Νευρο-Ασαφής Υπολογιστική (Neuro-Fuzzy Computing)

Fall Semester 2024-2025

Δημήτριος Κατσαρός (prof. Dimitrios Katsaros)

Coding_project

Νικόλαος-Παναγιώτης Ρήνος 03537

Ιωάννης Καλλέργης 03472

- Στην παρούσα εργασία στόχος είναι να αναπτύξουμε μία μεθοδολογία πρόβλεψης της ημερήσιας τιμής του Bitcoin χρησιμοποιώντας κάποιο είδος νευρωνικού της επιλογής μας.

  Το σύνολο των δεδομένων που δόθηκαν βρίσκονται στο αρχείο με όνομα «BTC-2021_01_01_2022_03_01_min_basis.csv» και περιλαμβάνει τις εξής πληροφορίες :

  - Τον χρόνο unix, δηλαδή τον αριθμό των μη-δίσεκτων δευτερολέπτων που έχουν περάσει από τις 00:00:00 UTC της 1ης Ιανουαρίου 1970.
  - Το πεδίο date που αναφέρεται στην ημερομηνία και ώρα (στην μορφή yyyy-MM-dd HH:mm:ss) της συναλλαγής.
  - Το πεδίο symbol, με σταθερή τιμή BTC/USD.
  - Το πεδίο open, δηλαδή την τιμή ανοίγματος του Bitcoin στην συγκεκριμένη ημερομηνία και ώρα.
  - Το πεδίο high, δηλαδή την μέγιστη τιμή του Bitcoin στην συγκεκριμένη ημερομηνία και ώρα.
  - Το πεδίο low, δηλαδή την ελάχιστη τιμή του Bitcoin στην συγκεκριμένη ημερομηνία και ώρα.
  - Το πεδίο close, δηλαδή την τιμή κλεισίματος του Bitcoin στην συγκεκριμένη ημερομηνία και ώρα.
  - Το πεδίο VolumeBTC, δηλαδή την ποσότητα των Bitcoins που ανταλλάχθηκε στην συγκεκριμένη ημερομηνία και ώρα.
  - Το πεδίο VolumeUSD, δηλαδή την ποσότητα των συναλλαγών σε δολάρια που ανταλλάχθηκε στην συγκεκριμένη ημερομηνία και ώρα.

  Από αυτές τις πληροφορίες, χρήσιμες κρίνονται εκείνες που αντιστοιχούν στα πεδία date και close.

  Ο κώδικας που περιέχει το νευρωνικό δίκτυο το οποίο δημιουργήθηκε ώστε να κάνει τις επιθυμητές προβλέψεις, αναπτύχθηκε σε MATLAB και περιλαμβάνεται μαζί με τα υπόλοιπα παραδοτέα στο αρχείο με όνομα «Coding_project.m».

- **Περιγραφή του κώδικα:**

  Αρχικά, εισάγονται τα επιθυμητά δεδομένα, date και close, από το αρχείο «BTC-2021_01_01_2022_03_01_min_basis.csv», μέσω της συνάρτησης `import_data(filename)`.

  Έπειτα, καλώντας την συνάρτηση `seperate_data(data_date, data_close)`, τα δεδομένα διαχωρίζονται σε εκείνα που θα χρειαστούν για το training του νευρωνικού (`train_data`) και σε εκείνα για το testing (`test_data`). Ακόμη, η συνάρτηση επιστρέφει τις ημερομηνίες στις οποίες αντιστοιχούν στα δεδομένα του test και θα

χρησιμοποιηθούν αργότερα για την οπτικοποίηση των αποτελεσμάτων (`test_dates`). Τα δεδομένα μέχρι και πριν τις 20 Φεβρουαρίου του 2022 θα χρησιμοποιηθούν για εκπαίδευση, ενώ εκείνα από τις 20 Φεβρουαρίου μέχρι και 1η Μαρτίου του 2022 θα χρησιμοποιηθούν για δοκιμή.

