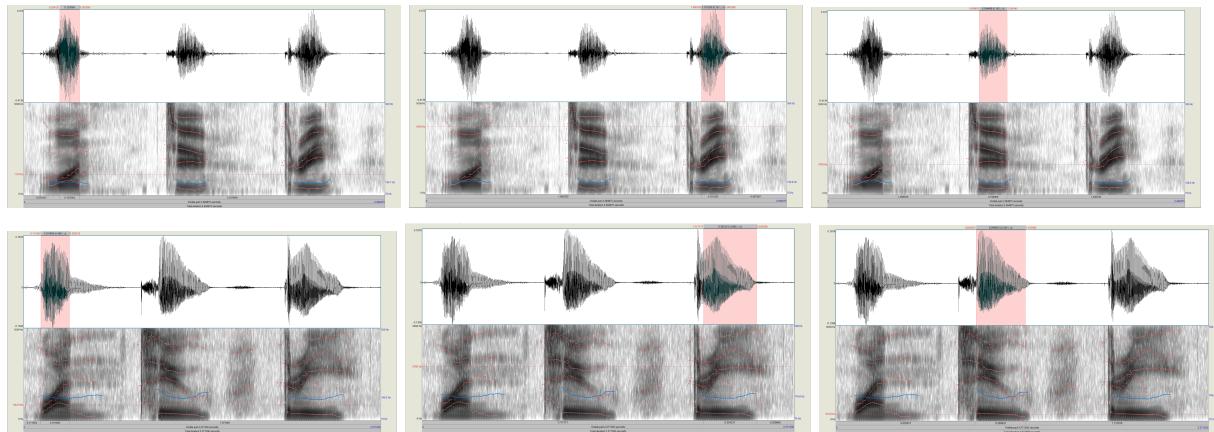


Αναγνώριση Προτύπων - 2η Εργαστηριακή Άσκηση

Κωνσταντίνος Παπαδάκης (ΕΔΕΜΜ 03400149)

Βήμα 1

Στις παρακάτω εικόνες, στην πρώτη σειρά έχουμε τα επιλεγμένα διαστήματα με τα οποία υπολογίστηκαν το pitch και τα formants για τον onetwothree1.wav, και στην δεύτερη για την onetwothree8.wav.



Συγκεκριμένα, τα αποτελέσματα είναι

onetwothree1	Pitch	F1	F2	F3
ɑ	134.68	759.46	1129.84	2409.28
ou	129.15	352.07	1780.08	2374.44
i	130.89	385.68	1868.60	2289.70

onetwothree8	Pitch	F1	F2	F3
α	180.21	774.07	1374.71	2821.53
ου	188.23	335.12	1750	2688
ι	179.59	405.80	2136.72	2804.88

Παρατηρούμε ότι η γυναίκα (onetwothree8) έχει υψηλότερο pitch και ότι ο συνδυασμός των τριών πρώτων formants ίσως να μπορεί να προβλέψει το φωνήν διότι είναι αρκετά διαφορετικά μεταξύ τους ενώ παίρνουν περίπου τις ίδιες τιμές στους δύο ομιλητές.

