Ανάπτυξη λογισμικού για αλγοριθμικά προβλήματα

Μέρος 3ο

Έχουν υλοποιηθεί όλα τα ζητούμενα της εργασίας. Οι παραδοχές οι οποίες χρειάστηκαν για την υλοποίηση της εργασίας αναφέρονται παρακάτω, όπως και ενδεικτικές εκτελέσεις μαζί με τα αποτελέσματά τους.
Η εργασία υλοποιήθηκε με χρήση της online πλατφόρμας Colab της Google και έγιναν επίσης δοκιμές σε τοπικό περιβάλλον χρησιμοποιώντας το προγραμματιστικό περιβάλλον Anaconda.

Εγκάτασταση περιβάλλοντος ανάπτυξης - Απαιτούμενες βιβλιοθήκες

1. Εγκατάσταση του προγραμματιστικού περιβάλλοντος Anaconda, το οποίο είναι διαθέσιμο για τα λειτουργικά συστήματα Linux, Windows, macOS. Στην συνέχεια μέσω της εντολής:
   conda env create --file project3_env.yml
   μπορεί να γίνει η εγκατάσταση του περιβάλλοντος και να εγκατασταθούν αυτόματα οι απαραίτητες βιβλιοθήκες (packages) για την εκτέλεση του προγράμματος.

2. Χρήση των online πλατφορμών Google Colab ή Kaggle, οι οποίες είναι έτοιμες προς χρήση με τις εγκαταστημένες βιβλιοθήκες για deep learning

Εκτέλεση των αρχείων

A) python forecast.py -d nasdaq2007_17.csv -n 10
B) python detect.py -d nasdaq2007_17.csv -n 10 -m 0.5
C) python reduce.py -d nasdaq_input.csv -q nasdaq_query.csv

Δομή αρχείων

- **Forecast/forecast.py :** Ο πηγαίος κώδικας για το πρώτο ερώτημα
- **Forecast/pre_trained_models/per_timeseries/ :** Περιέχει 100 προ-εκπαιδευμένα μοντέλα για την περίπτωση της ανά χρονοσειράς. Τα παραγόμενα μοντέλα είναι της μορφής .h5
- **Forecast/pre_trained_models/per_set_of_timeseries/ :** Περιέχει 1 προ-εκπαιδευμένο μοντέλο με όλες τις χρονοσειρές μαζί για την περίπτωση του συνόλου χρονοσειρών. Το παραγόμενο μοντέλο είναι της μορφής .h5
- **Detect/detect.py :** Ο πηγαίος κώδικας για το δεύτερο ερώτημα
- **Detect/pre_trained_models/ :** Περιέχει 1 προ-εκπαιδευμένο μοντέλο με όλες τις χρονοσειρές μαζί για την περίπτωση του συνόλου χρονοσειρών. Το παραγόμενο μοντέλο είναι της μορφής .h5
- **Convolutional_autoencoder/reduce.py :** Ο πηγαίος κώδικας για το τρίτο ερώτημα
  Convolutional_autoencoder/pre_trained_models/ : Περιέχει 1 προ-εκπαιδευμένο μοντέλο με όλες τις χρονοσειρές μαζί για την περίπτωση του συνόλου χρονοσειρών.Το παραγόμενο μοντέλο είναι της μορφής .h5
- **project3_env.yml :** Αρχείο σε μορφή YAML για την εγκατάσταση των απαραίτητων βιβλιοθηκών για την εγκάτασταση του περιβάλλοντος εκτέλεσης.

Ενδεικτικά αποτελέσματα για κάθε ερώτημα - Συμπεράσματα

Σημ: Έπειτα από κάθε layer που προσθέτουμε, βάζουμε και τον παράγοντα Dropout με rate=0.2. Αυτό μας βοηθά να αποφύγουμε το overfitting. Η χρήση του Dropout έχει ως αποτελέσμα να προσθέτει στην διαδικασία εκπαίδευσης παραπάνω θόρυβο, αναγκάζοντας τους κόμβους μέσα σε ένα layer να αναλάβουν πιθανολογικά περισσότερη ή λιγότερη επικοινωνία με το dataset που εκπαιδεύεται.