Τα παραπάνω δεδομένα, λοιπόν, κανονικοποιούνται βασιζόμενοι στην φόρμουλα Min-Max, μέσω της ανώνυμης συνάρτησης `normal_data = @(x)`. Έτσι, δημιουργούνται οι βάσεις δεδομένων για testing και training, καλώντας την συνάρτηση `create_database(data)`, η οποία επιστρέφει ένα ζεύγος εισόδων και εξόδων `[input, output]`. Τα ζεύγη αυτά προκύπτουν με βάση ένα «παράθυρο» μεγέθους 10, το οποίο κινείτε ανά μία θέση δημιουργώντας τις αλληλουχίες εισόδων `input_seq = data(i : i + window_size – 1)` και τις αντίστοιχες τιμές εξόδου τους `output_val = data(i + window_size)`.

Στην συνέχεια, ορίζονται η αρχιτεκτονική του νευρωνικού δικτύου που αναπτύξαμε (`layers = [...]`) και το σύνολο των παραμέτρων για την εκπαίδευση του (`options = [...]`). Έτσι, μέσω της συνάρτησης `trainNetwork(...)` εκπαιδεύουμε το νευρωνικό και έπειτα κάνει προβλέψεις με βάση τις αλληλουχίες εισόδων των δεδομένα δοκιμής, καλώντας την συνάρτηση `predict(net, test_input)`. Παράλληλα, καταγράφουμε και τους χρόνους εκπαίδευσης και απόκρισης του δικτύου (`training_time` και `inferense_time` αντίστοιχα).

Ακόμη, συγκρίνουμε την επίδοση της μεθόδου μας με την αντίστοιχη μιας ARIMA τεχνικής, κάνουμε εκτίμηση των παραμέτρων του με βάση τα δεδομένα εκπαίδευσης `estimate(...)` και εκτελούμε τις προβλέψεις του μοντέλου `forecast(...)`.

Τέλος, υπολογίζεται το μέσο τετραγωνικό σφάλμα για την μέθοδο που αναπτύξαμε και την τεχνική ARIMA, και αφού εκτυπωθούν στην έξοδο τα MSE, και οι χρόνοι εκπαίδευσης και απόκρισης, οπτικοποιούνται οι πραγματικές τιμές του Bitcoin για το τελευταίο δεκαήμερο του Φεβρουαρίου και οι αντίστοιχες προβλέψεις που έκανε το νευρωνικό δίκτυο που δημιουργήσαμε.

Είναι, επίσης σημαντικό να επισημανθεί ότι τα αποτελέσματα των προβλέψεων και για τις δύο μεθόδους, καθώς και οι πραγματικές τιμές του Bitcoin για το επιθυμητό χρονικό διάστημα, υπέστησαν αντίστροφη κανονικοποίηση στο τέλος, μέσω της ανώνυμης συνάρτησης `rev_normal_data = @(x_normal, min_x, max_x)`.

- **Λεπτομέρειες της αρχιτεκτονικής του νευρωνικού δικτύου:**

  Η αρχιτεκτονική του νευρωνικού δικτύου που αναπτύξαμε, με στόχο να προβλέψει τις ημερήσιες τιμές του Bitcoin για το τελευταίο δεκαήμερο του Φεβρουαρίου, βασίζεται σε ένα Long Short-term Memory – LSTM. Όπως γνωρίζουμε τα LSTM είναι μορφή τεχνητών ανατροφοδοτούμενων νευρωνικών δικτύων, που επιτρέπουν την επεξεργασία δεδομένων ως σειρές. Ο λόγος για τον οποίο επιλέξαμε να χρησιμοποιήσουμε το LSTM, είναι εξαιτίας της ικανότητας του να χειρίζεται σύνθετα προβλήματα χρονοσειρών και να μαθαίνει μακροχρόνια μοτίβα, καθώς και πολύπλοκες σχέσεις στα δεδομένα.

