ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ

Αναφορά Διπλωματικής Εργασίας:

Αλγόριθμος προβλέψεων τύπου Convolutional Neural Network με εφαρμογή στην αγορά συναλλαγμάτων(FOREX)

Δημήτριος Πετρόπουλος

Α.Μ.:2015519

Επιβλέπων:

Διονύσιος Ι. Ρεΐσης

Αναπλ. Καθηγητής

Αθήνα 2019

1 Περιεχόμενα

1 Περιεχόμενα 2
2 Εισαγωγή 3
3 Μορφοποίηση δεδομένων 4
4 Tutorial 6
5 Γενική περιγραφή αλγορίθμου 7
5.1 Επεξεργασία δεδομένων από την βάση δεδομένων 7
5.2 Δόμηση νευρωνικού δικτύου 10
5.3 Λήψη αποτελεσμάτων αλγορίθμου 13
5.4 Data de-preprocessing 15
6 Επίδραση preprocessing στο αποτέλεσμα 16
7 Βελτιστοποίηση Παραμέτρων 18
7.1 Epochs 19
7.2 Batch Size 22
7.3 Training Size 24
7.4 Predict 26
8 Αποτελέσματα βελτιστοποίησης 28
9 Συμπέρασμα 30
10 Αναφορές 32

2 Εισαγωγή

Στην εργασία αυτή παρουσιάζεται ένας αλγόριθμος τύπου Συνελικτικό Νευρωνικό Δίκτυο (Convolutional Neural Network) με σκοπό την πρόβλεψη τιμών για την αγορά συναλλαγμάτων FOREX στην ισοτιμία ευρώ-δολάριο.

Αρχικά παρουσιάζεται ο τρόπος μορφοποίησης των δεδομένων και ο τρόπος εισαγωγής τους στην βάση δεδομένων . Η διαδικασία της μορφοποίησης γίνεται διότι ο όγκος των δεδομένων είναι πολύ μεγάλος μιας και μπορεί σε κάθε δευτερόλεπτο να γίνουν πολλές συναλλαγές και άρα να αλλάξει αρκετές φορές η ισοτιμία στην διάρκεια αυτή. Κατά δεύτερον, τα δεδομένα μας παρουσιάζουν μεγάλη διακύμανση και άρα πρέπει να μορφοποιηθούν ώστε να πετύχουμε καλύτερα αποτελέσματα στις προβλέψεις μας.

Στην συνέχεια παρουσιάζεται η βασική δομή του Νευρωνικού Δικτύου με έμφαση στα επιμέρους κομμάτια επεξεργασίας των δεδομένων αλλά και στο κύριο μέρος του αλγορίθμου.

Στο κεφάλαιο 5.3 παρουσιάζονται τα αποτελέσματα της βασικής έκδοσης του αλγορίθμου αλλά και η επίδραση της βασικής τεχνικής βελτιστοποίησης, δηλαδή του scaling των δεδομένων.

Στο κεφάλαιο 7 γίνεται βελτιστοποίηση των παραμέτρων του αλγορίθμου όπως οι εποχές, το batch size, το training size κλπ.

3 Μορφοποίηση δεδομένων

Κατά την διαδικασία μορφοποίησης των δεδομένων υπήρξαν διάφορα ζητήματα τα οποία έχρηζαν επίλυσης είτε λόγο περιορισμών του συστήματος είτε λόγο της ποιότητας των δεδομένων που βρέθηκαν στο διαδίκτυο.

Η πρώτη διαδικασία που έπρεπε να γίνει είναι η διαίρεση των δεδομένων σε υποσύνολα και η καταχώριση τους στις αντίστοιχες μεταβλητές πριν την εισαγωγή τους στην βάση δεδομένων.

Τα αρχεία δεδομένα τα οποία βρέθηκαν στο διαδίκτυο περιείχαν μεγάλο αριθμό δειγμάτων τιμών συναλλάγματος μιας και ήταν όλες οι συναλλαγές με ακρίβεια δεκάτου του δευτερολέπτου. Ένα παράδειγμα από τα δεδομένα παραθέτεται στην εικόνα 1.

```
1  EUR/USD,20100103 21:27:59.694,1.43067,1.43097
2  EUR/USD,20100103 21:27:59.732,1.43075,1.43095
3  EUR/USD,20100103 21:28:08.152,1.43078,1.43099
4  EUR/USD,20100103 21:28:16.897,1.43076,1.43102
5  EUR/USD,20100103 21:28:28.757,1.43071,1.43101
6  EUR/USD,20100103 21:29:07.659,1.43071,1.43106
7  EUR/USD,20100103 21:29:16.747,1.43071,1.43103
8  EUR/USD,20100103 21:29:19.952,1.43076,1.43103
9  EUR/USD,20100103 21:29:30.144,1.43076,1.43104
10 EUR/USD,20100103 21:29:35.683,1.43076,1.43106
```

Εικόνα 1.

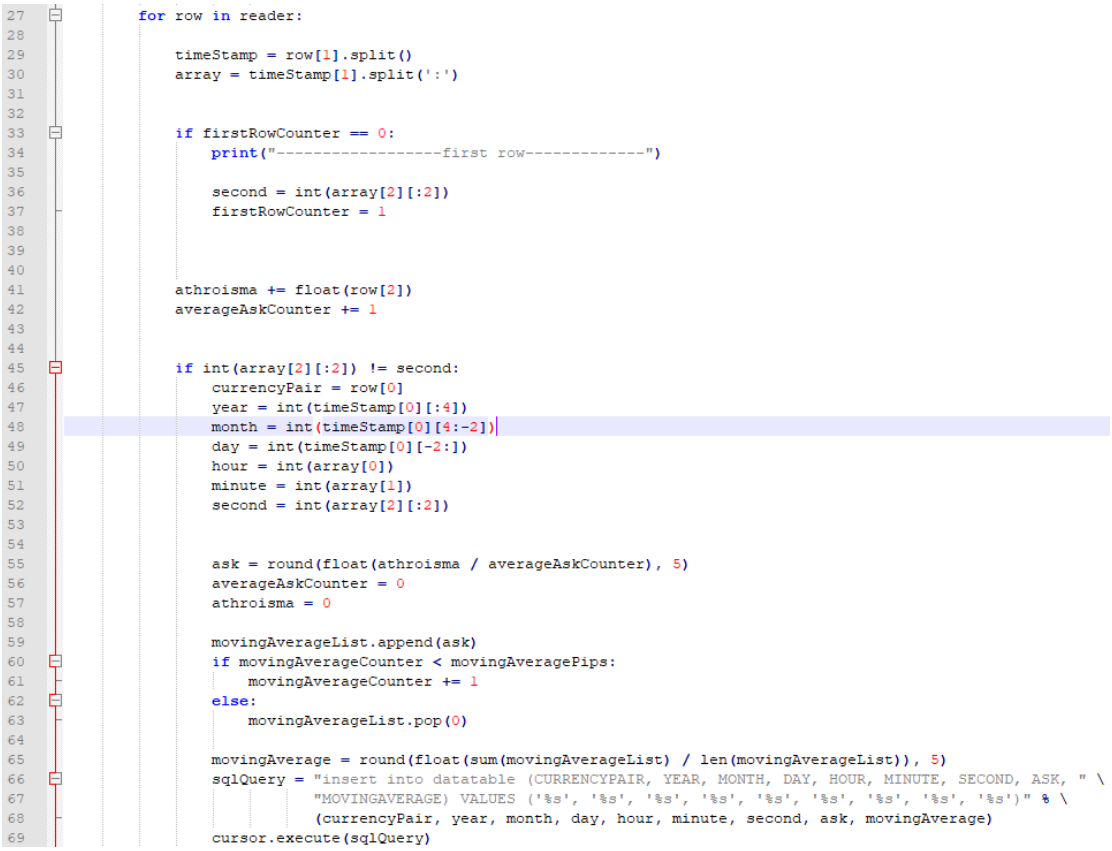
Παρατηρούμε λοιπόν ότι είναι ένα απλό αρχείο τύπου .csv αλλά τα δεδομένα δεν είναι όλα διαχωρισμένα με κόμμα. Αρχικά τα σύμβολα των ονομασιών μπορεί να είναι χωρισμένα με τα υπόλοιπα δεδομένα με κόμμα, αλλά μεταξύ του παρεμβάλλεται κάθετος «/». Η ίδια ασυμμετρία παρατηρείται και στο πεδίο της χρονοσφραγίδας όπου το έτος από τον μήνα και την ημέρα δεν χωρίζεται με κάποιο σημείο στίξης, αλλά από την ώρα χωρίζεται με ένα κενό. Στο πεδίο της χρονοσφραγίδας συναλλαγής παρατηρείται και μια ασυνέπεια στον τρόπο αποτύπωσης της ώρας, όπου η ώρα από τα λεπτά και τα δευτερόλεπτα χωρίζονται με άνω και κάτω τελεία ενώ με τις υποδιαιρέσεις του δευτερολέπτου γίνεται διαχωρισμός με τελεία.

Οι δύο τελευταίες τιμές που είναι διαχωρισμένες με κόμμα είναι οι τιμές bid price και ask price. Η τιμή bid price είναι η μέγιστη τιμή που ο αγοραστής είναι διατεθειμένος να πληρώσει ενώ το ask price (η αλλιώς offer price) είναι η ελάχιστη τιμή που ο πωλητής είναι διατεθειμένος να λάβει. Η διαφορά ανάμεσα στις δυο τιμές είναι ελάχιστη μιας και η ισοτιμία Ευρώ- Αμερικανικό Δολάριο είναι από αυτές με την μεγαλύτερη κίνηση παγκοσμίως. Γενικά μιλώντας, η διαφορά ανάμεσα στις δύο τιμές είναι ένας σημαντικός δείκτης ρευστότητας για το εκάστοτε χρηματιστηριακό προϊόν. Για λόγους απλότητας στην συγκεκριμένη εφαρμογή θα αγνοήσουμε την διάφορα αυτή και θα αποθηκευτεί στην βάση δεδομένων ο μέσος όρος των τιμών bid price και ask price.

