ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Σχολή Χρηματοοικονομικής και Στατιστικής

Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ

ΣΤΗΝ **ΕΦΑΡΜΟΣΜΕΝΗ ΣΤΑΤΙΣΤΙΚΗ**

ΠΡΟΒΛΕΨΕΙΣ ΧΡΟΝΟΛΟΓΙΚΩΝ ΣΕΙΡΩΝ ΜΕΣΩ ΤΕΧΝΗΤΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

Κωνσταντίνος Χαϊδεμένος

Διπλωματική Εργασία

που υποβλήθηκε στο Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στην *Εφαρμοσμένη Στατιστική*

Πειραιάς

Σεπτέμβριος 2023

Π Α Ν Ε Π Ι Σ Τ Η Μ Ι Ο Π Ε Ι Ρ Α Ι Ω Σ

Σχολή Χρηματοοικονομικής και Στατιστικής

Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ

ΣΤΗΝ **ΕΦΑΡΜΟΣΜΕΝΗ ΣΤΑΤΙΣΤΙΚΗ**

ΠΡΟΒΛΕΨΕΙΣ ΧΡΟΝΟΛΟΓΙΚΩΝ ΣΕΙΡΩΝ ΜΕΣΩ ΤΕΧΝΗΤΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

Κωνσταντίνος Χαϊδεμένος

Διπλωματική Εργασία

που υποβλήθηκε στο Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στην *Εφαρμοσμένη Στατιστική*

Πειραιάς

Σεπτέμβριος 2023

Η παρούσα Διπλωματική Εργασία εγκρίθηκε ομόφωνα από την Τριμελή Εξεταστική Επιτροπή που ορίσθηκε από τη ΓΣΕΣ του Τμήματος Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς στην υπ' αριθμ. ..... συνεδρίασή του σύμφωνα με τον Εσωτερικό Κανονισμό Λειτουργίας του Προγράμματος Μεταπτυχιακών Σπουδών στην Εφαρμοσμένη Στατιστική

Τα μέλη της Επιτροπής ήταν:
- Αναπληρωτής Καθηγητής Μπούτσικας Μιχαήλ (Επιβλέπων)
- Αναπληρωτής Καθηγητής Μπερσίμης Σωτήριος
- Αναπληρωτής Καθηγητής Πελέκης Νικόλαος

Η έγκριση της Διπλωματική Εργασίας από το Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς δεν υποδηλώνει αποδοχή των γνωμών του συγγραφέα.

UNIVERSITY OF PIRAEUS

School of Finance and Statistics

Department of Statistics and Insurance Science

POSTGRADUATE PROGRAM IN

APPLIED STATISTICS

NEURAL NETWORKS FOR TIME-SERIES FORECASTING

By

Konstantinos Chaidemenos

MSc Dissertation

submitted to the Department of Statistics and Insurance Science of the University of Piraeus in partial fulfilment of the requirements for the degree of Master of Science in Applied Statistics

Piraeus, Greece

September 2023

Περίληψη

Η δυνατότητα προβλέψεων με βάση ιστορικά δεδομένα χρονοσειρών είναι πολύ σημαντική και έχει εφαρμογές σε πολλούς τομείς, όπως η οικονομετρία, τα χρηματοοικονομικά, το περιβάλλον, η βιολογία, οι τηλεπικοινωνίες και πολλοί άλλοι. Σκοπός αυτής της εργασίας είναι η λεπτομερής παρουσίαση ενός μεγάλου εύρους μεθόδων πρόβλεψης χρονοσειρών που βασίζονται σε τεχνητά νευρωνικά δίκτυα πρόσθιας τροφοδότησης (feed-forward neural networks). Πιο συγκεκριμένα, θα περιγραφούν και θα υλοποιηθούν αλγόριθμοι ανάπτυξης νευρωνικών δικτύων Συνέλιξης (Convolutional Neural Networks) και Πολυεπίπεδων Αντίληπτρων (Multi Layer Perceptor). Τα δίκτυα αυτά θα εκπαιδευτούν χρησιμοποιώντας ιστορικά δεδομένα από τη χρονοσειρά του QQQ ETF της Invesco, η οποίο στοχεύει στο να αντιγράψει την πορεία του δείκτη Nasdaq-100. Επίσης, θα αξιοποιήσουμε και κάποιες από τις σημαντικότερες μετοχές που συμβάλουν στη διαμόρφωση αυτού του δείκτη. Η απόδοση των προβλέψεων των μοντέλων θα εξεταστεί εμπειρικά, και θα συγκριθεί με παραδοσιακές μεθόδους πρόβλεψης, όπως το υπόδειγμα ARIMA.

Λέξεις – Κλειδία: Νευρωνικά Δίκτυα, MLP, CNN, Χρονοσειρές

Abstract

The ability to make predictions based on historical time series data is highly significant and has applications in various fields such as econometrics, finance, environmental science, biology, telecommunications, and many others. The purpose of this MSc Thesis is to provide a detailed presentation of a wide range of time series prediction methods based on feed-forward artificial neural networks. More specifically, algorithms for developing Convolutional Neural Networks and Multi-Layer Perceptrons will be described and implemented. These networks will be trained using historical data from the QQQ ETF time series by Invesco, which aims to replicate the performance of the Nasdaq-100 index. Additionally, we will leverage some of the most significant stocks that contribute to the formation of this index. The performance of the model predictions will be empirically examined and compared to traditional forecasting methods such as the ARIMA model.

Keywords: Neural Networks, MLP, CNN, Time Series

Περιεχόμενα

Εισαγωγή

Η παρούσα διπλωματική εργασία αποτελείται από τρία κεφάλαια. Στα πρώτα δύο παρουσιάζεται το θεωρητικό υπόβαθρο του αντικειμένου της εργασίας, ενώ το τρίτο κεφάλαιο αφορά την πρακτική εφαρμογή μέσω της χρήσης του λογισμικού της R και του πακέτου Keras και TensorFlow. Συγκεκριμένα, στο πρώτο κεφάλαιο παρουσιάζονται τα νευρωνικά δίκτυα, περιγράφεται η έμπνευσή τους από τα βιολογικά νευρωνικά δίκτυα, αναφέρονται κάποιες από τις βασικότερες συναρτήσεις απώλειας, συναρτήσεις ενεργοποίησης και αλγόριθμοι βελτιστοποίησης. Στο τέλος του πρώτου κεφαλαίου περιγράφεται ο τρόπος λειτουργίας των MLP και CNN, καθώς επίσης και ορισμένες άλλες γνωστές αρχιτεκτονικές νευρωνικών δικτύων. Στο δεύτερο κεφάλαιο αναφέρονται τα βασικά χαρακτηριστικά των χρονοσειρών, όπως η τάση, η εποχικότητα, η κυκλικότητα και η στασιμότητα. Στο τέλος του κεφαλαίου αναφέρονται κάποιες από τις κλασικές μεθόδους πρόβλεψης χρονοσειρών. Στο τρίτο κεφάλαιο γίνεται η παρουσίαση των αποτελεσμάτων από την εφαρμογή των μοντέλων των νευρωνικών δικτύων. Οι προβλέψεις που δημιουργήθηκαν καλύπτουν κυρίως περιπτώσεις προβλέψεων για την μελλοντική τιμή της χρονοσειράς του QQQ, αλλά παρουσιάζονται επιπλέον και μοντέλα κατηγοριοποίησης που προβλέπουν αν θα υπάρξει άνοδος ή κάθοδος στην τιμή του QQQ μετά από τρεις μέρες. Στο τέλος του κεφαλαίου συνοψίζονται συγκεντρωτικά τα αποτελέσματα από τα καλύτερα μοντέλα που προέκυψαν από την εφαρμογή όλων των μεθόδων.

Κ Ε Φ Α Λ Α Ι Ο 1

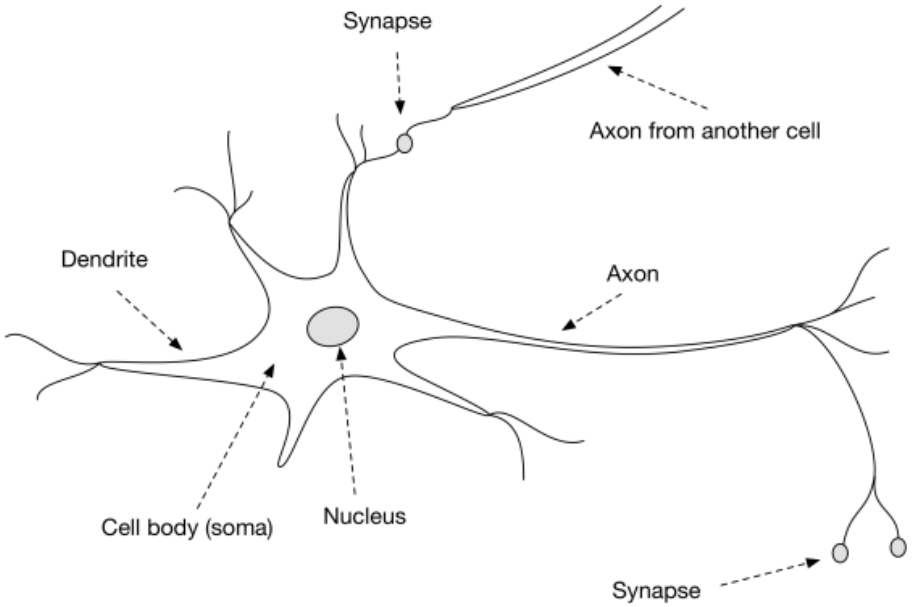
Νευρωνικά Δίκτυα

1.1 Εισαγωγή στα Νευρωνικά Δίκτυα

Τα τεχνητά νευρωνικά δίκτυα (*Artificial Neural Networks (ANN)*), αποτελούν ένα είδος αλγορίθμου της μηχανικής μάθησης εμπνευσμένο από τον τρόπο λειτουργίας του ανθρωπίνου εγκεφάλου. Η ιδέα πως οι λειτουργίες αυτές μπορούν να μελετηθούν και να προσομοιωθούν με τη χρήση μαθηματικών μεθόδων βασισμένων στην Άλγεβρα Μπουλ (*Boolean Algebra*), καθώς και το θεωρητικό πλαίσιο για την ανάπτυξη μιας αρχιτεκτονικής τεχνητού νευρωνικού δικτύου προτάθηκαν για πρώτη φορά το 1943, από το νευροφυσιολόγο Warren McCulloch και τον μαθηματικό Walter Pitts. Χρησιμοποιώντας προτασιακό λογισμό, με την έρευνα τους *"A Logical Calculus of Ideas Immanent in Nervous Activity"*, παρουσίασαν ένα απλοποιημένο μαθηματικό μοντέλο που περιέγραφε το πως οι βιολογικοί νευρώνες στους εγκεφάλους ζώων συνεργάζονται για να εκτελέσουν περίπλοκους υπολογισμούς (Géron, 2019). Πιο συγκεκριμένα, οι βιολογικοί νευρώνες είναι νευρικά κύτταρα και αποτελούν μέρος του νευρικού συστήματος ανθρώπων και ζώων, με τα κυριότερα μέρη τους να είναι το σώμα (*soma*), οι δενδρίτες (*dendrites*), οι άξονες (*axons*) και οι συνάψεις (*synapses*) (βλ. Σχήμα 1.1 - Patterson & Gibson, 2017). Το σώμα είναι το κεντρικό τμήμα του κυττάρου στο οποίο βρίσκεται και ο πυρήνας (*nucleus*) του και είναι υπεύθυνο για την επεξεργασία της πληροφορίας που λαμβάνεται από τους δενδρίτες. Από αυτό επεκτείνεται ένας άξονας και πολλοί δενδρίτες οι οποίοι με τη σειρά τους διακλαδώνονται εκατοντάδες φορές λαμβάνοντας και μεταδίδοντας ηλεκτροχημικά σήματα από γειτονικά νευρικά κύτταρα μέσω των συνάψεων που βρίσκονται στην άκρη τους (Aggarwal, 2018; Geron, 2019; Patterson & Gibson, 2017).

ΣΧΗΜΑ 1.1

Βιολογικός νευρώνας



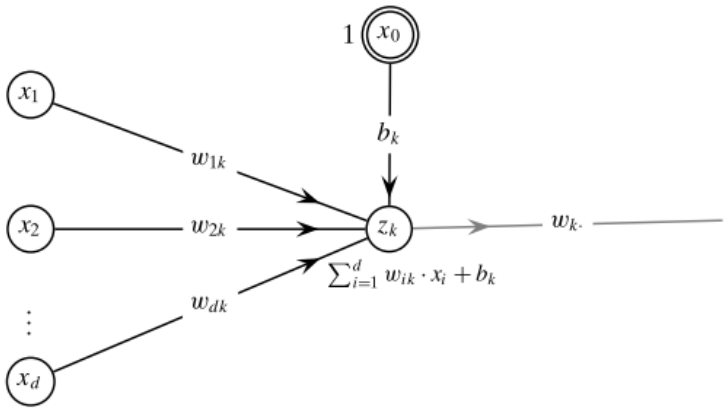
Όταν η συγκέντρωση ιόντων στους δενδρίτες ενός νευρώνα, από τα εισερχόμενα σήματα, ξεπεράσει ένα συγκεκριμένο "κατώφλι" τότε ο νευρώνας διεγείρεται και εκπέμπεται κατά μήκος του άξονα του ένα ηλεκτρικό σήμα, το οποίο φτάνοντας στις συνάψεις του κυττάρου προκαλεί τον ιονισμό των δενδριτών του επόμενου νευρώνα με αποτέλεσμα να συνεχίζεται η μετάδοση της πληροφορίας μεταξύ των διασυνδεδεμένων νευρώνων. Το σήμα που μεταδίδεται πέρα από διεγερτικό μπορεί να είναι και κατασταλτικό κάνοντας έτσι σε κάθε περίπτωση περισσότερο ή λιγότερο πιθανή την ενεργοποίηση του επόμενου νευρώνα. Και στις δύο περιπτώσεις, η ισχύς του σήματος (*action potential*) με τον καιρό προσαρμόζεται ανάλογα με τις ανάγκες του δικτύου. Για παράδειγμα, όταν μια διεργασία επαναλαμβάνεται τακτικά, οι συνάψεις που εμπλέκονται σε αυτή μπορεί να ενισχυθούν καθιστώντας έτσι την επανάληψη της στο μέλλον ευκολότερη (Meira & Zaki, 2020).

Στα τεχνητά νευρωνικά δίκτυα, ως νευρικά κύτταρα θεωρούμε τους κόμβους ή νευρώνες τους οποίους συμβολίζουμε με v_i. Ως συνάψεις επικοινωνίας μεταξύ δύο νευρώνων θεωρούμε τα κατευθυνόμενα τόξα (v_i, v_j) των κόμβων v_i και v_j (Meira & Zaki, 2020). Η ισχύς του σήματος επικοινωνίας των δύο νευρώνων αναπαρίσταται από τα βάρη (*weights*) w_{ij}. Σε αντίθεση με τους βιολογικούς νευρώνες, οι δυνατές τιμές των βαρών μπορεί να είναι και αρνητικές καθώς $w_{ij} \in \mathbb{R}, \forall\, i \neq j$. Ο δείκτης i αναφέρεται στον κόμβο από τον οποίο μεταδίδεται το σήμα ενώ ο δείκτης j αφορά τον κόμβο ο οποίος προσλαμβάνει το σήμα (Haykin, 1998). Βά-

ρη με μεγάλες τιμές σηματοδοτούν μεγαλύτερη συσχέτιση μεταξύ του σήματος εισόδου, με το οποίο σχετίζονται, και του αποτελέσματος του νευρωνικού δικτύου, καθώς συμβάλουν περισσότερο στον τρόπο με τον οποίο το δίκτυο ερμηνεύει τα δεδομένα, συγκριτικά με ένα σήμα εισόδου που έχει ταιριάξει με μικρά βάρη (Patterson & Gibson, 2017). Τα παραπάνω περιγράφουν τη σύνδεση μεταξύ δύο τεχνητών νευρώνων μέσω του σταθμισμένου κατευθυνόμενου γραφήματος (*weighted directed graph*) $G = (V, E)$, όπου $v_i \in V$ και $(v_i, v_j) \in E$. Πιο συγκεκριμένα (βλ. Σχήμα 1.2 - Meira & Zaki, 2020) ένας νευρώνας z_k δέχεται ως είσοδο ένα ή περισσότερα σήματα $x_i, i \in [1, d]$, (που προέρχονται από τους νευρώνες v_i (με τον συμβολισμό x_i συνηθίζεται να εννοούμε τον νευρώνα v_i αλλά και την τιμή του)) τα οποία πολλαπλασιάζονται με τα αντίστοιχα βάρη τους w_{ij}. Επιπλέον, θεωρούμε πως υπάρχει ακόμη ένας ειδικός νευρώνας που ονομάζεται πόλωση ή κατώφλι (*bias, threshold*). Θεωρούμε ότι η τιμή εισόδου του, x_0, είναι πάντα η μονάδα και το βάρος του προς τον νευρώνα z_k συμβολίζεται με b_k

ΣΧΗΜΑ 1.2

Τεχνητός νευρώνας



Έτσι, το σταθμισμένο άθροισμα των σημάτων εισόδου (*net input*) στον νευρώνα z_k ισούται με

$$net_k = b_k + \sum_{i=1}^{d} w_{ik} \cdot x_i = b_k + w^T x$$

Όπου, $w^T = (w_{1k}, w_{2k}, \dots, w_{bk})^T \in \mathbb{R}^d$ τα βάρη των συνάψεων και $x = (x_1, x_2, \dots, x_d)$ οι τιμές εισόδου (Meira & Zaki, 2020).

1.2 Συνάρτηση Ενεργοποίησης

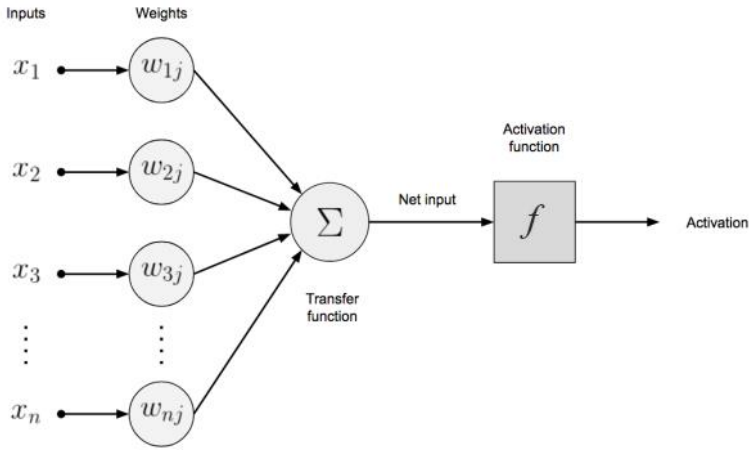
Ένα από τα πλεονεκτήματα των νευρωνικών δικτύων είναι η ικανότητα τους να αναγνωρίζουν μη γραμμικές σχέσεις μεταξύ των πληροφοριών εισόδου (*inputs*) και εξόδου (*outputs*), οι οποίες δεν θα μπορούσαν να αναπαρασταθούν με ακρίβεια με απλές γραμμικές συναρτήσεις. Η ιδιότητα τους αυτή προκύπτει από τη συνάρτηση ενεργοποίησης (*activation function*) που εφαρμόζεται στο σταθμισμένο άθροισμα κάθε τεχνητού νευρώνα. Το αποτέλεσμα αυτής της μετατροπής είναι αυτό που δίνεται τελικά ως πληροφορία εισόδου στον επόμενο νευρώνα (ή νευρώνες) του δικτύου. Όταν η τιμή που μεταβιβάζεται από τον έναν κόμβο στον άλλον είναι μη μηδενική τότε λέμε ότι ο κόμβος (που προωθεί τη πληροφορία) ενεργοποιήθηκε (βλ. Σχήμα 1.3 - Patterson & Gibson, 2017). Για να συμβεί αυτό, το σταθμισμένο του άθροισμα net_k θα πρέπει να υπερβαίνει το κατώτερο κατώφλι, δηλαδή έναν πραγματικό αριθμό, που επίσης καθορίζεται από τη εκάστοτε συνάρτηση ενεργοποίησης. (Lewis, 2017; Patterson & Gibson, 2017; Shanmugamani, 2018; Zaki & Meira, 2020).

Συνεπώς, από τα παραπάνω, η τελική τιμή εξόδου του κόμβου z_k, μετά την εφαρμογή της συνάρτησης ενεργοποίησης $f(\cdot)$, θα είναι

$$z_k = f(net_k)$$

ΣΧΗΜΑ 1.3

Ανάλυση Πληροφορίας Τεχνητού Νευρώνα



Όπως είναι φυσικό, οι νευρώνες διαφοροποιούνται ανάλογα με το είδος της συνάρτησης ενεργοποίησης που εφαρμόζεται. Μερικές από τις πιο βασικές συναρτήσεις ενεργοποίησης δίνονται παρακάτω.

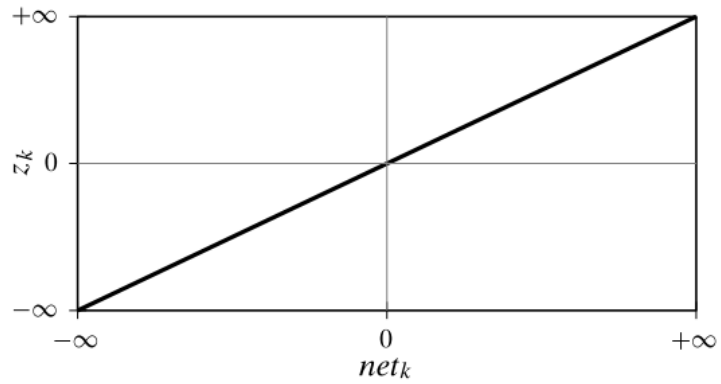
1.2.1 Ταυτοτική (Identity)

Η ταυτοτική, η αλλιώς η γραμμική συνάρτηση, είναι η απλούστερη συνάρτηση ενεργοποίησης και στην ουσία μεταβιβάζει αναλλοίωτο το σήμα που εισέρχεται σε αυτήν:

$$f(net_k) = net_k$$

ΣΧΗΜΑ 1.5

Ταυτοτική Συνάρτηση Ενεργοποίησης



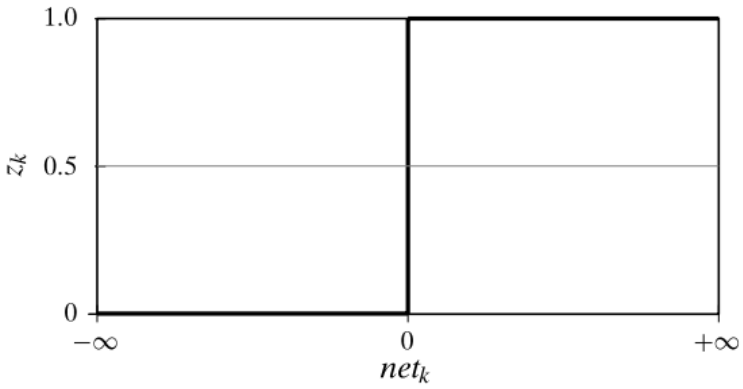
1.2.2 Βηματική (Step)

Η βηματική συνάρτηση είναι μια δυαδική συνάρτηση ενεργοποίησης με σήμα εξόδου 0 αν το σταθμισμένο άθροισμα είναι αρνητικό ή μηδέν, και μονάδα όταν είναι θετικό.

$$f(net_k) = \begin{cases} 0 & if\ net_k \leq 0 \\ 1 & if\ net_k > 0 \end{cases}$$

ΣΧΗΜΑ 1.6

Βηματική Συνάρτηση Ενεργοποίησης



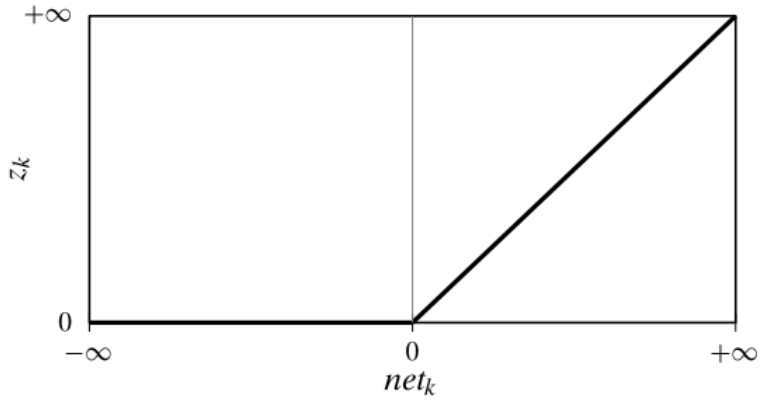
1.2.3 Διορθωμένη Γραμμική Μονάδα (Rectified Linear Unit (ReLU))

Η ReLU έχει αποδειχθεί ότι λειτουργεί αποτελεσματικά σε ένα μεγάλο εύρος προβλημάτων και συγχρόνως δεν είναι υπολογιστικά απαιτητική καθώς ορισμένοι νευρώνες του δικτύου παραμένουν αδρανείς. Πιο συγκεκριμένα, με τον μετασχηματισμό αυτό όταν το σταθμισμένο άθροισμα είναι μικρότερο του μηδενός το σήμα εξόδου είναι μηδέν και ο νευρώνας παραμένει αδρανής ενώ όταν είναι θετικό, ο κόμβος ενεργοποιείται και διαβιβάζει το σήμα χωρίς να το μεταβάλει (Shanmugamani, 2018). Το γεγονός ότι οι αρνητικές τιμές αντιστοιχούν σε μηδενικό αποτέλεσμα, και όταν αυτό συμβαίνει για ένα μεγάλο μέρος των νευρώνων του δικτύου, μπορεί να επηρεάσει αρνητικά την εκπαίδευση και επομένως την αποτελεσματικότητα του δικτύου, καθώς η παράγωγος της ReLU για τους κόμβους αυτούς θα είναι επίσης μηδέν, με αποτέλεσμα να μην ανανεώνονται τα βάρη για τους συγκεκριμένους νευρώνες και έτσι να μη συμβάλουν στη διαδικασία της εκπαίδευσης. Το παραπάνω πρόβλημα είναι γνωστό ως ”*Dying ReLU*”.

$$f(net_k) = \begin{cases} 0 & if\ net_k \leq 0 \\ net_k & if\ net_k > 0 \end{cases}$$

ΣΧΗΜΑ 1.7

Συνάρτηση Ενεργοποίησης ReLU



1.2.4 Διορθωμένη Γραμμική Μονάδα με Διαρροή (Leaky ReLU (LReLU))

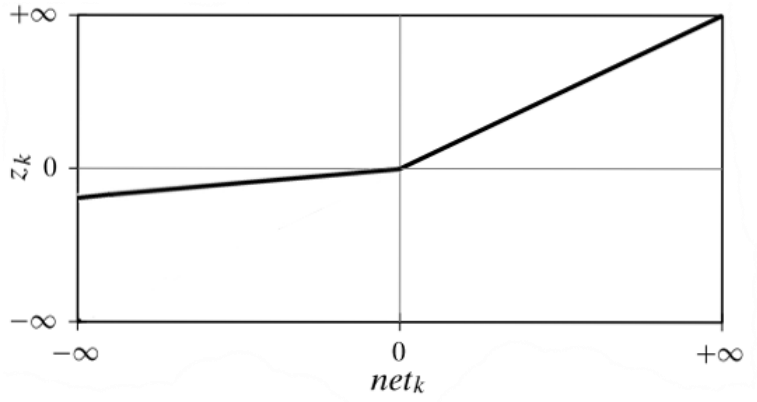
Προκειμένου να μετριαστεί το πρόβλημα της ”*Dying ReLU*” προτάθηκε η χρήση της Leaky ReLU η οποία προσδίδει μια μικρή θετική κλίση a στις αρνητικές τιμές (συνήθως $a = 0.01$), αντί να τις μηδενίζει. Παρ' όλα αυτά, τα αποτελέσματα δεν παρουσιάζουν πάντοτε βελτίωση, σε σχέση με εκείνα που προκύπτουν από τη χρήση της ReLU (Patterson & Gibson,

2017). Επομένως, απαιτούνται δοκιμές και πειραματισμός για να προσδιοριστεί ποια συνάρτηση ενεργοποίησης θα είναι πιο αποδοτική στο εκάστοτε πρόβλημα. Ο τύπος δίνεται παρακάτω:

$$f(net_k) = \begin{cases} a \cdot net_k & if\ net_k \leq 0 \\ net_k & if\ net_k > 0 \end{cases}$$

ΣΧΗΜΑ 1.8

Ταυτοτική Συνάρτηση Ενεργοποίησης LReLU



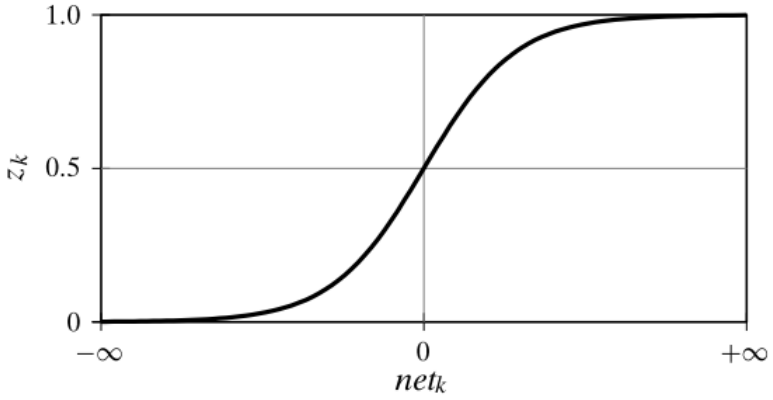
1.2.5 Σιγμοειδής ή Λογιστική (Sigmoid/Logistic)

Η σιγμοειδής είναι μια μη γραμμική συνάρτηση ενεργοποίησης που αντιστοιχίζει τα σήματα εισόδου σε έναν πραγματικό αριθμό μεταξύ του 0 και του 1, επομένως χρησιμοποιείται σε προβλήματα διωνυμικής κατηγοριοποίησης (*binary classification*), όπου η τιμή εξόδου από το νευρωνικό δίκτυο ερμηνεύεται ως η πιθανότητα που ανήκει σε μία από τις δύο κλάσεις. Μικρές αλλαγές, για τιμές εισόδου που βρίσκονται κοντά στο 0, επιφέρουν μεγάλες μεταβολές στις τιμές εξόδου, ενώ για πολύ μεγάλες ή πολύ μικρές τιμές εισόδου, λόγω της κλίσης της, η παράγωγος της πλησιάζει το μηδέν με αποτέλεσμα τα βάρη να ανανεώνονται πολύ αργά κατά τη διαδικασία της μάθησης του νευρωνικού δικτύου. Το φαινόμενο αυτό ονομάζεται το πρόβλημα των εξαφανιζόμενων κλίσεων (*vanishing gradient*) (Feng & Lu, 2019; Lewis, 2017; Shanmugamani, 2018).

$$f(net_k) = \frac{1}{1 + \exp\{-net_k\}}$$

ΣΧΗΜΑ 1.9

Σιγμοειδής Συνάρτηση Ενεργοποίησης



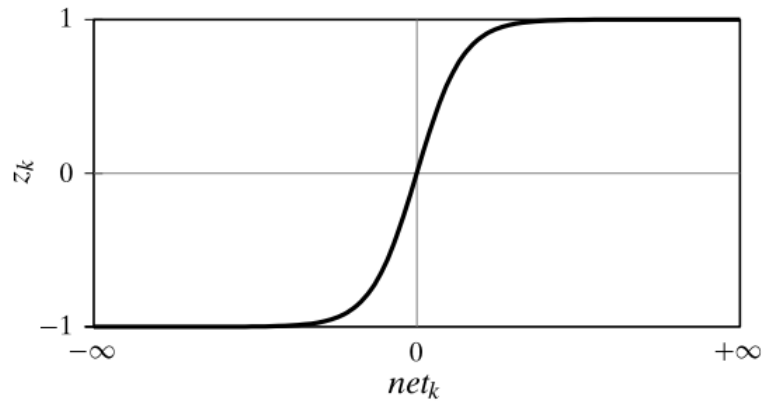
1.2.6 Υπερβολική Εφαπτομένη (tanh)

Η υπερβολική εφαπτομένη είναι η κανονικοποιημένη εκδοχή της σιγμοειδούς αντιστοιχίζει τις τιμές εισόδου στο διάστημα $[-1,1]$ (Shanmugamani, 2018). Επειδή κέντρο της είναι το μηδέν και η κλίση της μεγαλύτερη, συγκριτικά με τη σιγμοειδή, κάνει την διαδικασία της εκπαίδευσης ευκολότερη και με λιγότερο έντονο το πρόβλημα των εξαφανιζόμενων κλίσεων. Επιπλέον, η χρήση της είναι προτιμότερη στη περίπτωση που οι τιμές εξόδου θέλουμε να είναι και θετικές και αρνητικές. Τέλος, όπως και στη σιγμοειδή, επειδή προκαλούν αδιάκοπα την ενεργοποίηση των νευρώνων του δικτύου, είναι υπολογιστικά απαιτητικότερες (Aggarwal, 2018; Shanmugamani, 2018).

$$f(net_k) = \frac{\exp\{2 \cdot net_k\} - 1}{\exp\{2 \cdot net_k\} + 1}$$

ΣΧΗΜΑ 1.10

Υπερβολική Εφαπτομένη Συνάρτηση Ενεργοποίησης



1.2.7 Softmax

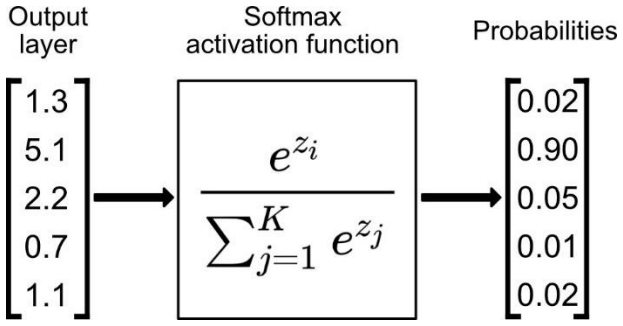
Η softmax αποτελεί γενίκευση της σιγμοειδούς και συνήθως χρησιμοποιείται στην τελευταία στιβάδα του δικτύου, όπου ο στόχος είναι να αντιστοιχίσουμε ένα σήμα εισόδου σε πολλαπλές πιθανές κατηγορίες (*multi-class classification*). Αξίζει να σημειωθεί, πως σε αντίθεση με της της συναρτήσεις ενεργοποίησης που εξαρτώνται μόνο από το σταθμισμένο άθροισμα net_k του εκάστοτε νευρώνα k, η softmax λαμβάνει υπόψιν τα σταθμισμένα αθροίσματα απ' όλους της νευρώνες του στρώματος εξόδου (*output layer*), το πλήθος των οποίων θα είναι όσες και οι πιθανές κατηγορίες της οποίες θέλουμε να κατατάξουμε το σήμα εισόδου (Zaki & Meira, 2020). Οπότε, αν $net_k = \left(net_1, net_2, \dots, net_p\right)^T$ είναι το διάνυσμα που περιέχει τα σταθμισμένα αθροίσματα για όλους της νευρώνες πλήθους p του στρώματος εξόδου, το αποτέλεσμα της συνάρτησης softmax για τον $k \in [1, p]$ νευρώνα θα είναι:

$$f(net_k|net) = \frac{exp\{net_k\}}{\sum_{i=1}^{p} exp\{net_i\}}$$

Στο Σχήμα 1.10 (Radečić, 2020, towardsdatascience.com) μπορούμε να δούμε το τρόπο λειτουργίας της συνάρτησης softmax που περιγράψαμε.