Η αρχιτεκτονική, λοιπόν, του νευρωνικού δικτύου περιλαμβάνει:

- Αρχικά ένα input layer, που δέχεται ως είσοδο 10 χαρακτηριστικά για κάθε χρονικό βήμα της ακολουθίας.
- Έπειτα, ένα LSTM layer, με 125 κρυφές μονάδες, που αντιστοιχούν στην ποσότητα πληροφορίας που θα θυμάται το layer μεταξύ των χρονικών βημάτων. Αυτός ο αριθμός είναι αρκετός ώστε το layer να μην κάνει overfit στα δεδομένα εκπαίδευσης που θα οδηγούσε σε χαμηλή απόδοση του μοντέλου σε νέα δεδομένα. Ακόμη, ο τρόπος εξόδου έχει καθοριστεί ως “last”, δηλαδή εξάγεται μόνο το τελευταίο χρονικό βήμα της ακολουθίας.
- Στην συνέχεια, ακολουθεί ουσιαστικά ένα dense layer, την λειτουργία του οποίου αναλαμβάνει το πλήρως συνδεδεμένο layer με μέγεθος εξόδου ίσο με ένα, την προβλεπόμενη δηλαδή τιμή του Bitcoin.
- Τέλος, υπάρχει ένα regression layer το οποίο υπολογίζει τη μισή μέση τετραγωνική απόκλιση (half-mean-squared-error) ανάμεσα στις προβλέψεις που κάνει το δίκτυο και τις πραγματικές τιμές.

- **Λεπτομέρειες για τον αλγόριθμο εκπαίδευσης:**

Για την εκπαίδευση του μοντέλου γίνεται χρήση του αλγορίθμου βελτιστοποίησης Adam (Adaptive Moment Estimation), ο οποίος βοηθάει στην αποδοτική προσαρμογή των παραμέτρων του. Επιπλέον, ο μέγιστος αριθμός από epochs, δηλαδή ο αριθμός των επαναλήψεων στα δεδομένα εκπαίδευσης, ορίζεται ίσος με 100 έτσι ώστε το μοντέλο να βελτιώσει τις παραμέτρους και την απόδοση του. Προκειμένου, ακόμη, να αυξηθεί η σύγκλιση, επιλέγεται ένα learning rate με αρχική τιμή ίση με 0,001. Τέλος, τα δεδομένα εκπαίδευσης θα ανακατεύονται πριν από κάθε epoch, ενώ το όριο κλίμακας του gradient είναι 1. Αν αυτή η τιμή ξεπεραστεί το gradient περιορίζεται σύμφωνα με την επιλογή GradientThresholdMethod = "l2norm" η οποία ορίζεται default από το MATLAB.

Ο ψευδοκώδικας του αλγορίθμου Adam :

```
Initialize learning rate α = 0.001
Initialize decay parameter for moment and squared moment β1=0.9, β2=0.999
Initialize parameter η = 10^-8
Initialize 1st moment vector m0 ← 0
Initialize 2nd moment vector v0 ← 0
Initialize step i ← 0
while θi not converged do
      i ← i + 1
      gi ← ∇θ fi(θi-1)
      mi ← β1·mi-1 + (1 − β1)·gi
      νi ← β2·νi-1 + (1 − β2)·gi^2
      m̂i ← mi/(1 − β1^i)
      ν̂i ← νi/(1 − β2^i)
      θi ← θi-1 − α·m̂i/(√ν̂i + η)
end while
return θi (resulting parameters)
```

