ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Αναγνώριση Προτύπων

2η εργαστηριακή άσκηση

Αναγνώριση Φωνής με Κρυφά Μαρκοβιανά Μοντέλα και Αναδρομικά Νευρωνικά

Κωνσταντίνος Τσιγγέλης: 03117149

Χρήστος Σιαφαρίκας: 03117097

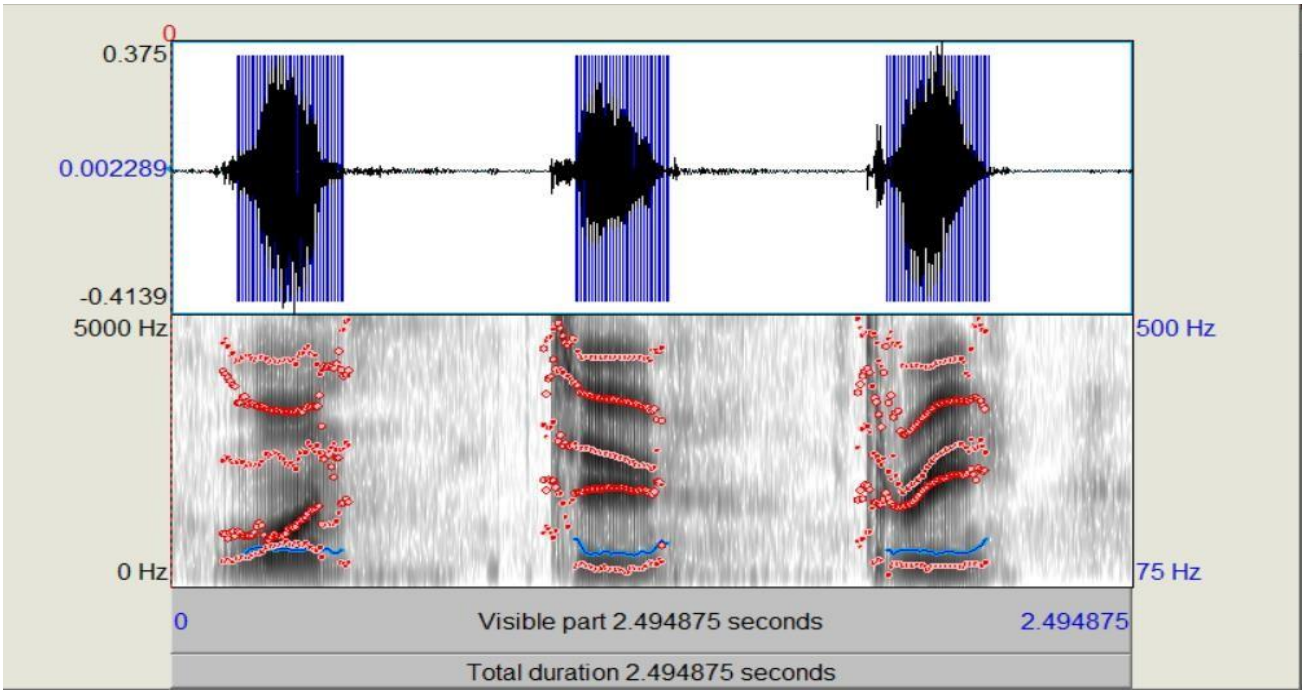
Σκοπός του εργαστηρίου

Σκοπός της συγκεκριμένης εργαστηριακής άσκησης είναι η ανάπτυξη και υλοποίηση ενός συστήματος αναγνώρισης ψηφίων από φωνητικά δεδομένα. Ιδιαίτερα με την κατάλληλη επεξεργασία και αναγνώριση φωνής καλούμαστε να αναπτύξουμε ένα σύστημα αναγνώρισης μεμονωμένων ψηφίων στα αγγλικά.

Εκτέλεση

Βήμα 1

Στο συγκεκριμένο βήμα κάνουμε χρήση της εφαρμογής Praat για ανάλυση των αρχείων ήχου onetwothree1.wav και onetwothree8.wav τα οποία περιέχουν την πρόταση “one two three” από τους ομιλητές 1 και 8 , οι οποίοι είναι άντρας και γυναίκα αντίστοιχα. Για τον ομιλητή 1 καταλήγουμε στο ακόλουθο διάγραμμα spectrogram με την αντίστοιχη κυματομορφή του σήματος φωνής:



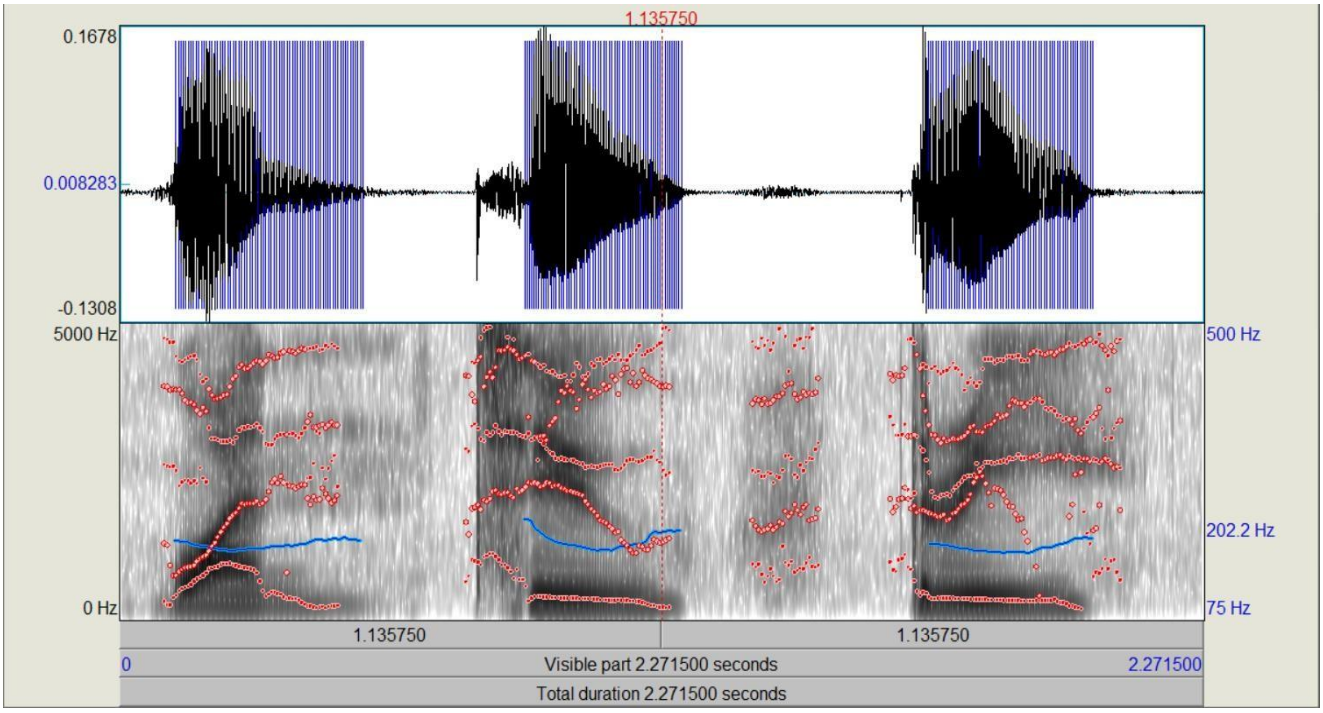
Στο 1ο διάγραμμα φαίνεται η αναπαράσταση του σήματος της φωνής στο φάσμα των συχνοτήτων. Οι παλμοί που φαίνονται με μπλε χρώμα αντιστοιχούν στην αναπαράσταση των φωνηέντων και ιδιαίτερα στα φωνήεντα “α” από τη λέξη “one”, “ου” από τη λέξη “two” και “ι” από τη λέξη “three” αντίστοιχα. Αυτή την διάσπαση του ενιαίου σήματος σε τρία κομμάτια θα χρησιμοποιήσουμε για την εξαγωγή των ζητούμενων χαρακτηριστικών. Στο 2ο διάγραμμα με μπλε χρώμα φαίνεται το pitch των λέξεων που αναπαριστά των θεμελιώδη συνιστώσα της συχνότητας του σήματος φωνής (F0). Αντίστοιχα με κόκκινο φαίνονται τα formants που είναι οι ανώτερες συνιστώσες της συχνότητας του σήματος. Για κάθε φωνήεν αναπαριστούνται πέντε επίπεδα formants (5 επίπεδα από κοκκινες βούλες) που αντιστοιχούν στις ανώτερες συνιστώσες 1 εώς 5. Τα χαρακτηριστικά που θέλουμε να εξάγουμε για κάθε φωνήεν είναι το μέσο του pitch και τα τρία πρώτα formants.

Επιλέγουμε λοιπόν κάθε φορά το κομμάτι ενδιαφέροντος του σήματος φωνής και χρησιμοποιούμε τις εντολές get_pitch και get_formant(i) για να ανλήσουμε τα ζητούμενα χαρακτηριστικά. Τα αποτελέσματά μας φαίνονται ακολούθως (στρογγυλοποίηση στα 3 δεκαδικά ψηφία):

Speaker1_male

φωνήεν	Mean_pitch	Formant_1	Formant_2	Formant_3
'α'	133.883 Hz	759.948 Hz	1284.976 Hz	2423.248 Hz
'ου'	131.704 Hz	378.692Hz	1767.777 Hz	2386.960 Hz
'ι'	132.441 Hz	403.720 Hz	1851.796 Hz	2302.589 Hz

Ακολουθούμε την ίδια συλλογιστική πορεία και για τον δεύτερο ομιλητή.



Speaker2_female

φωνήεν	Mean_pitch	Formant_1	Formant_2	Formant_3
'α'	182.874 Hz	553.280 Hz	1727.965 Hz	2843.524 Hz
'ου'	188.736 Hz	350.330Hz	1721.458 Hz	2685.97 Hz
'ι'	179.892 Hz	398.106 Hz	2116.29 Hz	2759.593 Hz

Από τα παραπάνω δεδομένα μπορούμε να εξάγουμε τα ακόλουθα συμπεράσματα: Αρχικά το mean_pitch βλέπουμε ότι διαφοροποιείται κυρίως ανά ομιλητή εφόσον

για διαφορετικά φωνήεντα του ίδιου ομιλητή παρουσιάζει πολύ μικρή μεταβλητότητα. Ιδιαίτερα επαληθεύεται η θεωρητική μας πρόβλεψη πώς το pitch για ανδρική φωνή κυμαίνεται μεταξύ 85 και 155 Hz ενώ για γυναικεία φωνή μεταξύ 165 και 255Hz. Συνεπώς το mean_pitch σαν χαρακτηριστικό μπορεί σίγουρα να χρησιμοποιηθεί για την διάκριση του φύλου του ομιλητή σε αρσενικού ή θηλυκού γένους .

Επιπλέον, όσον αφορά τις τιμές των formants φαίνεται ότι έχουν πιο σημαντικές διαφοροποιήσεις ανά φωνήεν παρά ανά ομιλητή. Συγκεκριμένα, φαίνεται ότι το “α” έχει και στα δυο φύλα τα υψηλότερα first και third formants και ταυτόχρονα τα χαμηλότερα third_formats , κάνοντας με αυτό τον τρόπο πιο εύκολη την αναγνώρισή του σε σχέση με τα άλλα ψηφία. Από την άλλη, όσον αφορά τη αναγνώριση κάποιου από τα φωνήεντα “ου” και “ι” παρατηρήθηκε ότι τα formants τους δεν παρουσιάζουν ενιαία συμπεριφορά αλλά αυτή επηρεάζεται από το φύλο του ομιλητή. Συγκεκριμένα, το φωνήεν “ου” για άντρα έχει μεγαλύτερη τιμή μόνο για το third_formant από το φωνήεν “ι” , ενώ για γυναίκα το φωνήεν “ι” υπερισχύει για κάθε formant.

