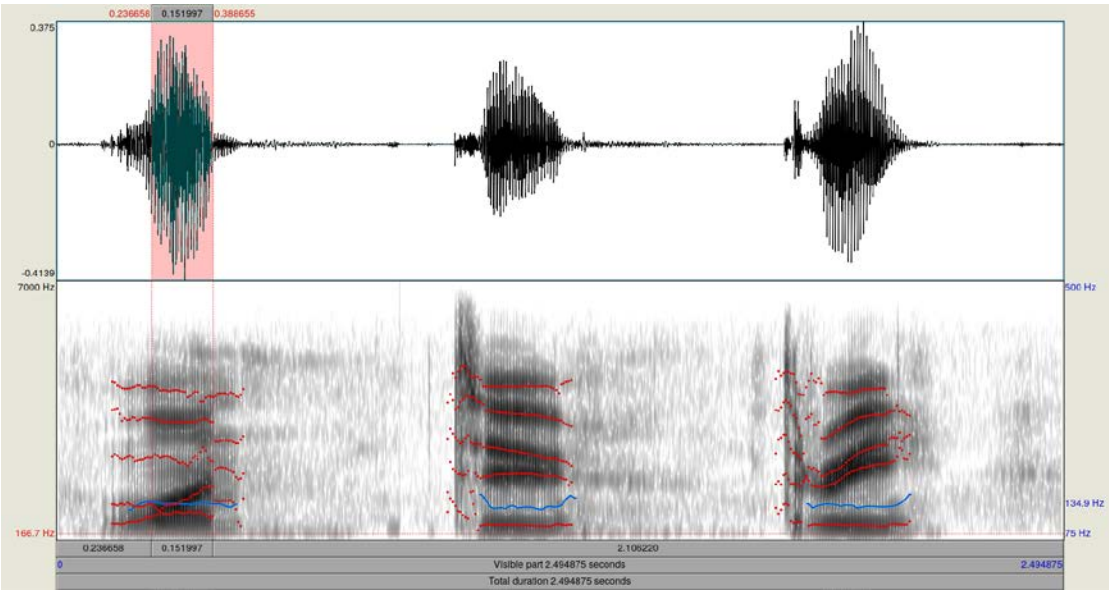
2η Εργαστηριακή Άσκηση Αναγνώρισης Προτύπων

Η εργασία αυτή εκπονήθηκε από τους φοιτητές της ΣΗΜΜΥ Κωνσταντίνο Τσαούση και Κωνσταντίνο Κωστόπουλο με ΑΜ 03117652 και 03117043 αντίστοιχα.

Βήμα 1

Onetwothree1.wav (wavelet & spectogram).



Οι μπλε περιοχές αποτελούν τα pitches και οι κόκκινες τα formants.

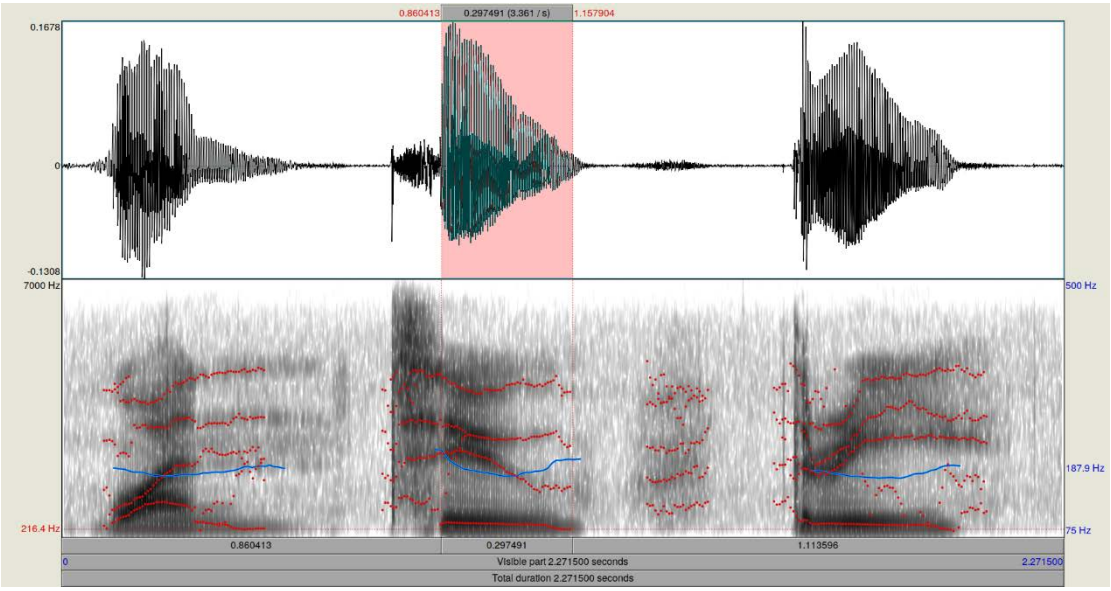
Μέση τιμή του pitch για το φωνήεν "α": 134.58423258658706 Hz
Μέση τιμή του pitch για το "ου": 129.69159861988481 Hz
Μέση τιμή του pitch για το φωνήεν "ι": 133.1886601757627 Hz

1ο formant για το φωνήεν "α": 849.6634778225167 Hz
2ο formant για το φωνήεν "α": 1558.8132522983956 Hz
3ο formant για το φωνήεν "α": 2454.8855903461977 Hz

1ο formant για το "ου": 326.28638769668464 Hz
2ο formant για το "ου": 1796.5397160157636 Hz
3ο formant για το "ου": 2303.957685226183 Hz

1ο formant για το φωνήεν "ι": 415.6614083216102 Hz
2ο formant για το φωνήεν "ι": 2051.4894089209574 Hz
3ο formant για το φωνήεν "ι": 2558.225806660655 Hz

Onetothree8.wav (wavelet & spectogram).



Οι μπλε περιοχές αποτελούν τα pitches και οι κόκκινες τα formants.

Μέση τιμή του pitch για το φωνήεν "α": 177.5997339299439 Hz
Μέση τιμή του pitch για το "ου": 179.9200427768357 Hz
Μέση τιμή του pitch για το φωνήεν "ι": 174.49946864298107 Hz

1ο formant για το φωνήεν "α": 745.6559370900545 Hz
2ο formant για το φωνήεν "α": 1967.5583632156558 Hz
3ο formant για το φωνήεν "α": 3141.621626187917 Hz

1ο formant για το "ου": 355.5379700424874 Hz
2ο formant για το "ου": 1832.7173958068765 Hz
3ο formant για το "ου": 2617.6940698972203 Hz

1ο formant για το φωνήεν "ι": 336.9624550482762 Hz
2ο formant για το φωνήεν "ι": 2038.444977373537 Hz
3ο formant για το φωνήεν "ι": 2871.3654971785745 Hz

Ο τόνος του άνδρα ομιλητή είναι χαμηλότερος από αυτόν της γυναίκας για όλα τα φωνήεντα. Αυτό μπορεί επίσης να συναχθεί από τον υψηλότερο ρυθμό ταλάντωσης στην κυματομορφή της γυναίκας ομιλήτριας. Στις περισσότερες περιπτώσεις, η διαφορά μεταξύ των formants των 2 ομιλητών αυξάνεται με το index του formant. Το δεύτερο και το τρίτο formants της γυναίκας ομιλήτριας είναι υψηλότερα από τα αντίστοιχα των ανδρών. Τα τμήματα που αντιστοιχούν σε κάθε φωνήεν δεν επιλέχθηκαν με συστηματικό τρόπο. Τόσο ο τόνος όσο και τα formants εξαρτώνται από υπολογισμούς οι οποίοι με τη σειρά τους εξαρτώνται από παραμέτρους που δεν είχαν ρυθμιστεί (πέρα από τα default του praat). Συνεπώς, τα αποτελέσματα θα πρέπει να λαμβάνονται με επιφύλαξη.

Βήμα 2

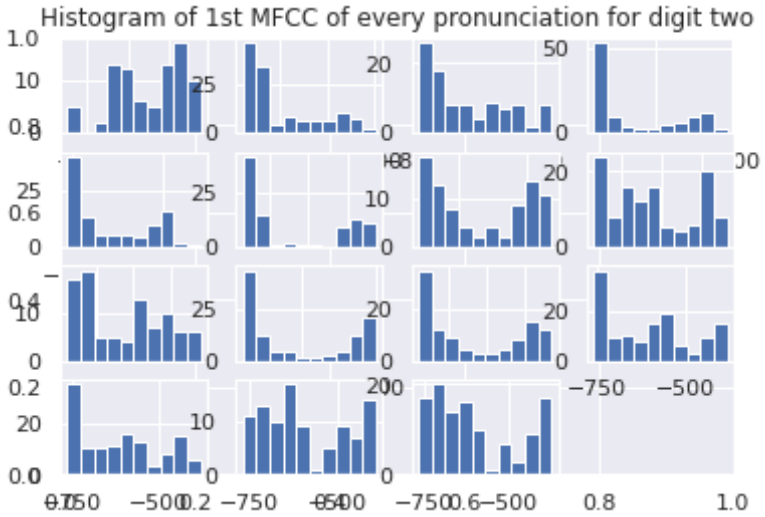
Προκειμένου να διαβάσουμε τα αρχεία, φτιάχνουμε τη συνάρτηση dataparser(). Η ανάγνωση των αρχείων πραγματοποιήθηκε με την εντολή google.colab.files.upload και με τη βοήθεια της librosa. Έπειτα, με κατάλληλα split μένουμε με αρχεία της μορφής *"two15"* και επομένως μένει να χωρίσουμε τους αριθμούς από τα γράμματα. Ορίζουμε την μεταβλητή *re.findall(r'(\w+?)(\d+)'*, και κάνοντας match με το πρώτο group για τα γράμματα, και με το δεύτερο group για τους αριθμούς.