Βέβαια, θα πρέπει να επισημάνουμε ότι δοκιμάσαμε και άλλους αλγόριθμος βελτιστοποίησης πέρα από τον Adam, όπως την μέθοδο SGDM(Stochastic Gradient Descent with Momentum), η οποία βοηθά στην επιτάχυνση των διανυσμάτων gradient προς τις σωστές κατευθύνσεις, οδηγώντας σε ταχύτερη σύγκλιση. Ακόμη, δοκιμάσαμε και τον αλγόριθμο RMSProp(Root Mean Square Propagation), ο οποίος προσαρμόζει τον ρυθμό εκμάθησης για κάθε παράμετρο ξεχωριστά λαμβάνοντας υπόψιν το μέγεθος των πρόσφατων gradients για αυτές τις παραμέτρους. Ωστόσο, σε καμία από τις

προηγούμενες δύο περιπτώσεις δεν είχαμε καλύτερα αποτελέσματα απ' ότι με τον αλγόριθμο Adam.

- **Οι τελικές (εκπαιδευμένες) τιμές των παραμέτρων του νευρωνικού δικτύου**

  Οι τελικές (εκπαιδευμένες) τιμές των παραμέτρων αποθηκεύονται, από τον κώδικα που αναπτύξαμε, στο αρχείο «trained_parameters.txt» και αναλόγως το επίπεδο περιλαμβάνουν :

  - Για το LSTM Layer, τα βάρη εισόδου (input weights), τα αναδρομικά βάρη (recurrent weights) και τα biases. Κάθε LSTM unit έχει τέσσερα σύνολα βαρών, από τα δεδομένα προς την είσοδο και τις τρεις πύλες (Input, Output και Forget Gates) και τέσσερα σύνολα αναδρομικών βαρών, από την έξοδο προς την είσοδο και τις τρεις πύλες.
  - Για το Dense Layer (Fully Connected Layer), τα βάρη και το bias.
  - Το Input Layer (Sequence Input Layer) και το Regression Layer δεν έχουν παραμέτρους.

  Λόγω μεγάλου αριθμού των input weights, recurrent weights και biases του LSTM layer, δεν περιέχεται όλο το σύνολο τους στην αναφορά, αλλά ένα ενδεικτικό μέρος αυτών. Τα πλήρη σύνολα των παραμέτρων αυτών διατίθενται στο αρχείο «trained_parameters.txt», που περιλαμβάνεται στα τελικά παραδοτέα.

  Για το LSTM Layer :

Input Weights :	Recurrent Weights :	Biases:
-0.038817	-0.000000	10.662134
0.000000	0.000000	-0.638774
-0.000000	0.000000	-0.654688
0.000000	0.000000	-0.634028
0.000000	0.000000	-0.664497
0.000003	-0.000000	10.662282
-0.000000	-0.000000	-0.449586
0.000011	-0.000000	0.962201
0.000000	-0.000000	-0.575862
0.000000	0.000000	-0.677256
-0.000000	0.000000	-0.456335
0.000000	0.000000	-0.639768
-0.000044	-0.000000	1.047248
-0.038818	-0.000000	1.000000
0.038812	0.000000	1.000000
0.038818	-0.000000	1.000000
0.000000	-0.000000	1.000000
0.000000	-0.000000	1.000000
-0.000000	0.000000	1.000000
0.000000	0.000000	1.000000
0.000000		
-0.038818		
0.000000		
0.000000		
0.000001		
-0.000000		

Για το Dense Layer :

