Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα

3η Προγραμματιστική Εργασία

Research

Εισαγωγή

Στόχος του παρακάτω εγγράφου, είναι να σας παρουσιάσουμε την μελέτη μας πάνω στον κώδικα που έχουμε υλοποιήσει, στο τι σημαίνει η κάθε παράμετρος, στον τρόπο με τον οποίο επηρεάζει τα αποτελέσματα μας, αλλά και να δώσουμε τις δικές μας παραμέτρους με τις οποίες παρατηρήσαμε ότι γίνεται η καλύτερη εκπαίδευση του μοντέλου. Προφανώς, για να κάνουμε συγκρίσεις θα πρέπει να κρατάμε ένα σύνολο παραμέτρων σταθερών και να πειραματιζόμαστε με μία έτσι ώστε να παρατηρήσουμε τα αποτελέσματα. Όλες οι παρατηρήσεις και τα αποτελέσματα συνοδεύονται από γραφικές παραστάσεις, για να γίνονται πιο κατανοητά τα αποτελέσματα και το τι σχολιάζουμε.

Ερώτημα Α

Το συγκεκριμένο ερώτημα αποτελείται από δύο σκέλη. Στο 1^{ο} θα παρουσιάσουμε την καλύτερη παραμετροποίηση που πετύχαμε πειραματικά, ενώ στο 2^{ο} σκέλος θα προχωρήσουμε σε συγκρίσεις με αλλαγές στις υπερπαραμέτρους για να δούμε πως επηρεάζουν τα αποτελέσματα.

Πρώτο σκέλος

Στο στάδιο αυτό θα σας παρουσιάσουμε τις μεταβλητές με τις οποίες βρήκαμε ότι το μοντέλο αυτό εκπαιδεύεται καλύτερα χωρίς να γίνεται overfitting. Για το μοντέλο χρησιμοποιήσαμε 1 στρώματα(Layers), κάθε στρώμα έχει 50 νευρώνες(units), και κάναμε χρήση ενός(1) Dropout. Ο optimizer είναι SGD. Τέλος, χρησιμοποιούμε epochs=30(αν και σταματάει την 22^{η} φορά μια και το σφάλμα αρχίζει να μένει σταθερό, batch size = 32 και το validation split είναι 0.3. Έχοντας ορίσει τις παραπάνω παραμέτρους, παίρνουμε το παρακάτω αποτέλεσμα.

Αρχικά όσον αφορά το στάδιο της εκμάθησης έχουμε:

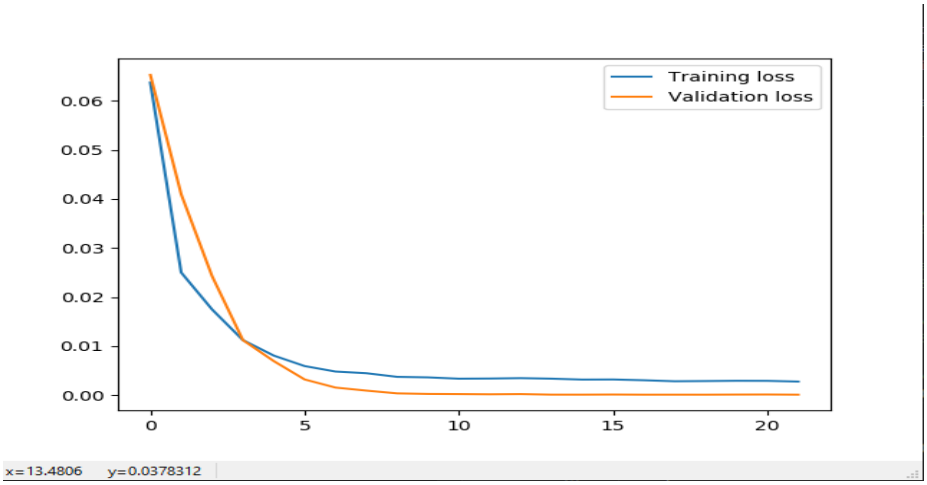
```
Epoch 1/30
64/64 [==============================] - 4s 26ms/step - loss: 0.0637 - val_loss: 0.0653
Epoch 2/30
64/64 [==============================] - 1s 17ms/step - loss: 0.0250 - val_loss: 0.0409
Epoch 3/30
64/64 [==============================] - 1s 16ms/step - loss: 0.0175 - val_loss: 0.0243
Epoch 4/30
64/64 [==============================] - 1s 16ms/step - loss: 0.0112 - val_loss: 0.0112
Epoch 5/30
64/64 [==============================] - 1s 17ms/step - loss: 0.0081 - val_loss: 0.0070
Epoch 6/30
64/64 [==============================] - 1s 17ms/step - loss: 0.0059 - val_loss: 0.0032
Epoch 7/30
64/64 [==============================] - 1s 16ms/step - loss: 0.0048 - val_loss: 0.0016
Epoch 8/30
64/64 [==============================] - 1s 17ms/step - loss: 0.0045 - val_loss: 9.3713e-04
Epoch 9/30
64/64 [==============================] - 1s 17ms/step - loss: 0.0037 - val_loss: 3.7499e-04
Epoch 10/30
64/64 [==============================] - 1s 17ms/step - loss: 0.0036 - val_loss: 2.6210e-04
Epoch 11/30
64/64 [==============================] - 1s 16ms/step - loss: 0.0034 - val_loss: 2.3054e-04
Epoch 12/30
64/64 [==============================] - 1s 16ms/step - loss: 0.0034 - val_loss: 1.8842e-04
Epoch 13/30
64/64 [==============================] - 1s 17ms/step - loss: 0.0035 - val_loss: 2.2959e-04
Epoch 14/30
64/64 [==============================] - 1s 16ms/step - loss: 0.0034 - val_loss: 1.1922e-04
Epoch 15/30
64/64 [==============================] - 1s 16ms/step - loss: 0.0032 - val_loss: 1.1644e-04
Epoch 16/30
64/64 [==============================] - 1s 17ms/step - loss: 0.0032 - val_loss: 1.4045e-04
Epoch 17/30
64/64 [==============================] - 1s 17ms/step - loss: 0.0031 - val_loss: 1.1149e-04
Epoch 18/30
64/64 [==============================] - 1s 16ms/step - loss: 0.0028 - val_loss: 1.1376e-04
Epoch 19/30
64/64 [==============================] - 1s 16ms/step - loss: 0.0029 - val_loss: 1.1175e-04
Epoch 20/30
64/64 [==============================] - 1s 17ms/step - loss: 0.0029 - val_loss: 1.3368e-04
Epoch 21/30
64/64 [==============================] - 1s 17ms/step - loss: 0.0029 - val_loss: 1.4649e-04
Epoch 22/30
64/64 [==============================] - 1s 17ms/step - loss: 0.0028 - val_loss: 1.1215e-04
```

Παρατηρούμε ότι γίνεται Early Stop μιας και το validation loss αρχίζει να μένει σταθερό και έτσι δεν χρειαζόμαστε παραπάνω Epoches. Επίσης παρατηρούμε ότι σωστά το validation loss ξεκινάει από μια μεγάλη τιμή και όσο περνάνε τα epochs τόσο καλύτερα εκπαιδεύται μειώνοντας σε μεγάλο βαθμό το λάθος. (Παρακάτω θα δούμε και γραφικά το παραπάνω λάθος.)

Σε αυτό το σημείο, θα θέλαμε να αποσαφηνίσουμε το τι σημαίνει το Overfitting αλλά και το Underfitting και να δείξουμε γιατί εμείς έχουμε αποφύγει και τις δύο περιπτώσεις. Με την έννοια Overfitting αναφερόμαστε σε ένα μοντέλο το οποίο έχει μάθει όλες τις λεπτομέρειες, μαζί και τον θόρυβο, σε βαθμό τέτοιον ώστε να επηρεάζεται αρνητικά η απόδοση του μοντέλου στα νέα δεδομένα. Αυτό σημαίνει πως οι τυχαίες διακυμάνσεις και ο θόρυβος συλλέγονται και μαθαίνονται, ενώ δεν ισχύουν και επηρεάζουν αρνητικά την ικανότητ γενίκευσης των δεδομένων. Από την άλλη το Underfitting αναφέρεται όταν ένα μοντέλο δεν μπορεί ούτε να διαμορφώσει τα training_data, αλλά και ούτε να προβλέψει σωστά τα νέα δεδομένα.