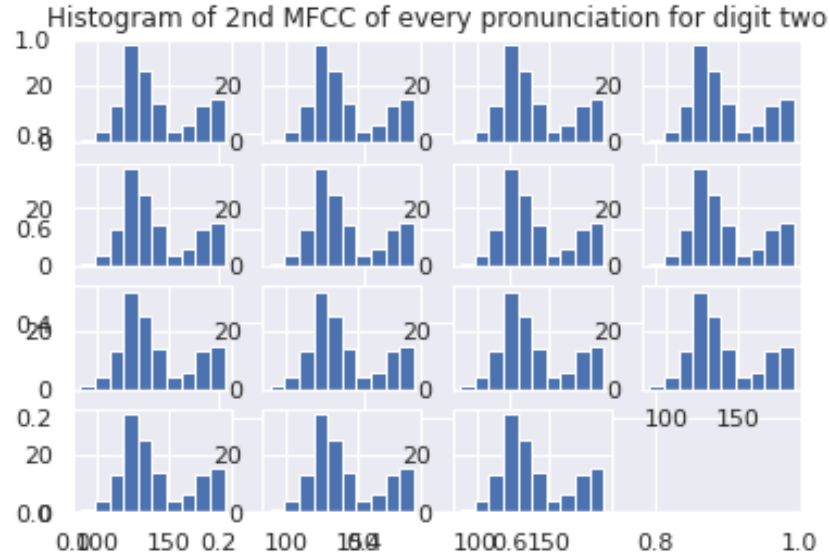
Βήμα 3

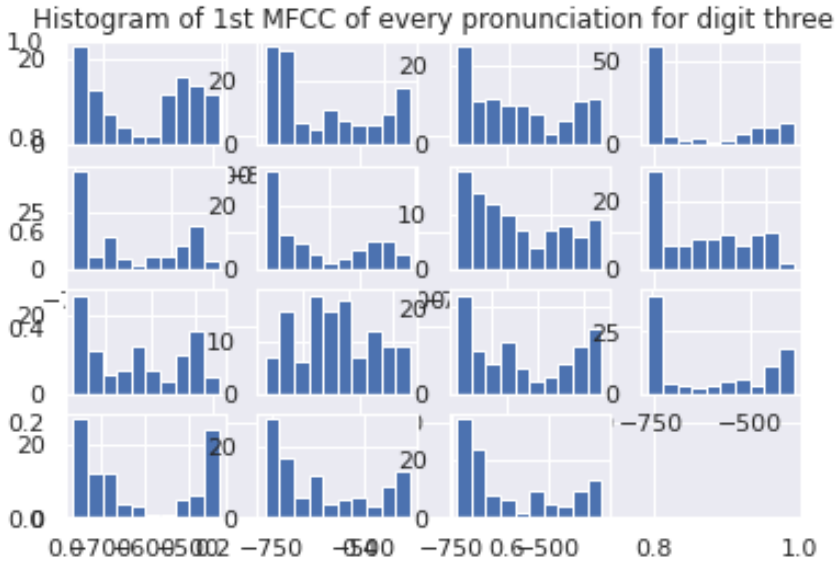
Εδώ, εξάγουμε για κάθε αρχείο ήχου τους συντελεστές MFCC (Mel-Frequency Cepstral Coefficients). Εμείς χρησιμοποιούμε μόνο τους 13 πρώτους συντελεστές. Από αυτούς εξάγουμε τις τοπικές παραγώγους πρώτης και δεύτερης τάξης (delta και delta-delta). Έτσι δημιουργούμε τρεις λίστες με τα παραπάνω στοιχεία. Προκειμένου να το επιτύχουμε αυτό, χρησιμοποιήσαμε τις συναρτήσεις feature.mfcc και feature.delta από τη βιβλιοθήκη librosa. Τα MFCC χρησιμοποιούνται πολύ συχνά ως χαρακτηριστικά στην αναγνώριση ομιλίας και έχουν αποδειχθεί ότι είναι πολύ αξιόπιστα. Προκειμένου να τα υπολογίσουμε, πρέπει να υποθέσουμε ότι σε μικρές χρονικές κλίμακες το σήμα δεν αλλάζει (και έτσι μπορούμε να το πλαισιώσουμε). Στη συνέχεια υπολογίζεται το φάσμα ισχύος κάθε πλαισίου, οι τιμές του οποίου συνδέονται με τη λειτουργία του ανθρώπινου κοχλία. Στη συνέχεια, εφαρμόζεται το mel filterbank και αθροίζουμε την ενέργεια σε κάθε φίλτρο, προκειμένου να απομονώσουμε μόνο την απαραίτητη πληροφορία. Στη συνέχεια, λαμβάνουμε τον λογάριθμο των ενεργειών του filterbank, μια ενέργεια που παρομοιάζει αυτή ανθρώπινου αυτιού, δεδομένου ότι αντιλαμβανόμαστε την ένταση σε λογαριθμική κλίμακα. Τέλος, υπολογίζεται η DCT η οποία αποδιαρθρώνει τις ενέργειες, και κρατάμε τις πρώτες επειδή οι υπόλοιπες αντιπροσωπεύουν την ταχύτερη αλλαγές στις filterbank ενέργειες, οι οποίες έχει αποδειχθεί ότι υποβαθμίζουν την απόδοση ταξινόμησης.

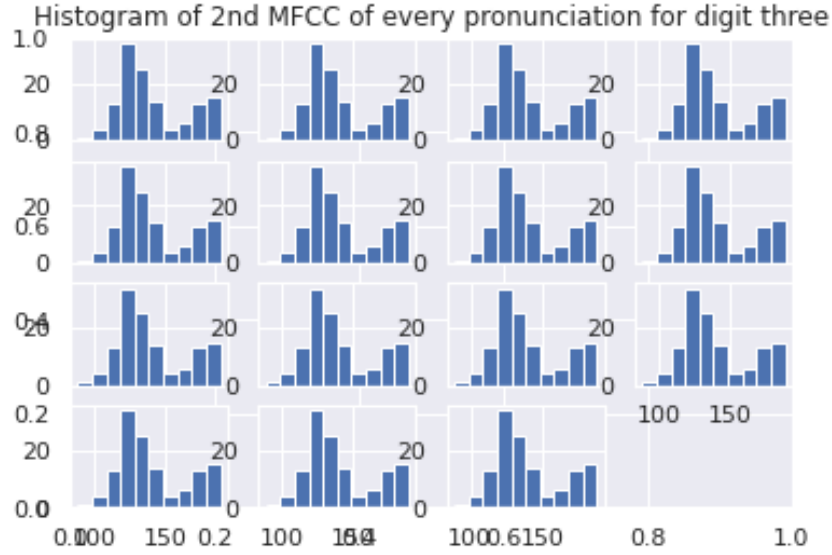
Βήμα 4

Λόγω των ΑΜ μας, έχουμε n1 = 3, n2 = 2. Παρακάτω μπορούμε να δούμε τα ιστογράμματα για το πρώτο και το δεύτερο MFCC για κάθε προφορά για το n1 και n2.