Bήμα 2

Βλέπε part1.data_parser

Bήμα 3

Βλέπε part1.step_3

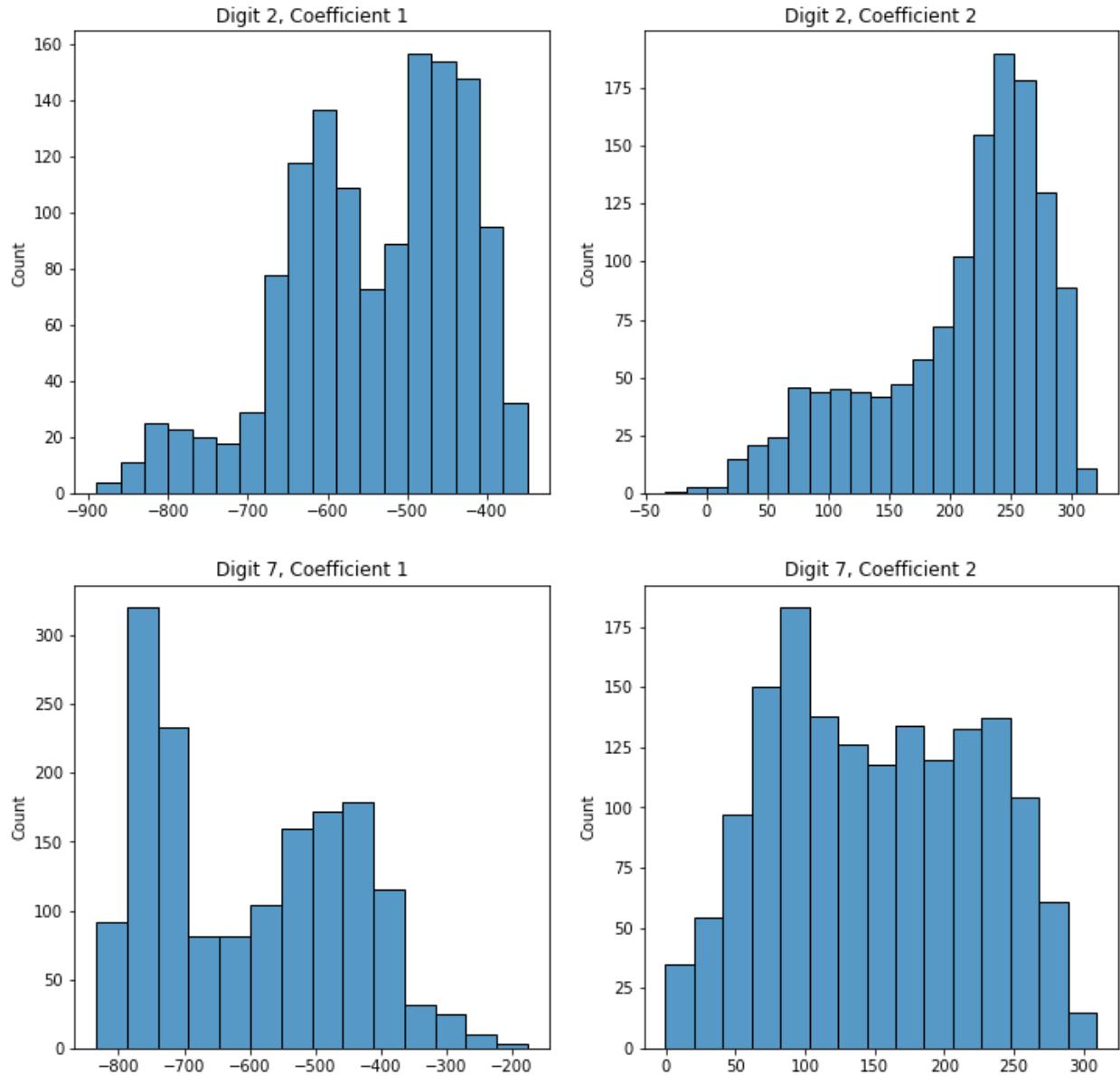
Bήμα 4

(part1.step_4)

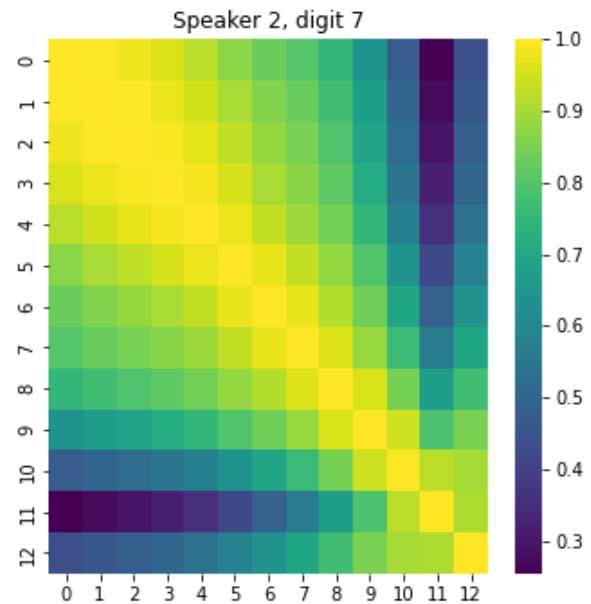
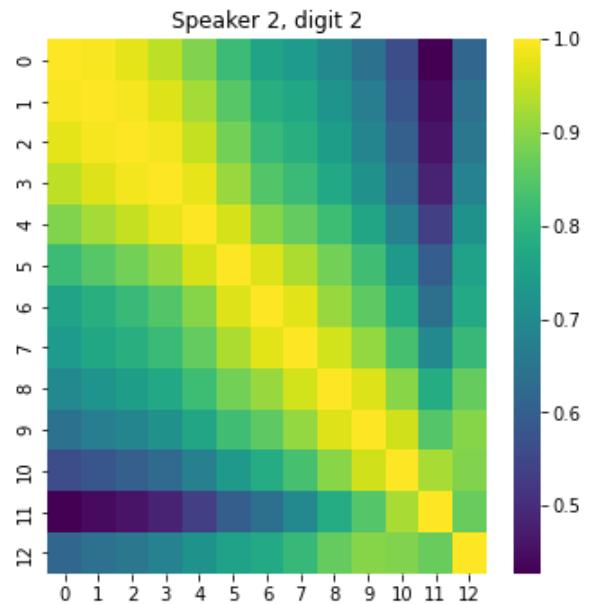
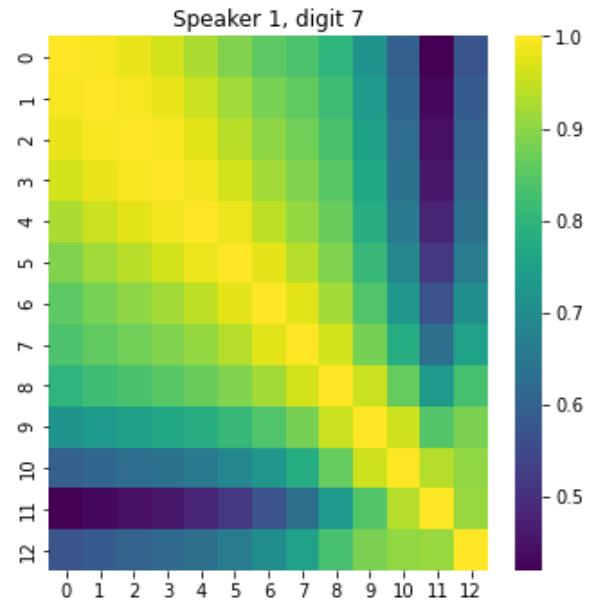
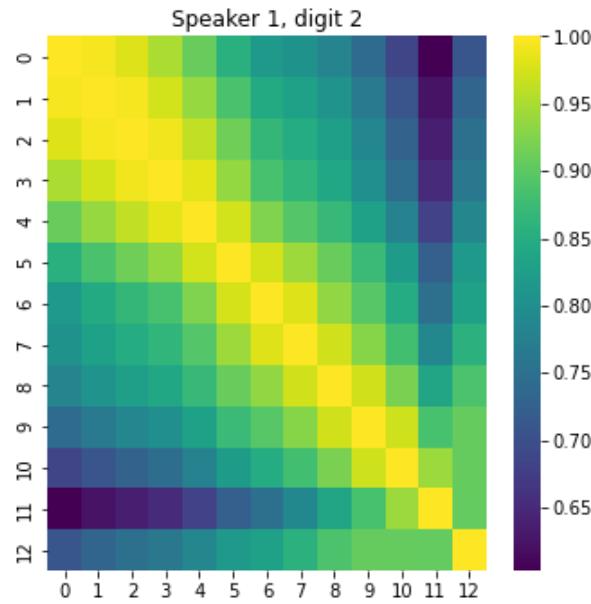
Τα ιστογράμματα βλέπουμε πως είναι όλα πολύ διαφορετικά μεταξύ τους. Δεν φαίνεται να μπορούμε να αποφανθούμε κάτι από αυτά.