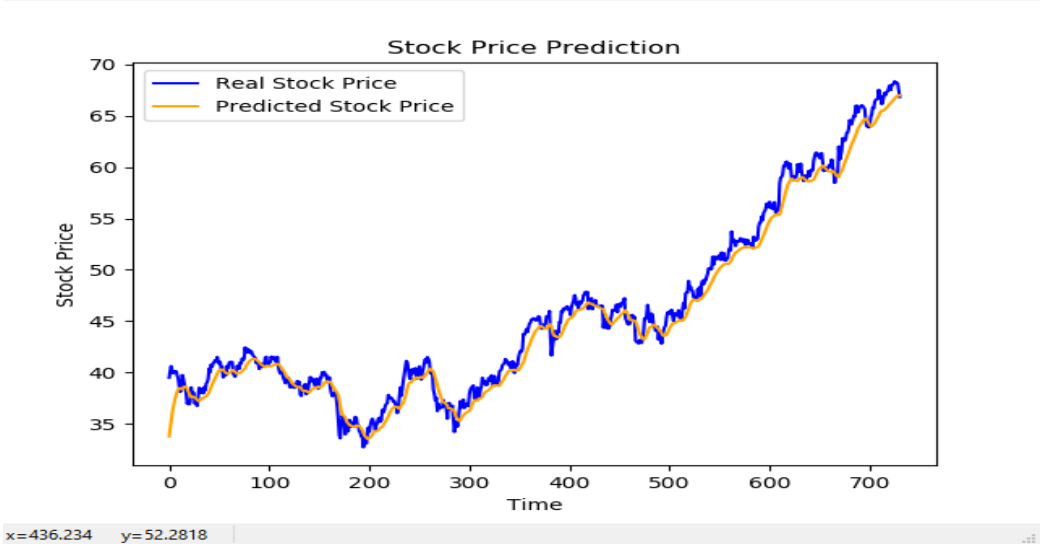
Εμείς, βάσει του Training Loss και του Validation Loss θα δούμε αν έχουμε εκπαιδεύσει σωστά το μοντέλο. Στην περίπτωση του Overfitting, τα χαρακτηριστικά που έχουν οι δύο απώλειες είναι πως, απέχουν πολύ μεταξύ τους και πολύ χαμηλή Training Loss η οποία αυξάνεται ελαφρώς με παραδείγματα εκπαίδευσης. Από την άλλη στην περίπτωση του Underfitting, έχουμε αύξηση Training Loss με προσθήκη παραδειγμάτων Training και ότι και τα δύο Loses πλησιάζουν στο τέλος.

Έτσι, βασισμένοι στο Training Loss και στο Validation Loss, αρχικά θα παρατηρήσουμε ότι δεν γίνεται ούτε Overfitting ούτε Underfitting, και δηλαδή έχουμε δημιουργήσει ένα μοντέλο το οποίο εκπαιδεύεται σωστά.



Αντίθετα, από το παραπάνω διάγραμμα παρατηρούμε ότι οι δύο απώλειες είναι αρκετά κοντά μεταξύ τους, και πως αρχικά και οι δύο απώλειες μειώνονται και στην συνέχεια αποκτούν μια επίπεδη πορεία. Τα παραπάνω δύο στοιχεία, είναι και σημάδια ότι έχει δημιουργηθεί ένα σωστό μοντέλο.

Τέλος, θα παρατηρήσουμε τα εκτιμώμενα αποτελέσματα σε σχέση με τα πραγματικά του test_set. Έτσι έχουμε:



Από την παραπάνω εικόνα παρατηρούμε πως το μοντέλο λειτουργεί αρκετά ικανοποιητικά, μιας και ακολουθεί την ροή των αποτελεσμάτων, και είμαστε σίγουροι κιόλας ότι έχουμε αποφύγει την περίπτωση του Overfitting.

