**ReadMe – Project 2021-22 – Assignment 3**

**Georgios Georgiou - Γεώργιος Γεωργίου - sdi1800220 - 1115201800220**

**Panagiotis Mavrommatis - Παναγιώτης Μαυρομμάτης - sdi1800115 – 1115201800115**

GitHub Link: <https://github.com/blackeye18/LSTM-Project-3-DIT>

Στην συγκεκριμένη εργασία χρησιμοποιήσαμε σε μεγάλο βαθμό το google colab σε συνδυασμό με το google drive, με αποτέλεσμα να μην έχουμε πολλά commits στο GitHub.

**Τίτλος:** Νευρωνικά δίκτυα LSTM – Forecast – Anomaly Detection – Encode – Decode

Τα μοντέλα που παραδίδουμε σε κάθε υποερώτημα έχουν εκπαιδευτεί με όλα τα training sets του dataset, δηλαδή με όλα τα training sets των 360 μετοχών. Οι υπερπαράμετροι με τους όποιους κάναμε train τα τελικά μας μοντέλα είναι οι τιμές που βρίσκονται σε κάθε train\_.py αρχείο αντίστοιχα.

**Περιεχόμενα φακέλων και τρόπος εκτέλεσης:**

*A\_Forecast:*

[Best\_model\_forecast.h5](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/A_Forecast%20(Needs%20comments)/best_model_forecast.h5)-αρχείο μοντέλου για το Α

[Forecast.py](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/A_Forecast%20(Needs%20comments)/forecast.py)-αρχείο πρόβλεψης

εντολή:$python forecast.py –d <dataset> -n <number of time series selected>

[Train\_forecast.py](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/A_Forecast%20(Needs%20comments)/train_forecast.py)-αρχείο παραγωγής μοντέλου

εντολή: $python train\_forecast.py –d <dataset> -n <number of time series selected> <OPTIONAL: name of model to save>

[Nasdaq2007\_17.csv](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/A_Forecast%20(Needs%20comments)/nasdaq2007_17.csv)-αρχείο dataset απο το eclass

*B\_Detect:*

[model\_detect\_best.h5](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/B_Detect%20(Needs%20coments)/model_detect_best.h5)-αρχείο μοντέλου για το B

[detect.py](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/B_Detect%20(Needs%20coments)/detect.py)-αρχείο πρόβλεψης

εντολή:$python detect.py –d <dataset> -n <number of time series selected> -mae <error value as double>

[train\_detect.py](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/B_Detect%20(Needs%20coments)/train_detect.py)-αρχείο παραγωγής μοντέλου

εντολή:$python train\_detect.py –d <dataset> -n <number of time series selected> <OPTIONAL: name of model to save>

[Nasdaq2007\_17.csv](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/A_Forecast%20(Needs%20comments)/nasdaq2007_17.csv)-αρχείο dataset απο το eclass

G\_Reduce:

[encoder1.h5](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/G_Reduce/encoder1.h5)-αρχείο μοντέλου για το Γ

[reduce.py](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/G_Reduce/reduce.py)-αρχείο εκτύπωσης σε csv αρχεία

εντολή:$python reduce.py –d <dataset> -q <queryset> -od <output\_dataset\_file> -oq <output\_query\_file>

[train\_reduce](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/G_Reduce/train_reduce.py)[.py](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/B_Detect%20(Needs%20coments)/train_detect.py)-αρχείο παραγωγής μοντέλου

εντολή: #$python train\_reduce.py –d <dataset> -n <number of time series selected> <OPTIONAL: name of model to save>