Όσον αφορά τα MFSC / MFCC, βλέπουμε πως η εφαρμογή του DCT παράγει ασυσχέτιστα μεταξύ τους χαρακτηριστικά. Τα ασυσχέτιστα χαρακτηριστικά είναι προτιμότερα για κλασικούς μοντέλα όπως HMM. Για deep learning μπορούμε να χρησιμοποιήσουμε και τα MFCC και μάλιστα να έχουμε καλύτερα αποτελέσματα γιατί δεν χάνουμε πληροφορία από το DCT.

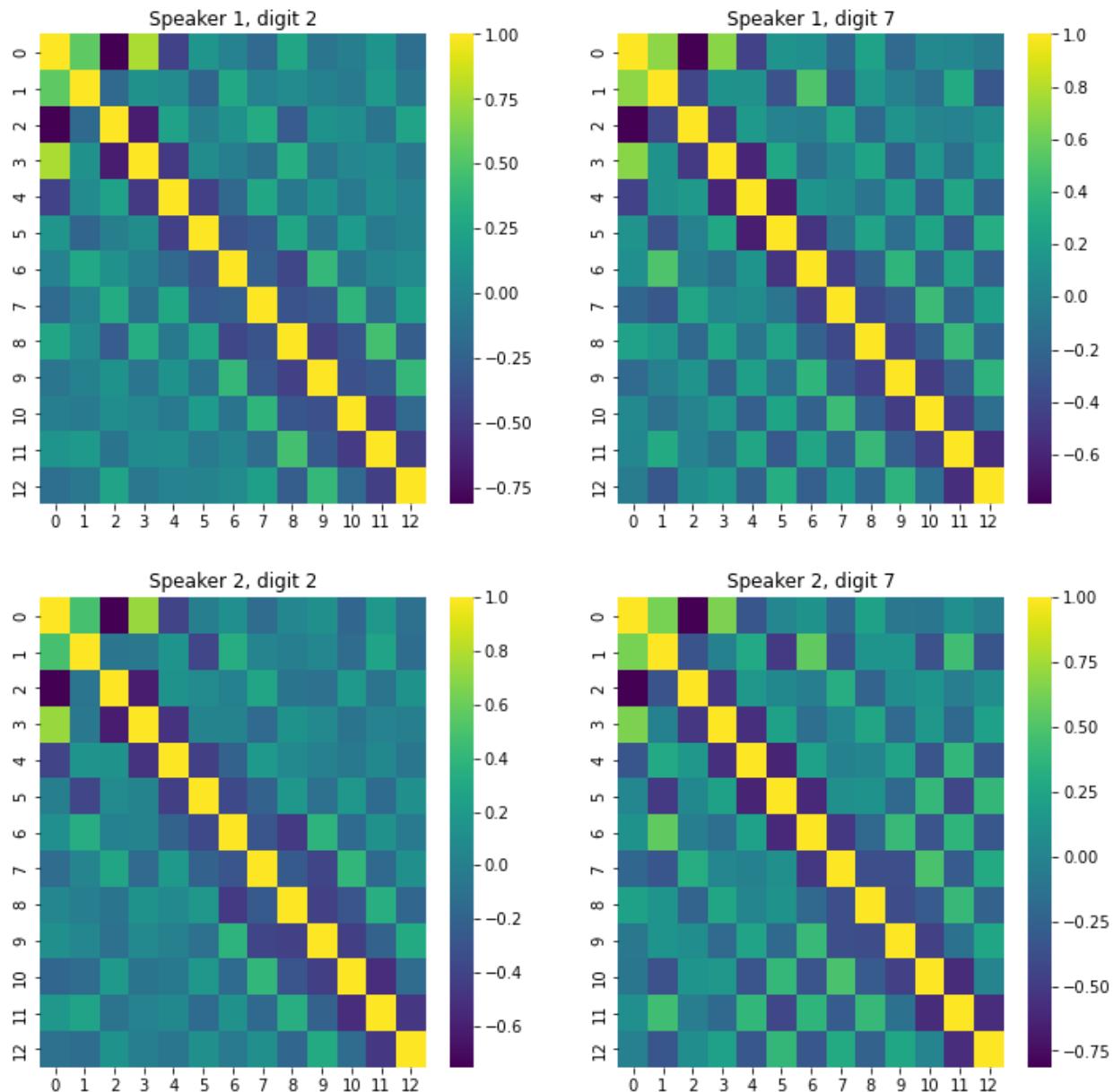
MFCCS



MFSC



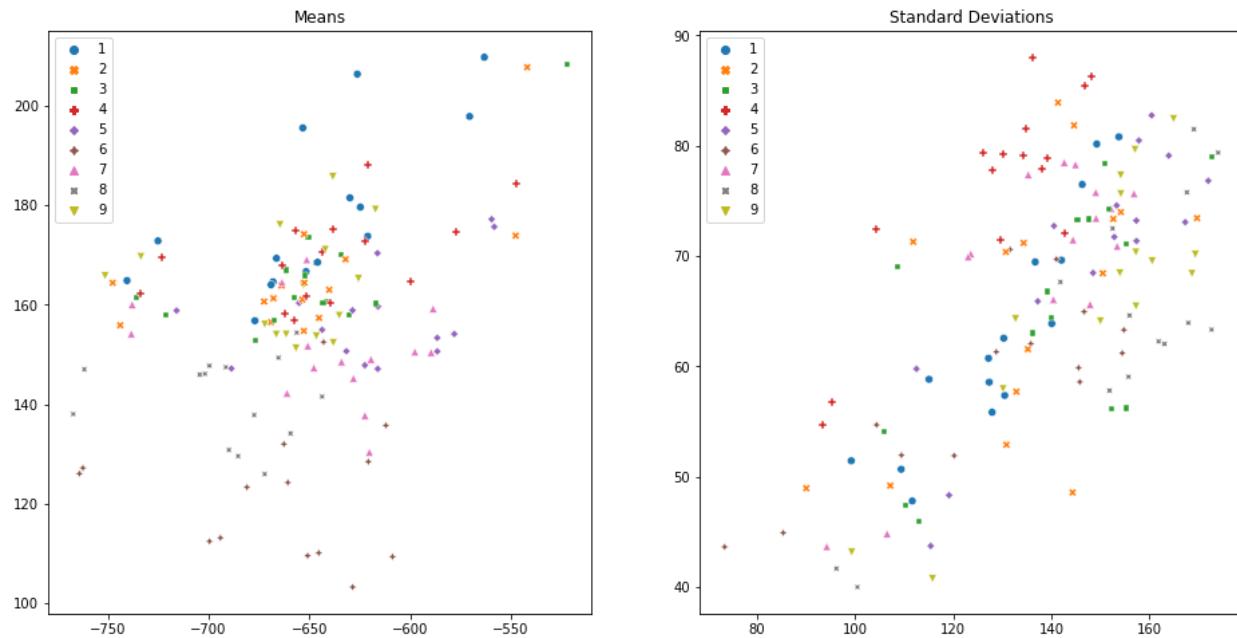
MFCC



Βήμα 5