Histogram of 2nd MFCC of every pronunciation for digit two

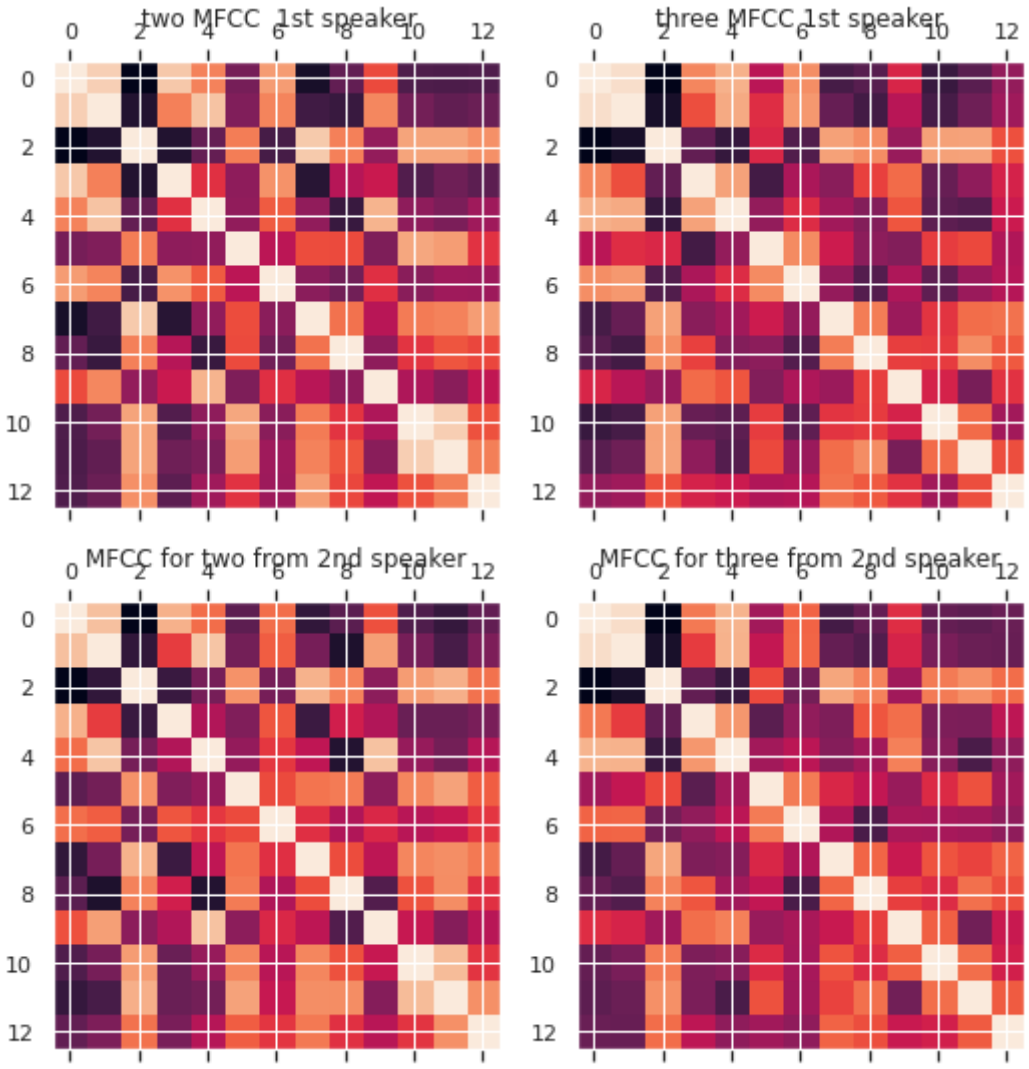

Histogram of 1st MFCC of every pronunciation for digit three


Histogram of 2nd MFCC of every pronunciation for digit three

Παρατηρώντας όλα τα ιστογράμματα βλέπουμε ότι οι ίδιοι συντελεστές σε διαφορετικές εκφωνήσεις του ίδιου ψηφίου έχουν παρόμοια κατανομή τιμών, οι μεγαλύτερες συχνότητες εμφάνισης εντοπίζονται στις ακραίες τιμές των συντελεστών και οι διαφορετικοί συντελεστές στην ίδια εκφώνηση έχουν κατανομές σε διαφορετικά διαστήματα.

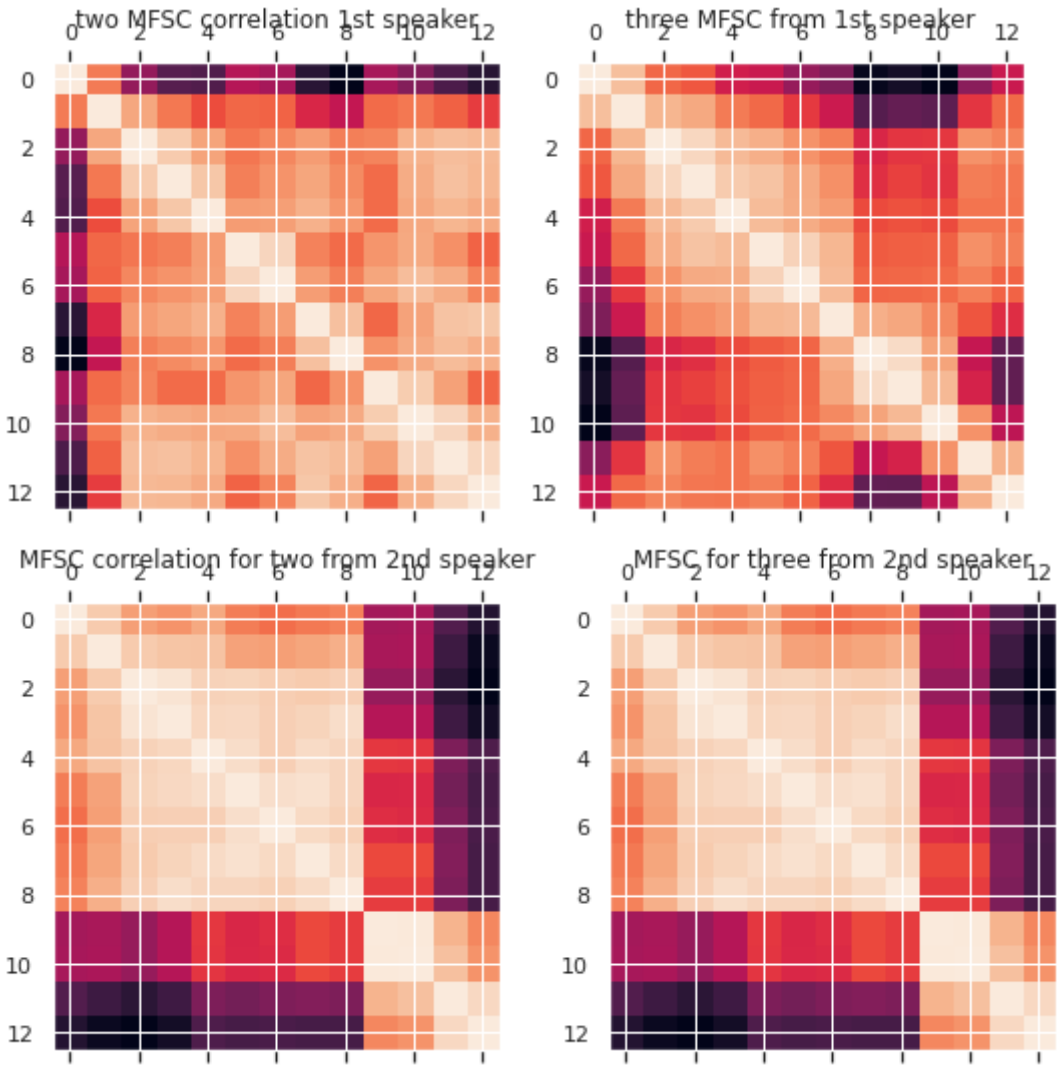
Για τον υπολογισμό του πίνακα συσχέτισης των MFSCs και των MFCCs χρησιμοποιούμε τη συνάρτηση corr() της pandas.DataFrame. Οι ιδανικές τιμές του πίνακα συσχέτισης είναι σε κάθε σημείο του πινάκα 0 εκτός από την διαγώνιο, όπου θα έχει τιμή 1. Αυτό είναι λογικό αφού επιθυμούμε τα χαρακτηριστηκά να είναι ασυσχέτιστα μεταξύ τους και επομένως να προσδίδουν πιο χρήσιμες πληροφορίες στη μάθηση.

MFCC:



Είναι σαφές ότι, εκτός από τη διαγώνιο, η συσχέτιση μεταξύ των συντελεστών είναι ως επί το πλείστον χαμηλή. Για να επιτευχθεί αυτό με τους MFCC, ένα βασικό βήμα είναι ο DCT (διακριτός μετασχηματισμός συνημιτόνου) των mel λογαριθμικών δυνάμεων, όπως επίσης αναφέρθηκε προηγουμένως. Για να το αποδείξουμε αυτό, δημιουργήσαμε τον πίνακα συσχέτισης των MFSC's (Mel Filterbank Spectral Coefficients), οι οποίοι παράγονται παρόμοια με τους MFCC, αλλά χωρίς DCT.