ΣΧΗΜΑ 1.10

Δράση Συνάρτησης Softmax



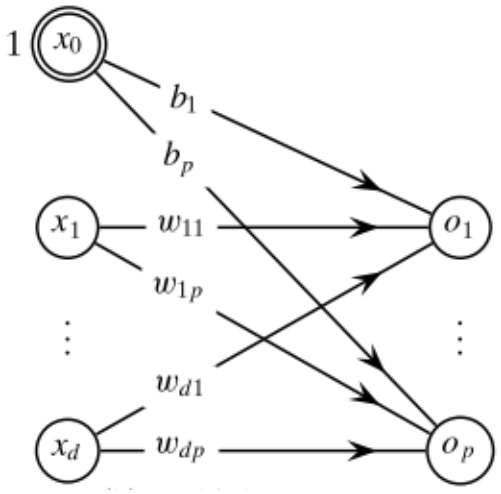
1.3 Συνάρτηση Απώλειας

Σημαντικό ρόλο στον σχεδιασμό και την εκπαίδευση ενός νευρωνικού δικτύου αποτελεί η επιλογή της συνάρτησης απώλειας (*loss function* ή αλλιώς συνάρτηση σφάλματος ή συνάρτηση κόστους (*error functions, cost functions*)) (Goodfelow *et al.,* 2016). Με αυτές μπορούμε και υπολογίζουμε τη διαφορά μεταξύ της προβλεπόμενης, από το νευρωνικό δίκτυο, τιμής εξόδου και της πραγματικής τιμής απόκρισης. Όσο μικρότερη είναι η διαφορά αυτή τόσο μικρότερο είναι και το αποτέλεσμα της συνάρτησης σφάλματος. Σκοπός τους δηλαδή είναι η ποσοτικοποίηση της απόδοσης του δικτύου στην εργασία που του αναθέσαμε (π.χ. παλινδρόμηση, κατηγοριοποίηση) και κατά μία έννοια όλα τα καλά ή κακά χαρακτηριστικά, ακόμη και ενός περίπλοκου νευρωνικού δικτύου, αποτυπώνονται πάνω σε έναν μόνο αριθμό, ο οποίος μας επιτρέπει να κάνουμε συγκρίσεις μεταξύ διαφορετικών μοντέλων και τελικά να συμπεραίνουμε ότι η βελτίωση του αριθμού αυτού (δηλαδή μείωση του) αποτελεί ένδειξη ενός καλύτερου μοντέλου (Patterson & Gibson, 2017; Reed & Marks II, 1999). Έτσι, η εκπαίδευση ενός νευρωνικού δικτύου μετατρέπεται σε ένα πρόβλημα βελτιστοποίησης (*optimization problem*), στο οποίο αναζητάμε τις παραμέτρους εκείνες (βάρη και πολώσεις) που ελαχιστοποιούν τη συνάρτηση απώλειας. Συνήθως το πρόβλημα αυτό δεν λύνεται αναλυτικά και επομένως επιστρατεύουμε επαναληπτικούς αλγορίθμους, όπως η μέθοδος καθόδου κλίσης (*gradient descent*), με τους οποίους οι παράμετροι του δικτύου αναπροσαρμόζονται σταδιακά προκειμένου να επιλύεται με επαρκή επιτυχία το προς εξέταση πρόβλημα (Patterson & Gibson, 2017).

Στη γενική περίπτωση, όπου τα δεδομένα εκπαίδευσης αποτελούνται από n σημεία διάστασης d, δηλαδή $x_i \in \mathbb{R}^d$, $i = 1, \dots, n$ με αντίστοιχες τιμές πρόβλεψης διάστασης p, δηλαδή $o_i \in \mathbb{R}^p$, όπου $o_i = \left(o_{i1}, o_{i2}, \dots, o_{ip}\right)^T, i = 1, \dots, n$ και επιθυμητές τιμές απόκρισης $y_i \in \mathbb{R}^p$, όπου $y_i = \left(y_{i1}, y_{i2}, \dots, y_{ip}\right)^T, i = 1, \dots, n$ (βλ. Σχήμα 1.11 - Zaki & Meira, 2020), κάποιες βασικές συναρτήσεις απώλειας που χρησιμοποιούνται σε προβλήματα παλινδρόμησης ή κατηγοριοποίησης διαμορφώνονται όπως αυτές δίνονται παρακάτω.

ΣΧΗΜΑ 1.11

Νευρωνικό Δίκτυο με Πολλαπλές Εξόδους



1.3.1 Συναρτήσεις Απώλειας για Παλινδρόμηση

- **Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error (MSE))**

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p}\sum_{j=1}^{p}\left(y_{ij} - o_{ij}\right)^2$$

Μία από τις πιο ευρέως διαδεδομένες συναρτήσεις απώλειας και προσφέρει ικανοποιητικά αποτελέσματα τις περισσότερες περιπτώσεις, αλλά λόγω του τετραγωνισμού της διαφοράς $y_{ij} - o_{ij}$ το σφάλμα αυξάνεται γρήγορα για τις ακραίες τιμές (*outliers*) των δεδομένων εκπαίδευσης. Έτσι, επειδή το μοντέλο δίνει μεγαλύτερη σημασία στην ύπαρξη τέτοιων τιμών, στη προσπάθεια του να τροποποιήσει τις παραμέτρους ώστε να μειώσει το σφάλμα που προκαλείται από τις ακραίες τιμές, τελικά η συνολική απόδοση του νευρωνικού δικτύου μειώνεται (Wang *et al.*, 2020).

- **Μέσο Απόλυτο Σφάλμα (Mean Absolute Error (MAE))**

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p}\sum_{j=1}^{p}\left|y_{ij} - o_{ij}\right|$$

Παρουσιάζει ευαισθησία στην ύπαρξη ακραίων τιμών, αλλά το σφάλμα δεν αυξάνεται με την ίδια ταχύτητα όπως στο MSE κάνοντας τη έτσι περισσότερο ανθεκτική.

- **Μέσο Απόλυτο Ποσοστιαίο Σφάλμα (Mean Absolute Percentage Error (MAPE))**

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p}\frac{|y_{ij} - o_{ij}|}{y_{ij}} \cdot 100$$

Το MAPE υπολογίζει το σχετικό σφάλμα των προβλεπόμενων και πραγματικών τιμών, επομένως αυξάνεται μόνο όταν παρατηρούνται μεγάλες ποσοστιαίες διαφορές μεταξύ τους. Η ιδιότητα της αυτή την κάνει πιο ανθεκτική στις ακραίες τιμές, αλλά και γενικότερα στο εύρος των τιμών των μεταβλητών εξόδου (Patterson & Gibson, 2017).

- **Μέσο Τετραγωνικό Λογαριθμικό Σφάλμα (Mean Absolute Log Error (MSLE))**

$$MSLE = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p}\left(logy_{ij} - logo_{ij}\right)^2$$

Όπως και το MAPE, βοηθά στην βελτίωση της απόδοσης του νευρωνικού δικτύου όταν η μεταβλητή πρόβλεψης εμφανίζει πολλές ακραίες τιμές ή κυμαίνεται σε μεγάλο εύρος, καθώς λόγω του λογαρίθμου το εύρος αυτό μειώνεται.

Αξίζει να σημειωθεί πως αν και, όπως ήδη αναφέραμε, το MAPE και MSLE χειρίζονται καλύτερα μεταβλητές μεγάλου εύρους, η χρήση τους και πάλι είναι περιορισμένη καθώς συνηθίζεται να κανονικοποιούμε τα δεδομένα εισόδου πριν εκπαιδεύσουμε ένα μοντέλο νευρωνικού δικτύου.

1.3.2 Συναρτήσεις Απώλειας για Κατηγοριοποίηση

Δεδομένου ότι οι συναρτήσεις απώλειας (είτε για παλινδρόμηση είτε για κατηγοριοποίηση) εξαρτώνται μόνο από τα βάρη (w) και τις πολώσεις (b) του νευρωνικού δικτύου παρακάτω θα χρησιμοποιήσουμε τον συμβολισμό $E(w, b)$ όταν αναφερόμαστε σε αυτές.

- **Απώλεια Hinge (Hinge Loss)**

$$E(w, b) = \frac{1}{n}\sum_{i=1}^{n}\max\left(0{,}1 - y_{ij} \cdot o_{ij}\right)$$

όπου y_{ij} ισούται με -1 ή 1 ανάλογα με την πραγματική κλάση των δεδομένων, ενώ $o_{ij} \in \mathbb{R}$. Χρησιμοποιείται κυρίως για διωνυμική κατηγοριοποίηση (αν και υπάρχει παραλλαγή της για κατηγοριοποίηση πολλών κλάσεων).

- **Διασταυρούμενη Εντροπία (Cross-Entropy)**

$$E(w,b) = \sum_{i=1}^{p} y_i \cdot \ln(o_i) = -\big(y_1 \cdot \ln(o_1) + \cdots + y_p \cdot \ln\big(o_p\big)\big)$$

Χρησιμοποιείται στην κατηγοριοποίηση p κλάσεων $\{c_1, c_2, \dots, c_p\}$, όπου κάθε κλάση έχει υποστεί κωδικοποίηση *one-hot encoding*, δηλαδή έχουν αναπαρασταθεί ως δυαδικά διανύσματα. Πιο συγκεκριμένα κάθε κλάση c_i εκφράζεται ως το διάνυσμα $e_i = \{e_{i1}, e_{i2}, \dots, e_{ip}\}$ με $e_{ii} = 1$ και $e_{ij} = 0$ για κάθε $i \neq j$. Οπότε, τώρα για τιμή εισόδου $x \in \mathbb{R}^d$ έχουμε πως αν η πραγματική απόκριση είναι $y = \big(y_1, y_2, \dots, y_p\big)$ τότε $y \in \big\{e_1, e_2, \dots, e_p\big\}$. Παρατηρούμε πως αν $y = e_i$ μόνο $y_i = 1$ και τα υπόλοιπα στοιχεία που αντιστοιχούν στις τιμές για τις άλλες κλάσεις θα είναι $y_j = 0$ για $j \neq i$.

- **Διωνυμική Διασταυρούμενη Εντροπία (Binary Cross-Entropy)**

$$E(w,b) = -(y \cdot \ln(o) + (1-y) \cdot \ln(1-o))$$

Παραλλαγή της συνάρτησης απώλειας όπου προηγήθηκε, που χρησιμοποιείται σε διωνυμική κατηγοριοποίηση. Αντί για τη μέθοδο *one-hot encoding* αντιστοιχίζομε τη θετική συνήθως κλάση στην μονάδα και την αρνητική στο μηδέν. Οπότε, για τιμή εισόδου $x \in \mathbb{R}^d$ με $y \in \{0,1\}$ υπάρχει μόνο ένας κόμβος εξόδου o.

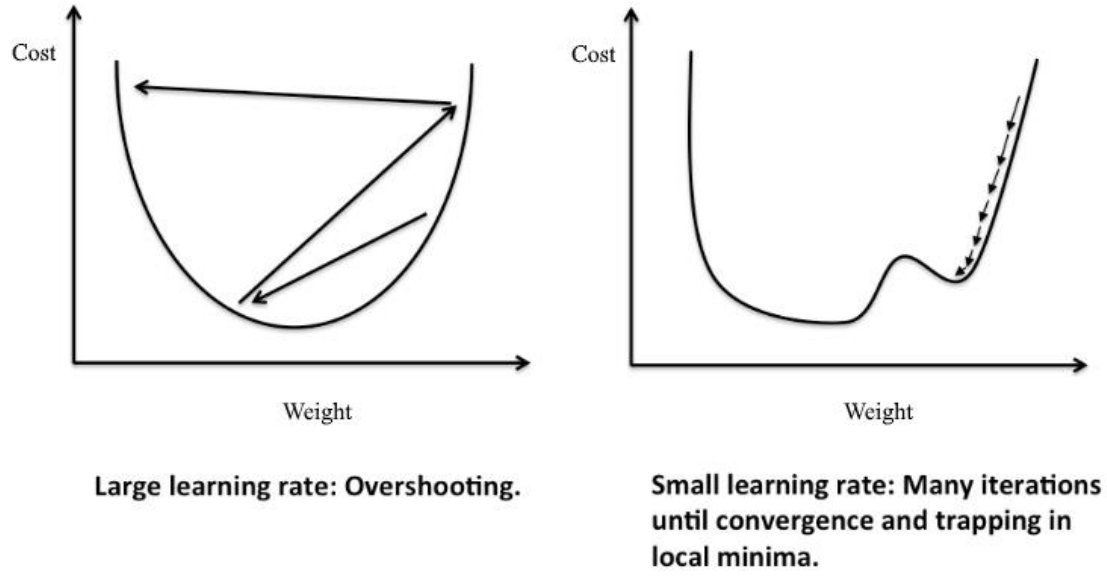
1.4 Αλγόριθμοι Βελτιστοποίησης

Η ελαχιστοποίηση της συνάρτησης απώλειας, προκειμένου να εκπαιδευτεί ένα νευρωνικό δίκτυο, γίνεται μέσω των αλγορίθμων βελτιστοποίησης (*optimization algorithms*). Πιο συγκεκριμένα, κατά την μάθηση του νευρωνικού δικτύου και στο τέλος μιας εποχής του, δηλαδή μετά από ένα πέρασμα όλων των δεδομένων εκπαίδευσης, το αποτέλεσμα της συνάρτησης απώλειας αξιολογείται και ο αλγόριθμος βελτιστοποίησης τροποποιεί τις παραμέτρους του μοντέλου, ώστε το σφάλμα μεταξύ προβλεπόμενων και πραγματικών τιμών να μειωθεί όσο το δυνατόν περισσότερο. Ελέγχουμε τη ταχύτητα με την οποία το μοντέλο θα προσαρμόζεται στο πρόβλημα που του έχουμε αναθέσει και ρυθμίζουμε το κατά πόσο μεγάλη θα είναι η μεταβολή της τιμής των παραμέτρων από εποχή σε εποχή, με έναν συντελεστή που ονομάζεται ρυθμός μάθησης (*learning rate*). Ο ρυθμός μάθησης συνήθως κυμαίνεται σε μια

τιμή ανάμεσα στο 0 και το 1. Θα πρέπει να επιλεγεί προσεκτικά και ίσως απαιτούνται εμπειρικές δοκιμές πριν τον προσδιορισμό του, καθώς με ένα μεγάλο ρυθμό μάθησης οι τιμές των παραμέτρων θα κάνουν μεγάλα "άλματα", και έτσι υπάρχει το ενδεχόμενο να μην πλησιάσουμε ποτέ το ελάχιστο της συνάρτησης απώλειας και να κινούμαστε συνεχώς εκατέρωθεν του, ενώ με ένα πολύ μικρό ρυθμό μάθησης μπορεί να οδηγηθούμε λανθασμένα σε ένα τοπικό ελάχιστο, αντί για το ολικό (ή τουλάχιστον κοντά σε αυτό) και επιπλέον η διαδικασία εκπαίδευσης θα είναι υπολογιστικά απαιτητικότερη (βλ. Σχήμα 1.12 – Raschka, 2015, sebastianraschka.com). Στα προηγούμενα προβλήματα μπορεί να βοηθήσει ένας άλλος συντελεστής που ονομάζεται ορμή (*momentum*), όπου διευκολύνει την εύρεση του ολικού ελαχίστου και βοηθάει τον αλγόριθμο στο να μην "παγιδεύεται" σε τοπικά ελάχιστα (Patterson & Gibson 2017).

ΣΧΗΜΑ 1.12

Προβλήματα Μεγάλου και Μικρού Ρυθμού Μάθησης



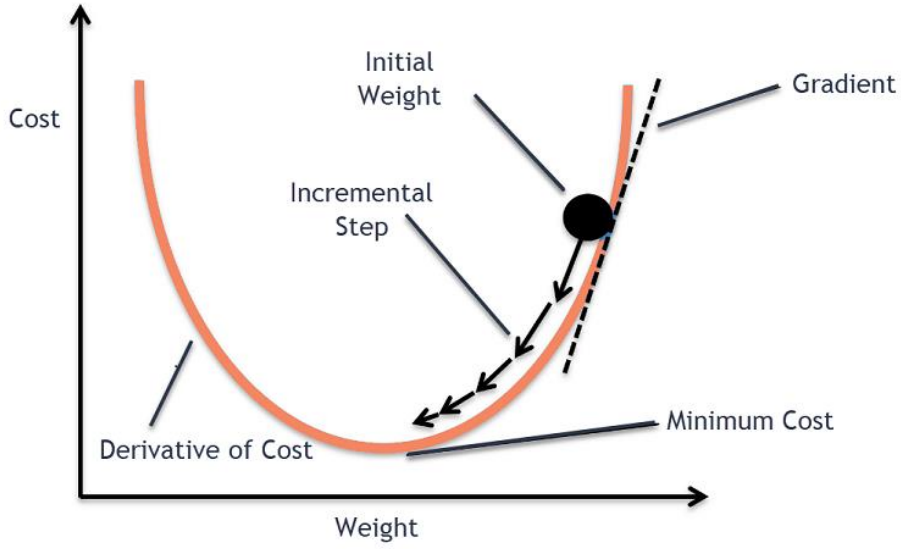
Στη συνέχεια ακολουθούν μερικοί από τους συνηθέστερους αλγορίθμους βελτιστοποίησης.

1.4.1 Μέθοδος Καθόδου Κλίσης

Είναι ίσως ο δημοφιλέστερος αλγόριθμος βελτιστοποίησης και ελαχιστοποιεί της συνάρτηση απώλειας $L(\theta)$ ανανεώνοντας επαναληπτικά τις παραμέτρους $\theta \in \mathbb{R}^d$ προς την αντίθετη κατεύθυνση της κλίσης (*gradient*), $\nabla_\theta L(\theta)$, της συνάρτησης. Στο Σχήμα 1.13 (Haji & Abdulazeez, 2021) φαίνεται το πως ξεκινώντας από ένα τυχαίο σημείο, το οποίο καθορίζεται από τις αρχικές τιμές των παραμέτρων, γίνονται διαδοχικά βήματα προς την αντίθετη κατεύθυνση της κλίσης του σημείου που βρίσκεται κάθε φορά ο αλγόριθμος (αφού αυτή θεωρείται η πιο "απότομη" και επομένως γρήγορη κατεύθυνση κατάβασης), προκειμένου αυτός να προσεγγίσει την ελάχιστη τιμή της συνάρτησης κόστους (Burkov, 2019; Haji & Abdulazeez, 2021). Πέρα από την κλίση, το μέγεθος των βημάτων αυτών, όπως ήδη αναφέραμε, εξαρτάται και από τον ρυθμό μάθησης η.

ΣΧΗΜΑ 1.13

Μέθοδος Καθόδου Κλίσης



Υπάρχουν τρεις παραλλαγές για την μέθοδο καθόδου κλίσης, κάθε μια από τις οποίες διαφοροποιείται στο πόσα δεδομένα χρησιμοποιεί για τον υπολογισμό της κλίσης. Συνήθως, αυτό επιτυγχάνεται με το να γίνεται ένας συμβιβασμός μεταξύ της ακρίβειας με την οποία ανανεώνονται οι παράμετροι και του χρόνου που απαιτείται για να υλοποιηθεί αυτός ο υπολογισμός. (Dogo, *et al.*, 2018; Ruder, 2016). Πιο συγκεκριμένα οι διαφορετικές αυτές εκδοχές του αλγορίθμου είναι οι εξής:

- **Μέθοδος Καθόδου Κλίσης κατά Παρτίδες (Batch Gradient Descent/Vanilla Gradient Descent/Gradient descent (BGD, GD))**

  Η ανανέωση των παραμέτρων δίνεται από τον τύπο:

  $$\theta_{new} = \theta_{old} - \eta \cdot \nabla_{\theta} L(\theta)$$

  Για μόνο μία ανανέωση των τιμών των παραμέτρων απαιτείται το πέρασμα μιας ολόκληρης εποχής, γεγονός που την κάνει υπολογιστικά απαιτητική, αφού όλα τα δεδομένά εκπαίδευσης πρέπει να είναι αποθηκευμένα στην προσωρινή μνήμη του υπολογιστή για να είναι διαθέσιμα στον αλγόριθμο, και έτσι δεν επιτρέπει την ανανέωση του μοντέλου στη περίπτωση που νέα δεδομένα προστίθενται συνεχώς στο σύνολο δεδομένων μας (*online model update*) . Ο υπολογισμός είναι αποτελεσματικός και προσφέρει σταθερή σύγκλιση, αλλά σε μη κυρτές συναρτήσεις η σύγκλιση μπορεί να τερματιστεί σε κάποιο τοπικό ελάχιστο και να μην καταφέρει να πετύχει το βέλτιστο αποτέλεσμα που μπορεί να προσφέρει το μοντέλο (Haji & Abdulazeez, 2021; Ruder, 2016).

- **Μέθοδος Στοχαστικής Καθόδου Κλίσης (Stochastic Gradient Descent (SGD))**

  Για την ανανέωση των παραμέτρων χρησιμοποιείται ένα δείγμα $x^{(i)}$ των δεδομένων εκπαίδευσης με αντίστοιχες τιμές στόχους $y^{(i)}$:

  $$\theta_{new} = \theta_{old} - \eta \cdot \nabla_{\theta} L\left(\theta; x^{(i)}; y^{(i)}\right)$$

  Με τη μέθοδο αυτή η κλίση υπολογίζεται κάθε φορά προσεγγιστικά για μία μόνο τυχαία τιμή (για τον λόγο αυτό άλλωστε ονομάζεται και στοχαστική). Έτσι, αφού δεν χρησιμοποιούμε όλα τα δεδομένα εκπαίδευσης υπάρχει μεγάλη διακύμανση στις ανανεώσεις των παραμέτρων, και κατ' επέκταση στο αποτέλεσμα της συνάρτησης απώλειας. Αυτό σε κάποιες περιπτώσεις μπορεί να της επιτρέψει να "μεταπηδήσει" ένα υποδεέστερο τοπικό ελάχιστο και να καταφέρει να συγκλίνει σε ένα πιθανώς καλύτερο ή ακόμη και στο ολικό μέγιστο. Παρότι το υπολογιστικό κόστος γενικά μειώνεται συγκριτικά με τον BGD, ο αριθμός των επαναλήψεων που απαιτείται για να συγκλίνει, ο αλγόριθμος είναι μεγαλύτερος, λόγω της αστάθειας που τον χαρακτηρίζει (Haji & Abdulazeez, 2021; Ruder, 2016).

- **Μέθοδος Καθόδου Κλίσης με Μικρού Μεγέθους Παρτίδες (Mini-Batch Gradient Descent)**

  Η μέθοδος αυτή αποτελεί συνδυασμό των δύο προηγούμενων καθώς διαιρεί το σύνολο δεδομένων εκπαίδευσης σε n μικρότερα δείγματα για καθένα από τα οποία εκτελείται η ανανέωση των παραμέτρων:

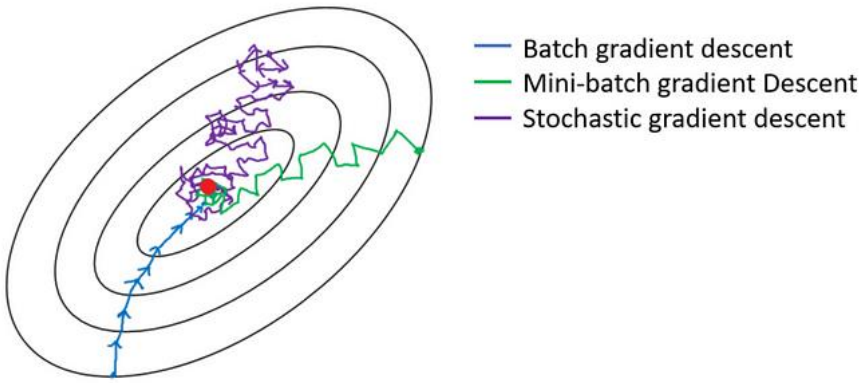
$$\theta_{new} = \theta_{old} - \eta \cdot \nabla_{\theta} L\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right),$$

  προσφέροντας έτσι μια ενδιάμεση λύση ανάμεσα στην αποδοτικότητα του GD και στην ανθεκτικότητα (*robustness*) του SGD (Haji & Abdulazeez, 2021; Ruder, 2016). Για παράδειγμα εάν το σύνολο των δεδομένων εκπαίδευσης αποτελείται από 2000 παρατηρήσεις και το μέγεθος παρτίδας (*batch*) ορίστηκε να είναι 100, τότε όλες οι παρατηρήσεις θα έχουν χωριστεί σε $n = 20$ μικρότερα δείγματα και έτσι θα γίνουν 20 ενημερώσεις των παραμέτρων σε κάθε εποχή. Δηλαδή, το πλήθος των εποχών και το πλήθος των παρτίδων λειτουργούν ως εμφωλευμένοι *for-βρόχοι* (*nested for-loops*), για την ανανέωση των παραμέτρων (με εξωτερικό βρόγχο την εποχή και εσωτερικό την παρτίδα). Οπότε, στο προηγούμενο παράδειγμα, αν επιπλέον ο συνολικός αριθμός των εποχών ήταν 40, τελικά θα γίνονταν $40 \cdot 20 = 800$ ανανεώσεις των παραμέτρων.

Στο Σχήμα 1.14 (Dabbura, 2017, towardsdatascience.com) παραθέτουμε ένα παράδειγμα, σχετικά με τη συμπεριφορά της τροχιάς σύγκλισης προς το (τοπικό) ελάχιστο, για τις τρεις διαφορετικές εκδοχές του αλγορίθμου GD.

ΣΧΗΜΑ 1.14

Τροχιές Αλγορίθμων Καθόδου Κλίσης



1.4.2 Μέθοδος Καθόδου Κλίσης με Προσαρμογή (Adaptive Gradient Descent (AdaGrad))

Σε αντίθεση με τους προηγούμενους αλγορίθμους βελτιστοποίησης όπου ο ρυθμός μάθησης παρέμενε ο ίδιος για όλες τις παραμέτρους θ_{new}, ο αλγόριθμος AdaGrad προσαρμόζει τον συντελεστή η για κάθε παράμετρο θ_i. Παράμετροι που αντιστοιχούν σε μεγάλες κλίσεις έχουν μειωμένο ρυθμό μάθησης ενώ παράμετροι με μικρές κλίσεις έχουν αυξημένο ρυθμό μάθησης. (Haji & Abdulazeez, 2021; Ruder, 2016). Το γεγονός αυτό τον κάνει περισσότερο αποτελεσματικό σε "αραιά" δεδομένα (*sparse data*), καθώς εκτελεί μικρότερες ανανεώσεις σε συχνές παραμέτρους και μεγαλύτερες σε παραμέτρους που είναι σπανιότερες (Ruder, 2016). Ο κανόνας ανανέωσης δίνεται από τον τύπο:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \cdot g_{t,i}$$

Όπου,

- $g_{t,i} = \nabla_\theta J(\theta_{t,i})$ η κλίση της παραμέτρου θ_i τη χρονική στιγμή t
- $G_t \in \mathbb{R}^{dxd}$ διαγώνιος πίνακας όπου κάθε στοιχείο της διαγωνίου i, i είναι το άθροισμα των τετραγώνων των κλίσεων θ_i μέχρι τη χρονική στιγμή t
- ε παράγοντας εξομάλυνσης ώστε να αποτρέψουμε τη διαίρεση με το 0, συνήθως παίρνει τιμές της τάξεως του 10^{-8}.

