Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα - Εργασία 3

Κωνσταντίνα Έλληνα - 1115201600046
Απόστολος Λύρας - 1115201600097

Παρατηρήσεις για Ερώτημα Α

Τα πειράματα που κάναμε για το συγκεκριμένο ερώτημα αφορούν τις υπερπαραμέτρους που ζητούνται από την εκφώνηση και παρακάτω παραθέτουμε τις παρατηρήσεις μας. Τα πειράματα έχουν γίνει για n = 1.

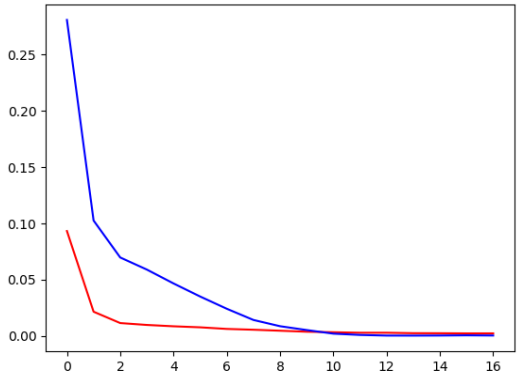
Delay

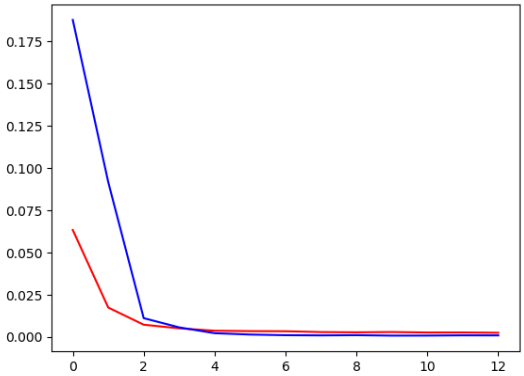
Αρχικά θέλαμε να δούμε τις διαφορές που θα έχει το εκτελέσιμο με διαφορετικό delay. Γι' αυτό το λόγο αφήσαμε σταθερές τις υπόλοιπες παραμέτρους και είδαμε τα εξής:

Με lstm_units = 8, lstm_layers = 1, dropout = 0.1, batch_size = 32, max_epochs = 20

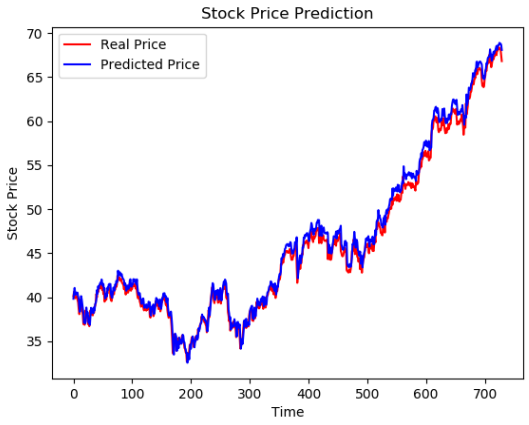
Delay	1	2	10	50	100	200
Test loss	0.00068	0.00093	0.02204	0.01161	0.01859	0.01067

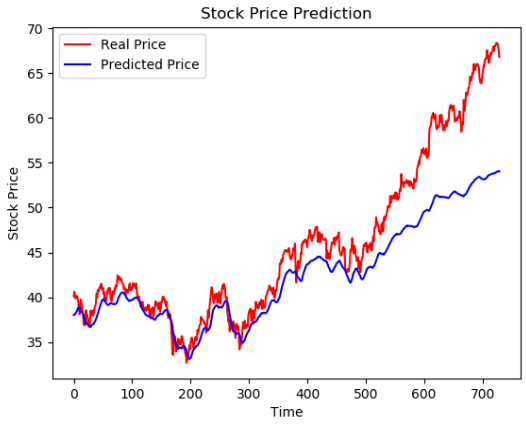
Βλέπουμε επίσης και την καλυτέρευση του **validation loss**, όσο αυξάνεται το delay, όπου Blue line = validation loss, Red line = Train loss.