Τα αρχικά αρχεία δεδομένων περιέχουν την ισοτιμία ευρώ-δολαρίου σχεδόν σε πραγματικό χρόνο, κατά συνέπεια τα δεδομένα που θα έπρεπε να επεξεργαστεί ο αλγόριθμος θα ήταν

πολύ περισσότερα από όσα μπορεί να επεξεργαστεί το σύστημα μας σε ένα σχετικά εύλογο χρονικό διάστημα.

Συνεπώς, προκειμένου να μειωθεί ο συνολικός όγκος των δεδομένων υπολογίσθηκε η μέση τιμή των ισοτιμιών κάθε δευτερολέπτου ως ένα πρώτο βήμα. Με τον τρόπο αυτό ο όγκος των δεδομένων που θα καταχωρηθούν στην βάση μειώθηκε σε ποσοστό μεγαλύτερο του 50%.



```
27          for row in reader:
28
29              timeStamp = row[1].split()
30              array = timeStamp[1].split(':')
31
32
33              if firstRowCounter == 0:
34                  print("------------------first row-------------")
35
36                  second = int(array[2][:2])
37                  firstRowCounter = 1
38
39
40
41              athroisma += float(row[2])
42              averageAskCounter += 1
43
44
45              if int(array[2][:2]) != second:
46                  currencyPair = row[0]
47                  year = int(timeStamp[0][:4])
48                  month = int(timeStamp[0][4:-2])
49                  day = int(timeStamp[0][-2:])
50                  hour = int(array[0])
51                  minute = int(array[1])
52                  second = int(array[2][:2])
53
54
55                  ask = round(float(athroisma / averageAskCounter), 5)
56                  averageAskCounter = 0
57                  athroisma = 0
58
59                  movingAverageList.append(ask)
60                  if movingAverageCounter < movingAveragePips:
61                      movingAverageCounter += 1
62                  else:
63                      movingAverageList.pop(0)
64
65                  movingAverage = round(float(sum(movingAverageList) / len(movingAverageList)), 5)
66                  sqlQuery = "insert into datatable (CURRENCYPAIR, YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, ASK, " \
67                             "MOVINGAVERAGE) VALUES ('%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s')" % \
68                             (currencyPair, year, month, day, hour, minute, second, ask, movingAverage)
69                  cursor.execute(sqlQuery)
```

Εικόνα 2.

Στην εικόνα 2 φαίνεται ο κώδικας στον οποίο γίνεται η ανάγνωση απ το αρχείο δεδομένων και καταχώριση στην μεταβλητή “row” .Στην συνέχεια γίνεται η διαίρεση των δεδομένων σε υποσύνολα και η ανάθεση τους στις κατάλληλες μεταβλητές πριν την εισαγωγή στην βάση δεδομένων.

Θέλοντας να μειώσουμε την μεγάλη διακύμανση που παρουσιάζουν οι ισοτιμίες χρησιμοποιήθηκε η κινούμενη μέση τιμή (moving average), όπου είναι μια συνήθης πρακτική ανάμεσα στους επενδυτές συναλλαγμάτων για την εξάλειψη του «θορύβου» που εμπεριέχεται εξαιτίας της μορφής των δεδομένων. Ο κινούμενος μέσος όρος είναι μια πρακτική στην όποια καταχωρούμε στην βάση δεδομένων ανά δευτερόλεπτο την μέση τιμή των τελευταίων δέκα δευτερολέπτων. Με τον τρόπο αυτό πέρα από την μείωση της διακύμανσης των τιμών διακρίνεται ευκολότερα η τάση (trend) που δημιουργείται και άρα ο αλγόριθμος θα είναι πιο αποδοτικός.

4 Tutorial

Προκειμένου να ξεκινήσει η διαδικασία εκπαίδευσης του αλγορίθμου και ακολούθως η διαδικασία πρόβλεψης τιμών συναλλάγματος χρειάζεται σχετική προεργασία. Η προεργασία απαιτείται έτσι ώστε να γίνει η μορφοποίηση των δεδομένων και ακολούθως η εισαγωγή του στην βάση δεδομένων. Προκειμένου να ξεκινήσει η διαδικασία μορφοποίησης και εισαγωγής των δεδομένων στην βάση δεδομένων θα πρέπει να τρέξει το ο κώδικας που περιέχεται στο αρχείο InsertToDatabase.py.

Αφού ολοκληρωθεί η διαδικασία μορφοποίησης και εισαγωγής των δεδομένων στην Βάση Δεδομένων με όνομα “testdb”, τότε μπορεί να ξεκινήσει την λειτουργία του το κύριο κομμάτι του αλγόριθμου που περιλαμβάνει την εκπαίδευση του neural network αλλά και την διαδικασία πρόβλεψης και αξιολόγησης των προβλέψεων.

5 Γενική περιγραφή αλγορίθμου

5.1 Επεξεργασία δεδομένων από την βάση δεδομένων

Ο αλγόριθμος ξεκινάει ζητώντας από την βάση δεδομένων το ID και το Moving Average για κάθε συναλλαγή. Στην συνέχεια καθώς αυτό που επιστρέφεται από την βάση είναι ένας πίνακας από tuples, τα μετατρέπουμε σε ένα απλό πίνακα χρησιμοποιώντας την συνάρτηση tuple_to_array() την οποία έχουμε γράψει στην αρχή του προγράμματος και ο κώδικας φαίνεται στην εικόνα 3.

```
49
50
51  def tuple_to_array(mytuple):
52      array = []
53      for item in mytuple:
54          array.append(float(item["MOVINGAVERAGE"]))
55      return array
56
57
```

Εικόνα 3.

Η έξοδος την άνω συνάρτησης δίνεται ως παράμετρος της συνάρτησης split_to_arrays(). Ό ρόλος της συνάρτησης αυτής είναι να διαχωρίσει τα δεδομένα σε πίνακες X και Y. Σε κάθε θέση του πίνακα X[] περιέχεται ένας πίνακας στοιχείων με χρήση των οποίου πρέπει να γίνει η πρόβλεψη από τον αλγόριθμο, η πραγματική τιμή συναλλάγματος βρίσκεται στην αντίστοιχη θέση του πίνακα Y[]. Το μέρος των πινάκων που θα χρησιμοποιηθεί για την εκπαίδευση του αλγορίθμου αποφασίζεται σε επόμενο βήμα. Ο κώδικας της συνάρτησης split_to_arrays() βρίσκεται στην εικόνα 4.

```
56
57
58  def split_to_arrays(array, training_array_size, predict, sampling_step, preprocessing_boolean):
59      X, Y = [], []
60
61      for i in range(0, len(array), sampling_step):
62          if i + training_array_size + predict < len(array):
63              x_i = array[i:i + training_array_size]
64              y_i = array[i + training_array_size + predict]
65
66              timeseries = numpy.array(array[i:i + training_array_size + predict])
67
68              if preprocessing_boolean:
69                  timeseries = preprocessing.scale(timeseries)
70              x_i = timeseries[:-predict]
71              y_i = timeseries[-1]
72
73              X.append(x_i)
74              Y.append(y_i)
75      return X, Y
76
77
```

Εικόνα 4.

Η συνάρτηση split_to_arrays() έχει ακόμη την δυνατότητα να κάνει scale τα δεδομένα αν η παράμετρος preprocessing_boolean είναι true.Η διαδικασία του scaling επεξεργάζεται τα δεδομένα και τα τοποθετεί γύρο από τον άξονα ανάλογα με την απόκλιση τους από την μέση τιμή. Η διαδικασία του scaling βοηθά σημαντικά τις επιδώσεις του αλγορίθμου, τα αποτελέσματα της σύγκρισης παραθέτονται στο κεφάλαιο 6.

Στην συνέχεια γίνεται ο διαχωρισμός των πινάκων που επέστρεψε η συνάρτηση split_to_arrays() στα στοιχεία τα οποία θα χρησιμοποιήσει το νευρωνικό δίκτυο για την εκπαίδευση του και στα στοιχεία που θα χρησιμοποιήσει για να γίνουν προβλέψεις. Τα αποτελέσματα των προβλέψεων είναι γνωστά μιας και τα δεδομένα είναι ισοτιμίες του παρελθόντος. Ο διαχωρισμός αυτός γίνεται από την συνάρτηση create_Xtimeseries_Ytimeseries() της οποίας ο κώδικας είναι στην εικόνα 5.

```
35
36
37  def create_Xtimeseries_Ytimeseries(X, y, percentage=0.8):
38      print(len(X))
39      X_train = X[0:int(len(X) * percentage)]
40      Y_train = y[0:int(len(y) * percentage)]
41
42      X_train, Y_train = shuffle_in_unison(X_train, Y_train)
43      X_test = X[int(len(X) * percentage):]
44      Y_test = y[int(len(y) * percentage):]
45
46      return X_train, X_test, Y_train, Y_test
47
48
```

Εικόνα 5.