Βήμα 2

Στο συγκεκριμένο βήμα δημιουργήσαμε μία συνάρτηση dataparser που διαβάζει όλα τα αρχεία ήχου που δίνονται μέσα στο φάκελο digits και επιστρέφει 3 λίστες οι οποίες περιέχουν :

- Το wav_file που διαβάστηκε με librosa
- τον αντίστοιχο ομιλητή
- το ψηφίο

Παραθέτουμε τα ακόλουθα στοιχεία για να επαληθεύσουμε την ορθη λειτουργία της συνάρτησής μας:

```
the lengths of the lists:
wav_list length: 133
speaker_list length: 133
digit_list length: 133
lets take a closer look at the lists
the first wav file of the wav_list: (array([0.00228882, 0.00271606, 0.0032959 , ...,
0.00253296, 0.00274658,
       0.00265503], dtype=float32), 16000)
the first ten speakers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
the fisrt ten digits: ['one', 'one', 'one', 'one', 'one', 'one', 'one', 'one',
'one', 'one']
```

Βήμα 3

Στο συγκεκριμένο βήμα με χρήση της συνάρτησης feature.mfcc της βιβλιοθήκης librosa εξάγουμε 13 χαρακτηριστικά για κάθε αρχείο ήχου. Αναλυτικότερα τα Mel Frequency Cepstral Coefficients (MFCC’s) αποτελούν βασικά χαρακτηριστικά τα

οποία μπορούν να περιγράψουν επαρκώς το γλωσσικό περιεχόμενο ενός αρχείο ήχου. Είναι πολύ σημαντικό να κατανοήσουμε πώς η μορφή του ήχου που παράγεται από κάποιον άνθρωπο επηρεάζεται από διάφορα φυσικά χαρακτηριστικά της φωνητικής οδού όπως πχ η γλώσσα, τα δόντια κτλ τα οποία επιδρούν άμεσα στην διαμόρφωση του ηχητικού σήματος. Με χρήση των παραπάνων features μπορούμε να εξάγουμε σημαντικές πληροφορίες για την μορφή αυτών των φυσικών χαρακτηριστικών ομιλίας και έτσι να έχουμε μία ακριβή αναπαράσταση του ηχητικού σήματος. Εμβαθύνοντας στην λειτουργία των mfcc το Mel frequency cepstrum είναι μία αναπαράσταση του βραχυπρόθεσμου φάσματος ισχύος του ήχου (short term power spectrum) σε μία κλίμακα συχνότητας mel η οποία προκύπτει από την ακόλουθη σχέση :

$$M(f) = 1125 \ln(1 + f/700)$$

Η μετατροπή αυτή γίνεται καθώς οι ζώνες συχνοτήτων ισαπέχουν στην κλίμακα mel, η οποία προσεγγίζει καλύτερα την απόκριση του ανθρώπινου ακουστικού συστήματος από τις γραμμικά αποστασιοποιημένες ζώνες συχνοτήτων που χρησιμοποιούνται στο κανονικό φάσμα. Αντίστοιχα οι μεταβλητές MFCC αποτελούν τα πλάτη του νέου φάσματος . Επιπλέον το σήμα ήχου συνεχώς εναλλάσσεται για αυτό θεωρούμε ότι σε μικρές κλίμακες χρόνου το σήμα ήχου δεν παρουσιάζει μεγάλη μεταβλητότητα. Για αυτό τον λόγο χρησιμοποιούμε μήκος παραθύρου 25ms με βήμα 10ms .

Στην συνέχεια υπολογίζουμε τις τοπικές παραγώγους deltas και delta-deltas , 1ης και 2ης τάξης αντίστοιχα. Ο υπολογισμός τους γίνεται με χρήση της συνάρτησης feature.delta(mfcc,order=i) της βιβλιοθήκης librosa για i=1,2 που καθορίζει την τάξη της παραγώγου. Οι παραπάνω παράγωγοι περιγράφουν την δυναμική συμπεριφορά των mel frequency cepstrum coefficients και προκύπτουν από την ακόλουθη μαθηματική σχέση :

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$

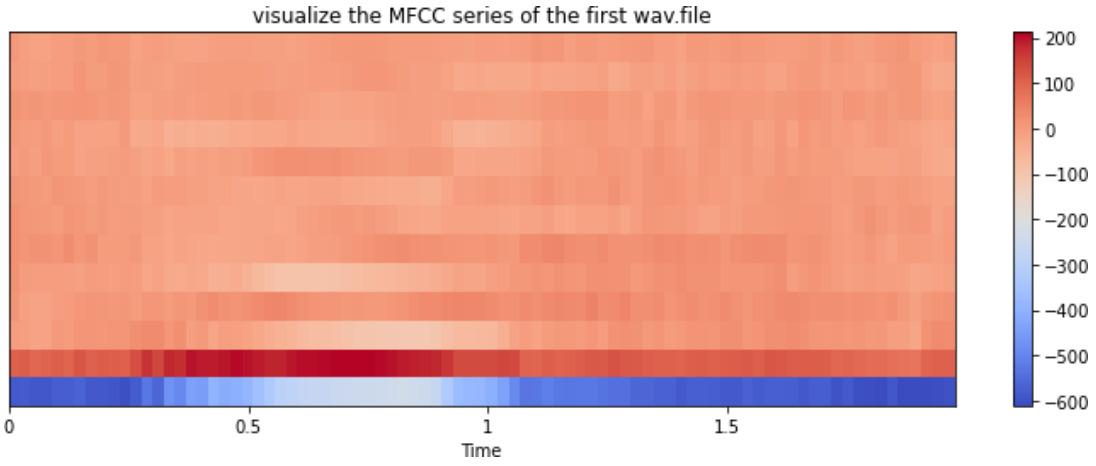
.

Στην συνέχεια παρουσιάζουμε τα αποτελέσματα μας για το πρώτο αρχείο ήχου. Παρατηρούμε ότι κάθε χαρακτηριστικό εκ των 13ων αποτελείται από μία ακολουθία τιμών η οποία είναι διαφορετικού μεγέθους για κάθε wav.file. Αυτό εύκολα εξηγείται καθώς κάθε αρχείο ήχου δεν έχει την ίδια διάρκεια και συνεπώς ο αριθμός των παραθύρων (windows length:25ms) που “χωράνε” σε κάθε αρχείο ήχου είναι διαφορετικός.

```
MFCC of the wav files:  [[-5.84121277e+02 -5.80639954e+02 -5.90789246e+02 ... -6.08510864e+02
  -6.07412476e+02 -6.00127747e+02]
 [ 1.08374863e+02  1.14976601e+02  9.90701752e+01 ...  1.02533386e+02
   1.07109459e+02  1.08890656e+02]
 [-4.57091749e-01 -4.43130350e+00 -1.59686699e+01 ...  2.05762825e+01
   3.25283813e+01  3.11435432e+01]
 ...
```

```
deltas(first derivative)of the MFCC features:  [[ 0.49525654  0.49525654  0.49525654 ...
-0.1688151  -0.1688151
  -0.1688151 ]
 [ 0.7908488   0.7908488   0.7908488  ...  2.2270658   2.2270658
   2.2270658 ]
 [ 2.9624639   2.9624639   2.9624639  ...  2.595571    2.595571
   2.595571  ]
 ...
```

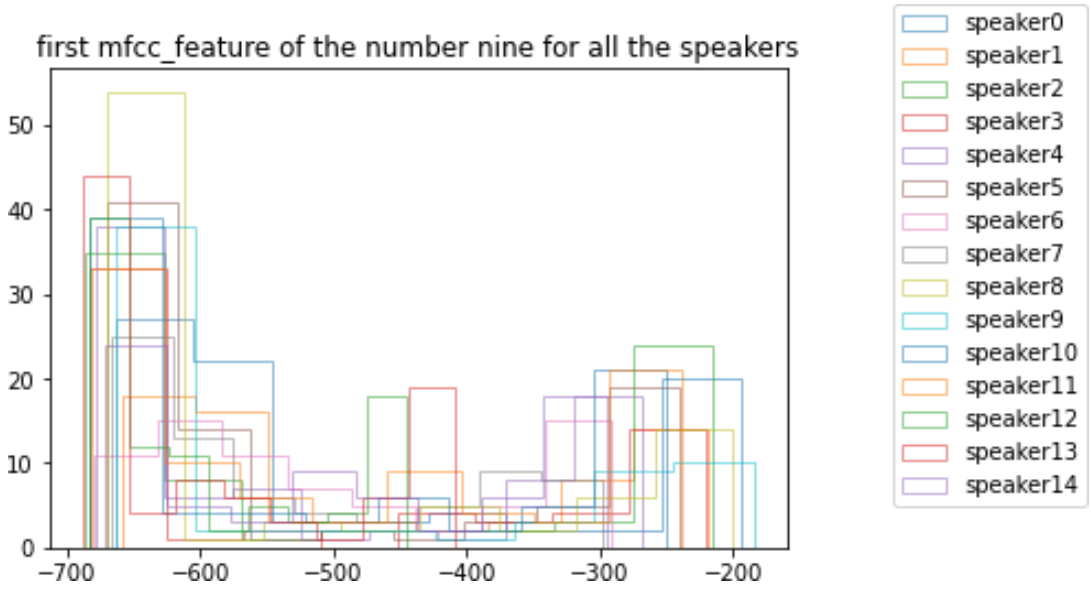
```
delta_deltas(second derivative) of the MFCC features:  [[-7.1147192e-01 -7.1147192e-01
-7.1147192e-01 ...  3.8382366e-01
   3.8382366e-01  3.8382366e-01]
 [ 4.9693665e-01  4.9693665e-01  4.9693665e-01 ...  2.7125599e+00
   2.7125599e+00  2.7125599e+00]
 [ 9.8638272e-01  9.8638272e-01  9.8638272e-01 ...  3.9480970e+00
   3.9480970e+00  3.9480970e+00]
 ...
```

