Γεωργίου Αλέξιος – Λάζαρος 3180027

Νασσάρ Κυριακίδου Χαλίμα Δήμητρα 3180127

Μέρος 1ο:

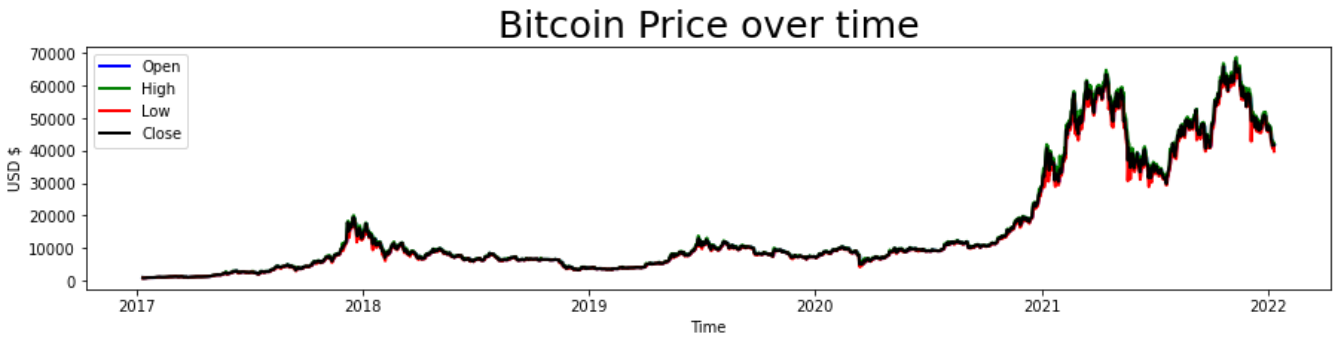
1) Κατεβάσαμε τα δεδομένα του bitcoin, τα τελευταία 5 χρόνια, από *11 Ιανουαρίου 2017 – 11 Ιανουαρίου 2022*. Το αρχείο βρίσκεται στο directory του κώδικα με όνομα «*BTC-USD.csv*» και περιέχει μια γραμμή για κάθε μέρα του χρονικού διαστήματος, συνολικά 1826 μέρες/γραμμές (και η πρώτη γραμμή με τα headers).

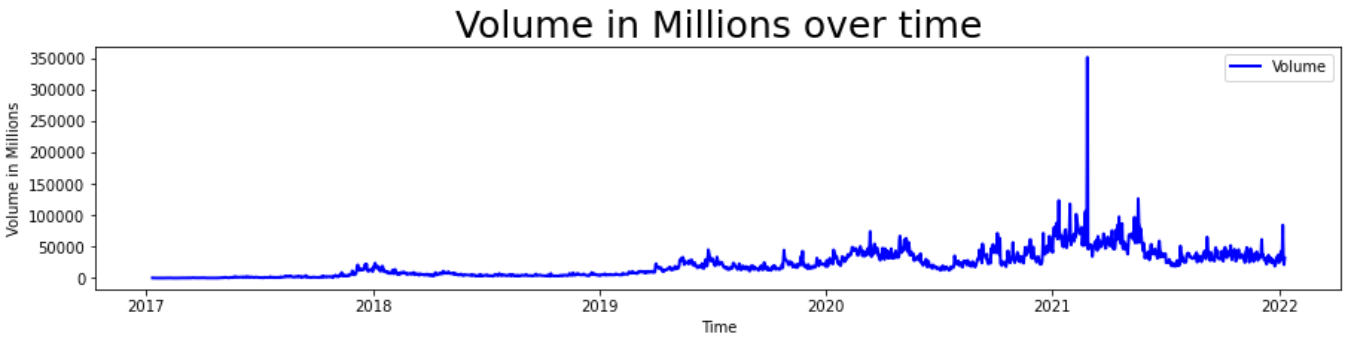
2) Παρατηρούμε ότι η στήλη Close και Adj Close είναι ίδιες, αυτό συμβαίνει γιατί δεν προσαρμόζεται η τιμή κλεισίματος του Bitcoin (πχ δεν μοιράζονται μερίσματα στην μετοχή αυτή). Οπότε η στήλη Adj Close είναι άχρηστη για τα δεδομένα μας.

Δεν υπάρχουν διπλότυπες γραμμές (λογικό αφού έχουμε χρονολογική σειρά) ή κενά κελιά.

Μετατρέπουμε τον τύπο της στήλης `Date` από `Object` σε `datetime64`.

Φτιάχνουμε δύο διαγράμματα για την οπτικοποίηση των δεδομένων, ένα (χρόνου, τιμών) και ένα (χρόνου, όγκου συναλλαγών). Η οπτικοποίηση μπορεί να γίνει πιο λεπτομερής αν «στοχεύσουμε» σε μικρότερη περιοχή χρόνου.





Οι τιμές Volume έχουν διαιρεθεί σε εκατομμύρια για να φαίνεται το διάγραμμα πιο καθαρό.

Χωρίζουμε τα δεδομένα σε train data (80%) και test data (20%). Τα train data είναι παλιότερα, δηλαδή τα πρώτα 4 χρόνια και τα test ο τελευταίος χρόνος.

Κανονικοποιούμε όλες τις στήλες εκτός από την στήλη Close που είναι η στήλη «στόχος» μας (και την στήλη Date που δεν κανονικοποιείται) από το 0 έως τον 1 με τη χρήση MinMaxScaler.