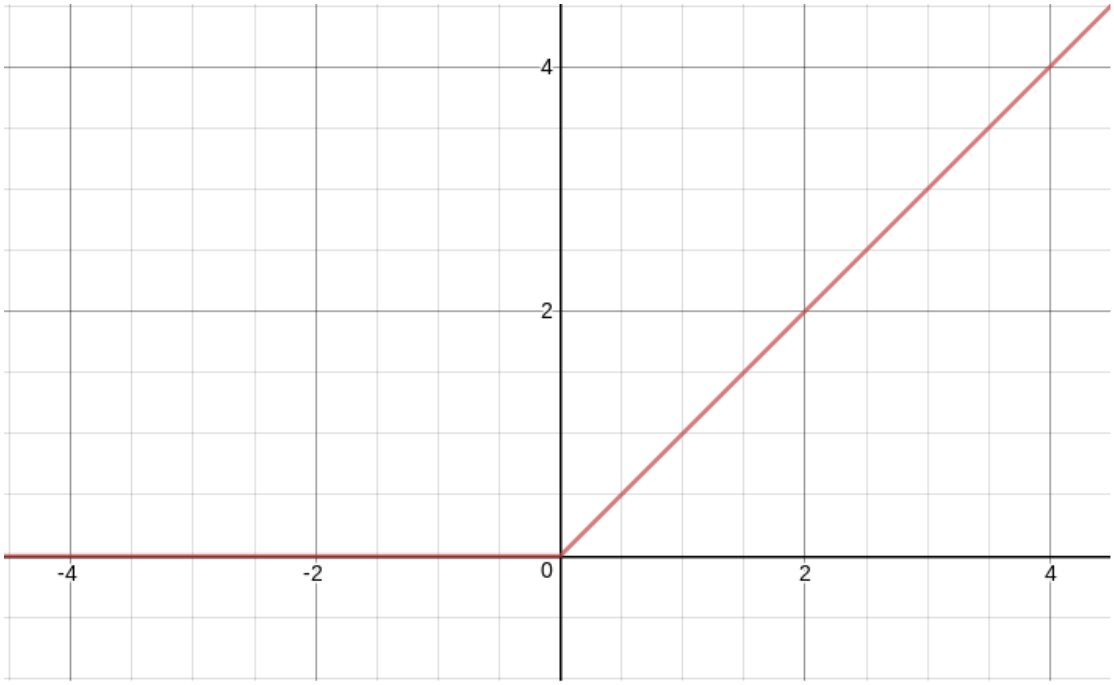
Η συνάρτηση shuffle_in_unison() που καλείται μέσα στην συνάρτηση create_Xtimeseries_Ytimeseries() είναι μια συνάρτηση η οποία προκαλεί ανάδευση των δεδομένων στους πίνακες X_train[] και Y_train[] αλλά χωρίς να χάνεται η σχέση μεταξύ του i-οστού στοιχείου που βρίσκονται στον πίνακα X_train[i] και του αποτελέσματος που βρίσκεται στον πινάκα Y_train[i]. Ο κώδικας της συνάρτησης shuffle_in_unison() παρουσιάζεται στην εικόνα 6.

```
17
18
19  def shuffle_in_unison(a, b):
20      assert len(a) == len(b)
21      shuffled_a = numpy.empty(a.shape, dtype=a.dtype)
22      shuffled_b = numpy.empty(b.shape, dtype=b.dtype)
23      permutation = numpy.random.permutation(len(a))
24      for old_index, new_index in enumerate(permutation):
25          shuffled_a[new_index] = a[old_index]
26          shuffled_b[new_index] = b[old_index]
27      return shuffled_a, shuffled_b
28
29
```

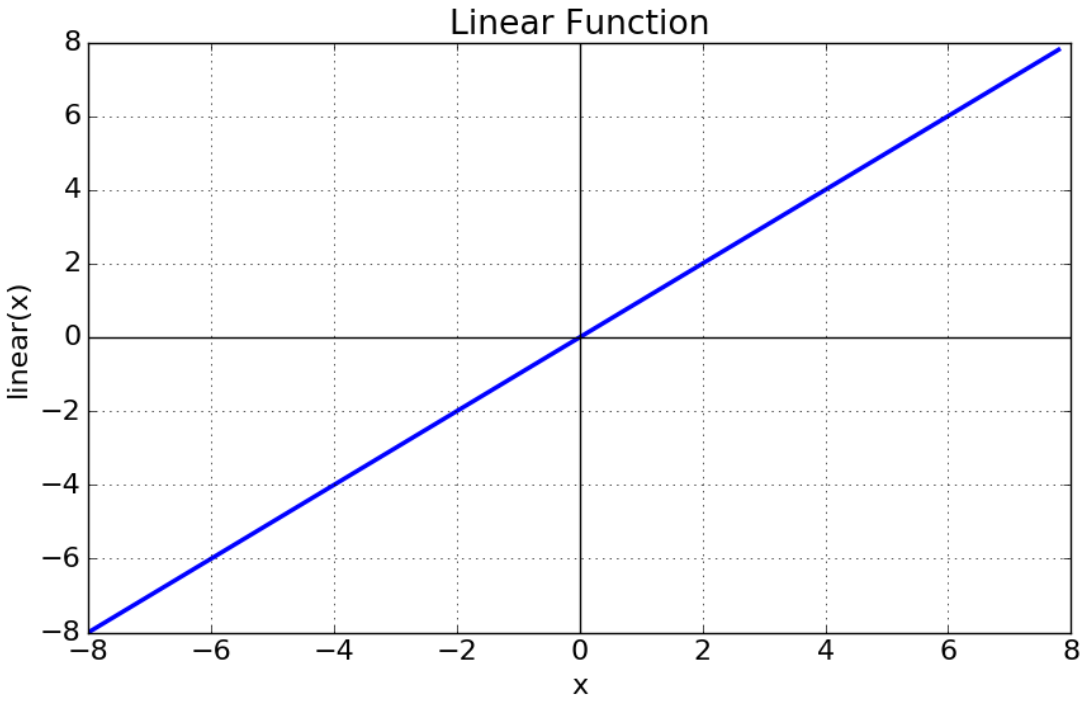
Εικόνα 6.

5.2 Δόμηση νευρωνικού δικτύου

Το νευρωνικό δίκτυο αποτελείται από δυο επίπεδα convolution layers μεγέθους 400 και 200. η συνάρτηση ενεργοποίησης στοιχείων μεταξύ των επιπέδων που χρησιμοποιούμε είναι η συνάρτηση “relu” εκτός από το fully connected layer όπου χρησιμοποιείται η συνάρτηση “linear”. Οι γραφικές παραστάσεις των συναρτήσεων “relu” και “linear” παραθέτονται στις εικόνες 7 και 8 αντίστοιχα.



Εικόνα7.



Εικόνα 8.

Μετά την επεξεργασία των δεδομένων και την οργάνωση τους σε κατάλληλους πίνακες ακολουθεί η διαδικασία της δόμησης των επιπέδων του νευρωνικού δικτύου. Όπως έχει αναφερθεί και προηγουμένως, το νευρωνικό δίκτυο αποτελείται από δυο επίπεδα convolution layers μεγέθους 400 και 200. Σε κάθε μια από τις δυο περιπτώσεις μετά από κάθε επίπεδο γίνεται η ενεργοποίηση των στοιχείων με χρήση της συνάρτησης “relu”, εκτός από το τελευταίο επίπεδο που χρησιμοποιείται η συνάρτηση “linear”. Ο κώδικας υλοποίησης των επιπέδων και της ενεργοποίησης των στοιχείων για τα δυο πρώτα επίπεδα αλλά και για το τελευταίο επίπεδο (fully connected layer) παρουσιάζεται στην εικόνα 9.

```
136
137
138    model = Sequential()
139    model.add(Dense(400, input_shape=(TRAINING_SIZE,)))
140    model.add(Activation('relu'))
141    model.add(Dropout(0.25))
142    model.add(Dense(200))
143    model.add(Activation('relu'))
144    model.add(Dense(1))
145    model.add(Activation('linear'))
146    model.compile(optimizer='adam', loss='mse')
147
```

Εικόνα 9.

Μετά την προσθήκη και του τελευταίου επιπέδου καλείται η συνάρτηση Sequential().fit() με αρχικές παραμέτρους epochs=5 και batch_size=128 προκειμένου να ξεκινήσει η διαδικασία εκπαίδευσης του αλγορίθμου. Η βελτιστοποίηση των παραμέτρων epochs και

batch_size θα γίνει στο κεφάλαιο 7 όπου εκεί γίνεται μια εκτενής μελέτη της του αποτελέσματος του αλγόριθμου βάση της μεταβολής των παραμέτρων εκπαίδευσης. Ο κώδικας της συνάρτησης Sequential().fit() παρουσιάζεται στην εικόνα 10.

```
147
148 model.fit(X_train_preprocessed,
149             Y_train_preprocessed,
150             epochs=5,
151             batch_size=128,
152             verbose=1,
153             validation_split=0.1)
154
```

Εικόνα 10.

5.3 Λήψη αποτελεσμάτων αλγορίθμου

Τα αποτελέσματα της εκπαίδευσης του αλγορίθμου πρέπει να λαμβάνονται υπόψη καθώς από την επιτυχή εκπαίδευση του αλγορίθμου τα κριθεί και η επιτυχία των προβλέψεων της τιμής συναλλάγματος αργότερα. Για τον λόγο αυτό επιθυμούμε την συνεχή βελτίωση του της παραμέτρου Loss η οποία δίνεται από τον αλγόριθμο ανά τακτά χρονικά διαστήματα κατά της διάρκεια της εκπαίδευσης του αλγορίθμου. Εκτενής ανάλυση της επίδρασης των παραμέτρων στην παράμετρο Loss γίνεται στο κεφάλαιο 7 που αφορά στην βελτιστοποίηση παραμέτρων του αλγορίθμου.

Προκειμένου να λάβουμε μια πρώτη αξιολόγηση για τα αποτελέσματα εκπαίδευσης καλούμε της συνάρτηση model.evaluate() η οποία δέχεται ως ορίσματα τους πίνακες X_test_preprocessed[] και Y_test_preprocessed[] όπου είναι οι πίνακες όπου έχουν καταχωρηθεί οι τιμές βάσει των οποίων επιθυμούμε να κάνουμε προβλέψεις. Η συνάρτηση αυτή μας δίνει και την δυνατότητα να χρησιμοποιήσουμε διαφορετικό batch size από αυτό που χρησιμοποιήσαμε κατά την διάρκεια της εκπαίδευσης του αλγορίθμου. Ο κώδικας για το κομμάτι αυτό φαίνεται στην εικόνα 11.

```
154
155  score = model.evaluate(X_test_preprocessed, Y_test_preprocessed, batch_size=128)
156  print(score)
157
```

Εικόνα 11.

Στην συνέχεια καθώς επιθυμούμε να λάβουμε ξεχωριστά τα αποτελέσματα των προβλέψεων διότι επιθυμούμε να τα παραστήσουμε και γραφικά, καλούμε την συνάρτηση model.predict() όπου ως παράμετρο της συνάρτησης δίνουμε τον πίνακα X_test_preprocessed[] που είναι ο πίνακας όπου έχουν καταχωρηθεί οι τιμές βάσει των οποίων επιθυμούμε να κάνουμε προβλέψεις. Τα αποτελέσματα που θα μας δώσει ο αλγόριθμος στην συγκεκριμένη περίπτωση θα είναι με preprocessed τιμές, άρα και τα αποτελέσματα τα οποία θα λάβουμε θα είναι preprocessed τιμές. Ο κώδικας από την κλήση της συνάρτησης model.predict() δίνεται στην εικόνα 12.

```
165
166  print("model.predict started...")
167  predicted = model.predict(X_test_preprocessed)
168
```

Εικόνα 12.

