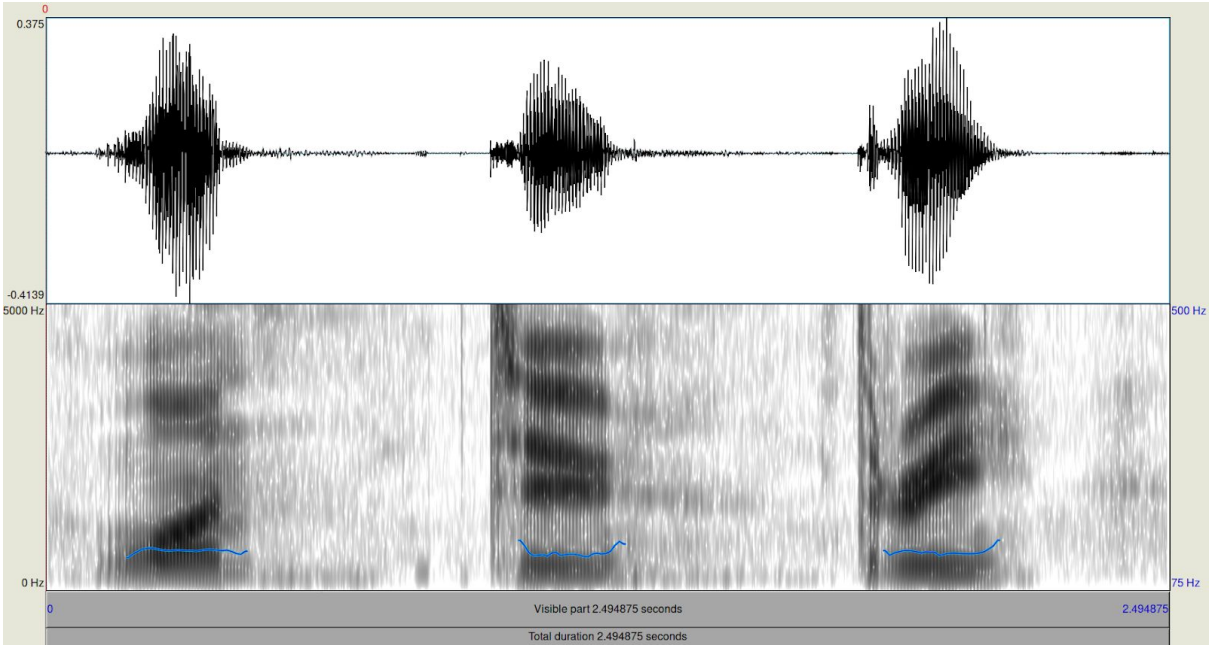
ΓΕΩΡΓΙΟΣ ΜΑΓΚΑΦΩΣΗΣ, 03116125

ΗΛΙΑΣ ΤΡΙΑΝΤΑΦΥΛΛΟΠΟΥΛΟΣ, 03116028

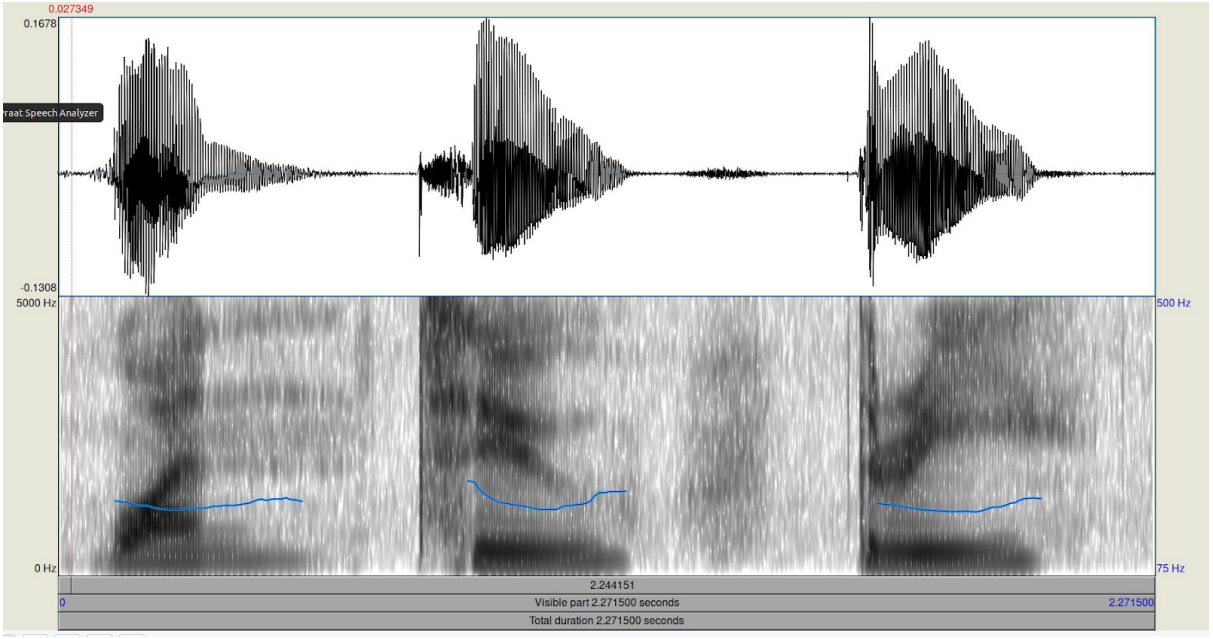
ΒΗΜΑ 1

Χρησιμοποιώντας το εργαλείο Praat, μπορέσαμε να αναλύσουμε και να εξάγουμε σημαντικά συμπεράσματα για δύο αρχεία ήχου που περιέχουν την φράση “one two three” από δύο διαφορετικούς εκφωνητές, έναν άντρα και μία γυναίκα.

Για τον πρώτο ομιλητή (που είναι άντρας), η κυματομορφή (πάνω) και το spectrogram (κάτω) φαίνονται παρακάτω :



Ενώ για τον δεύτερο ομιλητή (που είναι γυναίκα), τα αντίστοιχα διαγράμματα φαίνονται στην παρακάτω εικόνα :



Στη συνέχεια, εξάγουμε τη μέση τιμή του pitch για τα φωνήεντα “α”, “ου” και “ι” στα 3 ψηφία και για τους 2 ομιλητές και οι τιμές παρουσιάζονται στον πίνακα :

φύλο	φωνήεν	mean pitch
male	α	134.5723062614727 Hz
male	ου	132.25016201248198 Hz
male	ι	136.07471764944742 Hz
female	α	175.83033871488075 Hz
female	ου	187.67179954138987 Hz
female	ι	181.8536856222224 Hz

Παρατηρούμε, λοιπόν, ότι η τιμή του pitch είναι ενδεικτική του φύλου του ομιλητή. Όπως είναι λογικό, η φωνή μιας γυναίκας συνηθίζεται να έχει ψηλότερο pitch από εκείνη ενός άντρα. Όπως φαίνεται από τις τιμές, ωστόσο, δεν μπορούμε να συμπεράνουμε κάτι για το φωνήεν που ακούγεται κάθε φορά.

Τέλος, εξάγουμε τα 3 πρώτα formants του κάθε φωνήεντος :

φύλο	φωνήεν	first formant	second formant	third formant
male	α	780.6349199126736 Hz	1214.7880002097445 Hz	2423.6861232854794 Hz
male	ου	340.611674762988 Hz	1776.759546979861 Hz	2222.7291834409384 Hz
male	ι	385.3142125927083 Hz	2009.2899788371653 Hz	2510.3450245951453 Hz
female	α	903.2923712053964 Hz	1857.584713962902 Hz	3166.0838983757294 Hz
female	ου	318.8989688659077 Hz	1183.4539862472623 Hz	2669.100421509304 Hz
female	ι	334.6091057174193 Hz	2689.0823724010584 Hz	3614.980091852231 Hz

Παρατηρούμε τώρα ότι οι τιμές των formants αλλάζουν σημαντικά όταν αλλάζει και το φωνήεν, ενώ παραμένει σε ένα συγκεκριμένο εύρος τιμών όταν αλλάζει ομιλητής. Έτσι, μπορούμε να χρησιμοποιήσουμε τα formants σαν features για την εξαγωγή συμπερασμάτων σχετικά με το είδος του εκάστοτε φωνήεντος, αφού δεν επηρεάζεται σημαντικά από τις διαφορές στην φωνή του κάθε ομιλητή.

ΒΗΜΑ 2

Για τα παρακάτω βήματα, χρησιμοποιήσαμε το dataset που μας δόθηκε και το οποίο αποτελείται από 133 αρχεία ήχου 15 διαφορετικών εκφωνητών που αφορά τους αριθμούς 1 έως 9. Σε αυτό το βήμα φτιάχνουμε μια συνάρτηση parser που θα διαβάζει τα αρχεία ήχου και θα επιστρέφει 3 πίνακες :

- **wavfiles :** περιέχει τα αρχεία ήχου όπως διαβάστηκαν από το librosa. Ένα απόσπασμα των τιμών είναι :

- **ids :** περιέχει τον αριθμό που εκφωνείται κάθε φορά σε string τύπο δεδομένων :

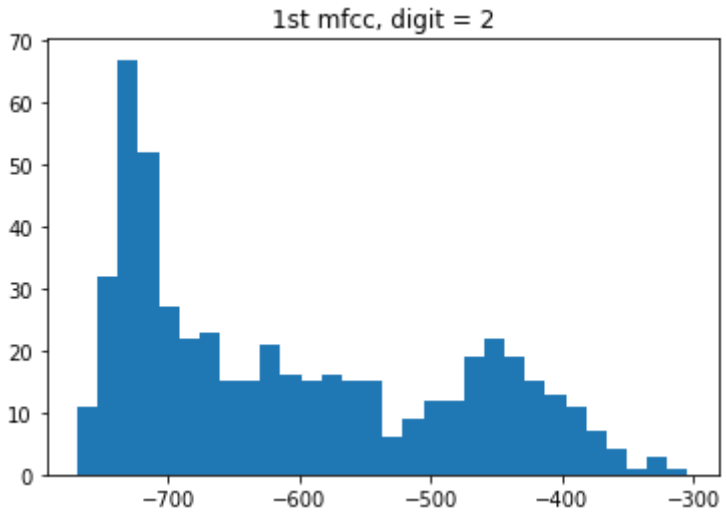
- **speakers :** περιέχει τον αριθμό του εκφωνητή για το κάθε αρχείο ήχου :

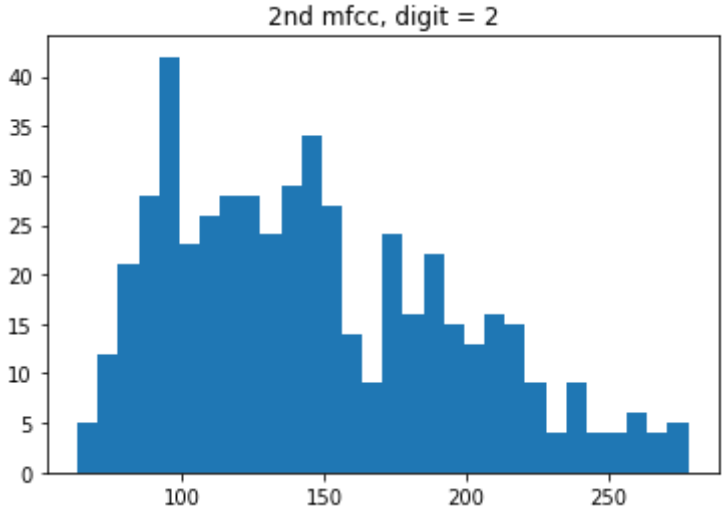
ΒΗΜΑ 3

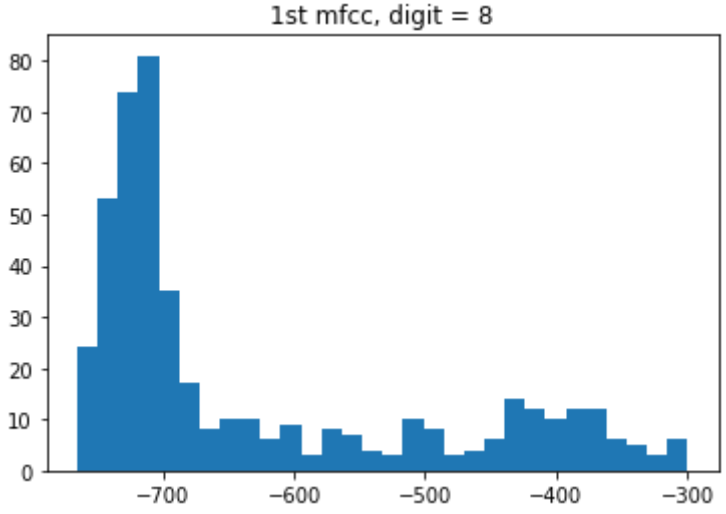
Χρησιμοποιώντας την έτοιμη συνάρτηση *librosa.feature.mfcc* εξάγουμε τα πρώτα 13 Mel-Frequency Cepstral Coefficients για το κάθε αρχείο ήχου, έχοντας ως μήκος παραθύρου 25ms και βήμα 10ms. Κάθε αρχείο ήχου έχει διαφορετική διάρκεια και επομένως προκύπτει διαφορετικό μέγεθος frames από MFCCs. Όμοια, λαμβάνουμε και την πρώτη και την δεύτερη τοπική παράγωγο των χαρακτηριστικών μέσω της έτοιμης συνάρτησης *librosa.feature.delta*, θέτοντας την μεταβλητή order ίση με 1 και 2 αντίστοιχα. Τα MFCCs προκύπτουν μέσα από μια διαδικασία επεξεργασίας του αρχείου ήχου, όπου μεταβαίνουμε αρχικά στο πεδίο της συχνότητας μέσω του power spectrum και στη συνέχεια εφαρμόζουμε mel filterbank και με έναν μετασχηματισμό DCT παίρνουμε το λεγόμενο Cepstrum μεταβαίνοντας στο πεδίο των “quefrency”.