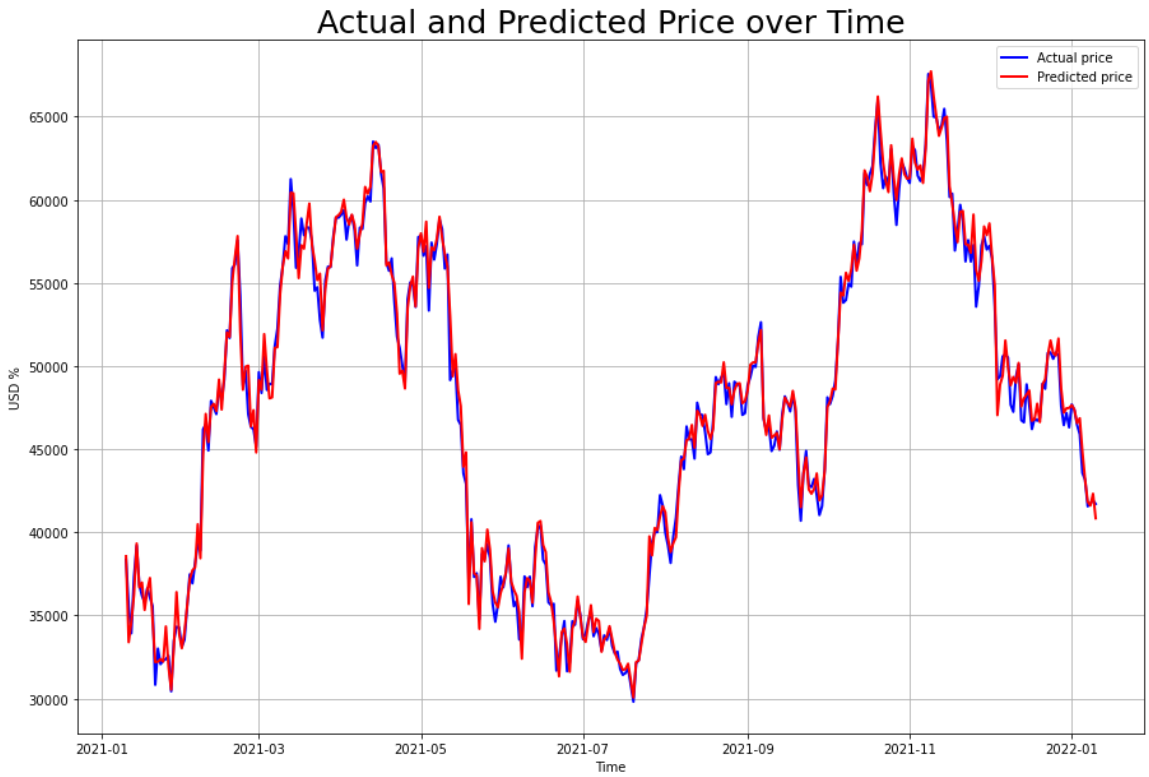
$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Ο MinMaxScaler χρησιμοποιεί τον παραπάνω τύπο για να κανονικοποιήσει κάθε στήλη ξεχωριστά των δεδομένων, παίρνοντας τις μέγιστες και ελάχιστες τιμές κάθε στήλης, τοποθετεί τις ενδιάμεσες τιμές στο (0,1). Στην συνέχεια η κλάση αυτή θυμάται τον συντελεστή για κάθε στήλη και μπορεί να ξεκανονικοποιήσει την πληροφορία. Χρησιμοποιούμαι κανονικοποιήση στα δεδομένα ώστε να επηρεάζουν ισορροπημένα οι τιμές το μοντέλο και όχι με βάση την τάξη μεγέθους τους, εδώ ο όγκος (Volume) έχει άλλη τάξη μεγέθους από τα υπόλοιπα δεδομένα.

*Οι συντελεστές Xmax, Xmin παίρνονται από την κανονικοποιήση των Train data, στην συνέχεια οι ίδιοι συντελεστές εφαρμόζονται στην κανονικοποιήση των Test data, έτσι ώστε να μην επηρεάζονται καθόλου τα Test data από τα Train data. (Αν και είτε λόγω στρογγυλοποίησης είτε λόγω ότι τα μέγιστα/ελάχιστα στα δύο σετ είναι πολύ κοντά, οι συντελεστές είναι σχεδόν ίδιοι).

α) Γραμμική Παλινδρόμηση (Multiple Linear Regression)

Με χωρισμένα τα δεδομένα και την βοήθεια τις sklearn βιβλιοθήκης, με εξαρτημένη μεταβλητή y την στήλη Close και ανεξάρτητες μεταβλητές τις στήλες Open, High, Low, εκπαιδεύουμε το μοντέλο μας.