  Έτσι λόγω της προσαρμοστικότητας του, συνήθως, δεν απαιτούνται διαδοχικές εμπειρικές δοκιμές του ρυθμού μάθησης μέχρι να πετύχουμε ένα ικανοποιητικό αποτέλεσμα. Από την άλλη, κύριο μειονέκτημα του είναι πως ο όρος G_t, που βρίσκεται στον παρονομαστή, συνεχώς αυξάνεται (αφού για κάθε νέα χρονική στιγμή κάθε επιπλέον όρος που αθροίζεται είναι θετικός), με αποτέλεσμα ο ρυθμός μάθησης να μειώνεται συνεχώς κατά τη διαδικασία της εκπαίδευσης και ο αλγόριθμος να μη μπορεί να αντλήσει επιπλέον πληροφορία από τα δεδομένα (Ruder, 2016).

1.4.3 Root Mean Square Propagation (RMSProp)

Ο αλγόριθμος RMSProp προτάθηκε για να επιλύσει το πρόβλημα της αδιάκοπης μείωσης του ρυθμού μάθησης του AdaGrad. Πιο συγκεκριμένα, αντί να αθροίζονται στον G_t όλες οι προηγούμενες τετραγωνικές κλίσεις g^2 (όπως συμβαίνει στον AdaGrad), το άθροισμα των τετραγωνικών κλίσεων ορίζεται αναδρομικά ως ο κινητός μέσος $E[g^2]_t$, που δίνεται από την παρακάτω σχέση:

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$

Οπότε ο $E[g^2]_t$ τη χρονική στιγμή t, εξαρτάται μόνο από τον προηγούμενο κινητό μέσο $E[g^2]_{t-1}$ και την τρέχουσα κλίση g_t. .Αυτό έχει ως αποτέλεσμα να μειώνεται σταδιακά η επίδραση των παλαιότερων κλίσεων, σε σύγκριση με την πιο πρόσφατη κλίση g_t. (Haji & Abdulazeez, 2021; Magaia *et al.*, 2021; Ruder, 2016). Έτσι, ο κανόνας ανανέωσης δίνεται από τον παρακάτω τύπο:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} \cdot g_t$$

1.4.4 Adaptive Moment Estimation (Adam)

Ο Adam είναι ένας ακόμη αλγόριθμος που υπολογίζει και προσαρμόζει το ρυθμό μάθησης για κάθε παράμετρο ξεχωριστά. Όπως και ο RMSProp, αξιοποιεί το κινητό μέσο των προηγουμένων τετραγωνικών κλίσεων υ_t, αλλά επιπλέον αποθηκεύει και τον κινητό μέσο των προηγουμένων (μη-τετραγωνικών) κλίσεων m_t. Έχει χαμηλό υπολογιστικό κόστος, δεσμεύει λιγότερο μέρος της προσωρινής μνήμης του υπολογιστή και η σύγκλισή του είναι γρήγορη. Ο κανόνας ανανέωσης δίνεται από τα παρακάτω:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ \upsilon_t &= \beta_2 \upsilon_{t-1} + (1-\beta_2) g_t^2 \end{aligned}, \qquad \begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{\upsilon}_t &= \frac{\upsilon_t}{1-\beta_2^t} \end{aligned},$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{\upsilon}_t + \varepsilon}} \cdot \hat{m}_t$$

Όπου, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\hat{m}_t$ και $\hat{\upsilon}_t$ οι εκτιμήσεις της πρώτης (μέση τιμή) και της δεύτερης ροπής (μη κεντροποιημένη διακύμανση), όπου ο ρόλος τους είναι παρόμοιος με αυτόν του συντελεστή ροπής, και $\hat{m}_t$, $\hat{\upsilon}_t$ είναι οι διορθωμένες ως προς την μεροληψία (*bias-corrected*) εκτιμήσεις των m_t και υ_t, καθώς χωρίς τη διόρθωση αυτή παρατηρήθηκε πως ιδιαίτερα στα πρώτα στάδια του αλγορίθμου τείνουν να είναι πολύ κοντά στο μηδέν (Magaia, 2021; Ruder, 2016).

Ένας προβληματισμός που μπορεί να δημιουργηθεί είναι το κατά πόσο είναι έγκυρη η χρήση συναρτήσεων ενεργοποίησης — που δεν είναι διαφορίσιμες σε όλο το $\mathbb{R}$ — σε αλγορίθμους βελτιστοποίησης, που βασίζονται στη μέθοδο της καθόδου κλίσης. Για παράδειγμα η

ReLU δεν είναι διαφορίσιμη στο μηδέν καθώς για $net_k = 0$ το αριστερό σκέλος της συνάρτησης δίνει παράγωγο 0 ενώ το δεξί 1. Στη πράξη όμως, για την εκπαίδευση ενός μοντέλου νευρωνικού δικτύου και για την παραγωγή καλύτερων αποτελεσμάτων από αυτό, αρκεί να μειωθεί σημαντικά η συνάρτηση απώλειας, χωρίς αυτό να σημαίνει απαραίτητα πως ο αλγόριθμος βελτιστοποίηση έφτασε ακριβώς πάνω στο ολικό ελάχιστο (κάτι το οποίο δεν συμβαίνει συνήθως ούτως η άλλως). Επομένως, αφού δεν περιμένουμε από τον αλγόριθμο να φτάσει το σημείο όπου η κλίση είναι 0, είναι αποδεκτό το ελάχιστο της συνάρτησης ενεργοποίησης να αντιστοιχεί σε σημεία που δεν ορίζεται η παράγωγος της. Επιπλέον, όταν εκπαιδεύεται ένα νευρωνικό δίκτυο σε κάποιο πρόγραμμα, συνήθως χρησιμοποιείται η τιμή της παραγώγου σύμφωνα με ένα από τα δύο σκέλη της συνάρτησης, αντί να μας επιστρέφεται σφάλμα ότι η παράγωγος εκεί δεν ορίζεται (Goodfelow *et al.*, 2016).

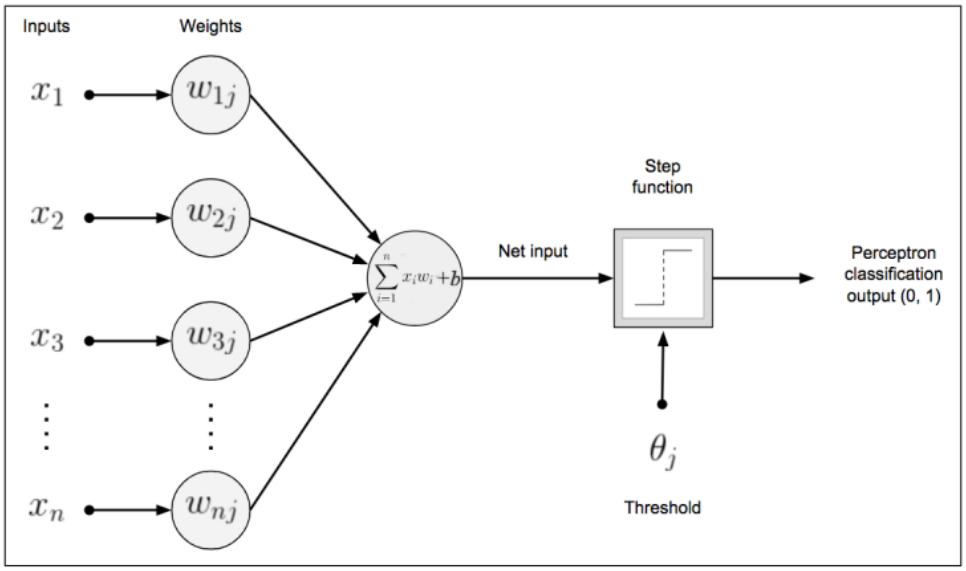
1.5 Κατηγορίες Νευρωνικών Δικτύων

1.5.1 Αντίληπτρο Ενός Στρώματος (Single Layer Perceptron (SLP))

Το SLP είναι η απλούστερη μορφή νευρωνικού δικτύου και είναι ένα γραμμικό μοντέλο που χρησιμοποιείται για διωνυμική κατηγοριοποίηση. Αποτελείται από n στο πλήθος εισόδους για τις οποίες υπολογίζεται το σταθμισμένο άθροισμα τους και στην συνέχεια αυτό αποστέλλεται στη βηματική συνάρτηση ενεργοποίησης με τιμή κατωφλιού συνήθως το 0.5. Η έξοδος του SLP είναι μία δυαδική τιμή που υποδεικνύει τη κλάση ή τη κατηγορία που ανήκουν τα δεδομένα εισόδου, δίνει αποτέλεσμα 1 όταν το σταθμισμένο άθροισμα είναι μεγαλύτερο από την τιμή του κατωφλιού και 0 στην αντίθετη περίπτωση. Στο Σχήμα 1.15 (Patterson & Gibson, 2017) αναπαρίσταται η σχέση τιμών εισόδου και εξόδου. Ο όρος πόλωσης b μετακινεί το όριο απόφασης (*decision boundary*), καθώς, αν κατά τη διαδικασία της εκπαίδευσης, η τιμή του γίνει αρνητική εξαναγκάζει τα βάρη να πρέπει να λάβουν μεγαλύτερες τιμές για να καταφέρουν να υπερβούν την τιμή του κατωφλιού και να δώσουν έξοδο την κλάση που αντιστοιχεί στη μονάδα.

ΣΧΗΜΑ 1.15

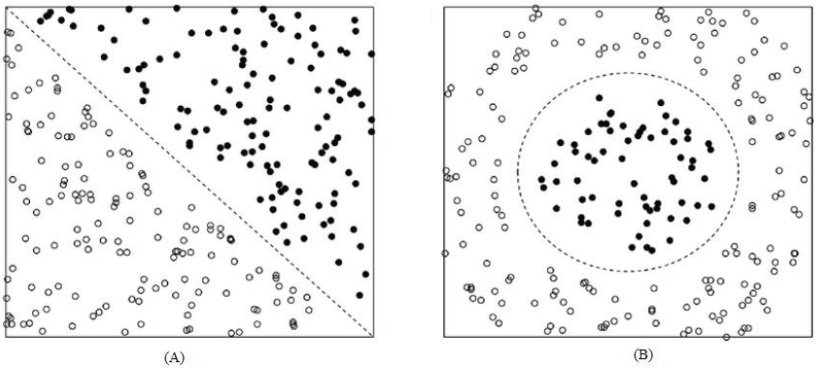
Αντίληπτρο Ενός Στρώματος



Οι τιμές των παραμέτρων ανανεώνονται μέχρις ότου όλες οι τιμές εισόδου να έχουν κατηγοριοποιηθεί σωστά και επομένως ο αλγόριθμος δεν τερματίζει αν τα δεδομένα δεν είναι γραμμικώς διαχωρίσιμα (*linearly seperatable*) (Patterson & Gibson, 2017). Αναφορικά, λέμε ότι δύο σύνολα είναι γραμμικώς διαχώρισμα όταν, για παράδειγμα στη περίπτωση των δύο διαστάσεων, μπορούμε να χαράξουμε μια γραμμή στο επίπεδο η οποία θα διαχωρίζει όλα τα σημεία του ενός συνόλου από του άλλου (βλ. Σχήμα 1.15. – Grove & Blinkhorn, 2020).

ΣΧΗΜΑ 1.16

Γραμμική (A) και Μη Γραμμική Διαχωρισιμότητα (B)

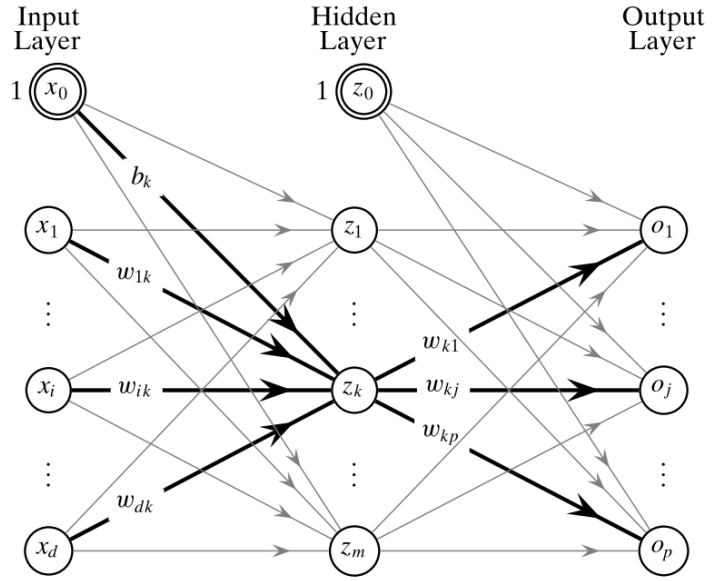


1.5.2 Αντίληπτρο Πολλαπλών Στρωμάτων (Multilayer Perceptron (MLP))

Το MLP νευρωνικό δίκτυο είναι γενίκευση του SLP και χρησιμοποιείται για την επίλυση πιο περίπλοκων προβλημάτων. Πέρα από την στιβάδα εισόδου (*input layer*) και εξόδου (*output layer*) που συναντάμε και στο SLP, ανάμεσα τους υπάρχει και μία ή περισσότερες κρυφές στιβάδες (*hidden layers*). Ένα νευρωνικό δίκτυο με πολλές κρυφές στιβάδες ονομάζεται βαθύ νευρωνικό δίκτυο (*deep neural network*). Η αύξηση στον αριθμό των κρυφών στιβάδων επιτρέπει στο δίκτυο να μάθει πιο περίπλοκα προβλήματα, αλλά η ικανότητα του να γενικεύει μειώνεται και ο χρόνος εκπαίδευσης αυξάνεται (Atkinson & Tatnall, 1997). Στο Σχήμα 1.17 (Zaki & Meira, 2020) μπορούμε να δούμε ένα MLP με μία κρυφή στιβάδα και σύμφωνα με τους συμβολισμούς που χρησιμοποιήθηκαν στις Ενότητες 1.1 και 1.3, με έντονη γραμμοσκίαση δίνονται οι συνδέσεις εισόδου και εξόδου για τον νευρώνα z_k.

ΣΧΗΜΑ 1.17

Αντίληπτρο Πολλαπλών Στρωμάτων με Μια Κρυφή Στιβάδα



Τα δεδομένα εκπαίδευσης προσδιορίζουν άμεσα ότι η στιβάδα εξόδου, για κάθε σήμα εισόδου, πρέπει να παραγάγει τιμές (πρόβλεψης) κοντά στις επιθυμητές τιμές απόκρισης που έχουν δοθεί. Όμως, η συμπεριφορά των άλλων στιβάδων δεν προσδιορίζεται ευθέως από τα δεδομένα εκπαίδευσης και αυτό είναι κάτι που πρέπει να αποφασίσει ο αλγόριθμος βελτιστοποίησης, προκειμένου να πετύχει τη καλύτερη δυνατή τιμή πρόβλεψης. Αυτός είναι και ο λό-

γος που οι στιβάδες αυτές ονομάστηκαν κρυφές στιβάδες (Goodfelow *et al.,* 2016). Στους κόμβους των κρυφών στιβάδων μπορούν να ενσωματωθούν μη γραμμικές συναρτήσεις ενεργοποίησης, οι οποίες όπως αναφέραμε και στην Ενότητα 1.2, δίνουν τη δυνατότητα στο νευρωνικό δίκτυο να αναγνωρίζει μη γραμμικές σχέσεις μεταξύ των πληροφοριών εισόδου και εξόδου (Patterson & Gibson, 2017; Zaki & Meira, 2020). Επιπλέον, έχει αποδειχτεί βάσει του θεωρήματος καθολικής προσέγγισης (*universal approximation theorem*), πως κάθε συνεχής συνάρτηση μπορεί να προσεγγιστεί ομοιόμορφα από ένα νευρωνικό δίκτυο που θα έχει μόνο μια κρυφή στιβάδα και πεπερασμένο αριθμό νευρώνων (Cybenko, 1989). Τις περισσότερες φορές η σύνδεση μεταξύ των στιβάδων γίνεται με τις εξερχόμενες συνδέσεις όλων των νευρώνων της προηγούμενης στιβάδας με όλους τους νευρώνες της αμέσως επόμενης (*fully-connected layers*), ενώ δεν υπάρχουν συνδέσεις μεταξύ των κόμβων της ίδιας στιβάδας. Στη περίπτωση αυτή, το νευρωνικό δίκτυο ονομάζεται Νευρωνικό Δίκτυο Εμπρόσθιας Διάδοσης (Feed-Forward Neural Network (FFNN)). Επίσης, συνηθίζεται όλοι οι κόμβοι που ανήκουν στην ίδια στιβάδα να έχουν την ίδια συνάρτηση ενεργοποίησης (πέρα από τη στιβάδα εισόδου που δεν χρησιμοποιεί κάποια συνάρτηση ενεργοποίησης και με το σήμα εξόδου της απλώς τροφοδοτεί το δίκτυο με τα μη επεξεργασμένα δεδομένα εισόδου). Ανάλογα με τη φύση του προβλήματος που θέλουμε να επιλύσουμε, το τελικό αποτέλεσμα του νευρωνικού δικτύου μπορεί να είναι μια πραγματική τιμή (προβλήματα παλινδρόμησης) ή ένα σύνολο τιμών στον χώρο των πιθανοτήτων (προβλήματα κατηγοριοποίησης). Αυτό καθορίζεται από τη συνάρτηση ενεργοποίησης στους κόμβους της στιβάδας εξόδου. Αν η συνάρτηση ενεργοποίησης είναι γραμμική, τότε το νευρωνικό δίκτυο είναι ένα μοντέλο παλινδρόμησης, ενώ για προβλήματα κατηγοριοποίησης συνηθίζεται να χρησιμοποιούνται οι συναρτήσεις softmax και sigmoid (Burkov, 2019; Patterson & Gibson, 2017).

Στα MLPs υπάρχουν δύο διαφορετικοί αλγόριθμοί διάδοσης πληροφορίας που χρησιμοποιούνται κατά τη διαδικασία της εκπαίδευσης του νευρωνικού δικτύου (Haykin, 1998):

- **Εμπρόσθια Διάδοση (Forward Propagation)**
  Τα σήματα εισόδου, τα οποία ξεκινώντας από τη στιβάδα εισόδου διαδίδονται προς τα εμπρός μέχρι να καταλήξουν στη στιβάδα εξόδου και να παραχθεί το αποτέλεσμα της συνάρτησης απώλειας. Όπως, είδαμε και στην Ενότητα 1.1 τα σήματα αυτά εισέρχονται σε έναν νευρώνα υπολογίζοντας το αντίστοιχο σταθμισμένο άθροισμα.

- **Οπισθοδρομική Διάδοση (Back-Propagation)**

  Ο αλγόριθμος αυτός επιτρέπει στην πληροφορία που λαμβάνουμε από την συνάρτηση απώλειας, στο τέλος της διαδικασίας εμπρόσθιας διάδοσης, να ενσωματωθεί στο δίκτυο. Αυτό επιτυγχάνεται υπολογίζοντας, με τον κανόνα της αλυσίδας (*chain rule*), την κλίση $\nabla_\theta L(\theta)$, ξεκινώντας από τη στιβάδα εξόδου και συνεχίζοντας στιβάδα-στιβάδα με κατεύθυνση προς τα πίσω μέχρι τη στιβάδα εισόδου. Αξίζει να σημειωθεί, πως αν και συχνά παρερμηνεύεται ως η διαδικασία με την οποία πραγματοποιείται ολόκληρη η εκπαίδευση του νευρωνικού δικτύου, στη πραγματικότητα με τον όρο back-propagation αναφερόμαστε μόνο στην μέθοδο υπολογισμού της κλίσης η οποία στη συνέχεια αξιοποιείται από τους αλγορίθμους βελτιστοποίησης για την εκπαίδευση του δικτύου (Goodfelow *et al.,* 2016). Έτσι, ο αλγόριθμος back-propagation στην ουσία υπολογίζει τη συνεισφορά στο σφάλμα πρόβλεψης που έχει κάθε σύνδεση του δικτύου ενώ τα βάρη των συνδέσεων τροποποιούνται στη συνέχεια από κάποιο αλγόριθμο βελτιστοποίησης (Geron, 2019; Patterson & Gibson, 2017). Για να πετύχει η διαδικασία της εκπαίδευσης, είναι σημαντικό οι αρχικές τιμές των βαρών των συνδέσεων που θα δοθούν στο μοντέλο του νευρωνικού δικτύου να είναι τυχαίες. Αν όλα τα βάρη και οι πολώσεις του δικτύου έχουν την ίδια αρχική τιμή, τότε για μια δεδομένη στιβάδα όλοι οι νευρώνες που θα υπάρχουν σε αυτή θα είναι ακριβώς οι ίδιοι, με αποτέλεσμα ο back-propagation αλγόριθμος να τους επηρεάσει με τον ίδιο ακριβώς τρόπο και τελικά, ακόμη και αν υπάρχουν εκατοντάδες νευρώνες σε κάθε στιβάδα, το μοντέλο θα συμπεριφέρεται σαν να υπήρχε μόνο ένας νευρώνας ανά στιβάδα (Geron, 2019).

1.5.3 Συνελικτικό Νευρωνικό Δίκτυο (Convolutional Neural Network (CNN))

Τα συνελικτικά νευρωνικά δίκτυα είναι ένα είδος FFNN που κατασκευάστηκαν για να εκμεταλλεύονται δεδομένα εισόδου τα οποία ακολουθούν κάποιο χωρικό ή χρονικό σχεδιασμό. Παραδείγματα τέτοιων δεδομένων είναι οι εικόνες και οι χρονοσειρές (π.χ. δεδομένα ήχου) στις οποίες παρατηρούνται συγκεκριμένα μοτίβα που επαναλαμβάνονται και διπλανές τιμές εισόδου σχετίζονται μεταξύ τους (Patterson & Gibson, 2017; Zaki & Meira, 2020). Επιπλέον, έχουν την ικανότητα να μειώνουν σημαντικά των αριθμό των παραμέτρων σε ένα βαθύ νευρωνικό δίκτυο χωρίς αυτό να σημαίνει ότι μειώνουν και τη ποιότητα του μοντέλου. Όπως είδαμε, στα MLPs όλοι οι νευρώνες μια στιβάδας l συνδέονται με όλους τους νευρώνες

της επόμενης στιβάδας $l+1$. Αυτό σημαίνει ότι στην περίπτωση επεξεργασίας εικόνων, όπου είναι και η συνηθέστερη εφαρμογή των CNNs, ο αριθμός των παραμέτρων θα γίνει τεράστιος, καθώς κάθε σήμα εισόδου του μοντέλου θα έχει διάσταση όσο και το συνολικό πλήθος των pixel της εικόνας (διαδικασία *flattening*), αφού η είσοδος σε ένα MLP πρέπει να δίνεται σε μορφή διανύσματος. Για παράδειγμα μια εικόνα διάστασης 100x100 pixels (δηλαδή απαιτεί $d = 10.000$ νευρώνες εισόδου) που χρησιμοποιείται σε ένα μοντέλο που απαρτίζεται από μια κρυφή στιβάδα με $m = 1.000$ το πλήθος νευρώνες και $p = 5$ νευρώνες στην στιβάδα εξόδου, θα έχει συνολικά $d \cdot m + m \cdot p = 10.005.000$ βάρη και επιπλέον $m + p = 1.005$ παραμέτρους πόλωσης πάνω στις οποίες θα πρέπει να εκπαιδευτεί το δίκτυο (Shanmugamani, 2018; Zaki & Meira, 2020). Στα CNNs το πρόβλημα αυτό λύνεται, αφού συνδέει ένα υποσύνολο γειτονικών νευρώνων της στιβάδας l με ένα μόνο νευρώνα της στιβάδας $l+1$. Έτσι, διαφορετικά "κυλιόμενα παράθυρα" (*sliding windows*) που αποτελούνται από μικρότερες περιοχές της στιβάδας l συνδέονται με διαφορετικούς νευρώνες της στιβάδας $l+1$. Τα κυλιόμενα αυτά παράθυρα μοιράζονται τις ίδιες παραμέτρους (*parameter sharing*), αφού το ίδιο σύνολο βαρών που ονομάζεται φίλτρο ή πυρήνας συνέλιξής (*filter, convolution kernel*) χρησιμοποιείται σε όλα τα κυλιόμενα παράθυρα (Zaki & Meira, 2020). Η διαδικασία αυτή ονομάζεται συνέλιξη και το αποτέλεσμα της ονομάζεται χάρτης χαρακτηριστικών ή αλλιώς χάρτης ενεργοποίησης (*feature map, activation map*).

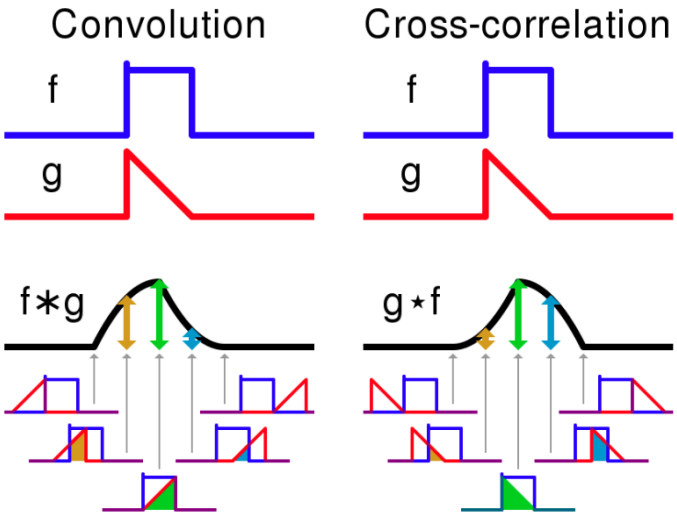
Αναφορικά η συνέλιξη πήρε το όνομα της από την μαθηματική διαδικασία η οποία εκφράζει το ποσό της επικάλυψης μιας συνάρτησης g καθώς αυτή μετατοπίζεται ως προς μία συνάρτηση f:

$$(f * g)(t) := \int f(\tau)\, g(t-\tau) d\tau$$

Όπως φαίνεται και στο Σχήμα 1.18 (en.wikipedia.org), στη συνέλιξη η συνάρτηση ανακλάται ως προς τον κατακόρυφο άξονα πριν μετακινηθεί κατά μήκος της συνάρτησης f. Αυτό είναι κάτι που δεν συμβαίνει κατά την συνέλιξη στα νευρωνικά δίκτυα (θεωρώντας ως συνάρτηση g τον πυρήνα συνέλιξης και ως f το σύνολο των νευρώνων της στιβάδας l), οπότε η διαδικασία, πιο επίσημα, είναι αυτή της δια-συσχέτισης (*cross-correlation*). Στη πραγματικότητα όμως η διαφορά αυτή δεν θα επηρέαζε την απόδοση του μοντέλου (αφού απλώς, κατά την εκπαίδευση, τα βάρη θα εκπαιδεύονταν σε διαφορετικά κελιά του φίλτρου για να προσαρμοστεί ο αλγόριθμος στην ανάκλαση) και επομένως έχει γίνει η σύμβαση να χρησιμοποιούμε τον όρο συνέλιξη.

ΣΧΗΜΑ 1.18

Σύγκριση Συνέλιξης και Διασυσχέτισης



Πιο συγκεκριμένα, σε μια δισδιάστατη συνέλιξη η είσοδος $\boldsymbol{X}$ είναι ένας $n \times n$ πίνακας, το δισδιάστατο φίλτρο $\boldsymbol{W}$ είναι ένας $k \times k$ πίνακας με τα βάρη και το $k \leq n$ είναι το μέγεθος του κυλιόμενου παραθύρου. Επιπλέον, έστω $\boldsymbol{X}_k(i,j)$ ο $k \times k$ υποπίνακας του $\boldsymbol{X}$ που ξεκινάει από την γραμμή i και στήλη j και ορίζεται ως:

$$\boldsymbol{X}_k(i,j) = \begin{pmatrix} x_{i,j} & x_{i,j+1} & \dots & x_{i,j+k-1} \\ x_{i+1,j} & x_{i+1,j+1} & \dots & x_{i+1,j+k-1} \\ \vdots & \vdots & \dots & \vdots \\ x_{i+k-1,j} & x_{i+k-1,j+1} & \dots & x_{i+k-1,j+k-1} \end{pmatrix}$$

με $1 \leq i,j \leq n-k+1$. Για έναν τυχαίο τετραγωνικό πίνακα $\boldsymbol{A} \in \mathbb{R}^{k \times k}$ ορίζουμε το άθροισμα $sum(\boldsymbol{A})$ ως το άθροισμα όλων των στοιχείων του, δηλαδή:

$$sum(\boldsymbol{A}) = \sum_{i=1}^{k} \sum_{j=1}^{k} \alpha_{i,j}$$

όπου, $\alpha_{i,j}$ είναι το στοιχείο της γραμμής i και στήλης j του πίνακα $\boldsymbol{A}$. Η δισδιάστατη συνέλιξη του $\boldsymbol{X}$ και $\boldsymbol{W}$ συμβολίζεται ως $\boldsymbol{X} * \boldsymbol{W}$ και ορίζεται ως:

$$\boldsymbol{X} * \boldsymbol{W} = \begin{pmatrix} sum(X_k(1,1) \odot W) & \dots & sum(X_k(1,n-k+1) \odot W \\ sum(X_k(2,1) \odot W) & \dots & sum(X_k(2,n-k+1) \odot W \\ \vdots & \dots & \vdots \\ sum(X_k(n-k+1,1) \odot W) & \dots & sum(X_k(n-k+1,n-k+1) \odot W) \end{pmatrix}$$

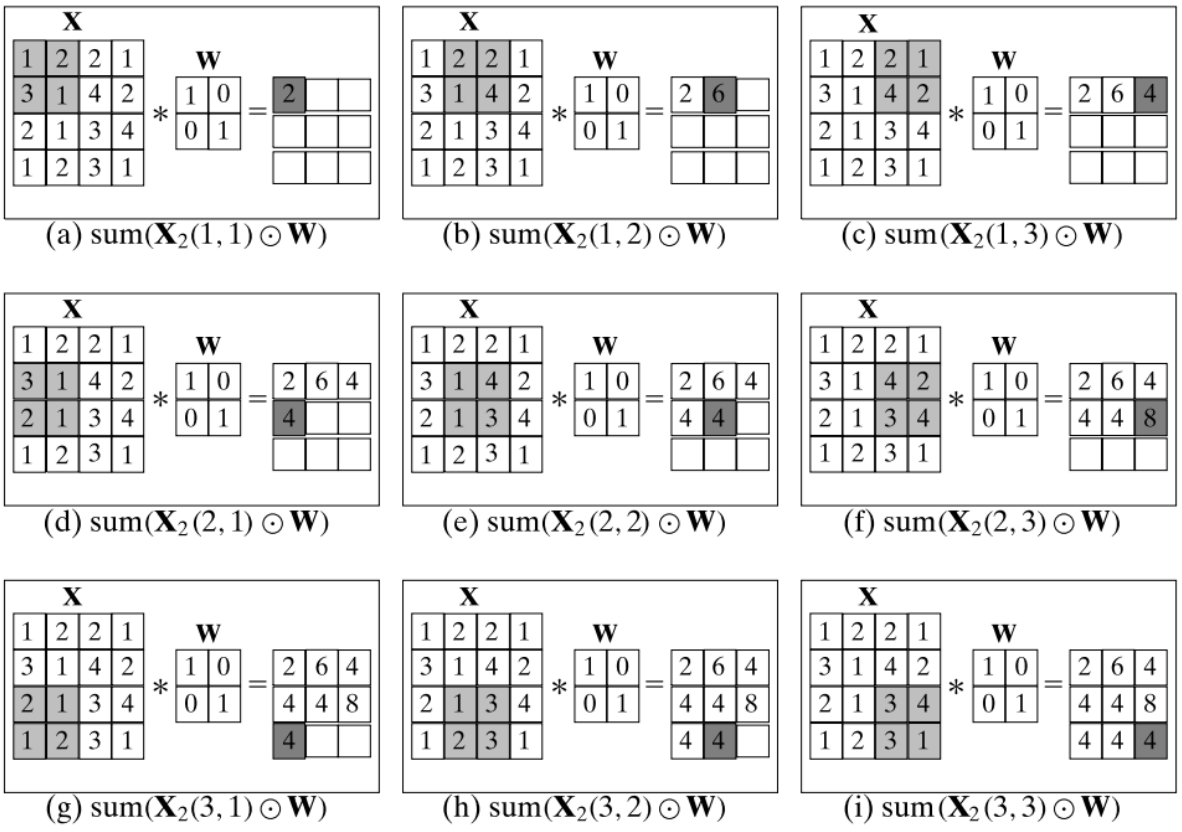
όπου, $\odot$ το γινόμενο Hadamard, δηλαδή το γινόμενο ανά στοιχείο των πινάκων $\boldsymbol{X}_k(i,j)$ και $\boldsymbol{W}$ έτσι ώστε να έχουμε ότι:

$$sum(\boldsymbol{X}_k(i,j) \odot W) = \sum_{a=1}^{k}\sum_{b=1}^{k} x_{i+a-1,j+b-1} \cdot w_{a,b}$$

για $i,j = 1,2,\dots,n-k+1$. Έτσι, η συνέλιξη του $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ και του $\boldsymbol{W} \in \mathbb{R}^{k\times k}$ μας δίνει έναν $(n-k+1) \times (n-k+1)$ πίνακα. Η διαδικασία στην μια και στις τρεις διαστάσεις είναι ανάλογη. Στο Σχήμα 1.19 (Zaki & Meira, 2020) φαίνεται η δισδιάστατη συνέλιξη διαφορετικών 2×2 παραθύρων κύλισης του $\boldsymbol{X}$ και του φίλτρου $\boldsymbol{W}$. Το αποτέλεσμα της συνέλιξης φαίνεται στο βήμα (i) του σχήματος.