(part1.step_5)

Παρατηρούμε πως και στην περίπτωση των μέσων, και στην περίπτωση των τυπικών είναι δύσκολο να διαχωρίσουμε τις κλάσεις χρησιμοποιώντας τα πρώτα δύο χαρακτηριστικά. Αυτό ίσως να μας κάνει να υποθέσουμε πως δεν θα είναι εύκολο να προβλέψουμε καλά τις κλάσεις μόνο με αυτά τα χαρακτηριστικά.



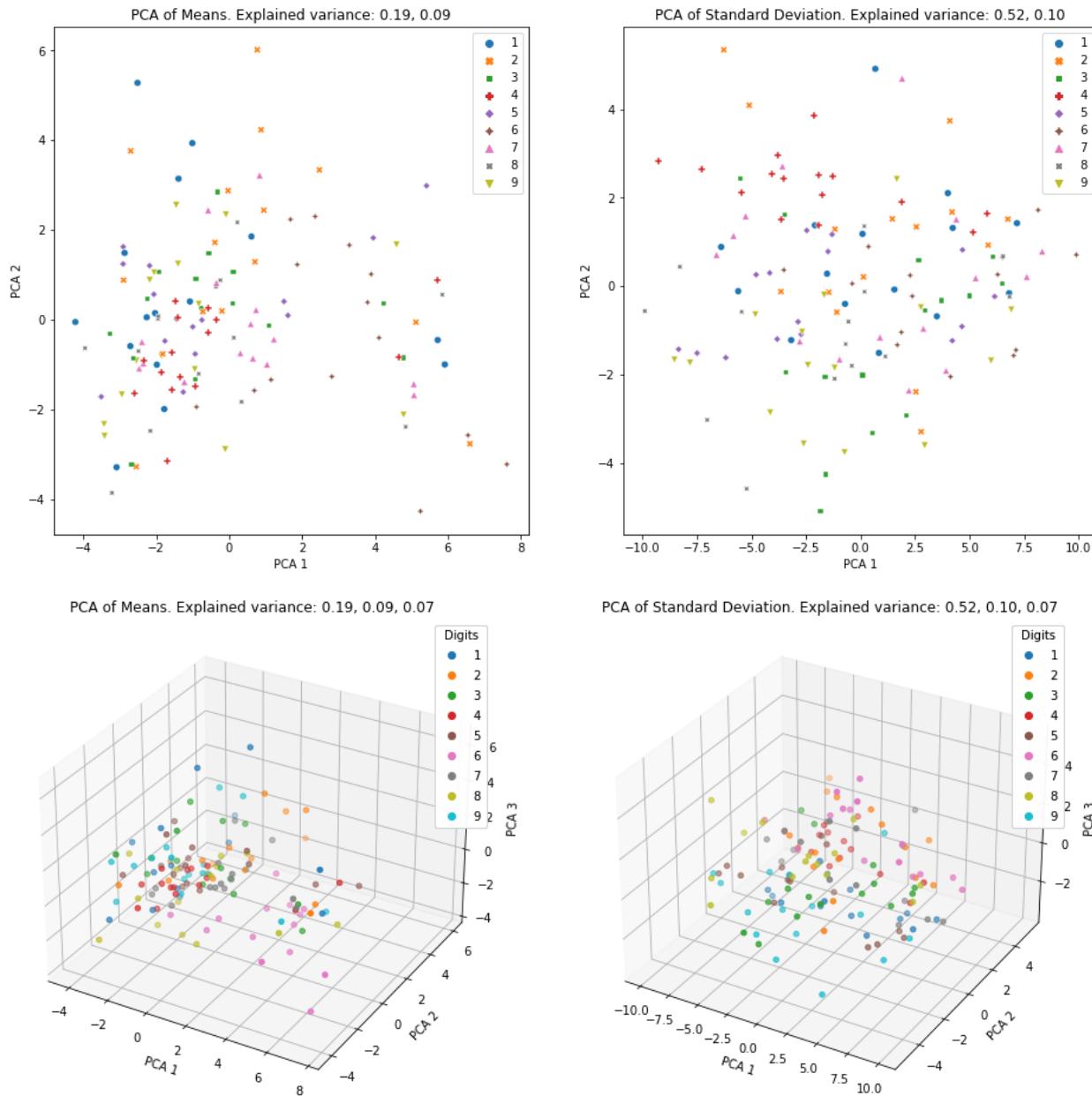
Βήμα 6

(part1.step_6)

Τα γραφήματα μαζί με τα ποσοστά της αρχικής διασποράς φαίνονται στον πίνακα.