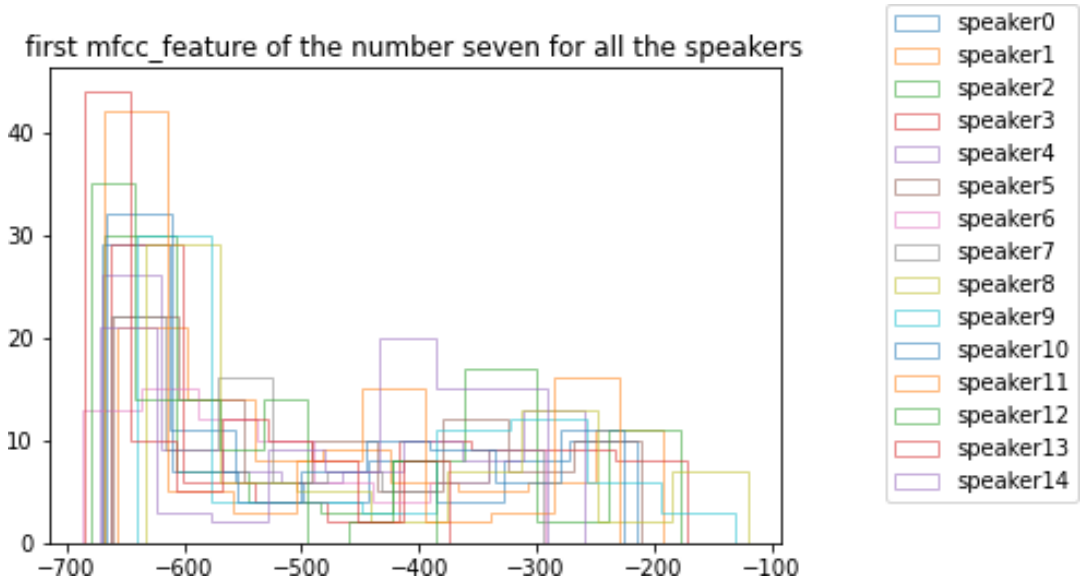


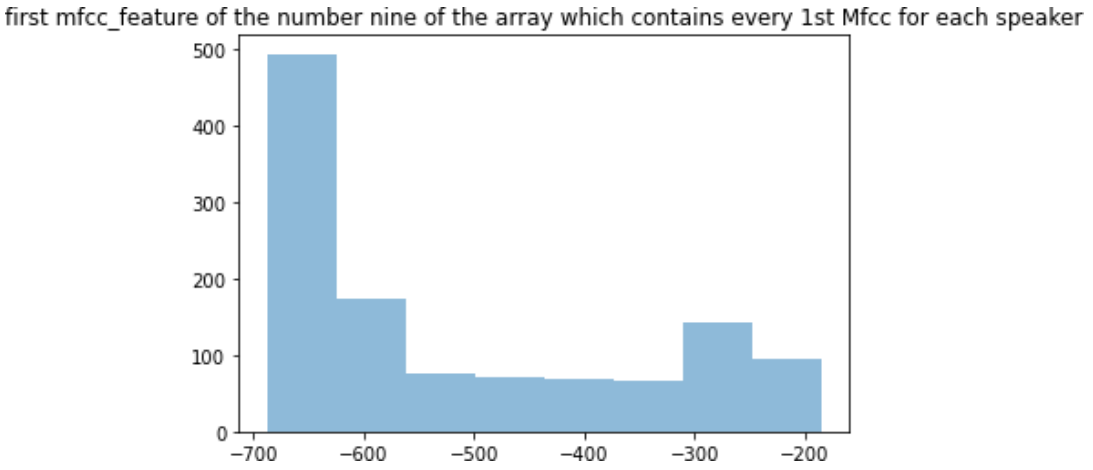
Βήμα 4

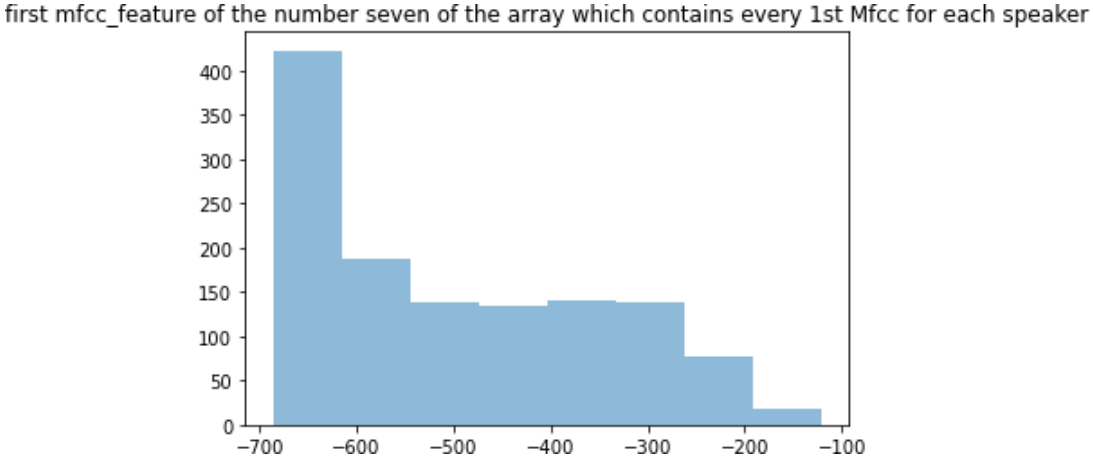
Αρχικά αναπαριστούμε τα ιστογράμματα του 1ου και του 2ου MFCC των ψηφίων 7 και 9, που αποτελούν τα τελευταία ψηφία του αριθμού μητρώου μας , για κάθε εκφώνησή τους . Το ιστόγραμμα είναι ένα πολύ σημαντικό διάγραμμα στην στατιστική και στην ανάλυση δεδομένων καθώς μας βοηθάει να κατανοήσουμε την κατανομή των δεδομένων στον χώρο τιμών. Αναλυτικότερα το διάγραμμα αυτό οπτικοποιεί την ποσότητα των δεδομένων με bar plots τα οποία ανήκουν σε ένα συγκεκριμένο διάστημα τιμών το οποίο καλείται bin. Γενικά ο αριθμός των bins στις οποίες θα χωρίσουμε το εύρος τιμών των δεδομένων μας δεν είναι σταθερός και διαφέρει για κάθε dataset αλλά και για κάθε επιθυμητή χρήση. Εμείς αποφασίσαμε να χρησιμοποιήσουμε 8 bins εφόσον παρατηρήσαμε ότι αρκούν για την διάκριση ψηφίων. Επίσης είναι σημαντικό να τονίσουμε ότι για κάθε mfcc (1ο και 2ο) παρουσιάζουμε 2 τύπους ιστογραμμάτων ώστε να γίνουν πιο κατανοητά τα

συμπεράσματά μας. Ιδιαίτερα παρουσιάζουμε αρχικά ένα ιστόγραμμα το οποίο περιέχει σε ένα κοινό διάγραμμα τα επιμέρους ιστογράμματα όλων των ομιλητών για δεδομένο ψηφίο και mfcc και στην συνέχεια παρουσιάζουμε ένα ιστόγραμμα το οποίο περιέχει την συνολική κατανομή των mfcc ($1^{ων}$ ή $2^{ων}$). Τα αποτελέσματά μας φαίνονται ακολούθως:

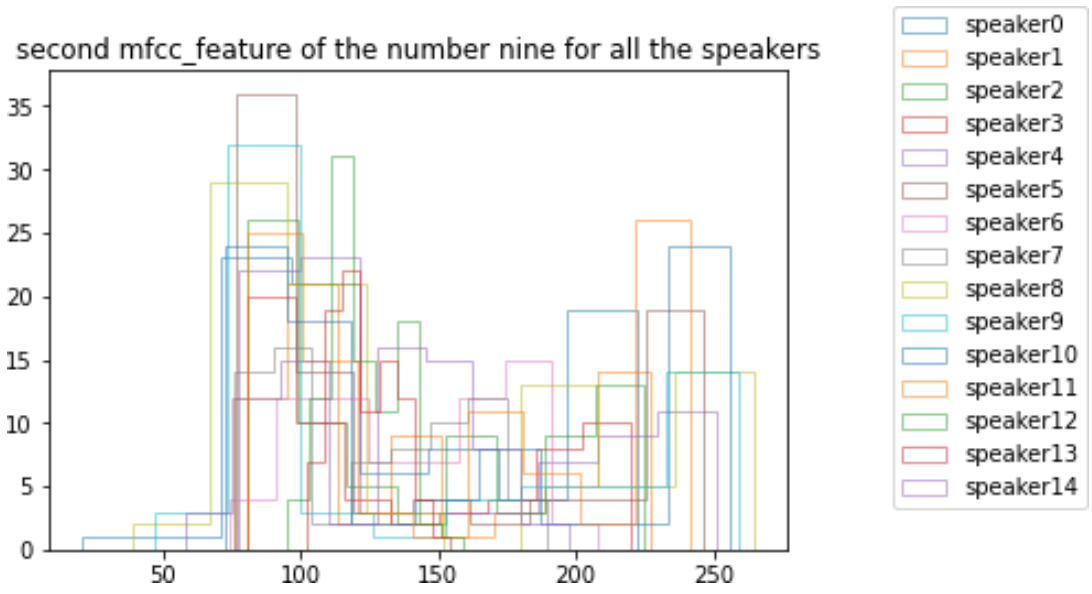


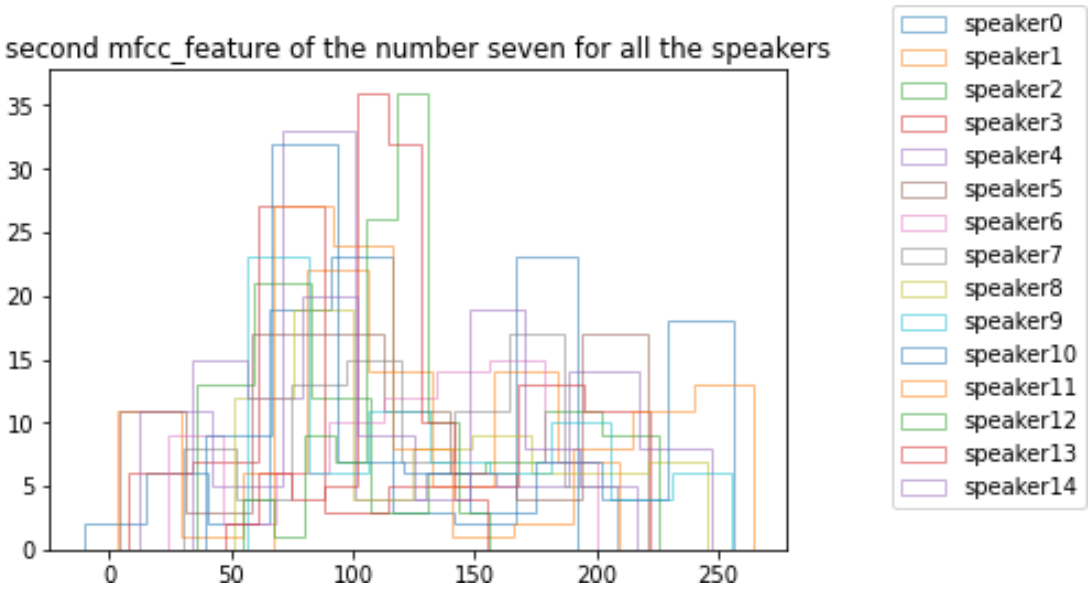


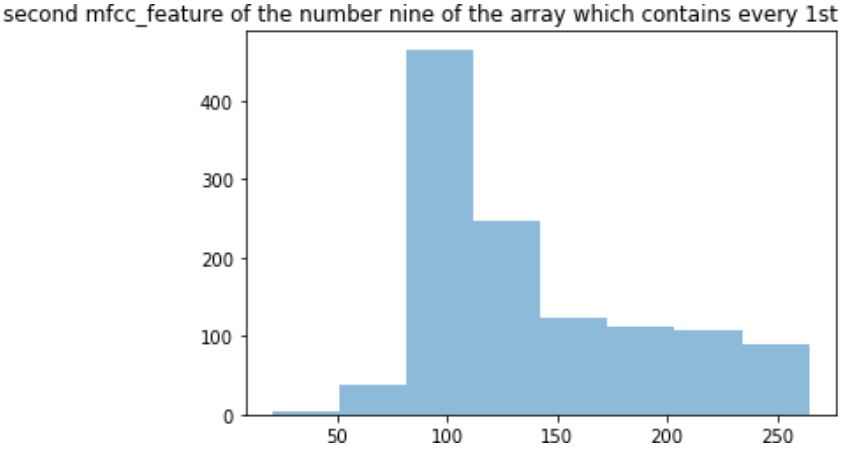


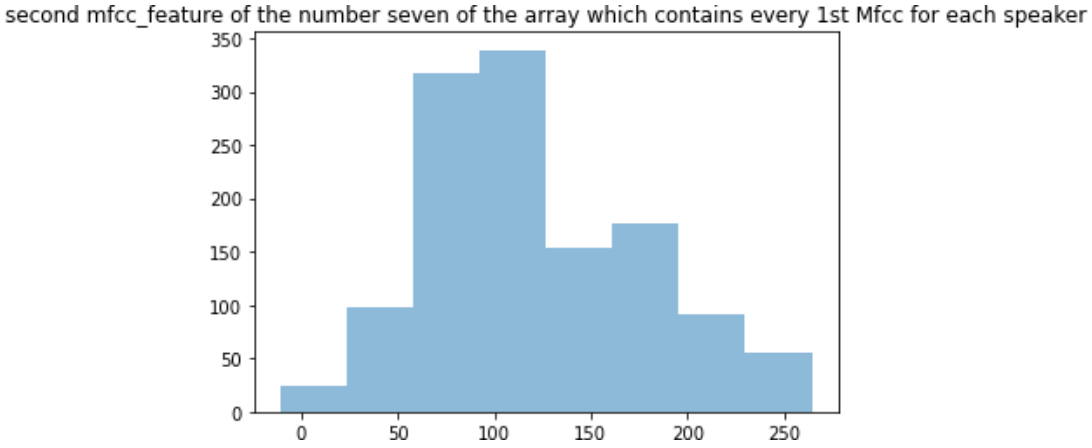


Από τα τελευταία δύο διαγράμματα φαίνεται ότι η διάκριση των ψηφίων 7 και 9 είναι δυνατή με αποκλειστική χρήση του 1ου mfcc εφόσον για το ψηφίο εννέα φαίνεται να έχουμε πολύ μεγαλύτερη συγκέντρωση τιμών τόσο σε πολύ μικρές τιμές mfcc(<-650) αλλά και σε μεγάλες τμές mfcc μεταξύ -350 και -200. Αντίθετα οι ενδιάμεσες τιμές από περίπου -500 έως -350 εμφανίζουν μεγαλύτερη συγκέντρωση στο ψηφίο 7.Επίσης σε αρκετές περιπτώσεις αν έχουμε προσδιορίσει το ψηφίο μπορούμε με χρήση των αρχικών διαγραμμάτων να καθορίσουμε και τον ομιλητή εφόσον παρατηρούμε ότι η κατανομή του 1ου mfcc παρουσιάζει μεταβλητότητα μεταξύ των ομιλητών.


second mfcc_feature of the number nine for all the speakers
speaker0
speaker1
speaker2
speaker3
speaker4
speaker5
speaker6
speaker7
speaker8
speaker9
speaker10
speaker11
speaker12
speaker13
speaker14

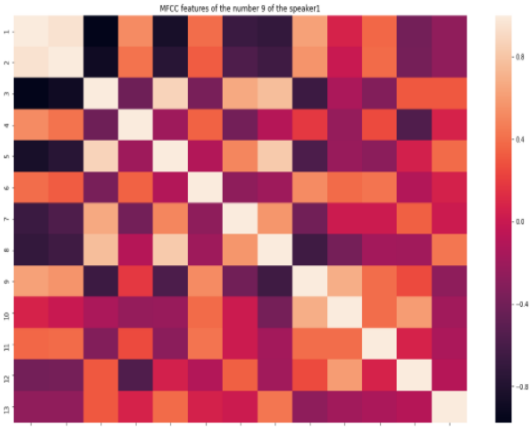

second mfcc_feature of the number seven for all the speakers
speaker0
speaker1
speaker2
speaker3
speaker4
speaker5
speaker6
speaker7
speaker8
speaker9
speaker10
speaker11
speaker12
speaker13
speaker14

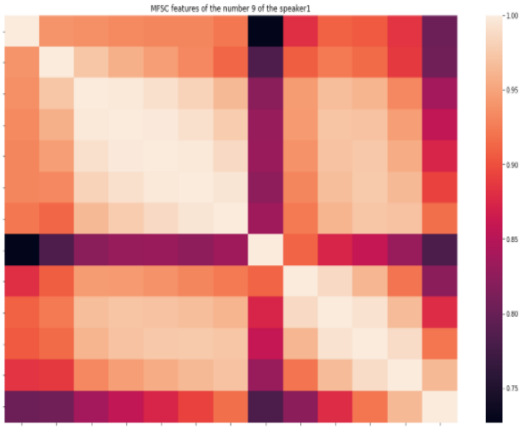

second mfcc_feature of the number nine of the array which contains every 1st Mfcc for each speaker