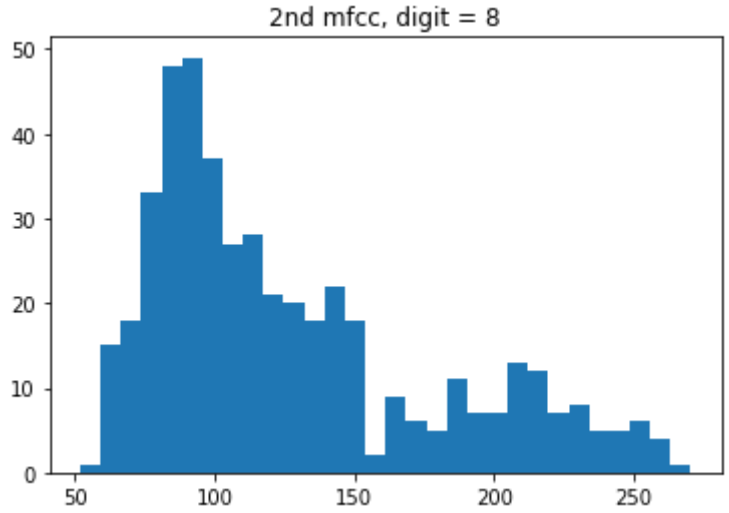
ΒΗΜΑ 4

Υλοποιούμε αυτό το ερώτημα για αριθμό μητρώου 03116028. Έτσι έχουμε n1 = 2 και n2 = 8. Λαμβάνουμε τις τιμές των δύο πρώτων MFCCs, όπως τα υπολογίσαμε παραπάνω για όλες τις εκφωνήσεις του two και του eight. Περνάμε όλες τις εκφωνήσεις σε έναν ενιαίο πίνακα και δημιουργούμε τα ζητούμενα ιστογράμματα :


1st mfcc, digit = 2
70
60
50
40
30
20
10
0
−700
−600
−500
−400
−300


2nd mfcc, digit = 2
40
35
30
25
20
15
10
5
0
100
150
200
250

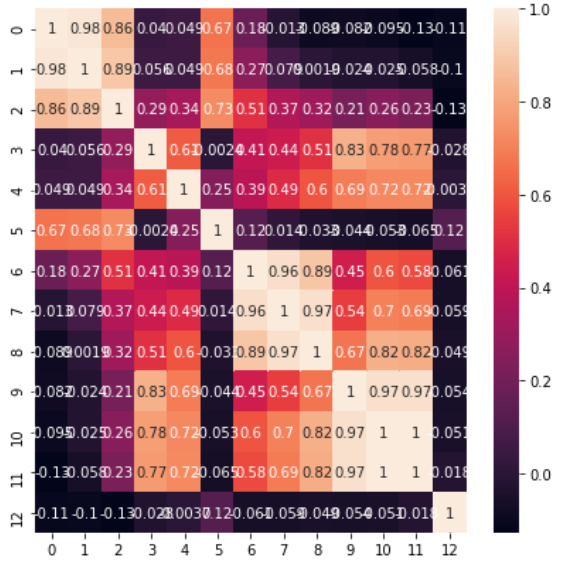

1st mfcc, digit = 8
80
70
60
50
40
30
20
10
0
−700
−600
−500
−400
−300

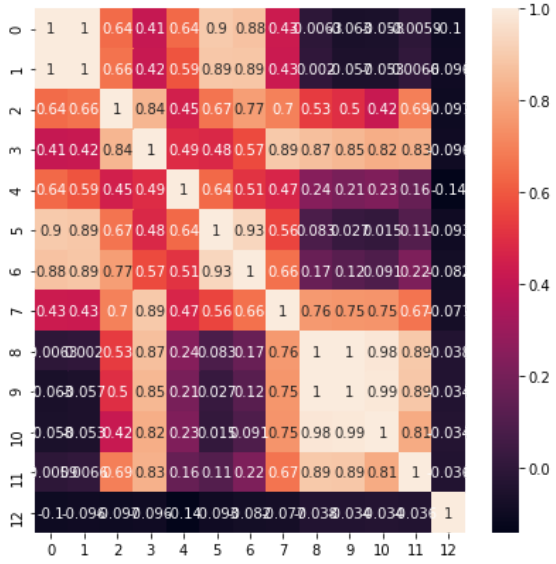


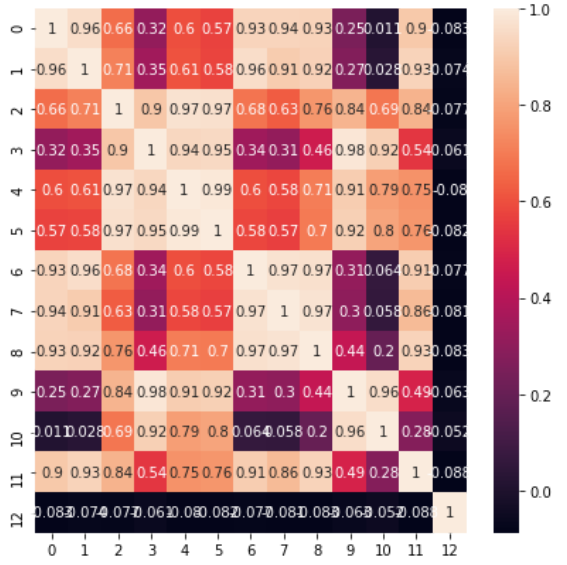
Παρατηρούμε ότι υπάρχει μία απόκλιση μεταξύ των δύο ψηφίων 2 και 8, η οποία όμως δεν είναι ιδιαίτερα σημαντική για το κάθε mfcc ξεχωριστά. Συγκεκριμένα, φαίνεται να ακολουθείται ένα σταθερό μοτίβο, αφού στο 1ο mfcc έχουμε συγκεντρωμένη πληροφορία στις χαμηλές συχνότητες, ενώ στο 2ο mfcc έχουμε μία πιο ομοιόμορφη κατανομή της πληροφορίας. Καταλήγουμε στο συμπέρασμα, ότι τα 2 πρώτα mfccs δεν είναι αρκετά για να αποτελέσουν τα μοναδικά features που θα είναι ικανά να διαχωρίσουν τα ψηφία.

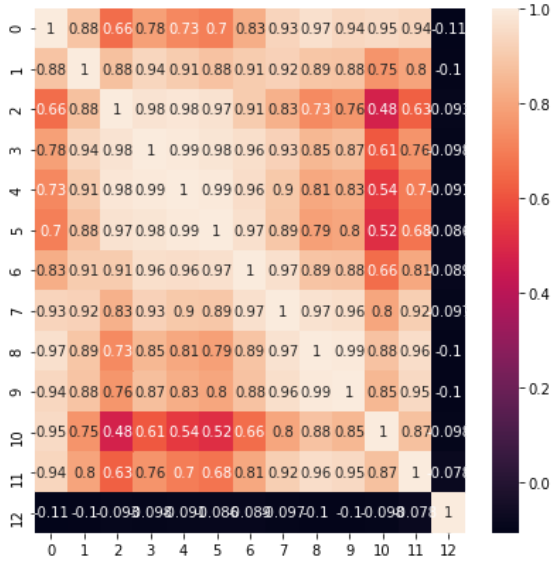
Έπειτα, εξάγουμε τα Mel Filterbank Spectral Coefficients, δηλαδή τα χαρακτηριστικά που εξάγονται αφού εφαρμοστεί η συστοιχία φίλτρων της κλίμακας Mel πάνω στο φάσμα του σήματος φωνής αλλά χωρίς να εφαρμοστεί στο τέλος ο μετασχηματισμός DCT από δύο διαφορετικούς ομιλητές, μέσω της βοηθητικής συνάρτησης *librosa.feature.melspectrogram.* Για τους ίδιους ακριβώς δύο ομιλητές εξάγουμε και τα mfccs. Και στις δύο περιπτώσεις, κρατάμε τους 13 πρώτους συντελεστές.

Με την βοήθεια της *sn.heatmap* αναπαριστούμε γραφικά τη συσχέτιση των MFSCs :

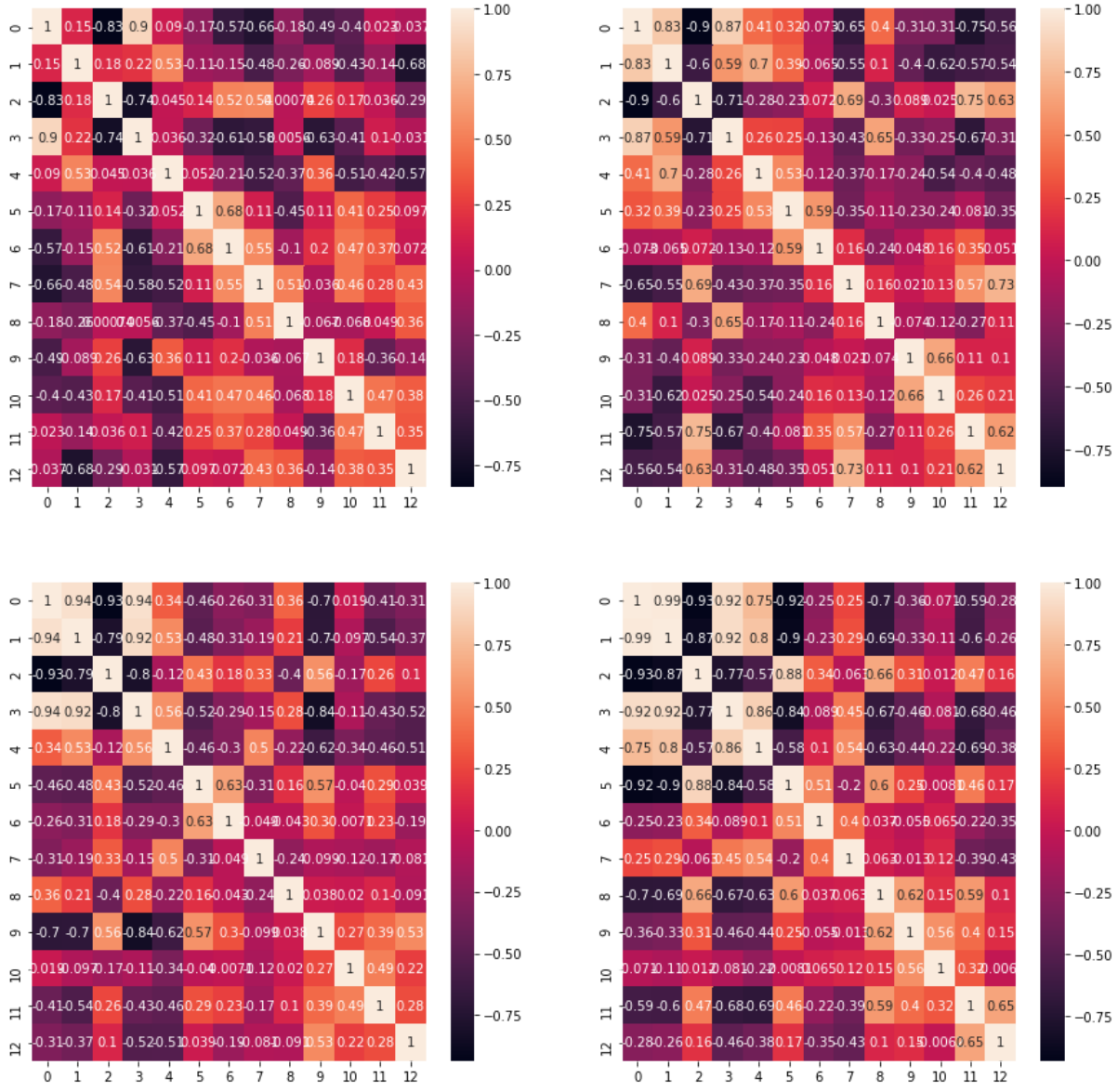








και των MFCCs :