Η τιμή του mean squared error μας δείχνει κατά πόσο ο αλγόριθμος μας μπορεί να κάνει σωστές προβλέψεις και με πόση επιτυχία μπορεί βρει την μελλοντική τιμή συναλλάγματος Ευρώ-Δολαρίου και η πορεία της μεταβλητής Loss δείχνει την πρόοδο κατά την εκπαίδευση του αλγορίθμου. Και στις δύο περιπτώσεις όσο μικρότερος ο αριθμός τόσο καλύτερα τα αποτελέσματα μας. Ο κώδικας υπολογισμού της μεταβλητής MSE φαίνεται στην εικόνα 13.

```
176
177
178    mse = mean_squared_error(predicted_decoded, Y_test)
179    print("mse is :")
180    print(mse)
181
```

Εικόνα 13.

5.4 Data de-preprocessing

Στην συνέχεια προκειμένου να εξάγουμε συμπεράσματα για τις επιδόσεις των προβλέψεων του αλγορίθμου θα πρέπει να «αποκωδικοποιήσουμε» τις preprocessed τιμές που έχουμε λάβει από τις προβλέψεις που πραγματοποίησε ο αλγόριθμος. Προκειμένου να επιτευχθεί κάτι τέτοιο θα πρέπει να πάρουμε την μέση τιμή (mean) και την τυπική απόκλιση (standard deviation) από τον πίνακα X_test[] με χρήση των συναρτήσεων mean() και std() αντίστοιχα. Η διαδικασία αυτή γίνεται από τον κώδικα που παρουσιάζεται στην εικόνα 14.

```
158
159   params = []
160   for xt in X_test:
161       xt = numpy.array(xt)
162       mean_ = xt.mean()
163       scale_ = xt.std()
164       params.append([mean_, scale_])
165
```

Εικόνα 14.

Ακολούθως γίνεται η αποκωδικοποίηση των προβλέψεων του αλγορίθμου χρησιμοποιώντας τις παραμέτρους της μέσης τιμής και της τυπικής απόκλισης που λάβαμε από την διαδικασία της εικόνας 14. Ο κώδικας ώστε να επιστρέψουμε από τιμές preprocessed σε πραγματικές τιμές συναλλάγματος φαίνεται στην εικόνα 15.

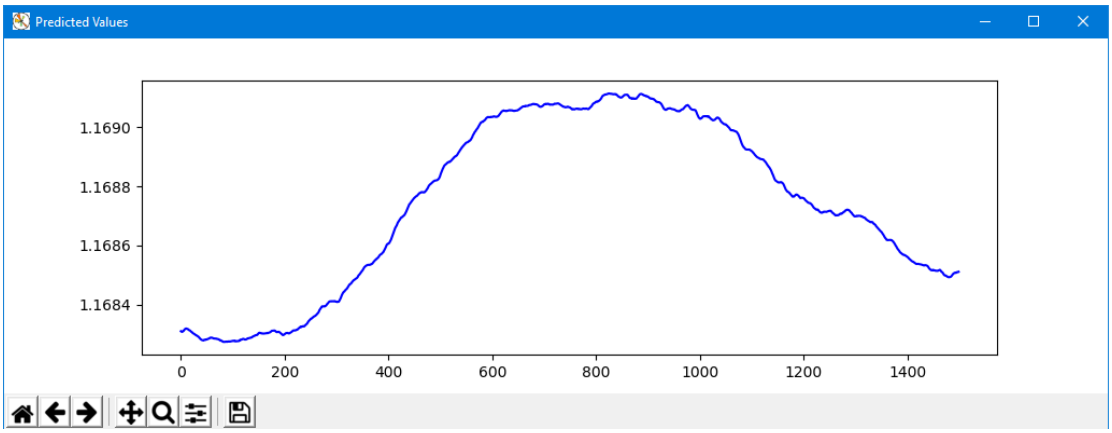
```
170
171   for pred, par in zip(predicted, params):
172       a = pred * par[1]
173       a += par[0]
174       predicted_decoded.append(a)
175
176
```

Εικόνα 15.

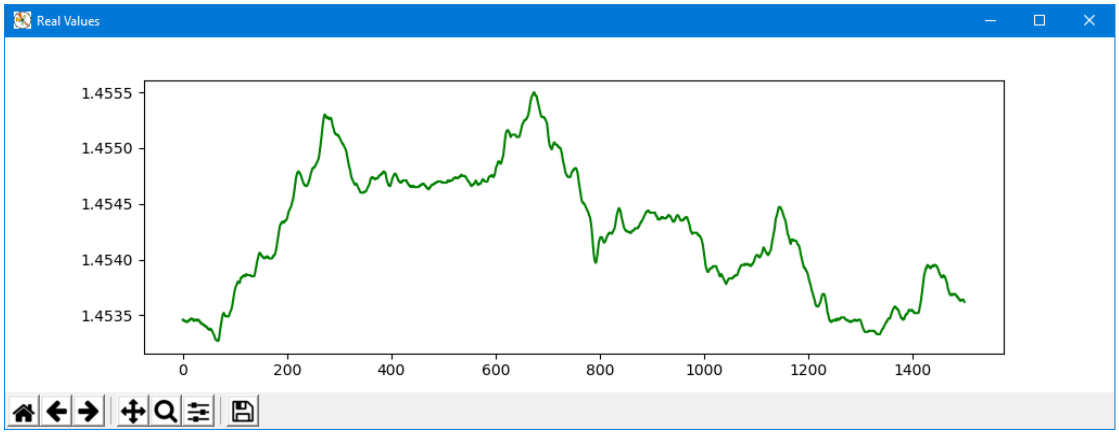
6 Επίδραση preprocessing στο αποτέλεσμα

Σε αυτό το σημείο αναλύεται η επίδραση που έχει η διαδικασία του preprocessing στο αποτέλεσμα του αλγόριθμου.

Η εκπαίδευση του αλγορίθμου έχει γίνει με τις πραγματικές τιμές συναλλάγματος και οι προβλέψεις που δίνει είναι επίσης τιμές που δεν χρειάζονται de-preprocessing .



Εικόνα 16.



Εικόνα 17.

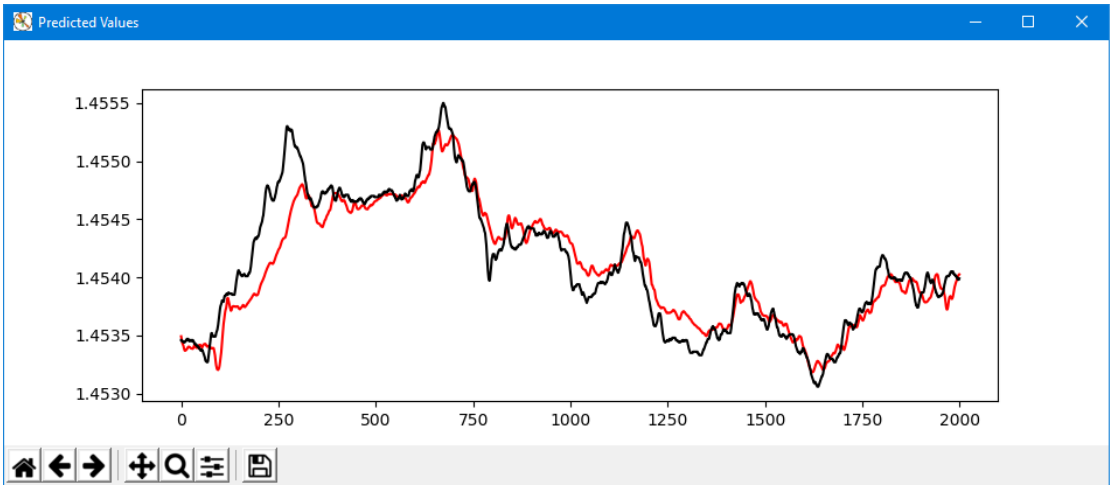
Στην εικόνα 16 είναι η τιμές συναλλάγματος που προέβλεψε ο αλγόριθμος όταν του δόθηκαν τιμές οι οποίες δεν είχαν γίνει preprocessed και στην εικόνα 17 είναι οι πραγματικές τιμές συναλλάγματος για τις χρονικές στιγμές που προβλέψαμε ώστε να μπορούμε και οπτικά να συγκρίνουμε τα αποτελέσματα του αλγορίθμου.

Στην συγκεκριμένη περίπτωση το σφάλμα (mean square error) του αλγορίθμου ήταν 0.089 το όποιο είναι ένα σημαντικό σφάλμα αν αναλογιστούμε ότι οι προβλέψεις έχουν μικρές διακυμάνσεις και γίνονται 10 δευτερόλεπτα μετά την κάθε χρονική στιγμή. Στην περίπτωσης αυτή ο αλγόριθμος απέτυχε να προβλέψει σωστά διότι οι πραγματικές συναλλάγματος κυμαίνονταν από περίπου 1,4535 έως και 1,4555 περίπου ενώ οι τιμές που προβλέφθηκαν ήταν από περίπου 1,1684 έως και 1,1690.

Παρά την αποτυχία του αλγορίθμου να προβλέψει σωστά μπορούμε να παρατηρήσουμε ότι ο αλγόριθμος έχει την δυνατότητα να ακολουθήσει έστω και στο ελάχιστο κάποιες

τάσεις που παρατηρούνται στις τιμές συναλλάγματος. Το γεγονός αυτό είναι αρκετά ελπιδοφόρο και μας δείχνει ότι αλγόριθμος μπορεί να χρήζει βελτίωσης αλλά έχει κάποιες σχετικές ικανότητες ακολουθούσης της τάσης των δεδομένων.

Στην συνέχεια προκειμένου να παρατηρηθεί η σημασία του preprocessing των δεδομένων παραθέτουμε στο ίδιο γράφημα τις πραγματικές τιμές και τις προβλεφθείσες τιμές του αλγορίθμου ώστε να μπορέσουμε να διακρίνουμε και οπτικά το κατά πόσο συγκλίνουν οι προβλεφθείσες τιμές με τις πραγματικές τιμές συναλλάγματος.



Εικόνα 18.

