Project 3

Στεβής Χαράλαμπος-Αντωνιος AM: 1115201600278

January 21, 2022

Contents

1 Ερώτημα Α

1.1 Πως στήσαμε το πείραμα

Παίρνοντας ως template τα παραδείγματα από Time-Series Forecasting: Predicting Stock Prices Using An LSTM Model κάναμε τα εξής:
Αρχικά καθαρίσαμε το dataset και το στήσαμε έτσι ώστε να μπορεί να διαβαστεί καθαρά από την python. Βάλαμε ένα extra column με το όνομα Dates που ξεκινάει από το 2007 και σταματάει μέχρι το 2017. Αυτό θα βοηθήσει στην οπτικοποίηση του προβλήματος.
Στην συνέχεια χωρίσαμε το dataset σε training και test set με ratio περίπου 80%-20%. Κάναμε normalize τις τιμές των input ώστε να έχουν όλα το ίδιο βάρος στο τεστάρισμα.
Ύστερα στήσαμε το νευρωνικό (περισσότερα για αυτό στην συνέχεια) και το τρέξαμε. Με το μοντέλο μας βγάλαμε τις predicted τιμές και τις συγκρίναμε με τα πραγματικά αποτελέσματα.
Τέλος οπτικοποιήσαμε τα αποτελέσματα του μοντέλου μας και των πραγματικών τιμών.

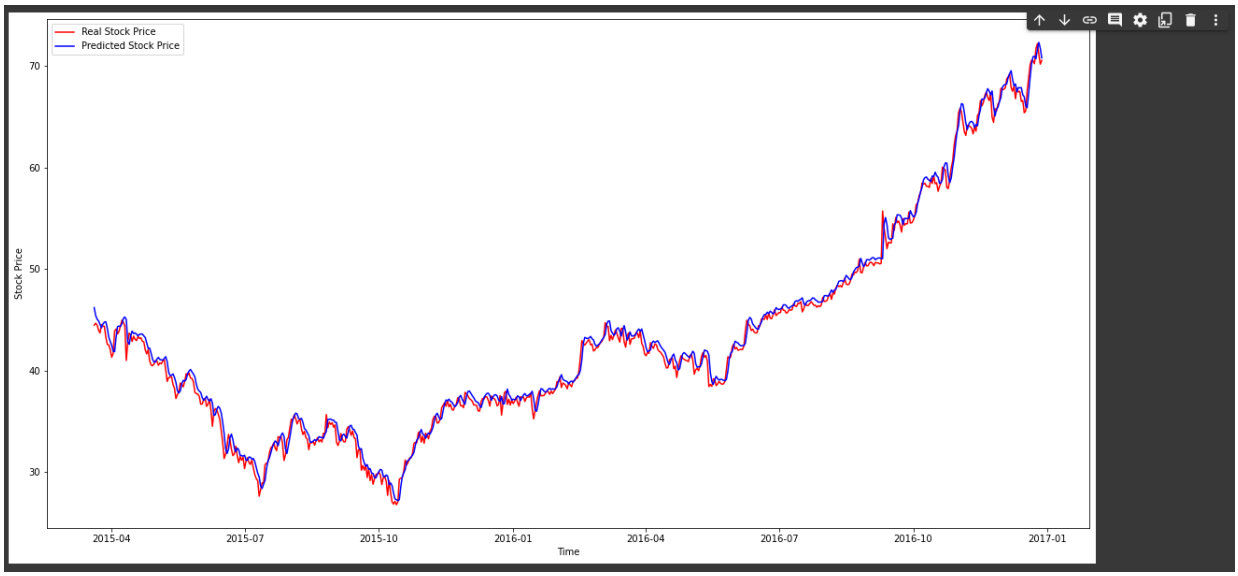
1.2 Πειράματα

1.2.1 Epochs και Batch Size

Το template με το οποίο ξεκινήσαμε είχε 4 layers με 50 κόμβους το καθένα και 1 layer για output. Ο optimizer ήταν Adam και το loss function ήταν Mean Squared Error. Τέλος το μοντέλο έκανε fit σε 100 εποχές με batch size 32 inputs. Το dropout chance σε όλα τα πειράματα παρέμεινε 0.2 γιατί ήταν αρκετό.
Ο χρόνος περίπου του default setup ήταν λεπτά και είχε τα εξής αποτελέσματα:

```
Epoch 94/100
92/92 [==============================] - 12s 128ms/step - loss: 8.1482e-04
Epoch 95/100
92/92 [==============================] - 12s 127ms/step - loss: 7.3314e-04
Epoch 96/100
92/92 [==============================] - 12s 128ms/step - loss: 7.1693e-04
Epoch 97/100
92/92 [==============================] - 12s 128ms/step - loss: 7.8510e-04
Epoch 98/100
92/92 [==============================] - 12s 126ms/step - loss: 7.5030e-04
Epoch 99/100
92/92 [==============================] - 12s 127ms/step - loss: 7.3260e-04
Epoch 100/100
92/92 [==============================] - 12s 128ms/step - loss: 7.4158e-04
```

Παρατηρούμε ότι το loss είναι αρκετά μικρό και αυτό είχε σαν κίνδυνο overfitting, κάτι που επιβεβαιώθηκε από τα γραφήματα όπως για παράδειγμα:



Στο γράφημα η μπλε και η κόκκινη γραμμή είναι σχεδόν ίδιες κάτι που αποτελέι πρόβλημα για πρόβλεψη γενικότερων inputs.

Μετά τρέξαμε την οριακή περίπτωση με 10 εποχές και 1 input στο batch size. Κάτι τέτοιο χρειάστηκε περίπου 30 λεπτά καθώς τα inputs ήταν περίπου 3000 σε κάθε εποχή. Το παραπάνω πείραμα είχε σαν αποτέλεσμα:

```
Epoch 1/10
2940/2940 [==============================] - 182s 60ms/step - loss: 0.0057
Epoch 2/10
2940/2940 [==============================] - 176s 60ms/step - loss: 0.0030
Epoch 3/10
2940/2940 [==============================] - 177s 60ms/step - loss: 0.0023
Epoch 4/10
2940/2940 [==============================] - 175s 60ms/step - loss: 0.0021
Epoch 5/10
2940/2940 [==============================] - 176s 60ms/step - loss: 0.0018
Epoch 6/10
2940/2940 [==============================] - 176s 60ms/step - loss: 0.0017
Epoch 7/10
2940/2940 [==============================] - 176s 60ms/step - loss: 0.0017
Epoch 8/10
2940/2940 [==============================] - 174s 59ms/step - loss: 0.0015
Epoch 9/10
2940/2940 [==============================] - 179s 61ms/step - loss: 0.0014
Epoch 10/10
2940/2940 [==============================] - 178s 60ms/step - loss: 0.0013
<keras.callbacks.History at 0x7f574fae4950>
```