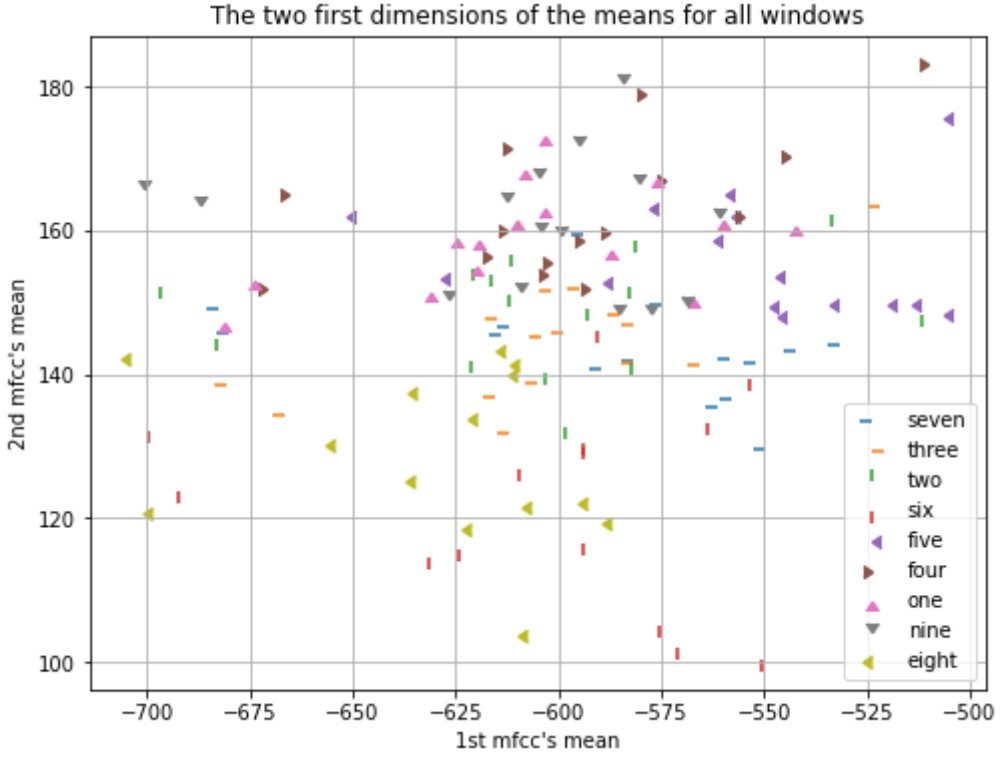
Παρατηρούμε, λοιπόν, ότι τα mfccs είναι χαμηλά συσχετίσιμα μεταξύ τους, ενώ τα mfscs έχουν σαφώς υψηλότερους βαθμούς συσχέτισης. Σε αυτό συμβάλλει ο μετασχηματισμός DCT, ο οποίος αποσυσχετίζει τους συντελεστές. Συνεπώς, είναι λογικό να προτιμήσουμε τα mfccs ως features για το πρόβλημα μας.

ΒΗΜΑ 5

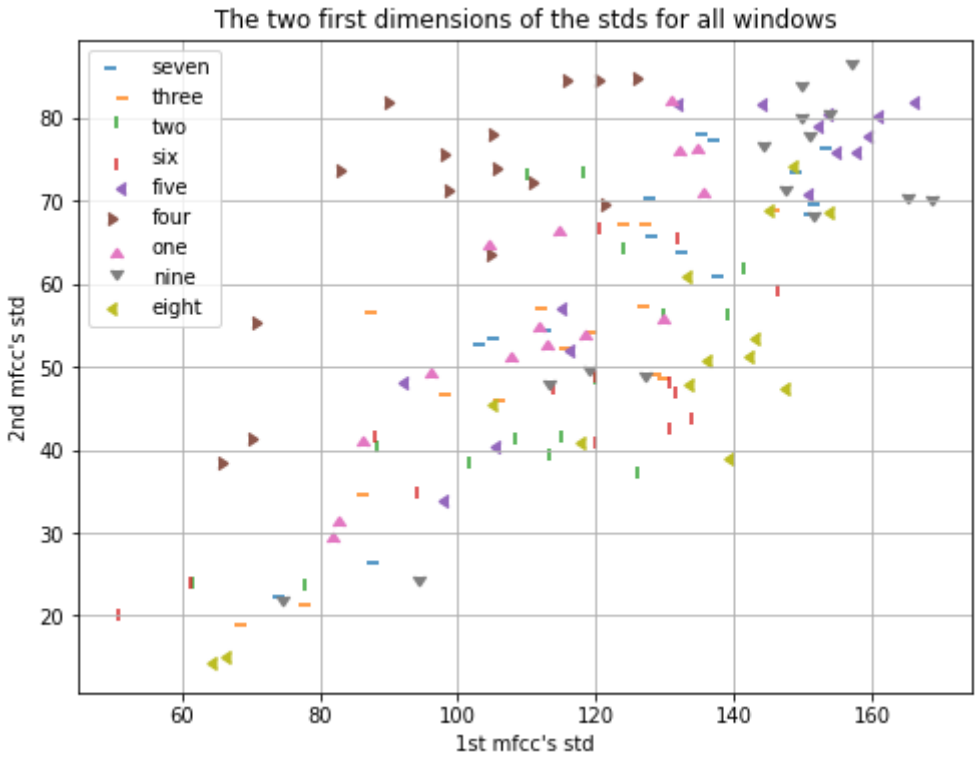
Ύστερα από τις παραπάνω παρατηρήσεις, θέλουμε να δημιουργήσουμε ένα μοναδικό διάνυσμα χαρακτηριστικών, μέσω του οποίου θα μπορέσουμε να καταλήξουμε σε συμπεράσματα, κάνοντας κατάλληλες ταξινομήσεις στο dataset μας. Αρχικά, λοιπόν, ενώνουμε σε ένα ενιαίο διάνυσμα τα mfccs, τα deltas και τα delta-deltas για κάθε παράθυρο για κάθε αρχείο ήχου. Όπως αναφέραμε και πριν, κάθε αρχείο ήχου έχει διαφορετικό αριθμό παραθύρων. Για την δημιουργία μοναδικού διανύσματος για το κάθε αρχείο, παίρνουμε τις μέσες τιμές του κάθε χαρακτηριστικού για όλα τα παράθυρα και σε διαφορετικό διάνυσμα παίρνουμε την

τυπική απόκλιση. Έτσι, εν τέλει, δημιουργούμε 3 πίνακες : έναν με τις μέσες τιμές διαστάσεων 1x39 (13 mfccs, 13 deltas, 13 delta-deltas), έναν με τις τυπικές αποκλίσεις 1x39 και τέλος έναν ενιαίο με τις μέσες τιμές και τις τυπικές αποκλίσεις διάστασης 1x78.

Στη συνέχεια, αναπαριστούμε σε ένα scatter plot τις πρώτες δύο διαστάσεις των μέσων τιμών για κάθε αρχείο ήχου, σημειώνοντας με διαφορετικό χρώμα το κάθε ψηφίο :



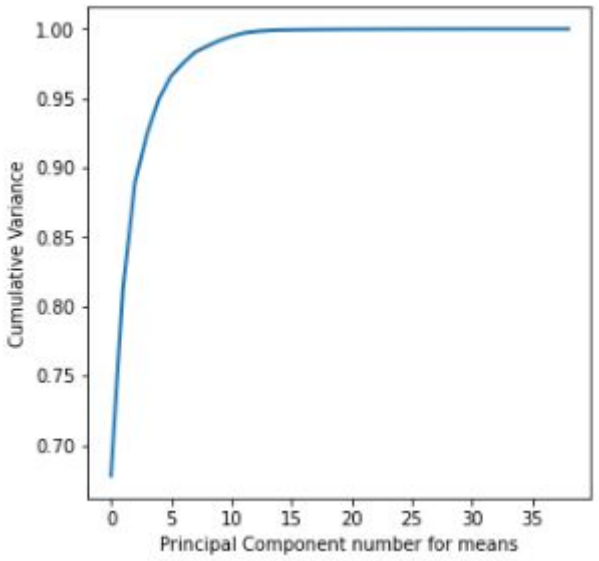
Το ίδιο και για τις τυπικές αποκλίσεις :

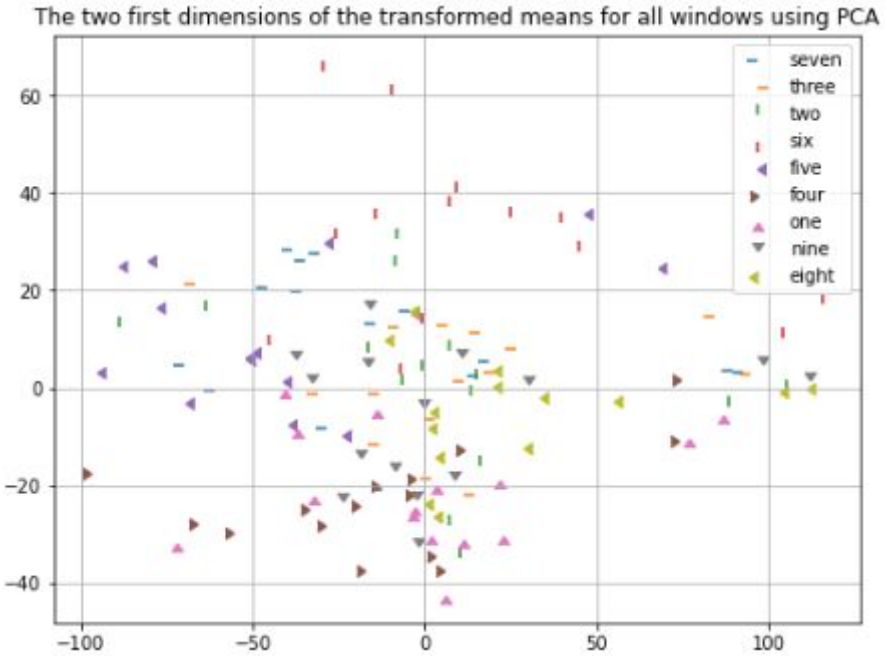


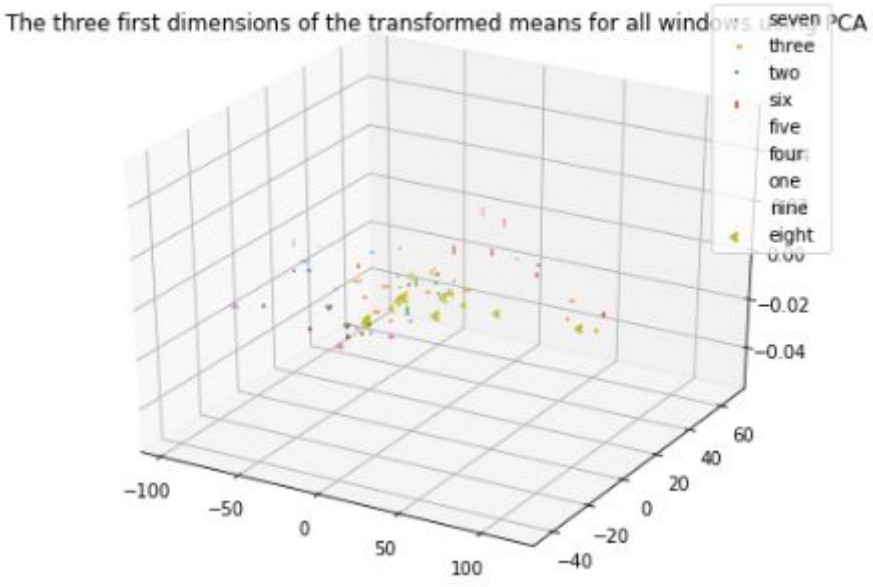
Παρατηρούμε ότι, ως προς τις πρώτες δύο διαστάσεις τουλάχιστον, τα διανύσματα μας είναι αρκετά χαοτικά. Αυτό συμβαίνει αφενός διότι έχουμε διαλέξει πολύ μικρό αριθμό διαστάσεων κατά την αναπαράσταση στο επίπεδο και αφετέρου διότι παίρνοντας μέσες τιμές και τυπικές αποκλίσεις για όλα τα παράθυρα σε κάθε εκφώνηση δεν λαμβάνουμε καθόλου υπόψη μας την χρονική συσχέτιση των δεδομένων μας.