Το ποσοστό διασποράς μιας συνεκτικής συνιστώσας είναι ο η διασποράς αυτής, ως προς το άθροισμα το διασπορών όλων των συνιστωσών. Σε κανονικοποιημένα δεδομένα, εκεί που υπάρχει μεγαλύτερη διακύμανση, συνήθως υπάρχει και μεγαλύτερη πληροφορία για τον διαχωρισμό των δεδομένων.

Στην περίπτωσή μας, τα ποσοστά διασποράς είναι σχετικά μικρά και για αυτόν τον λόγο είναι και πάλι δύσκολο να διαχωρίσουμε τα δεδομένα.



Βήμα 7

(part1.step7)

Τα μοντέλα που εκπαιδεύτηκαν ήταν τα GaussianNB ('gnb'), CustomNBClassifier ('cgnb'), linear SVC ('svc'), KNeighborsClassifier ('knn'), RandomForestClassifier ('rf')

Επαύξησα τα χαρακτηριστικά με zero_crossing_rate και poly_features τρίτης τάξης.

Επίσης, δοκίμασα StandardScaler, Normalizer, και τίποτα από τα δύο.

Τα αποτελέσματα όπως αποτυπώνονται από το πρόγραμμα φαίνονται παρακάτω.

Η επαύξηση βοήθησε αρκετά στα κανονικοποιημένα δεδομένα, ενώ στα υπόλοιπα τα αποτελέσματα ήταν μικτά. Το καλύτερο μοντέλο ήταν το linear svm στην περίπτωση χωρίς preprocessing με τα επαυξημένα δεδομένα με accuracy 77.5%. Τα όχι τόσο καλά αποτελέσματα πιθανώς να οφείλονται στο ότι τα μοντέλα που δοκιμάσαμε δεν είναι ακολουθιακά, πράγμα που κάνει δύσκολο το να ανακαλύψουν εξαρτήσεις στον χρόνο.

No preprocessing

```
Before augmenting: {'gnb': 0.5, 'cgnb': 0.5, 'svm': 0.675, 'knn': 0.525, 'rf': 0.55}  
After augmenting: {'gnb': 0.45, 'cgnb': 0.45, 'svm': 0.775, 'knn': 0.475, 'rf': 0.625}
```

Scaled

```
Before augmenting: {'gnb': 0.475, 'cgnb': 0.475, 'svm': 0.65, 'knn': 0.35, 'rf': 0.65}  
After augmenting: {'gnb': 0.5, 'cgnb': 0.5, 'svm': 0.625, 'knn': 0.45, 'rf': 0.6}
```

Normalized

```
Before augmenting: {'gnb': 0.425, 'cgnb': 0.425, 'svm': 0.05, 'knn': 0.5, 'rf': 0.5}  
After augmenting: {'gnb': 0.5, 'cgnb': 0.5, 'svm': 0.05, 'knn': 0.65, 'rf': 0.55}
```

Βήμα 8

(part1.step8)

Παρήχθησαν 1024 δείγματα και χρησιμοποιήθηκαν simple rnn, lstm, και gru.

Σε κάθε περίπτωση δοκιμάστηκαν bidirectional/unidirectional δίκτυα.

Τα αποτελέσματα των unidirectional δικτύων δεν μπορούσαν να προβλέψουν το πρώτο σημείο, ενώ τα υπόλοιπα 9 σημεία τα προέβλεπαν με ακρίβεια. Το bidirectional διόρθωσε αυτό το πρόβλημα. Στα παρακάτω γραφήματα φαίνονται τα learning curves και τα τελικά mse losses.

Όσον αφορά τον λόγο που χρησιμοποιούμε μοντέλα όπως τα LSTM και GRU, θα το γράψω στα Αγγλικά γιατί δεν είμαι εξοικειωμένος με την Ελληνική ορολογία.

LSTM and GRU are popular because they can learn long term dependencies easier. The basic problem with a simple RNN is that gradients propagated over many stage tend to either vanish or explode, with the former being the more common case. Even if we assume that the parameters are such that the recurrent network is stable, the difficulty with long-term dependencies arises from the exponentially smaller weights given to long-term interactions. Gated RNNs like LSTMs and GRUs are based on the idea of creating paths through time that have derivatives that neither vanish nor explode. They do this by adding those leaky connections from the past to now, with connection weights that may change at each time step.
Goodfellow 10.7, 10.10

Unidirectional predictions can't learn the first point

0.11	-0.50	-0.91	-0.98	-0.67	-0.11	0.50	0.91	0.98	0.67
------	-------	-------	-------	-------	-------	------	------	------	------

0.01	-0.46	-0.91	-0.99	-0.67	-0.11	0.50	0.91	0.98	0.67
------	-------	-------	-------	-------	-------	------	------	------	------

-0.86	-0.39	0.23	0.76	1.00	0.86	0.39	-0.23	-0.76	-1.00
-------	-------	------	------	------	------	------	-------	-------	-------