Αυξάνοντας το batch size σε 10 μας έδωσε τα εξής αποτελέσματα:

```
# Fitting the RNN to the Training set
model.fit(X_train, y_train, epochs = 10, batch_size = 10)
```

```
Epoch 1/10
294/294 [==============================] - 10s 16ms/step - loss: 0.0085
Epoch 2/10
294/294 [==============================] - 5s 16ms/step - loss: 0.0029
Epoch 3/10
294/294 [==============================] - 5s 16ms/step - loss: 0.0025
Epoch 4/10
294/294 [==============================] - 5s 16ms/step - loss: 0.0025
Epoch 5/10
294/294 [==============================] - 5s 16ms/step - loss: 0.0023
Epoch 6/10
294/294 [==============================] - 5s 16ms/step - loss: 0.0022
Epoch 7/10
294/294 [==============================] - 5s 16ms/step - loss: 0.0018
Epoch 8/10
294/294 [==============================] - 5s 16ms/step - loss: 0.0020
Epoch 9/10
294/294 [==============================] - 5s 17ms/step - loss: 0.0019
Epoch 10/10
294/294 [==============================] - 5s 17ms/step - loss: 0.0017
```

Παρατηρούμε ότι η διαφορά στο loss δεν είναι μεγάλη και είναι φυσικό καθώς το batch size άλλαξε ελάχιστα.

Αυξάνοντας ακόμα λίγο το batch size σε 100 μας έδωσε τα εξής αποτελέσματα:

```
# Fitting the RNN to the Training set
model.fit(X_train, y_train, epochs = 10, batch_size = 100)
```

```
Epoch 1/10
30/30 [==============================] - 8s 38ms/step - loss: 0.0638
Epoch 2/10
30/30 [==============================] - 1s 38ms/step - loss: 0.0050
Epoch 3/10
30/30 [==============================] - 1s 38ms/step - loss: 0.0034
Epoch 4/10
30/30 [==============================] - 1s 38ms/step - loss: 0.0032
Epoch 5/10
30/30 [==============================] - 1s 38ms/step - loss: 0.0030
Epoch 6/10
30/30 [==============================] - 1s 38ms/step - loss: 0.0028
Epoch 7/10
30/30 [==============================] - 1s 38ms/step - loss: 0.0027
Epoch 8/10
30/30 [==============================] - 1s 38ms/step - loss: 0.0025
Epoch 9/10
30/30 [==============================] - 1s 39ms/step - loss: 0.0026
Epoch 10/10
30/30 [==============================] - 1s 39ms/step - loss: 0.0025
```

Εδώ το loss βλέπουμε ότι μεγάλωσε ακόμα περισσότερο αλλά η κλίμακα τάξης παρέμεινε ίδια.

Τέλος αλλάξαμε το batch size σε 1000 κάτι που μείωσε δραματικά τον χρόνο του πειράματος:

```
# Fitting the RNN to the Training set
model.fit(X_train, y_train, epochs = 10, batch_size = 1000)
```

```
Epoch 1/10
3/3 [==============================] - 8s 257ms/step - loss: 0.1789
Epoch 2/10
3/3 [==============================] - 1s 246ms/step - loss: 0.1108
Epoch 3/10
3/3 [==============================] - 1s 235ms/step - loss: 0.0430
Epoch 4/10
3/3 [==============================] - 1s 237ms/step - loss: 0.0415
Epoch 5/10
3/3 [==============================] - 1s 245ms/step - loss: 0.0348
Epoch 6/10
3/3 [==============================] - 1s 245ms/step - loss: 0.0206
Epoch 7/10
3/3 [==============================] - 1s 234ms/step - loss: 0.0255
Epoch 8/10
3/3 [==============================] - 1s 240ms/step - loss: 0.0206
Epoch 9/10
3/3 [==============================] - 1s 242ms/step - loss: 0.0112
Epoch 10/10
3/3 [==============================] - 1s 232ms/step - loss: 0.0103
```

Εδώ παρατηρούμε ότι το loss έχει αλλάξει τάξη μεγέθους και από 0.00x έγινε 0.0x.

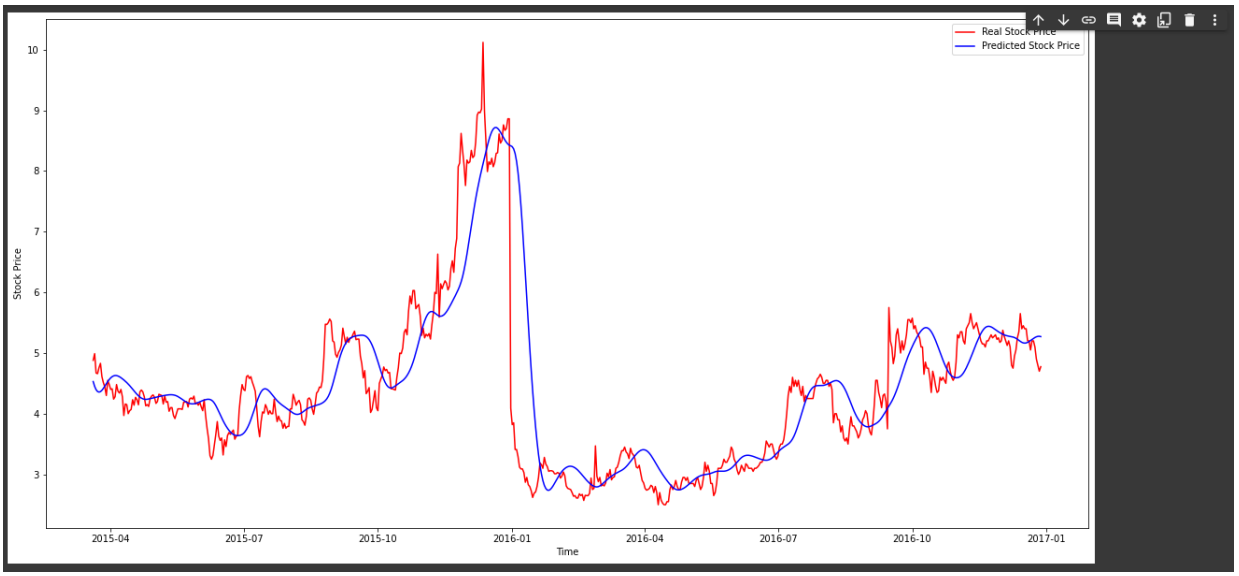
Στην συνέχεια αυξήσαμε τον αριθμό των εποχών σε 100 και τρέξαμε τα πειράματα ξανά με batch size 10,100 και 1000 αντίστοιχα:

```
Epoch 94/100
294/294 [==============================] - 5s 17ms/step - loss: 7.6756e-04
Epoch 95/100
294/294 [==============================] - 5s 17ms/step - loss: 7.6120e-04
Epoch 96/100
294/294 [==============================] - 5s 17ms/step - loss: 7.7619e-04
Epoch 97/100
294/294 [==============================] - 5s 17ms/step - loss: 7.5420e-04
Epoch 98/100
294/294 [==============================] - 5s 17ms/step - loss: 7.4855e-04
Epoch 99/100
294/294 [==============================] - 5s 17ms/step - loss: 7.3419e-04
Epoch 100/100
294/294 [==============================] - 5s 17ms/step - loss: 7.2256e-04
```

```
Epoch 94/100
30/30 [==============================] - 1s 37ms/step - loss: 0.0011
Epoch 95/100
30/30 [==============================] - 1s 38ms/step - loss: 0.0011
Epoch 96/100
30/30 [==============================] - 1s 38ms/step - loss: 0.0011
Epoch 97/100
30/30 [==============================] - 1s 38ms/step - loss: 0.0011
Epoch 98/100
30/30 [==============================] - 1s 37ms/step - loss: 0.0010
Epoch 99/100
30/30 [==============================] - 1s 37ms/step - loss: 0.0011
Epoch 100/100
30/30 [==============================] - 1s 37ms/step - loss: 0.0011
```