ΣΧΗΜΑ 1.19

Δισδιάστατη Συνέλιξη



Επιπλέον, με $b \in \mathbb{R}$ συμβολίζουμε την τιμή πόλωσης για το φίλτρο $\boldsymbol{W}$ και $\boldsymbol{X}^{l+1}$ είναι ο $(n-k+1) \times (n-k+1)$ τανυστής (*tensor*), που βρίσκεται αμέσως μετά τον $\boldsymbol{X}^{\boldsymbol{l}} = \boldsymbol{X}$, και α-

ποτελείται από τους νευρώνες της στιβάδας $l+1$, έτσι ώστε $x_{i,j}^{l+1}$ να είναι η τιμή του νευρώνα στη γραμμή i και στήλη j της στιβάδας $l+1$ με $1 \leq i,j \leq (n-\ k\ +1)$. Το αντίστοιχο του σήματος net, που είδαμε και στην Ενότητα 1.2, στο νευρώνα $x_{i,j}^{l+1}$ θα είναι:

$$net_{i,j}^{l+1} = sum\big(\boldsymbol{X}_k^l(i,j) \odot W\big) + b$$

Μέχρι στιγμής ο μετασχηματισμός είναι αφινικός (γραμμικός μετασχηματισμός συνοδευόμενος από μετατόπιση (*affine transformation*)). Οπότε, επιτυγχάνουμε τη μη γραμμικότητα των νευρώνων του δικτύου με το να εφαρμόζουμε στο παραπάνω net σήμα μία από τις γνωστές συναρτήσεις ενεργοποίησης f (όπως κάναμε και στα MLPs). Το στάδιο αυτό συχνά θεωρείται ως διαφορετική στιβάδα και ονομάζεται στιβάδα ανίχνευσης (*detector layer*) (Goodfelow *et al.*, 2016). Τελικά η τιμή του νευρώνα $x_{i,j}^{l+1}$ θα είναι:

$$x_{i,j}^{l+1} = f\big(sum\big(\boldsymbol{X}_k^l(i,j) \odot W\big) + b\big)$$

άρα οι τιμές στους νευρώνες της στιβάδας $l+1$ θα είναι:

$$\boldsymbol{X}^{l+1} = f\left(\left(\boldsymbol{X}^{\boldsymbol{l}} * W\right) \oplus b\right)$$

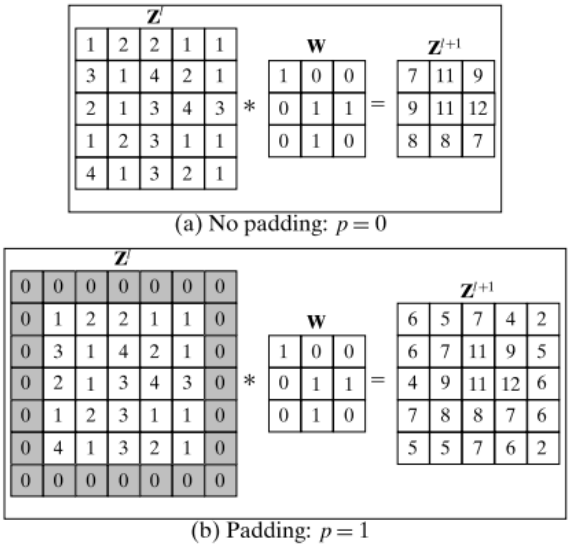
όπου $\oplus$ υποδεικνύει ότι ο όρος πόλωσης προστίθεται σε κάθε στοιχείο του $(n-\ k\ +1) \times (n-\ k\ +1)$ πίνακα $\boldsymbol{X}^{\boldsymbol{l}} * W$.

Έτσι, λόγω της συνέλιξης, το δίκτυο μπορεί και εντοπίζει πολλά τοπικά χαρακτηριστικά όπου η ακριβής τους τοποθεσία είναι λιγότερο σημαντική, καθώς η θέση τους, σχετικά με τη θέση των άλλων χαρακτηριστικών, κατά προσέγγιση διατηρείται (Haykin, 1998).

Όπως είδαμε, στο τέλος της συνέλιξης ο χάρτης ενεργοποίησης που προκύπτει είναι μικρότερης διάστασης από την είσοδό του. Έτσι, μετά από πολλές στιβάδες συνελίξεων και διαδοχικές μειώσεις στην διάσταση, εύκολα μπορεί να καταλήξουμε με μια στιβάδα που να αποτελείται από έναν μόνο νευρώνα. (Goodfelow *et al.*, 2016; Zaki & Meira, 2020). Για να αποφύγουμε αυτό το πρόβλημα χρησιμοποιούμε την τεχνική του γεμίσματος (*padding*), σύμφωνα με την οποία τοποθετούμε p στο πλήθος γραμμές και πάνω και κάτω από τον τανυστή εισόδου, αλλά και p στο πλήθος στήλες στα δεξιά και στα αριστερά του. Η τιμή του p δίνεται από τον τύπο $p = \left\lfloor \frac{k-1}{2} \right\rfloor$. Στο Σχήμα 1.20 (Zaki & Meira, 2020) βλέπουμε ένα παράδειγμα για $p = 0$ και για $p = 1$.

ΣΧΗΜΑ 1.20

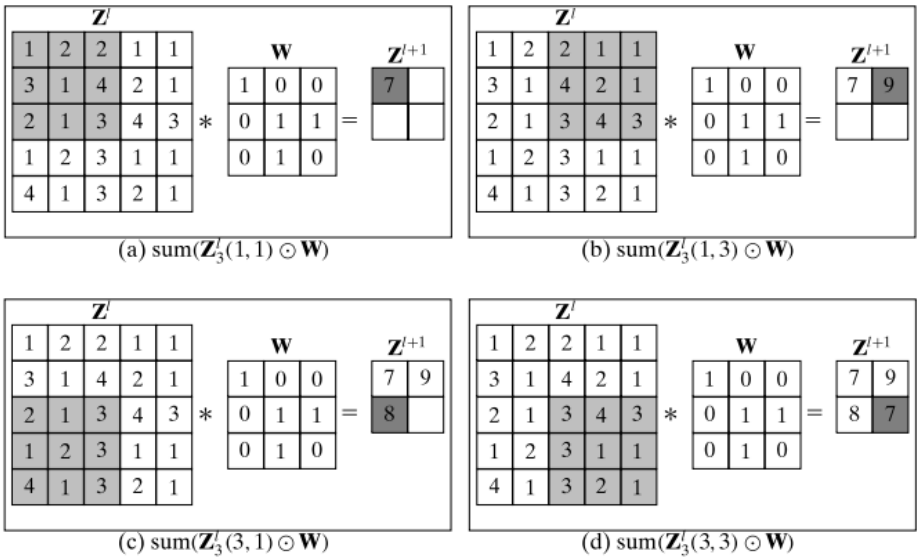
Τεχνική Γεμίσματος



Ένα ακόμη χαρακτηριστικό της συνέλιξης είναι ο βηματισμός (*stride*). Με τον βηματισμό ρυθμίζεται το πόσο μακριά θα μετακινηθεί το κυλιόμενο παράθυρο μετά από κάθε εφαρμογή του φίλτρου. Όταν η τιμή του βηματισμού είναι μεγάλη υπάρχει λιγότερη αλληλοεπικάλυψη μεταξύ των κυλιόμενων παραθύρων με αποτέλεσμα να μειωθεί το συνολικό πλήθος τους και το υπολογιστικό κόστος του αλγορίθμου. Πιο συγκεκριμένα αν s είναι η τιμή του βηματισμού, τότε η συνέλιξη του τανυστή $\boldsymbol{X^l} \in \mathbb{R}^{n \times n}$ με το φίλτρο $\boldsymbol{W} \in \mathbb{R}^{k \times k}$ θα μας δώσει έναν $(t+1) \times (t+1)$ πίνακα, όπου t είναι ο κατώτατος ακέραιος (*floor*) της ποσότητας $\frac{n-k}{s}$, δηλαδή $t = \left\lfloor \frac{n-k}{s} \right\rfloor$. Στο Σχήμα 1.21 (Zaki & Meira, 2020) δίνεται η συνέλιξη που πραγματοποιήθηκε στο παράδειγμα του Σχήματος 1.20 – (a) αλλά αυτή τη φορά ο βηματισμός αντί για $s = 1$ είναι $s = 2$. Το αποτέλεσμα είναι ένας 2×2 πίνακας (αφού, $t = \left\lfloor \frac{5-3}{2} \right\rfloor = 1$) σε αντίθεση με το προηγούμενο αποτέλεσμα που ήταν ένας 3×3 πίνακας.

ΣΧΗΜΑ 1.21

Δισδιάστατη Συνέλιξη με βηματισμό $s = 2$



Στην διαδικασία της συνέλιξης που ακολουθήσαμε παραπάνω, για να πάρουμε το αποτέλεσμα της εφαρμογής του φίλτρου πάνω σε ένα κυλιόμενο παράθυρο αθροίσαμε όλα τα στοιχεία που προκύπτουν από το γινόμενο Hadamard. Πέρα από τη συνάρτηση του αθροίσματος στα CNNs χρησιμοποιούνται και άλλες συναρτήσεις ομαδοποίησης (*aggregation functions*), όπως η μέση τιμή (*average*) και το μέγιστο (*maximum*). Η τεχνική στην οποία εφαρμόζονται τέτοιες συναρτήσεις ονομάζεται συσσώρευση (*pooling*) και συνήθως χρησιμοποιείται μεταξύ διαδοχικών στιβάδων συνελίξεων. Οι στιβάδες που προκύπτουν μετά την χρήση του pooling ονομάζονται στιβάδες συσσώρευσης (*pooling layers*). Συνηθίζεται να χρησιμοποιείται η συνάρτηση του μεγίστου και να συνδυάζεται με βηματισμό ίσο με το μέγεθος του φίλτρου (δηλαδή $s = k$), προκειμένου η συνάρτηση ομαδοποίησης να εφαρμόζεται σε κυλιόμενα παράθυρα που δεν μοιράζονται κοινούς νευρώνες.

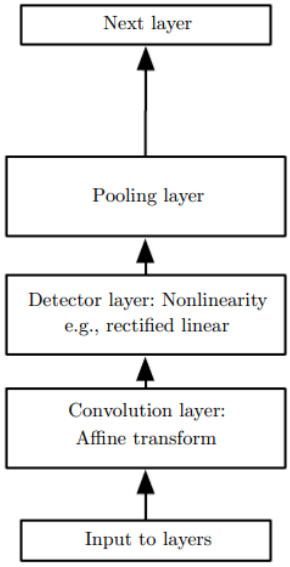
Τα φίλτρα που εφαρμόζονται στο στάδιο της συνέλιξης περιέχουν βάρη και πολώσεις τα οποία αξιοποιούνται για την εκπαίδευση του νευρωνικού δικτύου. Αντιθέτως το φίλτρο $\boldsymbol{W} \in \mathbb{R}^{k \times k}$ σε μια στιβάδα συσσώρευσης είναι ένας τανυστής που όλα του τα βάρη είναι σταθερά και ίσα με τη μονάδα και η τιμή πόλωσης του είναι επίσης σταθερή και ίση με το μηδέν. Αυτό σημαίνει ότι οι παράμετροι των στιβάδων συσσώρευσης δεν ανανεώνονται από τον αλγόριθμο βελτιστοποίησης κατά την μάθηση του δικτύου. Επιπλέον, η συνάρτηση ενεργοποίησης που χρησιμοποιείται σε τέτοιου είδους στιβάδες είναι αποκλειστικά η ταυτοτική και δεν συνηθίζεται να χρησιμοποιείται η τεχνική του γεμίσματος. (Patterson & Gibson, 2017; Zaki

& Meira, 2020). Ο σκοπός των στιβάδων συσσώρευσης είναι να μειώσουν τη διάσταση από τους χάρτες χαρακτηριστικών, προσπαθώντας όμως να διατηρήσουν το σημαντικότερο μέρος της πληροφορίας τους και επιπλέον βοηθάνε στον περιορισμό της υπερπροσαρμογής (*overfitting*) του δικτύου στα δεδομένα εκπαίδευσης (Goodfelow *et al.,* 2016;).

Τα επίπεδα, ενός CNN νευρωνικού δικτύου, που περιγράψαμε συνοψίζονται στο Σχήμα 1.22 (Goodfelow *et al.,* 2016;) που ακολουθεί.

ΣΧΗΜΑ 1.22

Δομικά Στοιχεία ενός Συνελικτικού Νευρωνικού Δικτύου



1.5.4 Αναδρομικά ή Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks RNNs))

Τα αναδρομικά νευρωνικά δίκτυα είναι μια οικογένεια νευρωνικών δικτύων τα οποία ειδικεύονται στην επεξεργασία ακολουθιακών δεδομένων (αλλά μπορούν να εφαρμοστούν και σε δισδιάστατα χωρικά δεδομένα όπως οι εικόνες) και είναι ιδιαίτερα χρήσιμα σε ένα μεγάλο εύρος εφαρμογών όπως η επεξεργασία της φυσικής γλώσσας (*natural language processing*), η αναγνώριση ομιλίας (s*peech recognition*) και η πρόβλεψη χρονοσειρών (Graves, 2019). Σε αντίθεση με τα δίκτυα εμπρόσθιας διάδοσης, όπου η πληροφορία μεταφέρεται προς μία μόνο κατεύθυνση, τα RNNs περιέχουν βρόχους ανάδρασης (*feedback loops*) που τους επιτρέπουν να δημιουργούν συνδέσεις μεταξύ διαφορετικών χρονικών βημάτων (*time-steps*). Σημαντικό κομμάτι της αρχιτεκτονικής τους είναι το γεγονός ότι μοιράζονται τις ίδιες παραμέτρους για πολλά χρονικά βήματα. Ο σχεδιασμός τους βασίζεται στη μονοδιάστατη συνέλι-

ξη στην οποία, όπως είδαμε, κάθε κομμάτι της εξόδου είναι συνάρτηση ενός μικρού αριθμού γειτονικών στοιχείων της εισόδου. Στα RNNs η κοινή χρήση των παραμέτρων διαμορφώνεται διαφορετικά, καθώς κάθε στοιχείο της εξόδου παράγεται χρησιμοποιώντας τον ίδιο κανόνα ανανέωσης με τις προηγούμενες εξόδους (Goodfelow *et al.*, 2016). Πιο συγκεκριμένα, κάθε κρυφό διάνυσμα $h_t \in \mathbb{R}^m$ μιας κρυφής στιβάδας l, τη χρονική στιγμή t, εξαρτάται από το διάνυσμα εισόδου x_t (δηλαδή την έξοδο της στιβάδας $l-1$) και από τη προηγούμενη κρυφή κατάσταση (*hidden state*) h_{t-1} που προέρχεται από τη χρονική στιγμή $t-1$. Έτσι η κρυφή κατάσταση h_t υπολογίζεται ως εξής:

$$h_t = f^h(W^T x_t + W_h^T h_{t-1} + b_h)$$

όπου,

- $X = \langle x_1, x_2, \dots, x_\tau \rangle$ η ακολουθία εισόδου από τα διανύσματα $x_t \in \mathbb{R}^d$, $t = 1,2, \dots, \tau$
- f^h μια συνάρτηση ενεργοποίησης (η οποία συνηθίζεται να είναι η υπερβολική εφαπτομένη ή ReLU),
- $W_i \in \mathbb{R}^{m \times p}$ ο πίνακας με τα βάρη μεταξύ των διανυσμάτων εισόδου και των διανυσμάτων των κρυφών καταστάσεων, με p η διάσταση ενός σήματος εξόδου $o_t \in \mathbb{R}^p$ που αντιστοιχεί σε μια πραγματική τιμή απόκρισης $y_t \in \mathbb{R}^p$
- $W_h \in \mathbb{R}^{m \times m}$ ο πίνακας με τα βάρη μεταξύ των διανυσμάτων της κρυφής κατάστασης στο χρόνο $t-1$ και t
- $b_h \in \mathbb{R}^m$ οι παράμετροι πόλωσης που σχετίζονται μόνο με τις κρυφές καταστάσεις (να σημειωθεί πως δεν χρειαζόμαστε διάνυσμα με τιμές πόλωσης μεταξύ της τιμής εισόδου και των κρυφών νευρώνων).

Έτσι το διάνυσμα εξόδου o_t τη χρονική τη χρονική στιγμή t για ένα δεδομένο διάνυσμα κρυφής κατάστασης h_t δίνεται από τον τύπο:

$$o_t = f^o(W_o^T h_t + b_o)$$

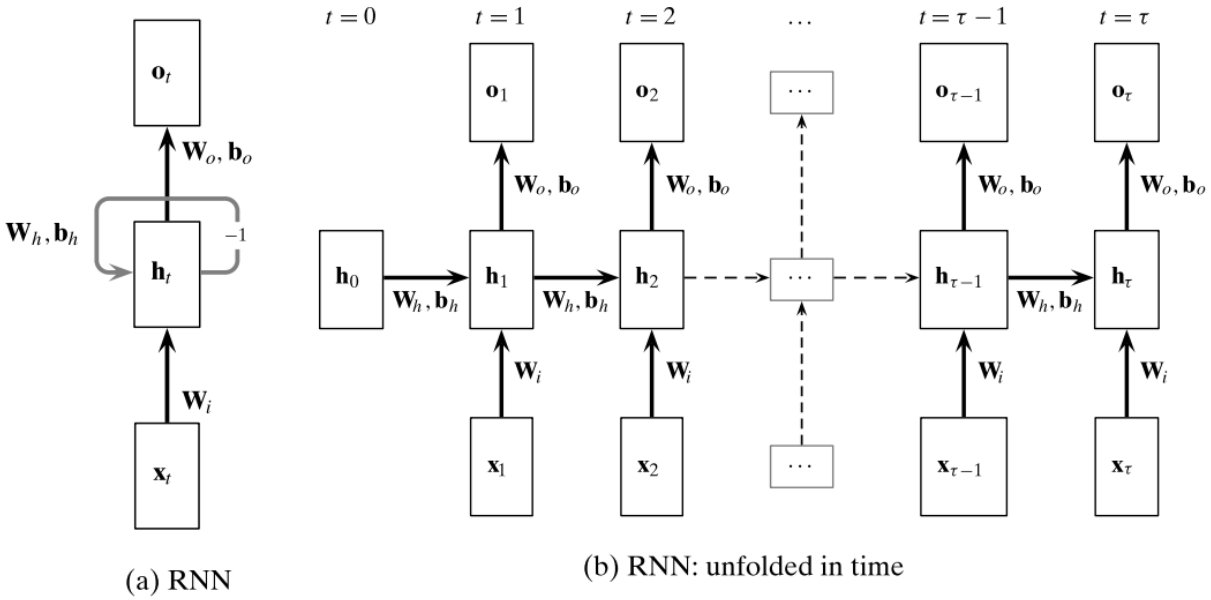
όπου, $W_o \in \mathbb{R}^{m \times p}$ είναι ο πίνακας των βαρών μεταξύ της κρυφής κατάστασης και των διανυσμάτων εξόδου με διάνυσμα παραμέτρων πόλωσης το b_o. Η συνάρτηση ενεργοποίησης f^o συνήθως είναι η γραμμική ή η ταυτοτική για προβλήματα παλινδρόμησης και η softmax για προβλήματα κατηγοριοποίησης.

Στο Σχήμα 1.23 (Zaki & Meira, 2020) φαίνεται η αναπαράσταση ενός RNN με μια κρυφή στιβάδα h_t. Η εξάρτηση της h_t από την χρονική στιγμή $t-1$ δηλώνεται με το -1 που αναγράφεται στην άκρη του βρόχου. Επιπλέον, από το σχήμα μπορούμε να δούμε την

κοινή χρήση των παραμέτρων μεταξύ των στιβάδων αφού οι πίνακες W_i, W_h και W_o μαζί με τα βάρη b_h και b_o μοιράζονται στο πέρασμα του χρόνου.

ΣΧΗΜΑ 1.23

Αναδρομικό Νευρωνικό Δίκτυο (a) Ξεδιπλωμένο στον Χρόνο (b)



(a) RNN

(b) RNN: unfolded in time

Στη συνέχεια θα αναφέρουμε δύο παραλλαγές των RNNs που παρουσιάζουν ιδιαίτερο ενδιαφέρον.

- **Αμφίδρομα Αναδρομικά Νευρωνικά Δίκτυα (Bidirectional RNNs (BRNNs))**

  Τα BRNNs επιτρέπουν στο δίκτυο να επεξεργαστεί τα ακολουθιακά δεδομένα προς και τις δύο κατευθύνσεις. Αυτό επιτυγχάνεται με το να συνδυάζει ένα RNN που κινείται προς τα εμπρός στο χρόνο από την αρχή της ακολουθίας $(x_1, x_2, \dots, x_\tau)$ και ένα άλλο RNN που κινείται πίσω στον χρόνο ξεκινώντας με την επεξεργασία της ακολουθίας από το τέλος της $(x_\tau, x_{\tau-1}, \dots, x_1)$. Σε κάθε περίπτωση αποθηκεύεται και ένα διαφορετικό διάνυσμα κρυφής κατάστασης και η τιμή εξόδου για μια χρονική στιγμή χρησιμοποιεί την πληροφορία και από τις δύο κρυφές καταστάσεις (κάθε μία από τις οποίες υπολογίζεται ξεχωριστά). Έτσι με τον σχεδιασμό αυτό για κάθε σημείο της ακολουθίας παρέχεται στη στιβάδα εξόδου πληροφορία και από το παρελθόν αλλά και από το μέλλον. Σε πολλές εφαρμογές έχει αποδειχτεί ότι η επίδοση των BRNNs ξεπερνά αυτή των RNNs αν και, όπως είναι λογικό, είναι αρκετά πιο απαιτητικά όσον

αφορά το υπολογιστικό κόστος (Graves, 2019). Παράδειγμα τέτοιων εφαρμογών είναι η αναγνώρισης ομιλίας, όπου η ερμηνεία ενός ακουστικού σήματος για μία λέξη εξαρτάται τόσο από τις λέξεις που προηγήθηκαν όσο και από αυτές που ακολουθούν, ιδιαίτερα στη περίπτωση όπου η λέξη που προσπαθούμε να αποσαφηνίσουμε είναι ακουστικά ίδια ή παρόμοια με άλλες λέξεις (Goodfelow *et al.,* 2016).

- **Δίκτυο Μακράς Βραχύχρονης Μνήμης (Long Short-term Memory (LSTM))**

  Ένα από τα προβλήματα εκπαίδευσης των RNNs είναι η ευαισθησία τους στο πρόβλημα της εξαφάνισης και της "έκρηξης" της κλίσης *(vanishing, exploding gradient*). Η εξαφάνιση κλίσης αφορά την τάση που έχει η κλίση της συνάρτησης απώλειας να γίνεται ολοένα και μικρότερη καθώς ο back-propagation αλγόριθμος προχωράει από τις στιβάδες που βρίσκονται κοντά στη στιβάδα εξόδου προς τις στιβάδες που βρίσκονται πλησιέστερα στην στιβάδα εισόδου. Έτσι, όταν στη συνέχεια ο αλγόριθμος βελτιστοποίησης ανανεώνει τα βάρη των πρώτων στιβάδων αυτά θα παραμένουν ουσιαστικά αμετάβλητα και η εκπαίδευση θα δυσκολευτεί να συγκλίνει σε μια καλή λύση (Geron, 2019). Στη περίπτωση που συμβαίνει το αντίθετο, δηλαδή έχουμε το πρόβλημα της έκρηξης της κλίσης, η κλίση γίνεται ολοένα μεγαλύτερη, οι ανανεώσεις στα βάρη είναι πολύ μεγάλες και τελικά ο αλγόριθμος αποκλίνει. Στα δίκτυα εμπρόσθιας διάδοσης τα προβλήματα αυτά είναι λιγότερο έντονα καθώς χρησιμοποιούνται διαφορετικοί πίνακες βαρών για κάθε στιβάδα (Goodfelow *et al.,* 2016). Στα RNNs όμως επειδή χρησιμοποιείται ο ίδιος πίνακας βαρών σε όλα τα χρονικά βήματα μία μικρή αύξηση ή μείωση στα πρώτα χρονικά βήματα μπορεί να κλιμακωθεί στη πορεία αν εκπαιδεύουμε μεγάλες ακολουθίες που αποτελούνται από πολλά χρονικά βήματα (Geron, 2019; Zaki & Meira, 2020). Ο πιο αποτελεσματικό τρόπος επίλυσης αυτών των προβλημάτων είναι τα LSTM (Patterson & Gibson, 2017). Ένας κόμβος (*LSTM block*) ενός LSTM δικτύου αποτελείται από την πύλη εισόδου (*input gate*) , τη πύλη λήθης (*forget gate*), την πύλη εξόδου (*output gate*) και το κελί μνήμης (*memory cell*). Στο κελί μνήμης διατηρούνται πληροφορίες από προηγούμενες θέσεις στο χρόνο, η πύλη εισόδου καθορίζει ποιες πληροφορίες είναι χρήσιμες από την τωρινή χρονική στιγμή και θα πρέπει να περάσουν στο κελί μνήμης, η πύλη λήθης καθορίζει ποιες πληροφορίες από το προηγούμενο χρονικό βήμα (από την προηγούμενη κατάσταση του κελιού μνήμης) θα πρέπει να διατηρηθούν (ή όχι) και τέλος η πύλη εξόδου καθο-

ρίζει ποια θα είναι η νέα κρυφή κατάσταση που θα δοθεί στο επόμενο χρονικό βήμα ή αλλιώς ποιες πληροφορίες του κελιού μνήμης θα "διαβαστούν" ώστε να δοθούν ως έξοδο. Με τον τρόπο αυτό τα LSTM δίκτυα έχουν τη δυνατότητα να διατηρούν ή να απορρίπτουν επιλεκτικά πληροφορίες προκειμένου να κάνουν πιο ακριβείς προβλέψεις (Geron, 2019; Goodfelow *et al.,* 2016; Graves, 2019).

Μια παραλλαγή των LSTM δικτύων είναι τα GRUs (Gated Recurrent Units), όπου και εκείνα αξιοποιούν τον μηχανισμό των πυλών. Τα GRUs μειώνουν τον αριθμό των πυλών σε δύο συνδυάζοντας την πύλη λήθης και την πύλη εισόδου στην πύλη ανανέωσης (*update gate*) και επιπλέον έχουν άλλη μια πύλη, την πύλη επαναρύθμισης (*reset gate*). Και τα GRUs ρυθμίζουν τη ροή της πληροφορίας μέσα στον κόμβο αλλά η βασική διαφορά τους από τα LSTM είναι ότι δεν χρησιμοποιούν το κελί μνήμης μειώνοντας έτσι τον αριθμό των παραμέτρων στο δίκτυο με αποτέλεσμα η σύγκλιση να είναι συνήθως πιο γρήγορη (Tang *et al.,* 2016) .

ΚΕΦΑΛΑΙΟ 2

Χρονοσειρές

2.1 Εισαγωγή στις Χρονοσειρές

Μια χρονολογική σειρά ή απλά χρονοσειρά (*time series*) είναι ένα πεπερασμένο σύνολο από παρατηρήσεις που καταγράφονται ακολουθιακά στη διάρκεια του χρόνου και συνήθως ισαπέχουν χρονικά. Στα μαθηματικά συμβολίζουμε με X_t, $t \in T$ τις παρατηρήσεις μιας χρονοσειράς X, με δείκτη t να ανήκει σε ένα σύνολο T που υποδηλώνει χρόνο. Παριστάνουν την εξέλιξη, ενός χαρακτηριστικού κάποιου στοχαστικού φαινομένου, στον χρόνο αν και η ταξινόμηση τους μπορεί να γίνει σύμφωνα και με μια άλλη διάσταση, πέρα από αυτή του χρόνου, όπως ο χώρος. (Abanda *et al.*, 2018*;* Anderson, 1994). Ένα τέτοιο παράδειγμα είναι μια ακολουθία από μετρήσεις των επιπέδων μόλυνσης του αέρα σε διαφορετικά σημεία μιας πόλης. Οι μετρήσεις αυτές μπορούν να ταξινομηθούν σύμφωνα με την απόσταση κάθε σημείου από το κέντρο της πόλης, όποτε η διάσταση ταξινόμησης θα είναι χωρική αντί για τη συνηθισμένη χρονική. Η σειρά με την οποία έχουν καταγραφεί οι παρατηρήσεις παίζει σημαντικό ρόλο στην ανάλυση μιας χρονοσειράς και αυτό είναι ένα βασικό χαρακτηριστικό που την διαφοροποιεί από άλλα είδη στατιστικής ανάλυσης (Anderson, 1994; Granger & Newbold, 1986). Σε πολλά προβλήματα οι παρατηρήσεις που εξετάζονται είναι στατιστικά ανεξάρτητες, στις χρονοσειρές όμως διαδοχικές τιμές μπορεί να είναι εξαρτημένες μεταξύ τους και επιπλέον η εξάρτηση αυτή μπορεί να σχετίζεται με τη θέση των παρατηρήσεων στην ακολουθία (*serially dependent*).