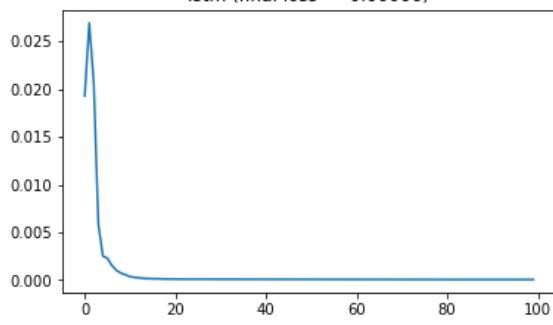
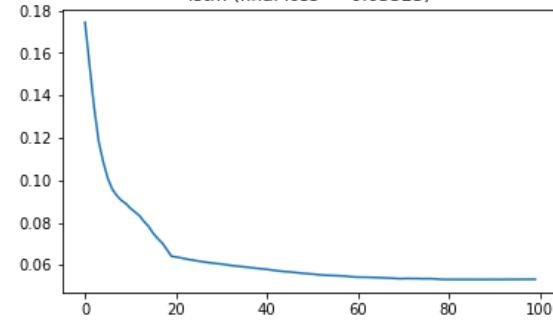
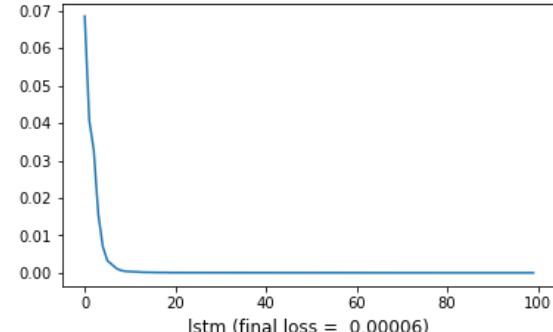
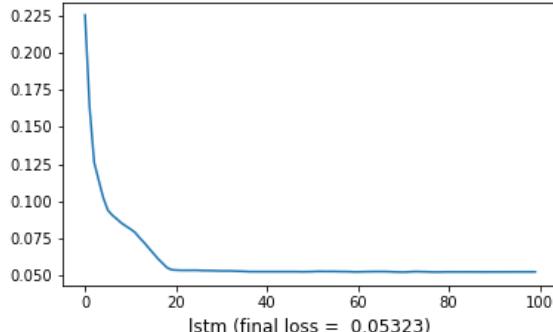
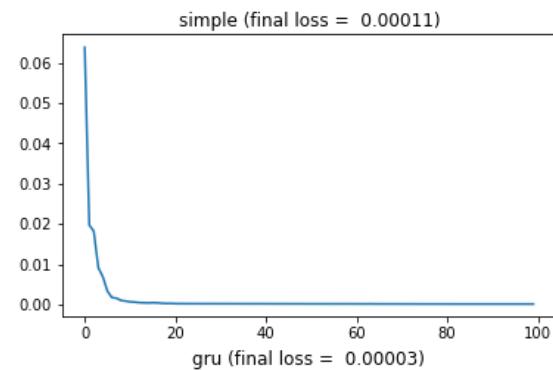
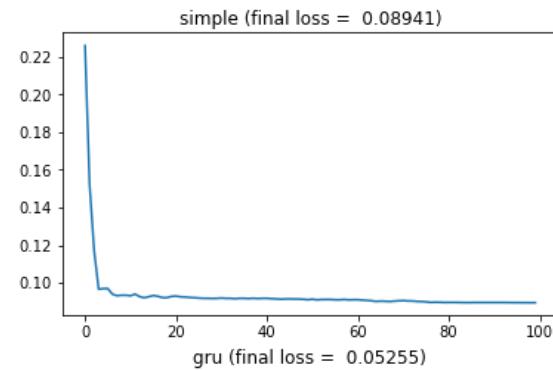
0.06	-0.36	0.22	0.75	1.00	0.85	0.39	-0.23	-0.76	-1.00
------	-------	------	------	------	------	------	-------	-------	-------

-0.07	-0.64	-0.97	-0.93	-0.53	0.07	0.64	0.97	0.93	0.53
-------	-------	-------	-------	-------	------	------	------	------	------

0.01	-0.60	-0.97	-0.93	-0.54	0.07	0.64	0.97	0.93	0.53
------	-------	-------	-------	-------	------	------	------	------	------

Unidirectional

Bidirectional



Βήμα 9

Βλέπε part2.step_9_10_11_12_13

Βήμα 10

Βλέπε part2.step_9_10_11_12_13

Βήμα 11

Βλέπε part2.step_9_10_11_12_13

Βήμα 12

(part2.step_9_10_11_12_13)

Γράφω στα αγγλικά γιατί δυσκολεύομαι με την ορολογία στα ελληνικά:

We use a validation set plus a test set, because when we choose the estimator with the best fit, we introduce an overestimating bias on the score, which might be significant.

To see this, consider the trivial case where we train and validate the same estimator multiple times and then we pick the best one of them. It's clear that the expected output of this process would not be the expected value of the accuracy, but higher than it. On the other hand, the process of evaluating the score on the test set, has an expected output equal to the true accuracy.

Βήμα 13

(part2.step_9_10_11_12_13)

Τα αποτελέσματα του Grid Search και τα confusion matrices φαίνονται παρακάτω.