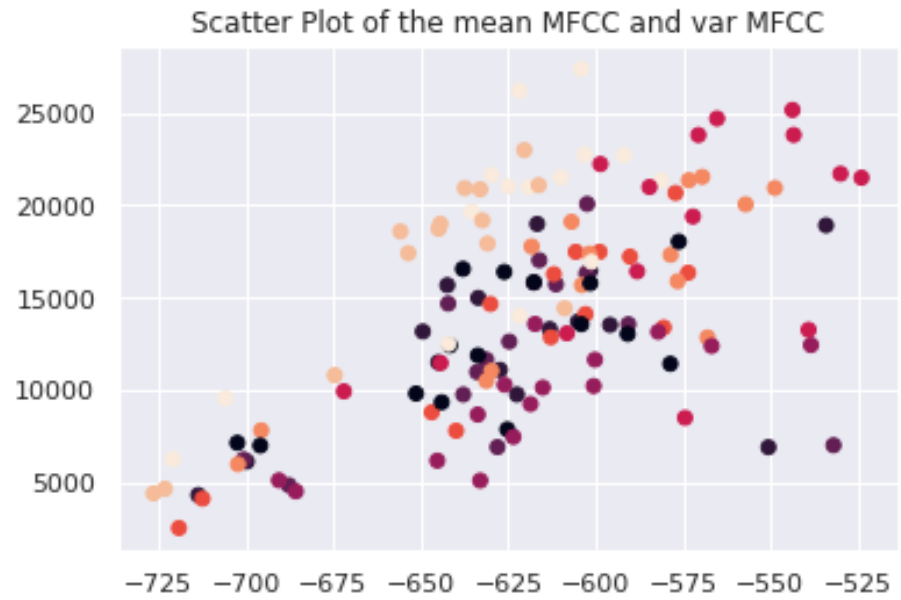
MFSC:



Είναι προφανές ότι μεταξύ πολλών συντελεστών παρατηρείται υψηλή συσχέτιση, η οποία δεν είναι κατάλληλη για τον σκοπό μας. Αυτό ήταν ένα αναμενόμενο αποτέλεσμα επειδή οι melbanks όλες επικαλύπτονται, παράγοντας συσχετιζόμενα αποτελέσματα. Ως εκ τούτου, προτιμώνται τα **MFCC**.

Βήμα 5

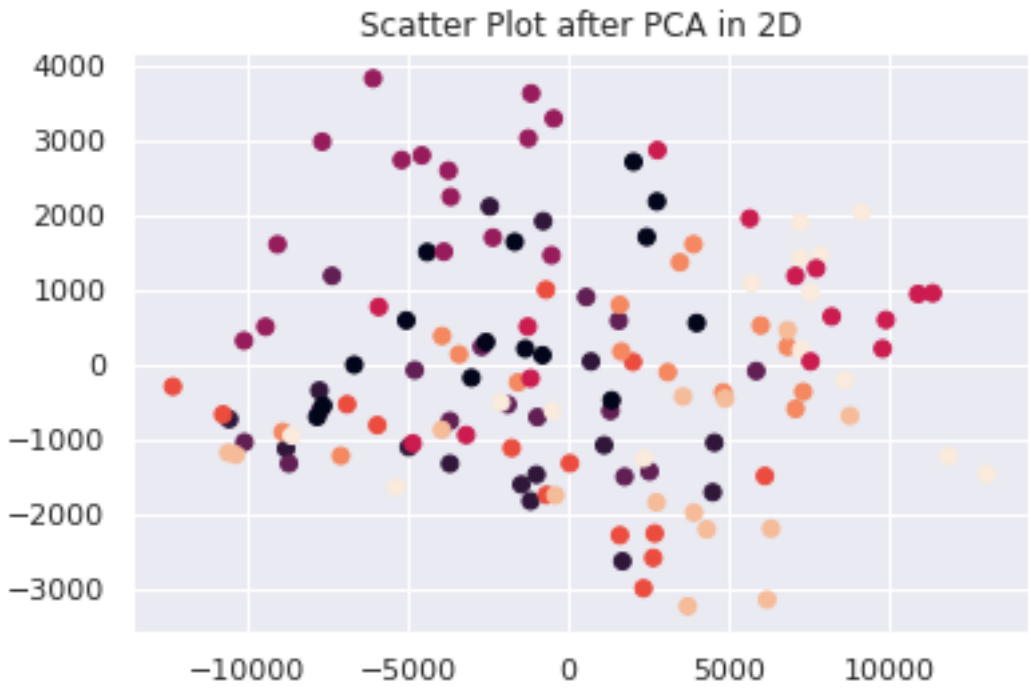
Δημιουργήσαμε έναν πίνακα, στον οποίο, για κάθε ένα από τα 133 αρχεία ήχου και για κάθε ένα από τα 13 συστατικά, αποθηκεύσαμε τους μέσους όρους και τις αποκλίσεις MFCC, τα deltas και τα deltas-deltas (6*13 τιμές για για κάθε αρχείο). Αρχικά, δημιουργήσαμε ένα 2D scatter plot των 2 πρώτων διαστάσεων αυτού του πίνακα (ο μέσος όρος των MFCC και των διαφορών των MFCC) απεικονίζοντας τα διαφορετικά ψηφία που προφέρονται, το οποίο παρουσιάζεται παρακάτω. Όπως βλέπουμε, λαμβάνοντας υπόψη ότι η ταξινόμηση αυτού του συνόλου δεδομένων είναι πολύ περίπλοκη για να υπολογιστεί με γραμμικούς τρόπους, ο διαχωρισμός των ψηφίων δεν είναι επιτυχής. Παρ' όλα αυτά, εμείς θα μπορούσαμε να εντοπίσουμε ορισμένα ασαφή clusters ψηφίων.



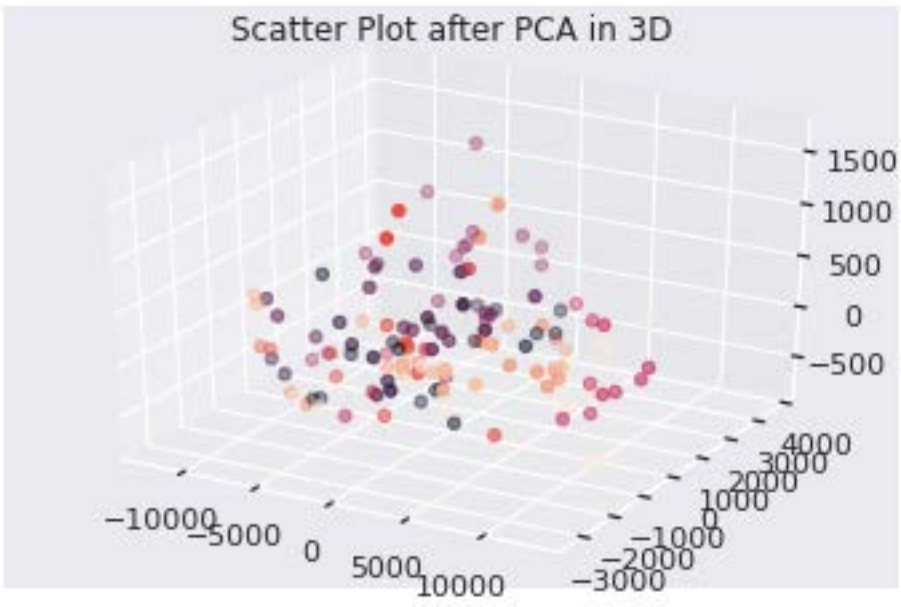
Βήμα 6

Με τη συνάρτηση της sklearn PCA(n_components), θέτοντας την μεταβλητή n_components στην επιθυμητή τιμή, επιτυγχάνεται η μείωση των διαστάσεων του διανύσματος.

n_components=2



n_components=3



Εδώ, όπως και παραπάνω, δεν είμαστε κοντά σε ένα σωστό διαχωρισμό των ψηφίων. Ωστόσο, τα clusters είναι λίγο πιο ευδιάκριτα από ό,τι παραπάνω, ιδίως στο τρισδιάστατο μοντέλο. Υπολογίζοντας τον λόγο της διακύμανσης των 3 πρώτων κύριων διαστάσεων σε σχέση με τις την αρχική διακύμανση έχουμε 0.9331945 για την πρώτη, 0.06126316 για τη δεύτερη και 0.00554229 για την τρίτη.

Μπορούμε εύκολα να παρατηρήσουμε ότι για την πρώτη κύρια διάσταση ο αλγόριθμος επιτυγχάνει ένα πραγματικά υψηλό ποσοστό της αρχικής διακύμανσης, σε αντίθεση με τις άλλες δύο που τα αποτελέσματα είναι πραγματικά χαμηλά. Αυτό σημαίνει ότι οι δύο τελευταίες συνιστώσες δεν προσφέρουν πολλά στην αναπαράσταση των δεδομένων μας, ενώ η πρώτη κατέχει τις περισσότερες πληροφορίες.

Βήμα 7

Στο ερώτημα αυτό το σύνολο των δεδομένων είναι οι 133 εκφωνήσεις με τα 78 χαρακτηριστικά της η κάθε μία. Με τη συνάρτηση train_test_split χωρίζουμε τα δεδομένα σε train και test και πετυχαίνουμε αναλογία 70%-30% train-test θέτοντας test_size=0.3.

Κανονικοποιούμε τα train και test δεδομένα με StandardScaler() και RobustScaler(). Ο RobustScaler() scaleάρει τα χαρακτηριστικά με χρήση στατιστικών που είναι ανθεκτικά σε outliers. Πιο συγκεκριμένα αφαιρεί το median και κλιμακώνει τα δεδομένα σύμφωνα με το quantile range. Τα αποτελέσματα με τον scaler αυτόν δεν είναι καλά, συνεπώς αφοσιωθήκαμε στον Standard.
Παρακάτω παραθέτουμε τις τιμές του accuracy και f1 score (macro & micro) για κάθε έναν από τους ταξινομητές που χρησιμοποιήσαμε. Χρησιμοποιήσαμε Gaussian Naive Bayes, CustomNBC, SVC, KNN, RandomForest.