Cross Validation:

Διαχωρίζουμε τα δεδομένα σε 6 ισόποσα κομμάτια. (~17% των δεδομένων το κάθε κομμάτι)

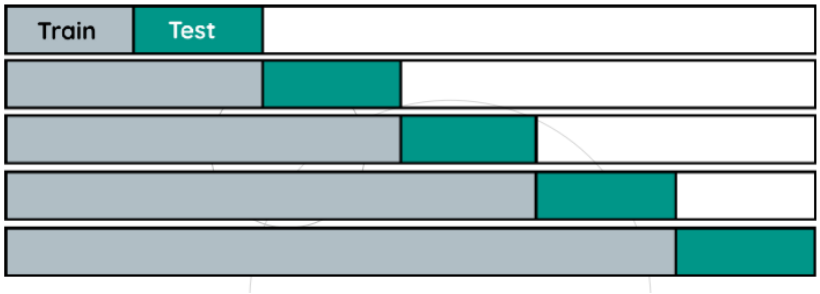
Για κάθε διαχωρισμό (από τον 1^{ο} μέχρι τον 5^{ο})

Εκπαιδεύουμε το μοντέλο μας στα πρώτα i-οστα κομμάτια και τα ελέγχουμε (test) στο επόμενο κομμάτι.

Δηλαδή εκπαιδεύουμε 5 μοντέλα τα οποία έχουν ως train-test data τα εξής:

CV:

```
Split 1:
 TRAIN data start from index 0 to 305
 TEST data start from index 306 to 609
Split 2:
 TRAIN data start from index 0 to 609
 TEST data start from index 610 to 913
Split 3:
 TRAIN data start from index 0 to 913
 TEST data start from index 914 to 1217
Split 4:
 TRAIN data start from index 0 to 1217
 TEST data start from index 1218 to 1521
Split 5:
 TRAIN data start from index 0 to 1521
 TEST data start from index 1522 to 1825
```



Εκτελείται κανονικοποιήση αντίστοιχη με το πρώτο μοντέλο για κάθε διαχωρισμό (πρώτα στα training και μετά στα test με τους ίδιους συντελεστές).

Το μοντέλο της γραμμικής παλινδρόμησης έχει R τετράγωνο score ~99.2%, RMSE: 824 και με CV μέσο R τετράγωνο 99.4%.

Το RMSE με κανονικοποιήμενη μεταβλητή στόχο (‘Close’) είναι ~0.02. Αυτό θα μας χρειαστεί αργότερα για την σύγκριση των μοντέλων.

β) Λογιστική Παλινδρόμηση (Logistic Regression)

Εδώ πρέπει να βρούμε τρόπο το μοντέλο να προβλέπει μια κατηγορηματική τιμή.

Φτιάχνουμε μια καινούργια στήλη η οποία θα έχει τον κινητό μέσο της Close τιμής των τελευταίων 7 ημερών. (SMA7). Στην συνέχεια θα αφαιρέσουμε τις πρώτες μέρες 6 στις οποίες δεν έχουμε το SMA7 από τα δεδομένα (NaN values).

Τώρα δημιουργούμε μια κατηγορηματική στήλη (True/False) με το όνομα Trend η οποία συγκρίνει σε κάθε γραμμή του πίνακα αν η σημερινή τιμή Close είναι μεγαλύτερη από την τιμή SMA7. Αν είναι μεγαλύτερη τότε το Trend είναι True δηλαδή η μετοχή έχει αυξητική τάση. Αυτή η στήλη θα είναι και η στήλη που θα προσπαθήσει να προβλέψει το μοντέλο της λογιστικής παλινδρόμησης.

Αφαιρούμε επίσης και τις υπόλοιπες στήλες που δεν χρειαζόμαστε πλέον (Volume,Close) και επαναφέρουμε το index στον πίνακα με 0.