A)

Εκπαιδεύσαμε τα μοντέλα μας για τις δύο περιπτώσεις εκπαίδευσης. Στην περίπτωση της εκπαίδευσης ανά χρονοσειράς έχουμε προ-εκπαιδεύσει και αποθηκεύσει 100 χρονοσειρές, ενώ για την περίπτωση της εκπαίδευσης του συνόλου χρονοσειρών έχουμε προ-εκπαιδεύσει το μοντέλο με όλες τις χρονοσειρές.

- Ανά χρονοσειρά: Έπειτα από πειραματισμούς και δοκιμές στις τιμές των υπερπαραμέτρων καταλήξαμε στην χρήση συνολικά 4 layers με 64 units το καθένα, 50 epochs, batch size = 256 για την ελαχιστοποίηση του loss.

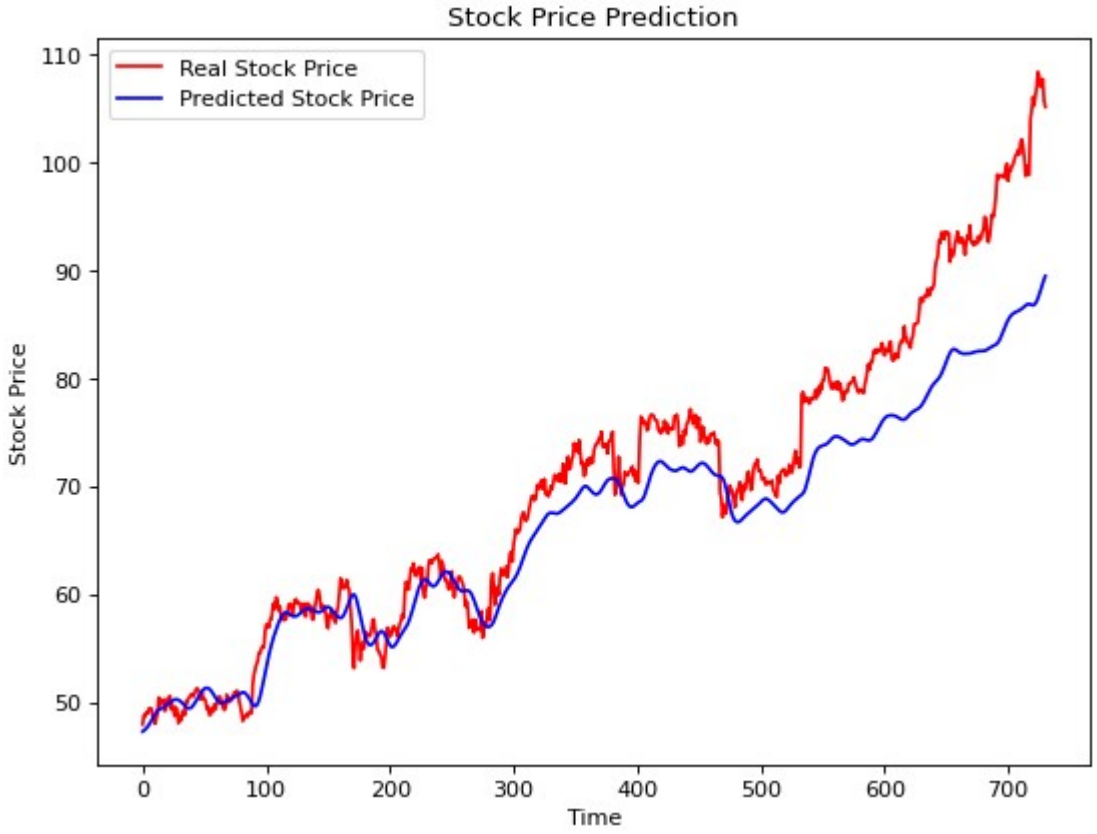
- Ανά σύνολο: Έπειτα από πειραματισμούς και δοκιμές στις τιμές των υπερπαραμέτρων καταλήξαμε στην χρήση συνολικά 2 layers με 64 units το καθένα, 100 epochs, batch size = 1024 για την ελαχιστοποίηση του loss.