Αρχικά, κάνουμε fit, predict, score και f1 score σε Out Of the Box δεδομένα. Στην ουσία δεν έχουμε υπολογίσει υπερπαραμέτρους, ούτε έχουμε χρησιμοποιήσει κάποιου είδους βελτιστοποίηση. Τα αποτελέσματα είναι:

	CustomNBC	GNB	SVC	KNN	RandomF
accuracy	0.48	0.475	0.4	0.4	0.375
f1-micro	0.48	0.475	0.4	0.4	0.375
f1-macro	0.47903	0.47013	0.42813	0.35444	0.38151

Στη συνέχεια, υπολογίζουμε υπερπαραμέτρους χρησιμοποιώντας τη βελτιστοποιημένη GridSearchCV από τη βιβλιοθήκη της Ray.io, ενώ προσθέτουμε και zero crossing rate. Η συνάρτηση είναι η TuneGridSearchCV, την οποία δοκιμάσαμε σε άλλο αρχείο. Προκύπτει:

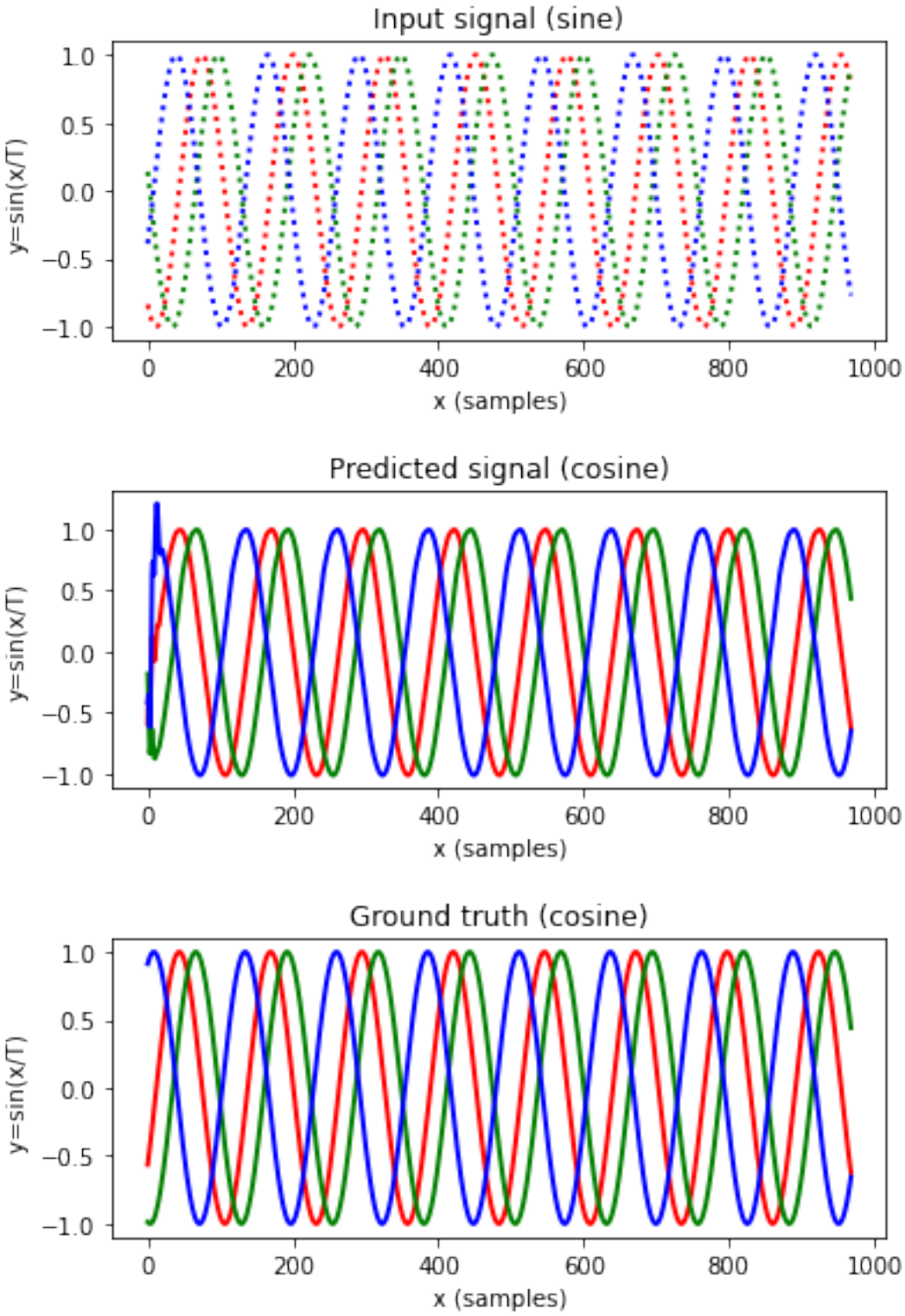
	CustomNBC	GNB	SVC	KNN	RandomF
accuracy	0.76	0.75	0.6	0.425	0.5
f1-micro	0.76	0.75	0.6	0.425	0.5
f1-macro	0.75204	0.74868	0.61360	0.43139	0.49433

Προσθέτοντας zero crossing rate αυξάνεται το accuracy, αλλά όχι σημαντικά. Αυτό που περιμέναμε (και όντως συνέβη) είναι να φτάσει ο αλγόριθμος σε ένα σημείο κορεσμού, καθώς οι κλασικές μέθοδοι ML σε ηχητικά χαρακτηριστικά χωρίς τη χρήση επιπλέον καταστάσεων (π.χ. hmm, rnn) δεν είναι ιδιαίτερα αποδοτικές.

Βήμα 8

Σε αυτό το βήμα, προσπαθήσαμε να προβλέψουμε μια καμπύλη cos έχοντας ως είσοδο το sin εκπαιδεύοντας ένα αναδρομικό νευρωνικό δίκτυο (RNN). Βασικά, υπολογίζει την έξοδο για κάθε χρονικό βήμα και την τοποθετεί στο Liner για να υπολογίσει την προβλεπόμενη έξοδο. Ο ρυθμός μάθησης επιλέχθηκε στο 0,02 και το χρονικό βήμα 10. Αρχικά, ορίζουμε το RNN ορίζοντας τη συνάρτηση προώθησης και τα επίπεδα. Υλοποιήθηκε με ένα στρώμα και επιλέχθηκαν 36 κρυφές μονάδες rnn. Στη συνέχεια υλοποιήσαμε την εκπαίδευση του μοντέλου. Σε κάθε εποχή μετακινηθήκαμε σταδιακά στις επόμενες π χρονικές μονάδες και δημιουργήσαμε 10 ισαπέχοντα σημεία σε αυτή τη μισή περίοδο του ημιτόνου. Το sin αυτών των στοιχείων είναι τα δεδομένα εισόδου μας και τα cos η έξοδος που θέλουμε να προσαρμόσουμε. Μετά από κάθε πρόβλεψη, επανασυσκευάζουμε την κρυφή κατάσταση προκειμένου να διακόψουμε τη σύνδεση από την τελευταία επανάληψη. Στη συνέχεια υπολογίζουμε την απώλεια και υπολογίζουμε ξανά τις κλίσεις με backpropagation. Στο τέλος εφαρμόζουμε τις κλίσεις για να βελτιστοποιήσουμε το αποτέλεσμα κάθε επανάληψης. Συνολικά, το RNN είναι σε θέση να κατανοήσει τη σχέση μεταξύ sin και cos και να χρησιμοποιήσει το παραμέτρους μέσα στο RNN για να αποφασίσει ποιο σημείο στην καμπύλη cos αντιστοιχεί σε ένα σημείο στο sin σε κάθε χρονικό βήμα. Αφού κάνουμε μια συνεχή γραφική παράσταση, μπορούμε να δούμε την πρόοδο του μοντέλου RNN μας στο παρακάτω σχήμα.

Οι εποχές είναι 100 και το loss 0.0047.