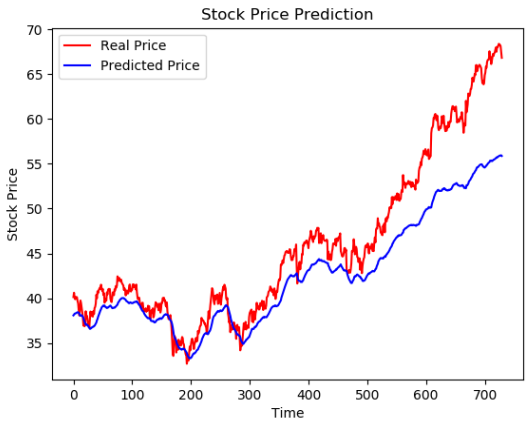


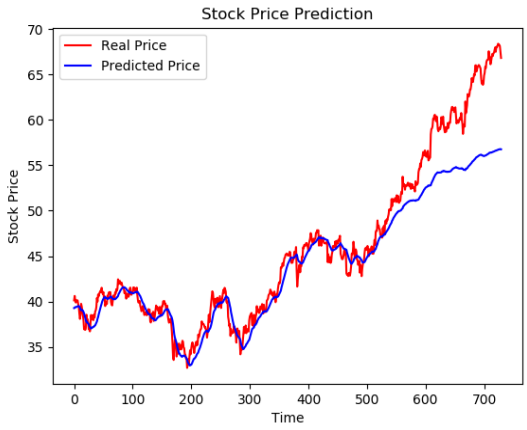


Παρακάτω φαίνεται πώς αλλάζει το **prediction** ανάλογα με την τιμή του delay. Ξεκινάει η τιμή του από 1 που βλέπουμε ότι ακολουθεί ακριβώς το μοντέλο, ύστερα με delay = 10 έως και delay = 100 η ακρίβεια δεν είναι τόσο καλή αλλά εξακολουθεί να κρατάει τη μορφή του μοντέλου και τέλος με delay = 200 ξαναβρίσκει την ακολουθία και το prediction είναι πάρα πολύ καλό.









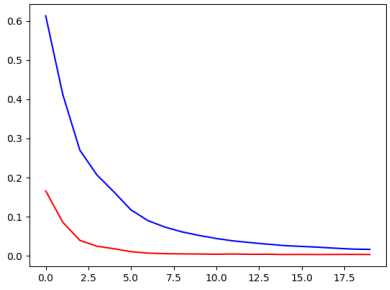
LSTM units

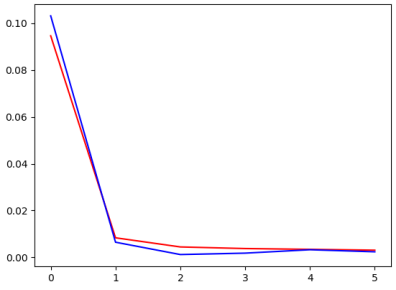
Με lstm_layers = 1, dropout = 0.1, max_epochs = 20, batch_size = 32, delay = 200

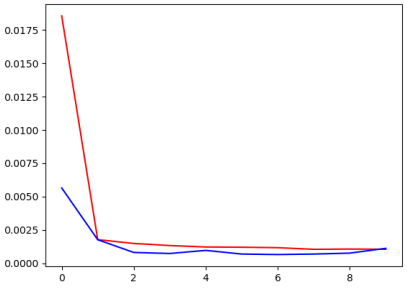
Units	2	8	16	32	64
Test loss	0.14590	0.02685	0.03303	0.00267	0.00245

Όσο αυξάνονται τα units τόσο πιο λίγες εποχές χρειάζονται για να μειωθεί το loss και με το early stopping σταματάει η εκτέλεση.