Δεύτερο σκέλος

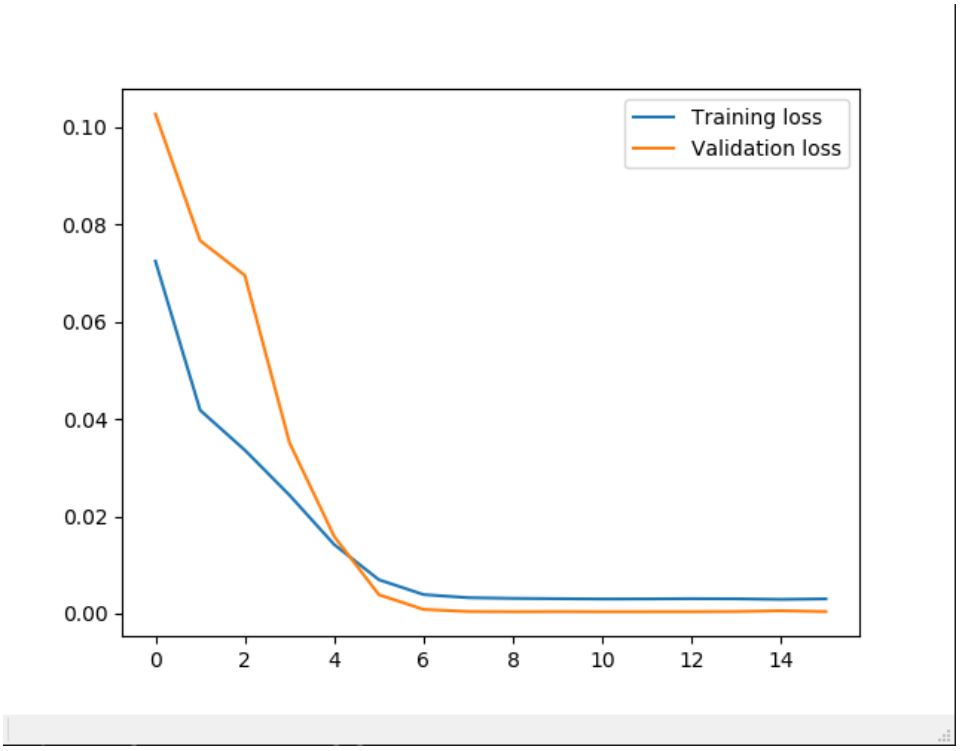
Στο σημείο αυτό, μιας και πλέον έχουμε παρουσιάσει την βέλτιστη παραμετροποίηση την οποία πειραματικά εντοπίσαμε, θα μεταβούμε στο επόμενο στάδιο της έρευνας μας που αποτελεί η σύγκριση αποτελεσμάτων με την χρήση διαφορετικών παραμέτρων. Οι μεταβλητές που θα αλλάξουμε είναι τα: Layers , dropout, epochs, batches, validation_set. Οπότε σε κάθε τρέξιμο του προγράμματος θα αναφέρουμε τις τιμές που έχουν και ποια τιμή θα αλλάζουμε κάθε φορά. Στόχος είναι να πειραματιστούμε κυρίως με οριακές τιμές για να δούμε πως αντιδράει το πρόγραμμα αλλά και η μοντελοποίηση. (Ως ιδανικό μοντέλο θα έχουμε τα αποτελέσματα τα οποία έχουμε βρει παραπάνω με την δικιά μας βέλτιστη παραμετροποίηση).

A)Layers

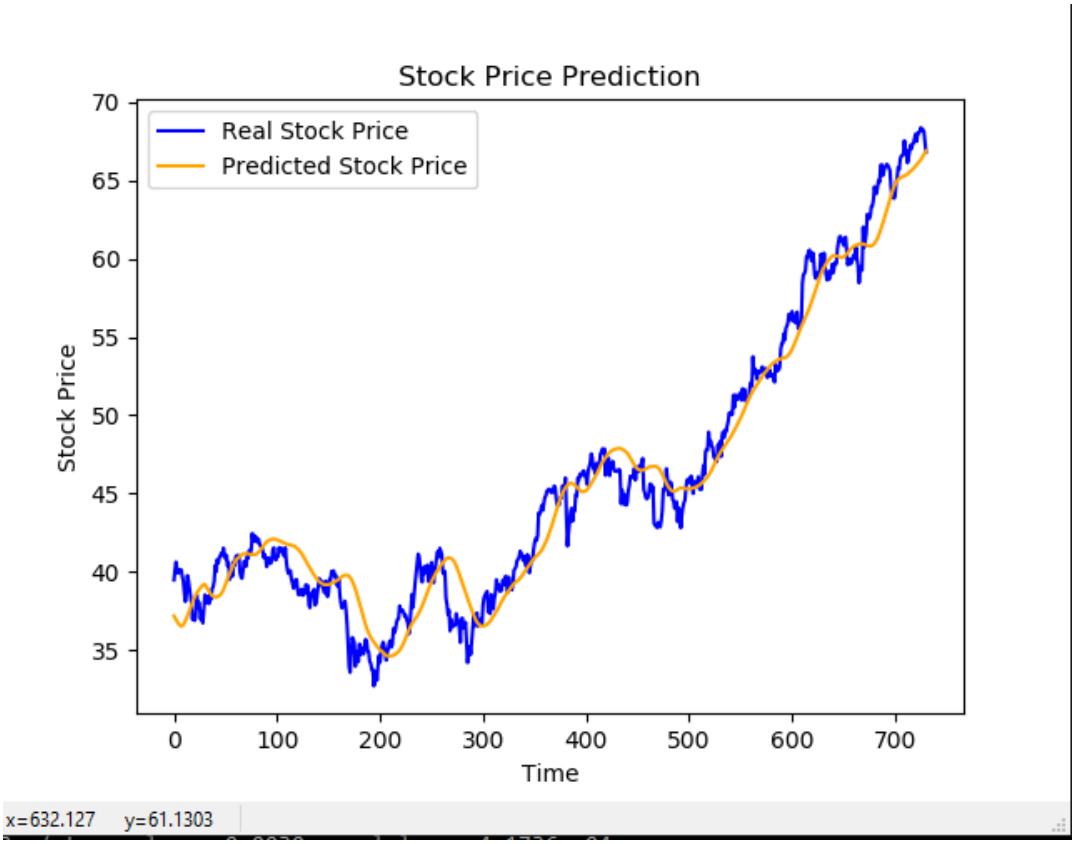
Αρχικά, θα πειραματιστούμε με την προσθήκη Layers.