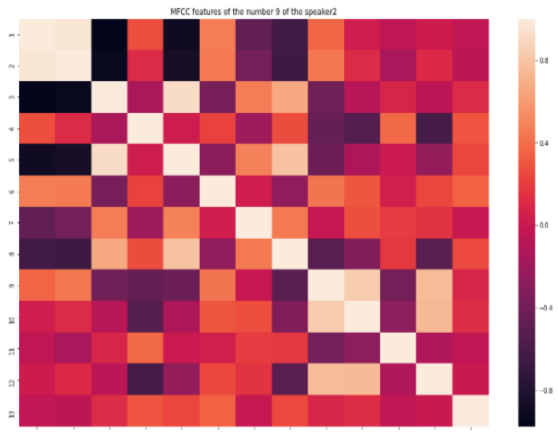


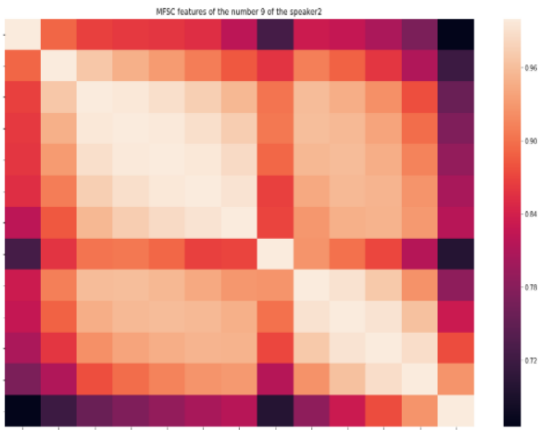
Σε αντίστοιχα συμπεράσματα με τα προηγούμενα καταλήγουμε αν παρατηρήσουμε την κατανομή του 2^{ου} mfcc για τα ψηφία 9 και 7 σύμφωνα με τα προηγούμενα διαγράμματα. Η διάκριση των ψηφίων είναι δυνατή με αποκλειστική χρήση του 2^{ου} mfcc εφόσον για το ψηφίο εννέα φαίνεται να έχουμε πολύ μικρότερη αυτή τη φορά συγκέντρωση τιμών τόσο σε πολύ μικρές τιμές mfcc(<70) αλλά και σε μεγάλες τμές mfcc μεταξύ 200 και 250 . Αντίθετα οι ενδιάμεσες τιμές από περίπου 80 έως 140 εμφανίζουν αρκετά μικρότερη συγκέντρωση στο ψηφίο 7.Επίσης σε αρκετές περιπτώσεις αν έχουμε προσδιορίσει το ψηφίο μπορούμε με χρήση των αρχικών διαγραμμάτων να καθορίσουμε και τον ομιλητή εφόσον παρατηρούμε ότι η κατανομή του 2^{ου} mfcc παρουσιάζει μεταβλητότητα μεταξύ των ομιλητών.

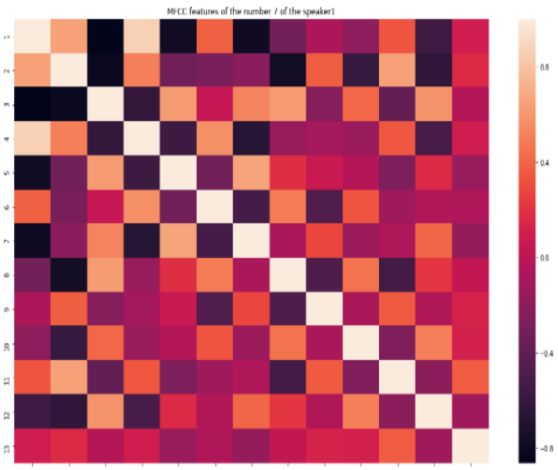
Στην συνέχεια εξάγουμε για 2 εκφωνήσεις των ψηφίων 7 και 9 από 2 διαφορετικούς ομιλητές τα Mel Filterbank Spectral Coefficients (MFSCs), δηλαδή τα χαρακτηριστικά που εξάγονται αφού εφαρμοστεί η συστοιχία φίλτρων της κλίμακας Mel πάνω στο φάσμα του σήματος φωνής αλλά χωρίς να εφαρμοστεί στο τέλος ο μετασχηματισμός DCT . Αναπαριστούμε γραφικά τη συσχέτιση των MFSCs και των MFCCs για την κάθε εκφώνηση. Τα αποτελέσματά μας φαίνονται ακολούθως:

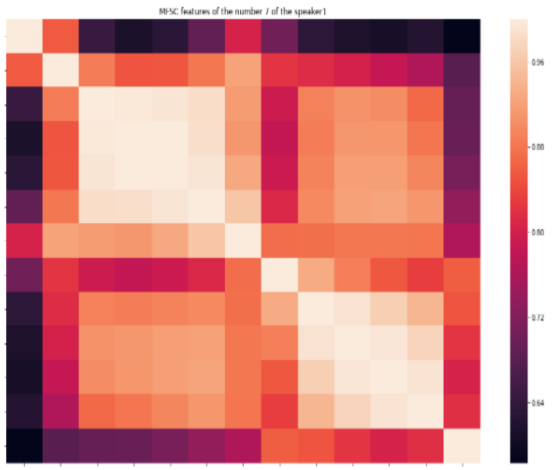


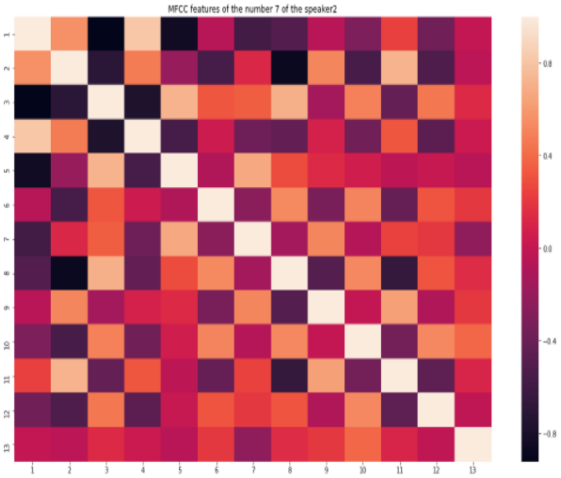


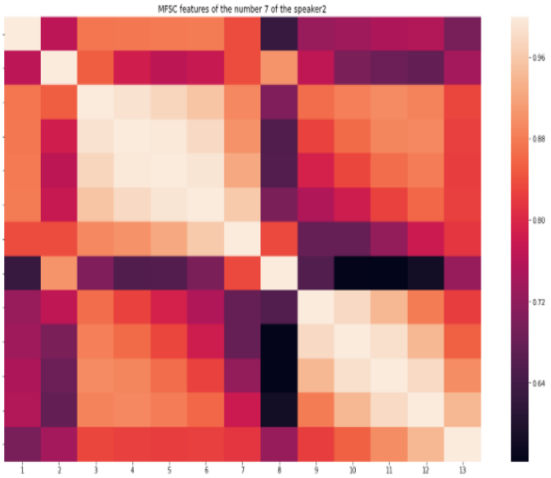












Από τους παραπάνω correlation_matrixes παρατηρούμε ότι τα mfcc χαρακτηριστικά (αριστερή στήλη) εμφανίζουν αισθητά μικρότερη συσχέτιση σε σχέση με τα mfsc χαρακτηριστικά. Αυτή είναι μια επιθυμητή ιδιότητα για τα χαρακτηριστικά που αξιοποιούνται στο learning. Θέλουμε να είναι ασυσχέτιστα μεταξύ τους ώστε το καθένα να προσθέτει μια καινούρια και μοναδική πληροφορία που χαρακτηρίζει το δείγμα. Με άλλα λόγια όσο περισσότερο ασυσχέτιστα είναι τα δεδομένα μας τόσο

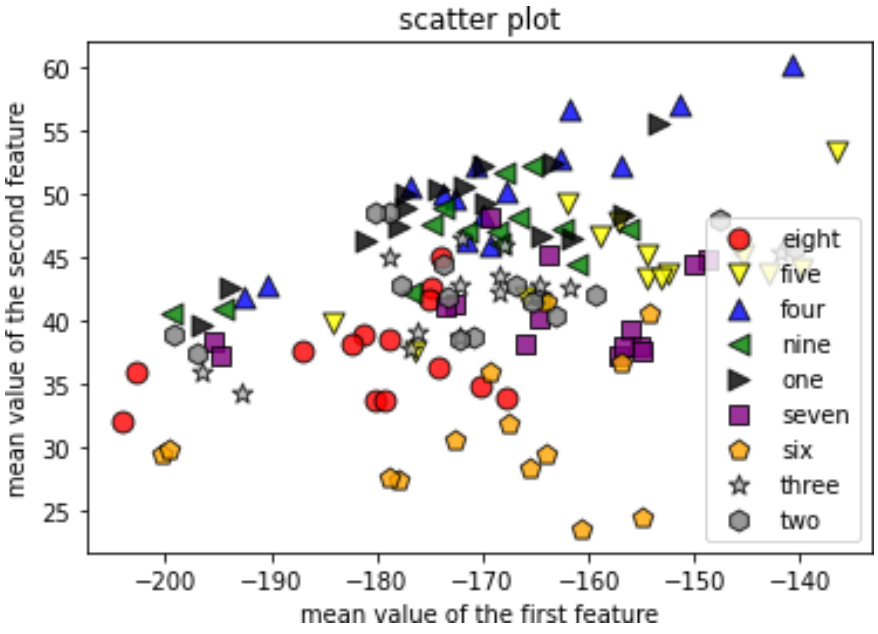
μεγαλύτερη και πιο συμπυκνωμένη πληροφορία παίρνουμε. Αυτός είναι και ο λόγος που προτιμάμε τα mfccs ως περιγραφητές.

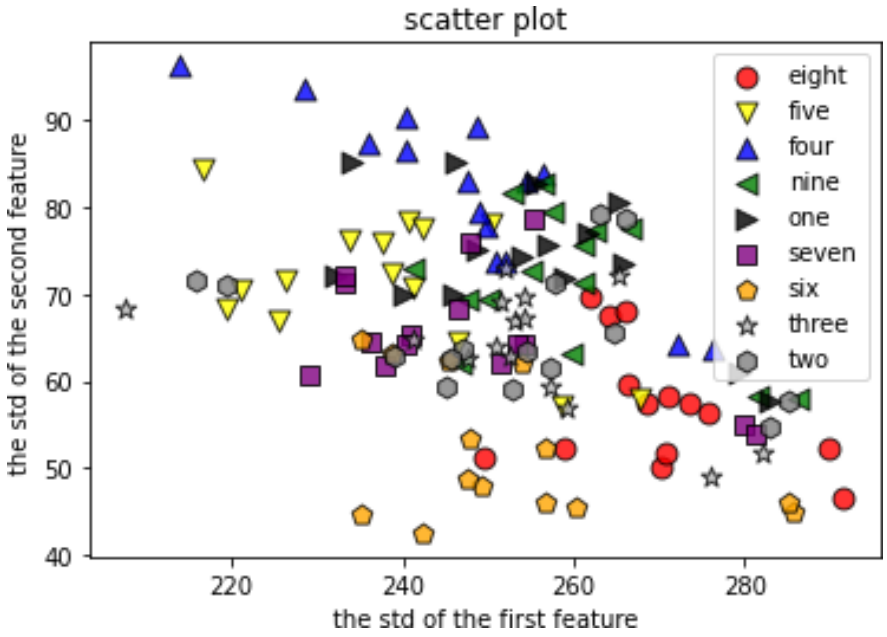
Βήμα 5

Στο συγκεκριμένο βήμα δημιουργούμε ένα ενιαίο διάνυσμα χαρακτηριστικών το οποίο προκύπτει ως εξής: συνενώνουμε τα mfccs – deltas – delta-deltas και έπειτα για κάθε εκφώνηση δημιουργούμε ένα διάνυσμα παίρνοντας τη μέση τιμή και την τυπική απόκλιση κάθε χαρακτηριστικού για όλα τα παράθυρα της εκφώνησης. Έτσι λοιπόν καταλήγουμε σε ένα διάνυσμα 26 τιμών που αντιπροσωπεύει κάθε εκφώνηση. Οι 13 πρώτες τιμές αντιστοιχούν στην μέση τιμή κάθε χαρακτηριστικού ενώ οι 13 επόμενες αντιστοιχούν στην τυπική απόκλιση καθενός. Αναλυτικότερα για τον πίνακα των χαρακτηριστικών:

```
lets take a look inside the feature array
the length of the feature array is: 133
the shape of the feature_array that describes the first wav file is: (26,)
the features that describe the first wac file are the following
 [-1.6162904e+02  4.6411625e+01 -4.2558022e+00  7.3691907e+00
 -4.8146443e+00  3.4674647e+00 -1.5104554e+00 -1.2147608e+00
 -4.4469640e-01 -4.5221763e+00  5.2166486e-01 -2.0092385e+00
 -4.7572985e-02  2.4000714e+02  6.9868599e+01  2.5001829e+01
  1.2966921e+01  2.0731041e+01  1.1362175e+01  6.9690285e+00
  8.2081079e+00  7.2592425e+00  1.1117569e+01  5.5376658e+00
  6.4181018e+00  4.0425367e+00]
```

Αναπαριστούμε τώρα με scatter_plot το dataset χρησιμοποιώντας αρχικά την μέση τιμή των πρώτων δύο χαρακτηριστικών (feature_array[:,0] και feature_array[:,1]) και στην συνέχεια την std τιμή των ίδιων χαρακτηριστικών (feature_array[:,13] και feature_array[:,14]).