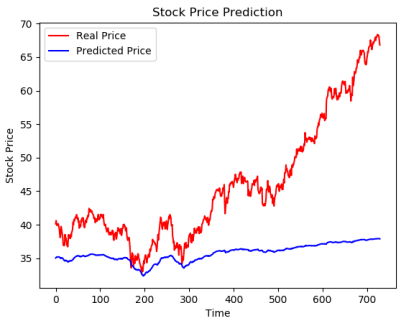
Βλέπουμε τη διαφορά που κάνουν τα units και στο validation αλλά και στο prediction.
Το **validation** αλλάζει ως εξής με units = 2, units = 16, units = 64:

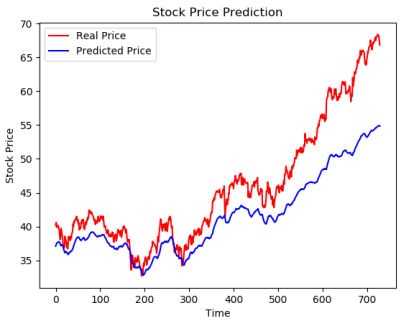


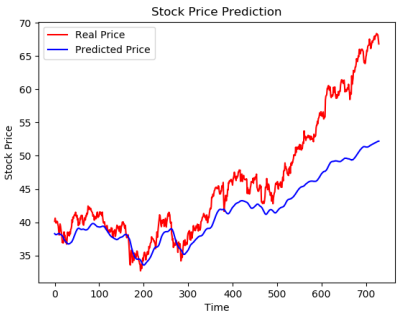


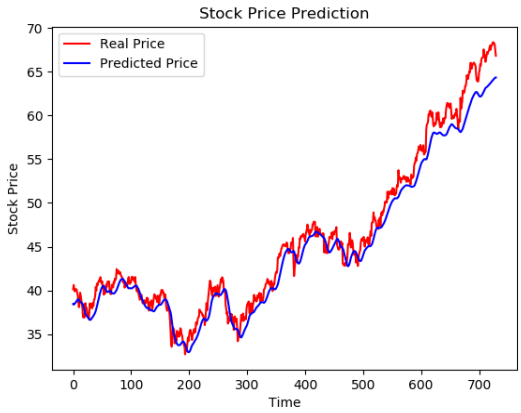


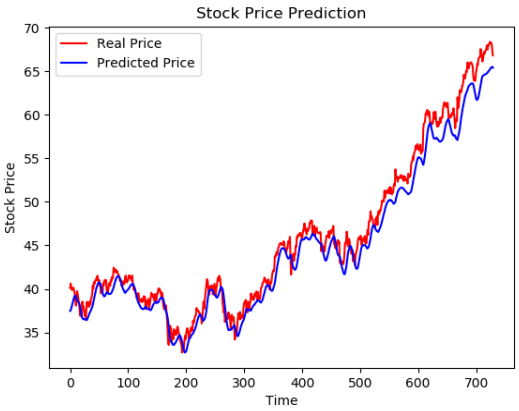
Ενώ βλέπουμε πόσο πολύ βελτιώνεται το **prediction**.
Με units = 2, units = 8, units = 16, units = 32, units = 64.









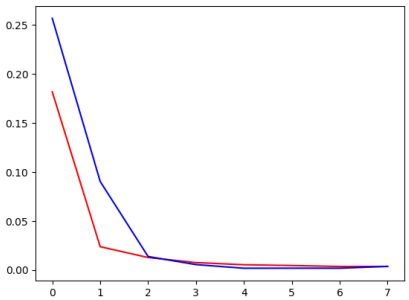


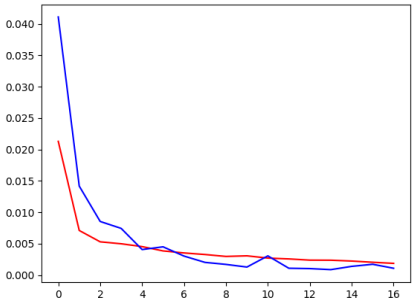
LSTM layers

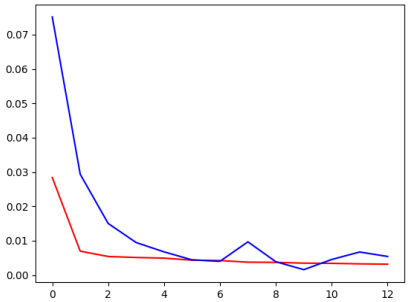
Mε lstm_units = 8, dropout = 0.1, max_epochs = 20, batch_size = 32, delay = 200