Weights :			
-0.000000	-0.000000	0.000000	0.000018
0.107476	0.000000	0.000016	0.000000
-0.000000	0.000002	0.000000	0.000000
0.000000	-0.107483	0.000001	-0.000000
0.107478	-0.107472	-0.000000	-0.000000
0.000000	0.107470	0.000000	-0.107481
0.000000	0.000000	-0.000075	-0.000000
0.000130	0.000000	-0.107483	0.000000
0.107481	-0.000000	0.107467	0.000000
-0.000000	-0.000047	0.107483	0.000000
-0.000000	0.000000	0.000002	-0.107481
0.000001	-0.000000	0.000000	-0.000000
-0.107483	-0.000000	-0.000000	0.000000
-0.000000	-0.000000	0.000000	-0.000000
-0.000000	-0.000000	0.000000	0.000000
0.000000	0.000000	-0.107482	-0.000000
0.000000	0.000000	-0.000000	0.000000
0.107492	-0.000000	0.000000	-0.000000
-0.000001	0.000002	0.000000	Bias :
-0.000000	-0.000001	0.000000	0.477117
0.107477	-0.000000	-0.000000	
-0.000001	0.107478	0.000001	
-0.107463	0.107476	-0.000000	
0.107499	-0.000165	0.000000	
-0.107481	0.107473	-0.107475	
0.107476	0.000000	-0.000000	
-0.000000	-0.000000	-0.000000	
-0.000000	0.000000	0.107480	
-0.107483	0.000000	0.000000	
0.000000	-0.000000	-0.000001	
0.000000	-0.107480	0.000000	
-0.000000	0.000000	-0.000000	
-0.000001	-0.000000	-0.000002	
0.000000	0.000000	-0.000000	
-0.107469	0.000000	-0.107482	
	0.000001	0.000000	

- **Αποτίμηση επίδοσης πάνω στα διαθέσιμα δεδομένα**

  Στόχος, λοιπόν, του νευρωνικού δικτύου είναι να προβλέψει τις τιμές του Bitcoin για το τελευταίο δεκαήμερο του Φεβρουαρίου (20/02/2022 – 01/03/2022), χρησιμοποιώντας το ιστορικό των τιμών από 01/01/2021 μέχρι και 19/02/2022. Όπως έχουμε ήδη αναφέρει, τα δεδομένα πριν την εκπαίδευση κανονικοποιούνται για καλύτερη σύγκλιση και μετά την εκπαίδευση οι προβλεπόμενες τιμές αντιστρέφονται στην αρχική κλίμακα. Για να ελέγξουμε την επίδοση του μοντέλου μας υπολογίζεται η μέση τετραγωνική τιμή του σφάλματος της συνάρτησης κόστους (MSE).

  Επιπλέον, η επίδοση της μεθόδου μας συγκρίνεται με την αντίστοιχη μιας ARIMA τεχνικής. Το ARIMA μοντέλο ορίζεται ως, ARIMA(p,d,q) όπου :
  - p = αριθμός αυτοπαλινδρομικών όρων,
  - d = βαθμός διαφοροποίησης,
  - q = αριθμός όρων κινούμενου μέσου όρο.

  Για κάθε συνδυασμό (p,d,q), το μοντέλο εκφράζεται ως:

  $$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t,$$

  όπου ϵ_t αναπαριστά λευκό θόρυβο.

  Η λογαριθμική πιθανοφάνεια $\mathcal{L}$ του εκτιμημένου μοντέλου υπολογίζεται ως:

  $$\mathcal{L} = -\frac{n}{2}\left(\log(2\pi) + \log(\sigma^2) + \frac{1}{n}\sum_{t=1}^{n}\left(\frac{y_t - \widehat{y_t}}{\sigma}\right)^2\right),$$

  όπου n είναι ο αριθμός παρατηρήσεων και σ^2 είναι η διακύμανση των υπολοίπων (σφαλμάτων πρόβλεψης-παρατήρησης).

  Το κριτήριο πληροφορίας Akaike (AIC) υπολογίζεται ως:

  $$AIC = -2\mathcal{L} + 2k,$$

  όπου k είναι ο αριθμός παραμέτρων στο μοντέλο, ίσος με $k = p + q + 2$.