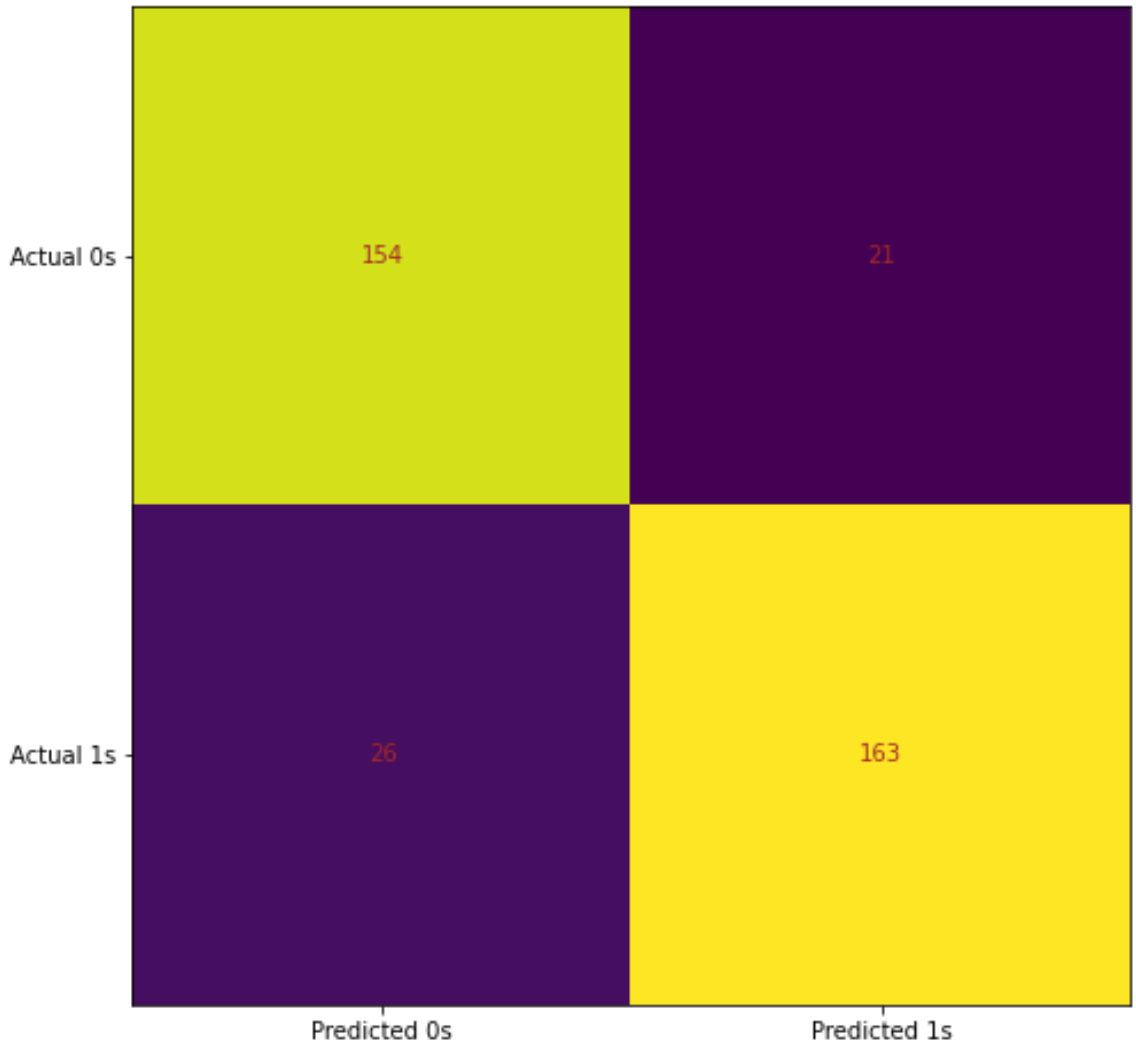
Χωρίζουμε τα δεδομένα με τον ίδιο τρόπο σε 80 training/ 20 testing.

Η κανονικοποιήση με τις βασικές μεθόδους (MinMax/Standard/Robust) δεν βοηθάει στην απόδοση του μοντέλου, οπότε δεν χρησιμοποιούμε (το σκορ πέφτει περίπου 10%).

Δημιουργούμε το μοντέλο της λογιστικής παλινδρόμησης και το εκπαιδεύουμε με είσοδο τις τιμές 'Open','High','Low','SMA7' και έξοδο την λογική τιμή Trend.

Το μοντέλο της λογιστικής παλινδρόμησης έχει R τετράγωνο score ~87.6%, RMSE 0.35 και με CV μέσο R τετράγωνο ~90.8%. Ο διαχωρισμός δεδομένων εκτελείται με τον ίδιο τρόπο όπως και στην γραμμική αν εξαιρέσουμε ότι εδώ έχουμε αφαιρέσει τις πρώτες 6 μέρες δεδομένων.

Το διάγραμμα δείχνει τις True Negative, False Negative, False Positive, True Positive τιμές των προβλέψεων έναντι των πραγματικών τιμών (y_test). Για




```
Precision:  0.8707125603864734
Recall:  0.8712169312169312
F_score:  0.8708001359362609
```

CV:

```
Split 1:
 TRAIN data start from index 0 to 304
 TEST data start from index 305 to 607
 Score r^2: 0.9372937293729373
 Confusion Matrix: [TN,FN] [135  14]
                   [FP,TP] [  5 149]
Split 2:
 TRAIN data start from index 0 to 607
 TEST data start from index 608 to 910
```

```
 Score r^2: 0.8844884488448845
 Confusion Matrix: [TN,FN] [116  26]
                   [FP,TP] [  9 152]
Split 3:
 TRAIN data start from index 0 to 910
 TEST data start from index 911 to 1213
 Score r^2: 0.933993399339934
 Confusion Matrix: [TN,FN] [145   8]
                   [FP,TP] [ 12 138]
Split 4:
 TRAIN data start from index 0 to 1213
 TEST data start from index 1214 to 1516
 Score r^2: 0.8943894389438944
 Confusion Matrix: [TN,FN] [97  7]
                   [FP,TP] [ 25 174]
Split 5:
 TRAIN data start from index 0 to 1516
 TEST data start from index 1517 to 1819
 Score r^2: 0.8943894389438944
 Confusion Matrix: [TN,FN] [137  18]
                   [FP,TP] [ 14 134]
AVERAGE Score r^2: 0.9089108910891088
```