Χρονοσειρές συναντάμε σε πολλούς τομείς της επιστήμης, καθώς σχεδόν σε κάθε επιστημονικό πεδίο υπάρχουν φαινόμενα, των οποίων η εξέλιξη και η μεταβολή με το πέρασμα του χρόνου, είναι σημείο ενδιαφέροντος. Μερικές από αυτές τις επιστήμες είναι η οικονομία (π.χ. δείκτης πληθωρισμού), η μετεωρολογία (π.χ. θερμοκρασία, βροχόπτωση), η κοινωνιολογία (π.χ. δείκτης εγκληματικότητας, μελέτη της τάσης με την οποία μεταβάλλεται ο πληθυσμός μιας πόλης), η αστρονομία (φωτεινότητα άστρων), η ιατρική (π.χ. ηλεκτροκαρ-

διογράφημα), η μηχανολογία (π.χ. κατανάλωση ηλεκτρικού ρεύματος), η σεισμολογία, η ωκεανογραφία κ.α. Τέλος, μερικοί από τους λόγους που μπορεί να χρησιμοποιήσουμε χρονοσειρές είναι η πρόβλεψη των μελλοντικών τιμών ενός χαρακτηριστικού με βάση τις τιμές του από το παρόν και το παρελθόν, ο έλεγχος ή η κατανόηση του μηχανισμού που παράγει τη χρονοσειρά (π.χ. αναγνώριση ασυνήθιστης συμπεριφοράς στα δεδομένα), η εξέταση των αλληλεπιδράσεων μεταξύ διάφορων σχετικών χρονοσειρών και η εκτίμηση της κοινής συμπεριφοράς τους κ.α. (Anderson, 1994; Box *et al*., 2008).

2.2 Κατηγορίες Χρονοσειρών

Οι χρονοσειρές ανάλογα με τη μορφή που έχει το σύνολο T διακρίνονται στις χρονοσειρές συνεχούς χρόνου (*continuous time series*) και στις χρονοσειρές διακριτού χρόνου (discrete time series).

2.2.1 Χρονοσειρές Συνεχούς Χρόνου

Οι χρονοσειρές συνεχούς χρόνου προκύπτουν στη περίπτωση που οι παρατηρήσεις (θεωρητικά) καταγράφονται συνεχώς για ένα χρονικό διάστημα, δηλαδή όταν $T = [a, b]$, με $a, b \in \mathbb{R}$. Στη περίπτωση αυτή συνηθίζεται να χρησιμοποιείται ο συμβολισμός $X(t)$ για την χρονοσειρά, υποδεικνύοντας έτσι ότι το σύνολο T είναι συνεχές. Παράδειγμα αυτής της κατηγορίας είναι η θερμοκρασία όπου είναι μια ποσότητα που μεταβάλλεται συνεχώς και η μέτρηση της μπορεί να γίνει σε οποιαδήποτε χρονική στιγμή.

2.2.2 Χρονοσειρές Διακριτού Χρόνου

Όταν το σύνολο T μιας χρονοσειράς είναι διακριτό, δηλαδή όταν οι μετρήσεις καταγράφονται από χρονικά διαστήματα που ισαπέχουν χρονικά (π.χ μια ώρα, μια ημέρα, μια βδομάδα κ.τ.λ.), λέμε ότι η χρονοσειρά είναι διακριτού χρόνου και τη συμβολίζουμε με X_t, με $T \in \mathbb{Z}$. Για παράδειγμα οι ημερήσιες πωλήσεις ενός καταστήματος είναι μια χρονοσειρά διακριτού χρόνου. Παρ' ολ' αυτά αξίζει να σημειωθεί πως σε πολλά προβλήματα διακριτών χρονοσειρών (π.χ. ημερήσιες τιμές μια μετοχής), ο χρόνος δειγματοληψίας μπορεί να μην είναι σταθερός. Στη περίπτωση αυτή θεωρούμε ως χρόνο αναφοράς εκείνον που εκφράζει καλύτερα τα δεδομένα και όχι τον φυσικό χρόνο της δειγματοληψίας. Πιο συγκεκριμένα, στο προηγούμενο παράδειγμα των ημερήσιων πωλήσεων ενός καταστήματος, θα πρέπει να λά-

βουμε υπόψιν πως το κατάστημα παραμένει κλειστό τις αργίες και τις Κυριακές. Έτσι ο βηματισμός της χρονοσειράς θα πρέπει να είναι τέτοιος ώστε από την μέτρηση που αφορά τις πωλήσεις του Σαββάτου να μεταβαίνουμε στη μέτρηση για τις πωλήσεις της Δευτέρας. Αυτό το πετυχαίνουμε με το να ορίζουμε ως χρόνο αναφοράς τις ημέρες που παραμένει ανοιχτό το κατάστημα (και όχι δηλαδή όλες τις μέρες της εβδομάδας) και έτσι μπορούμε να θεωρήσουμε πως το βήμα είναι σταθερό ακόμη και για τις μετρήσεις από Σάββατο σε Δευτέρα.

Ορισμένες φορές οι διακριτές χρονοσειρές χωρίζονται περεταίρω σε στιγμιαία καταγεγραμμένες (*instantaneously recorded*) και σε συσσωρευμένες (*accumulated*). Όταν η δειγματοληψία για το φαινόμενο που εξετάζουμε γίνεται σε διακριτό χρόνο παρότι θα μπορούσε να γίνεται σε συνεχή (π.χ. θερμοκρασία) η χρονοσειρά θεωρείται μια στιγμιαία καταγεγραμμένη διακριτή χρονοσειρά. Όταν το φαινόμενο που εξετάζουμε είναι δύσκολο ή και αδύνατο να μετρηθεί για κάθε χρονική στιγμή, επειδή προέρχεται από το συγκεντρωτικό άθροισμα μιας τιμής (π.χ. εκπομπές διοξειδίων του άνθρακα σε μια περιοχή, βροχόπτωση, αριθμός γεννήσεων) τότε η χρονοσειρά ονομάζεται συσσωρευμένη διακριτή (Box *et al*., 2008; Granger & Newbold, 1986).

2.3 Συνιστώσες Χρονοσειρών

Τα δομικά στοιχεία μιας χρονοσειράς, βάσει των οποίων μεταβάλλεται και εξελίσσεται η τιμή του εξεταζόμενου φαινομένου στο χρόνο, είναι η τάση (*trend*), η εποχικότητα (*seasonality*), οι κυκλικές διακυμάνσεις (*cyclical variations*), και οι τυχαίες ή ακανόνιστες διακυμάνσεις (*random*/*irregular variations*) (Dodge, 2008). Στη συνέχεια γίνεται μια πιο λεπτομερή αναφορά στο καθένα από αυτά.

2.3.1 Τάση

Η τάση περιγράφει τη γενικότερη κατεύθυνση που ακολουθεί μια χρονοσειρά κατά τη διάρκεια μιας μεγάλης χρονικής περιόδου και χαρακτηρίζεται ως ανοδική ή καθοδική ανάλογα με το αν η κίνηση των τιμών της χρονοσειράς αυξάνεται ή μειώνεται. Όταν αναφερόμαστε στο σύνολο της χρονοσειράς η τάση ονομάζεται καθολική (*global trend*), ενώ για τα επιμέρους χρονικά διαστήματα της χρονοσειράς η τάση ονομάζεται τοπική (*local trend*). Σε μια χρονοσειρά μπορεί να υπάρχουν χρονικά διαστήματα και ανοδικής και καθοδικής τοπικής τάσης σε αντίθεση με την καθολική τάση η οποία (αν υπάρχει) είναι μόνο ανοδική ή μόνο

καθοδική. Γι' αυτόν το λόγο ο αριθμός των παρατηρήσεων θα πρέπει να είναι επαρκής προκειμένου να προσδιορίσουμε με ασφάλεια την τάση της χρονοσειράς. Τέλος, αν οι παρατηρήσεις της χρονοσειράς φαίνεται να απεικονίζονται ως προς τον χρόνο γύρω από μία γραμμή η τάση ονομάζεται γραμμική, ενώ στην αντίθετη περίπτωση ονομάζεται μη γραμμική.

2.3.2 Εποχικότητα

Εποχικότητα εμφανίζεται όταν υπάρχει ένα μοτίβο στη κίνηση των τιμών της χρονοσειράς το οποίο επαναλαμβάνεται περιοδικά ανά συγκεκριμένα χρονικά διαστήματα, τα οποία συνήθως είναι μικρότερα του έτους. Το μοτίβο επαναλαμβάνεται σε ολόκληρη τη χρονοσειρά, επομένως μπορούμε να προβλέψουμε το πως αυτό θα διαμορφωθεί στο μέλλον εξετάζοντας τα ιστορικά δεδομένα. Ο λόγος εμφάνισης εποχικότητας σε μια χρονοσειρά περιλαμβάνει παράγοντες όπως είναι ο καιρός, οι περίοδοι των διακοπών και άλλα γεγονότα τα οποία επαναλαμβάνονται μέσα στο έτος.

2.3.3 Κυκλικές Διακυμάνσεις

Μερικές χρονοσειρές όπως για παράδειγμα ο δείκτης ανεργίας, εμφανίζουν διακυμάνσεις οι οποίες, σε αντίθεση με την εποχικότητα, δεν έχουν σταθερή συχνότητα και δεν σχετίζονται με κάποιο γνώρισμα του ημερολογιακού έτους. Επιπλέον, η χρονική διάρκεια του επαναλαμβανόμενου μοτίβου συνήθως έχει διάρκεια μεγαλύτερη των 2 ετών (Athanasopoulos & Hyndman, 2018). Για παράδειγμα ορισμένες φορές θεωρείται πως τα οικονομικά δεδομένα επηρεάζονται από επιχειρησιακούς κύκλους (*business cycles*) με περίοδο από τρία έως και δέκα χρόνια ανάλογα με την υπό εξέταση μεταβλητή (Chatfield, 2003).

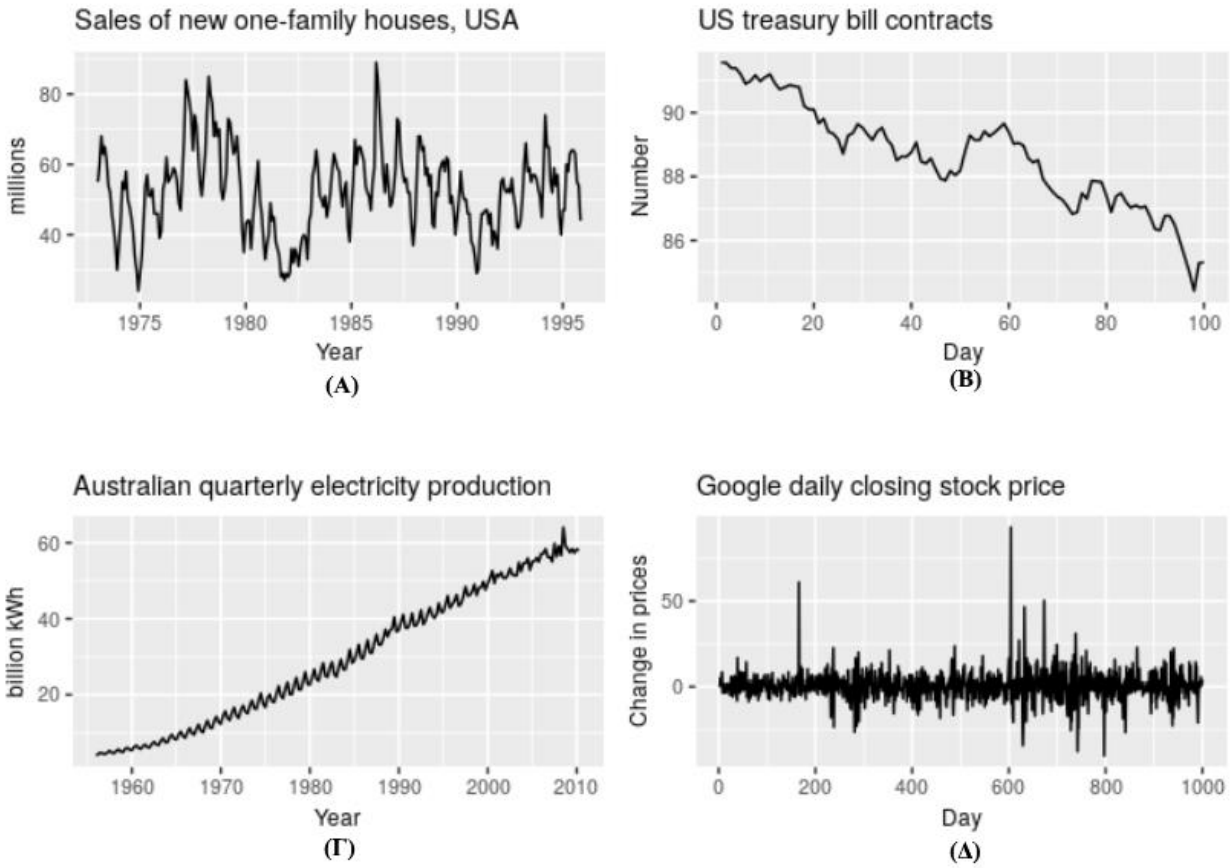
2.3.4 Ακανόνιστες Διακυμάνσεις

Εμφανίζονται σχεδόν σε όλες τις χρονοσειρές και σχετίζονται με τις τυχαίες και απρόβλεπτες διακυμάνσεις μιας χρονοσειράς οι οποίες δεν μπορούν να εξηγηθούν από τις υπόλοιπες συνιστώσες.

Στο παρακάτω Σχήμα 2.1 (Athanasopoulos & Hyndman, 2018) δίνονται μερικά παραδείγματα στα οποία μπορούμε να διακρίνουμε την ύπαρξη των παραπάνω τεσσάρων συνιστωσών.

ΣΧΗΜΑ 2.1

Παραδείγματα Χρονοσειρών



Πιο συγκεκριμένα, στο Σχήμα 2.1- Α φαίνεται να υπάρχει αρκετά έντονα το φαινόμενο της εποχικότητας για κάθε έτος που περνάει, παράλληλα διακρίνουμε κυκλική διακύμανση με περίοδο δέκα περίπου έτη ενώ στο σύνολο της χρονοσειράς δεν φαίνεται να υπάρχει (καθολική) τάση. Στο Σχήμα 2.1 – Β μπορούμε να δούμε μια χρονοσειρά με καθοδική τάση και χωρίς εποχικότητα, ενώ η χρονοσειρά του Σχήματος 2.1 – Γ έχει ανοδική τάση και εποχικότητα. Και στα δύο αυτά σχήματα δεν υπάρχει κυκλική διακύμανση. Τέλος, τα δεδομένα τους Σχήματος 2.1 – Δ δεν εμφανίζουν τάση εποχικότητα ή κυκλικές διακυμάνσεις. Σημειώνουμε πως, όπως είδαμε και στα παραπάνω παραδείγματα, δεν είναι απαραίτητο να εμφανίζονται και οι τέσσερις συνιστώσες μαζί σε μια χρονοσειρά.

2.3.5 Διάσπαση Χρονοσειρών

Μία τεχνική που μας επιτρέπει να απομονώσουμε τη κάθε συνιστώσα μιας χρονοσειράς και να αναγνωρίσουμε την ύπαρξη της ή όχι σε αυτήν, είναι η τεχνική της διάσπασης χρονοσειρών (*time series decomposition*). Με τον διαχωρισμό μιας χρονοσειράς στα επιμέρους δομικά της στοιχεία αποκτάμε μια καλύτερη εικόνα για το τρόπο με τον οποίο συμπεριφέρονται τα δεδομένα και το μοτίβο το οποίο ακολουθούν, με αποτέλεσμα να μπορούμε να επιλέξουμε και να εφαρμόσουμε τις μεθόδους πρόβλεψής μιας χρονοσειράς με μεγαλύτερη αποτελεσματικότητα.

Έτσι για μια χρονοσειρά $(X_t, t \in T)$ υποθέτουμε ότι οι τιμές που παράγονται από αυτήν καθορίζονται από την σχέση των τεσσάρων συνιστωσών της, δηλαδή της τάσης T_t, της εποχικότητας S_t των κυκλικών και των ακανόνιστων διακυμάνσεων C_t και I_t . Ανάλογα με το είδος αυτής της σχέσης, διακρίνουμε τα παρακάτω τρία υποδείγματα:

- **Προσθετικό Μοντέλο**

$$X_t = T_t + S_t + C_t + I_t$$

Στο προσθετικό μοντέλο οι τέσσερις συνιστώσες θεωρούνται ανεξάρτητες η μία από την άλλη, έτσι το μοντέλο προϋποθέτει πως η εποχικότητα δεν επηρεάζεται από την τάση και παραμένει σταθερή σε σχέση με εκείνη (Dodge, 2008). Για παράδειγμα, αν σε ένα κατάστημα οι πωλήσεις που παρατηρούνται τα Χριστούγεννα είναι σταθερά περισσότερες κατά ένα συγκεκριμένο αριθμό κάθε χρόνο, θα ήταν κατάλληλη η χρήση του προσθετικού μοντέλου.

- **Πολλαπλασιαστικό Μοντέλο**

$$X_t = T_t \cdot S_t \cdot C_t \cdot I_t$$

Όταν η ένταση της εποχικότητας είναι σχεδόν ανάλογη με την τάση της χρονοσειράς τότε κατάλληλο μοντέλο είναι το πολλαπλασιαστικό. Στη πράξη συνήθως υπάρχει εξάρτηση μεταξύ των συνιστωσών (με εξαίρεση κάποια φυσικά φαινόμενα) και γι' αυτό το πολλαπλασιαστικό μοντέλο προτιμάται έναντι του προσθετικού. Επιπλέον μπορούμε να μεταβούμε από ένα πολλαπλασιαστικό μοντέλο στο προσθετικό με τη βοήθεια των λογαρίθμων, έτσι ώστε:

$$\ln(X_t) = \ln(T_t) + \ln(S_t) + \ln(C_t) + \ln(I_t)$$

- **Μεικτό Μοντέλο**

$$X_t = S_t + (T_t \cdot C_t \cdot I_t)$$

ή

$$X_t = C_t + (T_t \cdot S_t \cdot I_t)$$

Το είδος αυτού του μοντέλου χρησιμοποιείται σπάνια (Dodge, 2008).

Αξίζει να σημειωθεί πως συνηθίζεται, για χρονοσειρές που παρατηρούνται σε μικρά χρονικά διαστήματα, να συμπεριλαμβάνουμε τη κυκλικότητα στην τάση και να γίνεται η εκτίμηση τους, από κοινού, με τη χρήση μίας μόνο συνιστώσας. Έτσι, για παράδειγμα το πολλαπλασιαστικό μοντέλο θα παίρνει την μορφή:

$$X_t = M_t \cdot S_t \cdot I_t$$

όπου M_t η συνιστώσα τάσης-κυκλικότητας (*trend-cycle component*) (Chatfield, 2003; Dozie *et al.*, 2020)

2.4 Στασιμότητα

Οι παρατηρήσεις μιας χρονοσειράς είναι μια ακολουθία τυχαίων μεταβλητών, η εξάρτηση των οποίων καθορίζεται επακριβώς από την από κοινού συνάρτηση κατανομής οποιουδήποτε πεπερασμένου υποσυνόλου k τυχαίων μεταβλητών της χρονοσειράς $(X_t, t \in T)$, δηλαδή:

$$F(x_1, x_2, \dots, x_k) = P\big(X_{t_1} < x_1, X_{t_2} < x_2, \dots, X_{t_k} < x_k\big)$$

για κάθε $k > 0$. Επειδή όμως στις περισσότερες περιπτώσεις η παραπάνω από κοινού συνάρτηση κατανομής είναι δύσκολο να προσδιοριστεί, και ακόμη και αν είναι γνωστή ο χειρισμός της είναι τεχνικά δύσκολος, συνήθως ορίζουμε απλούστερα την στοχαστική διαδικασία βασιζόμενοι στις δύο πρώτες ροπές των X_t και τις συνδιακυμάνσεις μεταξύ οποιωνδήποτε ζευγών X_t, X_s δηλαδή για κάθε χρονική στιγμή $t \in T$ χρησιμοποιούμε τους μέσους όρους $\mu_X(t) = E(X_t)$, τις διακυμάνσεις $\sigma_X^2(t) = Var(X_t)$ και τις αυτοσυνδιακυμάνσεις:

$$\gamma_X(s,u) = Cov(X_s, X_u) = E\left(\big(X_s - \mu_X(s)\big)\big(X_u - \mu_X(u)\big)\right) =$$
$$= E(X_s X_u) - \mu_X(s)\mu_X(u), \quad s,u \in T$$

Αν θεωρήσουμε ότι η στοχαστική διαδικασία είναι κανονική, δηλαδή αν $(X_{t_1}, X_{t_2}, \dots, X_{t_N})$ είναι μια πολυδιάστατη κανονική κατανομή για κάθε σύνολο $t_1, t_2, \dots, t_N$ και κάθε πεπερασμένο ακέραιο N, τότε οι προηγούμενες ποσότητες θα αρκούσαν για να προσδιοριστούν πλήρως οι ιδιότητες της στοχαστικής διαδικασίας (Shumway & Stoffer, 2014). Αν δεν μπορούμε να εξασφαλίσουμε την κανονικότητα της στοχαστικής διαδικασίας και αντ' αυτού θεωρήσουμε ότι η διαδικασία είναι γραμμική, δηλαδή ότι οι μεταβλητές X_t παράγονται από τον γραμμικό συνδυασμό των προηγούμενων X_t και των τιμών, από το παρόν και το παρελθόν, που προέρχονται από άλλες διαδικασίες, τότε και πάλι οι κύριες, τουλάχιστον, ιδιότητες της διαδικασίας αποτυπώνονται στις μέσες τιμές και στις συνδιακυμάνσεις (*Granger & Newbold, 1986*).

Όπως είδαμε οι ροπές αλλάζουν κάθε χρονική στιγμή και δεν παραμένουν σταθερές, οπότε ένα πρόβλημα που δημιουργείται είναι η εκτίμηση τους. Στις περιπτώσεις που είναι εφικτό να μελετήσουμε πολλαπλές, m στο πλήθος, πραγματοποιήσεις της ίδιας στοχαστικής διαδικασίας, συμβολίζοντας με $X_{jt}, t = 1, \dots, n, j = 1, \dots, m$ τις διάφορες πραγματοποιήσεις της, για παράδειγμα, μια πιθανή εκτίμηση της μέσης τιμής θα ήταν ο μέσος της στατιστικής συλλογής (*ensemble average*):

$$\hat{\mu}_X(t) = \frac{1}{m}\sum_{j=1}^{m} X_{jt}$$

Παρ' ολ΄ αυτά τις περισσότερες φορές είναι αδύνατη η υλοποίηση πολλαπλών πραγματοποιήσεων της ίδιας στοχαστικής διαδικασίας και επομένως στη διάθεση μας έχουμε μόνο μια παρατήρηση για κάθε χρονική στιγμή. Για να προσπεράσουμε αυτό το πρόβλημα θα εισάγουμε δύο επιπλέον περιορισμούς με την υπόθεση της στασιμότητας (*stationarity*) και της εργοδικότητας (*ergodicity*) (*Granger & Newbold, 1986*).

Στη πρόβλεψη χρονοσειρών είναι σημαντικό να προσδιορίσουμε αν η χρονοσειρά είναι στάσιμη ή όχι, καθώς πολλές από τις μεθόδους πρόβλεψης προϋποθέτουν στασιμότητα. Μια στάσιμη χρονοσειρά απλοποιεί την στοχαστική διαδικασία κάνοντας ευκολότερη την μοντελοποίηση της και την παραγωγή προβλέψεων για αυτή και αντιθέτως η μοντελοποίηση μιας μη στάσιμης χρονοσειράς, λόγω της δυναμικής μεταβολής της στο χρόνο, είναι πολύ πιο περίπλοκη (Atwan, 2022). Πιο συγκεκριμένα, η στασιμότητα διακρίνεται σε δύο είδη, την ισχυρή ή αυστηρή στασιμότητα (*strong/strict stationarity*) και την ασθενή στασιμότητα (*weak stationarity*). Μια χρονοσειρά $(X_t, t \in T)$ είναι αυστηρά στάσιμη αν οι πολυδιάστατη κατανομή του τυχαίου δείγματος $(X_{t_1}, X_{t_2}, \dots, X_{t_N})$ είναι ίδια με αυτή του $(X_{t_1-k}, X_{t_2-k}, \dots, X_{t_N-k})$

για κάθε $t_1 - k, t_2 - k, \dots, t_N - k \in T$ και $t_1, t_2, \dots, t_N \in T$, $k \geq 1$. Με άλλα λόγια οι πραγματοποιήσεις της χρονοσειράς εξαρτώνται μόνο από τις αρχικές τιμές τους και όχι από τον χρόνο που ξεκινούν (Boutsikas, 2020; Grazzini 2012). Όμως, η αυστηρή στασιμότητα είναι δύσκολο να επαληθευτεί στη πράξη και γι' αυτό συνήθως χρησιμοποιείται η ασθενής στασιμότητα. Μια χρονοσειρά $(X_t, t \in T)$ καλείται ασθενώς στάσιμη αν έχει σταθερό μέσο όρο:

$$\mu_X(t) = \mu_X, \quad \text{για κάθε } t \in T$$

σταθερή πεπερασμένη διακύμανση:

$$\sigma_X^2 = \sigma_X^2(t) < \infty, \quad \text{για κάθε } t \in T$$

και οι αυτοσυνδιακυμάνσεις για δύο διαφορετικές χρονικές στιγμές εξαρτώνται μόνο από την μεταξύ τους απόσταση (και όχι από τον ίδιο τον χρόνο):

$$\gamma_X(k) = \mathrm{Cov}(X_t, X_{t-k}) = \mathrm{Cov}(X_t, X_{t+k}), \quad \text{για κάθε } t, t+k \in T$$

Επίσης, από τα παραπάνω παρατηρούμε πως για $k = 0$ προκύπτει

$$\gamma_X(0) = \mathrm{Cov}(X_t, X_t) = E[(X_t - \mu_X)^2] = \mathrm{Var}(X_t) = \sigma_X^2$$

Αξίζει να σημειωθεί πως οι περισσότερες χρονοσειρές είναι μη στάσιμες λόγω της τάσης, της εποχικότητας και της κυκλικής διακύμανσης που εμφανίζουν. Γι' αυτό τον λόγο έχουν προταθεί διάφορες τεχνικές εξάλειψης των συνιστωσών αυτών προκειμένου να μετατραπεί η χρονοσειρά σε στάσιμη. Μία από αυτές τις μεθόδους είναι η μέθοδος των διαφορών κατά την οποία εφαρμόζουμε επανειλημμένα πρώτες διαφορές $Y_t = \nabla X_t = X_t - X_{t-1}$ στη χρονοσειρά μέχρις ότου αυτή να συμπεριφέρεται ως στάσιμη.

Όπως αναφέρθηκε νωρίτερα, η δεύτερη υπόθεση για να εκτιμηθούν οι πληθυσμιακές τιμές των ροπών από τις δειγματικές ροπές μιας μόνο πραγματοποίησης της στοχαστικής διαδικασίας είναι η εργοδικότητα. Εργοδικότητα πρακτικά σημαίνει πως η μέση τιμή σε σχέση με τον χρόνο (*time average*) είναι ανεξάρτητη του αρχικού σημείου της χρονοσειράς (Cowpertwait & Metcalfe, 2009) και επομένως καθώς αυτή εξελίσσεται στον χρόνο τροφοδοτεί συνεχώς την μέση τιμή με νέες και χρήσιμες πληροφορίες (*Granger & Newbold, 1986*). Έτσι η μέση τιμή σε σχέση με τον χρόνο:

$$\overline{X}_t = \frac{1}{T}\sum_{t=1}^{T} X_t$$

θα είναι ένας αμερόληπτος (*unbiased*) και συνεπής (*consistent*) εκτιμητής της πληθυσμιακής μέσης τιμής μ, ώστε $Var(\bar{X}_t) \to 0$ και $E(\bar{X}_t) \to \mu$ καθώς $T \to \infty$. Ο έλεγχος της εργοδικότητας με μία μόνο πραγματοποίηση της διαδικασίας είναι αδύνατος (αφού δεν θα έχουμε στη διάθεση μας τον μέσο της στατιστικής συλλογής για να ελέγξουμε αν είναι ίσος με τον δειγ-

ματικό μέσο). Μια όμως αναγκαία συνθήκη (αλλά όχι ικανή) για την εργοδικότητα είναι: $Cov(X_t, X_{t-k}) \to 0$ με έναν αρκετά γρήγορο ρυθμό, καθώς το k αυξάνεται (*Granger & Newbold, 1986*).

2.5 Συσχέτιση στις Χρονοσειρές

Η στασιμότητα, η τάση όπως και οι υπόλοιπες συνιστώσες μιας χρονοσειράς είναι χαρακτηριστικά τα οποία πρέπει μελετηθούν πριν προχωρήσουμε σε περεταίρω ανάλυση της. Ένα ακόμη τέτοιο χαρακτηριστικό είναι η αυτοσυσχέτιση (*autocorrelation*). Πολλές από τις μεθόδους πρόβλεψης των χρονοσειρών προϋποθέτουν να μην υπάρχει αυτοσυσχέτιση στα δεδομένα και επομένως η μελέτη της είναι ιδιαίτερα σημαντική.

Γενικότερα η συσχέτιση δύο τυχαίων μεταβλητών Y και X μετράει τον βαθμό της γραμμικής τους συσχέτισης και δίνεται από τον τύπο:

$$\rho_{Y,X} = \frac{Cov(Y,X)}{\sqrt{Var(Y)}\sqrt{Var(X)}} = \frac{\sum_{t=1}^{n}(Y_i - \mu_Y)(X_i - \mu_X)}{\sqrt{\sum_{t=1}^{n}(Y_i - \mu_Y)^2}\sqrt{\sum_{t=1}^{n}(X_i - \mu_X)^2}}$$

όπου $Cov(Y,X)$ η συνδιακύμανση των δύο μεταβλητών και μ_Y, μ_X οι μέσες τιμές τους, ενώ ισχύει ότι $-1 \leq \rho_{Y,X} \leq 1$.