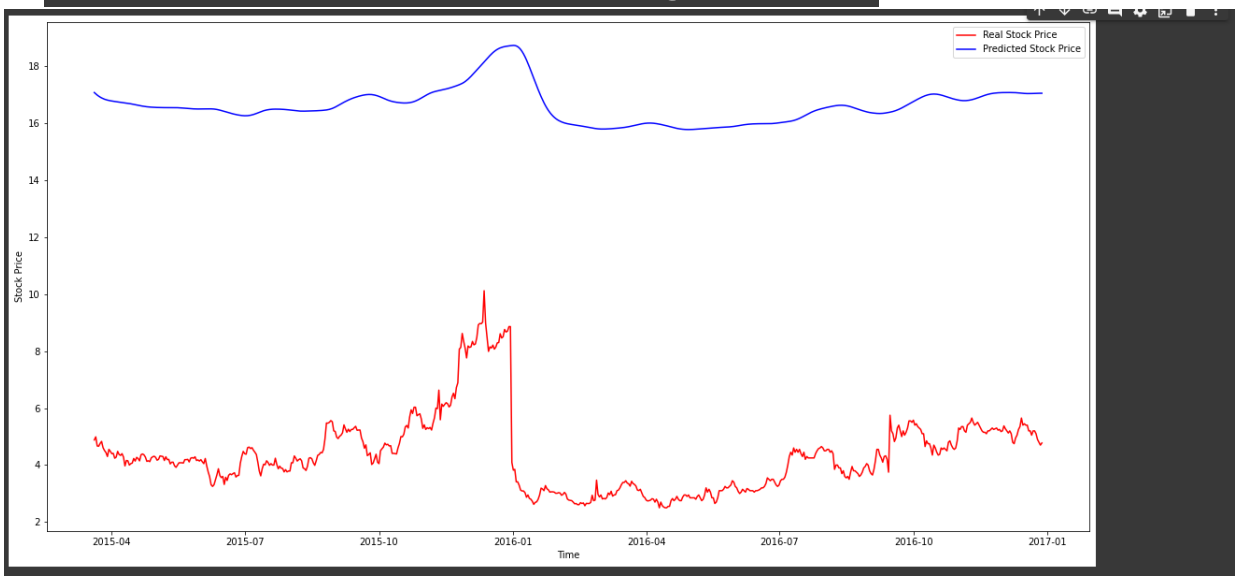
```
Epoch 94/100
3/3 [==============================] - 1s 240ms/step - loss: 0.0024
Epoch 95/100
3/3 [==============================] - 1s 239ms/step - loss: 0.0026
Epoch 96/100
3/3 [==============================] - 1s 240ms/step - loss: 0.0025
Epoch 97/100
3/3 [==============================] - 1s 231ms/step - loss: 0.0025
Epoch 98/100
3/3 [==============================] - 1s 243ms/step - loss: 0.0024
Epoch 99/100
3/3 [==============================] - 1s 234ms/step - loss: 0.0024
Epoch 100/100
3/3 [==============================] - 1s 229ms/step - loss: 0.0023
```

Όπως βλέπουμε όσο μικρότερο το batch size τόσο μικρότερο loss. Ωστόσο το γράφημα του 100,1000 ήταν το ιδανικό ώστε να αποφύγουμε το overfitting:


Real Stock Price
Predicted Stock Price
Stock Price
Time
10
9
8
7
6
5
4
3
2015-04
2015-07
2015-10
2016-01
2016-04
2016-07
2016-10
2017-01

Τέλος θέλαμε να τρέξουμε σχετικά όριακές περιπτώσεις όπως το epochs=10, batch size = 1. Έτσι δοκιμάσαμε 10 εποχές αλλά με 3000 batch size (όσο και το μέγεθος των input , που σημαίνει ότι περνάει μόνο μία φορά από κάθε input) και είχαμε τα εξής αποτελέσματα:

```
Epoch 5/10
1/1 [==============================] - 4s 4s/step - loss: 0.0139
Epoch 6/10
1/1 [==============================] - 4s 4s/step - loss: 0.0362
Epoch 7/10
1/1 [==============================] - 4s 4s/step - loss: 0.0433
Epoch 8/10
1/1 [==============================] - 4s 4s/step - loss: 0.0219
Epoch 9/10
1/1 [==============================] - 4s 4s/step - loss: 0.0108
Epoch 10/10
1/1 [==============================] - 4s 4s/step - loss: 0.0131
```