Στην εικόνα 18 με κόκκινο χρώμα βλέπουμε τις προβλεφθείσες τιμές του αλγορίθμου και με μαύρο τις πραγματικές τιμές συναλλάγματος.

Παρατηρούμε ότι χωρίς καμία αλλαγή στις υπόλοιπες παραμέτρους του αλγορίθμου οι προβλέψεις μας συγκλίνουν με τις πραγματικές τιμές. Το σφάλμα έχει μειωθεί σημαντικά το 5.933×10^{-8} και μπορουμε οπτικά να παρατηρήσουμε ότι αλγόριθμος μπορεί με επιτυχία να προβλέψει τις τιμές συναλλάγματος αλλά και να ακολουθήσει τις τάσεις των πραγματικών τιμών.

7 Βελτιστοποίηση Παραμέτρων

Το κεφάλαιο αυτό παρουσιάζει τη βελτιστοποίηση των παραμέτρων εκπαίδευσης του αλγορίθμου. Η βελτιστοποίηση των παραμέτρων του αλγορίθμου αποτελεί ένα από τα σημαντικότερα κομμάτια αυτής της αναφοράς καθώς η βελτιστοποίηση αυτών των παραμέτρων θα κρίνει και τα τελικά αποτελέσματα αυτού του αλγορίθμου.

Οι παράμετροι που να βελτιστοποιηθούν είναι :

Εποχές: αφορά το πόσες φορές θα γίνει η διαδικασία βελτιστοποίησης των βαρών

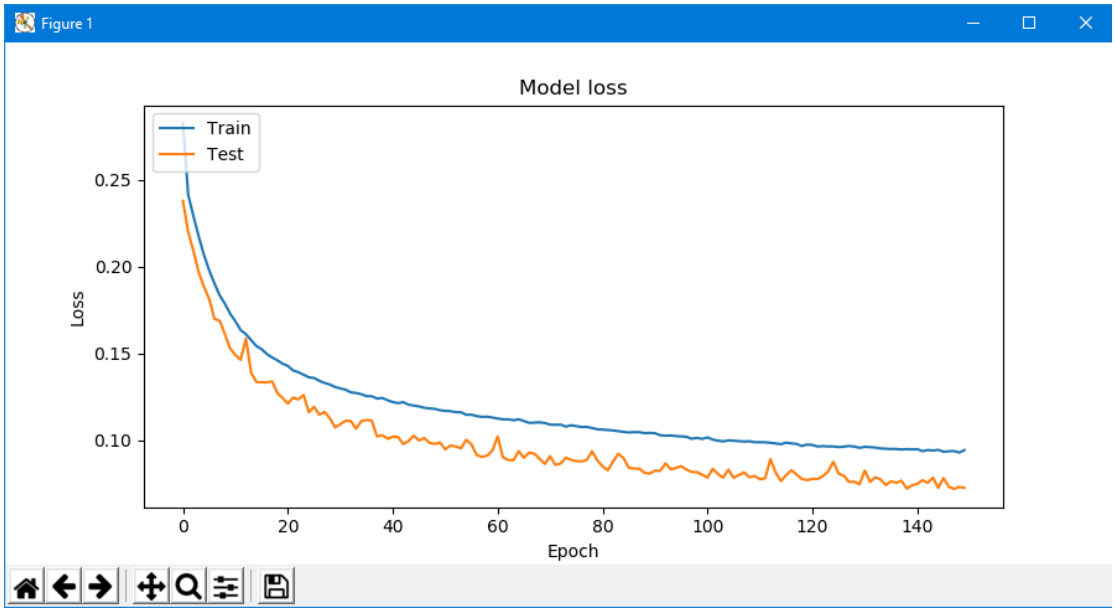
Batch Size: είναι ο αριθμός των δειγμάτων όπου μεταδίδονται μέσα στο δίκτυο και τα όποια λαμβάνονται υπόψη όταν γίνεται η βελτιστοποίηση απωλειών και η διαδικασία των προβλέψεων.

Training size: είναι το πλήθος των δειγμάτων που λαμβάνονται υπόψη κατά την διαδικασία εκπαίδευσης. Επίσης ορίζει το μέγεθος και σχήμα των δεδομένων εισόδου στο δίκτυο

Predict: εκφράζει το βάθος χρόνου πρόβλεψης σε δευτερόλεπτα.

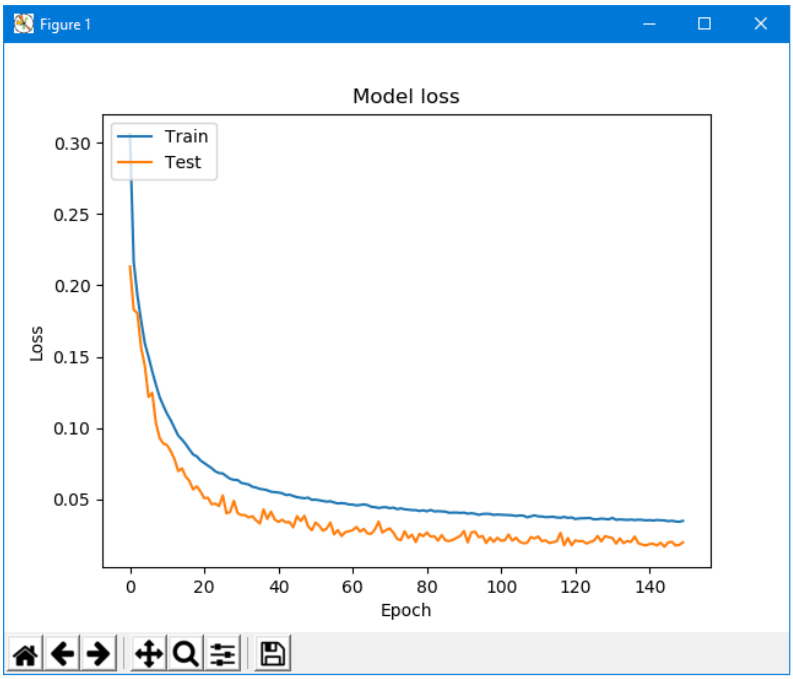
7.1 Epochs

Αρχικά αναλύεται η επίδραση της μεταβολής του αριθμού των εποχών κατά την διαδικασία εκπαίδευσης του αλγορίθμου για τα διάφορα μεγέθη δεδομένων. Στόχος είναι η ελαχιστοποίηση της μεταβλητής Loss και άρα η βελτίωση των προβλέψεων του αλγόριθμου

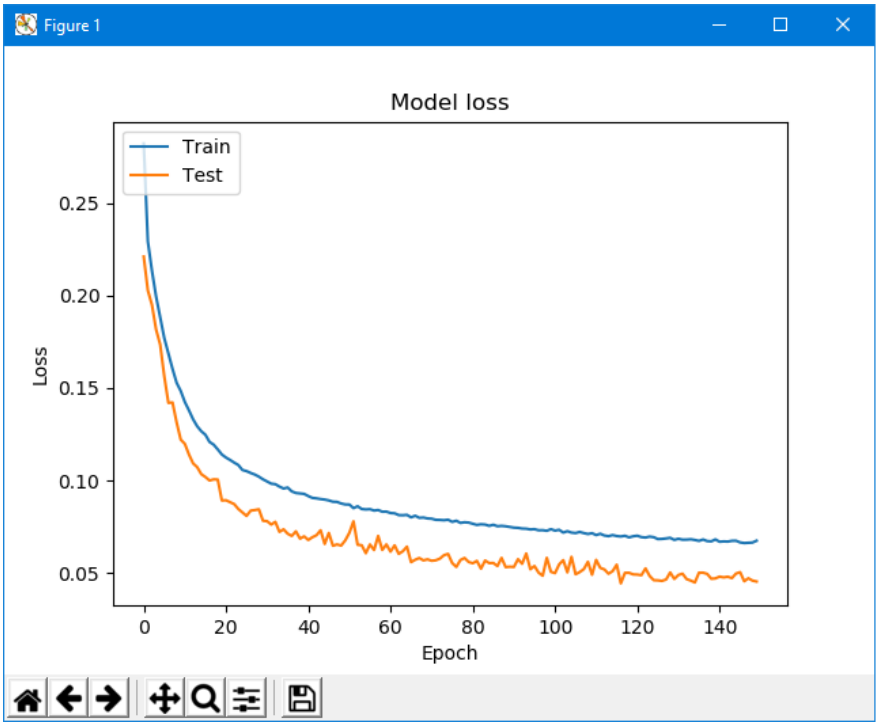


Εικόνα 19.

Παρατηρούμε στην Εικόνα 19 ότι όσο αυξάνεται ο αριθμός των εποχών με τις οποίες γίνεται η εκπαίδευση του αλγορίθμου μειώνετε αισθητά το loss έως ότου πια δεν έχει νόημα η περεταίρω αύξηση του αριθμού τους. Το μέγεθος του πίνακα δεδομένων ήταν στις 300.000 τιμές συναλλάγματος.

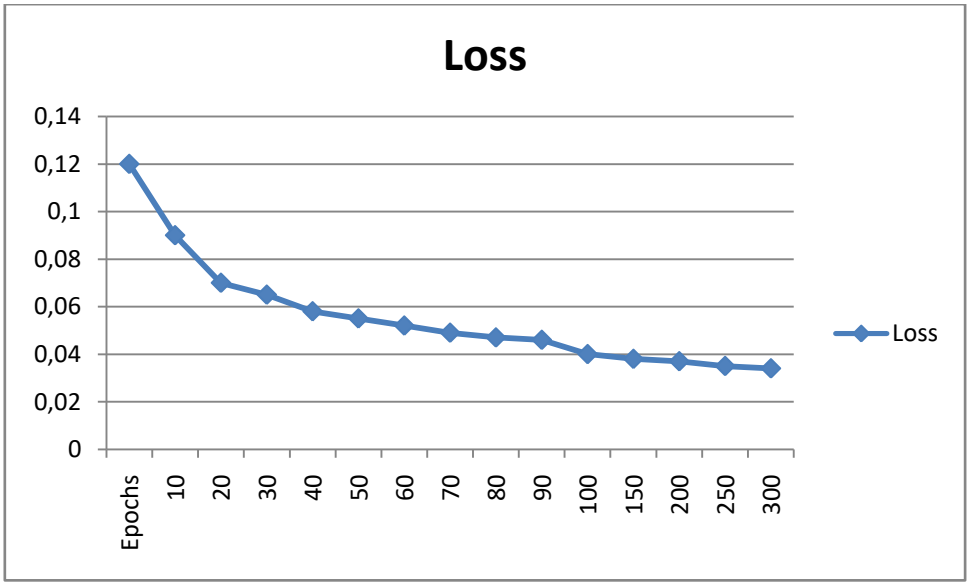


Εικόνα 20.