ΒΗΜΑ 6

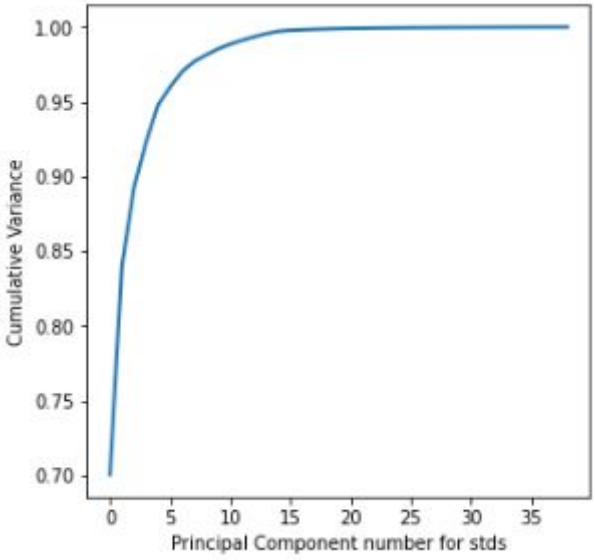
Προκειμένου να έχουμε μία καλύτερη απεικόνιση των διανυσμάτων μας στις δύο και τρεις διαστάσεις εφαρμόζουμε έναν μετασχηματισμό PCA στα δεδομένα μας, τις μέσες τιμές και τυπικές αποκλίσεις για όλα τα παράθυρα. Για να έχουμε μία εικόνα του πόσο επιτυχημένη είναι αυτή η μείωση διαστατικότητας εμφανίζουμε σε κάθε περίπτωση και ένα διάγραμμα που αναπαριστά την σχέση μεταξύ της διάστασης της εξόδου του μετασχηματισμού και του ποσοστού της πληροφορίας (διασποράς) που διατηρούμε από το dataset στον αρχικό χώρο.
Άρα έχουμε τα εξής διαγράμματα αρχικά χρησιμοποιώντας ως χαρακτηριστικά τις μέσες τιμές:

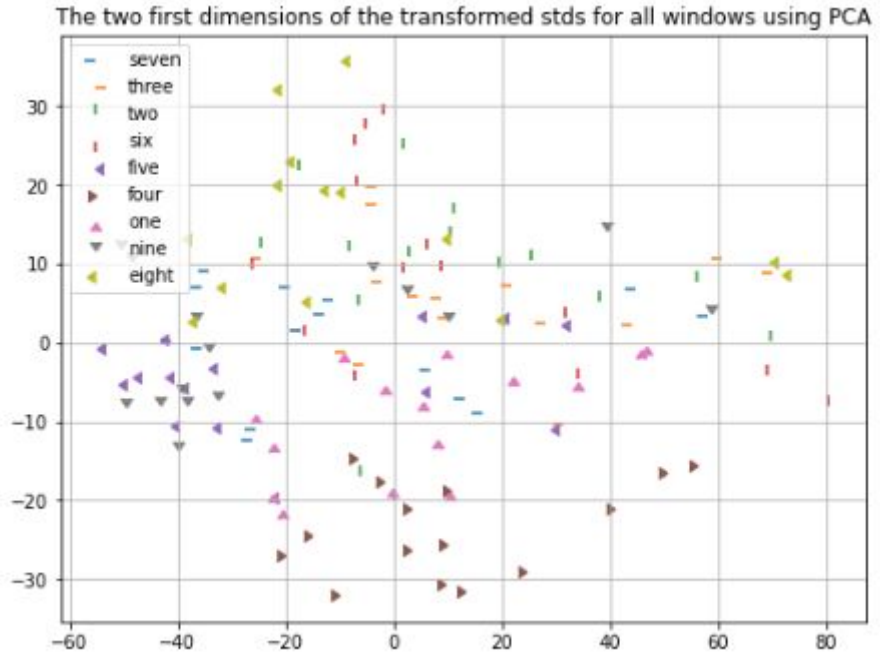

1.00
0.95
0.90
0.85
0.80
0.75
0.70
Cumulative Variance
0
5
10
15
20
25
30
35
Principal Component number for means

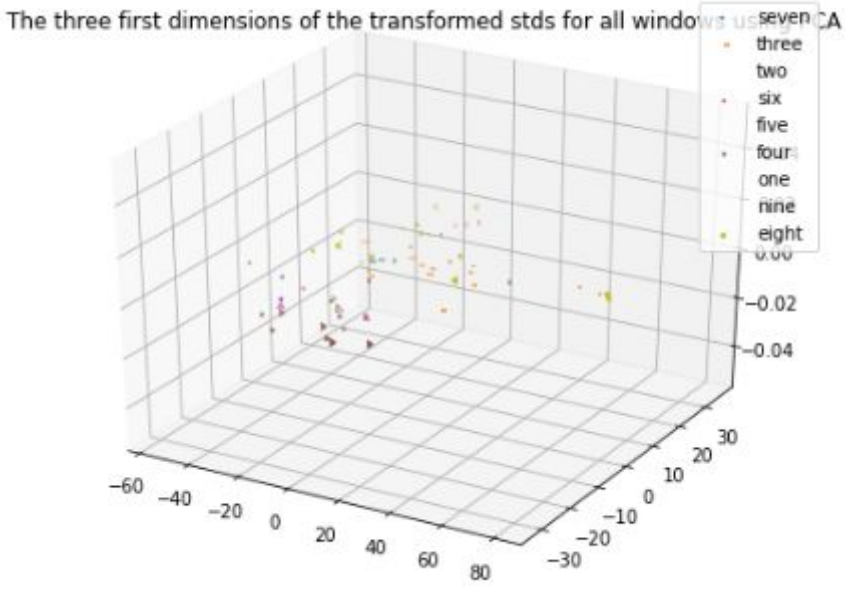

The two first dimensions of the transformed means for all windows using PCA
seven
three
two
six
five
four
one
nine
eight
60
40
20
0
−20
−40
−100
−50
0
50
100



Και για τα διανύσματα των τυπικών αποκλίσεων:

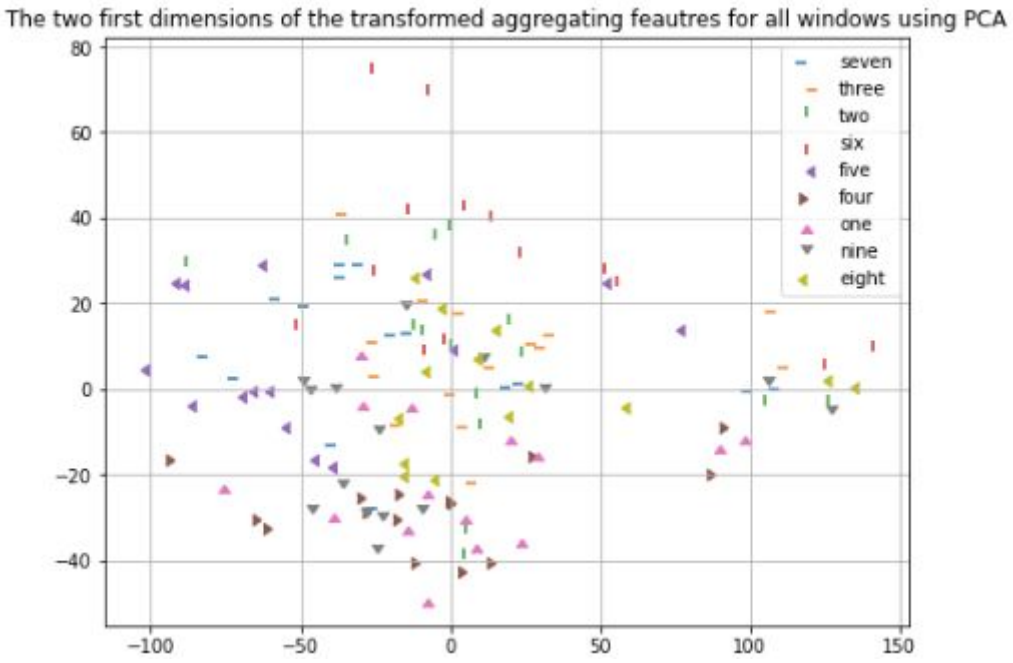


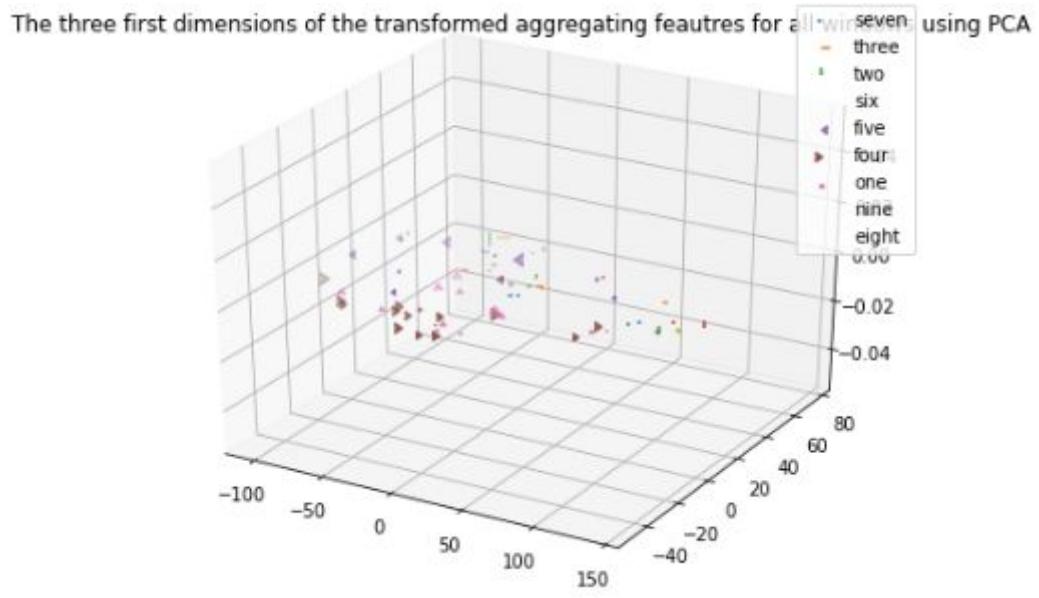




Παρατηρούμε ότι ακόμα και μετά την εφαρμογή της τεχνικής PCA τα δεδομένα συνεχίζουν να μην έχουν κάποια εμφανή διάταξη στον δισδιάστατο ή τρισδιάστατο χώρο που να είναι συνεπής με την κατηγορία στην οποία ανήκουν. Αυτό φαίνεται και από τις καμπύλες των ποσοστών διασποράς για τα διάφορα πλήθη κύριων συνιστωσών. Για πλήθος συνιστωσών 2 και 3 για τις μέσες τιμές έχουμε 81% και 88.9% αντίστοιχα, ενώ για τις τυπικές αποκλίσεις έχουμε 83.9% και 89.2% αντίστοιχα. Άρα, συμπεραίνουμε ότι η μείωση διαστάσεων είναι μη επιτυχημένη.

Τέλος, για να εξαντλήσουμε κάθε περιθώριο με αυτή την προσέγγιση αποτυπώσαμε σε 2 και 3 διαστάσεις τα συνδυασμένα διανύσματα μέσων τιμών και τυπικών αποκλίσεων αφού εφαρμόσαμε PCA.





Και αυτή η απόπειρα φαίνεται αποτυχημένη.

ΒΗΜΑ 7

Σε αυτό το βήμα θα ασχοληθούμε με μεθόδους ταξινόμησης που αξιολογήθηκαν στην προηγούμενη εργαστηριακή άσκηση, προκειμένου να βγάλουμε συμπεράσματα για την δυνατότητα ή όχι ταξινόμησης των ψηφίων από τα ηχητικά δεδομένα μας. Πριν προχωρήσουμε στις διάφορες τεχνικές, αρχικά μετατρέπουμε τα ψηφία από string της μορφής “one”, “two”, etc σε int της μορφής 1, 2 ,κλπ.

Στη συνέχεια παίρνουμε τον πίνακα των κοινών χαρακτηριστικών μέσης τιμής και τυπικής απόκλισης που δημιουργήσαμε στο βήμα 5 και το χωρίζουμε σε train-test dataset με αναλογία 70%-30%. Μετατρέπουμε τα set μας σε πίνακες για διευκόλυνση και οι διαστάσεις που προκύπτουν είναι :

```
(40, 78)
(93, 78)
(40,)
(93,)
```

Για το X_train, X_test, y_train, y_test αντίστοιχα. Κανονικοποιούμε τα δεδομένα μας (και τα train και τα test) μέσω της *StandardScaler* και αρχικά ταξινομούμε με βάση τον Bayesian ταξινομητή που φτιάξαμε στην προηγούμενη άσκηση. Το αποτέλεσμα στο test set είναι :

```
The success rate of our NB classifier's predictions is 60.0 %
```

Έπειτα, χρησιμοποιούμε κάποιους ακόμα ταξινομητές.