2.5.1 Αυτοσυσχέτιση

Ειδική περίπτωση της συσχέτισης μπορεί να θεωρηθεί η αυτοσυσχέτιση, η οποία μετράει τον βαθμό της γραμμικής εξάρτησης μεταξύ τιμών της χρονοσειράς με χρονική υστέρηση (*lag*) k. Με άλλα λόγια υπολογίζει το κατά πόσο μια τιμή η οποία εμφανίζεται στον χρόνο t εξαρτάται από την τιμή που εμφανίστηκε σε παρελθοντικό χρόνο $t-k$ (*Granger & Newbold, 1986*). Πιο συγκεκριμένα, για μια ασθενώς στάσιμη χρονοσειρά X ο συντελεστής γραμμικής συσχέτισης μεταξύ της παρατήρησης X_t και X_{t-1} ονομάζεται k αυτοσυσχέτιση της X και δίνεται από τον τύπο:

$$\rho(k) = Corr(X_t, X_{t-k}) = \frac{Cov(X_t, X_{t-k})}{\sqrt{Var(X_t)}\sqrt{Var(X_{t-k})}}$$

όπως είδαμε στην προηγούμενη ενότητα $\gamma(0) = Var(X_t)$ και επιπλέον λόγω της στασιμότητας ισχύει $Var(X_t) = Var(X_{t-k})$. Οπότε:

$$\rho(k) = \frac{Cov(X_t, X_{t-k})}{\sqrt{Var(X_t)}\sqrt{Var(X_{t-k})}} = \frac{Cov(X_t, X_{t-k})}{\sqrt{Var(X_t)Var(X_t)}} = \frac{Cov(X_t, X_{t-k})}{Var(X_t)} = \frac{\gamma(k)}{\gamma(0)}$$

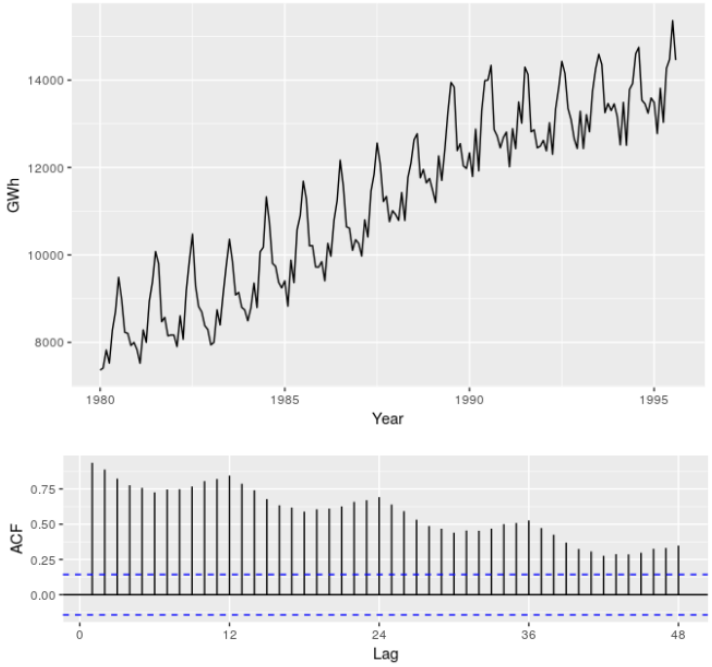
Τέλος, από την τελευταία σχέση παρατηρούμε ότι $\rho(k) = \rho(-k)$ και $\rho(0) = 1$.

Κατά την μελέτη μιας χρονοσειράς συνηθίζεται να κατασκευάζουμε τη συνάρτηση αυτοσυσχέτισης (*Autocorrelation Function (ACF)*) και να την απεικονίζουμε γραφικά. Η ACF υπολογίζει τις αυτοσυσχετίσεις της χρονοσειράς ως συνάρτηση της χρονικής υστέρησης k. Η γραφική της παράσταση ονομάζεται διάγραμμα αυτοσυσχετίσεων (correlogram) και απεικονίζει το γράφημα των τιμών $\rho(k)$ για $k = 0,1,2, \dots$. Όταν η αυτοσυσχέτιση είναι θετική για κάποια χρονική υστέρηση τότε υπάρχει ένδειξη ότι η χρονοσειρά παρουσιάζει την ίδια συμπεριφορά για αυτή τη τιμή του lag ενώ όταν η αυτοσυσχέτιση είναι αρνητική περιμένουμε η συμπεριφορά της χρονοσειράς σε εκείνο το lag να είναι η αντίθετη. Επιπλέον, για μία στάσιμη χρονοσειρά η συνάρτηση αυτοσυσχέτισης θα πρέπει φθίνει γρήγορα προς το μηδέν καθώς τα lag αυξάνονται (Athanasopoulos & Hyndman, 2018).

Στο Σχήμα 2.2 (Athanasopoulos & Hyndman, 2018) δίνεται η γραφική απεικόνιση της χρονοσειράς για την μηνιαία κατανάλωση ηλεκτρικού ρεύματος στην Αυστραλία για τα έτη 1980-1995. Στο ίδιο σχήμα επίσης βλέπουμε το διάγραμμα αυτοσυσχετίσεων για $k = 0,1,2, \dots ,48$. Παρατηρούμε ότι η εκτίμηση της συνάρτησης αυτοσυσχέτισης μειώνεται πολύ αργά καθώς τα lags αυξάνονται, λόγω της τάσης και κατ' επέκταση της μη-στασιμότητας που εμφανίζει η χρονοσειρά, ενώ η κυματοειδής μορφή του σχήματος οφείλεται στην εποχικότητα της χρονοσειράς.

ΣΧΗΜΑ 2.2

Απεικόνιση Χρονοσειράς (πάνω) και του Διαγράμματος Αυτοσυσχετίσεών της (κάτω)



Στη συνέχεια αναφέρονται δύο παραδείγματα χρονοσειρών με μηδενική αυτοσυσχέτιση. Η μία είναι στάσιμη και ονομάζεται λευκός θόρυβος (*White Noise*) ενώ η άλλη είναι μη στάσιμη και ονομάζεται τυχαίος περίπατος (*Random Walk*).

2.5.2 Λευκός Θόρυβος

Λευκός θόρυβος καλείται μια χρονοσειρά $W = \{W_t, t \in T\}$ που αποτελείται από ασυσχέτιστες τυχαίες μεταβλητές, δηλαδή:

$$\gamma(k) = Cov(W_t, W_{t-k}) = \begin{cases} \sigma^2, & k = 0 \\ 0, & k \geq 1 \end{cases}$$

και κατ' επέκταση:

$$\rho(k) = Corr(W_t, W_{t-k}) = \begin{cases} 1, & k = 0 \\ 0, & k \geq 1 \end{cases}$$

με μηδενική μέση τιμή $E(W_t) = 0$ και σταθερή διασπορά σ^2. Τη συμβολίζουμε $WN(0, \sigma^2)$. και αν επιπλέον τα στοιχεία της W ακολουθούν την κανονική κατανομή, τότε η χρονοσειρά καλείται Γκαουσιανός λευκός θόρυβος (*Gaussian White Noise* – $GWN(0, \sigma^2)$).

2.5.3 Τυχαίος Περίπατος

Ο τυχαίος περίπατος είναι μία μη στάσιμη χρονοσειρά όπου κάθε στοιχείο της X_t προκύπτει από το προηγούμενο της X_{t-1} όταν σε αυτό προστεθεί η τιμή μιας τυχαίας μεταβλητής W_t. Πιο συγκεκριμένα, η στοχαστική ανέλιξη $X = (X_t, t \in T)$ καλείται τυχαίος περίπατος σε διακριτό χρόνο αν:

$$X_t = X_{t-1} + W_t, \qquad t = 1,2, \dots$$

όπου $W = (W_t, t \in T)$ μια χρονοσειρά λευκού θορύβου $W \sim WN(0, \sigma^2)$. Επιπλέον αν $X_0 = 0$ είναι η αρχική τιμή του περιπάτου τη χρονική στιγμή $t = 0$ εφαρμόζοντας επαναληπτικά την παραπάνω σχέση προκύπτει ότι:

$$X_t = \sum_{i=1}^{t} W_i, \qquad t \in T = \{1,2, \dots\}$$

Έτσι η μέση τιμή του τυχαίου περιπάτου είναι $E(X_t) = 0$, η διακύμανση $Var(X_t) = t\sigma^2 < \infty$ και για $k \geq 0$ είναι:

$\gamma(k) = Cov(X_{t+k}, X_t) = Cov(X_t + W_{t+1} + \cdots + W_{t+k}, X_t) = Cov(X_t, X_t) = Var(X_t) = t\sigma^2$

και οπότε, αφού το $\gamma(k)$ εξαρτάται από τον χρόνο t (επειδή η διακύμανση αυξάνεται με τον χρόνο), αποδείξαμε ότι πράγματι η χρονοσειρά X_t δεν είναι στάσιμη (Brockwell & Davis 2010).

2.6 Μέθοδοι Πρόβλεψης Χρονοσειρών

2.6.1 Απλές Μέθοδοι Πρόβλεψης

Παρακάτω θα δούμε μερικές μεθόδους οι οποίες, αν και εξαιρετικά απλές στον υπολογισμό τους, είναι αρκετά αποτελεσματικές. Πολλές φορές χρησιμοποιούνται ως σημείο αναφοράς για τις υπόλοιπες μεθόδους πρόβλεψής, καθώς για να θεωρηθεί μία μέθοδος αποτελεσματική θα πρέπει τα αποτελέσματα της να είναι ακριβέστερα από αυτά των παρακάτω μεθόδων (Athanasopoulos & Hyndman, 2018).

- **Απλοϊκή (Naive) Μέθοδος**

  Αποτελεί την πιο απλή μέθοδο πρόβλεψης αφού θεωρούμε πως οι τιμές όλων των επόμενων προβλέψεων θα είναι ίσες με την τελευταία παρατήρηση, δηλαδή:

  $$\hat{X}_{T+h} = X_T$$

  όπου h ο ορίζοντας πρόβλεψης (*forecast horizon*), που δηλώνει για πόσες χρονικές περιόδους στο μέλλον γίνεται η πρόβλεψή μας. Είναι περισσότερο αποδοτική στις χρηματοοικονομικές χρονοσειρές και για ορίζοντα πρόβλεψης μιας περιόδου, αφού, για παράδειγμα, η τιμή κλεισίματος μιας μετοχής συνήθως δεν διαφέρει σημαντικά από εκείνη της προηγούμενης χρηματιστηριακής ημέρας.

- **Μέθοδος Απλού Μέσου**

  Με αυτή τη μέθοδο θεωρούμε πως όλες οι μελλοντικές τιμές είναι ίσες με τον μέσο όρο των ιστορικών δεδομένων, πλήθους T, που βρίσκονται στη διάθεση μας:

  $$\hat{X}_{T+h} = \frac{1}{T}\sum_{i=1}^{T} X_i$$

- **Μέθοδος Περιπλάνησης**

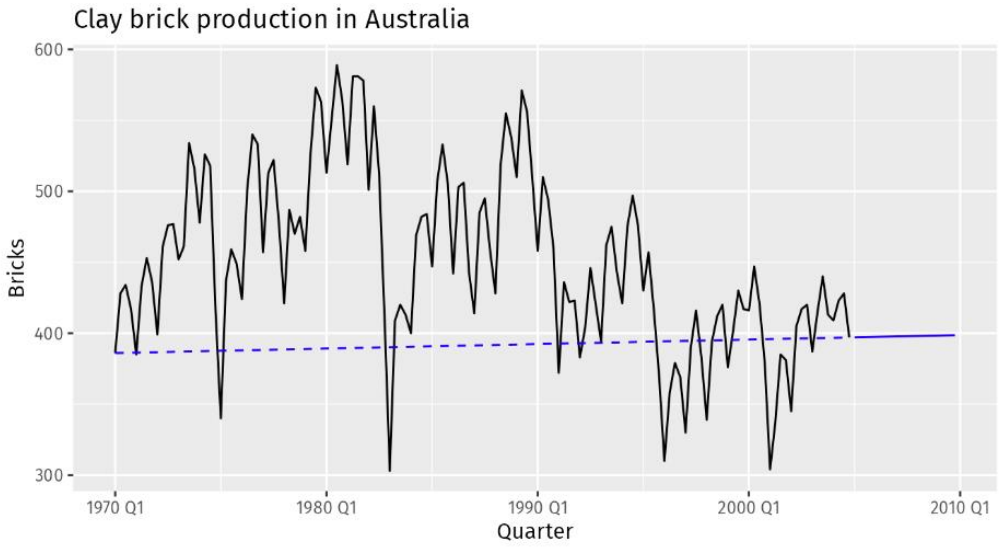
  Αποτελεί μία παραλλαγή της μεθόδου Naive, όπου επιτρέπει στις προβλέψεις να αυξάνονται ή να μειώνονται με την πάροδο του χρόνου. Το ποσό της μεταβολής ονομάζεται περιπλάνηση (*drift*) και ορίζεται ως η μέση αλλαγή που παρατηρείται στα ιστορικά δεδομένα. Ο υπολογισμός της πρόβλεψης για την χρονική στιγμή $T + h$ δίνεται από την σχέση:

$$\hat{X}_{T+h} = X_T + \frac{h}{T-1}\sum_{t=2}^{T}(X_t - X_{t-1}) = X_T + h\left(\frac{X_T - X_1}{T-1}\right)$$

  Πρακτικά, όπως φαίνεται και στο Σχήμα 2.3 (Athanasopoulos & Hyndman, 2018), αυτό είναι ισοδύναμο με το να προεκτείνουμε την ευθεία γραμμή που συνδέει την πρώτη και την τελευταία παρατήρηση του δείγματός μας.

ΣΧΗΜΑ 2.3

Απεικόνιση των Προβλέψεων της Μεθόδου Περιπλάνησης



2.6.2 Πρόβλεψή με Μεθόδους Εξομάλυνσης

Η εξομάλυνση έχει ως στόχο τη εξάλειψη των ακανόνιστων διακυμάνσεων μεταξύ των χρονικών βημάτων μιας χρονοσειράς, προκειμένου να τονιστεί η ύπαρξη των υπολοίπων

στοιχείων της που αναφέρθηκαν στην Ενότητα 2.3. Έτσι τα δεδομένα κατά μία έννοια φιλτράρονται, ώστε να φανερωθεί σε αυτά με μεγαλύτερη ακρίβεια η υποκείμενη διαδικασία σύμφωνα με την οποία παράγονται τα στοιχεία της χρονοσειράς (Athanasopoulos & Hyndman, 2018). Παρ' όλα αυτά, οι τεχνικές αυτές εκτός από την προεπεξεργασία των δεδομένων μπορούν να χρησιμοποιηθούν και απευθείας για την παραγωγή προβλέψεων, συνήθως σε περιπτώσεις όπου δεν υπάρχει τάση (και κυκλικότητα) και εποχικότητα στα δεδομένα (Brownlee, 2020). Στη συνέχεια παρουσιάζονται μερικές από αυτές τις μεθόδους.

- **Μέθοδος Κινητού Μέσου (Moving Average)**

  Σε αντίθεση με την μέθοδο του απλού μέσου όπου η πρόβλεψή για όλες τις μελλοντικές παρατηρήσεις είναι μία και υπολογίζεται από τη μέση τιμή του συνόλου των δεδομένων, η μέθοδος κινητού μέσου χρησιμοποιεί τις n πιο πρόσφατες τιμές που είναι διαθέσιμες την στιγμή t, υπολογίζει την μέση τους τιμή και αυτή είναι η πρόβλεψη για την επόμενη περίοδο $t+1$. Ονομάζεται *κινητός* μέσος διότι για κάθε νέα παρατήρηση που εισέρχεται στο δείγμα το "παράθυρο" μήκους n μετακινείται μία θέση και έτσι για τον υπολογισμό της πρόβλεψης της νέας περιόδου η παλαιότερη παρατήρηση (που χρησιμοποιήθηκε στην πρόβλεψη της προηγούμενης περιόδου) καταργείται και στη θέση της χρησιμοποιείται η νέα παρατήρηση που προστέθηκε (Brownlee, 2020). Δηλαδή:

  $$\hat{X}_{T+1} = \frac{1}{n} \sum_{i=T-n+1}^{T} X_i$$

  Εναλλακτικά, υπογραμμίζοντας τη σχέση της πρόβλεψης $\hat{X}_{T+1}$ με την προηγούμενη της $\hat{X}_T$, ο τύπος μπορεί να γραφεί:

  $$\hat{X}_{T+1} = \frac{1}{n} \sum_{i=T-n}^{T} X_i + \frac{1}{n}(X_T - X_{T-n}) = \hat{X}_T + \frac{1}{n}(X_T - X_{T-n})$$

  όπου X_{T-n}, X_T είναι τιμή της χρονοσειράς που αφαιρείται και προστίθεται αντίστοιχα, κατά τον επανυπολογισμό της μέσης τιμής για τον προσδιορισμό της πρόβλεψης $\hat{X}_{T+1}$.

- **Μέθοδος Απλής Εκθετικής Εξομάλυνσης (*Simple Exponential Smoothing (SES)*)**

  Όπως είδαμε, η μέθοδος Naive θεωρούσε πως η πιο πρόσφατη παρατήρηση ήταν η σημαντικότερη για την παραγωγή της πρόβλεψης και δεν λάμβανε υπόψη την πληρο-

φορία που υπήρχε στις προηγούμενες παρατηρήσεις. Από την άλλη, η μέθοδος του απλού μέσου θεωρούσε πως όλες οι παρατηρήσεις ήταν ισάξιες, αναθέτοντάς τους ίσα βάρη κατά τον υπολογισμό της πρόβλεψης. Συνήθως όμως είναι πιο λογικό να υπάρχει μια ισορροπία μεταξύ αυτών των δύο μεθόδων, έτσι ώστε οι πιο πρόσφατες παρατηρήσεις να έχουν μεγαλύτερη βαρύτητα από εκείνες που βρίσκονται μακριά στο παρελθόν. Η ιδέα αυτή υλοποιείται με τη μέθοδο της απλής εκθετικής εξομάλυνσης. Όπως και η μέθοδος του κινητού μέσου, προϋποθέτει πως η χρονοσειρά είναι στάσιμη, αλλά σε αντίθεση με εκείνη κάνει χρήση όλων των παρατηρήσεων που είναι διαθέσιμες σε κάθε χρονική στιγμή. Πιο συγκεκριμένα, οι προβλέψεις υπολογίζονται με σταθμισμένους μέσους όρους, όπου τα βάρη μειώνονται εκθετικά όσο οι παρατηρήσεις με τις οποίες σχετίζονται βρίσκονται μακρύτερα στο παρελθόν. Έτσι, η πρόβλεψη για την χρονική στιγμή $T+1$ αξιοποιεί ολόκληρη την χρονοσειρά $X_1, \dots, X_T$ και η τιμή της υπολογίζεται σύμφωνα με το παρακάτω σταθμισμένο μέσο:

$$\hat{X}_{T+1} = aX_T + a(1-a)X_{T-1} + a(1-a)^2X_{T-2} + \cdots$$

όπου $0 \leq a \leq 1$ η παράμετρος εξομάλυνσης η οποία καθορίζει τον ρυθμό με τον οποίο μειώνονται τα βάρη. Όπως είναι φανερό και από τον τύπο, όσο πιο κοντά στο 0 βρίσκεται η τιμή του a τόσο μεγαλύτερο και το βάρος που δίνεται σε παλαιότερες παρατηρήσεις, ενώ για τιμές κοντά στο 1 όλο και μεγαλύτερη η βαρύτητα που δίνεται στις πιο πρόσφατες τιμές της χρονοσειράς. Επίσης, παρατηρούμε πως για $a = 1$ έχουμε ότι $\hat{X}_{T+1} = X_T$, όποτε οι προβλέψεις είναι ίσες με αυτές του μοντέλου Naive.

Εναλλακτικά, υπογραμμίζοντας τη σχέση της πρόβλεψης $\hat{X}_{T+1}$ με την προηγούμενη της $\hat{X}_T$, αποδεικνύεται (Brownlee, 2020; Chatfield, 2003) ότι ο τύπος μπορεί να γραφεί:

$$\hat{X}_{T+1} = aX_T + (1-\alpha)\hat{X}_T$$

- **Μέθοδος Εκθετικής Εξομάλυνσης του Holt (*Holt's Exponential Smoothing*)**

  Η μέθοδος εκθετικής εξομάλυνσης του Holt (ή αλλιώς διπλή εκθετική εξομάλυνση (*double exponential smoothing*)) αποτελεί επέκταση της SES και χρησιμοποιείται όταν υπάρχει τάση στα δεδομένα της χρονοσειράς. Το μοντέλο αποτελείται από την εξίσωση πρόβλεψης και δύο εξισώσεις εξομάλυνσης, μία για το επίπεδο (*level*, δηλαδή τη μέση τιμή της χρονοσειράς σε μια δεδομένη χρονική περίοδο) και μία για την τάση.

  Εξίσωση Πρόβλεψης: $\hat{X}_{T+h} = l_t + hb_t$

Εξίσωση Επιπέδου: $l_t = aX_t + (1-a)(l_{t-1} + b_{t-1})$

Εξίσωση Τάσης : $b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$

όπου l_t, b_t η εκτίμηση του επιπέδου και της τάσης της χρονοσειράς αντιστοίχως, για τη χρονική στιγμή t, ενώ $0 \leq a \leq 1$ και $0 \leq \beta \leq 1$ οι παράμετροι εξομάλυνσης για το επίπεδο και τη τάση. Από την εξίσωση επιπέδου βλέπουμε πως το επίπεδο τη χρονική στιγμή t είναι το σταθμισμένο άθροισμα της παρατήρησης X_t και του επιπέδου της προηγούμενης χρονικής περιόδου l_{t-1} προσαρμοσμένο προς την τάση. Από την εξίσωση της τάσης παρατηρούμε πως το b_t είναι το σταθμισμένο άθροισμα της εκτίμηση της τάσης b_{t-1} τη προηγούμενη χρονική περίοδο και της πιο πρόσφατης μεταβολής του επιπέδου $l_t - l_{t-1}$ (Athanasopoulos & Hyndman, 2018; Shmueli & Lichtendahl Jr, 2016). Οι παράμετροι εξομάλυνσης a και β και οι αρχικές τιμές l_0 και b_0 συνήθως υπολογίζονται από τα παρατηρούμενα δεδομένα ελαχιστοποιώντας το άθροισμα των τετραγώνων των σφαλμάτων (Athanasopoulos & Hyndman, 2018):

$$SSE = \sum_{t=1}^{T} \left(X_t - \hat{X}_t\right)^2$$

- **Μέθοδος Εκθετικής Εξομάλυνσης των Holt-Winters (*Holt-Winters' Exponential Smoothing*)**

  Η μέθοδος εκθετικής εξομάλυνσης των Holt-Winter (ή αλλιώς τριπλή εκθετική εξομάλυνση (*triple exponential smoothing*)) είναι επέκταση της διπλής εκθετικής εξομάλυνσης η οποία, πέρα από την τάση, λαμβάνει υπόψη και την ύπαρξη εποχικότητας στα δεδομένα. Ανάλογα με το τον τρόπο δράσης της εποχικότητας υπάρχουν δύο παραλλαγές της μεθόδου. Η προσθετική μέθοδος προτιμάται όταν οι εποχιακές διακυμάνσεις είναι περίπου σταθερές σε όλη τη χρονοσειρά, ενώ η πολλαπλασιαστική χρησιμοποιείται όταν οι τιμές σε διαφορετικές εποχές διαφέρουν ποσοστιαία μεταξύ τους. Έτσι, στο μοντέλο προστίθεται μια επιπλέον εξίσωση εξομάλυνσης για τη συνιστώσα εποχικότητας s_t (Athanasopoulos & Hyndman, 2018; Shmueli & Lichtendahl Jr, 2016).

  Πιο συγκεκριμένα για την προσθετική μέθοδο έχουμε:

  Εξίσωση Πρόβλεψης: $\hat{X}_{T+h} = l_t + hb_t + s_{t+h-m(k+1)}$

  Εξίσωση Επιπέδου: $l_t = a(X_t - s_{t-m}) + (1-a)(l_{t-1} + b_{t-1})$

  Εξίσωση Τάσης : $b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$

$$\text{Εξίσωση Εποχικότητας : } s_t = \gamma(X_t - l_t) + (1-\gamma)s_{t-m}$$

όπου, m δηλώνει τη συχνότητα της εποχικότητας δηλαδή το πλήθος των εποχών σε ένα έτος, k το ακέραιο μέρος της της διαίρεσης $(h-1)/m$, το οποίο διασφαλίζει ότι οι εκτιμήσεις των εποχιακών δεικτών που χρησιμοποιούνται για την πρόβλεψη προέρχονται από το τελευταίο έτος του δείγματος και $0 \leq \gamma \leq 1$ η παράμετρος εξομάλυνσης για την εποχικότητα.

Από την άλλη, για την πολλαπλασιαστική μέθοδο έχουμε:

$$\text{Εξίσωση Πρόβλεψης: } \hat{X}_{T+h} = (l_t + hb_t)s_{t+h-m(k+1)}$$

$$\text{Εξίσωση Επιπέδου: } l_t = a\frac{X_t}{s_{t-m}} + (1-a)(l_{t-1} + b_{t-1})$$

$$\text{Εξίσωση Τάσης : } b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$$

$$\text{Εξίσωση Εποχικότητας : } s_t = \gamma\frac{X_\tau}{l_t} + (1-\gamma)s_{t-m}$$

- **Μοντέλο ARIMA**

Το ολοκληρωμένο αυτοπαλίνδρομο μοντέλο κινητού μέσου (AutoRegressive Integrated Moving Average) είναι μία από τις πιο αποτελεσματικές τεχνικές για την πρόβλεψη χρονοσειρών και σε αντίθεση με τα μοντέλα εκθετικής εξομάλυνσης που βασίζονται στην περιγραφή της τάσης και της εποχικότητας, τα μοντέλα ARIMA στοχεύουν στη περιγραφή της αυτοσυσχέτισης των δεδομένων (Athanasopoulos & Hyndman, 2018). Μπορεί να εφαρμοστεί σε χρονοσειρές με τάση καθώς με τη διαφόριση που πραγματοποιείται γίνεται προσπάθεια εξάλειψης της μη-στασιμότητας (όσον αφορά τη μέση τιμή). Πιο συγκεκριμένα ένα (μη-εποχικό) μοντέλο ARIMA το οποίο συνήθως συμβολίζεται $ARIMA(p,d,q)$ αποτελείται από τον συνδυασμό των παρακάτω τριών στοιχείων:

1. Αυτοπαλινδρόμιση ($AR(p)$)

   Είναι ένα είδος παλινδρόμησης όπου βασίζεται στην εξάρτηση μιας παρατήρησης από τις προηγούμενές της. Η παράμετρος p ονομάζεται τάξη υστέρησης (*lag order*) και καθορίζει τον αριθμό των παρατηρήσεων προηγούμενων βημάτων που θα συμπεριληφθούν στο μοντέλο. Έτσι το $AR(p)$ υπόδειγμα έχει την παρακάτω μορφή

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + W_t$$

όπου X στάσιμη χρονοσειρά, $\varphi_1, \varphi_2, \dots, \varphi_p \in \mathbb{R} \quad (\varphi_p \neq 0)$ και $W_t \sim GWN(0, \sigma^2)$. Η μέση τιμή μ, της X είναι μηδέν, σε διαφορετική περίπτωση αντικαθιστούμε το X_t με $X_t - \mu$ και έχουμε

$$X_t - \mu = \phi_1(X_{t-1} - \mu) + \phi_2(X_{t-2} - \mu) + \cdots + \phi_p(X_{t-p} - \mu) + W_t$$

ή

$$X_t = a + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + W_t$$

όπου $a = \mu(1 - \varphi_1 - \cdots - \varphi_p)$. Επίσης, με τη χρήση του συντελεστή μετατόπισης B, όπου $BX_t = X_{t-1}$, εναλλακτικά μπορούμε και γράφουμε ότι

$$\left(1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_p B^p\right) X_t = W_t$$

ή

$$\varphi(B) X_t = W_t$$

όπου, $\varphi(x) = 1 - \varphi_1 x - \phi_2 x^2 - \cdots - \phi_p x^p$ είναι το χαρακτηριστικό πολυώνυμο του υποδείγματος $AR(p)$ (Shumway & Stoffer, 2014).

2. Μοντέλο Κινητού Μέσου ($MA(q)$)

Το μοντέλο κινητού μέσου μπορεί να εκφραστεί με παρόμοιο τρόπο με αυτόν του μοντέλου αυτοπαλινδρόμισης, με τη διαφορά ότι η παρατήρηση της χρονοσειράς σε κάθε χρονική στιγμή είναι γραμμικός συνδυασμός του τωρινού (W_t) και των παρελθοντικών σφαλμάτων ($W_{t-1}, \dots, W_{t-q}$, που εμφάνισε το μοντέλο $MA(q)$ σε προηγούμενες περιόδους), αντί για τις τιμές της χρονοσειράς αυτές καθαυτές. Η παράμετρος q ονομάζεται τάξη (*order*) του κινητού μέσου. Αξίζει να σημειωθεί πως η τεχνική αυτή, όπως είδαμε, διαφέρει από τον υπολογισμό του κινητού μέσου που αναφέραμε στην υποενότητα 2.6.2. Το $MA(q)$ υπόδειγμα έχει την παρακάτω μορφή

$$X_t = W_t + \theta_1 W_{t-1} + \theta_2 W_{t-2} + \cdots + \theta_p W_{t-q}$$

όπου X στάσιμη χρονοσειρά, $\theta_1, \theta_2, \dots, \theta_q \in \mathbb{R} \quad (\theta_q \neq 0)$ και $W_t \sim GWN(0, \sigma^2)$. Όπως και στο αυτοπαλίνδρομο μοντέλο, το υπόδειγμα $MA(q)$ μπορεί να γραφεί

$$X_t = \left(1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q\right) W_t$$

ή

$$X_t = \theta(B) W_t.$$

(Shumway & Stoffer, 2014)

Συνδυάζοντας τα παραπάνω δύο στοιχεία μπορούμε να θεωρήσουμε ότι η (στάσιμη) χρονοσειρά X που εξετάζουμε είναι κατά ένα μέρος $AR(p)$ και κατά ένα άλλο $MA(q)$. Με αυτή την υπόθεση προκύπτει το υπόδειγμα $ARMA(p,q)$ με την παρακάτω μορφή

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + W_t + \theta_1 W_{t-1} + \cdots + \theta_q W_{t-q}$$

όπου $W_t \sim GWN(0, \sigma^2)$ και $\phi_p \neq 0, \theta_q \neq 0$.

Παρατηρούμε πως όταν $q = 0$, το μοντέλο που προκύπτει είναι το $AR(p)$ ενώ όταν $p = 0$ το μοντέλο που προκύπτει είναι ο κινητός μέσος τάξης q, $MA(q)$. Επιπλέον, πιο συνοπτικά, μπορούμε και γράφουμε το μοντέλο $ARMA(p,q)$ ως εξής:

$$\varphi(B)X_t = \theta(B)W_t$$

Αξίζει να σημειωθεί πως για να είναι στάσιμη μια $AR(p)$ χρονοσειρά, και κατ' επέκταση μια $ARMA(p,q)$, θα πρέπει οι χαρακτηριστικές ρίζες $r_1 = 1/x_1, r_2 = 1/x_2, \dots, r_p = 1/x_p$ (δηλαδή $\varphi(x) = (1 - r_1 x)(1 - r_2 x) \dots (1 - r_p x)$) που είναι λύσεις της χαρακτηριστικής εξίσωσης $\varphi(x) = 0$ να είναι κατά απολυτή τιμή μικρότερες της μονάδας, δηλαδή $|r_i| < 1, i = 1,2, \dots, p$.

3. Ολοκληρωμένο ($I(d)$)

   Το κομμάτι αυτό του μοντέλου αναφέρεται στη διαφόριση των παρατηρήσεων (δηλαδή λαμβάνοντας πρώτες διαφορές) προκειμένου να επιτευχθεί στασιμότητα στα δεδομένα. Η παράμετρος d ονομάζεται βαθμός διαφόρισης (*degree of differencing*) και καθορίζει το πόσες φορές θα εφαρμοστεί η διαφόριση στα δεδομένα.