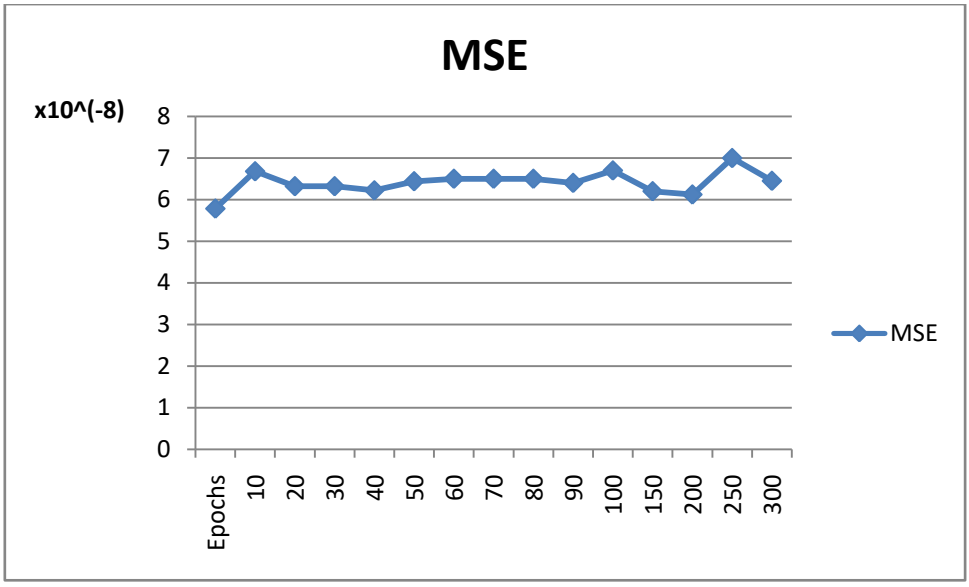
Εικόνα 21.

Στις εικόνες 20 και 21 βλέπουμε την μείωση της παραμέτρου loss διατηρώντας σταθερό τον αριθμό τον εποχών εκπαίδευσης του αλγορίθμου για 100.000 τιμές συναλλάγματος και 200.000 τιμές συναλλάγματος αντίστοιχα.



Εικόνα 22.

Στην εικόνα 22 βλέπουμε πως μεταβάλλεται η παράμετρος Loss για 300.000 τιμές συναλλάγματος για έως και 300 εποχές εκπαίδευσης.



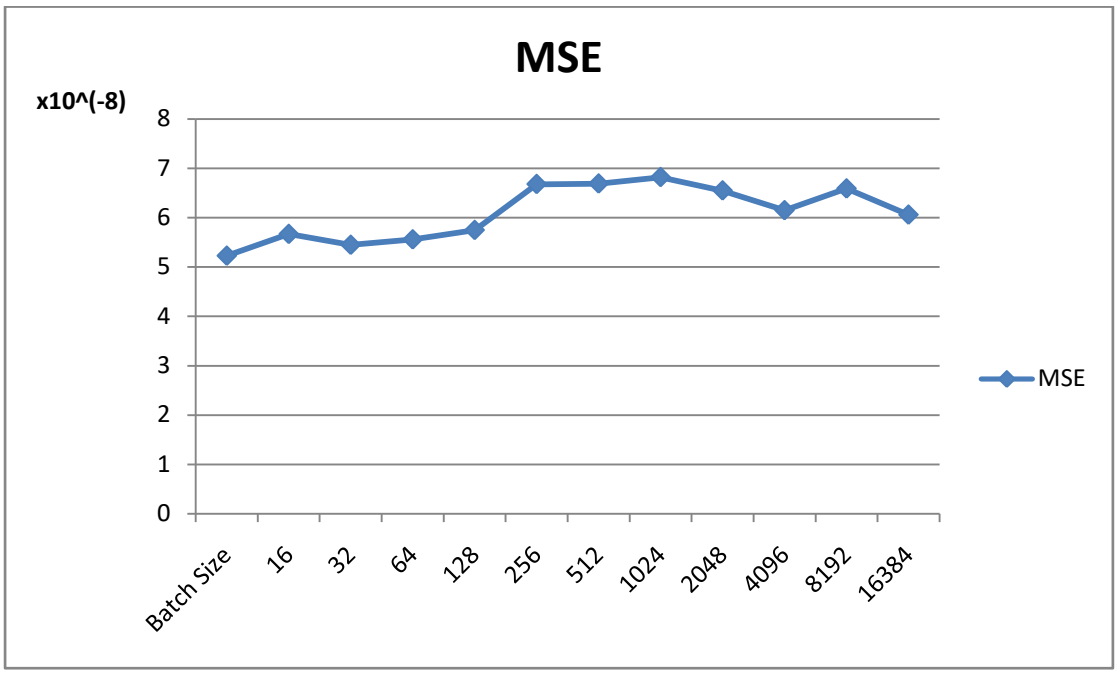
Εικόνα 23.

Στην εικόνα 23 βλέπουμε και την πορεία της τιμής MSE. Ίσως να αναμενόταν μια πτώση της τιμής MSE καθώς αυξάνονται οι εποχές εκπαίδευσης αλλά το διάγραμμα δείχνει μια σταθερή πορεία, ειδικά για τις μικρές τιμές της μεταβλητής epoch παρατηρούνται και μεγάλες διακυμάνσεις στην μεταβλητή MSE.

7.2 Batch Size

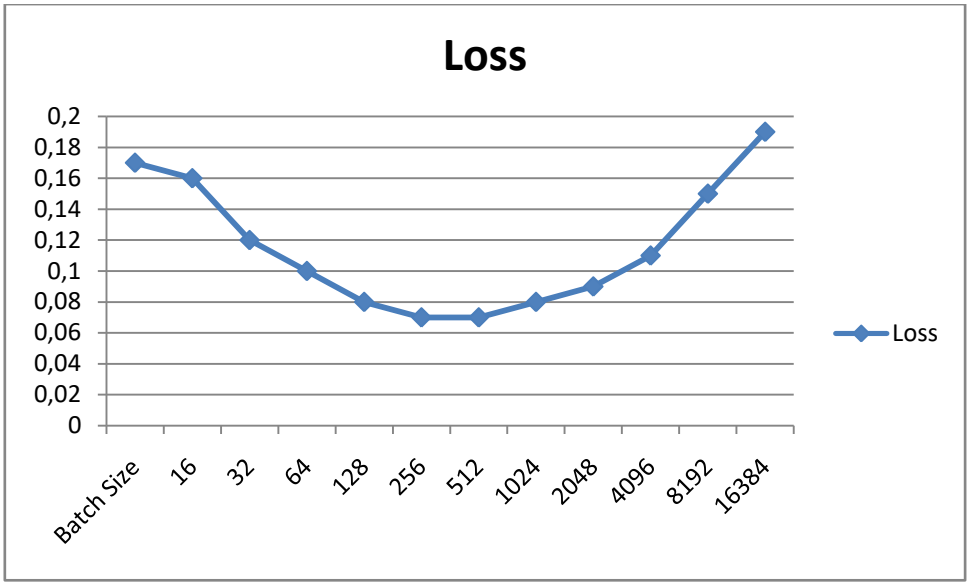
Σε αυτή την υποενότητα θα αναλυθεί η επίδραση της μεταβλητής batch size κατά την διαδικασία εκπαίδευσης του αλγορίθμου. Στόχος είναι η ελαχιστοποίηση της μεταβλητής MSE (mean square error) και άρα η βελτίωση των προβλέψεων του αλγόριθμου

Προκειμένου να εξαχθούν ξεκάθαρα συμπεράσματα θα διατηρηθούν αμετάβλητες οι παράμετροι epochs=100, το πλήθος των δεδομένων συναλλάγματος με χρήση των οποίων γίνεται η εκπαίδευση (300,000 στιγμές), το βάθος χρόνου όπου γίνεται η πρόβλεψη (50 δευτερόλεπτα) αλλά και το Training Size.



Εικόνα 24.

Στο παραπάνω διάγραμμα παρατηρούμε ότι καθώς αυξάνεται η παράμετρος Batch Size υπάρχει και μια τάση αύξησης του σφάλματος MSE. Το φαινόμενο αυτό είναι απολύτως λογικό καθώς γίνεται καλύτερη εκπαίδευση του αλγορίθμου με μικρότερες τιμές της παραμέτρου αυτής.



Εικόνα 25.