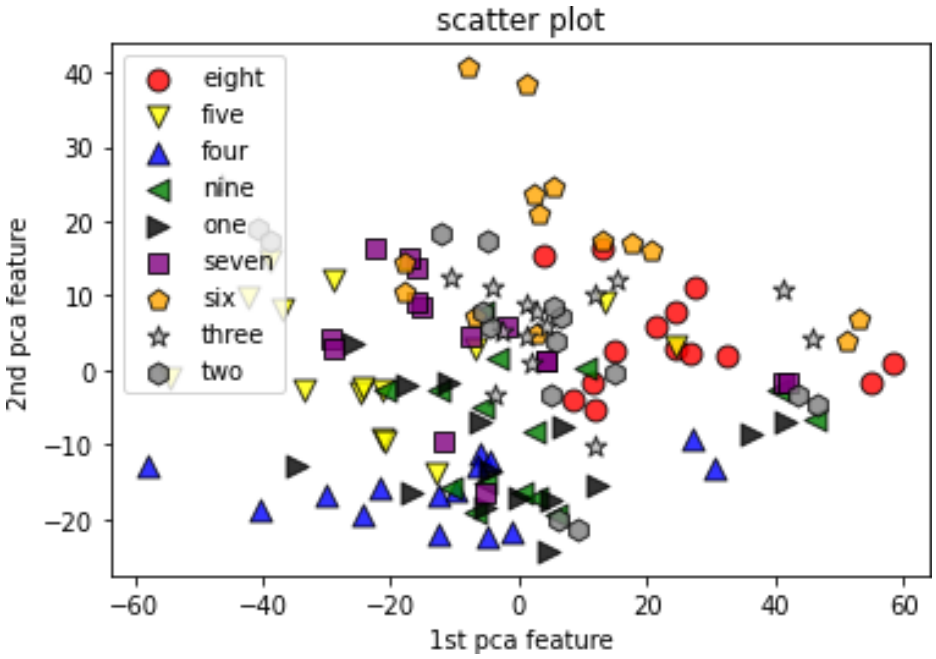


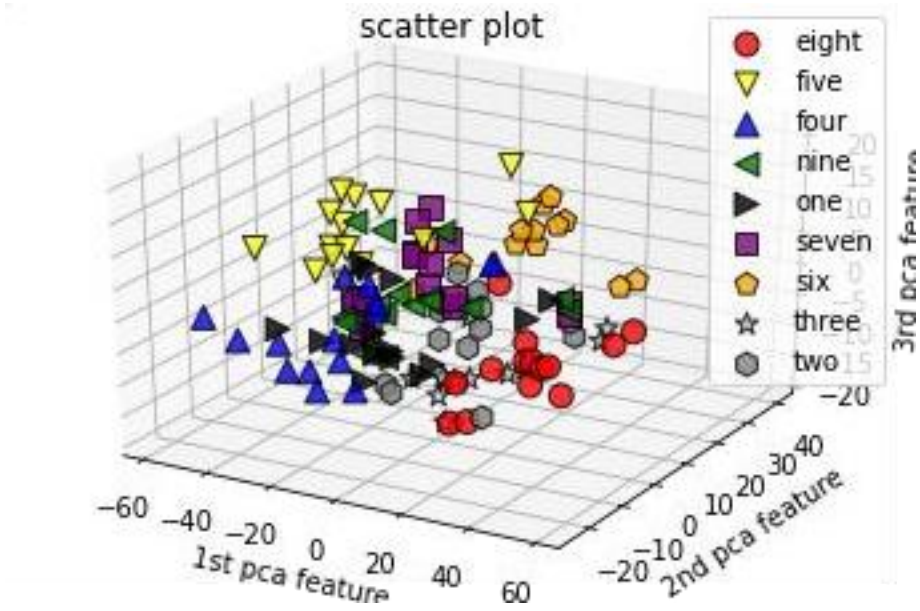


Από τα παραπάνω scatter plots συμπεραίνουμε ότι η κλάσεις δεν είναι διαχωρίσιμες με χρήση αυτών των ζευγών χαρακτηριστικών (μέσης τιμής και τυπικής απόκλισης) παρόλα αυτά εμφανίζουν σε κάποιο ποσοστό μία ομαδοποίηση.

Βήμα 6

Στο συγκεκριμένο βήμα εφαρμόζουμε PCA (Principal Component Analysis) για να μειώσουμε την διάσταση του διανύσματος των χαρακτηριστικών μας και απεικονίζουμε τα αποτελέσματά μας. Αρχικά εφαρμόζουμε pca για n_components=2 και στην συνέχεια για n_components=3 και παρουσιάζουμε τα αντίστοιχα scatter_plots αλλά και το ποσοστό της αρχικής διασποράς που διατηρούν οι συνιστώσες που προέκυψαν χρησιμοποιώντας την εντολή pca.explained_variance_ratio_.






```
Το ποσοστό της αρχικής διασποράς που διατηρούν τα n=2 pca components [0.6066023  0.18282329]
Το ποσοστό της αρχικής διασποράς που διατηρούν τα n=3 pca components [0.6066023  0.18282329 0.06786005]
```

Για να δούμε τι σημαίνουν οι παραπάνω τιμές, ας τις φανταστούμε ως διανύσματα πάνω από τα δεδομένα εισόδου. Αυτά τα διανύσματα αντιπροσωπεύουν τους *κύριους άξονες* των δεδομένων και το μήκος του διανύσματος είναι μια

ένδειξη του πόσο "σημαντικός" είναι αυτός ο άξονας για την περιγραφή της κατανομής των δεδομένων .Πιο συγκεκριμένα, είναι ένα μέτρο της διακύμανσης των δεδομένων όταν προβάλλεται σε αυτόν τον άξονα. Η προβολή κάθε σημείου δεδομένων στους κύριους άξονες είναι τα principal components των δεδομένων. Βλέπουμε λοιπόν πως κυρίαρχος άξονας είναι ο πρώτος με τιμή explained_variance_ratio =0.60 με τους υπόλοιπους άξονες να υπολείπονται σημαντικά αυτού. Η αντίστοιχη τιμή για τον 3^{ο} άξονα είναι **μόλις 0.06!!** Αυτό σημαίνει πως η συμβολή του 3^{ου} principal component για την περιγραφή της κατανομής των δεδομένων στον χώρο είναι αμελητέα ενώ η αντίστοιχη συμβολή του 1^{ου} και 2^{ου} principal components είναι καταλητική.

Όσον αφορά το scatter plot ούτε σε αυτήν την περίπτωση ,που έχουμε εφαρμόσει pca για μείωση των διαστάσεων ,οι κλάσεις είναι διαχωρίσιμες. Βέβαια το 3D plot φαίνεται να εμφανίζει καλύτερα αποτελέσματα ως προς το clustering.

Βήμα 7

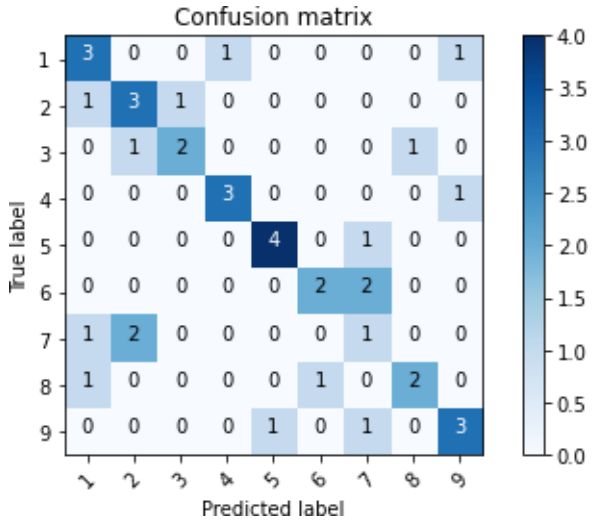
Στο συγκεκριμένο βήμα μελετάμε την λειτουργία ορισμένων ταξινομητών πάνω στο dataset μας και στο διάνυσμα χαρακτηριστικών που υλοποιήσαμε. Αναλυτικότερα οι ταξινομητές που χρησιμοποιούμε (σε default τιμές) είναι οι LogisticRegression(),SVC(),GausiianNB(),K-NearestNeighbot() καθώς και ο CustomNBClassifier() που είχαμε υλοποιήσει στην πρώτη εργαστηριακή άσκηση.

Αρχικά μετατρέπουμε τα labels των δεδομένων μας από strings σε ints ούτως ώστε να λειτουργήσουν καλύτερα τόσο οι classifiers όσο και η συνάρτηση που πλοταρει τον confusion matrix.

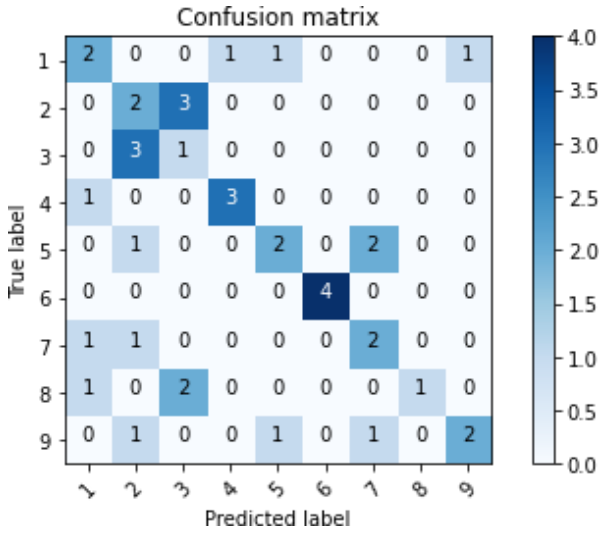
Στην συνέχεια διαχωρίζουμε το dataset μας σε train και test με αναλογία 70%/30% αντίστοιχα .Μάλιστα κάνουμε stratified_train_test_split ούτως ώστε να εμπεριέχονται οι ίδιες αναλογίες κλάσεων τόσο στο train όσο και στο test set.

Τέλος πριν τρέξουμε τους ταξινομητές κανονικοποιούμε και τα δεδομένα μας με χρήσης της StandardScaler() της βιβλιοθήκης sklearn.preprocessing. Τα αποτελέσματά μας φαίνονται ακολούθως:

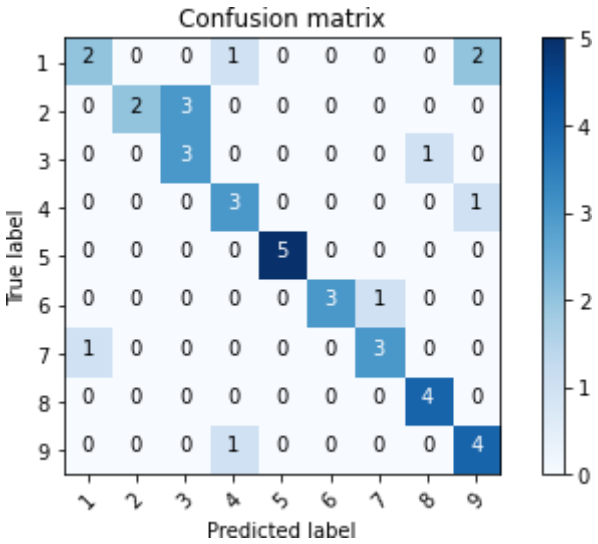
The performance of GaussianNB(priors=None, var_smoothing=1e-09) classifier is : accuracy_score is 0.575.



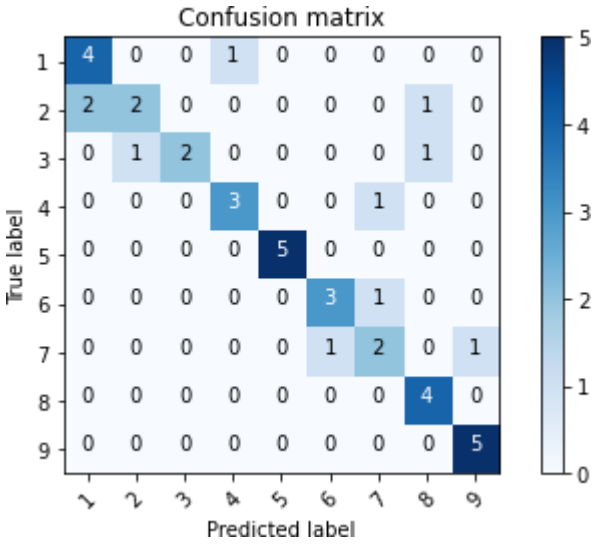
The performance of KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski', , n_neighbors=5, p=2, weights='uniform') classifier is : accuracy_score is 0.475



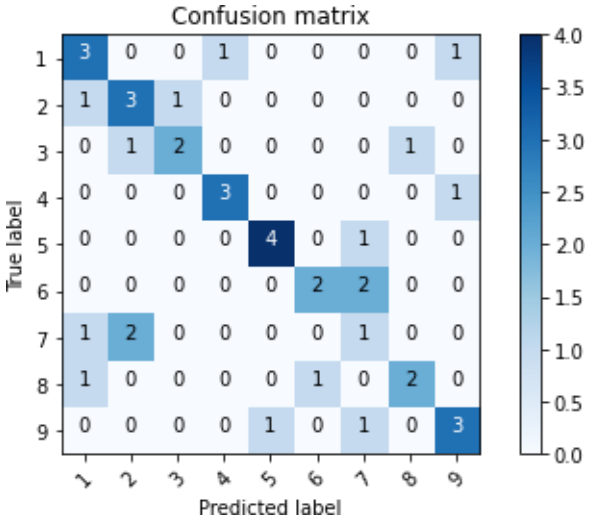
The performance of SVC(degree=3, gamma='auto_deprecated', kernel='rbf') classifier is : accuracy_score is 0.725



The performance of LogisticRegression(max_iter=100 ,penalty='l2', random_state=None, solver='warn', tol=0.0001) classifier is : accuracy_score is 0.75



The performance of CustomNBClassifier() classifier is : accuracy_score is 0.575



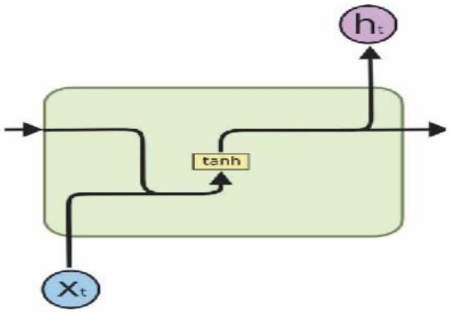
Από τα παραπάνω βλέπουμε ότι την καλύτερη επίδοση εμφανίζει ο ταξινομητής LogisticRegression().