  Ο αλγόριθμος, λοιπόν, δοκιμάζει αύξοντες συνδυασμούς (p,d,q) :
  - Για κάθε συνδυασμό, εκτιμάται το μοντέλο και υπολογίζεται το AIC.
  - Αν το τρέχον AIC είναι μικρότερο από το καλύτερο AIC που έχει βρεθεί μέχρι τώρα, το τρέχον μοντέλο θεωρείται ως το καλύτερο.
  - Ενημερώνεται το καλύτερο AIC, (p,d,q) ως καλύτερου συνδυασμού και αποθηκεύονται οι εκτιμώμενες παραμέτροι του μοντέλου.
  - Αν δεν βρεθεί το καλύτερο AIC έγκαιρα, η επανάληψη σταματά νωρίς.

  Αφού επιλεγεί το καλύτερο μοντέλο, οι προβλέψεις για μελλοντικές τιμές υπολογίζονται ως:

  $$\widehat{y_{t+h}} = \phi_1 y_t + \phi_2 y_{t-1} + \cdots + \phi_p y_{t-p+1} + \theta_1 \widehat{\epsilon_t} + \cdots + \theta_q \hat{\epsilon}_{t-q+1}$$
  $$= \phi_1 y_t + \phi_2 y_{t-1} + \cdots + \phi_p y_{t-p+1}$$ όπου h είναι ο ορίζοντας πρόβλεψης.

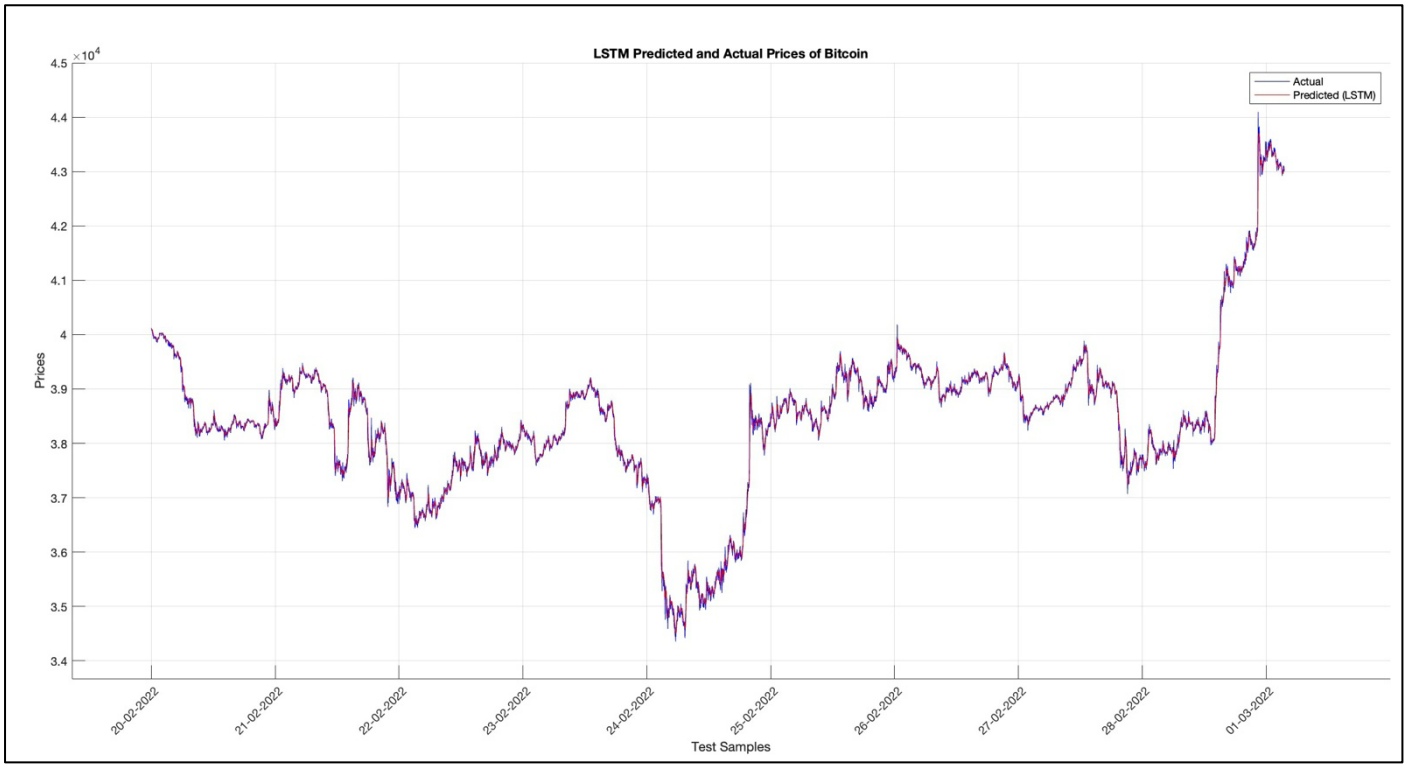
  Τέλος, υπολογίζεται η μέση τετραγωνική τιμή του σφάλματος της συνάρτησης κόστους (MSE).