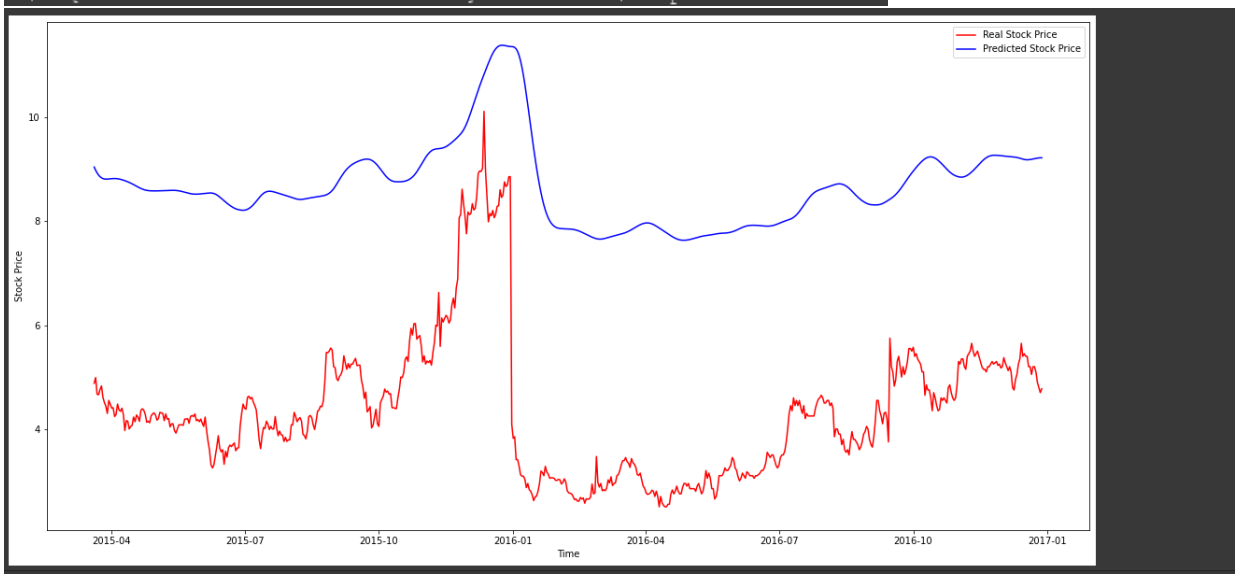
Εδώ παρατηρούμε ότι οι λίγες εποχές δεν έχουν αφήσει το μοντέλο να κάνει fit και έτσι έχουμε περίπτωση underfitting.

1.2.2 Layers και Unit

Χρησιμοποιόντας 100 εποχές και 1000 batch size αρχίσαμε να πειραματιζόμαστε με τα layers και τους κόμβους.

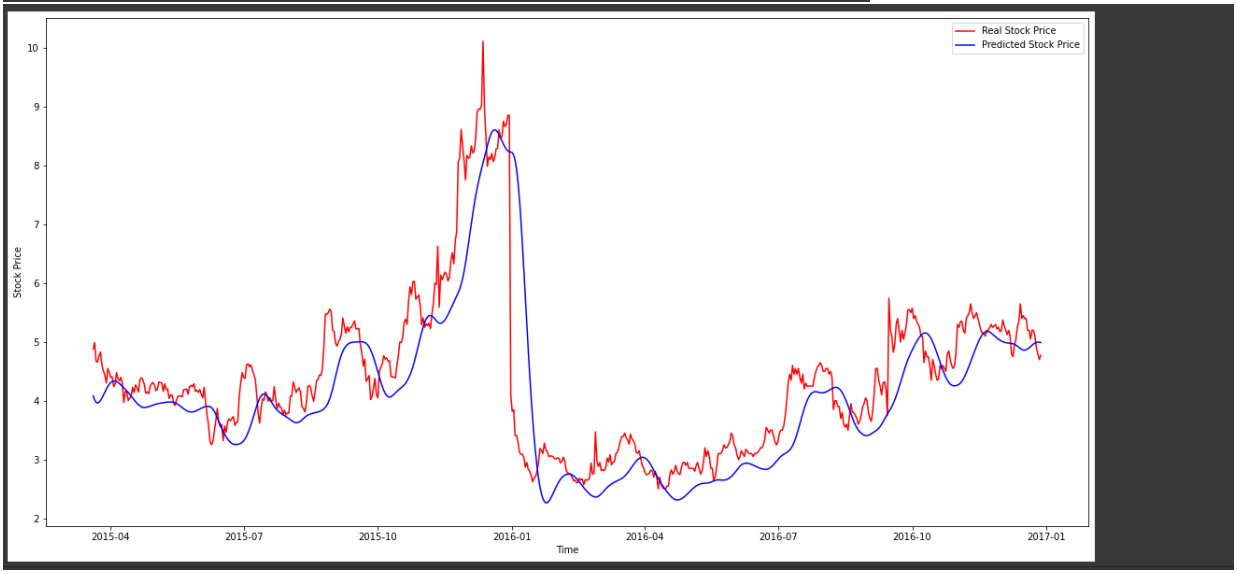
Αρχικά μειώσαμε τους κόμβους σε 10. Αν και το fitting έτρεξε πολύ πιο γρήγορα τα αποτελέσματα του loss και της οπτικοποίησης ήταν κακά όπως είναι φυσικό. Μειώσαμε την πολυπλοκότητα και έτσι κερδίσαμε χρόνο όμως δεν δώσαμε χώρο στο νευρονικό να εξελιχθεί:

```
Epoch 94/100
3/3 [==============================] - 1s 284ms/step - loss: 0.0067
Epoch 95/100
3/3 [==============================] - 1s 283ms/step - loss: 0.0067
Epoch 96/100
3/3 [==============================] - 1s 286ms/step - loss: 0.0069
Epoch 97/100
3/3 [==============================] - 1s 258ms/step - loss: 0.0071
Epoch 98/100
3/3 [==============================] - 1s 282ms/step - loss: 0.0070
Epoch 99/100
3/3 [==============================] - 1s 292ms/step - loss: 0.0066
Epoch 100/100
3/3 [==============================] - 1s 288ms/step - loss: 0.0065
```



Στην συνέχεια δοκιμάσαμε με 4 layers και 100 units σε κάθε κόμβο κάτι που αύξησε το χρόνο αλλά βελτίωσε το loss:

```
Epoch 94/100
3/3 [==============================] - 10s 3s/step - loss: 0.0022
Epoch 95/100
3/3 [==============================] - 10s 3s/step - loss: 0.0021
Epoch 96/100
3/3 [==============================] - 10s 3s/step - loss: 0.0021
Epoch 97/100
3/3 [==============================] - 10s 3s/step - loss: 0.0022
Epoch 98/100
3/3 [==============================] - 10s 3s/step - loss: 0.0021
Epoch 99/100
3/3 [==============================] - 10s 3s/step - loss: 0.0021
Epoch 100/100
3/3 [==============================] - 10s 3s/step - loss: 0.0021
```