3) Neural Network

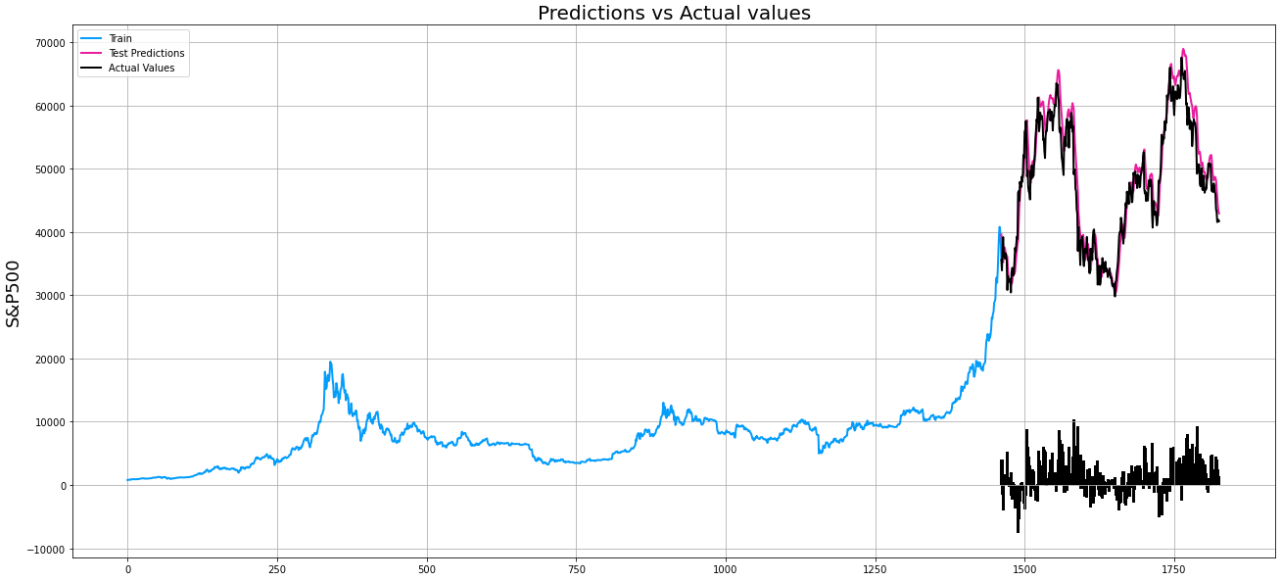
Αρχικά συγκεντρώνουμε τα δεδομένα μας στο dataframe df. Στην συνέχεια χωρίζουμε τα δεδομένα μας σε 80% training data και 20% test data. Έπειτα, προχωράμε στην κανονικοποίηση των train δεδομένων και χρησιμοποιούμε την MinMaxScaler, όπως και παραπάνω.

Θέλουμε το Neural Network να αποτελείται από 4 επίπεδα: το input, 2 κρυφά επίπεδα και το output. Επειδή θέλουμε το input να περιλαμβάνει 50 νευρώνες θα επεξεργαστούμε τα δεδομένα μας ανάλογα. Όπως ζητείται, δημιουργούμε τα δεδομένα εκπαίδευσης με βάση τα οποία θα εκπαιδεύσουμε το Neural Network. Έτσι, δημιουργούμε πολλαπλές πτυχές των δεδομένων εκπαίδευσης, mini-batches που περιέχουν 50 συνεχόμενες τιμές η κάθεμια. Κατά την εκπαίδευση το Neural Network τις επεξεργάζεται μία προς μία και δημιουργεί μια ξεχωριστή πρόβλεψη για κάθε mini-batch. Δηλαδή κατά την διάρκεια της εκπαίδευσης τα βάρη των συνδέσεων μεταξύ των νευρώνων αλλάζουν μέσω των επαναλήψεων ώστε να μειωθούν οι λανθασμένες προβλέψεις. Για τον λόγο αυτό χρησιμοποιούμε την λίστα y_train, ώστε να συγκρίνουμε την πραγματική τιμή με την τιμή της πρόβλεψης του μοντέλου μας να υπολογίσουμε το training error και να το μειώσουμε όσο προχωράει η διαδικασία της εκπαίδευσης.

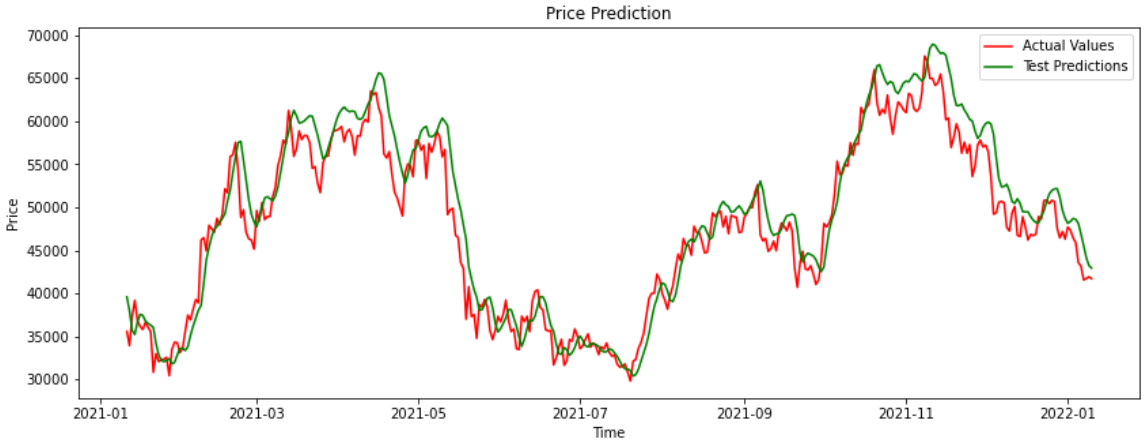
Αναλυτικά, όσον αφορά τα επίπεδα, έχουμε 3 LSTM επίπεδα και 1 Dense επίπεδο. Όπως γνωρίζουμε και σύμφωνα με το tensorflow documentation τα LSTM επίπεδα περιέχουν ήδη activation functions οπότε δεν προσθέτουμε κάποια άλλη. Στο τελευταίο επίπεδο λαμβάνουμε την πρόβλεψη για την επόμενη μέρα. Τέλος, κάνουμε fit το μοντέλο στα training data μας.