Παρακάτω έχουμε κάνει δύο παραδείγματα για να αναδείξουμε τις δυνατότητες του κάθε μοντέλου.

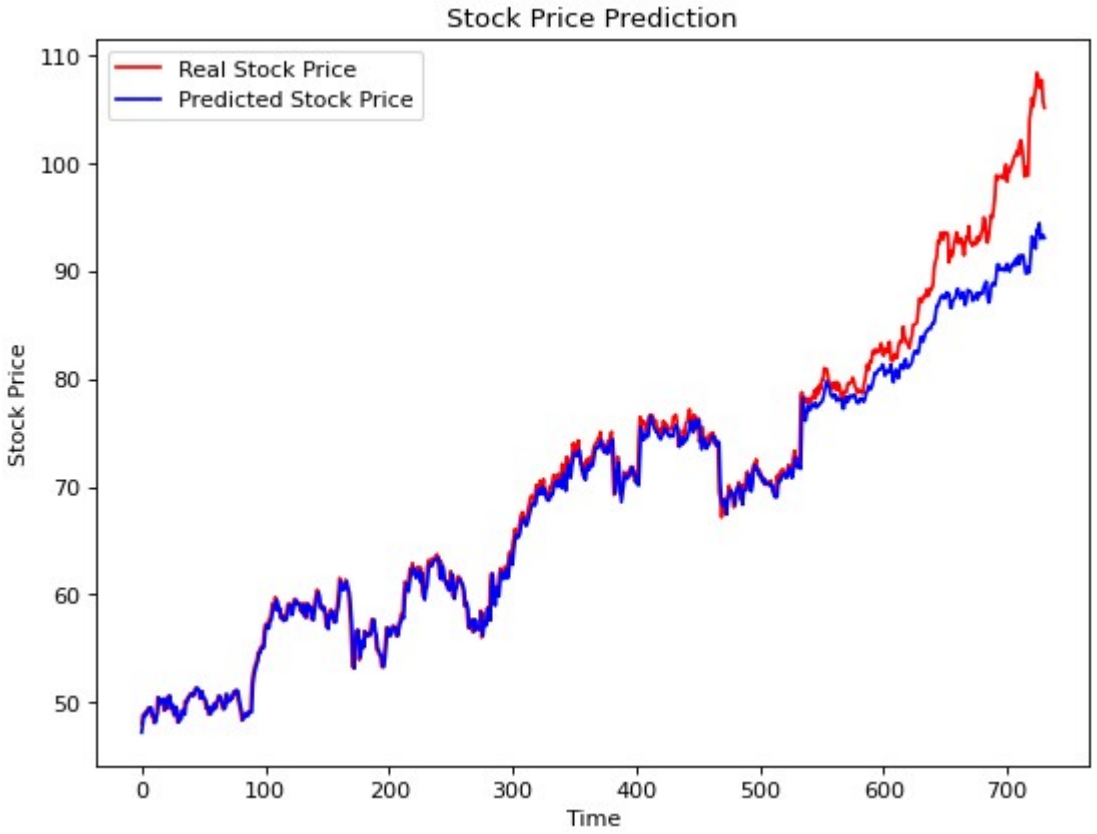
Συμπέρασμα: Παρατηρούμε ότι καθώς στην περίπτωση της εκπαίδευσης ανά συνόλου χρονοσειρών έχουν εκπαιδευτεί όλες οι χρονοσειρές και προκύπτει ένα πιο γενικό και μεγαλύτερο σύνολο σε σχέση με την περίπτωση της εκπαίδευσης ανά χρονοσειράς που η εκπαίδευση γίνεται μεμονωμένα σε μία μόνο χρονοσειρά την φορά, είναι αναμενόμενο ότι η περίπτωση της εκπαίδευσης του συνόλου χρονοσειρών αποφέρει καλύτερα αποτελέσματα. Το συμπέρασμα αυτό επιβεβαιώνεται από την θεωρία των νευρωνικών δικτύων, καθώς όταν το μοντέλο εκπαιδεύεται και εκτίθεται σε περισσότερα δεδομένα μπορεί να κάνει καλύτερες προβλέψεις. Επομένως, η γενική εκπαίδευση που έχει υποστεί το μοντέλο στην περίπτωση της εκπαίδευσης ολόκληρου του dataset αποδίδει με καλύτερο τρόπο τις τιμές των μετοχών.