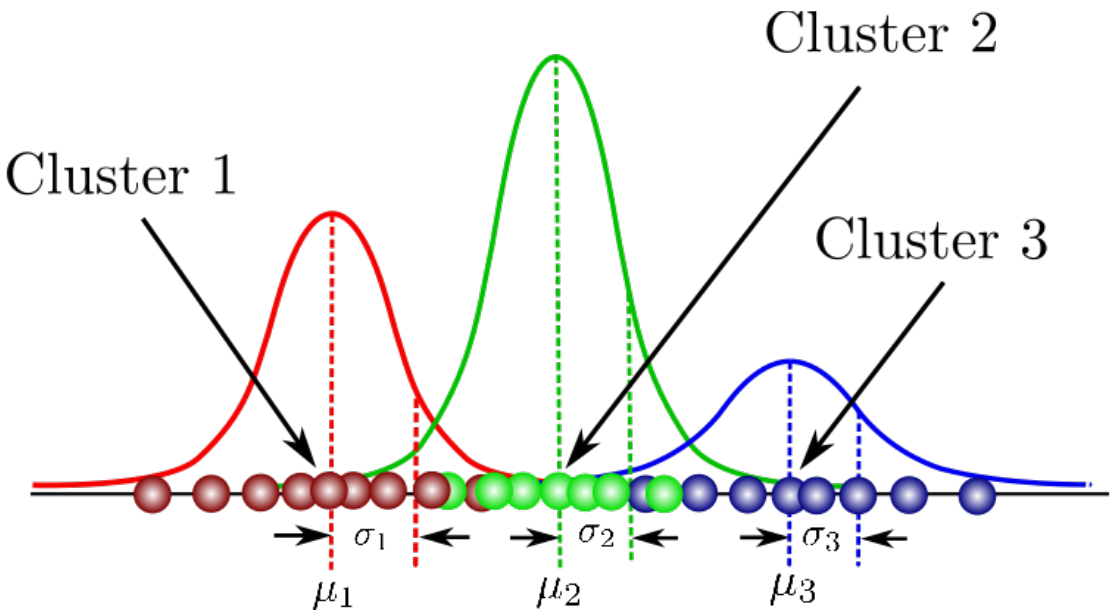


Βήμα 9

Χρησιμοποιώντας την συνάρτηση "*StratifiedShuffleSplit*" διαχωρίζουμε τα δεδομένα με τέτοιο τρόπο ώστε να έχουμε τον ίδιο αριθμό διαφορετικών ψηφίων σε κάθε set, και με ποσοστό 80%-20%.

Βήμα 10

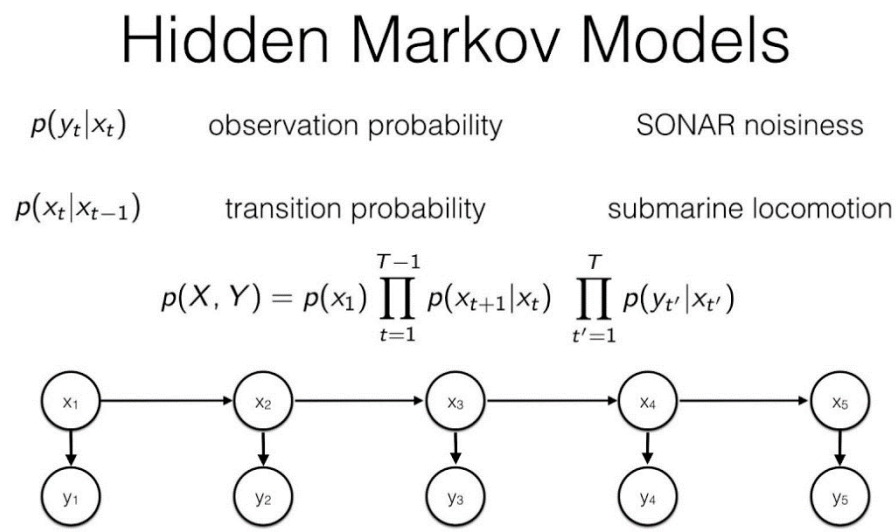
Όπως ήδη γνωρίζουμε, τα γκαουσιανά μοντέλα είναι εύκολο να χρησιμοποιηθούν στη Μηχανική Μάθηση και παράγουν εξαιρετικά αποτελέσματα. Αλλά συνήθως (όπως στην περίπτωσή μας) τα δεδομένα είναι πολύ πολύπλοκα για να περιγραφούν με ένα Γκαουσιανό μοντέλο. Για το λόγο αυτό χρησιμοποιούνται τα Μοντέλα Γκαουσιανής Μίξης, τα οποία συνδυάζουν μια σειρά από Γκαουσιανές κατανομές, δίνοντας την ευελιξία που απαιτείται στην αναγνώριση ομιλίας. Πιο συγκεκριμένα, όσον αφορά το speech recognition, μια σειρά χαρακτηριστικών είναι μια πιθανή παρατήρηση σε μια κατάσταση, και δεδομένου ότι επιτρέπονται συνεχείς αλλαγές αυτής της παρατήρησης, είναι δυνατή η μοντελοποίηση με GMM.



Το μοντέλο HMM είναι χρήσιμο προκειμένου να αναπαρασταθεί η ομιλία ως μια ακολουθία παρατηρήσεων. Στη δική μας περίπτωση, χρησιμοποιούμε τα φωνήματα ως ακολουθίες για να παρατηρήσουμε τα προφερόμενα ψηφία.
Στην περίπτωσή μας, προσαρμόζουμε 10 μοντέλα, ένα για κάθε ψηφίο, και προκειμένου να προβλέψουμε τιμές, υπολογίζουμε την πιθανότητα κάθε μοντέλου χρησιμοποιώντας τον αλγόριθμο EM και επιλέγουμε ως πρόβλεψη το ψηφίο που αντιστοιχεί στο μοντέλο με τη μεγαλύτερη πιθανότητα. Πιο συγκεκριμένα, εάν W είναι το ψηφίο (λέξη) του οποίου την πιθανότητα προσπαθούμε να προβλέψουμε, πρέπει να υπολογίσουμε

$$argmax_w\ P(W|X) = argmax_w\ P(X|W)P(W)/P(X) = argmax_w\ P(X|W)P(W)$$

αφού ότι η P(X) είναι σταθερά. Για κάθε μοντέλο, λαμβάνουμε τα MFCC (/frames) για το ισοδύναμο ψηφίο στο GMM. Δηλαδή επειδή τα GMM εκτελούν εκτίμηση πυκνότητας, η οποία είναι μια μη unsupervised μέθοδος. Ως εκ τούτου, προσπαθεί να προσεγγίσει τη συνάρτηση πιθανότητας πυκνότητας από τα πλαίσια ενός ψηφίου.
Το μοντέλο HMM αποτελείται από κρυφές μεταβλητές και παρατηρήσιμες μεταβλητές. Οι κάτω κόμβοι είναι τα ηχητικά χαρακτηριστικά και οι πάνω είναι τα φωνήματα (κρυφές μεταβλητές).



Οι πιθανότητες μετάβασης στις αλυσίδες Markov ορίζονται στον πίνακα μετάβασης μας. Όπως περιγράφεται, οι μόνες μη μηδενικές τιμές είναι οι ai,i και ai,i+1 ∀i. Επιλέξαμε το 1/2 για κάθε τιμή, και για το 10 πιθανότητα εκπομπής (τερματισμού), έχουμε 1/2 για την τελευταία κατάσταση και μηδέν για τις υπόλοιπες. Επιπλέον, η αρχική μας κατάσταση είναι πάντα η πρώτη.

Βήμα 11

Στο σημείο αυτό τρέχουμε την συνάρτηση που υλοποιήσαμε στο παραπάνω βήμα για κάθε ένα ψηφίο. Η έτοιμη συνάρτηση της python η fit χρησιμοποιεί τον αλγόριθμο Expectation Maximization (EM).

Βήμα 12

Στο βήμα αυτό υλοποιούμε την συνάρτηση eval_models για την αναγνώριση μεμονωμένων ψηφίων-Testing. Τρέχουμε τη *viterbi* της python για να υπολογίσουμε το log likelihood για κάθε εκφώνηση. Έπειτα, με την *numpy.argmax* βρίσκουμε την μέγιστη πιθανοφάνεια και συνεπώς το αποτέλεσμα της αναγνώρισης για την συγκεκριμένη εκφώνηση.

Έπειτα, στο validation test με n_states = 4 και n_mixtures = 2 έχουμε τα βέλτιστα accuracies.
Validation data: 0.97778
Test data: 0.97

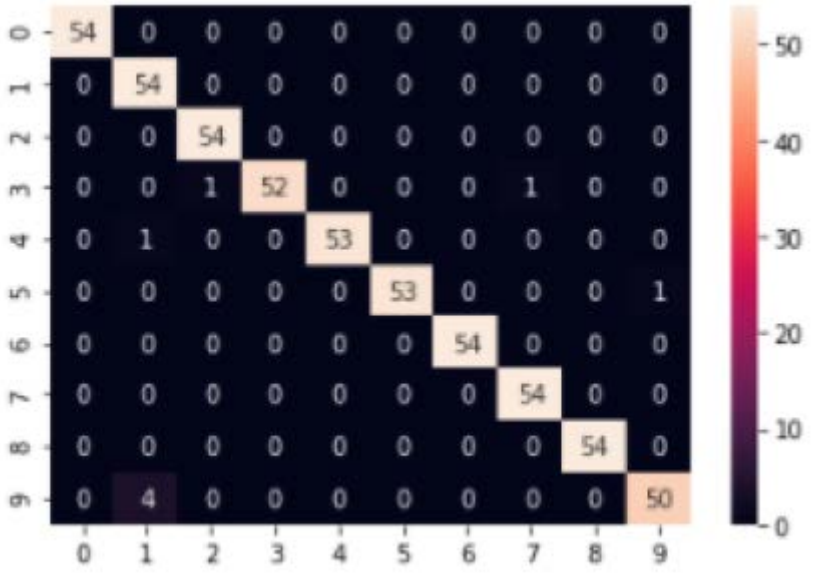
Συνήθως, χρησιμοποιούμε validation set για να αποφύγουμε το overfitting και να υπολογίσουμε τις καλύτερες παραμέτρους (cross-validation), και είναι ανεξάρτητο και από το training και από το test set. Αντιθέτως, το test set το χρησιμοποιούμε μόνο για τον υπολογισμό της επίδοσης ενός ήδη εκπαιδευμένου μοντέλου.