BEST PARAMS: {'covariance_type': 'full', 'n_components': 3, 'n_mix': 5}

VALIDATION ACCURACY: 0.990741

TEST ACCURACY: 0.986667

```
'mean_test_score': array([0.32037037, 0.45555556, 0.53333333, 0.63148148, 0.7037037 ,  
 0.37962963, 0.57777778, 0.64814815, 0.7462963 , 0.81481481,  
 0.4     , 0.59814815, 0.8     , 0.81481481, 0.84259259,  
 0.5037037 , 0.75555556, 0.80740741, 0.89814815, 0.93148148,  
 0.68703704, 0.78518519, 0.84814815, 0.88703704, 0.91666667,  
 0.75185185, 0.85740741, 0.92407407, 0.93518519, 0.95925926,  
 0.82777778, 0.93888889, 0.97592593, 0.97407407, 0.97222222,  
 0.86851852, 0.93518519, 0.95185185, 0.97777778, 0.98703704,
```

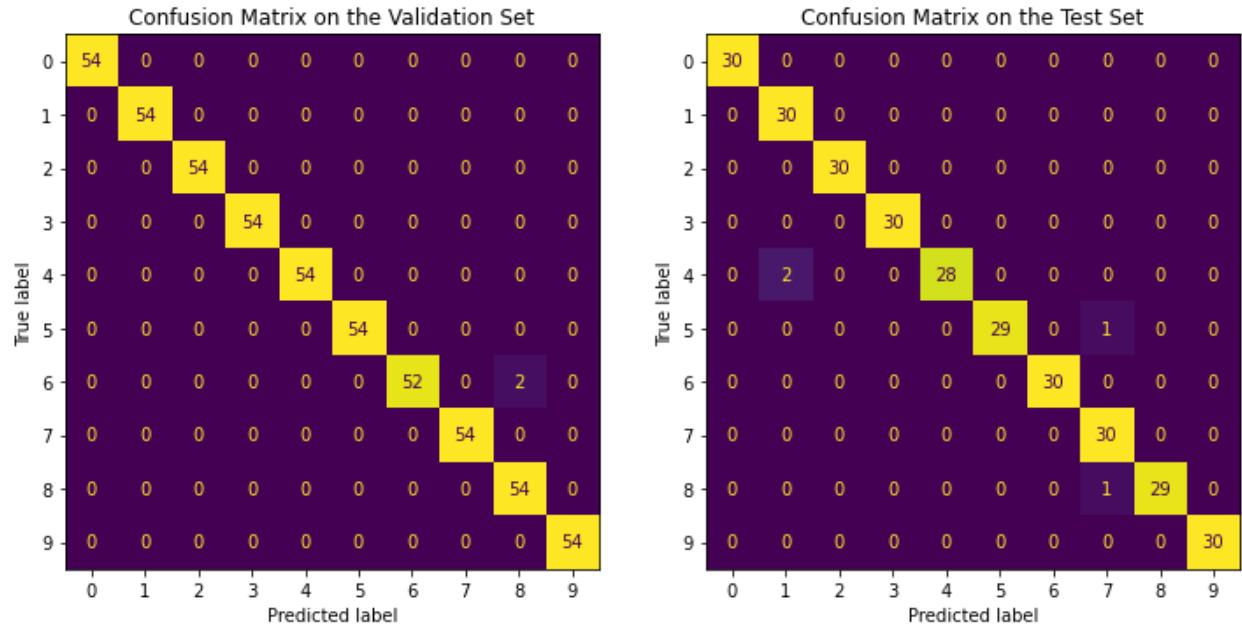
```

0.93333333, 0.95740741, 0.95    , 0.94814815, 0.96111111,
0.96296296, 0.97407407, 0.97592593, 0.97962963, 0.98333333,
0.97407407, 0.95555556, 0.98148148, 0.98333333, 0.99074074,
0.97962963, 0.96111111, 0.99074074, 0.99074074,    nan,
0.93333333, 0.94259259, 0.94814815, 0.95185185, 0.95555556,
0.96296296, 0.96666667, 0.96111111, 0.94814815, 0.98148148,
0.98703704, 0.97407407, 0.97777778, 0.98888889, 0.98148148,
0.98333333, 0.98148148, 0.97777778, 0.99074074, 0.98148148]),

'params': [{"covariance_type': 'spherical', 'n_components': 1, 'n_mix': 1},
{'covariance_type': 'spherical', 'n_components': 1, 'n_mix': 2},
{'covariance_type': 'spherical', 'n_components': 1, 'n_mix': 3},
{'covariance_type': 'spherical', 'n_components': 1, 'n_mix': 4},
{'covariance_type': 'spherical', 'n_components': 1, 'n_mix': 5},
{'covariance_type': 'spherical', 'n_components': 2, 'n_mix': 1},
{'covariance_type': 'spherical', 'n_components': 2, 'n_mix': 2},
{'covariance_type': 'spherical', 'n_components': 2, 'n_mix': 3},
{'covariance_type': 'spherical', 'n_components': 2, 'n_mix': 4},
{'covariance_type': 'spherical', 'n_components': 2, 'n_mix': 5},
{'covariance_type': 'spherical', 'n_components': 3, 'n_mix': 1},
{'covariance_type': 'spherical', 'n_components': 3, 'n_mix': 2},
{'covariance_type': 'spherical', 'n_components': 3, 'n_mix': 3},
{'covariance_type': 'spherical', 'n_components': 3, 'n_mix': 4},
{'covariance_type': 'spherical', 'n_components': 3, 'n_mix': 5},
{'covariance_type': 'spherical', 'n_components': 4, 'n_mix': 1},
{'covariance_type': 'spherical', 'n_components': 4, 'n_mix': 2},
{'covariance_type': 'spherical', 'n_components': 4, 'n_mix': 3},
{'covariance_type': 'spherical', 'n_components': 4, 'n_mix': 4},
{'covariance_type': 'spherical', 'n_components': 4, 'n_mix': 5},
{'covariance_type': 'diag', 'n_components': 1, 'n_mix': 1},
{'covariance_type': 'diag', 'n_components': 1, 'n_mix': 2},
{'covariance_type': 'diag', 'n_components': 1, 'n_mix': 3},
{'covariance_type': 'diag', 'n_components': 1, 'n_mix': 4},
{'covariance_type': 'diag', 'n_components': 1, 'n_mix': 5},
{'covariance_type': 'diag', 'n_components': 2, 'n_mix': 1},
{'covariance_type': 'diag', 'n_components': 2, 'n_mix': 2},
{'covariance_type': 'diag', 'n_components': 2, 'n_mix': 3},
{'covariance_type': 'diag', 'n_components': 2, 'n_mix': 4},
{'covariance_type': 'diag', 'n_components': 2, 'n_mix': 5},
{'covariance_type': 'diag', 'n_components': 3, 'n_mix': 1},
{'covariance_type': 'diag', 'n_components': 3, 'n_mix': 2},
{'covariance_type': 'diag', 'n_components': 3, 'n_mix': 3},
{'covariance_type': 'diag', 'n_components': 3, 'n_mix': 4},
{'covariance_type': 'diag', 'n_components': 3, 'n_mix': 5},
{'covariance_type': 'diag', 'n_components': 3, 'n_mix': 1},
{'covariance_type': 'diag', 'n_components': 3, 'n_mix': 2},
{'covariance_type': 'diag', 'n_components': 3, 'n_mix': 3},
{'covariance_type': 'diag', 'n_components': 3, 'n_mix': 4},
{'covariance_type': 'diag', 'n_components': 3, 'n_mix': 5}]]
```



```
{'covariance_type': 'tied', 'n_components': 4, 'n_mix': 5}],
```