Πρώτο παράδειγμα:

Εκπαίδευση ανά χρονοσειρά

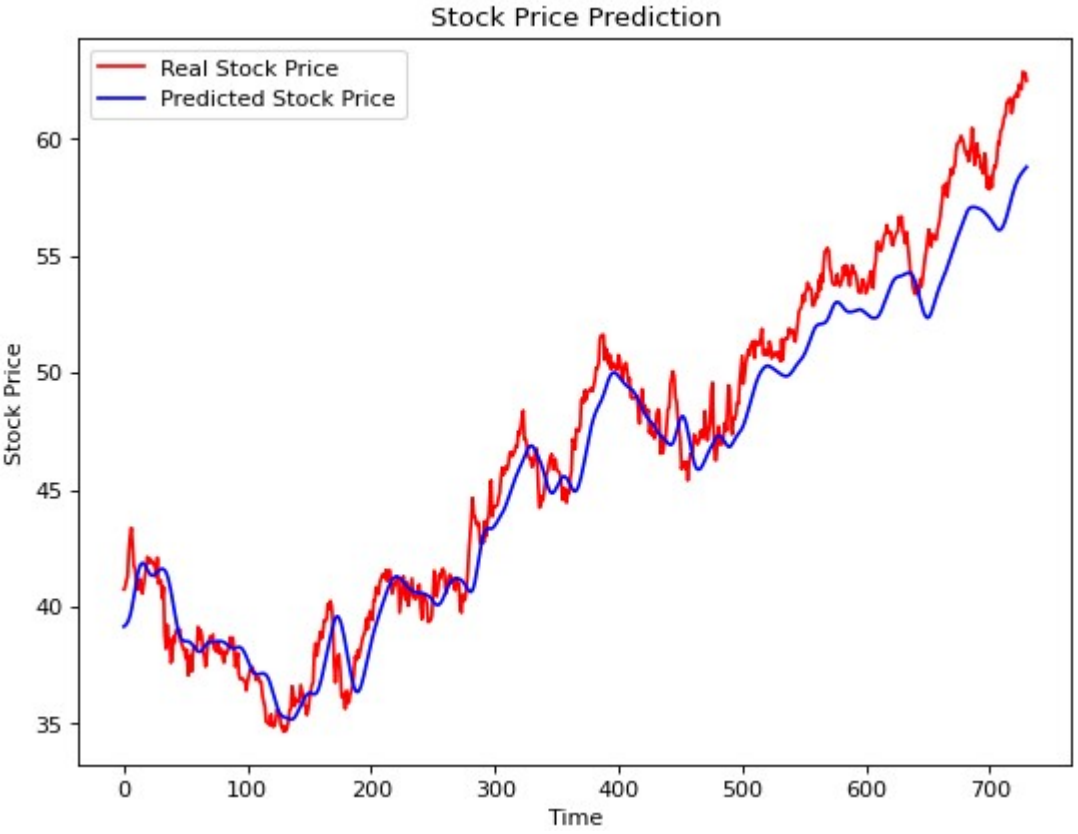


Εκπαίδευση ανά σύνολο χρονοσειρών

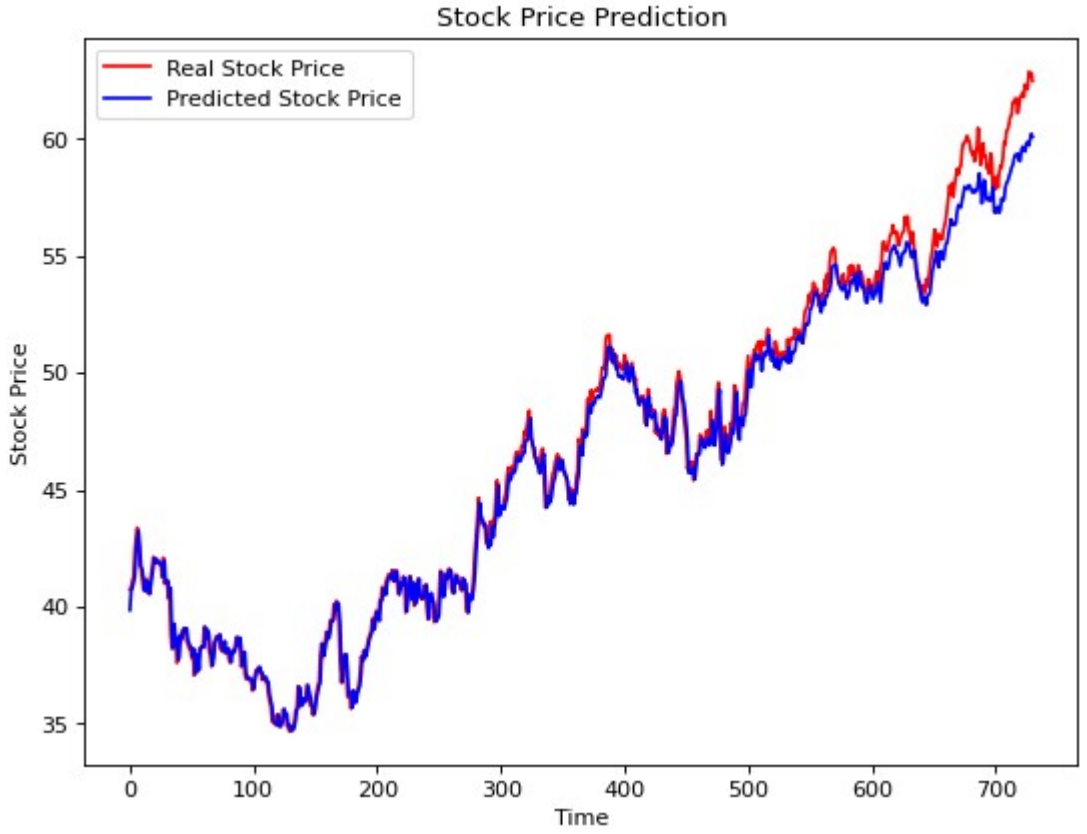