Naive Bayes :

```
The success rate of the library's NB classifier's predictions is 60.0 %
```

5-Nearest-Neighbors :

```
The success rate of the 5NN classifier's predictions is 57.49999999999999 %
```

SVM με rbf kernel :

```
The success rate of the SVM classifier's predictions is 57.49999999999999 %
```

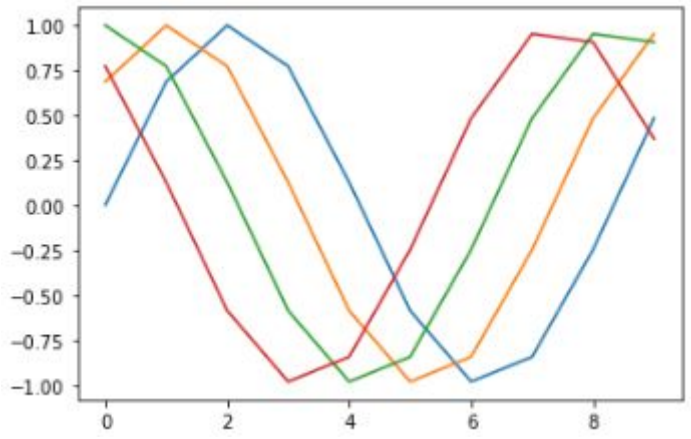
MLP με max_iter=1000 :

```
The success rate of the MLP classifier's predictions is 67.5 %
```

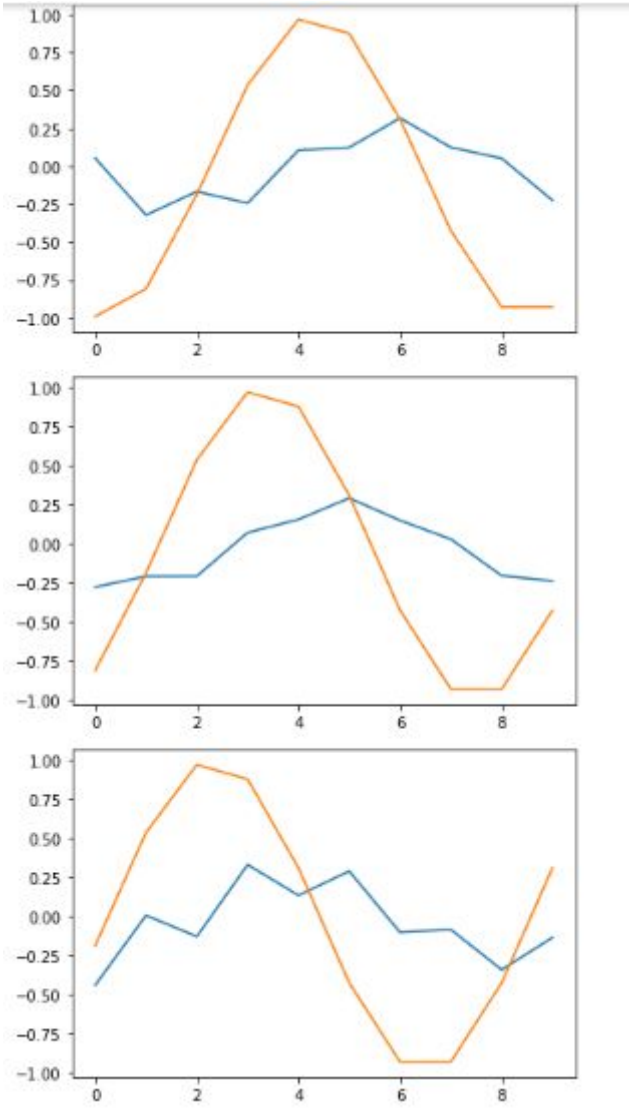
Παρατηρούμε ότι τα ποσοστά που πετυχαίνουμε είναι γενικά χαμηλά σε κάθε πιθανό ταξινομητή. Έτσι, καταλήγουμε στο συμπέρασμα ότι είτε η προεπεξεργασία των δεδομένων και η εξαγωγή των συγκεκριμένων features δεν ήταν επαρκή για μια ικανοποιητική ταξινόμηση ή οι ταξινομητές αυτοί δεν μπορούν να μας δώσουν ικανά αποτελέσματα, καθώς δεν λαμβάνεται υπόψη κάποια χρονική συσχέτιση που υπάρχει στα ηχητικά αρχεία. Καλύτερα αποτελέσματα μας δίνει ο MLP ταξινομητής, καθώς πρόκειται για ένα βαθύ νευρωνικό δίκτυο, το οποίο μπορεί να μας δώσει καλύτερα αποτελέσματα σε παρόμοια προβλήματα αναγνώρισης φωνής. Το dataset μας επίσης είναι αρκετά μικρό για να μπορέσουμε να απαιτήσουμε καλύτερα αποτελέσματα από τους ταξινομητές μας.

ΒΗΜΑ 8

Σε αυτό το βήμα της άσκησης σκοπός είναι η εξοικείωση με τις βιβλιοθήκες του pytorch για αναδρομικά νευρωνικά δίκτυα. Για αυτό, δημιουργούμε 100 ακολουθίες μήκους 10 δειγμάτων ημιτόνου με πλάτος 1, συχνότητα 40 και βήμα δειγματοληψίας 0.003 και τα αντίστοιχα δείγματα συνημιτόνων.
Μία αναπαράσταση των 4 πρώτων ακολουθιών ημιτόνου:

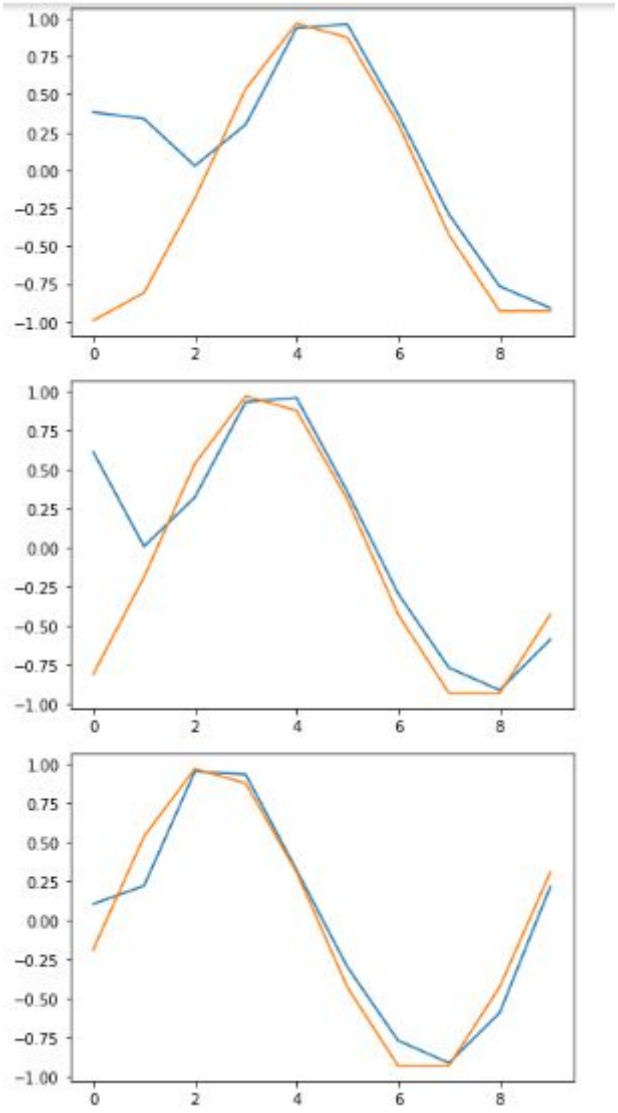


Ο στόχος είναι δεδομένης μιας ακολουθίας ημιτόνου να μπορεί το νευρωνικό μας δίκτυο να παράξει την αντίστοιχη ακολουθία συνημιτόνου. Χωρίζουμε λοιπόν το σύνολο δεδομένων μας σε training (80) και testing (20). Αρχικά, εκπαιδεύουμε ένα απλό RNN για 2000 εποχές με συνάρτηση σφάλματος Μέσου τετραγωνικού λάθους. Τα αποτελέσματα στο testing set (παρουσιάζουμε 3 εδώ) δεν είναι ιδιαίτερα ικανοποιητικά. Φαίνεται μία ποιοτική ομοιότητα με την ακολουθία συνημιτόνου όμως δεν είναι αρκετά επιτυχημένη. Αυτό ίσως οφείλεται στο ότι το αναδρομικό μας δίκτυο δεν μπορεί να μεταβιβάσει σε αρκετά μελλοντικούς νευρώνες την πληροφορία για πιο παλιά δείγματα της ακολουθίας.

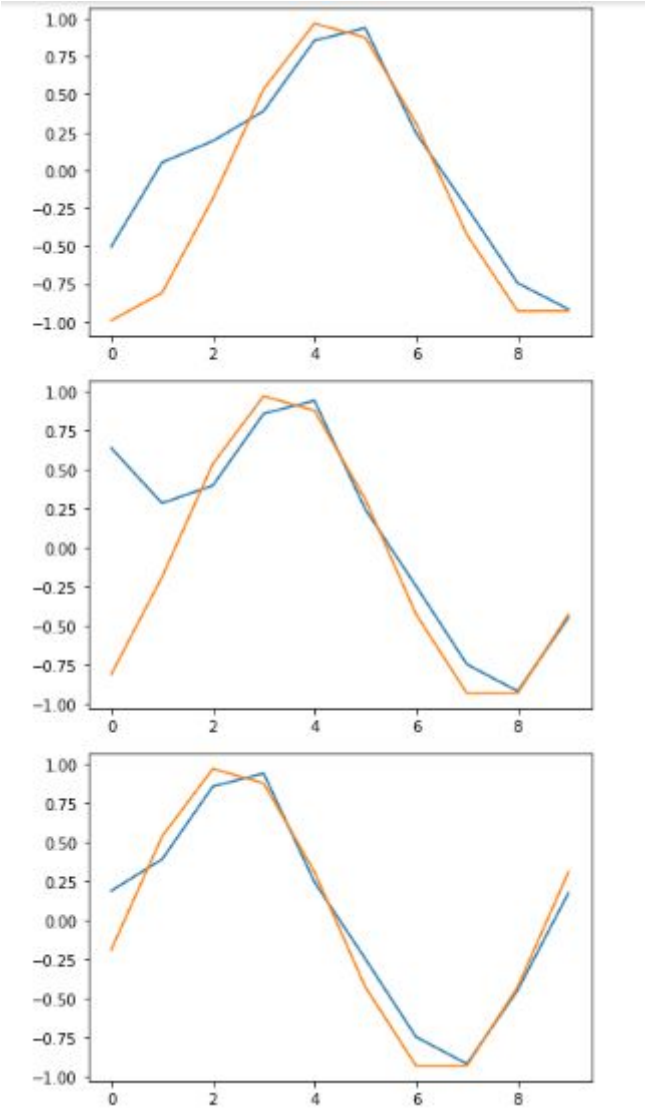


Επομένως, επαναλάβαμε την παραπάνω διαδικασία για τις υλοποιήσεις των LSTM και GRU του pytorch. Τα αποτελέσματα είναι σαφώς καλύτερα και πολύ πιο κοντά στο επιθυμητό. Οι λόγοι για τους οποίους αυτά τα δίκτυα να είναι πιο επιτυχημένα και άρα πιο διαδεδομένα είναι αρχικά το ότι έχουν μεγαλύτερη δυνατότητα να μεταφέρουν πληροφορία για πιο παλιά σημεία της ακολουθίας εισόδου σε μεταγενέστερους νευρώνες. Ακόμα, λόγω της αρχιτεκτονικής τους αυτά τα μοντέλα αντιμετωπίζουν πολύ καλύτερα το πρόβλημα του vanishing gradient, και άρα προσαρμόζονται καλύτερα στα δεδομένα.

Αποτελέσματα LSTM:



Αποτελέσματα GRU:



ΒΗΜΑ 9

Στα παρακάτω βήματα θα δουλέψουμε με ένα διαφορετικό dataset, αρκετά μεγαλύτερο από αυτό που είχαμε μέχρι τώρα, προκειμένου να μπορέσουμε να καταλήξουμε και σε πιο εύλογα συμπεράσματα. Το dataset μας περιέχει 3000 αρχεία ήχου ψηφίων από 0 έως 9. Χρησιμοποιώντας την έτοιμη συνάρτηση parser που μας παρέχεται, διαβάζουμε τα δεδομένα και τα χωρίζουμε σε κατάλληλα σύνολα training, testing και validation. Κατά τον διαχωρισμό του αρχικού training set μας σε training και validation, θέλουμε να διατηρηθεί ίδιος ο αριθμός των διαφορετικών ψηφίων σε κάθε set. Για τον λόγο αυτό χρησιμοποιούμε stratified split. Εν τέλει, τα 3000 αρχεία ήχου χωρίζονται ως :

Training set : 2160

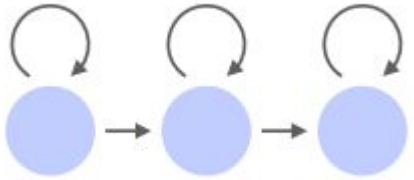
Testing set : 300

Validation set : 540

Τα αρχεία περιέχουν ως features τα πρώτα 6 mfccs του κάθε αρχείου ήχου.