Βήμα 8

Στο βήμα αυτό δημιουργήσαμε ακολουθίες 10 σημείων ενός ημιτόνου και ενός συνημιτόνου με συχνότητα f = 40 Hz επιλέγοντας σταθερή και μικρή απόσταση ανάμεσα στα διαδοχικά σημεία. Στη συνέχεια παρουσιάζομαι τα διάφορα δίκτυα που χρησιμοποιούμε.

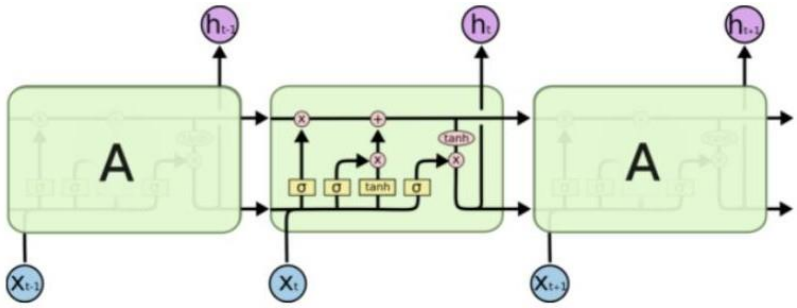
RNN

Το RNN παίρνει είσοδο από το προηγούμενο βήμα και την τρέχουσα είσοδο. Εδώ το tanh είναι η λειτουργία ενεργοποίησης ενώ αντί για tanh μπορούμε να χρησιμοποιήσουμε και άλλη λειτουργία ενεργοποίησης. Παρακάτω δίνουμε και την υλοποίηση του σε φωτογραφία.



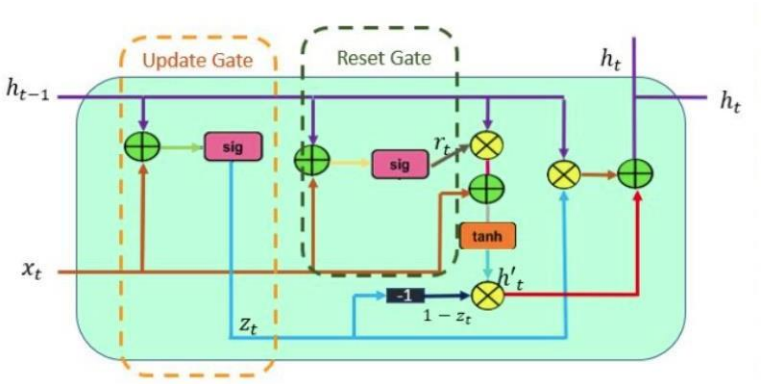
LSTM

Η δημοτικότητα του LSTM οφείλεται στον μηχανισμό Getting που εμπλέκεται με κάθε κύτταρο LSTM. Σε ένα κανονικό κελί RNN, η είσοδος στη χρονική σήμανση και στην κρυφή κατάσταση από το προηγούμενο χρονικό βήμα περνά μέσα από το επίπεδο ενεργοποίησης για να ληφθεί μια νέα κατάσταση ενώ στο LSTM η διαδικασία είναι ελαφρώς πιο περίπλοκη, όπως μπορείτε να δείτε στην παρακάτω αρχιτεκτονική. Κάθε φορά λαμβάνει είσοδο από τρεις διαφορετικές καταστάσεις όπως η τρέχουσα κατάσταση εισόδου, η βραχυπρόθεσμη μνήμη από το προηγούμενο κελί και τέλος η μακροπρόθεσμη μνήμη.



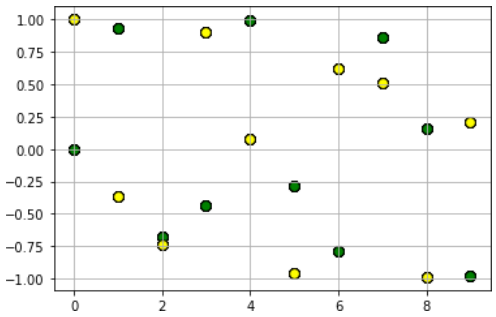
GRU

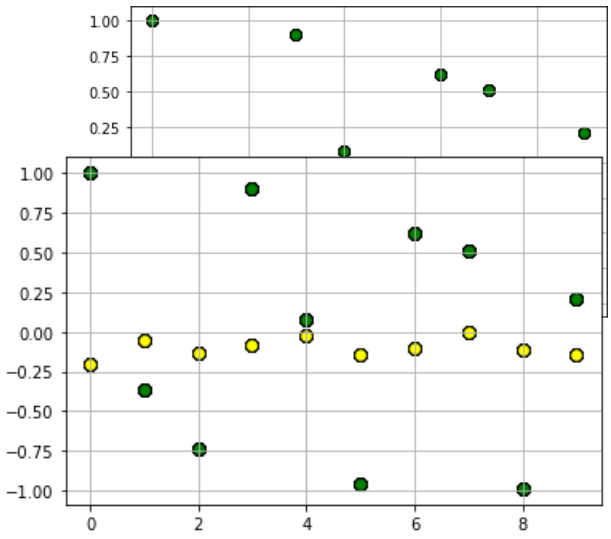
Η ροή εργασίας της Gated Recurrent Unit, εν συντομία GRU, είναι ίδια με το RNN, αλλά η διαφορά έγκειται στη λειτουργία και τις πύλες που σχετίζονται με κάθε μονάδα GRU. Για να λύσει το πρόβλημα που αντιμετωπίζει το τυπικό RNN, η GRU ενσωματώνει τους δύο μηχανισμούς λειτουργίας πύλης που ονομάζονται πύλη ενημέρωσης και πύλη επαναφοράς. Παρακάτω δίνουμε και την υλοποίηση του.



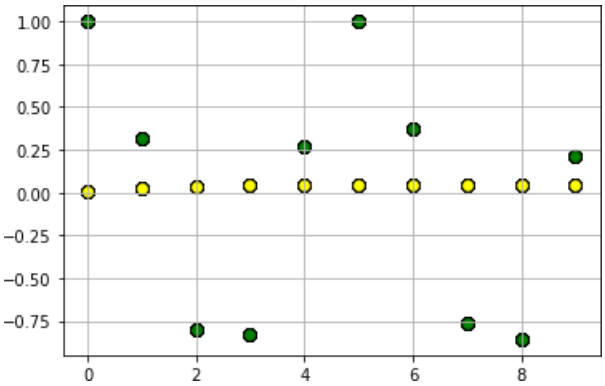
Ενδεικτικά κάποια διαγράμματα.

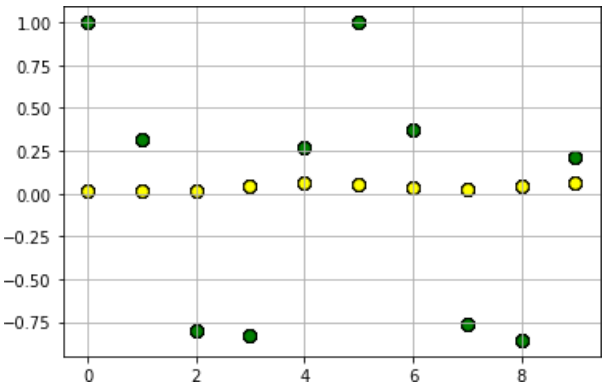
RNN

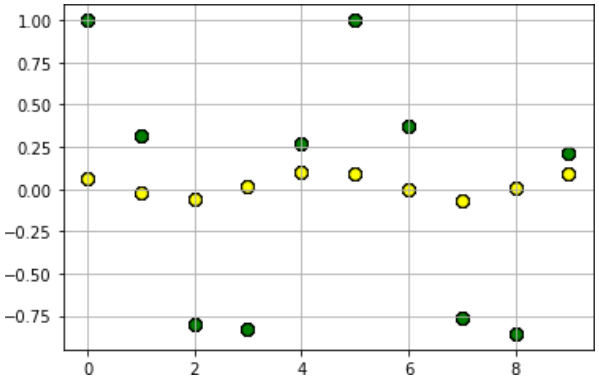

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8

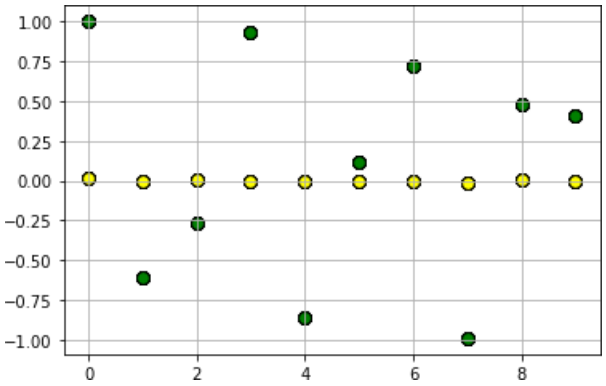
LSTM

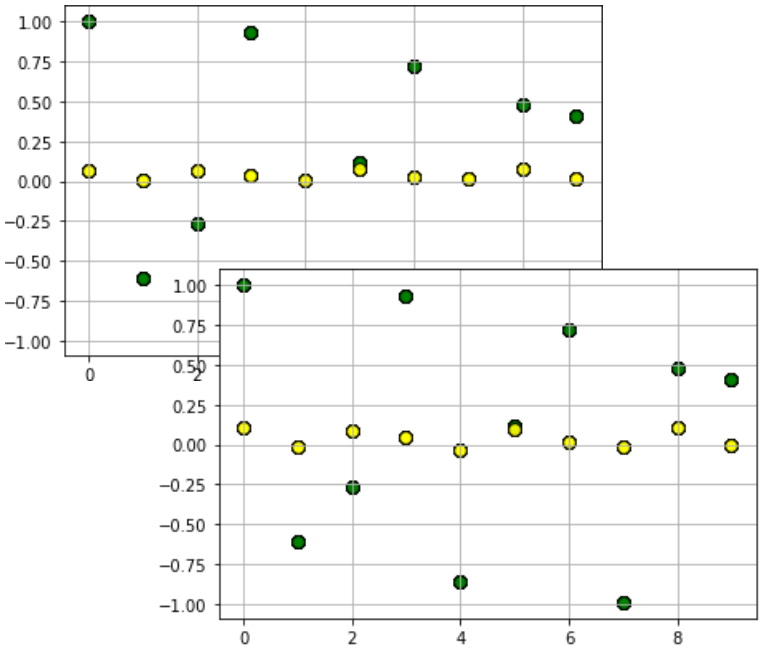

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0
2
4
6
8


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0
2
4
6
8


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0
2
4
6
8

GRU





Βήμα 9

Για το κυρίως μέρος της άσκησης θα χρησιμοποιηθεί ένα μεγαλύτερο σετ δεδομένων, το Free Spoken Digit Dataset (FSDD), το οποίο κατεβάζουμε από το github: https://github.com/Jakobovski/free-spoken-digit-dataset.

Με βάση τις συναρτήσεις που μας έχουν δοθεί στο βοηθητικό υλικό υλοποιούμε μία συνάρτηση parser η οποία διαβάζει τα wav αρχεία ,εξάγει τα mfcc χαρακτηριστικά (13 στο πλήθος) και κανονικοποιεί τα δεδομένα μας με χρήση της StandardScaler(). Τέλος χωρίζει και το dataset σε train και test set.

```
lets take a look into the train and the test set
the length of the train set is: 2700
the length of the test set is: 300
the shape of the feature_array for the first wav file is: (44, 13)
the feature_array for the first wav file is: [[-0.44175017 -3.1606588   0.15431444  1.0152982  -0.43095726  1.2294226
  -0.523808    2.0038197  -0.61440116  0.64472723 -0.6027715   1.3812845
   0.38106018]
 [-0.7590125  -2.827432    0.46134356  1.1109686   0.19795308  1.7012955
  -0.3427637   2.103306   -0.23468001  1.2745342  -0.39175054  0.88707864
  -0.04341581]
 [-1.3139998  -1.2492635  -0.08355827 -0.09560532  0.18403004  1.220369
  -0.3065677   1.2500912  -0.23079728  0.7390443   1.6344454   0.9292701
  -0.38656047]
 [-0.4727013  -0.7012266  -0.40429202  0.30962136  0.4098145   0.96458966
  -0.05485974  0.8226754   0.55830073  0.8288073   0.6587547   0.10060805
  -0.21596901]
 [-0.515158   -1.5047289  -0.6789595   0.38771987  0.24995393  0.9599639
   0.0265132   0.97391677 -0.7468353  -0.06415401  1.3697305   1.8394302
   0.95870966]]
```

Τέλος χωρίζουμε το train_set σε train και validation set με αναλογία 80%-20% ορίζοντας την παράμετρο stratify ίση με το y_train ούτως ώστε να διατηρηθεί ίδιος ο αριθμός των διαφορετικών ψηφίων σε κάθε set.