Δεύτερο παράδειγμα:

Εκπαίδευση ανά χρονοσειρά

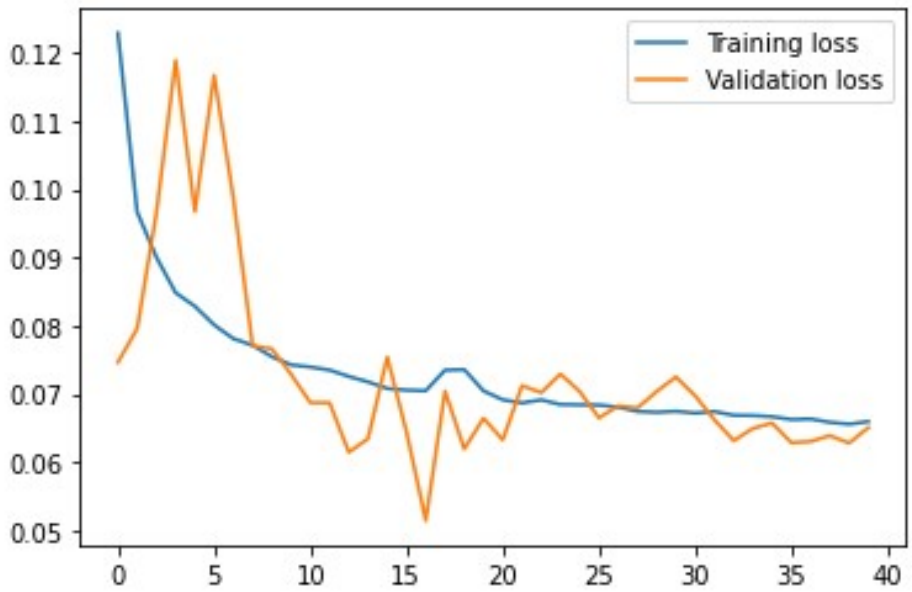


Εκπαίδευση ανά σύνολο χρονοσειρών



B)