Layers	1	2	3
Test loss	0.03627	0.00861	0.04627

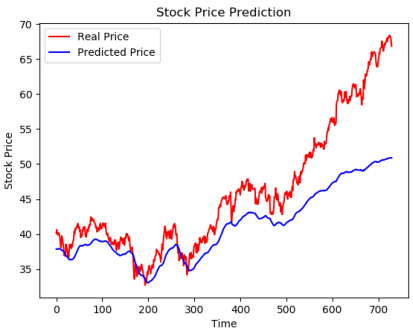
Το **validation loss** για layers = 1, layers = 2, layers = 3:

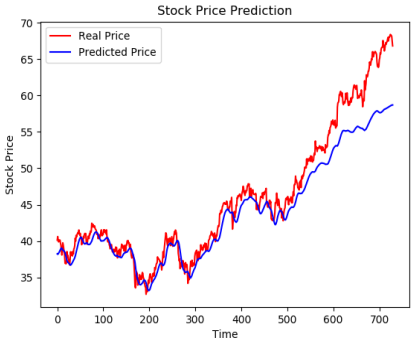


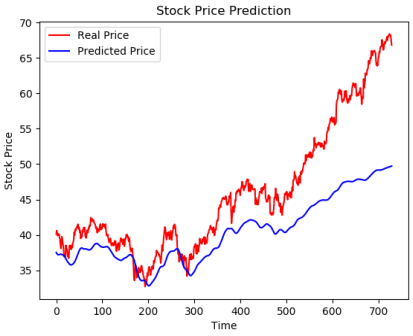




Παρατηρούμε ότι για 1 layer και για 3 layers το prediction ακολουθεί τη μορφή, αλλά δεν είναι πολύ ακριβές, ενώ για layers = 2 το prediction έχει βγει πολύ καλό.







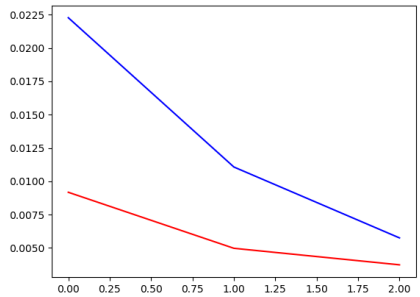
Epochs

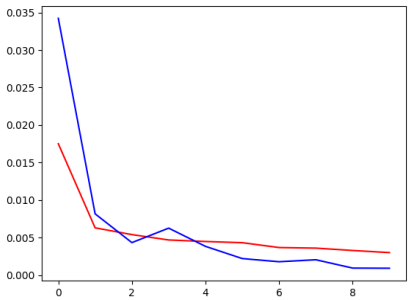
Με lstm_units = 8, lstm_layers = 1, dropout = 0.1, batch_size = 32, delay = 200

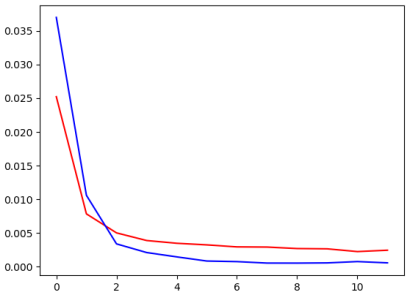
Όσο περισσότερες εποχές τόσο πιο πολύ μειώνεται το loss, καθώς κάνει περισσότερα περάσματα στο train set.

Epochs	1	3	10	40
Test loss	0.39935	0.06403	0.02355	0.00310

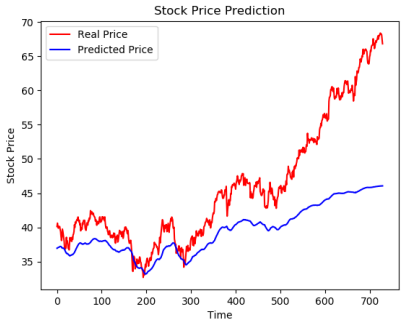
Το **validation** με epochs=3, epochs=10, epochs=40.