Αφού έχει πλέον ολοκληρωθεί η εκπαίδευση του μοντέλου μας, ξεκινάμε την διαδικασία αξιολόγησης του. Έτσι παρέχουμε τα test_data στο μοντέλο μας και καλούμε την predict.

Από τις τιμές MAE = 0.04, RMSE = 0.045637807715879476 και R τετράγωνο = 0.898668859742042, καθώς και από το διάγραμμα μας αντιλαμβανόμαστε ότι οι προβλέψεις μας είναι κοντά στις πραγματικές τιμές. Αναλυτικά, στο πρώτο διάγραμμα μας η ροζ γραμμή είναι οι προβλέψεις, η μαύρη γραμμή είναι οι πραγματικές τιμές και η μπλε γραμμή είναι τα training data. Τέλος, έχουμε προσθέσει την απόλυτη τιμή της διαφοράς των πραγματικών τιμών και των τιμών που προβλέψαμε. Όταν η απόλυτη τιμή είναι αρνητική σημαίνει ότι έχουμε προβλέψει υψηλότερη τιμή από την πραγματική, ενώ όταν η απόλυτη τιμή είναι θετική σημαίνει ότι έχουμε προβλέψει χαμηλότερη από την πραγματική τιμή.

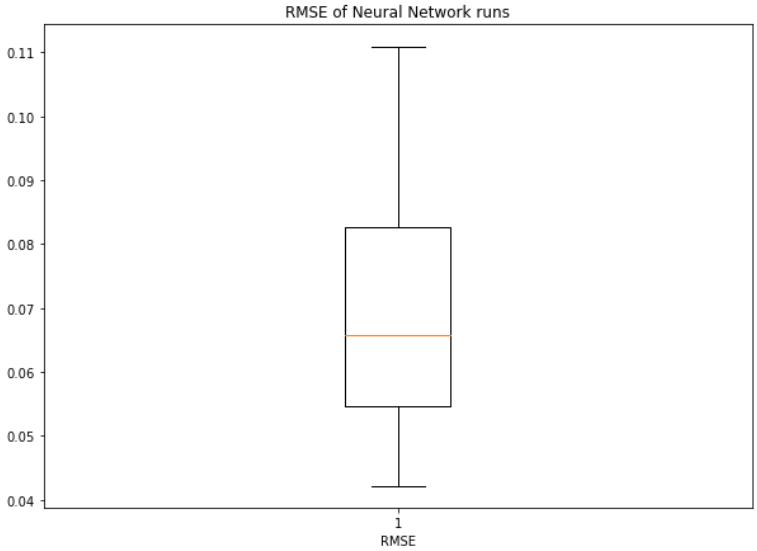


Στο δεύτερο διάγραμμα απεικονίζουμε τις τιμές που προβλέπουμε, καθώς και τις πραγματικές τιμές. Η πράσινη γραμμή αντιστοιχεί στις προβλέψεις και η κόκκινη γραμμή στις πραγματικές τιμές.

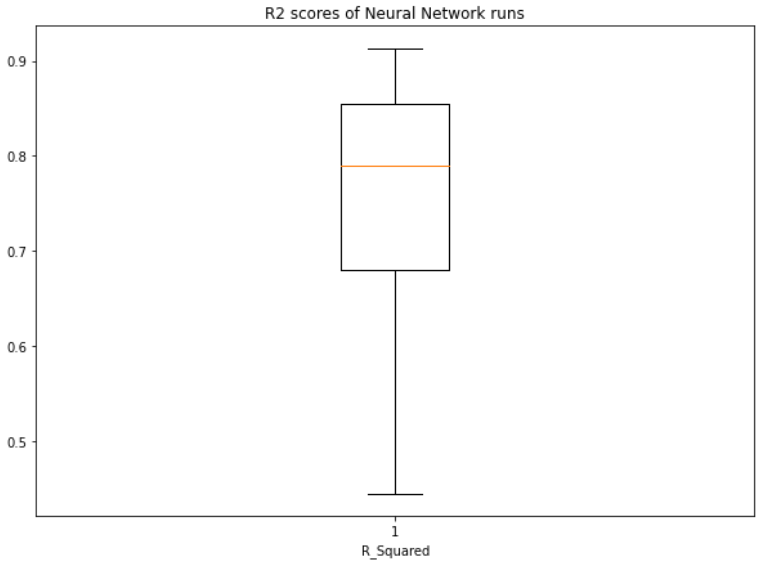


Αξίζει να σημειώσουμε ότι πρόκειται για έναν στοχαστικό αλγόριθμο. Οι στοχαστικοί αλγόριθμοι δεν μαθαίνουν τυχαία, αλλά μαθαίνουν με βάση δεδομένα και παραμέτρους που παρέχουμε. Η διαφορά στις τιμές των MAE, RMSE και R τετράγωνο οφείλεται στις τυχαία διαφορετικές 'αποφάσεις' που λαμβάνει το μοντέλο κατά την εκπαίδευση του κάθε φορά. Στην πραγματικότητα κάθε φορά που ένας στοχαστικός αλγόριθμος μηχανικής μάθησης εκπαιδεύεται στα ίδια δεδομένα μαθαίνει ένα διαφορετικό μοντέλο. Αυτό έχει ως αποτέλεσμα διαφορετικές προβλέψεις, δηλαδή παραγωγή διαφορετικών τιμών κατά την αξιολόγηση του. Παρακάτω φαίνονται δύο boxplot με 26 ζεύγη τιμών των RMSE και R τετράγωνο που παράχθηκαν από τον αλγόριθμο μας.