ΒΗΜΑ 10

Σκοπός του συγκεκριμένου βήματος είναι η αναγνώριση ψηφίων με βάση ένα μοντέλο GMM-HMM. Θα υλοποιήσουμε ένα left-to-right model, που είναι της μορφής:



Θεωρούμε την πρώτη κατάσταση του μοντέλου μας ως την πιο πιθανή κατάσταση αρχής και έτσι για τις αρχικές πιθανότητες των καταστάσεων έχουμε :

$$\pi_i = \{0\ i \neq 1\ 1\ i = 1$$

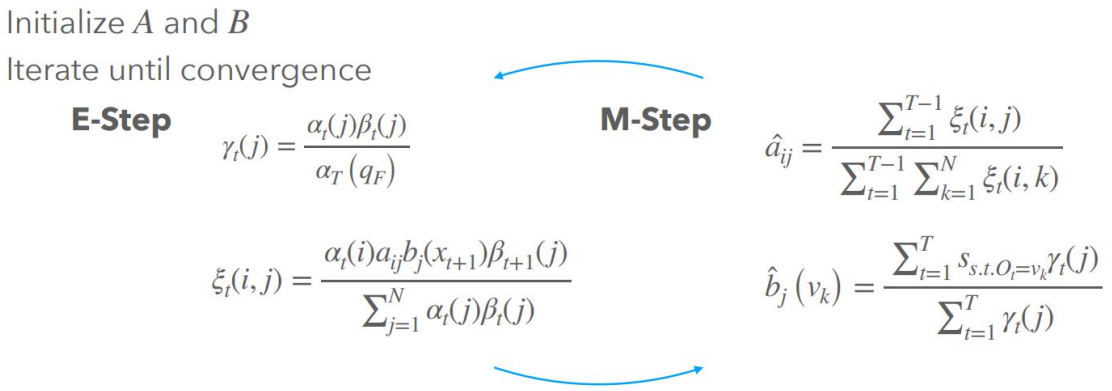
Ως μοναδική πιθανή τελική κατάσταση θεωρούμε μόνο την τελευταία κατάσταση του μοντέλου μας. Οπότε, κατασκευάζουμε ένα διάνυσμα διάστασης ίσης με τον αριθμό των καταστάσεων μας, το οποίο θα αποτελείται από μηδενικά εκτός της τελευταίας θέσης.

Στη συνέχεια, κατασκευάζουμε τον πίνακα μετάβασης. Παρατηρούμε ότι κάθε κατάσταση έχει 2 πιθανές μεταβάσεις : 1 στον εαυτό της και 1 στην δεξιότερη της. Έτσι, αναθέτουμε σε κάθε πιθανή μετάβαση πιθανότητα 0.5. Εξαίρεση αποτελεί η τελευταία κατάσταση που μπορεί να μεταβεί μόνο στον εαυτό της και έτσι θα έχει πιθανότητα 1.

Στην αναγνώριση φωνής, είναι ιδιαίτερα διαδεδομένη η χρήση GMM. Αυτό οφείλεται στο γεγονός ότι χρειαζόμαστε μια ευέλικτη δομή για την μοντελοποίηση της πιθανότητας των παρατηρήσεων που λαμβάνουμε από την επεξεργασία ενός πλαισίου φωνής και αναλογούν σε συγκεκριμένες καταστάσεις, καθώς είναι επιτρεπτές οι συνεχείς μεταβολές αυτών. Έτσι, για τις emission κατανομές θα χρησιμοποιήσουμε GMM από Multivariate Gaussian κατανομές.

Για την υλοποίηση των παραπάνω χρησιμοποιούμε την βιβλιοθήκη pomegranate και τον έτοιμο σκελετό κώδικα που μας παρέχεται, μεταβάλλοντας κατάλληλα αυτά που εξηγήσαμε παραπάνω.

Για την εκπαίδευση του μοντέλου μας θα χρησιμοποιήσουμε τον default αλγόριθμο της βιβλιοθήκης μας, τον Baum-Welch, καθώς φαίνεται να είναι και ο πιο κατάλληλος. Σχηματικά, ο επαναληπτικός αλγόριθμος φαίνεται παρακάτω :



Τέλος, κατασκευάζουμε μία λίστα που θα περιλαμβάνει στο αντίστοιχο index της όλα τα training δεδομένα του συγκεκριμένου ψηφίου-index. Αυτό θα μας βοηθήσει σημαντικά στην κατασκευή των 10 μοντέλων.

ΒΗΜΑ 11

Δημιουργούμε μια κενή λίστα models, που θα περιλαμβάνει τα 10 μοντέλα, ένα για το κάθε ψηφίο μας. Χρησιμοποιώντας τον πίνακα του βήματος 10, καλούμε την συνάρτηση train_hmm για το κάθε διαφορετικό ψηφίο και έτσι δημιουργείται ένα μοντέλο για το κάθε ένα από αυτά και αποθηκεύεται στην λίστα. Χρησιμοποιούμε αλγόριθμο Expectation - Maximization, όπως μας ζητείται, καθώς ταυτίζεται με τον Baum-Welch. Ο αλγόριθμος αυτός εφαρμόζεται είτε για συγκεκριμένο αριθμό max_iter επαναλήψεων που εμείς θέτουμε χειροκίνητα ή μέχρι να υπάρξει σύγκλιση. Η σύγκλιση ελέγχεται μέσω της μεταβολής του αλγορίθμου της πιθανοφάνειας. Τρέχουμε τον αλγόριθμο για max_iter = 5, states = 3 και gaussians = 5 (μέγιστος αριθμός επαναλήψεων, αριθμός καταστάσεων και αριθμός mixtures από gaussians στο GMM αντίστοιχα).

ΒΗΜΑ 12

Έχοντας εκπαιδεύσει τα 10 μοντέλα μας στο προηγούμενο βήμα, θα προχωρήσουμε στην διαδικασία ταξινόμησης με τη βοήθεια του λογαρίθμου της πιθανοφάνειας. Για κάθε εκφώνηση του testing set, υπολογίζουμε τον λογάριθμο της πιθανοφάνειας για το κάθε μοντέλο από τα 10 που έχουμε, τρέχοντας τον αλγόριθμο Viterbi και επιστρέφοντας τους λογάριθμους. Το μοντέλο που θα μας δώσει την μέγιστη τιμή είναι και εκείνο που θα μας δώσει το αποτέλεσμα της εκάστοτε εκφώνησης. Έτσι,

κατασκευάζουμε μια συνάρτηση evaluate_hmms, όπου περνάμε σαν ορίσματα το test set μαζί με τα labels και το σύνολο των μοντέλων μας. Αποθηκεύουμε σε κατάλληλο πίνακα το prediction που προκύπτει μέσω της παραπάνω διαδικασίας για κάθε εκφώνηση, κατασκευάζοντας παράλληλα μια προσομοίωση του confusion matrix. Τελικά, επιστρέφουμε το confusion matrix και το accuracy.

Τρέχοντας την συνάρτηση για το validation set, προκύπτουν :

```
[[54.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. 54.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. 53.  0.  0.  0.  1.  0.  0.  0.]
 [ 0.  0.  1. 52.  0.  0.  0.  0.  1.  0.]
 [ 0.  1.  0.  0. 53.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0. 53.  0.  0.  0.  0.]
 [ 0.  0.  0.  3.  0.  0. 49.  0.  2.  0.]
 [ 0.  1.  0.  1.  0.  0.  0. 52.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  1.  0. 53.  0.]
 [ 0.  1.  0.  0.  0.  0.  0.  0.  0. 53.]]
 0.9740740740740741
```

Στη συνέχεια, μέσω του validation set, επιδιώκουμε να βρούμε τον βέλτιστο συνδυασμό των υπερπαραμέτρων που περνάμε σαν ορίσματα για την εκπαίδευση του hmm (ο αριθμός των states, των gaussians και του max iterations). Συγκεκριμένα εκπαιδεύουμε τα μοντέλα μας με τις παρακάτω δυνατές τιμές των υπερπαραμέτρων :

states = 1, 2, 3, 4

gaussians (mixtures) = 1, 2, 3, 4, 5

max_iterations = 5, 11, 15

και σε κάθε περίπτωση κάνουμε αποτίμηση στο validation set. Διατηρούμε σε κατάλληλες μεταβλητές τα βέλτιστα αποτελέσματα κάθε φορά και εν τέλει προκύπτουν :

```
the best parameters are :
states =  4
mixtures =  4
max iterations =  15
accuracy =  0.9796296296296296
```

Εν τέλει, χρησιμοποιούμε τις ανωτέρω τιμές για να εκπαιδεύσουμε τα 10 μοντέλα μας και κάνουμε μια τελική αποτίμηση στο testing set, το οποίο μας δίνει :

```
(array([[30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0., 30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0., 30.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  1.,  0.,  0., 29.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0., 30.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  2.,  0.,  0., 25.,  2.,  1.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  0., 30.,  0.,  0.],
        [ 0.,  0.,  0.,  1.,  0.,  0.,  1.,  0., 28.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0., 30.]]),
 0.9733333333333334)
```

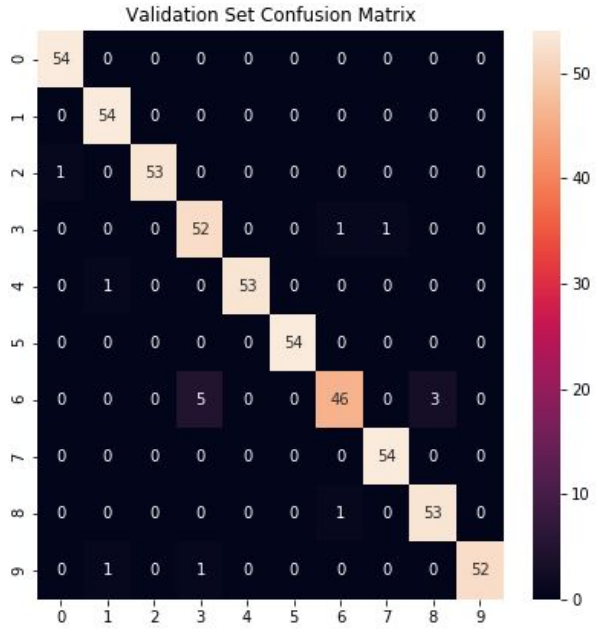
Η διαδικασία που ακολουθούμε αποτελείται από την εκπαίδευση των μοντέλων μας μέσω του training set, την ρύθμιση των υπερπαραμέτρων μας μέσω του validation set και τέλος την αξιολόγηση μέσω του testing set. Είναι σημαντικό να τηρήσουμε αυτή την διαδικασία, καθώς αφενός το validation set έχει προκύψει μέσω stratified split και άρα αποτελεί ένα καλό δείγμα για την εύρεση βέλτιστης εκπαίδευσης (έχοντας ίσα ποσοστά από κάθε ψηφίο) και αφετέρου το validation set πλέον αποτελεί κατά κάποιον τρόπο μέρος της εκπαίδευσης και άρα δεν μπορούμε να το χρησιμοποιήσουμε και στην τελική αξιολόγηση, διότι σε αυτή την περίπτωση θα είχαμε biased αποτελέσματα. Αξιολογώντας το σύστημα μοντέλων μας πάνω στο testing set, πετυχαίνουμε καλύτερη αμεροληψία, αφού το testing set δεν χρησιμοποιείται ούτε κατά την εκπαίδευση, αλλά ούτε και κατά την εύρεση βέλτιστων υπερπαραμέτρων (το οποίο μπορεί να θεωρηθεί μέρος της εκπαίδευσης). Σε οποιαδήποτε αντίθετη περίπτωση, όπου θα χρησιμοποιούσαμε κάποιο set 2 φορές, θα προσθέταμε σημαντικό bias στο μοντέλο μας και θα έχανε την αξιοπιστία του.