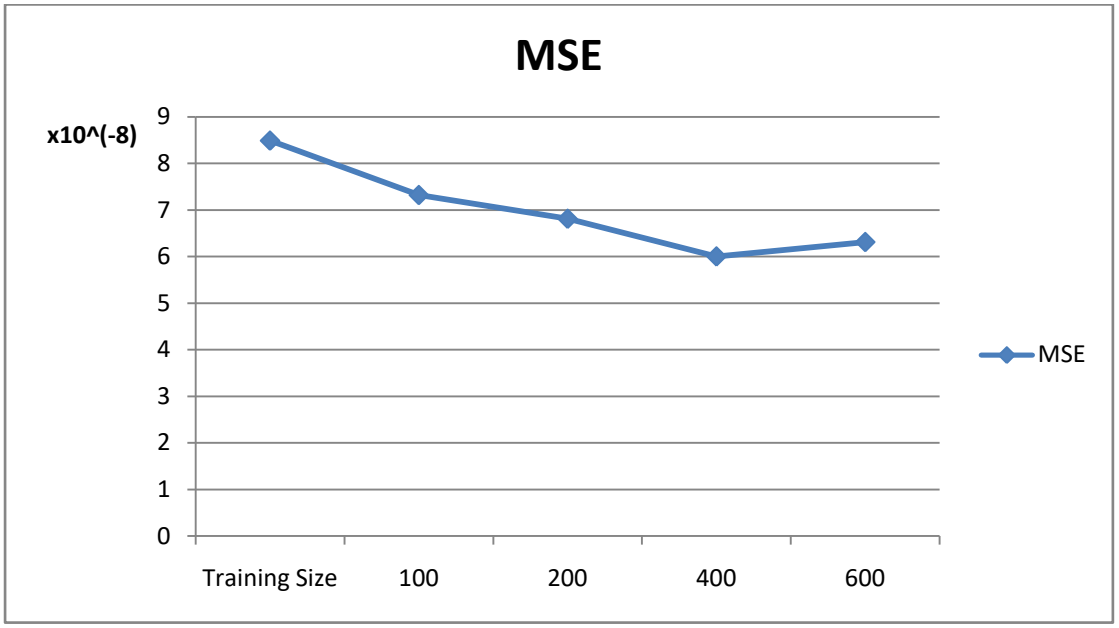
Στην εικόνα 25 φαίνονται τα αποτελέσματα της μεταβλητής loss που δείχνει τις απώλειες κατά την διαδικασία την εκπαίδευσης του αλγορίθμου

Ο λόγος όμως που επιλέγεται ο καθορισμός της μεταβλητής Batch Size στο 1024 στα περισσότερα κομμάτια τις βελτιστοποίησης είναι διότι υπάρχουν περιορισμοί του συστήματος και άρα ο χρόνος εκτέλεσης του αλγορίθμου ανεβαίνει σημαντικά για τις πολύ μικρές τιμές της παραμέτρου αυτής.

7.3 Training Size

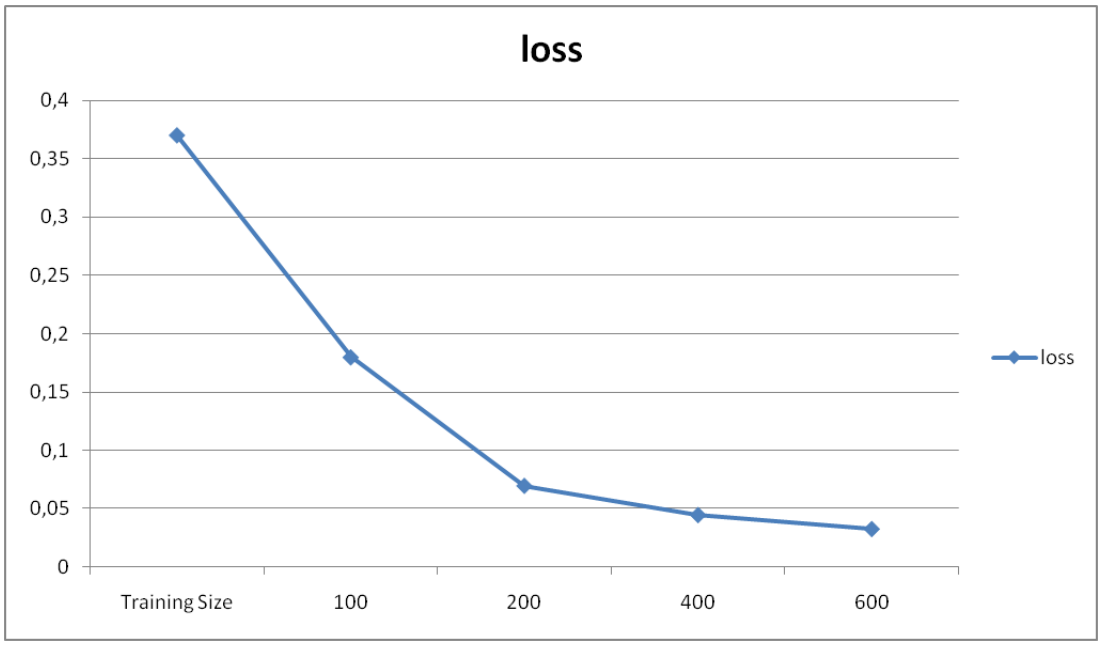
Σε αυτή την υποενότητα αναλύουμε την επίδραση που έχει η μεταβλητή training size στην αποτελεσματικότητα του αλγορίθμου. Η μεταβλητή TRAINING_SIZE εκφράζει το πλήθος των δεδομένων που λαμβάνει υπόψη ο αλγόριθμος ώστε να κάνει μια πρόβλεψη.

Προκειμένου να παραχθούν ξεκάθαρα συμπεράσματα θα διατηρηθούν αμετάβλητες οι παράμετροι epochs=100, το πλήθος των δεδομένων συναλλάγματος με χρήση των οποίων γίνεται η εκπαίδευση σε 300.000 δεδομένα αλλά και το βάθος χρόνου όπου γίνεται η πρόβλεψη (50 δευτερόλεπτα).



Εικόνα 26.

Στην εικόνα 26 παρατηρούμε την συνεχή μείωση του σφάλματος πρόβλεψης καθώς αυξάνεται η παράμετρος training size, κάτι το όποιο είναι απολύτως φυσιολογικό καθώς η παράμετρος αυτή του αλγορίθμου ορίζει πόσες τιμές θα λάβει υπόψη του ο αλγόριθμος προκειμένου να κάνει μια πρόβλεψη κατά την διάρκεια της εκπαίδευσης.



Εικόνα 27.

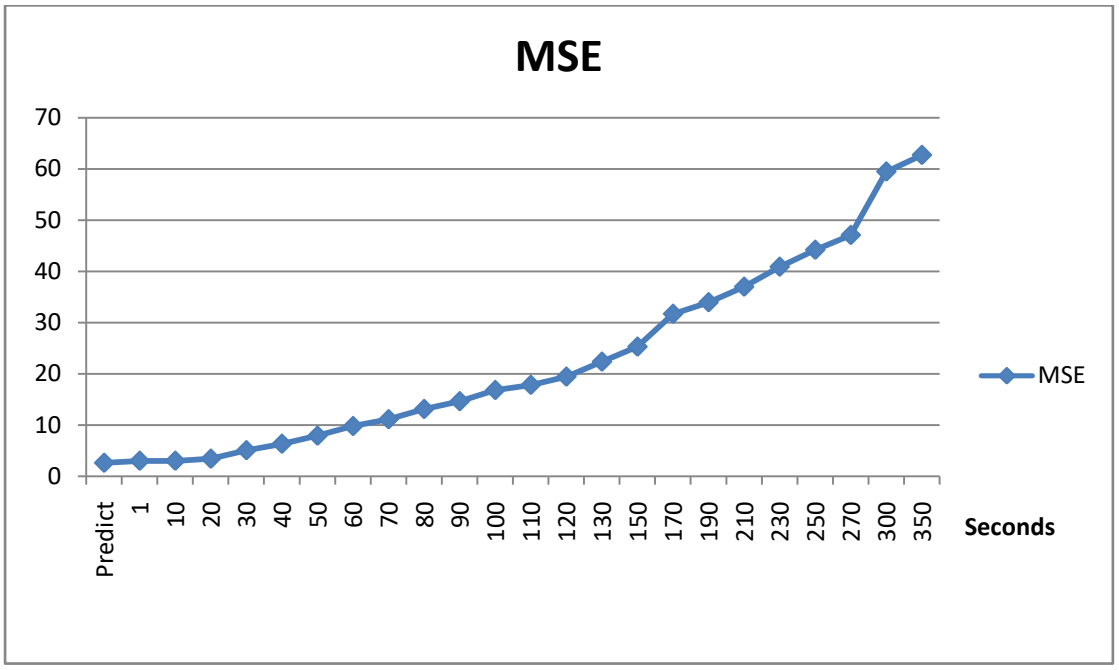
Και όπως είναι αναμενόμενο ανάλογη καθοδική τάση παρατηρούμε και στις απώλειες κατά τη εκπαίδευση του αλγορίθμου.

Σύμφωνα με την μέχρι τώρα μελέτη καθορίζουμε το training size στα περισσότερα σημεία της βελτιστοποίησης σε 600 τιμές. Θα είχε πολύ ενδιαφέρον να βλέπαμε τα αποτελέσματα του αλγορίθμου για τιμές μεγαλύτερες του 600 αλλά οι περιορισμοί του συστήματος δεν μας επιτρέπουν κάτι τέτοιο διότι ανάλογα με την μεταβλητή Training Size αυξάνονται ανάλογα και οι απαιτήσεις σε μνήμη RAM.

7.4 Predict

Στο σημείο αυτό αναλύεται η επίδραση του χρόνου που παρεμβάλετε από την χρονική στιγμή της τελευταίας τιμής συναλλάγματος μέχρι την χρονική στιγμή για την οποία κάνουμε μια πρόβλεψη.

Προκειμένου να εξαχθούν ξεκάθαρα συμπεράσματα θα διατηρηθούν αμετάβλητες οι παράμετροι epochs=100, το πλήθος των δεδομένων συναλλάγματος με χρήση των οποίων γίνεται η εκπαίδευση σε 300.000 δεδομένα αλλά και η παράμετρος Batch_Size=1024.



Εικόνα 28.

Στο παραπάνω διάγραμμα φαίνεται η μεταβολή του σφάλματος πρόβλεψης καθώς μεταβάλλεται το βάθος χρόνου πρόβλεψης (μεταβλητή predict). Το σφάλμα πρόβλεψης είναι ελάχιστο για μικρές τιμές

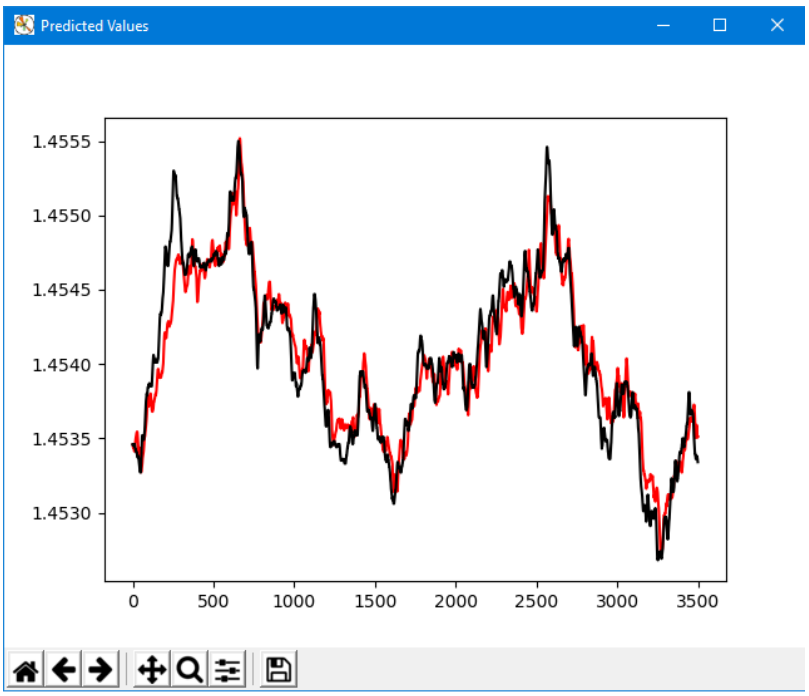
Εικόνα 29.