RMSE:



R τετράγωνο:

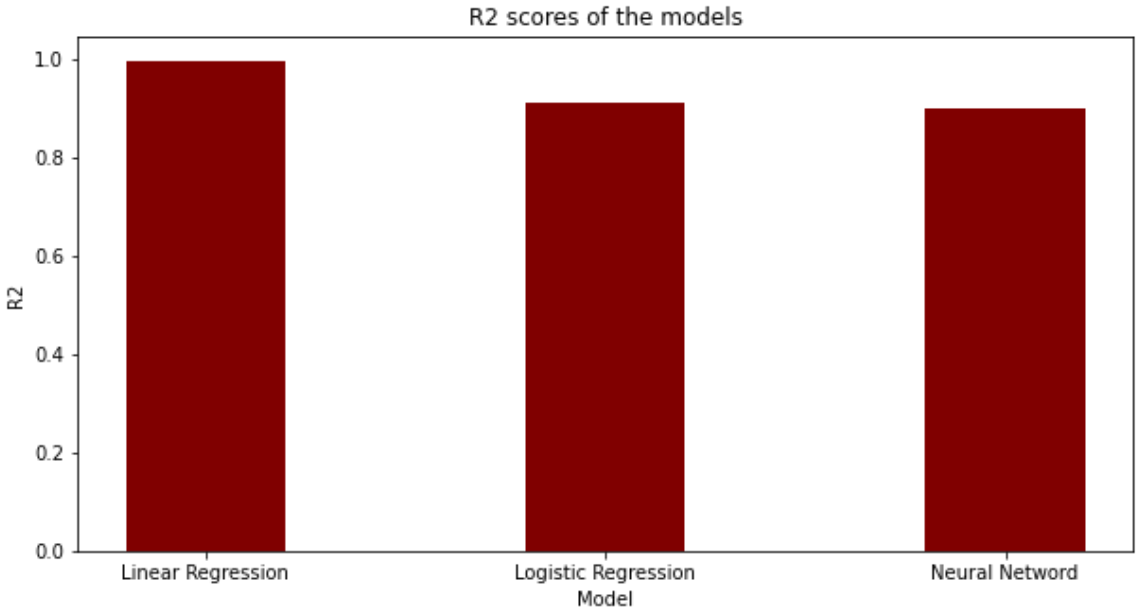


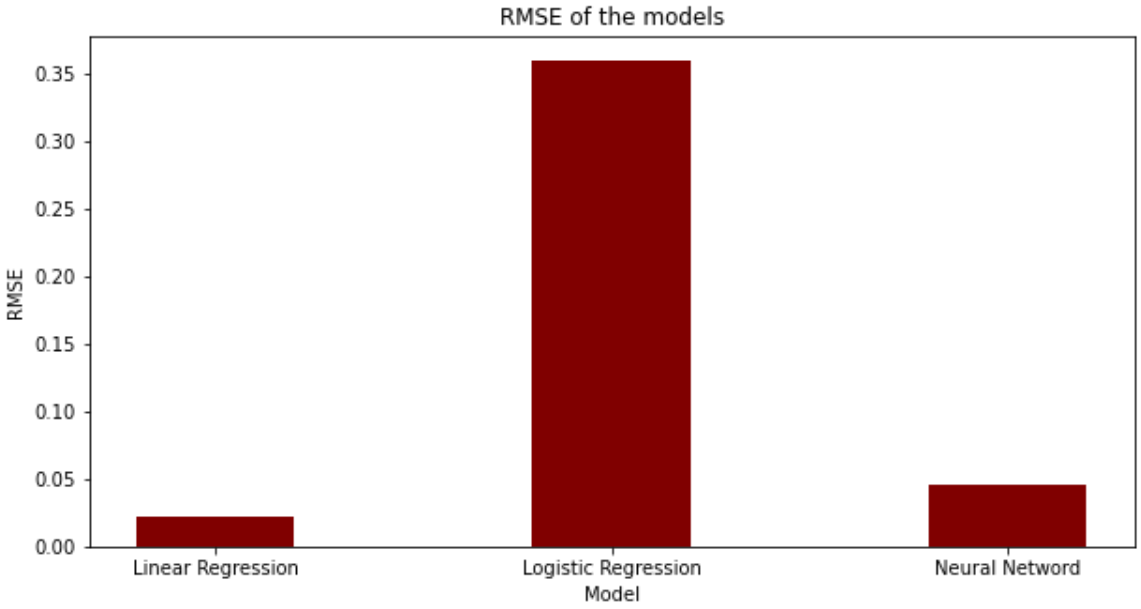
4) Βρήκαμε το r^2 και το RMSE για κάθε μοντέλο, τα οποία φαίνονται στο παρακάτω διάγραμμα.