Έχουμε πειραματιστεί αρκετά με τις υπερπαραμέτρους των μοντέλων κατά την εκπαίδευση τους και δώσαμε ιδιαίτερη σημασια ως προς το traininig loss και validation loss ώστε να έχουμε χαμηλό σφάλμα και το validation loss να τείνει στο μηδέν. Οι δύο τιμές χρείαζεται να ειναι χαμηλες και να συγκλίνουν για να αποφύγουμε την περίπτωση του overfitting, καθώς επίσης χρειάζεται το training loss να είναι χαμηλότερο του validation loss για να αποφύγουμε και την περίπτωση του underfitting.
Για να το πετύχουμε αυτό κατά την διάρκεια των πειραμάτων κάναμε learning curves με την εκπαίδευση ανά epoch και τυπώναμε γραφήγματα με τις αντίστοιχες καμπύλες των training loss και validation loss. Παρατηρήσαμε ότι όσο προχωρούσαν τα epochs αρχικά οι τιμές θα συγκλίνουν και μετά θα αποκλίνουν. Όταν ξεκινήσουν να αποκλίνουν ξεκινάει το overfitting. Για αυτό τον λόγο προσέξαμε να μην αποκλίνουν οι τιμές και το validation loss να τείνει στο μηδέν.
Κατά την διαδικασία του fit βάλαμε επιπλέον την παράμετρο validation_split = 0.2 , έτσι ώστε να παίρνουμε τις τιμές του validation loss και training loss και να μπορούμε να τυπώσουμε την αντίστοιχη γραφική παράσταση. Καθώς όπως γνωρίζουμε το training loss υποδεικνύει πόσο καλά μπορεί το μοντέλο να κάνει fit το trainging set και το validation loss υποδεικνύει πόσο καλά το μοντέλο μπορεί να τα πάει σε νέα δεδομένα. Στην παρακάτω γραφική παράσταση παρατηρούμε ότι το loss από την αρχή κυμαίνεται σε χαμηλό επίπεδο, καθώς ξεκινάει από το 0.12 και όσο προχωράνε οι εποχές (συνολικά 40 εποχές) οι δύο τιμές μικραίνουν και συγκλίνουν.



Έπειτα από πειραματισμούς και δοκιμές στις τιμές των υπερπαραμέτρων καταλήξαμε στην χρήση συνολικά 4 layers με 64 units το καθένα, 40 epochs, batch size = 64, validation_split = 0.2 για την ελαχιστοποίηση του loss.
Παρακάτω έχουμε τυπώσει για κάθε χρονοσειρά δύο γραφικές παραστάσεις:

1. Απεικονίζεται test loss και το threshold
2. Απεικονίζεται η τιμή της μετοχής και μαρκάρεται η ανωμαλία