Βήμα 10-13

Στα ακόλουθα βήματα προσπαθούμε να φτιάξουμε έναν ταξινομητή αναγνώρισης ψηφίων με GMM-HMM (Gaussian Mixture Models – Hidden Markov Models) χρησιμοποιώντας την βιβλιοθήκη pomegranate. Ένα διάνυσμα ακουστικών χαρακτηριστικών, όπως αυτό εξάγεται από την επεξεργασία ενός πλαισίου φωνής, αποτελεί μια πιθανή παρατήρηση σε κάποια κατάσταση. Λόγω του ότι είναι επιτρεπτές συνεχείς μεταβολές τέτοιων παρατηρήσεων, η πιθανότητα τους μοντελοποιείται με ένα μίγμα Γκαουσιανών κατανομών (GMM). Ο GMM παίζει πολύ σημαντικό ρόλο καθώς μοντελοποιεί τα δεδομένα μας με ένα μίγμα γκαουσιανών ,k στο πλήθος, εφαρμόζοντας ένα πρωταρχικό clustering μεταξύ ομοίων δειγμάτων. Από μαθηματικής άποψης κάθε Gaussian αποτελέι στην ουσία μία multivariate Gaussian κατανομή με παραμέτρους τον πίνακα μ της μέσης τιμής, Σ της διασποράς αλλά και μίας παραμέτρους π η οποία καλείται mixing propability και καθορίζει το πόσο μεγάλη θα είναι η Gaussian κατανομή .Συνεπώς οι παράμετροι του μοντέλου μας που πρέπει να βελτιστοποιήσουμε είναι οι **θ={π,μ,Σ}**. Από μαθηματικής άποψης η πιθανότητα ένα δείγμα να ανήκει σε μία κλάση δίνεται από την ακόλουθη σχέση:

$$p(z_k = 1|\mathbf{x}_n) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \Sigma_j)} :$$

Στην συνέχεια αφού μοντελοποιήσουμε τα δεδομένα μας με GMM εφαρμόζουμε το Hidden Markov Model το οποίο αποτελεί ένα ακολουθιακό μοντέλο πρόβλεψης πιθανοτήτων. Στην ουσία το HMM αποτελεί έναν γράφο όπου οι κόμβοι του είναι οι GMM που παράχθηκαν προηγουμένως και οι πιθανότητες μετάβασης από έναν κόμβο σε κάποιον άλλον ανήκουν στον transition matrix (τον οποίο αρχικοποιούμε τυχαία με απαραίτητη συνθήκη το άθροισμα κάθε γραμμής να ισούται με 1.0).Με χρήση αυτού του μοντέλου δεδομένης της ακολουθίας δεδομένων εισόδου μπορούμε να προβλέψουμε την ακολουθία των labels. Το μοντέλο μας είναι της μορφής left –right. Στην συνέχεια εκπαιδεύουμε τα 10 μοντέλα μας (ένα HMM για κάθε ψηφίο) με χρήση του αλγορίθμου Expectation Maximization ο οποίος είτε συγκλίνει είτε σταματάει μετά από καθορισμένο πλήθος επαναλήψεων (παραμετρος max_iteration).Τρέχουμε τον Viterbi αλγόριθμο και επιστρέφουμε το log-likelihood για κάθε HMM ενώ τελικά κρατάμε το HMM που μας έδωσε τηη μέγιστη τιμή του log-likelihood. Έτσι γίνεται και η πρόβλεψη κλάσης εφόσον δεν πρέπει να ξεχνάμε πώς κάθε HMM συνδέεται με ένα μοναδικό ψηφίο.

Στην συνέχεια βελτιστοποιούμε τις υπερπαραμέτρους του μοντέλου μας : τις καταστάσεις HMM(n_states),το πλήθος των Gaussian(n_mixtures) και το πλήθος των μέγιστων επαναλήψεων για σύγκλιση του HMM μοντέλου (max_iterations) και χρησιμοποιούμε το validation set για τον υπολογισμό του score. Γιατί όμως χρησιμοποιούμε το validation_set για βελτιστοποίηση των υπεπαραμέτρων; Η κύρια ιδέα του διαχωρισμού του συνόλου δεδομένων σε ένα σύνολο επικύρωσης είναι να αποτραπεί το overfitting του μοντέλου μας, δηλαδή, το μοντέλο γίνεται πολύ καλό στην ταξινόμηση των δειγμάτων στο σετ εκπαίδευσης, αλλά δεν μπορεί να γενικεύσει και να κάνει ακριβείς ταξινομήσεις στα δεδομένα που δεν έχει δει πριν. Με άλλα λόγια αν δεν χρησιμοποιήσουμε το validation set υπάρχει κίνδυνος το υψηλό score στο training να αντιστοιχεί σε χαμηλό score στο test set.

Τα αποτελέσματά μας φαίνονται ακολούθως:

Αρχικά παρουσιάζουμε την κλάση του μοντέλου GMM-HMM:

```
class HMM(BaseEstimator, ClassifierMixin):
    # Hidden Markov Model using Gaussian Mixtures Model
    # Labels are supposed to be in range[0, len(n_labels)]
    def __init__(self):
        self.hmm=[]

    def fit(self, X_train,y_train,n_states,n_mixtures,max_iterations):
        digits_train_3d=[]
        digits_train_2d=[]
        for i in range(10):
            help_array=[]
            for j in range(len(X_train)):
                if y_train[j]==i :
                   X_help=X_train[j].real
                   X=X_help.astype('float64')
                   help_array.append(X)
            digits_train_3d.append(help_array)
            digits_train_2d.append(np.vstack(help_array))

        for i in range(10):
            # for each digit 0-9

            X = digits_train_2d[i] # data from a single digit (can be a numpy array)

            gmm = True # whether to use GMM or plain Gaussian
            dists = [] # list of probability distributions for the HMM states
            for j in range(n_states):
                if gmm:
                    a = GeneralMixtureModel.from_samples(MultivariateGaussianDistribution, n_mixtures, X)
                else:
                    a = MultivariateGaussianDistribution.from_samples(X)
                dists.append(a)

            # Transition probabilities
            trans_mat = np.diag(np.ones(n_states), k=0)*0.5 + np.diag(np.ones(n_states-1), k=1)*.5  # transition matrix
            trans_mat[-1,-1] = 1 #last state

            starts = [1] + ((n_states-1)*[0]) # starting probability matrix
            ends = starts[::-1] # ending probability matrix

            # Define the GMM-HMM
            self.hmm.append(HiddenMarkovModel.from_matrix(transition_probabilities=trans_mat, distributions=dists,
                                                starts=starts, ends=ends,
                                                state_names=['s{}'.format(i) for i in range(n_states)]))
        print(len(self.hmm))

        for i in range(10):
            # Fit the model
            self.hmm[i].fit(digits_train_3d[i], max_iterations=max_iterations, algorithm='baum-welch')

        return self
```

```
    def predict(self, X_test):
        self.y_pred = []
        for sample in X_test:
            predictions=[]
            for i in range(len(self.hmm)):
                # Run viterbi algorithm and return log-probability
                logp,_= self.hmm[i].viterbi(sample)
                predictions.append(logp)
            # Store the index of the maximum log-probability
            self.y_pred.append(np.argmax(predictions))
        return self.y_pred
    def score(self,X_test,y_test):
        self.predict(X_test)
        y_test=np.asarray(y_test)
        cm=confusion_matrix(y_test,self.y_pred)
        classes=np.unique(y_test)
        plot_confusion_matrix(cm,classes)
        plt.show()
        print(classification_report(y_test,self.y_pred))
        return np.sum(self.y_pred == y_test) / y_test.shape[0]
```

Στην συνέχεια όπως φαίνεται και στον κώδικα για κάθε δοκιμή στις παραμέτρους εκτυπώνουμε τόσο το confusion_matrix όσο και το classification report για να ελέγχουμε την επίδοση του μοντέλου. Τελικά βέλτιστη επίδοση τόσο σε accuracy_score όσο και σε f1_score δίνουν οι ακόλουθες υπερπαράμετροι:

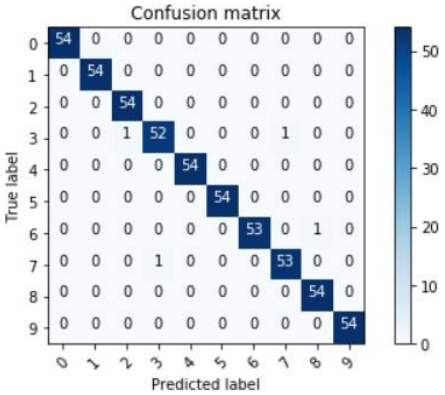
N_states=4

N_mixtures=5

Max_iterations=15

Τώρα για τις βέλτιστες υπεπαραμέτρους τυπώνουμε την επίδοση τόσο για το validation_set όσο και για το test_set.

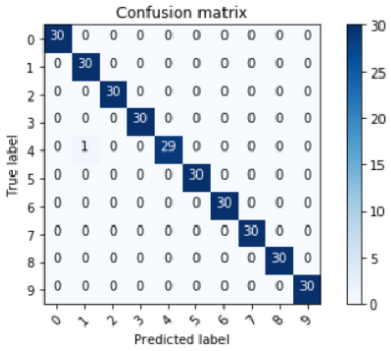
Validation set



	precision	recall	f1-score	support
0	1.00	1.00	1.00	54
1	1.00	1.00	1.00	54
2	0.98	1.00	0.99	54
3	0.98	0.96	0.97	54
4	1.00	1.00	1.00	54
5	1.00	1.00	1.00	54
6	1.00	0.98	0.99	54
7	0.98	0.98	0.98	54
8	0.98	1.00	0.99	54
9	1.00	1.00	1.00	54
accuracy			0.99	540
macro avg	0.99	0.99	0.99	540
weighted avg	0.99	0.99	0.99	540

Accuracy_score= 0.9925925925925926

Test set



	precision	recall	f1-score	support
0	1.00	1.00	1.00	30
1	0.97	1.00	0.98	30
2	1.00	1.00	1.00	30
3	1.00	1.00	1.00	30
4	1.00	0.97	0.98	30
5	1.00	1.00	1.00	30
6	1.00	1.00	1.00	30
7	1.00	1.00	1.00	30
8	1.00	1.00	1.00	30
9	1.00	1.00	1.00	30
accuracy			1.00	300
macro avg	1.00	1.00	1.00	300
weighted avg	1.00	1.00	1.00	300

Accuracy_score= 0.9966666666666667

Βλέπουμε λοιπόν συνολικά πως έχουμε πολύ καλή επίδοση τόσο στο validation set όσο και στο test set και επιβεβαιώνει το γεγονός πως δεν έχουμε επιβαρυνθεί από κάποιο πρόβλημα overfitting.