Επομένως σε αρκετές περιπτώσεις μια μη στάσιμη χρονοσειρά μπορεί να μετασχηματιστεί σε στάσιμη ARMA μετά από μια ή περισσότερες εφαρμογές του τελεστή διαφορών. Έτσι, μια χρονοσειρά X είναι $ARIMA(p,d,q)$ αν

$$\mathrm{Y_t} = \nabla^d X_t = (1 - B)^d X_t$$

είναι $ARMA(p,q)$. Το μοντέλο γράφεται ως εξής:

$$\varphi(B)(1 - B)^d X_t = \theta(B)W_t,$$

Ενώ αν $E(\mathrm{Y_t}) = \mathrm{E}(\nabla^d X_t) = \mu \neq 0$ το μοντέλο γράφεται

$$\varphi(B)(1 - B)^d X_t = c + \theta(B)W_t,$$

όπου $c = \mu(1 - \varphi_1 - \cdots - \varphi_p)$ (Shumway & Stoffer, 2014).

Αφού προσδιορίσουμε την τάξη ενός μοντέλου ARIMA (τιμές p, q, d) θα πρέπει να εκτιμήσουμε τις παραμέτρους $c, \varphi_1, \varphi_2, \dots, \varphi_p$ και $\theta_1, \theta_2, \dots, \theta_q$. Η πιο συνηθισμένη

μέθοδος είναι οι εκτιμήτριες μέγιστης πιθανοφάνειας (*Maximum Likelihood Estimation (MLE)*). Η τεχνική αυτή βρίσκει τις τιμές των παραμέτρων οι οποίες μεγιστοποιούν τη πιθανότητα απόκτησης των δεδομένων που έχουμε παρατηρήσει. Για μεγάλα δείγματα $X_t, t = 1, \dots, n,$ η MLE ασυμπτωτικά ταυτίζεται με τις εκτιμήτριες ελαχίστων τετραγώνων (*MSE*) που θα λαμβάνονται με την ελαχιστοποίηση της

$$\sum_{t=1}^{n} W_t^2$$

(Athanasopoulos & Hyndman, 2018).

Τέλος, αναφέρεται πως για δεδομένα που παρουσιάζουν και εποχικότητα εφαρμόζεται το εποχικό ολοκληρωμένο αυτοπαλίνδρομο μοντέλο κινητού μέσου (*Seasonal AutoRegressive Integrated Moving Average (SARIMA)*) , το οποίο συμβολίζεται ως $SARIMA(p, d, q)(P, D, Q, m)$ όπου τα P, D, Q είναι οι ίδιοι όροι με τους p, d, q αλλά για το εποχικό κομμάτι του μοντέλου, ενώ η παράμετρος m δηλώνει το πλήθος των περιόδων σε μια εποχή.

ΚΕΦΑΛΑΙΟ 3

Πρόβλεψη Χρονοσειρών στην R

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο, θα εξετάσουμε την πρακτική εφαρμογή των νευρωνικών δικτύων στην πρόβλεψη χρονοσειρών. Η χρονοσειρά την οποία θα αφορούν οι προβλέψεις μας είναι οι τιμές κλεισίματος του QQQ *ETF* της Invesco. Αναφορικά, το ακρώνυμο ETF σημαίνει Διαπραγματεύσιμο Αμοιβαίο Κεφάλαιο (Exchange-Traded Fund) και είναι ένα προϊόν το οποίο ακολουθεί την απόδοση ενός δείκτη, ενός ομολόγου ή ακόμη και ενός συνδυασμού προϊόντων. Σε αντίθεση με άλλα αμοιβαία κεφάλαια, τα ETF αγοράζονται και πωλούνται στο χρηματιστήριο. Η απόδοση ενός ETF αντικατοπτρίζει τις κινήσεις των τιμών των υποκείμενων προϊόντων που περιλαμβάνονται στο κεφάλαιο. Στην προκειμένη περίπτωση, το QQQ ETF στοχεύει στο να αντιγράψει την απόδοση του δείκτη Nasdaq-100 και επομένως συγκροτείται από κλάσματα μετοχών εταιρειών που περιλαμβάνονται στον συγκεκριμένο δείκτη.

Για τη δημιουργία των νευρωνικών δικτύων, θα αξιοποιήσουμε το πακέτο Keras και TensorFlow. Το TensorFlow, αναπτυγμένο από την ομάδα Google Brain, είναι ένα λογισμικό ανοιχτού κώδικα που επιτρέπει την εφαρμογή διαφόρων αλγορίθμων της μηχανικής μάθησης με ιδιαίτερη έμφαση στα νευρωνικά δίκτυα. Ένα από τα πλεονεκτήματα του TensorFlow είναι η ικανότητα του να αξιοποιεί την μονάδα επεξεργασίας γραφικών (GPU) του υπολογιστή (αντί για την κεντρική μονάδα επεξεργασίας, CPU), επιταχύνοντας έτσι σημαντικά την εκτέλεση διεργασιών με υψηλό υπολογιστικό κόστος, όπως είναι τα βαθιά νευρωνικά δίκτυα. Το Keras αποτελεί μία Διεπαφή Προγραμματισμού Εφαρμογών (Application Programming interface, API), η οποία αν και αρχικά ήταν ανεξάρτητη βιβλιοθήκη, στην πορεία ενσωματώθηκε στο TensorFlow, συνδυάζοντας έτσι το φιλικό προς τον χρήστη περιβάλλον της Keras με την υπολογιστική δύναμη και αποδοτικότητα του TensorFlow.

3.2 Συλλογή και Επεξεργασία Δεδομένων

3.2.1 Περιγραφή Δεδομένων

Λαμβάνοντας υπόψη ότι το QQQ ETF αποτελείται σε ποσοστό μεγαλύτερο από το 50% από μετοχές που ανήκουν στον τομέα της τεχνολογίας, για την πρόβλεψη της τιμής κλεισίματος του θα αξιοποιήσουμε επίσης και τις τιμές από τις δέκα μεγαλύτερες μετοχές που συμβάλουν στη διαμόρφωση του δείκτη Nasdaq-100, εκ των οποίων οι εννέα δραστηριοποιούνται στον τομέα της τεχνολογίας. Τα δεδομένα αφορούν τις 820 χρηματιστηριακές ημέρες που μεσολάβησαν την χρονική περίοδο από 1η Οκτωβρίου, 2019 έως 30 Δεκεμβρίου, 2022 (βλ. Σχήμα 3.1) και συλλέχτηκαν από την ιστοσελίδα Yahoo Finance (https://finance.yahoo.com).

ΣΧΗΜΑ 3.1

Απεικόνιση Χρονοσειράς των Τιμών Κλεισίματος του QQQ

Αναφορικά, στις 27 Φεβρουαρίου του 2020, λόγω των αυξανόμενων ανησυχιών για την πανδημία του COVID-19, ο Nasdaq-100, δέχτηκε την μεγαλύτερη πτώση του μετά την παγκόσμια οικονομική ύφεση της περιόδου 2007-2008. Η πτώση συνεχίστηκε μέχρι τα τέλη Μαρτίου του ίδιου χρόνου όπου σταδιακά ο δείκτης άρχισε να ανακάμπτει, μαζί με το υπόλοιπο χρηματιστήριο. Αξίζει να σημειωθεί πως στα πρώτα στάδια της έρευνας, προκειμένου να αποτυπωθεί καλύτερα στα δεδομένα η επιρροή του COVID-19 στην εξέλιξη του δείκτη, στα μοντέλα νευρωνικών δικτύων συμπεριελήφθησαν και ορισμένοι δείκτες, όπως π.χ. ο δείκτης ανεργίας και το επιτόκιο δανεισμού μεταξύ των τραπεζών (όπου ήταν σχεδόν 0% τη πε-

ρίοδο εκείνη), αλλά δεν επέφεραν σημαντική βελτίωση στις προβλέψεις και γι' αυτό δεν χρησιμοποιήθηκαν στα μοντέλα που θα παρουσιαστούν παρακάτω. Πιο συγκεκριμένα, τα ανεπεξέργαστα δεδομένα αποτελούνται από 21 μεταβλητές που αφορούν την ημερομηνία της κάθε χρηματιστηριακής ημέρας (*Date*), την τιμή ανοίγματος (*Open*), τιμή κλεισίματος (*Close_QQQ*), προσαρμοσμένη τιμή κλεισίματος (*Adj Close*), την μεγαλύτερη και μικρότερη τιμή του QQQ (*High, Low*) για την κάθε χρηματιστηριακή ημέρα και ο όγκος συναλλαγών για τον QQQ (*Volume*), οι δείκτες μακροοικονομίας (*GDP, FEDFUNDS, CPILFESL, UNRATE*) και οι δέκα μετοχές που αναφέραμε νωρίτερα (*Close_AAPL, Close_MSFT, Close_AMZN, Close_NVDA, Close_TSLA, Close_GOOGL, Close_GOOG, Close_META, Close_AVGO, Close_PEP*). Επίσης, δεν υπήρχαν ελλειπούσες τιμές ή ασυνέπειες στα δεδομένα και το 70% από αυτά χρησιμοποιήθηκε για την εκπαίδευση των μοντέλων (*training set*), ενώ το 15% των δεδομένων αξιοποιήθηκε για την επικύρωση (*validation set*) και την βελτίωση της απόδοσής του μοντέλου κατά τη διάρκεια της εκπαίδευσης. Το υπόλοιπο 15% αντιστοιχίστηκε στο σύνολο ελέγχου (*test set)*, με σκοπό την αξιολόγηση της γενίκευσης του μοντέλου σε ανεξάρτητα δεδομένα.

3.2.2 Κανονικοποίηση κατά Παρτίδες (Batch Normalization)

Μια τεχνική που θα εφαρμόσουμε κατά την εκπαίδευση των νευρωνικών δικτύων και παρουσιάζει ιδιαίτερο ενδιαφέρον είναι αυτή της κανονικοποίησης κατά παρτίδες. Μια από τις δυσκολίες που παρουσιάζονται στην εκπαίδευση ενός νευρωνικού δικτύου είναι το γεγονός πως τα δεδομένα εισόδου σε κάθε στιβάδα επηρεάζονται από τις παραμέτρους όλων των προηγούμενων στιβάδων, έτσι η επιρροή ακόμη και μικρών αλλαγών στις παραμέτρους του δικτύου μεγεθύνεται όσο το δίκτυο γίνεται βαθύτερο. Η αλλαγή αυτή στην κατανομή των δεδομένων εισόδου αποτελεί πρόβλημα, καθώς οι στιβάδες πρέπει συνεχώς να προσαρμόζονται στην νέα κατανομή που προκύπτει. Το πρόβλημα αυτό προσπαθεί να επιλύσει η κανονικοποίησης κατά παρτίδες, μια τεχνική που εισήχθη από τους Ioffe και Szegedy (2015). Με την τεχνική αυτή η κανονικοποίηση είναι μέρος της αρχιτεκτονικής του δικτύου και εκτελείται για κάθε παρτίδα εκπαίδευσης του δικτύου και για όποιες στιβάδες επιθυμούμε.

Όπως είδαμε στο Κεφάλαιο 1, το σήμα εξόδου ενός νευρώνα z υπολογίζεται από την συνάρτηση ενεργοποίησης $f(\cdot)$ πάνω στο σταθμισμένο άθροισμα ($w^T x$) των σημάτων εισόδου (x), που δίνονται από την προηγουμένη στιβάδα, και την πόλωση b δηλαδή $z = f(net) = f(w^T x + b)$. Η μέθοδος των Ioffe και Szegedy στοχεύει στην κανονικοποίηση του

σήματος που δέχεται μια στιβάδα, πριν αυτό τροποποιηθεί από την συνάρτηση ενεργοποίησης. Πιο συγκεκριμένα, αν στην εκπαίδευση του νευρωνικού δικτύου οι πληροφορίες εισόδου, π.χ. εικόνες, έχουν χωριστεί σε παρτίδες μεγέθους m, τότε, για κάθε παρτίδα, σε έναν νευρώνα z της στιβάδας που θέλουμε να κανονικοποιήσουμε, θα αντιστοιχούν m τιμές $\{net^{(1)}, net^{(2)}, \dots, net^{(m)}\}$ της ποσότητας $w^T x + b$ (που θα υπολογίζεται για καθεμία από τις εικόνες που ανήκουν στη παρτίδα). Έτσι οι τιμές $\{net^{(1)}, net^{(2)}, \dots, net^{(m)}\}$ πριν περάσουν στην συνάρτηση ενεργοποίησης κανονικοποιούνται, δηλαδή

$$\widehat{net}^{(i)} = \frac{net^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}},$$

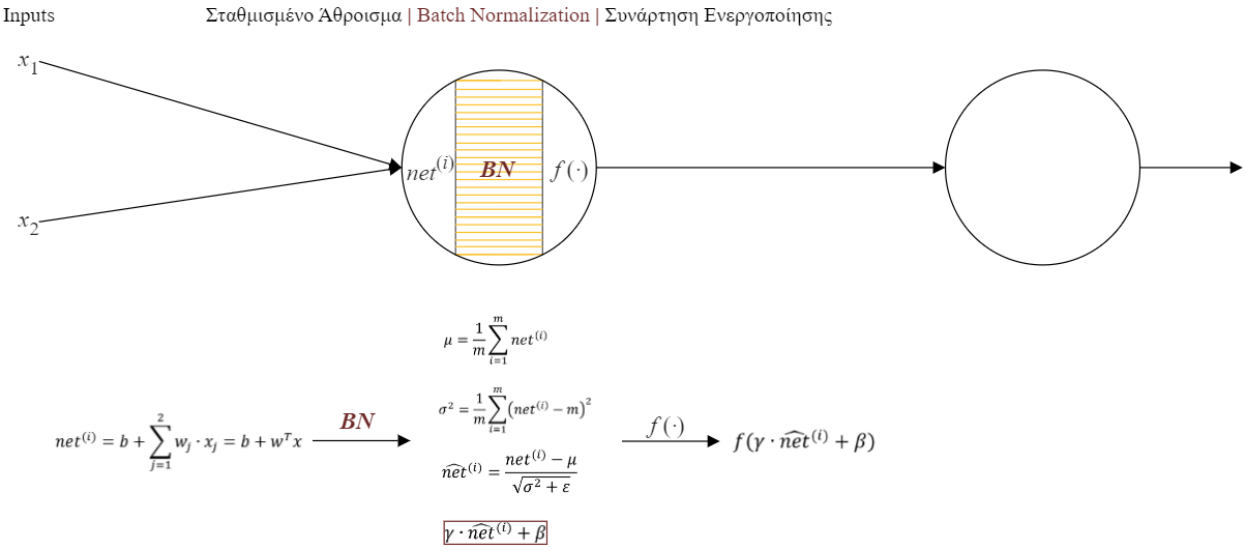
όπου $\mu = \frac{1}{m}\sum_{i=1}^{m} net^{(i)}$ η μέση τιμή της παρτίδας, $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}\left(net^{(i)} - m\right)^2$ η διακύμανση της παρτίδας και ε μια μικρή θετική ποσότητα, όπως για παράδειγμα 10^{-8}, για να αποφύγουμε την διαίρεση με το μηδέν.

Με τον τρόπο αυτό όμως μπορεί να μειωθεί η ερμηνευτική ικανότητα του δικτύου καθώς η είσοδος που θα δέχεται κάθε φορά η συνάρτηση ενεργοποίησης θα έχει πάντα μέση τιμή 0 και διακύμανση 1 (Goodfelow *et al.*, 2016; Ioffe & Szegedy ,2015). Για να αντιμετωπιστεί αυτό το πρόβλημα αντικαθιστούμε τις τιμές $net^{(i)}$ με $\gamma \cdot \widehat{net}^{(i)} + \beta$ αντί για την απλή κανονικοποίηση $\widehat{net}^{(i)}$. Οι παράμετροι γ και β μαθαίνονται μαζί με τις υπόλοιπες παραμέτρους του δικτύου και επαναφέρουν την ερμηνευτική ικανότητά του (για $\gamma = \sqrt{\sigma^2 + \varepsilon}$ και $\beta = \mu$, αν κριθεί σκόπιμο από το δίκτυο, το αποτέλεσμα είναι η αρχική, μη κανονικοποιημένη τιμή net). Το να ορίσουμε αρχικά τη μέση τιμή 0 και στη συνέχεια να εισάγουμε μια νέα παράμετρο που της επιτρέπει να πάρει πάλι μια οποιαδήποτε μέση τιμή β, φαινομενικά μπορεί να μοιάζει ως μια ανώφελη διαδικασία, παρ' όλ' αυτά είναι ιδιαιτέρως χρήσιμη στην εκπαίδευση του δικτύου. Η νέα παραμετροποίηση μπορεί να αναπαραστήσει την ίδια οικογένεια συναρτήσεων εισόδου, όπως η παλιά παραμετροποίηση, όμως αλλάζει την δυναμική της μάθησης. Στην μη κανονικοποιημένη παραμετροποίηση η μέση τιμή net καθορίζεται από τις περίπλοκες αλληλεπιδράσεις των παραμέτρων (βάρη, πολώσεις) που εμφανίστηκαν σε όλες τις προηγούμενες στιβάδες. Με τη νέα όμως παραμετροποίηση η μέση τιμή του $\gamma \cdot \widehat{net}^{(i)} + \beta$ εξαρτάται μόνο από το β, γεγονός που απλοποιεί το πρόβλημα βελτιστοποίησης που λύνεται καθώς το νευρωνικό δίκτυο εκπαιδεύεται (Brownlee, 2019; Goodfelow *et al.*, 2016; Ioffe & Szegedy ,2015). Επιπλέον, με την μέθοδο αυτή, μειώνεται το σφάλμα γενίκευσης του δικτύου

και κάνει περισσότερο ανθεκτικές τις προβλέψεις του. Στο Σχήμα 3.2 φαίνονται τα βήματα με τα οποία γίνεται η κανονικοποίηση κατά παρτίδες πάνω σε έναν νευρώνα.

ΣΧΗΜΑ 3.2

Κανονικοποίηση κατά Παρτίδες



Αν και οι δημιουργοί της μεθόδου προτείνουν η κανονικοποίηση να εφαρμοστεί αμέσως πριν τη μη-γραμμικότητα (συνάρτηση ενεργοποίησης), δηλαδή στον μετασχηματισμό $w^T x + b$, η κανονικοποίηση μπορεί να εφαρμοστεί εναλλακτικά κατευθείαν στις τιμές εξόδου x της προηγούμενης στοιβάδας (πριν αυτές μετασχηματιστούν από το σταθμισμένο άθροισμα) και αρκετές φορές αυτή η προσέγγιση οδηγεί σε βελτίωση της απόδοσης του δικτύου (Brownlee, 2019). Σημειώνουμε ότι στην περίπτωση που κανονικοποιούμε το $w^T x + b$ μπορούμε να παραλείψουμε τις πολώσεις b καθώς η επίδραση τους αναιρείται από την αφαίρεση της μέσης τιμής μ, και αναλαμβάνει η παράμετρος β τον ρόλο των πολώσεων στην νέα παραμετροποίηση (Ioffe & Szegedy, 2015). Τέλος, μπορούμε να κανονικοοποιήσουμε και τις ανεπεξέργαστες πληροφορίες εισόδου, προσθέτοντας στην αρχή του δικτύου, την στοιβάδα `layer_batch_normalization()`.

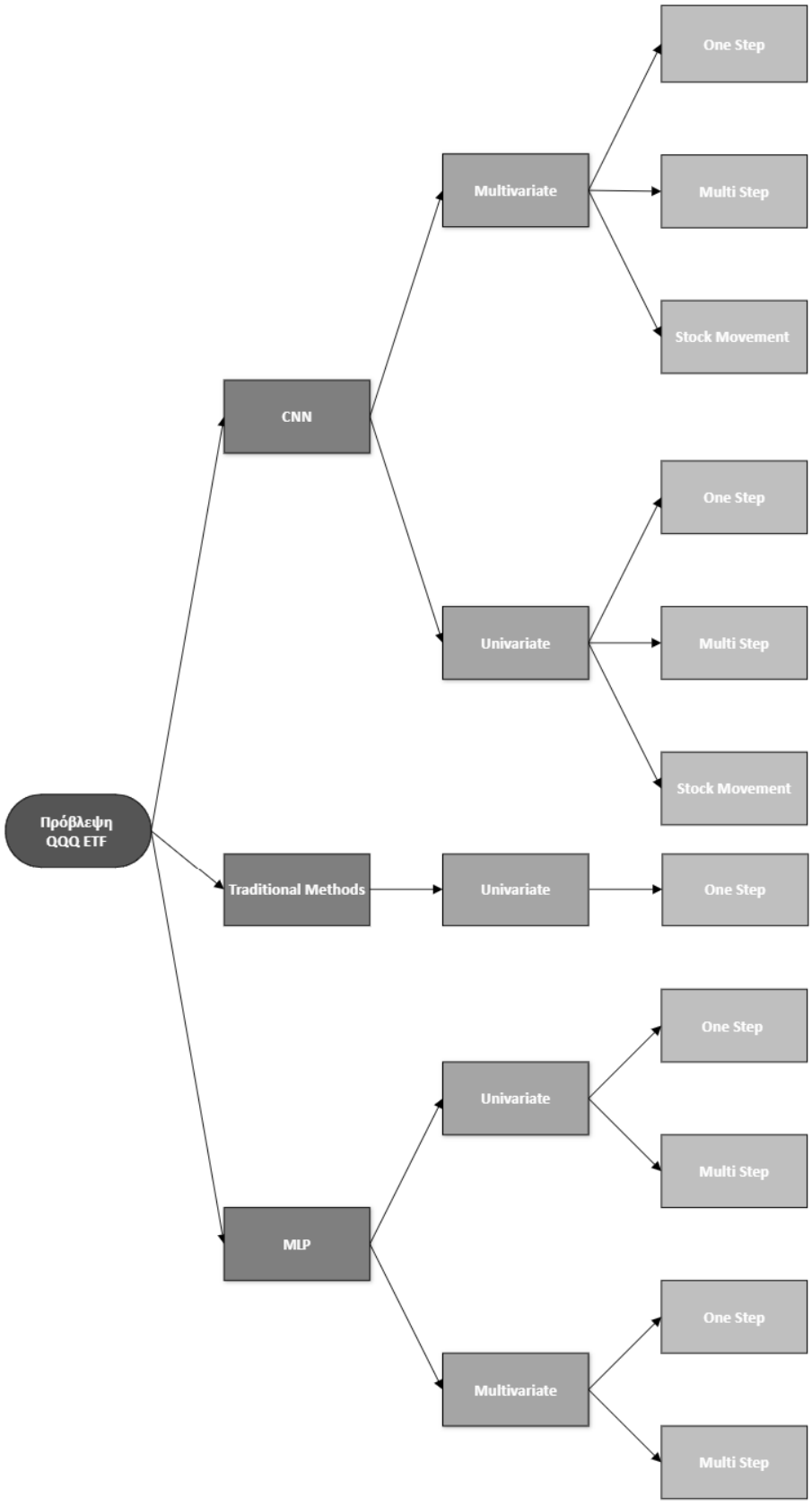
3.3 Εφαρμογή στην R

Οι κύριες κατηγορίες μοντέλων που θα εξετάσουμε σε αυτή την ενότητα είναι αυτά των Συνελικτικών Νευρωνικών Δικτύων (CNN) και των Αντιλήπτρων Πολλαπλών Στρωμάτων (MLP). Θα αναλύσουμε τόσο την περίπτωση μονομεταβλητών (Univariate) όσο και την περίπτωση πολυμεταβλητών (Multivariate) μοντέλων, προκειμένου να αξιολογήσουμε την επίδοσή τους σε διαφορετικά σενάρια. Ειδικότερα, θα εξετάσουμε τις προβλέψεις για ένα βήμα (One Step), όπου το μοντέλο προβλέπει κάθε φορά την τιμή της επόμενης χρηματιστηριακής ημέρας, και τις προβλέψεις για πέντε βήματα (Multi Step) μπροστά, όπου κάθε φορά δημιουργείται ένα διάνυσμα που περιέχει τις προβλεπόμενες τιμές για τις πέντε επόμενες χρηματιστηριακές ημέρες. Ακόμη, θα περιλάβουμε παραδοσιακές μεθόδους πρόβλεψης (Traditional Methods), όπως η μέθοδος Εκθετικής Εξομάλυνσης και ARIMA, προκειμένου να συγκρίνουμε την απόδοσή τους με αυτήν των νευρωνικών δικτύων.

Τέλος, με την χρήση του Functional API του Keras, που προσφέρει αυξημένη ευελιξία στη δομή του νευρωνικού δικτύου, θα εξετάσουμε πιο προχωρημένες προσεγγίσεις πρόβλεψης. Μέσω της χρήσης αρχιτεκτονικών CNN με υπο-μοντέλα θα προβλέψουμε την κίνηση (αύξηση ή μείωση τιμής) του QQQ μετά από τρεις ημέρες (Stock Movement with Sub-Models). Η πρόβλεψη για την κίνηση της τιμής του QQQ θα εξεταστεί επίσης και με μοντέλο CNN μιας αλλά και πολλών μεταβλητών (Univariate Stock Movement, Multivariate Stock Movement). Συγκεντρωτικά, όσα αναφέραμε περιγράφονται στο παρακάτω σχήμα.

ΣΧΗΜΑ 3.3

Επισκόπηση Μοντέλων Πρόβλεψης



Στα νευρωνικά δίκτυα παλινδρόμησης, ως συνάρτηση απώλειας (`loss`) χρησιμοποιήθηκε η μετρική MSE. Παράλληλα, το Keras παρέχει τη δυνατότητα να παρακολουθούμε μια επιπλέον μετρική μέσω της παραμέτρου `metrics` όπου στα προβλήματα παλινδρόμησης θέτουμε να είναι το MAE. Αυτή η μετρική αξιολογείται όπως και η συνάρτηση απώλειας, αλλά δεν επηρεάζει την διαδικασία εκπαίδευσης του δικτύου και δεν χρησιμοποιείται για την ενημέρωση των παραμέτρων. Για παράδειγμα, στα μοντέλα κατηγοριοποίησης που θα εφαρμόσουμε, συνάρτηση απώλειας είναι η Binary Cross-Entropy και η συνάρτηση βελτιστοποίησης επιδιώκει να ελαχιστοποιήσει αυτήν τη συνάρτηση. Παράλληλα, όμως, προσθέτουμε και την παράμετρο `metrics` με όρισμα την μετρική ακρίβειας (accuracy) του μοντέλου. Αυτό μας επιτρέπει να αξιολογήσουμε καλύτερα την συνολική απόδοση του μοντέλου, λαμβάνοντας υπόψη την ικανότητά του να κατηγοριοποιεί σωστά τα δείγματα.

Για να προσεγγίζουμε καλύτερα το πραγματικό επίπεδο απόδοσης του κάθε νευρωνικού δικτύου και να μειώσουμε την επίδραση της τυχαιότητας, θα εκτελούμε κάθε μοντέλο πέντε ξεχωριστές φορές. Η απόδοση του τελικά θα αξιολογείται από την μέση τιμή του MAE και του RMSE για τα δεδομένα ελέγχου. Επιπλέον με την παράμετρο `callbacks`, εκτός από το μοντέλο που προκύπτει στο τέλος της εκπαίδευσης, θα αποθηκεύουμε κάθε φορά και το μοντέλο που εμφανίστηκε στην εποχή εκείνη, με τη μικρότερη τιμή στη συνάρτηση απώλειας για τα δεδομένα επικύρωσης (`val_loss`). Με τον τρόπο αυτό διατηρούμε μοντέλα που εμφανίστηκαν σε νωρίτερα στάδια της εκπαίδευσης και αποφεύγουμε την υπερεκπαίδευση (overfitting), δηλαδή το φαινόμενο όπου το μοντέλο προσαρμόζεται υπερβολικά στα δεδομένα εκπαίδευσης και δυσκολεύεται να γενικεύσει σε νέα δεδομένα. Αν, για παράδειγμα, παρατηρήσουμε ότι μετά από μια εποχή η συνάρτηση απώλειας μειώνεται για τα δεδομένα εκπαίδευσης, αλλά αυξάνεται για τα δεδομένα επικύρωσης, αυτό μπορεί να είναι ένδειξη υπερεκπαίδευσης.

3.3.1 Multilayer Perceptrons

Αρχικά θα παρουσιάσουμε τα μονομεταβλητά και στη συνέχεια τα πολυμεταβλητά μοντέλα Αντιλήπτρων Πολλαπλών Στρωμάτων για τις περιπτώσεις προβλέψεων one step και mutli step.

3.3.1.1 Univariate MLP Μοντέλα

A) Univariate One step MLP Μοντέλα

Τα μονομεταβλητά δεδομένα αποτελούνται μόνο από την χρονοσειρά που προκύπτει από τις τιμές κλεισίματος του QQQ Trust και το νευρωνικό δίκτυο θα πρέπει να μάθει να αντιστοιχεί μια ακολουθία προηγούμενων παρατηρήσεων (input) με την αμέσως επόμενη παρατήρηση (output). Για να επιτευχθεί αυτό, θα αναλύσουμε την ακολουθία σε δείγματα, που αποτελούνται από ζευγάρια εισόδου-εξόδου, τα οποία θα δημιουργηθούν με βάση μια τιμή υστέρησης k. Για παράδειγμα, για την ακολουθία των πρώτων εφτά τιμών της χρονοσειράς *Close_QQQ*

$$[187.27, 184.05, 186.07, 188.81, 188.24, 185.42, 187.23, \dots,]$$

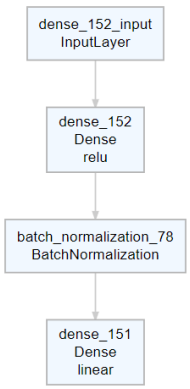
αν η $k = 3$, τότε τα δείγματα που θα κατασκευάσουμε θα είναι

$$[187.27, 184.05, 186.07],\ [188.81]$$
$$[184.05, 186.07, 188.81],\ [188.24]$$
$$[186.07, 188.81, 188.24],\ [185.42]$$
$$[\ 188.81, 188.24, 185.42],\ [187.23]$$
$$\vdots$$

όπου κάθε δείγμα έχει τρία χρονικά βήματα ως input και ένα βήμα ως output. Η διάσταση των τιμών εισόδου πρέπει να δηλωθούν και στο νευρωνικό δίκτυο με το όρισμα `input_shape=3` στη πρώτη κρυφή στιβάδα του. Στην ουσία το δίκτυο βλέπει κάθε βήμα της ακολουθίας σαν διαφορετικό χαρακτηριστικό. Παρακάτω φαίνεται η αρχιτεκτονική του νευρωνικού δικτύου.