Τα αποτελέσματα της επίδοσης τόσο του μοντέλου μας, όσο και του μοντέλου ARIMA είναι τα ακόλουθα :

Μοντέλο	Mean squared error (MSE)
LSTM	8683.576172
ARIMA	5129269.126744

Συμπεραίνουμε, λοιπόν, ότι το LSTM μοντέλο που αναπτύξαμε έχει πολύ καλύτερη επίδοση σε σχέση με την ARIMA τεχνική που χρησιμοποιήθηκε για την πρόβλεψη των τιμών του Bitcoin. Συγκεκριμένα, η μέση τετραγωνική τιμή του σφάλματος για το LSTM είναι ίση με 8683.576172, πράγμα που σημαίνει ότι κάνει σχετικά μικρά λάθη στις προβλέψεις του. Αντίθετα, η μέση τετραγωνική τιμή του σφάλματος για το ARIMA είναι 5129269.126744. Οι προβλέψεις του, δηλαδή, απέχουν σε πολύ μεγάλο βαθμό από τις αντίστοιχες πραγματικές τιμές.

Στο παρακάτω διάγραμμα μπορούμε εύκολα να διαπιστώσουμε ότι οι τιμές που προέβλεψε το μοντέλο μας απέχουν σε αρκετά μικρό βαθμό από τις πραγματικές τιμές του Bitcoin για το χρονικό διάστημα που μας ενδιαφέρει.