Βήμα 14

Στο βήμα αυτό παρουσιάζουμε διάφορες υλοποιήσεις του LSTM δικτύου ξεκινώντας από το απλό και στη συνέχεια προσθέτουμε διάφορα νέα χαρακτηριστικά. Πριν το απλό LSTM ακολουθούμε ένα zero padding στα δεδομένα μας ώστε να τα φέρουμε στην κατάλληλη μορφή για να τα περάσουμε στο μοντέλο μας. Στη συνέχεια αρχικοποιούμε ότι χρειαζόμαστε για τη υλοποίηση του μοντέλου μας. Τέλος παίρνουμε όλες τις εξόδους, κρατάμε την τελευταία και την περνάμε εκ νέου από το δίκτυο. Έτσι λοιπόν δημιουργούμε την κλάση μας παντα με βάση το δοσμένο κώδικα. Για να την καλέσουμε επιλέγουμε τις κατάλληλες τιμές, το κριτήριο του σφάλματος καθώς και τον optimizer με τα οποία κατλήγουμε στο κατώθι αποτέλεσμα.

```
Epoch [1/5], Step [1/135], Loss: 2.3017
Epoch [1/5], Step [9/135], Loss: 2.2986
Epoch [1/5], Step [17/135], Loss: 2.2561
Epoch [1/5], Step [25/135], Loss: 2.2561
Epoch [1/5], Step [33/135], Loss: 2.1637
Epoch [1/5], Step [41/135], Loss: 2.0634
Epoch [1/5], Step [49/135], Loss: 1.9399
Epoch [1/5], Step [57/135], Loss: 1.8848
Epoch [1/5], Step [65/135], Loss: 1.8558
Epoch [1/5], Step [73/135], Loss: 1.6469
Epoch [1/5], Step [81/135], Loss: 1.7091
Epoch [1/5], Step [89/135], Loss: 1.6157
Epoch [1/5], Step [97/135], Loss: 1.4020
Epoch [1/5], Step [105/135], Loss: 1.4257
Epoch [1/5], Step [113/135], Loss: 1.2666
Epoch [1/5], Step [121/135], Loss: 1.2428
Epoch [1/5], Step [129/135], Loss: 0.9948
Validation Accuracy = 68.33333333333333 %
Epoch [2/5], Step [1/135], Loss: 0.9009
Epoch [2/5], Step [9/135], Loss: 1.2949
Epoch [2/5], Step [17/135], Loss: 0.8180
Epoch [2/5], Step [25/135], Loss: 1.0662
Epoch [2/5], Step [33/135], Loss: 0.5775
Epoch [2/5], Step [41/135], Loss: 0.7373
Epoch [2/5], Step [49/135], Loss: 0.8923
Epoch [2/5], Step [57/135], Loss: 0.6675
Epoch [2/5], Step [65/135], Loss: 0.6805
Epoch [2/5], Step [73/135], Loss: 0.7352
Epoch [2/5], Step [81/135], Loss: 0.4720
Epoch [2/5], Step [89/135], Loss: 0.6900
Epoch [2/5], Step [97/135], Loss: 1.0566
Epoch [2/5], Step [89/135], Loss: 0.6900
Epoch [2/5], Step [97/135], Loss: 1.0566
Epoch [2/5], Step [105/135], Loss: 0.9050
Epoch [2/5], Step [113/135], Loss: 0.6702
Epoch [2/5], Step [121/135], Loss: 0.7090
Epoch [2/5], Step [129/135], Loss: 0.6705
Validation Accuracy = 82.0 %
Epoch [3/5], Step [1/135], Loss: 0.3415
Epoch [3/5], Step [9/135], Loss: 0.8993
Epoch [3/5], Step [17/135], Loss: 0.4751
Epoch [3/5], Step [25/135], Loss: 0.3985
Epoch [3/5], Step [33/135], Loss: 0.7189
Epoch [3/5], Step [41/135], Loss: 0.4703
Epoch [3/5], Step [49/135], Loss: 0.6184
Epoch [3/5], Step [57/135], Loss: 0.5873
Epoch [3/5], Step [65/135], Loss: 0.2154
Epoch [3/5], Step [73/135], Loss: 1.0859
Epoch [3/5], Step [81/135], Loss: 0.2711
Epoch [3/5], Step [89/135], Loss: 0.2718
Epoch [3/5], Step [97/135], Loss: 0.2964
Epoch [3/5], Step [105/135], Loss: 0.2129
Epoch [3/5], Step [113/135], Loss: 0.3423
Epoch [3/5], Step [121/135], Loss: 0.4632
Epoch [3/5], Step [129/135], Loss: 0.6959
Validation Accuracy = 84.66666666666667 %
Epoch [4/5], Step [1/135], Loss: 0.1747
Epoch [4/5], Step [9/135], Loss: 0.2982
Epoch [4/5], Step [17/135], Loss: 0.3904
Epoch [4/5], Step [25/135], Loss: 0.2476
Epoch [4/5], Step [33/135], Loss: 0.3851
Epoch [4/5], Step [41/135], Loss: 0.3053
Epoch [4/5], Step [49/135], Loss: 0.4402
Epoch [4/5], Step [57/135], Loss: 0.1569
```

Όπως παρατηρούμε όσο περισσότερες εποχές τρέξουμε τόσο καλύτερο είναι το accuracy και ταυτόχρονα το λάθος μειώνεται. Για το επόμενο ερώτημα εκτελούμε

κατά την εκπαίδευση Dropout και L2 Regularization. Το πρώτο αφαιρεί τη λειτουργία κάποιων νευρώνων τυχαία με αποτέλεσμα να μην έχουμε κάποιες ενημερώσεις στα βάρη. Όταν συμβαίνει αυτό άλλοι νευρώνες πρέπει να κάνουν τις προβλέψεις για αυτούς που παραλείψαμε. Έτσι πετυχαίνουμε ένα δίκτυο λιγότερο ευαίσθητο στα ειδικά βάρη των νευρώνων δηλαδή ένα δίκτυο που ανταποκρίνεται καλύτερα στις γενικεύσεις. Το L2 Regularization βελτιστοποιεί το μέσο κόστος, το οποίο είναι συχνά μέτρο της απόδοσης. Επίσης μας βοηθά σε περιπτώσεις που οι τιμές που έχουμε δεν είναι ακραίες ενώ ακόμα σχεδόν πάντα μας δίνει μοναδική λύση. Τα αποτελέσματα μας βέβαια δεν είναι αρκετά καλύτερα αλλά παρουσιάζουν μια ελάχιστη βελτίωση όπως φαίνεται παρακάτω.

```
Epoch [1/5], Step [1/135], Loss: 2.3031
Epoch [1/5], Step [9/135], Loss: 2.2690
Epoch [1/5], Step [17/135], Loss: 2.2693
Epoch [1/5], Step [25/135], Loss: 2.2201
Epoch [1/5], Step [33/135], Loss: 2.1208
Epoch [1/5], Step [41/135], Loss: 1.9915
Epoch [1/5], Step [49/135], Loss: 1.9363
Epoch [1/5], Step [57/135], Loss: 1.9397
Epoch [1/5], Step [65/135], Loss: 1.6704
Epoch [1/5], Step [73/135], Loss: 1.6002
Epoch [1/5], Step [81/135], Loss: 1.6422
Epoch [1/5], Step [89/135], Loss: 1.3516
Epoch [1/5], Step [97/135], Loss: 1.2140
Epoch [1/5], Step [105/135], Loss: 1.2646
Epoch [1/5], Step [113/135], Loss: 1.2079
Epoch [1/5], Step [121/135], Loss: 0.9967
Epoch [1/5], Step [129/135], Loss: 0.8450
Validation Accuracy = 72.0 %
Epoch [2/5], Step [1/135], Loss: 0.9917
Epoch [2/5], Step [9/135], Loss: 0.7038
Epoch [2/5], Step [17/135], Loss: 0.6008
Epoch [2/5], Step [25/135], Loss: 0.7111
Epoch [2/5], Step [33/135], Loss: 0.9265
Epoch [2/5], Step [41/135], Loss: 0.8721
Epoch [2/5], Step [49/135], Loss: 0.7638
Epoch [2/5], Step [57/135], Loss: 0.5611
Epoch [2/5], Step [65/135], Loss: 0.5248
Epoch [2/5], Step [73/135], Loss: 0.3494
Epoch [2/5], Step [81/135], Loss: 0.3819
```

```
Epoch [2/5], Step [1/135], Loss: 0.9917
Epoch [2/5], Step [9/135], Loss: 0.7038
Epoch [2/5], Step [17/135], Loss: 0.6008
Epoch [2/5], Step [25/135], Loss: 0.7111
Epoch [2/5], Step [33/135], Loss: 0.9265
Epoch [2/5], Step [41/135], Loss: 0.8721
Epoch [2/5], Step [49/135], Loss: 0.7638
Epoch [2/5], Step [57/135], Loss: 0.5611
Epoch [2/5], Step [65/135], Loss: 0.5248
Epoch [2/5], Step [73/135], Loss: 0.3494
Epoch [2/5], Step [81/135], Loss: 0.3819
Epoch [2/5], Step [89/135], Loss: 0.8029
Epoch [2/5], Step [97/135], Loss: 0.7044
Epoch [2/5], Step [105/135], Loss: 1.0446
Epoch [2/5], Step [113/135], Loss: 0.7660
Epoch [2/5], Step [121/135], Loss: 0.3950
Epoch [2/5], Step [129/135], Loss: 0.5718
Validation Accuracy = 83.33333333333333 %
Epoch [3/5], Step [1/135], Loss: 0.5018
Epoch [3/5], Step [9/135], Loss: 0.2362
Epoch [3/5], Step [17/135], Loss: 0.7123
Epoch [3/5], Step [25/135], Loss: 0.3272
Epoch [3/5], Step [33/135], Loss: 0.4342
Epoch [3/5], Step [41/135], Loss: 0.2973
Epoch [3/5], Step [49/135], Loss: 0.2860
Epoch [3/5], Step [57/135], Loss: 0.4606
Epoch [3/5], Step [65/135], Loss: 0.4195
Epoch [3/5], Step [73/135], Loss: 0.5182
Epoch [3/5], Step [81/135], Loss: 0.1752
Epoch [3/5], Step [89/135], Loss: 0.3790
Epoch [3/5], Step [97/135], Loss: 0.5051
```

Το bidirectional μας δίνει την ευκαιρία να λάβουμε υπόψιν εξαρτήσεις των δεδομένων οι οποίες υπάρχουν στην αντίθετη κατεύθυνση και δε θα τις βρίσκαμε αποθηκεύοντας μόνο πληροφορία για προηγούμενα κομμάτια της εισόδου. Με το early stopping επιτυγχάνουμε να αποφύγουμε συνέπειες από τις πολλές εποχές, δηλαδή κάποια over/underfitting. Τελικά με αυτό το συνδυασμό έχουμε διαθέσιμες μερικές extra πληροφορίες τις οποίες σε άλλη περίπτωση δε θα είχαμε. Παρακάτω κάποια εικόνα από τα αποτελέσματα.

```
Epoch [1/5], Step [1/135], Loss: 0.3131
Epoch [1/5], Step [5/135], Loss: 0.2517
Epoch [1/5], Step [9/135], Loss: 0.2621
Epoch [1/5], Step [13/135], Loss: 0.2480
Epoch [1/5], Step [17/135], Loss: 0.3127
Epoch [1/5], Step [21/135], Loss: 0.1455
Epoch [1/5], Step [25/135], Loss: 0.1959
Epoch [1/5], Step [29/135], Loss: 0.3377
Epoch [1/5], Step [33/135], Loss: 0.1684
Epoch [1/5], Step [37/135], Loss: 0.1521
Epoch [1/5], Step [41/135], Loss: 0.2050
Epoch [1/5], Step [45/135], Loss: 0.2192
Epoch [1/5], Step [49/135], Loss: 0.3389
Epoch [1/5], Step [53/135], Loss: 0.0432
Epoch [1/5], Step [57/135], Loss: 0.2437
Epoch [1/5], Step [61/135], Loss: 0.2400
Epoch [1/5], Step [65/135], Loss: 0.3108
Epoch [1/5], Step [69/135], Loss: 0.7155
Epoch [1/5], Step [73/135], Loss: 0.3059
Epoch [1/5], Step [77/135], Loss: 0.3319
Epoch [1/5], Step [81/135], Loss: 0.2556
Epoch [1/5], Step [85/135], Loss: 0.3100
Epoch [1/5], Step [89/135], Loss: 0.0441
Epoch [1/5], Step [93/135], Loss: 0.4854
Epoch [1/5], Step [97/135], Loss: 0.0203
Epoch [1/5], Step [101/135], Loss: 0.1556
Epoch [1/5], Step [105/135], Loss: 0.2593
Epoch [1/5], Step [109/135], Loss: 0.2288
Epoch [1/5], Step [113/135], Loss: 0.2756
Epoch [1/5], Step [117/135], Loss: 0.3503
Epoch [1/5], Step [121/135], Loss: 0.3352
Epoch [1/5], Step [125/135], Loss: 0.1539
```

```
Epoch [2/5], Step [113/135], Loss: 0.1149
Epoch [2/5], Step [117/135], Loss: 0.2500
Epoch [2/5], Step [121/135], Loss: 0.4254
Epoch [2/5], Step [125/135], Loss: 0.2307
Epoch [2/5], Step [129/135], Loss: 0.1577
Epoch [2/5], Step [133/135], Loss: 0.1928
Epoch [3/5], Step [1/135], Loss: 0.1166
Epoch [3/5], Step [5/135], Loss: 0.2370
Epoch [3/5], Step [9/135], Loss: 0.3476
Epoch [3/5], Step [13/135], Loss: 0.1325
Epoch [3/5], Step [17/135], Loss: 0.1562
Epoch [3/5], Step [21/135], Loss: 0.1919
Epoch [3/5], Step [25/135], Loss: 0.1596
Epoch [3/5], Step [29/135], Loss: 0.1826
Epoch [3/5], Step [33/135], Loss: 0.0417
Epoch [3/5], Step [37/135], Loss: 0.0805
Epoch [3/5], Step [41/135], Loss: 0.0331
Epoch [3/5], Step [45/135], Loss: 0.2020
Epoch [3/5], Step [49/135], Loss: 0.2779
Epoch [3/5], Step [53/135], Loss: 0.0668
Epoch [3/5], Step [57/135], Loss: 0.4272
Epoch [3/5], Step [61/135], Loss: 0.1482
Epoch [3/5], Step [65/135], Loss: 0.1272
Epoch [3/5], Step [69/135], Loss: 0.2019
Epoch [3/5], Step [73/135], Loss: 0.1628
Epoch [3/5], Step [77/135], Loss: 0.1356
Epoch [3/5], Step [81/135], Loss: 0.2969
Epoch [3/5], Step [85/135], Loss: 0.1637
Epoch [3/5], Step [89/135], Loss: 0.7009
```