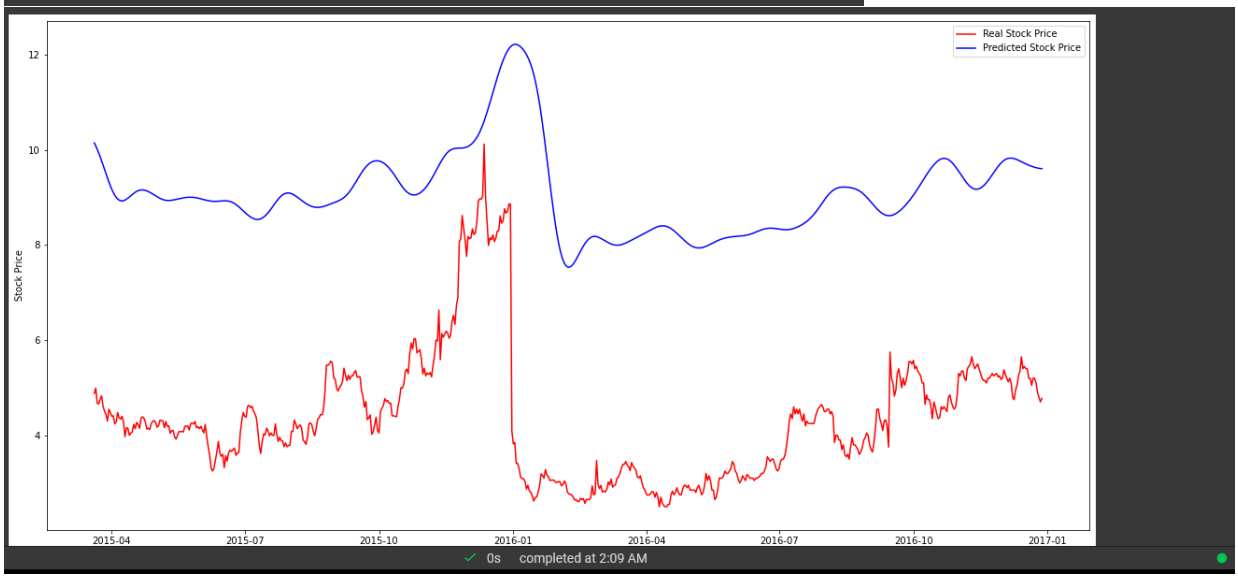
Παρατηρούμε ότι το loss έχει μειωθεί αρκετά και η οπτικοποίηση είναι πολύ καλύτερη, με κόστος βέβαια τον χρόνο (περίπου 20 λεπτά).

Τέλος δοκιμάσαμε το πείραμα με 100 εποχές, 1000 batch size και 50 κάμβους αλλάζοντας τα layers. Αρχικά τρέξαμε το πείραμε με 1 layer και ακόμα ένα για το output:

```
Epoch 94/100
3/3 [==============================] - 2s 625ms/step - loss: 0.0664
Epoch 95/100
3/3 [==============================] - 2s 635ms/step - loss: 0.0664
Epoch 96/100
3/3 [==============================] - 2s 636ms/step - loss: 0.0664
Epoch 97/100
3/3 [==============================] - 2s 635ms/step - loss: 0.0664
Epoch 98/100
3/3 [==============================] - 2s 621ms/step - loss: 0.0664
Epoch 99/100
3/3 [==============================] - 2s 621ms/step - loss: 0.0663
Epoch 100/100
3/3 [==============================] - 2s 577ms/step - loss: 0.0663
```

Βλέπουμε ότι το σφάλμα δεν αλλάζει κάτι που είναι φυσικό καθώς τα inputs δεν περνάνε από επεξεργασία. Στην συνέχεια δοκιμάσαμε 10 layers:

```
Epoch 94/100
3/3 [==============================] - 10s 3s/step - loss: 0.0043
Epoch 95/100
3/3 [==============================] - 10s 3s/step - loss: 0.0043
Epoch 96/100
3/3 [==============================] - 10s 3s/step - loss: 0.0045
Epoch 97/100
3/3 [==============================] - 10s 3s/step - loss: 0.0042
Epoch 98/100
3/3 [==============================] - 10s 3s/step - loss: 0.0042
Epoch 99/100
3/3 [==============================] - 10s 3s/step - loss: 0.0041
Epoch 100/100
3/3 [==============================] - 10s 3s/step - loss: 0.0041
```



Ενώ βλέπουμε ότι το loss είναι σχετικά καλό η οπτικοποίηση του προβλήματος μας πληροφορεί για το αντίθετο.

1.2.3 Optimizer και Loss Function

Για ό,τι τρέχαμε μέχρι τώρα χρησιμοποιούσαμε τον optimizer adam. Τρέξαμε το πρόγραμμα με SGD, Adamax και RMSProp και είχαμε τα εξής αποτελέσματα:

SGD:

```
Epoch 94/100
3/3 [==============================] - 4s 1s/step - loss: 0.0048
Epoch 95/100
3/3 [==============================] - 4s 1s/step - loss: 0.0045
Epoch 96/100
3/3 [==============================] - 4s 1s/step - loss: 0.0046
Epoch 97/100
3/3 [==============================] - 4s 1s/step - loss: 0.0049
Epoch 98/100
3/3 [==============================] - 4s 1s/step - loss: 0.0047
Epoch 99/100
3/3 [==============================] - 4s 1s/step - loss: 0.0046
Epoch 100/100
3/3 [==============================] - 4s 1s/step - loss: 0.0049
```

Adam:

```
Epoch 94/100
3/3 [==============================] - 4s 1s/step - loss: 0.0035
Epoch 95/100
3/3 [==============================] - 4s 1s/step - loss: 0.0034
Epoch 96/100
3/3 [==============================] - 4s 1s/step - loss: 0.0034
Epoch 97/100
3/3 [==============================] - 4s 1s/step - loss: 0.0033
Epoch 98/100
3/3 [==============================] - 4s 1s/step - loss: 0.0035
Epoch 99/100
3/3 [==============================] - 4s 1s/step - loss: 0.0032
Epoch 100/100
3/3 [==============================] - 4s 1s/step - loss: 0.0034
```