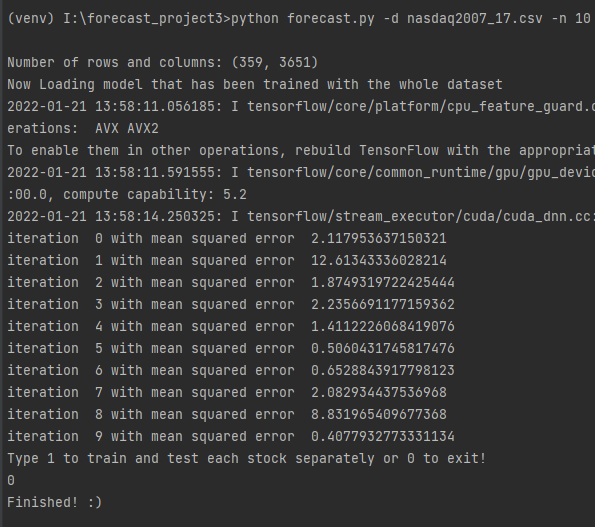
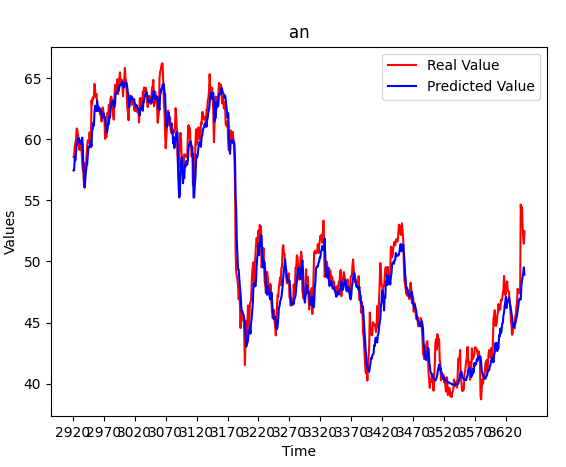
[Nasdaq2007\_17.csv](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/A_Forecast%20(Needs%20comments)/nasdaq2007_17.csv),[nasdaq2007\_17\_fretchet.csv](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/G_Reduce/nasdaq2007_17_fretchet.csv)-αρχεία για πειράματα

[Output\_dataset\_file.csv](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/G_Reduce/output_dataset_file.csv),[output\_dataset\_file\_frechet.csv](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/G_Reduce/output_dataset_file_frechet.csv),[output\_query\_file.csv](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/G_Reduce/output_query_file.csv),[output\_query\_file\_frechet.csv](https://github.com/blackeye18/LSTM-Project-3-DIT/blob/main/G_Reduce/output_query_file_frechet.csv)-αρχεία που παράγονται από τα αρχεία πειράματος.

**Περιγραφή:**

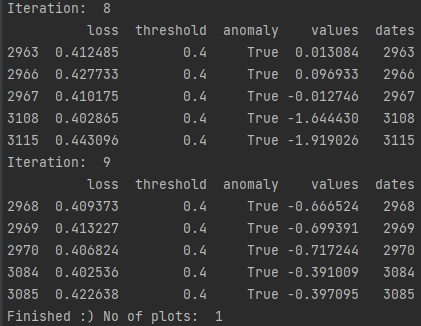
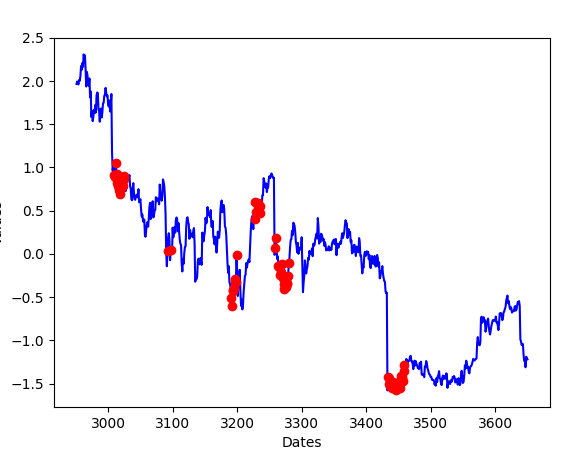
**A Forecast**: Σε αυτό το υποερώτημα αναπτύξαμε αναδρομικό νευρωνικό δίκτυο LSTM πολλαπλών στρωμάτων. Σύμφωνα με τα πειράματα μας, για να έχουμε βέλτιστα αποτελέσματα, μειώσαμε τα epoch στα 50, από 100, και τα units μειώνονται σε κάθε layer(60, 50, 50, 40). Επιπλέον αυξήσαμε κατά 0.1 το dropout στο τελευταίο layer. Καλώντας το forecast.py γίνεται load το μοντέλο και κάνουμε predict χρησιμοποιώντας το τελευταίο 20% των μετοχών. Τα αποτελέσματα εμφανίζονται στο terminal. Επιπλέον εμφανίζονται δειγματοληπτικά, το πολύ 10% του n, γραφικές παραστάσεις. Στο τέλος, ο χρήστης έχει την επιλογή να κάνει train και test κάθε μετοχή σε ξεχωριστό μοντέλο.