Layers = 4 , epochs =30 , batch_size=32, validation_set=0.3

Με την αύξηση των Layers, από έναν σε 4 παρατηρούμε ότι σιγά σιγά αυξάνεται η διαφορά των δύο αυτών γραμμών. Υποθέτουμε, πως αυτό το οποίο ισχύει είναι ότι επειδή το dataset είναι μικρό, δημιουργείται θόρυβος στον οποίο εκπαιδεύεται το μοντέλο με αποτέλεσμα τα αποτελέσματα να γίνονται λίγο χειρότερα

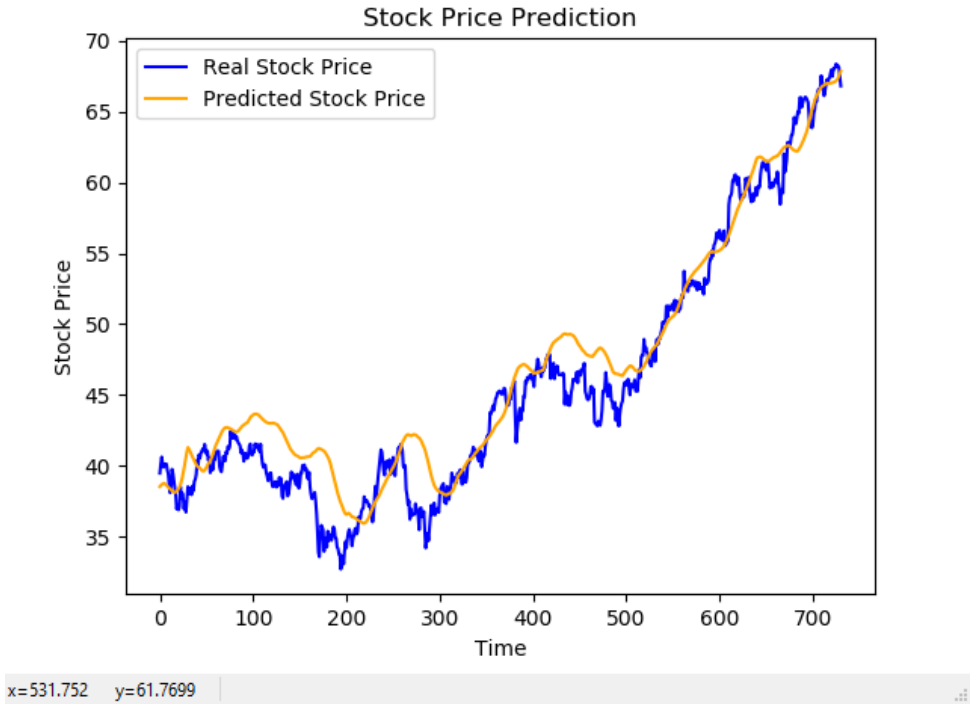


Παρακάτω μπορούμε να παρατηρήσουμε πως και τα αποτελέσματα των τιμών που μαντεύονται αποκλίνουν περισσότερο, από ότι με έναν Layer.



Θέλοντας να επιβεβαιώσουμε τον ισχυρισμό μας δοκιμάσαμε και προσθέσαμε 2 Layers ακόμα και το αποτέλεσμα ήταν το εξής

Layers = 6 , epochs =30 , batch_size =32, validation_set=0.3



Στην παρακάτω εικόνα παρατηρούμε ότι τα αποτελέσματα αποκλείνουν ακόμα περισσότερο από πριν. Άρα ο παραπάνω ισχυρισμός προκύπτει να είναι σωστός.