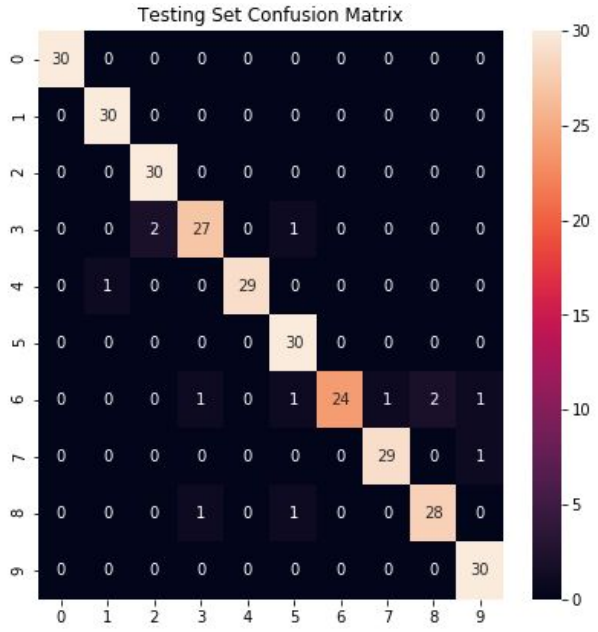
ΒΗΜΑ 13

Για να έχουμε μία καλύτερη οπτικοποίηση και συγκέντρωση των παραπάνω αποτελεσμάτων τα παρουσιάζουμε κι εδώ. Το Accuracy των μοντέλων μας πάνω στο validation και στο training set αντίστοιχα είναι:

```
Accuracy on Validation Set: 0.9722222222222222
Accuracy on Testing Set: 0.9566666666666667
```

Και οι πίνακες confusion σε μορφή heatmap:





ΒΗΜΑ 14

Σε αυτό το βήμα θα προσπαθήσουμε να κάνουμε ταξινόμηση με χρήση ενός διαφορετικού μοντέλου. Θα χρησιμοποιήσουμε ένα LSTM. Προκειμένου να έχουν όλες οι ακολουθίες που θα δίνουμε ως είσοδο στο LSTM υλοποιούμε μία κλάση FrameLevelDataset, η οποία θα είναι υπεύθυνη για το padding των δεδομένων ενός batch και για την διαχείριση των διανυσμάτων χαρακτηριστικών και των ετικετών τους. Το padding γίνεται με προσθήκη μηδενικών στο τέλος της κάθε ακολουθίας ώστε όλες να έχουν το ίδιο μήκος, ίσο με το μήκος της μέγιστης ακολουθίας.

Έπειτα υλοποιούμε το LSTM μας ως ένα torch.nn.Module με βάση την κλάση nn.LSTM και σαν δεύτερο επίπεδο ένα γραμμικό στρώμα με χρήση της έτοιμης υλοποίησης torch.nn.Linear. Με κατάλληλες μεθόδους φροντίζουμε ώστε το αποτέλεσμα που θα δίνει το νευρωνικό να είναι αυτό που παράγεται στον νευρώνα του LSTM στον οποίο δίνεται ως είσοδος το τελευταίο δείγμα κάθε ακολουθίας. Για αυτό τον σκοπό είναι απαραίτητο να παρέχονται και τα πραγματικά μήκη των ακολουθιών, πριν το padding.

Ακόμα, φροντίζουμε ώστε να μετατρέπουμε τις ετικέτες των διανυσμάτων σε μορφή one hot encoding για να απαλείψουμε οποιαδήποτε σχέση διάταξης μεταξύ τους αλλά και γιατί σε αυτή την μορφή θα είναι τα διανύσματα που θα προβλέπει το νευρωνικό μας. Για την εκπαίδευση χρησιμοποιούμε συνάρτηση λάθους Μέσου Τετραγωνικού Λάθους και 30 εποχές. Ακόμα, παρουσιάζουμε το training dataset στο δίκτυο μας σε μικρές παρτίδες (batches) με τυχαία διάταξη.

Εκπαιδεύουμε, λοιπόν, το μοντέλο μας τυπώνοντας αρχικά για κάθε εποχή το σφάλμα. Το αποτέλεσμα είναι το παρακάτω:

```
epoch :  0 ,training loss =  0.07282869300957936
epoch :  1 ,training loss =  0.04685428933198772
epoch :  2 ,training loss =  0.02985253490841211
epoch :  3 ,training loss =  0.020128031289065953
epoch :  4 ,training loss =  0.014045595281771314
epoch :  5 ,training loss =  0.008610769623278905
epoch :  6 ,training loss =  0.00829797729835915
epoch :  7 ,training loss =  0.008380273169379181
epoch :  8 ,training loss =  0.006915336547753037
epoch :  9 ,training loss =  0.004448093077751683
epoch :  10 ,training loss =  0.004841557688274379
epoch :  11 ,training loss =  0.003804843353260475
epoch :  12 ,training loss =  0.003776886615095966
epoch :  13 ,training loss =  0.005043905061572346
epoch :  14 ,training loss =  0.004571452052262959
epoch :  15 ,training loss =  0.006729431349589531
epoch :  16 ,training loss =  0.0049536992055572455
epoch :  17 ,training loss =  0.008452610140527362
epoch :  18 ,training loss =  0.003934274426723983
epoch :  19 ,training loss =  0.004342158912305734
epoch :  20 ,training loss =  0.003249184272787186
epoch :  21 ,training loss =  0.0022567376436199993
epoch :  22 ,training loss =  0.0018168503697440306
epoch :  23 ,training loss =  0.0016911058411273215
epoch :  24 ,training loss =  0.002513808136156051
epoch :  25 ,training loss =  0.003345749684984782
epoch :  26 ,training loss =  0.004208253998893188
epoch :  27 ,training loss =  0.002116342246316985
epoch :  28 ,training loss =  0.003327249238309242
epoch :  29 ,training loss =  0.00209306020579829
```

Παρατηρούμε ότι γενικά το σφάλμα μειώνεται σημαντικά και αρκετά γρήγορα. Για να έχουμε όμως μία καλύτερη εικόνα του τρόπου που εκπαιδεύεται το μοντέλο μας σε κάθε εποχή τυπώνουμε και το Accuracy που πετυχαίνει το μέχρι εκείνη τη στιγμή εκπαιδευμένο μοντέλο στο validation set. Τα αποτελέσματα παρουσιάζονται παρακάτω :

```
epoch :  0 ,training loss =  0.06893675357325753
Accuracy in validation set =  67.96296296296296 %
epoch :  0 ,validation loss =  0.003466065274551511
epoch :  1 ,training loss =  0.0432912453508644
Accuracy in validation set =  79.44444444444444 %
epoch :  1 ,validation loss =  0.0020510072354227304
epoch :  2 ,training loss =  0.027206948054815407
Accuracy in validation set =  89.25925925925927 %
epoch :  2 ,validation loss =  0.0012718779034912586
epoch :  3 ,training loss =  0.016757222239984507
Accuracy in validation set =  92.77777777777779 %
epoch :  3 ,validation loss =  0.0008859737426973879
epoch :  4 ,training loss =  0.011648230730736656
Accuracy in validation set =  93.7037037037037 %
epoch :  4 ,validation loss =  0.0008372677839361131
epoch :  5 ,training loss =  0.01001917618785554
Accuracy in validation set =  95.0 %
epoch :  5 ,validation loss =  0.0007893528090789914
epoch :  6 ,training loss =  0.009121269164884935
Accuracy in validation set =  91.2962962962963 %
epoch :  6 ,validation loss =  0.0010202716803178191
epoch :  7 ,training loss =  0.010881359383130252
Accuracy in validation set =  93.7037037037037 %
epoch :  7 ,validation loss =  0.0008852249593473971
epoch :  8 ,training loss =  0.007571980923608835
Accuracy in validation set =  95.55555555555556 %
epoch :  8 ,validation loss =  0.0006002190639264882
epoch :  9 ,training loss =  0.004329011858955248
Accuracy in validation set =  97.03703703703704 %
epoch :  9 ,validation loss =  0.0004711854853667319
epoch :  10 ,training loss =  0.005951574444075796
Accuracy in validation set =  95.37037037037037 %
epoch :  10 ,validation loss =  0.0006104034255258739
epoch :  11 ,training loss =  0.005593958708680054
Accuracy in validation set =  94.25925925925925 %
epoch :  11 ,validation loss =  0.0005821601953357458
epoch :  12 ,training loss =  0.0048757418295118345
Accuracy in validation set =  96.48148148148148 %
epoch :  12 ,validation loss =  0.0004851123667322099
epoch :  13 ,training loss =  0.004964162289762675
Accuracy in validation set =  95.55555555555556 %
epoch :  13 ,validation loss =  0.0005589139764197171
epoch :  14 ,training loss =  0.0069515703179970825
Accuracy in validation set =  96.11111111111111 %
epoch :  14 ,validation loss =  0.0005869267042726278
epoch :  15 ,training loss =  0.005175890263739918
Accuracy in validation set =  95.18518518518519 %
epoch :  15 ,validation loss =  0.0005560435820370913
```

```
epoch :  16 ,training loss =  0.004661492337889747
Accuracy in validation set =  95.18518518518519 %
epoch :  16 ,validation loss =  0.0006195440073497593
epoch :  17 ,training loss =  0.003244129177217664
Accuracy in validation set =  97.77777777777777 %
epoch :  17 ,validation loss =  0.0002887739392463118
epoch :  18 ,training loss =  0.0027217524805551035
Accuracy in validation set =  96.66666666666667 %
epoch :  18 ,validation loss =  0.000435823603766039
epoch :  19 ,training loss =  0.005775290431016917
Accuracy in validation set =  94.25925925925925 %
epoch :  19 ,validation loss =  0.0008404311374761164
epoch :  20 ,training loss =  0.006098397696432449
Accuracy in validation set =  95.55555555555556 %
epoch :  20 ,validation loss =  0.000531802827026695
epoch :  21 ,training loss =  0.004817398615987666
Accuracy in validation set =  94.25925925925925 %
epoch :  21 ,validation loss =  0.0006594366277568042
epoch :  22 ,training loss =  0.006857061543182206
Accuracy in validation set =  95.74074074074073 %
epoch :  22 ,validation loss =  0.0004818620509468019
epoch :  23 ,training loss =  0.0023625075309745857
Accuracy in validation set =  97.5925925925926 %
epoch :  23 ,validation loss =  0.0003163479268550873
epoch :  24 ,training loss =  0.0020573291946689026
Accuracy in validation set =  96.66666666666667 %
epoch :  24 ,validation loss =  0.0004061594663653523
epoch :  25 ,training loss =  0.0017832654798507635
Accuracy in validation set =  96.66666666666667 %
epoch :  25 ,validation loss =  0.0003549090470187366
epoch :  26 ,training loss =  0.002214055156957275
Accuracy in validation set =  95.18518518518519 %
epoch :  26 ,validation loss =  0.0005254363641142845
epoch :  27 ,training loss =  0.004077976612971901
Accuracy in validation set =  97.5925925925926 %
epoch :  27 ,validation loss =  0.0005218751612119377
epoch :  28 ,training loss =  0.00238787181546856
Accuracy in validation set =  96.29629629629629 %
epoch :  28 ,validation loss =  0.0004644711734727025
epoch :  29 ,training loss =  0.0029362068996319907
Accuracy in validation set =  96.11111111111111 %
epoch :  29 ,validation loss =  0.0004074738535564393
```

Παρατηρούμε, ότι πολύ γρήγορα ήδη από την 9η-10η εποχή το Accuracy στο validation set είναι αρκετά υψηλό περίπου 97%. Μετά από αυτό το σημείο εμφανίζει διάφορα τοπικά ελάχιστα και μέγιστα σε αυτήν την περιοχή τιμών 94%-97%, ενώ το σφάλμα στο training set συνεχίζει να μειώνεται. Με αυτόν τον τρόπο μπορούμε ίσως να αντιληφθούμε το αν το μοντέλο μας πάσχει από overfitting.

Πριν εφαρμόσουμε κάποια τεχνική για να το αντιμετωπίζουμε ελέγχουμε την επίδοση στο testing set.

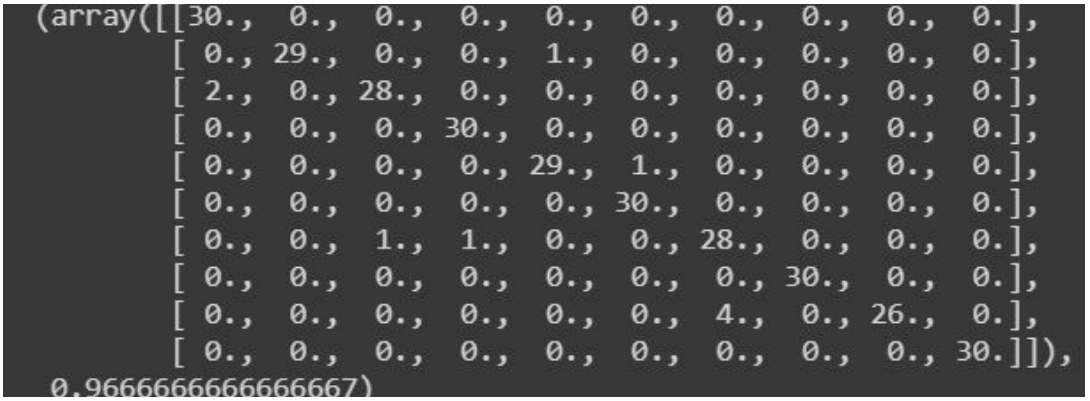
```
(array([[30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0., 30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0., 29.,  0.,  0.,  0.,  0.,  1.,  0.],
        [ 0.,  0.,  0.,  0., 29.,  1.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0., 30.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  1.,  0.,  0., 29.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  0., 30.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  2.,  0., 28.,  0.],
        [ 0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0., 29.]]), 0.98)
```

Το ποσοστό επιτυχίας είναι εντυπωσιακά υψηλό και παρατηρούμε και αυτή την επιτυχία του μοντέλου και στον πίνακα confusion.