Παράδειγμα εκτέλεσης με n=10:



**B Detect:**

Σε αυτό το υποερώτημα αναπτύξαμε αναδρομικό νευρωνικό δίκτυο LSTM αυτοκωδικοποιησης χρονοσειρών με στρώματα κωδικοποίησης και αποκωδικοποίησης. Σύμφωνα με τα πειράματα μας για να έχουμε βέλτιστα αποτελέσματα χρησιμοποιήσαμε περισσότερα epoch (25) και τα units μειώνονται σε καθε layer, όπως και στο ερώτημα Α. Τα αποτελέσματα εμφανίζονται στο terminal. Επιπλέον εμφανίζονται δειγματοληπτικά, το πολύ 10% του n, γραφικές παραστάσεις.



**Γ Reduce:**

Σε αυτό το υποερώτημα αναπτύξαμε νευρωνικό μοντέλο συμπίεσης και αποσυμπίεσης διαστάσεων (train\_reduce.py). Για να το πετύχουμε αυτό χωρίζουμε το σύνολο των τιμών μας σε 10δες στην μέση του μοντέλου(encoder), οι 10δες γίνονται 3αδες και ακολούθως κάνουμε αποσυμπίεση των διαστάσεων, δηλαδή γίνονται και παλι 10δες. Αυτό το κάνουμε έτσι ώστε να μπορούμε να συγκρίνουμε τα δεδομένα στο τέλος με τα αρχικά και να υπολογίσουμε την απόκλιση τους. Όσο μικρότερη απόκλιση τόσο πιο ιδανική συμπίεση γίνεται. Στο τέλος αποθηκεύουμε μόνο τον encoder μιας και αυτός μας χρειάζεται για την συμπίεση. Επίσης, προσθέσαμε ένα debug flag στο train\_reduce.py το οποίο όταν είναι ενεργοποιημένο, δηλαδή έχει τιμή 0, εμφανίζονται κάποιες γραφικές παραστάσεις που δείχνουν την αρχική γραφική παράσταση και την γραφική παράσταση μετά από το encode – decode. Τις συγκεκριμένες γραφικές παραστάσεις τις χρησιμοποιήσαμε για την επιλογή των καλύτερων υπερπαραμέτρων.

Στην συνέχεια (reduce.py), κάνουμε load το μοντέλο συμπίεσης και γράφουμε τα συμπιεσμένα δεδομένα σε νέα csv αρχεία, τα οποία χρησιμοποιούμε για το ερώτημα Δ όπως ζητείται.

**Δ Σύγκριση Αποτελεσμάτων:**

Σχετικά με τον χρόνο εκτέλεσης:

Παρατηρούμε ότι σε όλες τις περιπτώσεις υπάρχει μεγάλη μείωση του χρόνου εκτέλεσης με τα αρχεία που προκύπτουν από το reduce.py σε σχέση με τα αρχικά αρχεία. Πιο συγκεκριμένα, κατά μέσο όρο υπάρχει περίπου υποδεκαπλασιασμός του χρόνου εκτέλεσης. Στην περίπτωση του Search με τον αλγόριθμο LSH\_Vector, η μείωση των συντεταγμένων είχε ως αποτέλεσμα να κάνει την πιο συμφέρον τον brute force αλγόριθμο, καθώς επηρεάστηκε περισσότερο από ότι η μέθοδος LSH. Αυτό θεωρούμε πως συμβαίνει λόγω του χρόνου που απαιτείται για την δημιουργία και αρχικοποίηση των δομών του LSH.