Ο ορισμός των υπερπαραμέτρων με τη χρήση του validation set είναι μια απαραίτητη διαδικασία. Πρώτα απ' όλα, το σύνολο επικύρωσης έχει προκύψει από stratified split (και έτσι διατηρούνται τα ποσοστά των των διαφόρων ψηφίων) και αυτό το καθιστά ένα καλό σύνολο αναφοράς για το σύνολο εκπαίδευσής μας.
Πρωτίστως όμως, δεν θα μπορούσαμε να χρησιμοποιήσουμε το σύνολο ελέγχου, δεδομένου ότι το σύνολο που χρησιμοποιείται για τον προσδιορισμό των καλύτερων υπερπαραμέτρων

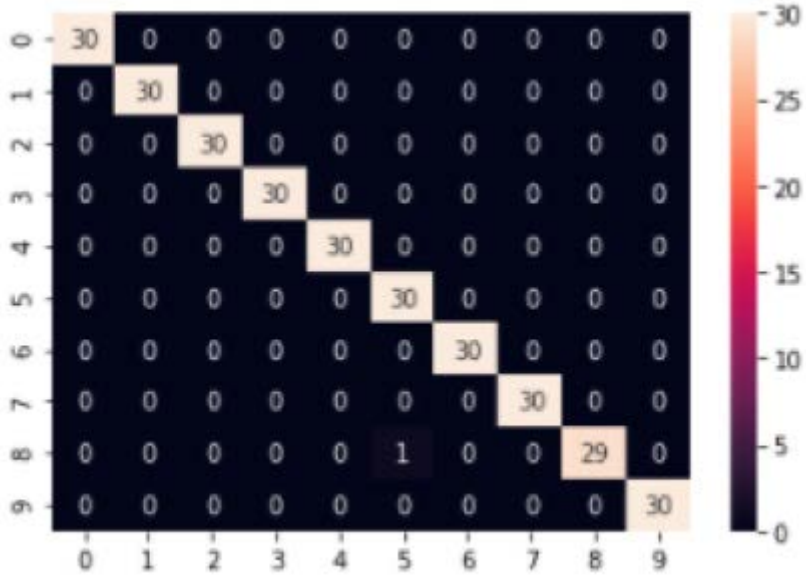
γίνεται μέρος της προσαρμογής του μοντέλου. Επομένως, θα ήταν μάταιο και θα παρήγαγε υπεραισιόδοξα αποτελέσματα. Το ίδιο θα συνέβαινε και στην περίπτωση που χρησιμοποιούσαμε το ίδιο validation set για τη μέτρηση της απόδοσης.

Βήμα 13

Validation set's confusion matrix:



Test set's confusion matrix:



Βήμα 14

Μέρη 1 – 6

Υλοποιήσαμε ένα δίκτυο LSTM που στοχεύει στην ταξινόμηση προφορικών ψηφίων. Η είσοδος αποτελείται από ακολουθίες μεταβλητού μήκους, τις οποίες συμπληρώσαμε με 0.

Σε κάθε χρονικό βήμα η είσοδος είναι 6 χαρακτηριστικά (σε αυτή την εργασία αναγνώρισης ομιλίας χρησιμοποιούνται συντελεστές mel-frequency cepstrum coefficients). Μετά από προσεκτική χειροκίνητη ρύθμιση των υπερπαραμέτρων και παρατηρώντας την ακρίβεια τις καθορίσαμε ως εξής:

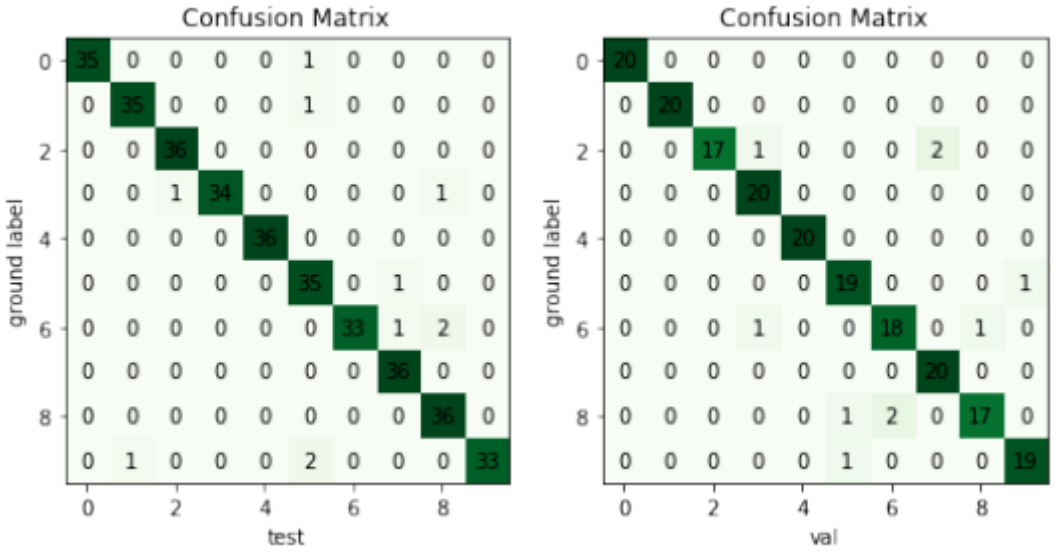
hiddensize = 16 και number_of_layers = 2, epochs = 25.

Στο LSTM μας εισαγάγαμε μια νέα υπερπαράμετρο, την πιθανότητα εγκατάλειψης (dropout probability) που καθορίζει την πιθανότητα με την οποία οι έξοδοι του στρώματος εγκαταλείπονται, ή αντίστροφα, την πιθανότητα με την οποία έξοδοι του στρώματος διατηρούνται. Την ορίσαμε ως 0,4. Η πιθανολογική απόρριψη κόμβων σε ένα δίκτυο είναι μια απλή και αποτελεσματική μέθοδος κανονικοποίησης, η οποία μειώνει το overfitting και βελτιώνει το σφάλμα γενίκευσης. Στην περίπτωσή μας έχουμε περιορισμένα δεδομένα εκπαίδευσης και πολλά από τα περίπλοκα σχέσεις μεταξύ εισόδων και εξόδων θα είναι αποτέλεσμα sampling noise, οπότε θα υπάρχουν στο train set αλλά όχι στο test set. Αυτό μπορεί να οδηγήσει σε overfitting. Η προφανής ιδέα του μέσου όρου των εξόδων πολλών χωριστά εκπαιδευμένων δικτύων είναι απαγορευτικά δαπανηρή. Εφαρμόζοντας dropout σε ένα νευρωνικό δίκτυο ισοδυναμεί με τη δειγματοληψία ενός "αραιωμένου" δικτύου από αυτό. Το αραιωμένο δίκτυο αποτελείται από από όλες τις μονάδες που επέζησαν από το dropout. Έτσι, η εκπαίδευση ενός νευρωνικού δικτύου με dropout μπορεί να θεωρηθεί ως εκπαίδευση μιας συλλογής εκθετικά πολλών αραιωμένων δικτύων με εκτεταμένο διαμοιρασμό βαρών και συνεπώς μπορεί να ερμηνευθεί ως μια μορφή μέσης τιμής μοντέλου. Εκτός από το dropout, μια άλλη τεχνική κανονικοποίησης που εφαρμόσαμε είναι η κανονικοποίηση L2. Εμείς εισαγάγαμε έναν όρο αποσύνθεσης των βαρών και τον ορίσαμε ως 0,001. Η κανονικοποίηση L2 (παλινδρόμηση κορυφογραμμής) προσθέτει "τετραγωνικό μέγεθος" του συντελεστή ως penalty στη loss function και έτσι "τιμωρεί" τα μεγάλα βάρη.