Μια αντίστοιχη τάση παρατηρείται και στο διάγραμμα του Loss κατά την εκπαίδευση του αλγορίθμου.

Η ανάλογη αύξηση του MSE και του Loss καθώς αυξάνεται η μεταβλητή Predict (δηλαδή το βάθος χρόνου πρόβλεψης) είναι κάτι απολύτως φυσιολογικό μιας και το εύρος των πιθανών τιμών αυξάνεται σημαντικά.

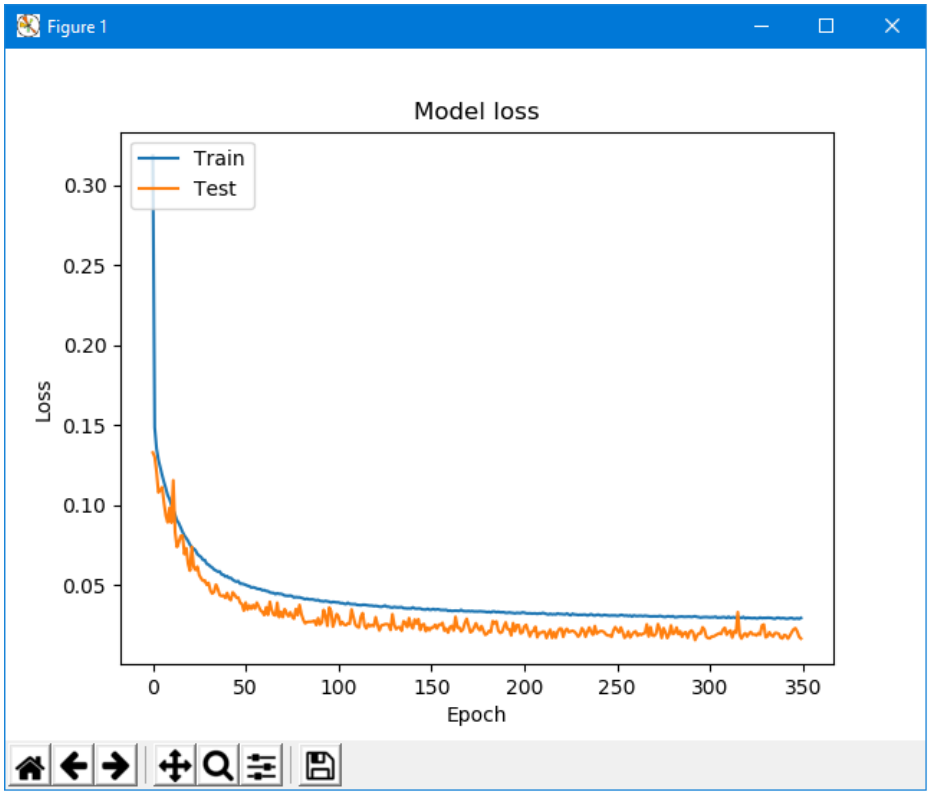
8 Αποτελέσματα βελτιστοποίησης

Μετά την συλλογή πληροφοριών από την βελτιστοποίησης παραμέτρων του αλγορίθμου τα αποτελέσματα του αλγόριθμο φαίνονται στα επόμενα διαγράμματα.



Εικόνα 30.

Στην εικόνα 30 βλέπουμε και γραφικά τα αποτελέσματα του αλγορίθμου συγκριτικά με τις πραγματικές τιμές συναλλάγματος για τις προβλεφθείσες χρονικές στιγμές. Το αποτέλεσμα αυτό οφείλεται στην προσθήκη των βέλτιστων τιμών παραμέτρων του αλγορίθμου.



Εικόνα 31.

Στην Εικόνα 31 φαίνονται και οι τιμές του loss κατά την διάρκεια της εκπαίδευσης του τελικού αλγορίθμου

9 Συμπέρασμα

Με σκοπό την ελαχιστοποίηση σφάλματος για αλγόριθμο τύπου convolutional neural network με εφαρμογή στην αγορά συναλλαγμάτων έγινε σύγκριση διάφορων τεχνικών επεξεργασίας δεδομένων και βελτιστοποίηση παραμέτρων.

Στο πρώτο κομμάτι όπου τα δεδομένα δεν ήταν preprocessed τα ποσοστά επιτυχίας του αλγορίθμου ήταν σχεδόν μηδενικά κάτι το οποίο φαινόταν όχι μόνο από την απώλεια κατά την εκπαίδευση (loss) και το σφάλμα αποτελεσμάτων (MSE), αλλά και δια οφθαλμού όταν φέρναμε σε αντιπαραβολή τα πραγματικά δεδομένα με τις πραγματικές τιμές.

Η διαδικασία του preprocessing των δεδομένων άλλαξε άρδην το ποσοστό επιτυχίας του αλγορίθμου και από εκεί και πέρα μπορεί να παρατηρηθεί μια σχετική επιτυχία στις προβλέψεις των αποτελεσμάτων του αλγορίθμου.

Μιας και η πρόβλεψη μελλοντικών τιμών συναλλάγματος είναι πολύ δύσκολο εγχειρήματα με το οποίο ασχολούνται πολλές χρηματιστηριακές εταιρίες, συμπεραίνουμε ότι αυτό που ξεχώρισε την επιτυχία από την επιτυχία του εγχειρήματος της πρόβλεψης τιμών συναλλάγματος είναι η διαδικασία του preprocessing.

Στην συνέχεια ακολούθησε ο λεπτός συντονισμός των παραμέτρων του αλγορίθμου ώστε να βελτιστοποιήσουμε τα αποτελέσματα που πήραμε από τον αλγόριθμο με την συγκεκριμένη δομή και αρχιτεκτονική. Λόγο περιορισμών του συστήματος δεν ήταν εφικτό να πειραματιστούμε με πολύ μεγάλες αλλά και πολύ μικρές τιμές κάποιον παραμέτρων, όπως η παράμετρος training size και data size.

Παρά τους περιορισμούς που μας επιβάλει το υπολογιστικό σύστημα μπορούν να παρατηρηθούν γραμμικές βελτιωτικές τάσεις των αποτελεσμάτων του αλγορίθμου κατά την μεταβολή όλων των παραμέτρων. Παρότι σίγουρα η βελτίωση των αποτελεσμάτων του αλγορίθμου δεν θα συνεχίζεται επ’ άπειρον, διότι θεωρείται σχεδόν σίγουρο ότι θα υπάρξει κάποια κόπωση στην καμπύλη απόδοσης, μπορούμε να πούμε με σχετική σιγουριά ότι πιθανότατα θα υπήρχε βελτίωση για τιμές μεγαλύτερες από αυτές που μας επιτρέπει το σύστημα καθώς οι μεταβολές απόδοσης που παρατηρήσαμε δεν δικαιολογούν απότομη καθοδική τάση στα αποτελέσματα για τιμές παραμέτρων οι όποιες θα ήταν λίγο μεγαλύτερες από αυτές που μας επέτρεψε το σύστημα να δοκιμάσουμε.

Η τελική γραφική παράσταση που παρουσιάζεται στην εικόνα 30 μας επιτρέπει και δια γυμνού οφθαλμού να επιβεβαιώσουμε την δυνατότητα του αλγόριθμου να ακολουθεί την τάση των τιμών συναλλαγμάτων και να μην αποκλίνει πολύ από αυτή.

Το γεγονός αυτό είναι αρκετά ενθαρρυντικό μιας και αν θέλουμε να δούμε το πρόβλημα στην πραγματική του διάσταση ο πρώτο στόχος θα ήταν να προβλέψουμε την ανοδική ή καθοδική τάση των τιμών συναλλάγματος και ως δεύτερο στόχο θα είχαμε την ακριβή πρόβλεψη της ακριβούς τιμής συναλλάγματος. Αυτό συμβαίνει διότι στην περίπτωση όπου ο στιγματισμός ήταν πραγματικός και όχι απλός ένα αντικείμενο μελέτης και βελτιστοποίησης, πρώτο μέλημα μας θα ήταν να ακολουθήσουμε την τάση και ως δεύτερο στόχο θα είχαμε την επίτευξη υψηλών αποδόσεων. Ο αντίλογος για το συγκεκριμένο επιχείρημα θα ήταν ότι αν δεν μπορούμε να προβλέψουμε με σχετική ακρίβεια την τιμή

τότε δεν θα μπορούσαμε να προβλέψουμε και την επιτυχία της συναλλαγής μιας και ο κάθε online broker χρεώνει μια προμήθεια ανά συναλλαγή και άρα δεν θα μπορούσαμε να υπολογίσουμε αν τα πιθανά κέρδη θα μπορούσαν να είναι περισσότερα από την προμήθεια του online broker.

10 Αναφορές

1. Stanford Course CS231: Convolutional Neural Networks for Visual Recognition http://cs231n.stanford.edu/
2. Neural networks for algorithmic trading. Correct time series forecasting. https://medium.com/@alexrachnog/neural-networks-for-algorithmic-trading-1-2-correct-time-series-forecasting-backtesting-9776bfd9e589
3. https://stackoverflow.com/users/190280/josh-bleecher-snyder
4. https://medium.com/@alexrachnog/neural-networks-for-algorithmic-trading-part-one-simple-time-series-forecasting-f992daa1045a
5. Marcos Lopez de Prado, Advances in Financial Machine Learning- ISBN-13: 978-1119482086