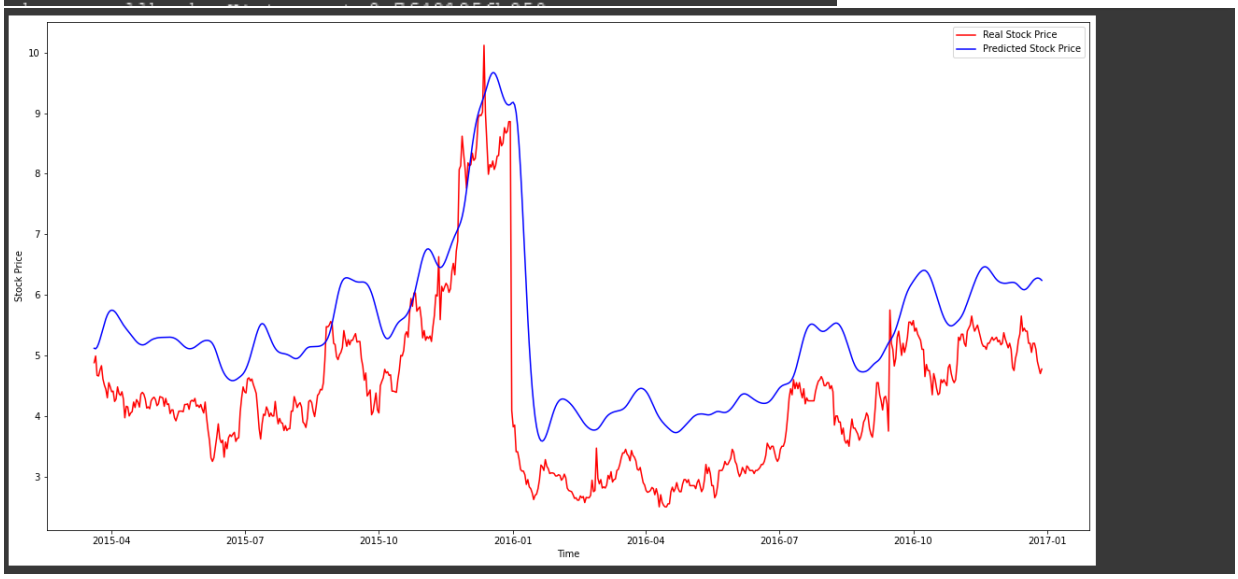
RMSProp:

```
Epoch 94/100
3/3 [==============================] - 4s 1s/step - loss: 0.0031
Epoch 95/100
3/3 [==============================] - 4s 1s/step - loss: 0.0044
Epoch 96/100
3/3 [==============================] - 4s 1s/step - loss: 0.0057
Epoch 97/100
3/3 [==============================] - 4s 1s/step - loss: 0.0047
Epoch 98/100
3/3 [==============================] - 4s 1s/step - loss: 0.0035
Epoch 99/100
3/3 [==============================] - 4s 1s/step - loss: 0.0033
Epoch 100/100
3/3 [==============================] - 4s 1s/step - loss: 0.0045
```

Όσο για την οπτικοποίηση κανένας από τους παραπάνω optimizers δεν έφτασε την εγκυρώτητα και την ακρίβεια του Adam χωρίς φυσικά overfitting, με εξαίρεση τον RMSProp που ήταν αρκετά ακριβές.

Τέλος για τα functions loss είχαμε default την Mean squared Error και τρέξαμε επίσης την Mean Absolute Error με τα εξής αποτελέσματα:

```
Epoch 94/100
3/3 [==============================] - 4s 1s/step - loss: 0.0315
Epoch 95/100
3/3 [==============================] - 4s 1s/step - loss: 0.0316
Epoch 96/100
3/3 [==============================] - 4s 1s/step - loss: 0.0322
Epoch 97/100
3/3 [==============================] - 4s 1s/step - loss: 0.0312
Epoch 98/100
3/3 [==============================] - 4s 1s/step - loss: 0.0313
Epoch 99/100
3/3 [==============================] - 4s 1s/step - loss: 0.0316
Epoch 100/100
3/3 [==============================] - 4s 1s/step - loss: 0.0313
```



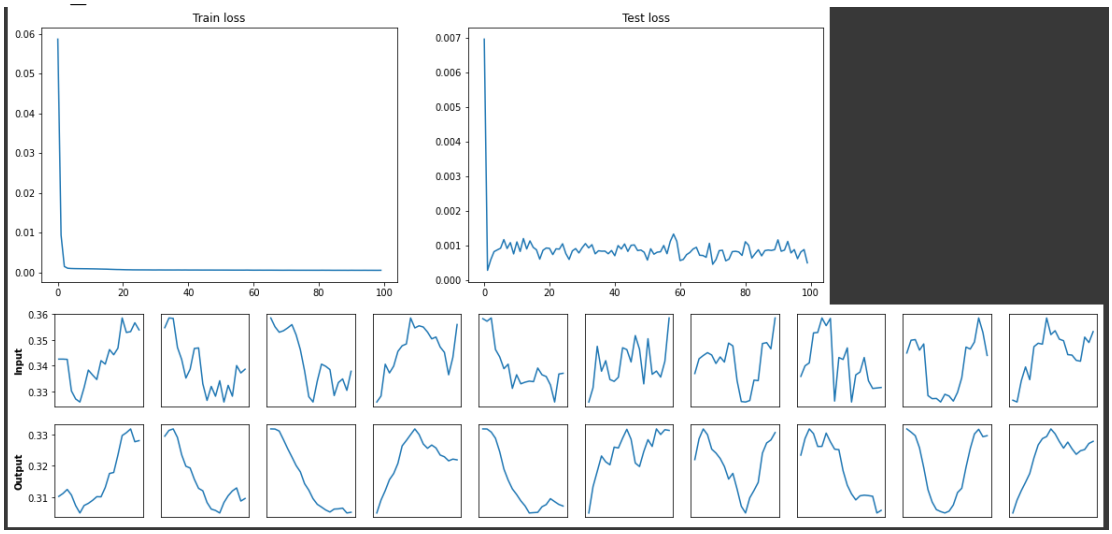
2 Ερώτημα Γ

Σχόλια, τιμές δοκιμών και αλγόριθμοι είναι μέσα στην εργασία, εδώ απλώς αναφέρονται ποιά μοντέλα χρησιμοποιήθηκαν και με ποιές τιμές παράχθηκαν τα αρχεία για το ερώτημα Δ που ακολουθεί:
Γενικές μεταβλητές που μένουν ίδιες στα πρώτα τα μοντέλα:

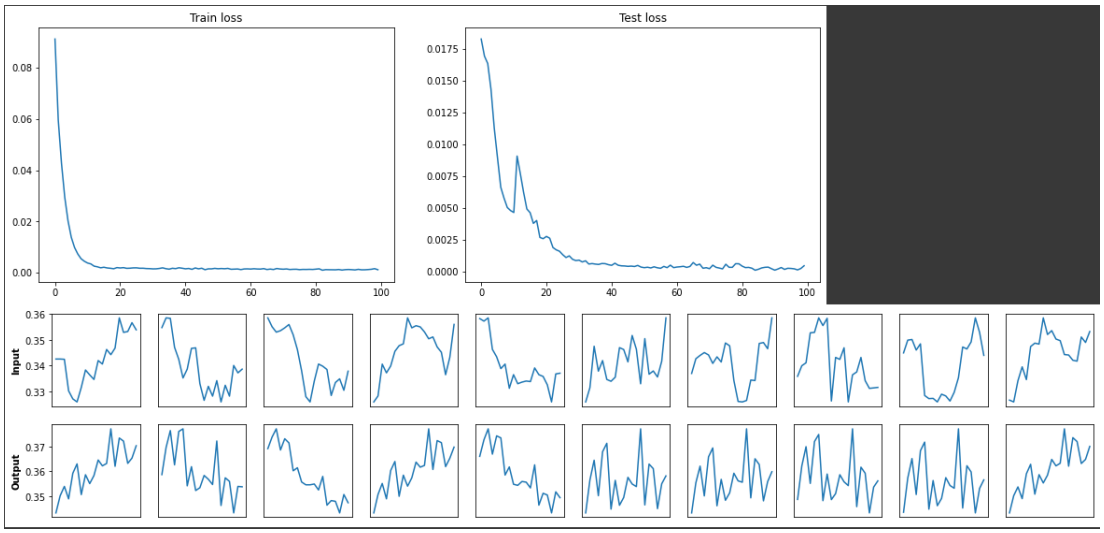
1. startdate = "01/01/2007"
2. window_length = 20
3. encoding_dim = 10
4. epochs = 100
5. test_samples = 500

Μοντέλα:

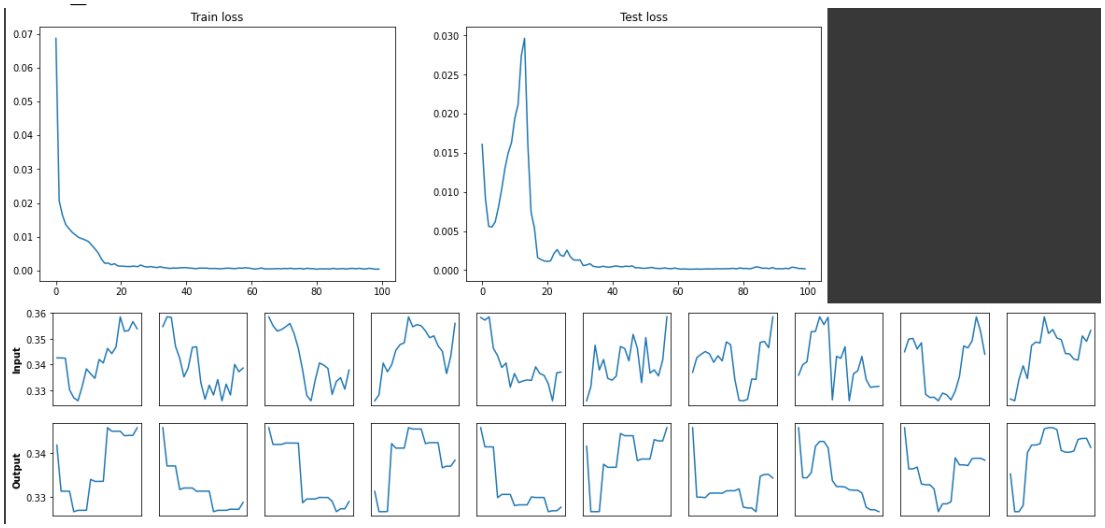
1. Simple feed-forward autoencoder -> optimizer=adam, loss=mean_squared_error, batch_size=10



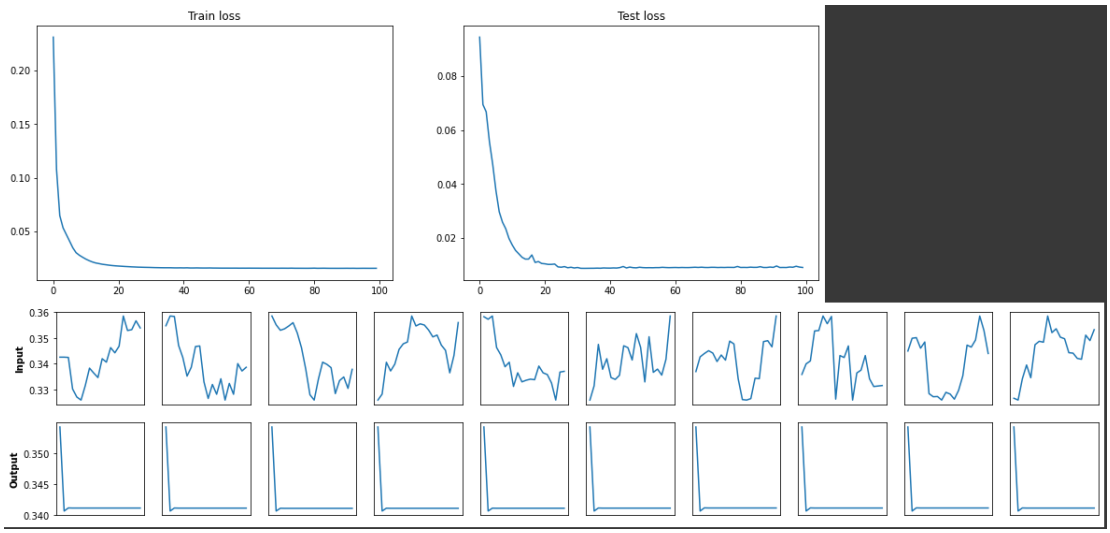
2. Deep autoencoder -> optimizer='adam', loss='mean_squared_error', batch_size=60



3. 1D Convolutional autoencoder -> optimizer='adam', loss='mean_squared_error', batch_size=150