Σχετικά με τα αποτελέσματα:

Παρατηρούμε ότι τα αποτελέσματα δεν είναι ακριβώς ίδια και διαφέρουν, συνήθως κατά 10%, είτε προς το καλύτερο είτε προς το χειρότερο. Θεωρούμε πως αυτή η διαφορά προκύπτει από την τυχαιότητα που έχουν οι αλγόριθμοι μας στην 2η εργασία, LSH kmeans++ , και δεν έχουν σχέση με την μείωση των συντεταγμένων. Δεδομένου λοιπόν την τυχαιότητα των αλγορίθμων θεώρουμε πως τα αποτελέσματα είναι «ίδια».

Εν κατακλείδι: Λαμβάνοντας υπόψιν τις παραπάνω παρατηρήσεις, η μείωση των συντεταγμένων είχε ως αποτέλεσμα την μείωση του χρόνου εκτέλεσης των προγραμμάτων χωρίς να χάνουμε σημαντική, αν όχι καθόλου, ακρίβεια στα αποτελέσματα.

Σημείωση: Στο clustering με assignment Classic και update MeanCurve καθώς και στο search με algorithm Frechet και metric discrete και continuous, χρησιμοποιήθηκε ένα μικρότερο αρχείο που περιέχει 20 μετοχές. Και σε αυτές τις περιπτώσεις παρατηρήθηκαν ίδιας τάξης μειώσεις.

|  |  |  |
| --- | --- | --- |
| Clustering: LSH Frechet | | |
|  | Before Reduce | After Reduce |
| Clustering Time: | 957.089 | 37.1045 |
| Total Silhouette: | 0.098619 | 0.118806 |

|  |  |  |
| --- | --- | --- |
| Clustering: LSH Vector | | |
|  | Before Reduce | After Reduce |
| Clustering Time: | 0.402157 | 0.10854 |
| Total Silhouette: | 0.124351 | 0.100955 |

|  |  |  |
| --- | --- | --- |
| Clustering: Hypercube | | |
|  | Before Reduce | After Reduce |
| Clustering Time: | 0.137885 | 0.032506 |
| Total Silhouette: | 0.00605375 | 0.0604349 |

|  |  |  |
| --- | --- | --- |
| Clustering: Classic Vector | | |
|  | Before Reduce | After Reduce |
| Clustering Time: | 0.021035 | 0.003601 |
| Total Silhouette: | [0.99938,-nan,-nan] | 0.0244257 |

Πριν το reduce το πρόγραμμα μας με τον αλγόριθμο Lloyds και update MeanVector έβαζε όλα τα vector στο ίδιο cluster με αποτέλεσμα να μην μπορεί να υπολογιστεί το συνολικό silhouette. Αυτό μάλλον συμβαίνει λόγω του πολύ μεγάλου πλήθους σημείων που έχει κάθε vector στο συγκεκριμένο αρχείο καθώς με κανένα από τα προηγούμενα αρχεία που μας είχαν δοθεί δεν αντιμετωπίσαμε κάποιο παρόμοια πρόβλημα.

|  |  |  |
| --- | --- | --- |
| Clustering: Classic Frechet | | |
|  | Before Reduce | After Reduce |
| Clustering Time: | 15.8839 | 1.81491 |
| Total Silhouette: | 0.552282 | 0.296283 |

|  |  |  |
| --- | --- | --- |
| Search: LSH Vector | | |
|  | Before Reduce | After Reduce |
| tApproximateAverage: | 0.00109351 | 0.000848975 |
| tTrueAverage: | 0.00230392 | 0.000652911 |

|  |  |  |
| --- | --- | --- |
| Search: LSH Discrete | | |
|  | Before Reduce | After Reduce |
| tApproximateAverage: | 1.04401 | 0.107782 |
| tTrueAverage: | 5.2343 | 0.439613 |

|  |  |  |
| --- | --- | --- |
| Search: LSH Continuous | | |
|  | Before Reduce | After Reduce |
| tApproximateAverage: | 1.83276 | 0.349294 |
| tTrueAverage: | 226.35 | 66.4738 |

|  |  |  |
| --- | --- | --- |
| Search: Hypercube Vector | | |
|  | Before Reduce | After Reduce |
| tApproximateAverage: | 0.000355827 | 0.000184423 |
| tTrueAverage: | 0.00228811 | 0.0014192 |