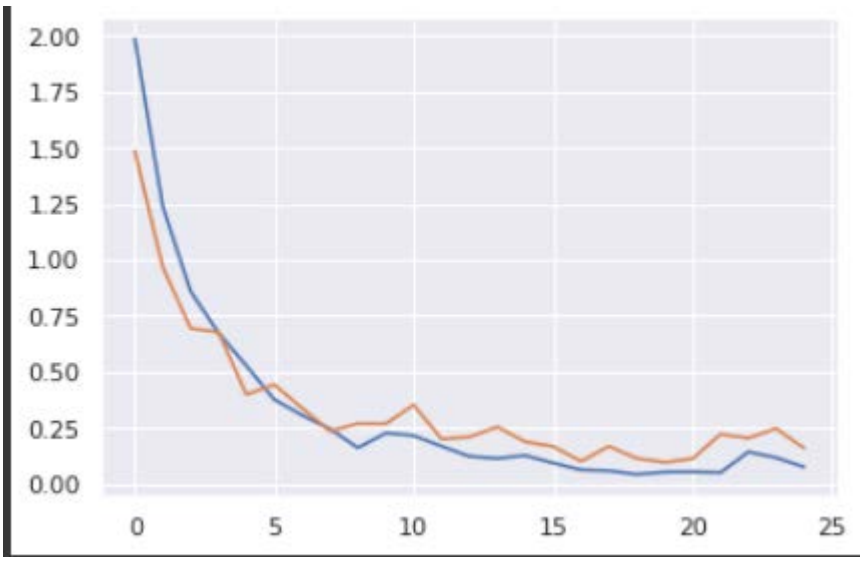
$$Loss = Error(y, \hat{y}) + \sum_{i=1}^{N} w_i^2$$

Η κανονικοποίηση προσπαθεί να μειώσει τη διακύμανση του εκτιμητή απλοποιώντας τον, κάτι που το οποίο θα αυξήσει το bias, με τέτοιο τρόπο ώστε να μειωθεί το αναμενόμενο σφάλμα. Ως τελευταίο βήμα εφαρμόσαμε early stopping, με υπομονή 10 εποχών. Εάν το validation loss δεν παρουσιάζει καμία βελτίωση σε αυτές τις εποχές, η φάση της εκπαίδευσης σταματά. Το early stopping είναι μια μορφή κανονικοποίησης που χρησιμοποιείται για την αποφυγή του overfitting. Μέχρι ενός σημείου η απόδοση του εκπαιδευόμενου σε δεδομένα εκτός του train set βελτιώνεται. Μετά από αυτό το σημείο, η βελτίωση της προσαρμογής του μαθητή στα training data είναι σε βάρος του αυξημένου generalization error. Οι κανόνες του early stopping παρέχουν καθοδήγηση ως προς το πόσες επαναλήψεις μπορούν να εκτελεστούν προτού εμφανιστεί overfitting.

Confusion Matrices:



Loss Plot:



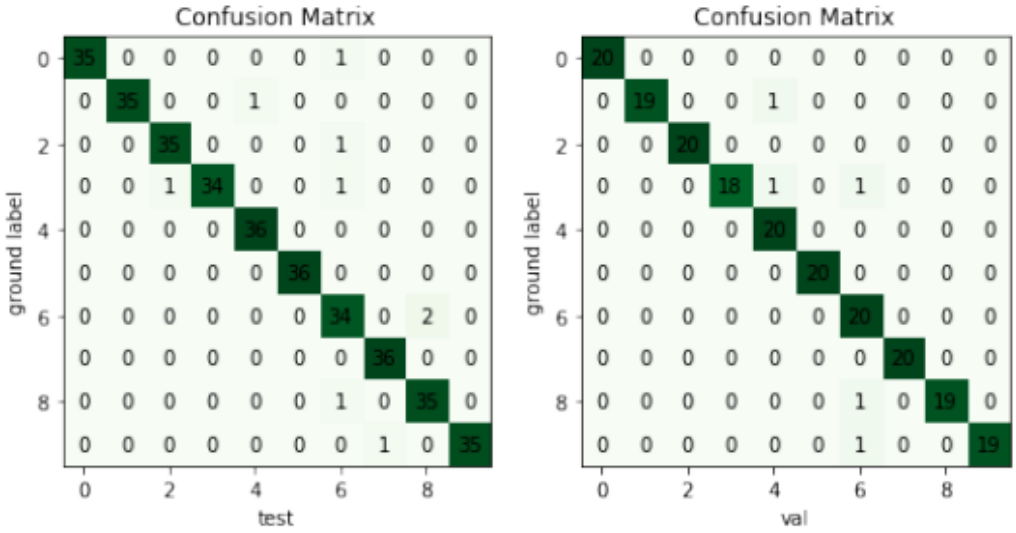
Results:

```
Epoch: 8/25 Step: 432 Train Loss: 0.357304
Epoch: 9/25 Step: 486 Train Loss: 0.070508
Validation loss(from 0.308356 to 0.261338)
Epoch: 10/25 Step: 540 Train Loss: 0.114839
Epoch: 11/25 Step: 594 Train Loss: 0.377465
Epoch: 12/25 Step: 648 Train Loss: 0.141783
Validation loss(from 0.261338 to 0.199140)
Epoch: 13/25 Step: 702 Train Loss: 0.050330
Epoch: 14/25 Step: 756 Train Loss: 0.037272
Epoch: 15/25 Step: 810 Train Loss: 0.059957
Epoch: 16/25 Step: 864 Train Loss: 0.030679
Validation loss(from 0.199140 to 0.150511)
Epoch: 17/25 Step: 918 Train Loss: 0.042318
Epoch: 18/25 Step: 972 Train Loss: 0.027212
Validation loss(from 0.150511 to 0.110212)
Epoch: 19/25 Step: 1026 Train Loss: 0.021070
Epoch: 20/25 Step: 1080 Train Loss: 0.052705
Validation loss(from 0.110212 to 0.102473)
Epoch: 21/25 Step: 1134 Train Loss: 0.018517
Epoch: 22/25 Step: 1188 Train Loss: 0.165695
Epoch: 23/25 Step: 1242 Train Loss: 0.252725
Epoch: 24/25 Step: 1296 Train Loss: 0.095546
Epoch: 25/25 Step: 1350 Train Loss: 0.041955
```

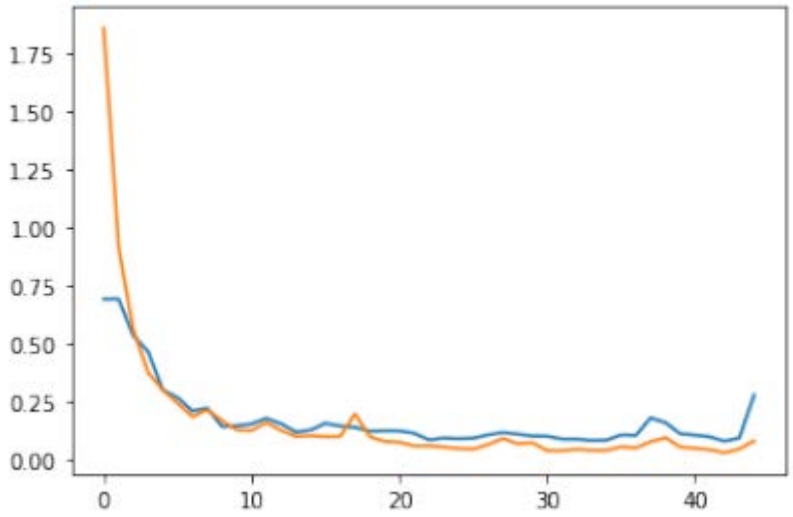
Μέρη 7

Σε πολλές περιπτώσεις είναι επιθυμητό να ξεπεραστούν οι περιορισμοί για να χρησιμοποιηθούν πληροφορίες τόσο από το παρελθόν (προς τα πίσω) όσο και από τις μελλοντικές (προς τα εμπρός) καταστάσεις ταυτόχρονα. Η ιδέα των αμφίδρομων επαναλαμβανόμενων νευρωνικών δικτύων (BRNNs) είναι η διάσπαση των νευρώνων κατάστασης ενός κανονικού RNN σε ένα μέρος που είναι υπεύθυνο για τη θετική χρονική κατεύθυνση (προς τα εμπρός καταστάσεις) και ένα μέρος για τις αρνητικές χρόνο (καταστάσεις προς τα πίσω). Αυτά τα δύο επαναλαμβανόμενα κρυφά στρώματα μοιράζονται το ίδιο στρώμα εξόδου. Στο speech recognition η παροχή της ακολουθίας σε δύο κατευθύνσεις δικαιολογείται επειδή υπάρχουν ενδείξεις ότι το πλαίσιο της όλης εκφώνησης χρησιμοποιείται για την ερμηνεία του τι λέγεται και όχι μια γραμμική (σε χρονική) ερμηνεία. Σε αυτό το βήμα χρησιμοποιήσαμε ένα αμφίδρομο LSTM με τις ίδιες υπερπαραμέτρους με το προηγούμενο μοντέλο, αλλάζοντας τις εποχές σε 45. Παρατηρήσαμε ελαφρώς υψηλότερη ακρίβεια στο σύνολο δεδομένων δοκιμής. Παρουσιάζουμε τα αποτελέσματά μας παρακάτω.

Confusion Matrices:



Loss Plot:



Results:

Test data accuracy: 98% with a loss of 0.1545

Validation data accuracy: 98% with a loss of 0.081