4. LSTM -> optimizer='adam', loss='mean_absolute_error', batch_size=100

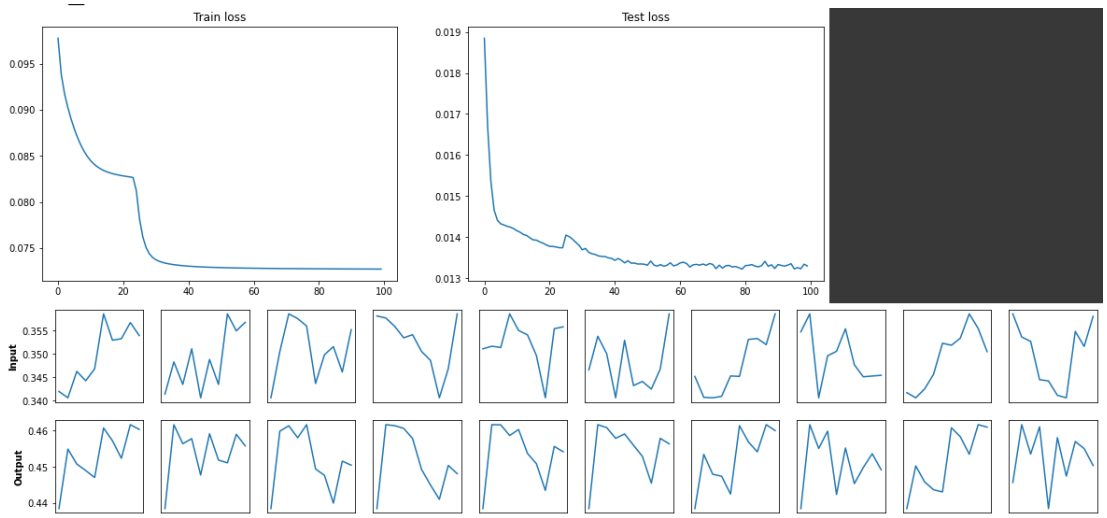


Για τα δύο τελευταία χρησιμοποιήθηκαν διαφορετικοί παράμετροι:

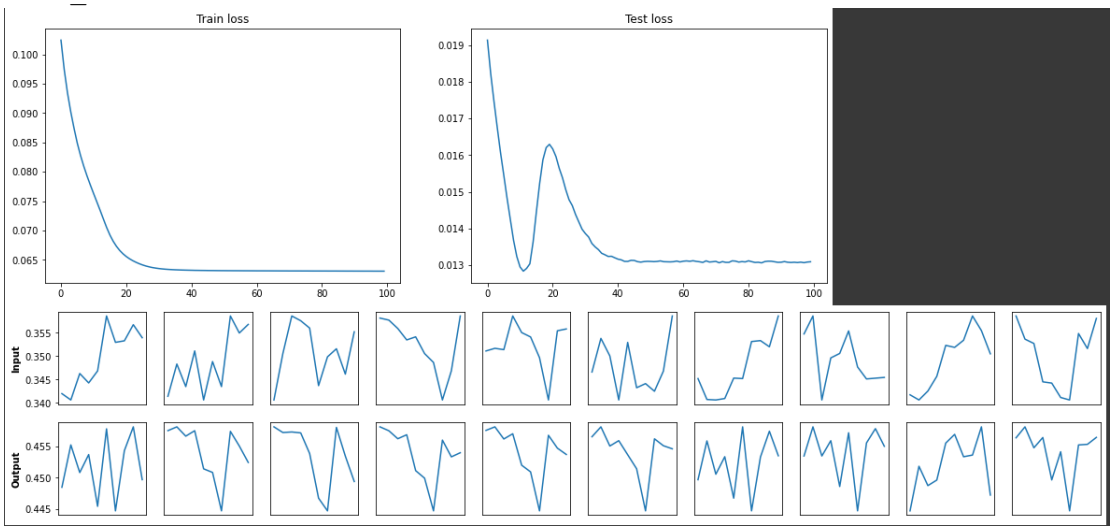
1. startdate = "01/01/2007"
2. window_length = 10
3. encoding_dim = 3
4. epochs = 100
5. test_samples = 500

Μοντέλα:

1. Simple AE + augmention with synthetic datar -> optimizer=adam, loss=mean_squared_error, batch_size=2000



2. Deep autoencoder + synthetic data -> optimizer='adam', loss='mean_squared_error', batch_size=6000



3 Ερώτημα Δ

3.1 Γενικά

Τα αρχεία στα οποία έγιναν οι συγκρίσεις θα βρίσκονται μαζί με την εργασία. Το query file παρέμενε ίδιο σε όλους τους αλγορίθμους και αποτελείτε από τα τελευταία 10 queries του dataset που δώθηκε.
Τρέξαμε τα μοντέλα που αναφέρονται παραπάνω με τους εξής αλγορίθμους: LSH, HyperCube, LSH with Discrete Frechet.
Συνολικά παράχθηκαν 21 αρχεία, 3 εκ των οποίων είναι οι αρχικοί αγλόριθμοι και 18 εκ των οποίων είναι predicted dataset από τα μοντέλα. Οι συγκρίσεις γίνονται μεταξύ των αλγορίθμων original dataset με dataset των μοντέλων.

3.2 Παρατηρήσεις

3.2.1 LSH

Σε όλα τα αρχεία των μοντέλων φαίνεται πως το distanceLSH και το distanceTrue είναι τα ίδια. Επίσης παρατηρούμε ότι οι Approximate και True Neighbors είναι επίσης οι ίδιοι. Κάτι τέτοιο δεν συμβαίνει στο original αρχείο. Τέλος οι χρόνοι για την εύρεση του NN φαίνεται πως είναι πάντα 0 για τον true χρόνο ενώ για τον LSH χρόνο είναι 80+.

3.2.2 Hypercube

Εδώ παρατηρούμε διαφοτερικά αποτελέσματα. Approximate με True neighbor διαφέρουν καθώς διαφορετικές είναι συνήθως και οι αποστάσεις. Οι χρόνοι συνεχίζουν στα προηγούμενα πρότυπα με χρόνους Cube να βρίσκονται στα 30+ ενώ οι πραγματικοί χρόνοι είναι μηδενικοί.

3.2.3 Frechet

Τέλος τα αποτελέσματα για το frechet ακολουθούν τα πρότυπα του LSH αφού χρησιμοποιούν το ίδιο αλγόριθμο και έτσι βλέπουμε ίδια μεταξύ τους distances και neighbors ενώ πάλι οι χρόνοι φάνονται να είναι 80+ για τον Frechet και 0 για τον true χρόνο.