Βήμα 14

Γράφω στα αγγλικά γιατί δυσκολεύομαι με την ορολογία στα ελληνικά:

Dropout forces the model to learn while randomly missing several of its neurons. This helps neuron overspecialization, thus making the model more robust against new data. It is a form of regularization and it can be seen as having an effect similar to L2 regularization.

L2 regularization is essentially forcing a normal prior on the weights, centered at 0 and with standard deviation $1/C$ where C is coefficient with which we multiply the sum of squares of the weights.

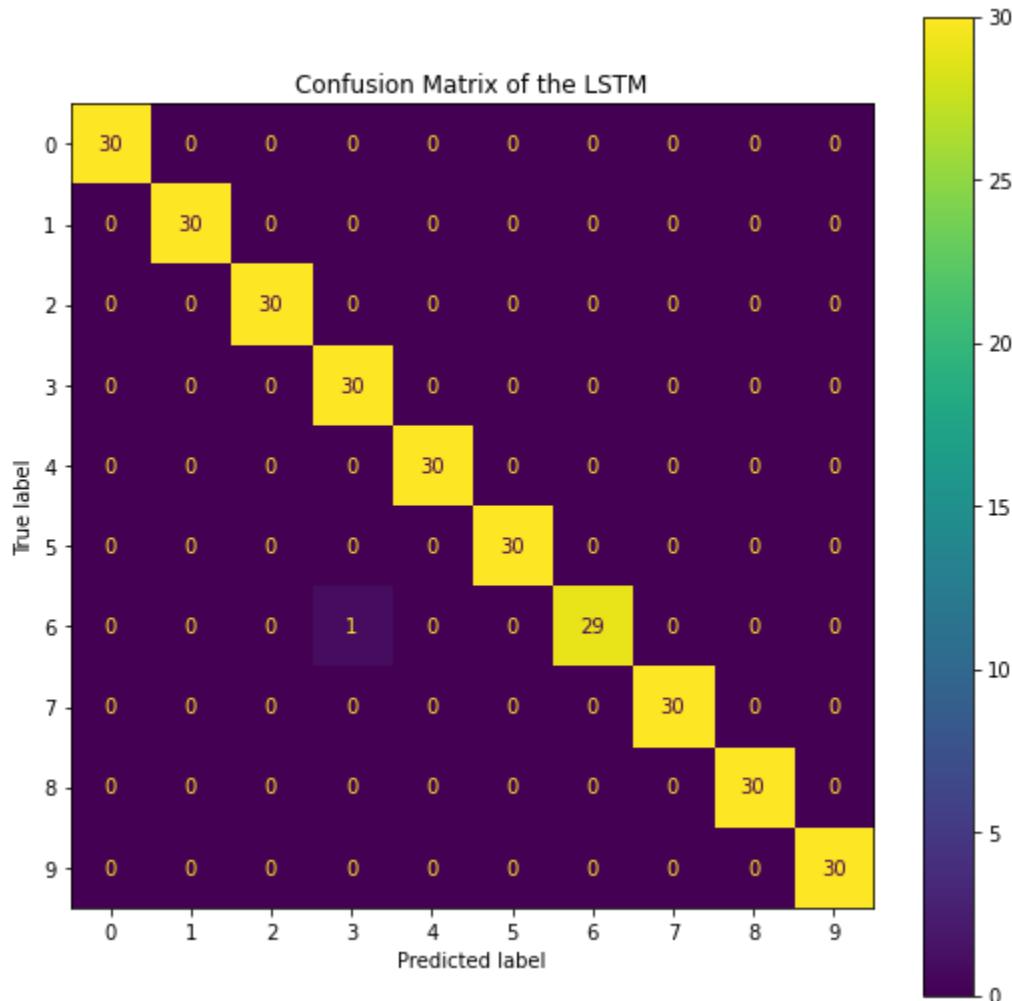
Both techniques help us prevent overfitting, by introducing more bias to the model and lowering its variance. Such regularization techniques work best on complex models / few data.

Early stopping is also a form of regularization. We prevent the model from overfitting on the training data, by monitoring its performance on a different set (the validation set). If we see the performance on that different set drop significantly and consistently, then we can assume that the model is overtraining and we stop the training procedure. The parameters we keep are those with the best validation loss.

Bidirectional RNNs are great when there are dependencies, not only from left to right, but from right to left as well. The main drawback of bidirectional RNNs is that we can't use them for online predictions, we always need the entire sequence.

Η ακρίβεια του μοντέλου είναι $299/300 = 99.667$

To Confusion Matrix και τα Learning Curves φαίνονται παρακάτω.



Learning Curves of the LSTM