Ανίχνευση ανωμαλιών: 46 σημεία

Ανίχνευση ανωμαλιών: 36 σημεία

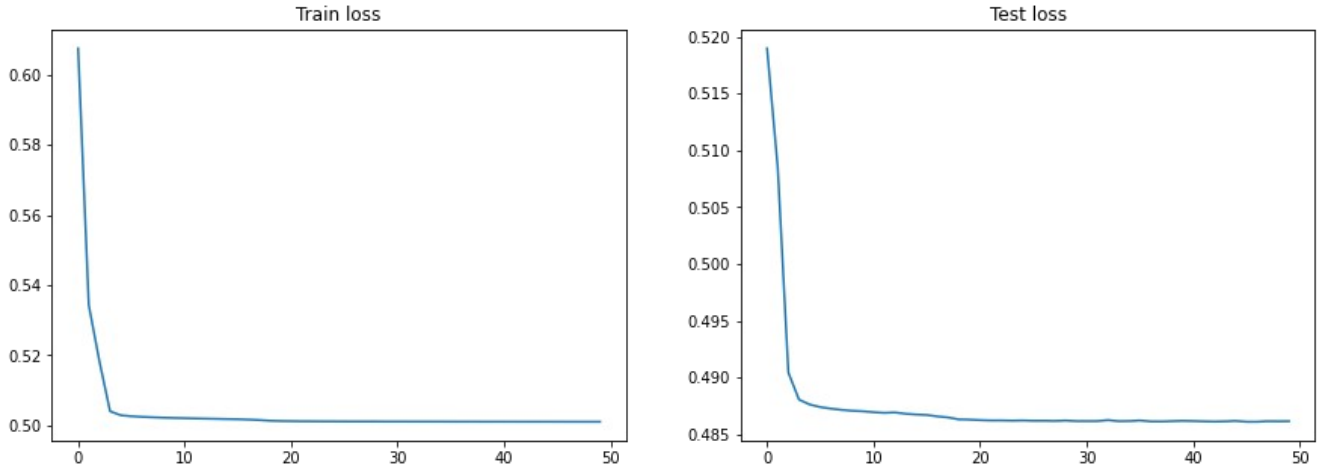
Γ)

Για το συνελικτικό νευρωνικό δίκτυο αυτοκωδικοποίησης , χρησιμοποιήσαμε ένα μοντέλο αρκέτα παρόμοιο με του tutorial που μας δόθηκε. Έγιναν αρκετές προσπάθειες παραμετροποίησης , από το μέγεθος του παραθύρου και του dim, εως τον αριθμό των conv και maxpooling layers, αλλά τελικά παρατηρήσαμε τα καλύτερα αποτελέσματα σύμφωνα με το αρχικό μοντέλο. Αυτό αποτελείται από 2 conv1d layers ακολουθούμενα απο max_pooling layers για την επιτυχή κωδικοποίηση των χρονοσειρών και ύστερα 2 conv1d layers ακολουθούμενα απο up_sampling για την προσπάθεια δημιουργίας/επανάκτησης της αρχικής χρονοσειράς.
Παρακάτω παραθέτουμε το summary του μοντέλου.

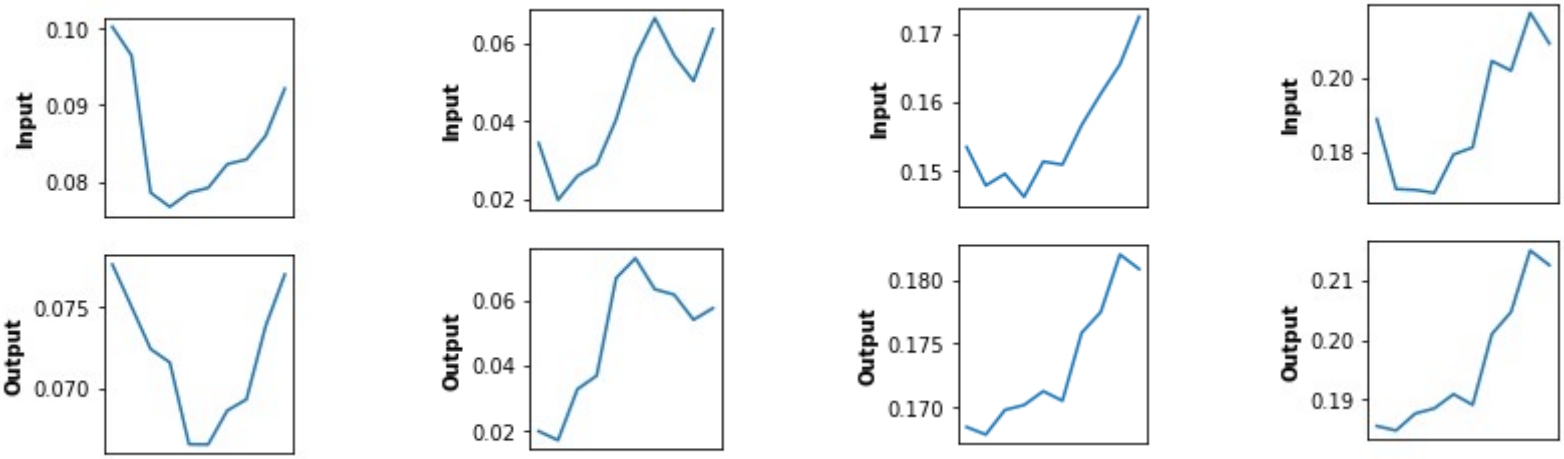
```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 10, 1)]           0

 conv1d (Conv1D)             (None, 10, 16)            64

 max_pooling1d (MaxPooling1D  (None, 5, 16)            0
 )

 conv1d_1 (Conv1D)           (None, 5, 1)              49

 max_pooling1d_1 (MaxPooling  (None, 3, 1)             0
 1D)

 conv1d_2 (Conv1D)           (None, 3, 1)              4

 up_sampling1d (UpSampling1D  (None, 6, 1)             0
 )

 conv1d_3 (Conv1D)           (None, 5, 16)             48

 up_sampling1d_1 (UpSampling  (None, 10, 16)           0
 1D)

 conv1d_4 (Conv1D)           (None, 10, 1)             49

=================================================================
Total params: 214
Trainable params: 214
Non-trainable params: 0
```