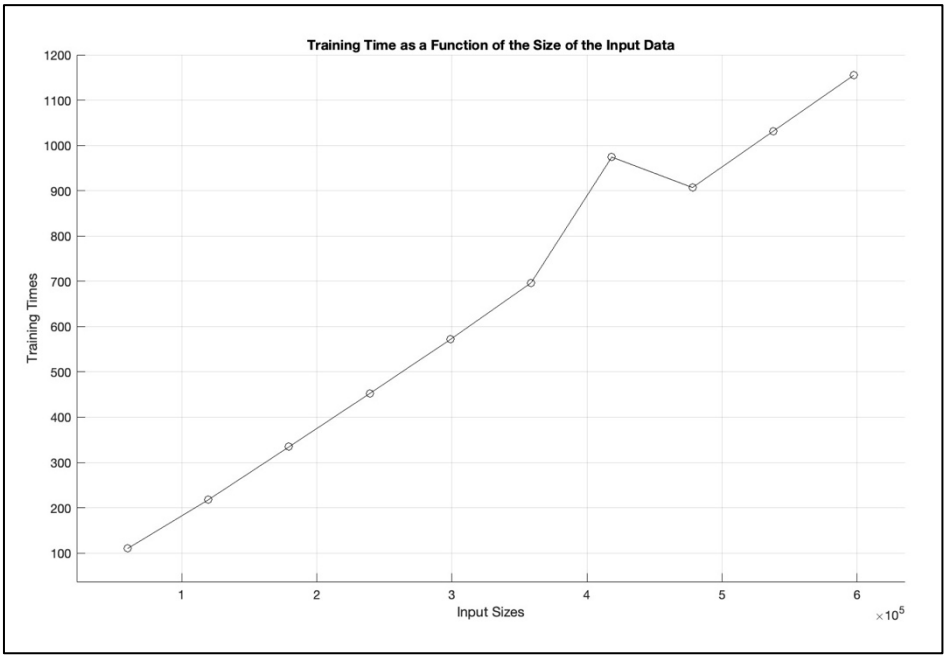


- **Καταγραφή των χρόνων εκπαίδευσης (training time) του δικτύου ως συνάρτηση του μεγέθους των δεδομένων εισόδου**
  Ο κώδικας που αναπτύξαμε, κάνει εκπαίδευση του LSTM νευρωνικού δικτύου χρησιμοποιώντας διαφορετικά ποσοστά των συνολικών δεδομένων που έχουν επιλεγεί για το training. Συγκεκριμένα, ξεκινώντας από ένα ποσοστό του 10% των δεδομένων σε κάθε επανάληψη αυξάνεται σε 20%, 30%, μέχρι και το 100% των δεδομένων. Για να καταγράψει τον χρόνο που χρειάζεται για την εκπαίδευση συναρτήσει διαφορετικών μεγεθών δεδομένων, ο κώδικας χρησιμοποιεί τις συναρτήσεις “tic” και “toc” και αποθηκεύει το αποτέλεσμα στον πίνακα “training_times”. Τα αποτελέσματα που καταγράφηκαν είναι τα ακόλουθα :

Percentage of data	Training Time
10%	110.001277 sec
20%	217.597457 sec
30%	334.538194 sec
40%	452.235520 sec
50%	571.634916 sec
60%	696.579191 sec
70%	974.223670 sec
80%	906.952317 sec
90%	1031.313749 sec
100%	1155.145942 sec

Παρατηρούμε, λοιπόν ότι ο χρόνος για την εκπαίδευση του νευρωνικού δικτύου αυξάνεται με σχετικά σταθερό ρυθμό, σχεδόν γραμμικά, όσο αυξάνεται και το μέγεθος των δεδομένων, με το 100% των δεδομένων να χρειάζεται τον διπλάσιο περίπου χρόνο (1155.145942 sec) από τον αντίστοιχο για το 50% των δεδομένων (571.634916 sec). Ωστόσο, παρατηρούμε μία απότομη αύξηση του χρόνου περνώντας από το 60% των δεδομένων στο 70% (696.579191 sec - 974.223670 sec) και μία μείωση του χρόνου από το 70% στο 80% (974.223670 sec - 906.952317 sec).

Όλες αυτές τις παρατηρήσεις μπορούμε πολύ εύκολα να τις διαπιστώσουμε βλέποντας το διπλανό διάγραμμα που παρουσιάζει την αύξηση του χρόνου εκπαίδευσης συναρτήσει του μεγέθους των δεδομένων εισόδου :



- **Καταγραφή του χρόνου απόκρισης (inference time) του δικτύου**

  Ο χρόνος απόκρισης αναφέρεται στον χρόνο που απαιτείται ώστε να εφαρμοστεί το μοντέλο, μετά την εκπαίδευση του, και να υπολογίσει τις προβλέψεις του για τα δεδομένα που έχουν επιλεγεί για δοκιμή. Όπως και για τον χρόνο εκπαίδευσης (training time), έτσι και εδώ ο κώδικας χρησιμοποιεί τις συναρτήσεις “tic” και “toc” για την καταγραφή του χρόνου απόκρισης (inference time). Ο χρόνος αυτός είναι ίσος με : inference_time = 0.278210 sec. Το μοντέλο LSTM, λοιπόν, είναι σχετικά γρήγορο στην παραγωγή των προβλέψεων του.