Στα linear/logistic regression, έχουμε πάρει το r^2 του μέσου όρου των splits των CV το οποίο είναι μια καλή εκτίμηση στην απόδοση των μοντέλων.

Στο Neural Network το score που παίρνουμε είναι αρκετά ακριβής αφού ο αλγόριθμος προσαρμόζεται επαναληπτικά. (Δεν είναι πάντα το ίδιο λόγω του στοχαστικού χαρακτήρα του αλγορίθμου).

Τα RMSE έχουν υπολογιστεί σε κανονικοποιήμενα δεδομένα (target values).





Τα μοντέλα έχουν διαφορετικά inputs και outputs οπότε η σύγκριση ή επιλογή του καθενός θα γίνεται ανάλογα τα δεδομένα που έχουμε σαν input και το ποιο συμπέρασμα θέλουμε να βγάλουμε. Πιο συγκεκριμένα, στο Linear έχουμε input τα Open, High, Low κάθε ημέρας και output το Close της ημέρας. Στο Logistic έχουμε ένα μοντέλο που έχει input τα Open, High, Low, SMA7 και προβλέπει την τάση της τελευταίας εβδομάδας, δηλαδή προβλέπει την κατεύθυνσή της μετοχής κάθε μέρα σε σχέση με το πρόσφατο παρελθόν και όχι την ακριβή τιμή που θα έχει. Τέλος στο Neural Network έχουμε input το ιστορικό της τιμής Close και προσπαθούμε να προβλέψουμε την τιμή κλεισίματος της επόμενης μέρας, το μοντέλο αυτό είναι λογικό να μην έχει τόσο accuracy αφού δεν έχει αρκετά δεδομένα για να είναι ακριβής στην πρόβλεψη του.

Φανταστείτε ότι θέλουμε να προβλέψουμε την επόμενη μέρα της μετοχής, προφανώς δεν μπορούμε να βασιστούμε στην τιμή της High, Low αφού αυτές είναι άγνωστες, οπότε το μοντέλο της Linear είναι μεν αποδοτικό αλλά απαιτεί και πληροφορία που είναι σχεδόν αδύνατο να γνωρίζουμε στο μέλλον. Η logistic είναι ένα μοντέλο πιο γενικό το οποίο προβλέπει την τάση με τις ίδιες απαιτήσεις input. Το NN είναι το μόνο που θα μπορούσαμε να χρησιμοποιήσουμε για ημέρες του αύριο. Τα linear/logistic regression είναι επίσης χρήσιμα για προβλέψεις εντός της ημέρας με μια εκτίμηση του High/Low της ημέρας ή με ένα επαναπροσδιορισμό του μοντέλου με τη χρήση Open (και SMA για logistic).

Γενικά θέλουμε σε ένα μοντέλο μεγάλο R squared (Πόσο καλά το μοντέλο μπορεί να προβλέψει την μεταβλητή στόχο με βάση τα input) και μικρό RMSE η μέση απόκλιση των προβλέψεων.

Το linear έχει καλύτερο R squared και μικρότερο RMSE από το logistic.

Γενικά θέλουμε σε ένα μοντέλο μεγάλο R squared (Πόσο καλά το μοντέλο μπορεί να προβλέψει την μεταβλητή στόχο με βάση τα input) και μικρό RMSE η μέση απόκλιση των προβλέψεων.

Το linear έχει καλύτερο R squared και μικρότερο RMSE από τα υπόλοιπα, οπότε με βάση τα στατιστικά είναι το πιο ακριβής μοντέλο.

Το logistic είναι αρκετά ακριβής (η τιμή RMSE είναι φυσικό να είναι υψηλή αφού το μοντέλο είτε θα πετύχει την τιμή και που στην περίπτωση που δεν την πετύχει η απόκλιση είναι ίση με 1, δηλαδή μεγάλη).

Το NN είναι λιγότερο ακριβής και με μεγαλύτερη απόκλιση από το γραμμικό, αλλά είναι φυσικό αφού έχει μικρότερη πληροφορία σαν είσοδο και προσπαθεί να προβλέψει μια νέα ημέρα. Επιπλέον, όπως αναφέραμε και παραπάνω τα Linear Regression & Logistic Regression είναι ντετερμινιστικά μοντέλα, που σημαίνει ότι με το ίδιο dataset μαθαίνουν κάθε φορά το ίδιο μοντέλο. Αντίθετα, το Neural Network πρόκειται για μη ντετερμινιστικό, δηλαδή στοχαστικό μοντέλο, που σημαίνει ότι η συμπεριφορά του επηρεάζεται σε ένα βαθμό και από τυχαίες ‘αποφάσεις’ που λαμβάνει ο αλγόριθμος κατά την εκπαίδευση και διαφέρουν κάθε φορά που εκπαιδεύεται. Μοντέλα όπως το NN είναι χρήσιμα όταν έχουμε περιορισμένα δείγματα δεδομένων, όπως αυτό στο οποίο βασιζόμαστε μόνο στην τιμή κλεισίματος, αλλά μεγάλα σε πλήθος δεδομένα (π.χ. δεδομένα μιας δεκαετίας εφόσον πρόκειται για time series δεδομένα,

και άλλα παρόμοια προβλήματα όπου το dataset έχει πολλές τιμές δεδομένων και περιορισμένα δείγματα τους).