Για την επεξεργασία των δεδομένων κάναμε την εξής προετοιμασία:
Αρχικά χωρισαμε τα δεδομένα σε train και test κομματια (πρώτο 80% του αρχειου – τελευταίο 20% του αρχείου).Έπειτα κάναμε scaling στα δεδομένα κάθε χρονοσειράς. Χωρίσαμε κάθε χρονοσειρά σε 10αδες (όσο είναι το παράθυρο για την εισαγωγή) και ενώσαμε όλες τις 10αδες σε ένα array ώστε να γίνει 1 fit.
Ενδεικτικές παραμετροι ειναι: batch_size:256 , epochs:50

Κατά την διαρκεία του fit υπολογίστηκαν τα train/test loss ώστε να μπορούμε να τα συγκρίνουμε.Καθώ βλέπουμε τις εικόνες παρατηρούμε ότι και οι 2 ακολουθουν παρόμοια πορεία και κατά την ολοκλήρωση της τελευταίας εποχής εχούν και οι 2 παραστάσεις παρόμοιες τιμές.



Εδώ παρατηρούμε μερικά ενδεικτικά αποτελέσματα.Είναι φανερό ότι ο autoencoder παρόλου που πετάει πάνω απο τα 2/3 της πληροφοριάς καταφέρνει να ανακατασκευάσει με σχετικά μεγάλη ακρίβεια τις αρχικές χρονοσειρές. Ακόμα και όταν δεν πετυχαίνει ακριβώς τις τιμές , πετυχαίνει το σχήμα.



Έπειτα χρησιμοποιείται ο encoder του εκπαιδευμένου μοντέλου στο πρόγραμμα reduced για να κωδικοποιήση τις χρονοσειρές που θα συγκρίνουμε στο ερώτημα Δ.

Παρατηρήσεις – Σχόλια

- Χρησιμοποιήθηκε το προγραμματιστικό εργαλείο git για την ομαλή συνεργασία των μελών της ομάδας.
- Σε κάθε αρχείο υπάρχουν τα απαραίτητα επεξηγηματικά σχόλια στα σημεία που έχει κριθεί απαραίτητο, τα οποία διευκολύνουν την ανάγνωση και κατανόηση του κώδικα.
- Η εργασία υλοποιήθηκε με χρήση της online πλατφόρμας Colab της Google και έγιναν επίσης δοκιμές σε τοπικό περιβάλλον χρησιμοποιώντας το προγραμματιστικό περιβάλλον Anaconda.