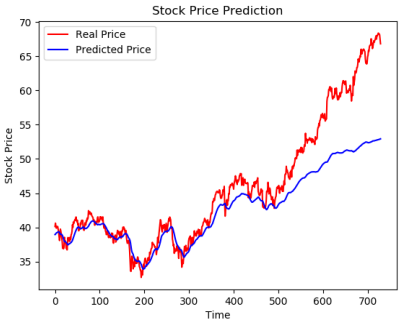


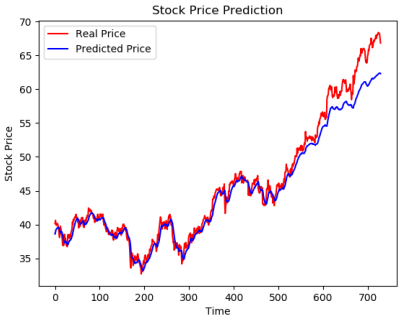




Όσο περισσότερες εποχές τόσο καλύτερο **prediction** κάνει το μοντέλο, καθώς σε κάθε πέρασμα μπορεί να βρει κι άλλες τιμές που ίσως να έχασε προηγουμένως και αυτό κάνει το αποτέλεσμα πιο ακριβές.





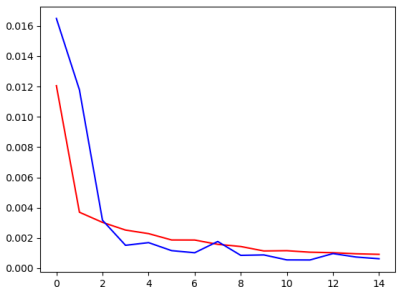


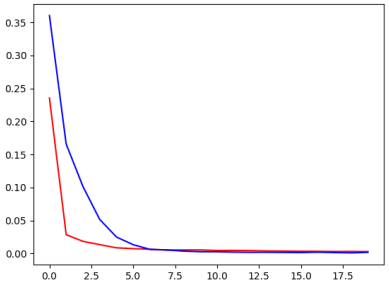
Batch Size

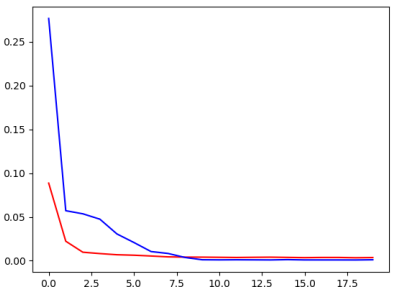
Με lstm_units = 8, lstm_layers = 1, dropout = 0.1, epochs = 20, delay = 200

Batch size	8	32	64	128
Test loss	0.00100	0.02377	0.00399	0.00766

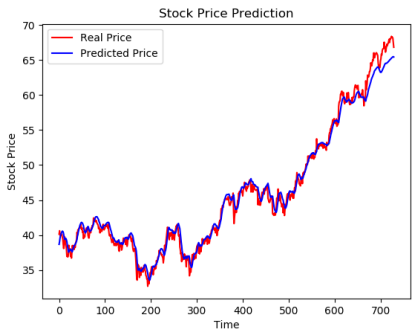
Το **validation** αλλάζει ως εξής με batch_size = 8, batch_size=32, batch_size = 128:

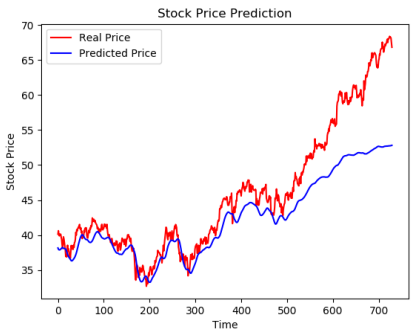


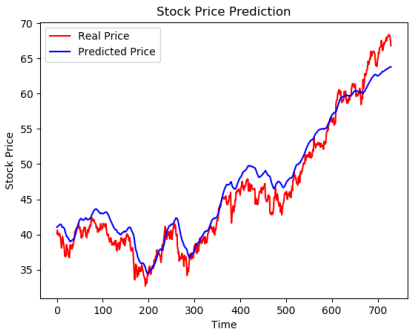




Εδώ βλέπουμε ότι το **prediction** χαλάει για μεγάλα νούμερα του batch_size. Παρατηρούμε δηλαδή ότι όταν ο αριθμός των δειγμάτων εκπαίδευσης που πρέπει να επεξεργαστούν είναι μικρός τότε το prediction είναι πιο ακριβές.







Στο φάκελο του ερωτήματος Α υπάρχουν στους αντίστοιχους φακέλους όλα τα αποτελέσματα που βγάλαμε, τα οποία δείχνουμε παραπάνω.