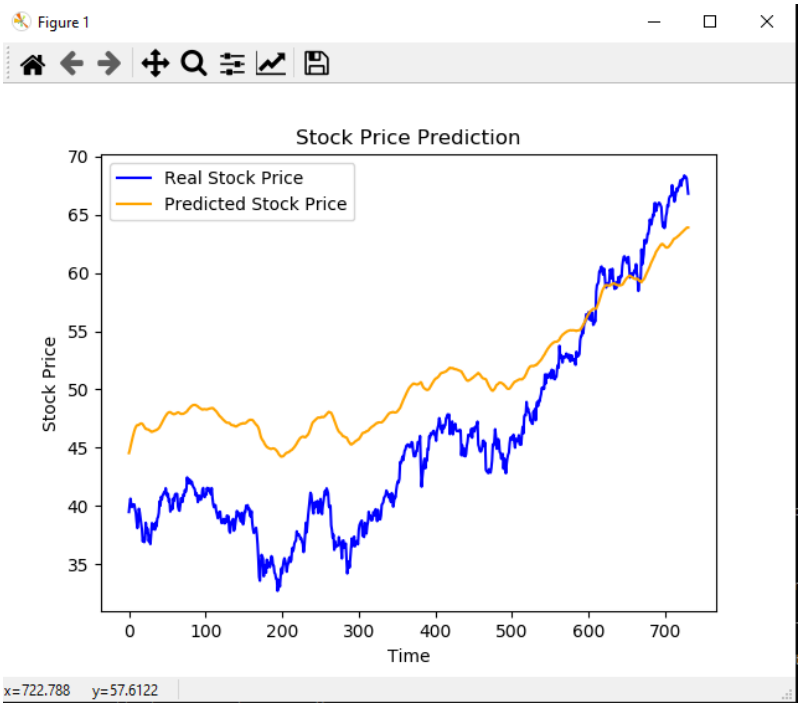
Σημαντικό επίσης είναι να συμπληρώσουμε πως η προσθήκη παραπάνω Layers δημιουργούσε επιπλέον καθυστέρηση σε κάθε Epoch, το οποίο είναι και λογικό.

B) Epochs

Αφού πλέον έχουμε δει πως συμπεριφέρεται το συγκεκριμένο πρόβλημα με τα Layers, θα ελέγξουμε τώρα πως θα το επηρεάσει η αυξομείωση των epochs.

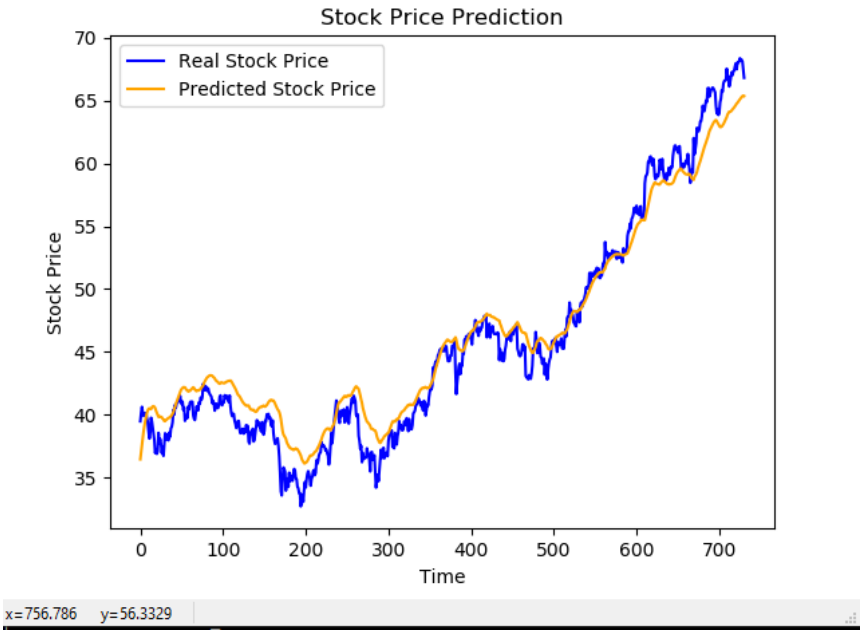
Βέβαια δεδομένου ότι ήδη το πρόγραμμα σταματάει πριν τα 30 epochs θα ελέγξουμε μόνο την ακραία τιμή του 1, μιας και να ανεβάσουμε τον αριθμό δεν θα είχε νόημα αφού το τερματίζουμε όταν το validation loss μένει σταθερό. Τα epochs γενικά καθορίζουν τον αριθμό τον οποίο ένας αλγόριθμος θα ολοκληρώσει την εκμάθηση. Οπότε από την στιγμή που θα εκτελέσουμε την εκμάθηση μόνο μια φορά, αναμένουμε ότι το αποτέλεσμα θα είναι αρκετά κακό .