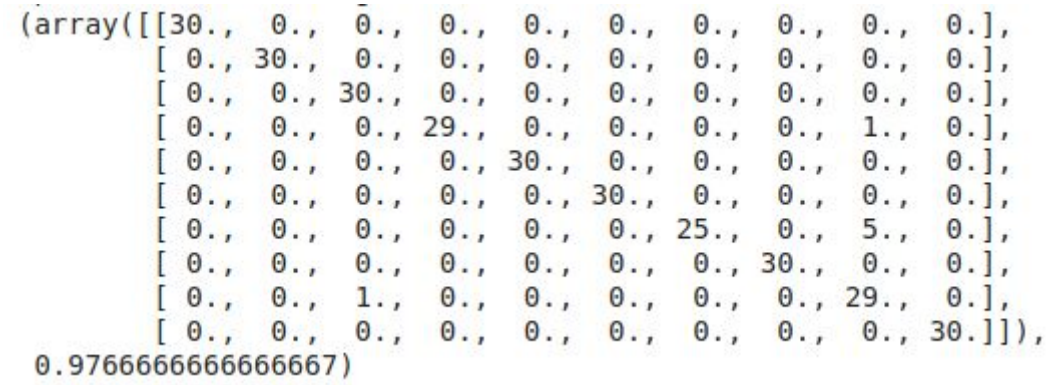
Παρ'ότι δεν φαίνεται να υπάρχει πρόβλημα overfitting, εφαρμόζουμε διάφορες τεχνικές που το περιορίζουν. Πρώτη τεχνική που δοκιμάζουμε είναι το dropout. Σύμφωνα με αυτή ένα μέρος από τις μεταβολές που γίνονται κατά το back propagation δεν εφαρμόζεται στα βάρη του δικτύου με πιθανότητα p ορισμένη από εμάς πριν από την εκπαίδευση του δικτύου. Τα αποτελέσματα της χρήσης αυτής της τεχνικής για p=0.5:

```
(array([[30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 28.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  1.],
        [ 0.,  0., 29.,  1.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0., 30.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0., 29.,  1.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0., 30.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0., 27.,  0.,  2.,  1.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  0., 30.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0., 30.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0., 30.]]),
 0.9766666666666667)
```

Μία άλλη τεχνική που αντιμετωπίζει το overfitting είναι η L2 regularization. Αυτή η μέθοδος προσπαθεί να ελαττώσει την πολυπλοκότητα του μοντέλου βάζοντας έναν παράγοντα κανονικοποίησης στη συνάρτηση λάθους με την οποία γίνεται η εκπαίδευση. Αυτός ο παράγοντας είναι εν προκειμένω ένα πολλαπλάσιο της L2 νόρμας όλων των βαρών του μοντέλου με μία παράμετρο alpha. Τα αποτελέσματα για alpha=0.00001:



Τα αποτελέσματα από τον συνδυασμό των δύο τεχνικών:



Γενικά το μοντέλο μας μάλλον γενικεύει καλά σύμφωνα με την επίδοση του πάνω στο training set και επομένως αυτές οι τεχνικές δεν βελτιώνουν την ικανότητά του.

Δύο άλλες προσθήκες που κάναμε στην διαδικασία εκπαίδευσης του μοντέλου μας είναι η εφαρμογή Early Stopping και η τήρηση Checkpoints κάθε 5 εποχές. Η εφαρμογή του Early Stopping είναι πάρα πολύ σημαντική κατά την εκπαίδευση. Περιορίζει το ενδεχόμενο overfitting και μειώνει ενδεχομένως τον χρόνο εκπαίδευσης του μοντέλου. Η παραλλαγή που χρησιμοποιούμε διατηρεί ένα παράθυρο μήκους 4 εποχών και σε κάθε εποχή μετράει το Accuracy στο validation set. Αν αυτό είναι χειρότερο από όλες τις τιμές στο παράθυρο, δηλαδή από την επίδοση του μοντέλου όπως έχει εκπαιδευτεί από τις 4 τελευταίες εποχές, με ένα περιθώριο 1% τότε η εκπαίδευση διακόπτεται και επιστρέφουμε ως αποτέλεσμα το μοντέλο με το καλύτερο Accuracy από αυτά του παραθύρου. Τα αποτελέσματα για εκπαίδευση σε 50 εποχές με εκτύπωση της επίδοσης στο validation set.

….

```
Accuracy in validation set =  94.81481481481482 %
epoch :  21 ,validation loss =  0.0006411634967662394
epoch :  22 ,training loss =  0.0034603212756771982
Accuracy in validation set =  94.81481481481482 %
epoch :  22 ,validation loss =  0.0005463103880174458
epoch :  23 ,training loss =  0.003181735539301507
Accuracy in validation set =  95.37037037037037 %
epoch :  23 ,validation loss =  0.0004485778044909239
epoch :  24 ,training loss =  0.0043838606268139695
Accuracy in validation set =  94.44444444444444 %
epoch :  24 ,validation loss =  0.0006392365321516991
================== Saved checkpoint ==================
epoch :  25 ,training loss =  0.006506407159074808
Accuracy in validation set =  95.18518518518519 %
epoch :  25 ,validation loss =  0.0005693535786122084
epoch :  26 ,training loss =  0.003749156004137624
Accuracy in validation set =  95.92592592592592 %
epoch :  26 ,validation loss =  0.00045113288797438145
epoch :  27 ,training loss =  0.0043680845115180895
Accuracy in validation set =  89.81481481481481 %
epoch :  27 ,validation loss =  0.0010214074281975627
---------------------------------------------------------------
Early stopping. Returning model of epoch 25 with accuracy 0.9574074074074074 on evaluation test
(array([[30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 28.,  0.,  0.,  0.,  2.,  0.,  0.,  0.,  0.],
        [ 0.,  0., 30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0., 30.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 5.,  0., 15.,  0., 10.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0., 28.,  1.,  1.,  0.,  0.],
        [ 0.,  0.,  0.,  1.,  0.,  0., 29.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  0., 30.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  1.,  0., 29.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  1.,  0.,  3.,  0., 26.]]), 0.9)
```

Παρατηρούμε ότι όταν η επίδοση του μοντέλου άρχισε να μειώνεται, και μάλιστα πολύ απότομα, η διαδικασία του early stopping μας προστατεύει από την περεταίρω απομάκρυνση του μοντέλου από την πραγματική κατανομή των δεδομένων μας. Ακόμη, βλέπουμε ότι η εκπαίδευση σταμάτησε στις 27 εποχές από τις 50 εξοικονομώντας μας σημαντικό χρόνο.

Μία άλλη δοκιμή που κάναμε ήταν η χρήση ενός bidirectional lstm. Με χρήση διπλά κατευθυνόμενου lstm οι ακολουθίες εισόδου που δίνουμε εμφανίζονται στους νευρώνες μία φορά από την αρχή προς το τέλος (παρελθόν -> μέλλον) και μία φορά από το τέλος προς την αρχή (μέλλον -> παρελθόν). Με αυτόν τον τρόπο είναι δυνατόν να αξιοποιήσουμε την χρονική συσχέτιση των δειγμάτων της ακολουθίας μας και προς τις δύο κατευθύνσεις αφού θα έχουμε πληροφορία που προέρχεται από το μέλλον. Έτσι, θεωρητικά τουλάχιστον, το δίκτυο μας θα έχει καλύτερη κατανόηση του περικειμένου του κάθε δείγματος της ακολουθίας.
Τα σχετικά αποτελέσματα:

```
epoch :  0 ,training loss =  0.047536926955651884
epoch :  1 ,training loss =  0.026092396826664015
epoch :  2 ,training loss =  0.015064025353362312
epoch :  3 ,training loss =  0.012328997462876697
epoch :  4 ,training loss =  0.009617073996576355
epoch :  5 ,training loss =  0.006118966589695705
epoch :  6 ,training loss =  0.005800955643905188
epoch :  7 ,training loss =  0.0050716829597393966
epoch :  8 ,training loss =  0.005796455701729699
epoch :  9 ,training loss =  0.004534276065055225
epoch :  10 ,training loss =  0.0032915679856078393
epoch :  11 ,training loss =  0.004855575689823547
epoch :  12 ,training loss =  0.006003348792750222
epoch :  13 ,training loss =  0.004597751799025642
epoch :  14 ,training loss =  0.0030949861749506266
epoch :  15 ,training loss =  0.0029630874684984003
epoch :  16 ,training loss =  0.0030308284565681285
epoch :  17 ,training loss =  0.0018712875881552028
epoch :  18 ,training loss =  0.0013310264721304288
epoch :  19 ,training loss =  0.0012168204694214875
epoch :  20 ,training loss =  0.0011038511566671806
epoch :  21 ,training loss =  0.0007758810095412573
epoch :  22 ,training loss =  0.00152176303950611
epoch :  23 ,training loss =  0.00150893430343704
epoch :  24 ,training loss =  0.0015629064361217307
epoch :  25 ,training loss =  0.0016699027454258122
epoch :  26 ,training loss =  0.0024230563300731245
epoch :  27 ,training loss =  0.002140815978793344
epoch :  28 ,training loss =  0.006217878724538735
epoch :  29 ,training loss =  0.004464255499917625
(array([[30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0., 29.,  1.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0., 29.,  0.,  0.,  1.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0., 29.,  1.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0., 30.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0., 30.,  0.,  0.,  0.],
        [ 0.,  0.,  1.,  0.,  0.,  1.,  2., 26.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  2.,  0., 28.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0., 30.]]), 0.97)
```

Παρατηρούμε όντως πολύ καλή επίδοση στο testing set.

Bonus

Για το bonus κομμάτι τροποποιούμε την υλοποίηση του lstm και κάνουμε χρήση της μεθόδου pack_padded_sequence. Για να λειτουργήσει αυτό θα πρέπει οι tensors των διανυσμάτων που δίνουμε ως είσοδο κατά την εκπαίδευση και κατά την αξιολόγηση να είναι ταξινομημένοι ως προς το μήκος των διανυσμάτων, και βέβαια αντίστοιχη τροποποίηση θα πρέπει να γίνει και στις ετικέτες. Με την νέα αυτή υλοποίηση τα αποτελέσματα είναι τα εξής:

```
epoch :  0 ,training loss =  0.06970760860105059
epoch :  1 ,training loss =  0.06335207708735964
epoch :  2 ,training loss =  0.05374296070702041
epoch :  3 ,training loss =  0.03714255410343854
epoch :  4 ,training loss =  0.030817514897060038
epoch :  5 ,training loss =  0.026978222535116905
epoch :  6 ,training loss =  0.01791516217444815
epoch :  7 ,training loss =  0.01329494987044539
epoch :  8 ,training loss =  0.012347852369881611
epoch :  9 ,training loss =  0.007923218647518488
epoch :  10 ,training loss =  0.009121411446996255
epoch :  11 ,training loss =  0.009615363016265876
epoch :  12 ,training loss =  0.006288898762996628
epoch :  13 ,training loss =  0.007595129139068078
epoch :  14 ,training loss =  0.007878604855861015
epoch :  15 ,training loss =  0.007059202001277191
epoch :  16 ,training loss =  0.00480435990360079
epoch :  17 ,training loss =  0.0053950995050454096
epoch :  18 ,training loss =  0.004493725088202575
epoch :  19 ,training loss =  0.003755073202947683
epoch :  20 ,training loss =  0.00385470252393731
epoch :  21 ,training loss =  0.003430457714486367
epoch :  22 ,training loss =  0.005907561451912538
epoch :  23 ,training loss =  0.0056137764833367135
epoch :  24 ,training loss =  0.0035830915604940436
epoch :  25 ,training loss =  0.002362487009733415
epoch :  26 ,training loss =  0.003272302241113259
epoch :  27 ,training loss =  0.003816095227723135
epoch :  28 ,training loss =  0.0024862443440100317
epoch :  29 ,training loss =  0.0016021136808688683
(array([[30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0., 30.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0., 29.,  0.,  0.,  0.,  1.,  0.,  0.],
        [ 0.,  0.,  0.,  0., 29.,  0.,  0.,  0.,  1.,  0.],
        [ 0.,  0.,  0.,  0.,  0., 30.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0., 29.,  0.,  0.,  1.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  0., 30.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  1.,  0., 29.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0., 29.]]),
 0.9833333333333333)
```

Ο συνολικός χρόνος εκπαίδευσης μειώνεται μεν, ελάχιστα δε.