Layers = 1 , **epochs =1** , batch_size =32, validation_set=0.3



Το παραπάνω αποτέλεσμα είναι λογικό, για τον λόγο που αναφέραμε παραπάνω. Για να δούμε όμως μια αύξηση σε 5 πόσο καλυτερεύει το αποτέλεσμα.

Layers = 1 , **epochs =5**, batch_size =32, validation_set=0.3

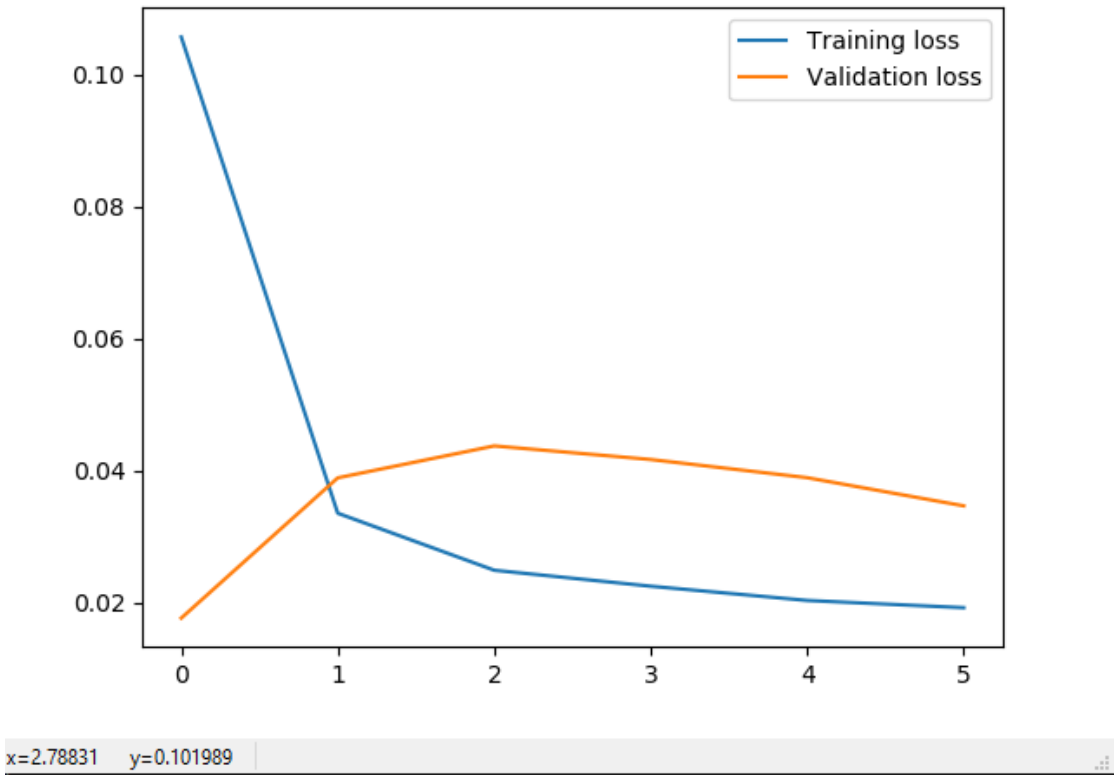


Εύκολα παρατηρούμε την μεγάλη βελτίωση που έχει η προσθήκη epochs.

Γ) Batch_size

Στην συνέχεια θα κάνουμε δοκιμές με το batch_size. Γενικά το batch_size είναι ο συνολικός αριθμός παραδειγμάτων εκπαίδευσης

Layers = 1 , epochs =30, **batch_size=128** , validation_set=0.3



Από ότι μπορούμε να καταλάβουμε από την παραπάνω εικόνα η μεγάλη αύξηση του batch_size οδηγεί σε ένα είδος overfitting, και πιθανότατα να διοχετεύει πολύ θόρυβο. Αυτό έχει και ως αναπόφευκτο αποτέλεσμα τα αποτελέσματα του μοντέλου να είναι κακά.

Δ) Ρόλος Validation_Set

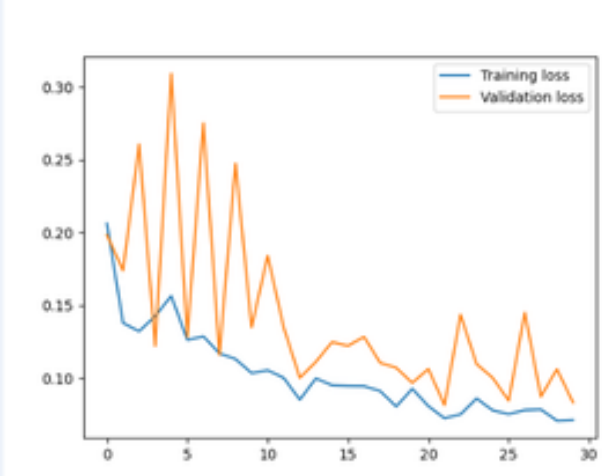
Τέλος θα θέλαμε να σχολιάσουμε τον σημαντικό ρόλο που έχει να δώσουμε κατάλληλο μέγεθος στο validation_set. Οι μεταβλητές που είδαμε παραπάνω μπορούμε να τις βρούμε μόνο με πειραματισμό για να καταλήξουμε σε αυτές που ταιριάζουν περισσότερο στο συγκεκριμένο πρόβλημα. Παρόλα αυτά παρατηρήσαμε πως αν μειώναμε το validation_set σε 0.1, σε μεγάλα προβλήματα το μοντέλο δεν θα προπονούταν σωστά αφού δεν είχε μεγάλο πλήθος στοιχείων για να προπονηθεί.

Ερώτημα Β

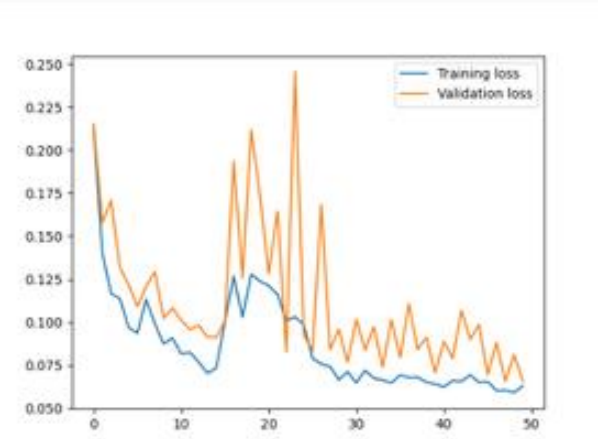
Στο ερώτημα αυτό θα παρουσιάσουμε την καλύτερη παραμετροποίηση που πετύχαμε πειραματικά

Στο στάδιο αυτό θα σας παρουσιάσουμε τις μεταβλητές με τις οποίες βρήκαμε ότι το μοντέλο αυτό εκπαιδεύεται καλύτερα χωρίς να γίνεται overfitting. Για το μοντέλο χρησιμοποιήσαμε σε κάθε στρώμα έχει 128 νευρώνες(units), και κάναμε χρήση δύο(2) Dropout. Ο optimizer είναι Adam. Τέλος, χρησιμοποιούμε batch size = 64 και το validation split είναι 0.3. Τώρα θα σας παρουσιάσουμε τα αποτελέσματα τα οποία εντοπίσαμε με διαφορετικά Epochs.

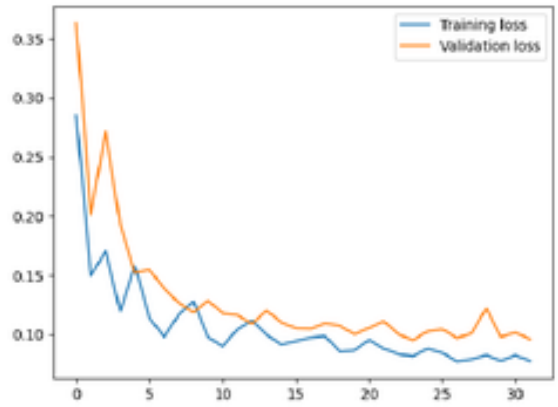
Στην αρχή τρέξαμε τον κώδικα με **30 Epochs**. Και το αποτέλεσμα που λάβαμε είναι το εξής:



Έπειτα αυξήσαμε τα Epochs σε **50.** Η γραφική συνάρτηση που προέκυψε ήταν η εξής.



Τέλος, αφού είδαμε πως δεν έκανε Early Stopping, δοκιμάσαμε και με **100.** Τα αποτελέσματα που πήραμε ήταν και αυτή την φορά πολύ καλύτερα



Αυτό το οποίο παρατηρήσαμε, είναι πως υπάρχουν πολλές αυξομειώσεις. Αυτό οφείλεται στο μικρό μέγεθος του Batch Size επειδή χρησιμοποιήσουμε τον optimizer Adan.Μερικές μικρές τέτοιες παρτίδες έχουν κακά δεδομένα τα οποία χρησιμοποιούνται στην βελτιστοποίηση. Αυτό είναι και που προκαλεί αυτές τις αυξομειώσεις.