ΣΧΗΜΑ 3.4

Αρχιτεκτονική Univariate One Step MLP Μοντέλων



Layer (type)	Output Shape	Param #	Trainable
(Dense)	(None, 16)	64	Y
Batch (BatchNormalization)	(None, 16)	64	Y
dense2 (Dense)	(None, 1)	17	Y

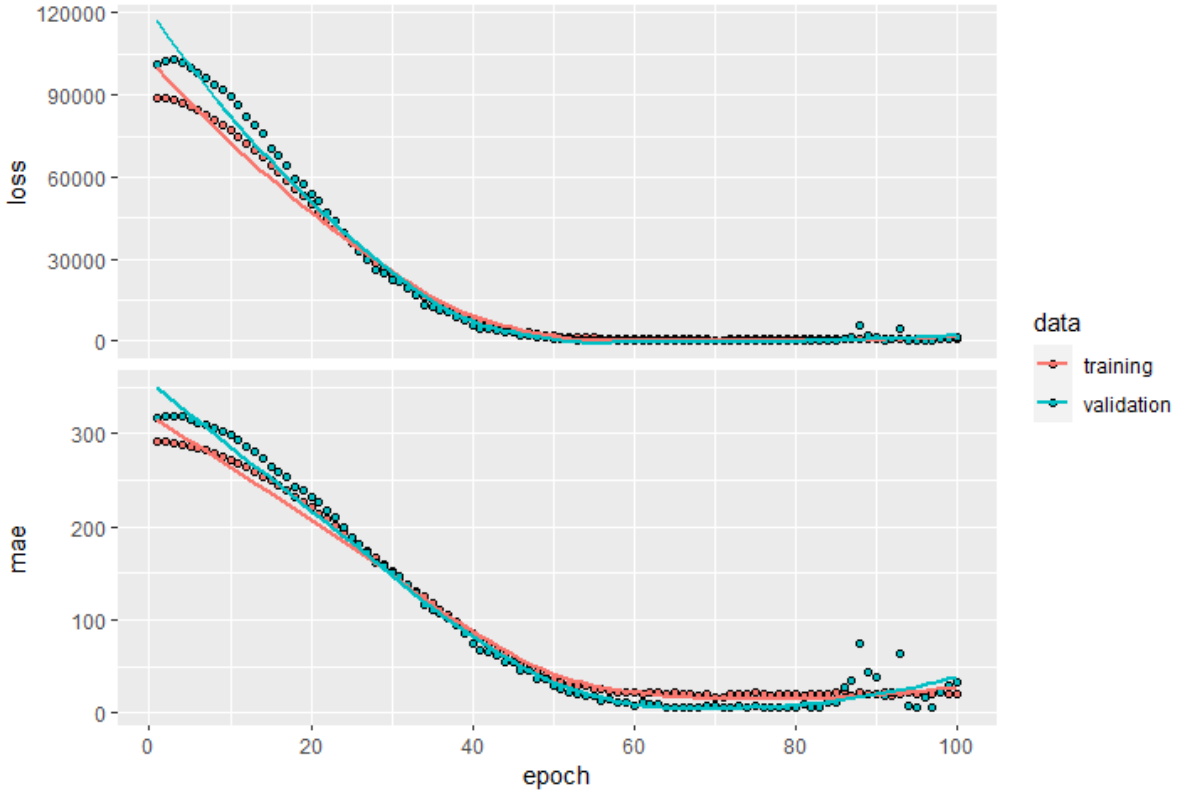
Total params: 145
Trainable params: 113
Non-trainable params: 32

Όπως βλέπουμε, πέρα από τις στιβάδες εισόδου και εξόδου, το δίκτυο αποτελείται από μία κρυφή στιβάδα διάστασης (πλήθος νευρώνων) 16 (στήλη Shape), όπου το πλήθος των παραμέτρων της είναι 64 (στήλη *Param #*), και από την στιβάδα κανονικοποίησης. Επίσης, η διάσταση *None* στον πίνακα του σχήματος 3.2 υποδηλώνει πως μπορούμε να χρησιμοποιήσουμε παρτίδες οποιουδήποτε μεγέθους. Το μοντέλο θα εκπαιδευτεί για 100 εποχές και μέγεθος παρτίδων ίσο με 8, πέντε φορές με την αλγόριθμο βελτιστοποίησης Adam και 5 φορές με τον RMSprop. Ο ρυθμός μάθησης η και για τους δύο αλγορίθμους είναι ίσος με 0.001. Επίσης για τιμές υστέρησης θα δοκιμαστούν $k = \{3,5,10\}$

Παρακάτω, ενδεικτικά παρουσιάζουμε το γράφημα που απεικονίζει την πρόοδο εκπαίδευσης του δικτύου (Adam, $k = 3$)

ΣΧΗΜΑ 3.5

Πρόοδος Εκπαίδευσης Univariate One Step MLP



Στο πάνω μέρος του γραφήματος απεικονίζεται η εξέλιξη της συνάρτησης απώλειας, δηλαδή του MSE, για τα δεδομένα εκπαίδευσης (κόκκινο χρώμα) και τα δεδομένα επικύρωσης (πράσινο χρώμα). Στο κάτω μέρος του γραφήματος παρουσιάζεται η μετρική MAE. Το διάγραμμα αυτό επιβεβαιώνει την σημαντικότητα της παραμέτρου `callbacks`, καθώς αν και προς το

τέλος της εκπαίδευσης το σφάλμα επικύρωσης για το MSE φαίνεται να παραμένει σταθερό, παρατηρούμε αστάθεια στο σφάλμα επικύρωσης όσον αφορά τη μετρική MAE. Πιο συγκεκριμένα, παρατηρούμε πως ενώ το σφάλμα εκπαίδευσης παραμένει σχετικά σταθερό, το σφάλμα επικύρωσης για τη μετρική MAE αυξάνεται. Αυτό αποτελεί ένδειξη υπερεκπαίδευσης του δικτύου, και επομένως το μοντέλο που προέκυψε από τον καθορισμό των παραμέτρων στο τέλος της εκπαίδευσης δεν είναι το προτιμότερο. Το παραπάνω επιβεβαιώνεται με το MSE και το MAE που προκύπτει για τα δεδομένα ελέγχου χρησιμοποιώντας το μοντέλο στο τέλος της εκπαίδευσης

```
> model %>% evaluate(test_x, test_y)
4/4 [==============================] - 0s 2ms/step - loss: 1022.6806 - mae: 31.4690
      loss        mae
1022.68060   31.46903
```

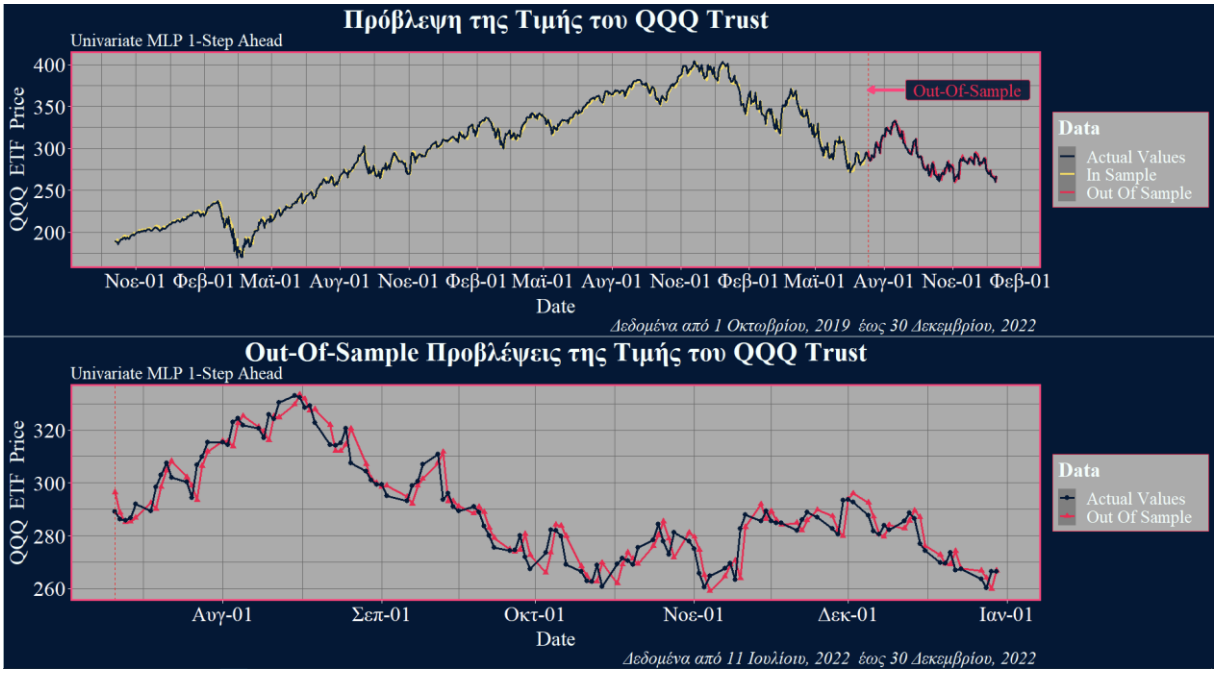
έναντι του μοντέλου που αποθηκεύτηκε με την χρήση της παραμέτρου `callbacks`

```
> best_model %>% evaluate(test_x, test_y)
4/4 [==============================] - 0s 2ms/step - loss: 29.6141 - mae: 4.3895
     loss       mae
29.614138  4.389476
```

Στο πάνω μέρος του παρακάτω σχήματος, με μπλε χρώμα απεικονίζονται οι πραγματικές τιμές για την τιμής κλεισίματος του QQQ Trust, με κίτρινο χρώμα οι *in sample* (training set και validation set) προβλέψεις, ενώ με κόκκινο χρώμα απεικονίζονται οι *out-of-sample* (test set) προβλέψεις. Στο κάτω μέρος του γραφήματος συγκρίνονται σημείο-σημείο οι πραγματικές τιμές με αυτές που προέβλεψε το μοντέλο για το test set. Αυτό μας δίνει μια ακριβέστερη εικόνα για την απόδοση του μοντέλου στα συγκεκριμένα δεδομένα. Παρατηρούμε ότι το μοντέλο αναγνωρίζει την τάση της χρονοσειράς τόσο για τα δεδομένα εκπαίδευσης όσο και για τα δεδομένα ελέγχου, με μικρή απόκλιση στις διακυμάνσεις.

ΣΧΗΜΑ 3.6

Απόδοση Univariate One Step MLP

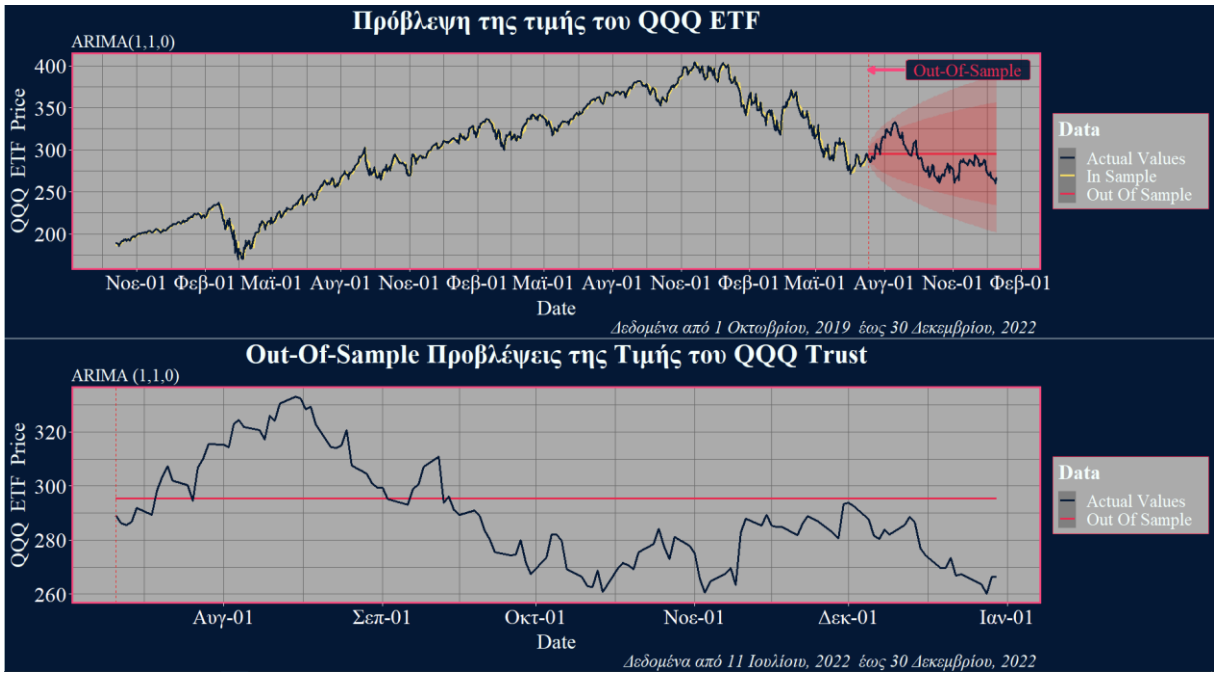


Στον Πίνακα 3.1 συνοψίζονται τα αποτελέσματα όλων των Univariate MLP μοντέλων που εκτελέστηκαν και τα αποτελέσματα των κλασικών μεθόδων. Οι κλασικές τεχνικές πρόβλεψης χρονοσειρών που χρησιμοποιήσαμε περιλαμβάνουν την εκθετική εξομάλυνση, τη μέθοδο Holt και τη μέθοδο ARIMA. Για τη δημιουργία του μοντέλου ARIMA, πιο συγκεκριμένα, αναφέρουμε ότι αξιοποιήσαμε το πακέτο `forecast` και τη συνάρτηση `auto.arima()` με την παράμετρο `ic=aic`. Με αυτόν τον τρόπο λαμβάνουμε ως αποτέλεσμα το βέλτιστο μοντέλο ARIMA (1,1,0) βάσει του Κριτηρίου Πληροφοριών Akaike (Akaike Information Criterion, AIC). Στο Σχήμα 3.5 παρουσιάζονται οι προβλέψεις του βέλτιστου μοντέλου ARIMA για τα in sample και out-of-sample δεδομένα. Η περιοχή με κόκκινη σκίαση αντιστοιχεί στο 95% διάστημα εμπιστοσύνης των προβλέψεων, ενώ εντός αυτής, και με έντονη κόκκινη σκίαση, εμφανίζεται το 80% διάστημα εμπιστοσύνης των προβλέψεων. Λόγω της απουσίας εμφανούς μοτίβων στα δεδομένα, το μοντέλο δεν μπορεί να εντοπίσει την ανάγκη για πιο περίπλοκες προβλέψεις. Ως αποτέλεσμα, οι προβλέψεις του για μελλοντικά χρονικά σημεία κινούνται γύρω από μια σταθερή μέση τιμή, χωρίς να λαμβάνονται υπόψη ενδεχόμενες διακυμάνσεις. Σημειώνουμε ότι κατά τη διάρκεια της εκπαίδευσης, το μοντέλο ARIMA χρησιμοποιεί όλα τα διαθέσιμα ιστορικά δεδομένα για να κατασκευάσει τις προβλέψεις για ένα επόμενο χρονικό βήμα. Ωστόσο, κατά την αξιολόγηση του, στο test set, οι προβλέψεις γίνονται

για πολλά χρονικά βήματα μπροστά και κάθε νέα πρόβλεψη ή παρατήρηση του test set δεν λαμβάνεται υπόψη από το μοντέλο κατά την πρόβλεψη των επόμενων βημάτων και αυτός είναι ο συνηθέστερος τρόπος εφαρμογής ενός μοντέλου ARIMA (Athanasopoulos & Hyndman, 2018). Αυτός είναι και ο λόγος όπου στα in sample δεδομένα οι προβλέψεις βρίσκονται πολύ κοντά στις πραγματικές τιμές ενώ η μορφή των προβλέψεων αλλάζει για τα out-of-sample δεδομένα.

ΣΧΗΜΑ 3.7

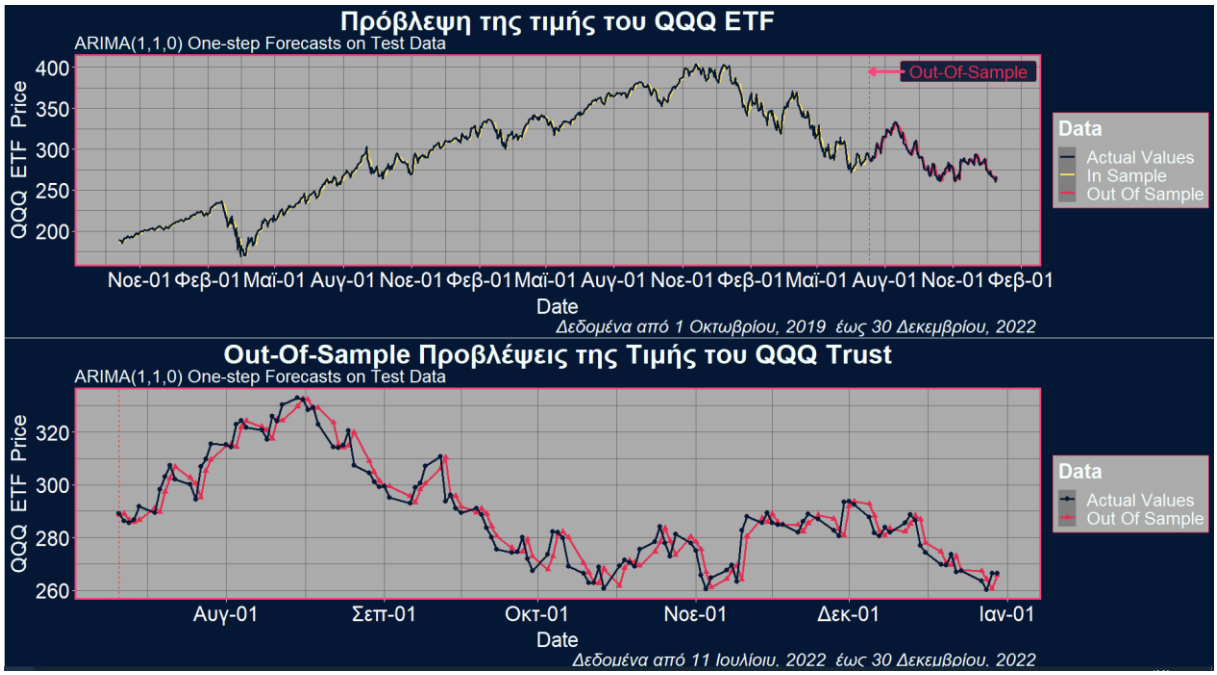
Απόδοση Μοντέλου ARIMA



Μια τεχνική (*one-step forecasts on test data*) για τη βελτίωση της απόδοσης του μοντέλου περιλαμβάνει το να πραγματοποιούμε διαδοχικές προβλέψεις για ένα βήμα μπροστά στο test set, χωρίς ωστόσο να γίνεται επανεκτίμηση του μοντέλου για κάθε νέο χρονικό βήμα (του test set). Με αυτόν τον τρόπο το μοντέλο κάνει προβλέψεις καθώς νέα δεδομένα γίνονται διαθέσιμα χρησιμοποιώντας όλα τα ιστορικά δεδομένα (και training και test set), χωρίς όμως να αλλάζει τις παραμέτρους του μοντέλου, οι οποίες υπολογίστηκαν μόνο πάνω στα δεδομένα εκπαίδευσης. Έτσι στο Σχήμα 3.8 φαίνεται η απόδοση του μοντέλου ARIMA(1,1,0) με την εφαρμογή της τεχνικής που περιγράψαμε, με χρήση του πακέτου *forecast* των Hyndman και Athanasopoulos.

ΣΧΗΜΑ 3.8

Απόδοση Μοντέλου ARIMA με One-Step Test Set Forecasts



Συγκεντρωτικά όλα τα αποτελέσματα φαίνονται στον παρακάτω πίνακα.

Πίνακας 3.1 Σύγκριση Univariate 1-Step MLP μοντέλων και Παραδοσιακών μεθόδων πρόβλεψης χρονοσειρών

Χρονική Υστέρηση	Αλγόριθμος Βελ-τιστοποίησης	Προβλεπτική Ακρίβεια (out-of sample)	
k	*optimizer*	*MAE*	*RMSE*
3	Adam	4.389476	5.441888
3	Adam	4.346385	5.411989
3	Adam	4.531118	5.534027
3	Adam	4.29241	5.424921
3	Adam	4.41269	5.444999
5	Adam	4.3931	5.45994
5	Adam	4.91544	5.97831
5	Adam	4.43314	5.43818
5	Adam	4.41168	5.49598
5	Adam	4.41503	5.46727
10	Adam	4.484765	5.575167
10	Adam	4.311111	5.454985
10	Adam	4.402398	5.45506
10	Adam	4.246698	5.422485
10	Adam	4.791342	6.168313
3	RMSprop	4.408402	5.438628

3	RMSprop	4.657157	6.625332
3	RMSprop	4.384506	5.436147
3	RMSprop	4.351961	5.446255
3	RMSprop	4.308378	5.409171
5	RMSprop	4.214069	5.387583
5	RMSprop	4.360541	5.430365
5	RMSprop	4.900392	7.875724
5	RMSprop	4.309134	5.366659
5	RMSprop	4.598143	5.628791
10	RMSprop	4.852949	5.971238
10	RMSprop	4.404939	5.501462
10	RMSprop	4.35789	5.563161
10	RMSprop	4.484495	5.475642
10	RMSprop	4.336353	5.626457
Adam 3-Lags Mean		4.3944158	5.451565
Adam 5-Lags Mean		4.5136774	5.5679342
Adam 10-Lags Mean		4.4472628	5.615202
RMSprop 3-Lags Mean		4.4220808	5.6711066
RMSprop 5-Lags Mean		4.4764558	5.9378244
RMSprop 10-Lags Mean		4.4873252	5.627592
ARIMA (1,1,0)		17.02069	19.79745
ARIMA (1,1,0) One-Step		**4.280042**	**5.393995**
Exponential Smoothing		16.9901	19.77375
Holt's Trend		24.14034	27.43035

Σημείωση: Με έντονη γραφή δηλώνουμε το μοντέλο με την καλύτερη απόδοση.

Από τον πίνακα προκύπτει ότι η απόδοση των νευρωνικών δικτύων στο σύνολο ελέγχου υ-περτερεί σημαντικά των παραδοσιακών μεθόδων πρόβλεψης, παρ' ολ' αυτά το μοντέλο ARIMA με τη μέθοδο προβλέψεων ενός βήματος, πετυχαίνει καλύτερη απόδοση. Ειδικότερα, το νευρωνικό δίκτυο που είχε την υψηλότερη απόδοση, αντιστοιχεί στο μοντέλο με χρονική υστέρηση 3 και χρήση της συνάρτησης βελτιστοποίησης Adam. Επιπλέον, θυμίζουμε από το κεφάλαιο 1 ότι το RMSE δίνει μεγαλύτερη έμφαση στις μεγάλες αποκλίσεις, ενώ το MAE παρέχει μια πιο ισορροπημένη εικόνα της γενικής απόδοσης του μοντέλου, και λόγω αυτού του χαρακτηριστικού, οι μετρικές αυτές μπορεί να οδηγήσουν σε διαφορετικές επιλογές ως προς το ποιο μοντέλο είναι καλύτερο όταν συγκρίνουμε δύο μοντέλα (π.χ. σύγκριση RMSprop 5-Lags Mean και RMSprop 10-Lags Mean).

B) Univariate Mutli Step MLP Μοντέλα

Στην πράξη, η λειτουργία του MLP παρουσιάζει ελάχιστες διαφορές σχετικά με το αν προβλέπει ένα διάνυσμα εξόδου που αντιστοιχεί σε διαφορετικές μεταβλητές ή ένα διάνυσμα που αντιπροσωπεύει πολλά συνεχόμενα χρονικά βήματα για μία μόνο μεταβλητή (Univariate Mutli Step MLP). Η επεξεργασία των δεδομένων πριν εκπαιδεύσουμε το νευρωνικό δίκτυο είναι παρόμοια με την περίπτωση του ενός βήματος. Η διαφορά τώρα είναι ότι θα εμφανίζονται χρονικά βήματα και στο output Για παράδειγμα, για την ακολουθία των πρώτων εφτά τιμών της χρονοσειράς *Close_QQQ*, που είδαμε και νωρίτερα,

$$[187.27, 184.05, 186.07, 188.81, 188.24, 185.42, 187.23, \dots,]$$

αν η χρονική υστέρηση είναι 3 και θέλουμε να προβλέψουμε τις επόμενες 2 τιμές τα δείγματα που θα κατασκευάσουμε θα είναι

$$[187.27, 184.05, 186.07],\ [188.81, 188.24]$$
$$[184.05, 186.07, 188.81],\ [188.24, 185.42]$$
$$[186.07, 188.81, 188.24],\ [185.42, 187.23]$$
$$\vdots$$

όπου κάθε δείγμα έχει τρία χρονικά βήματα ως input και τα δύο επόμενα χρονικά βήματα ως output. Το πλήθος των δειγμάτων εξαρτάται από την χρονική υστέρηση και από τον ορίζοντα των μελλοντικών προβλέψεων. Ο τύπος από τον οποίο προκύπτουν τα δείγματα είναι

$$n - k - (l - 1)$$

όπου, n είναι το μήκος της χρονοσειράς, k είναι η χρονική υστέρηση και l το πλήθος των μελλοντικών βημάτων που θέλουμε να προβλέψουμε. Για παράδειγμα αν έχουμε τη χρονοσειρά $[1,2,3,4,5,6,7,8,9,10]$ και θέλουμε να χρησιμοποιήσουμε χρονική υστέρηση $k = 3$ για να προβλέψουμε $l = 4$ μελλοντικά βήματα, τότε ο αριθμός των δειγμάτων που θα προκύψουν θα είναι

$$n - k - (l - 1) = 10 - 3 - (4 - 1) = 4$$

Τα τέσσερα αυτά δείγματα, είναι οι γραμμές με πράσινη σκίαση του παρακάτω Πίνακα 3.2.

Πίνακάς 3.2 Παράδειγμα Input-Output Δειγμάτων

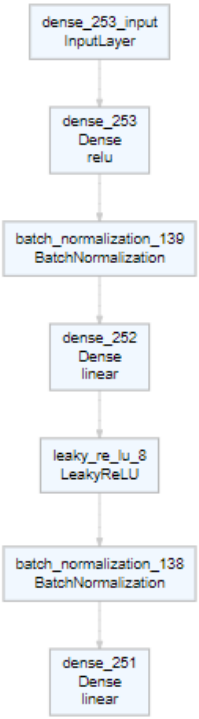
Input			Output			
Lag 3	**Lag 2**	**Lag 1**	**Data**	**Lead 1**	**Lead 2**	**Lead 3**
NA	NA	NA	1	2	3	4
NA	NA	1	2	3	4	5
NA	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	NA
6	7	8	9	10	NA	NA
7	8	9	10	NA	NA	NA
8	9	10				
9	10					
10						

Όπως παρατηρούμε και από τον πίνακα, ο προηγούμενος τύπος λαμβάνει υπόψη την απώλεια γραμμών στην αρχή των δειγμάτων, λόγω της χρονικής υστέρησης, και την απώλεια γραμμών στο τέλος των δειγμάτων, λόγω του αριθμού των μελλοντικών προβλέψεων που θέλουμε να δημιουργήσουμε.

Παρακάτω (βλ**.** Σχήμα 3.9) φαίνεται η αρχιτεκτονική του νευρωνικού δικτύου όπου αποτελείται από 2 κρυφές στιβάδες. Η μία στιβάδα έχει συνάρτηση ενεργοποίησης την ReLU και η άλλη την LReLU, όπου η τιμή της παραμέτρου a για την κλίση των αρνητικών τιμών είναι $a = 0.1$. Επιπλέον, στο τέλος κάθε κρυφής στιβάδας έχουμε ορίσει μια στιβάδα κανονικοποίησης. Σημειώνουμε ότι για να δηλώσουμε την τιμή της παραμέτρου a στην LReLU, πρέπει να αξιοποιήσουμε το Keras Functional API και να δηλώσουμε πρώτα τη στιβάδα με το πλήθος των νευρώνων της και ξεχωριστά τη συνάρτηση ενεργοποίησης.

ΣΧΗΜΑ 3.9

Αρχιτεκτονική Univariate Mutli Step MLP Μοντέλων



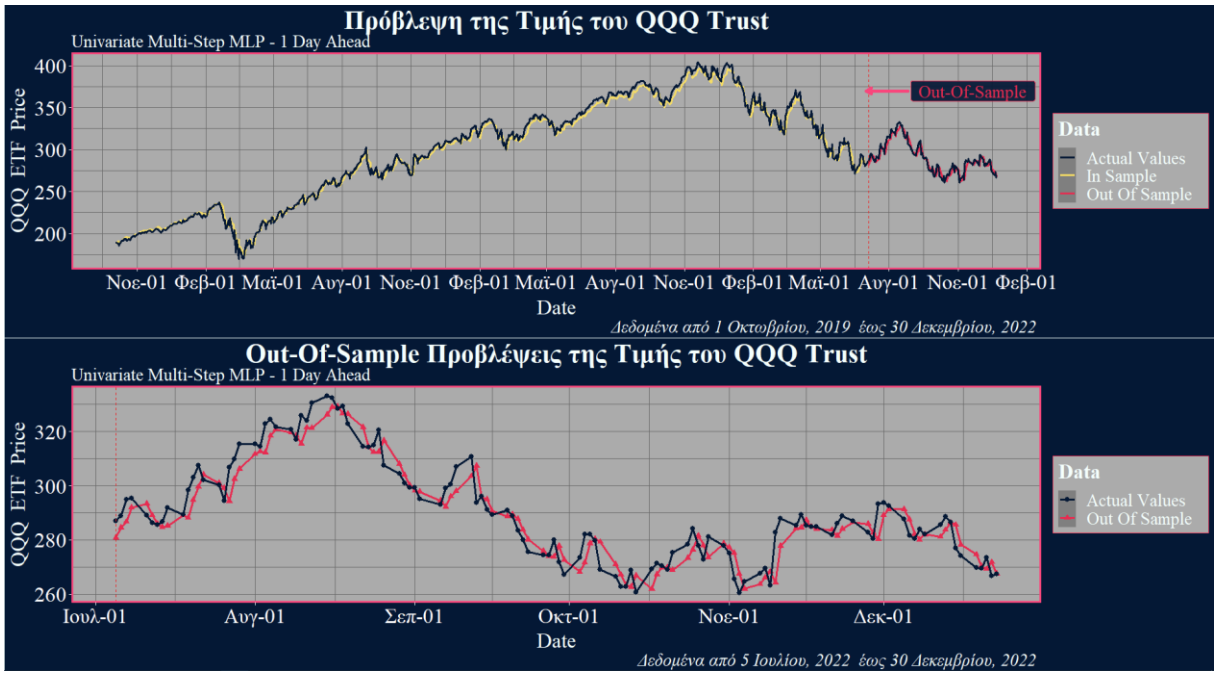
Layer (type)	Output Shape	Param #	Trainable
dense_253 (Dense)	(None, 32)	128	Y
batch_normalization_139 (BatchNormalization)	(None, 32)	128	Y
dense_252 (Dense)	(None, 16)	528	Y
leaky_re_lu_8 (LeakyReLU)	(None, 16)	0	Y
batch_normalization_138 (BatchNormalization)	(None, 16)	64	Y
dense_251 (Dense)	(None, 5)	85	Y

Total params: 933
Trainable params: 837
Non-trainable params: 96

Στη συνέχεια, επιλέγουμε την πρώτη και την πέμπτη θέση από τα διανύσματα εξόδου που προκύπτουν από τις προβλέψεις του νευρωνικού δικτύου με χρονική υστέρηση $k = 3$ και αλγόριθμο βελτιστοποίησης τον Adam. Με αυτόν τον τρόπο παίρνουμε τις προβλέψεις για την επόμενη χρηματιστηριακή ημέρα και για την χρηματιστηριακή ημέρα πέντε βήματα μετά, τις οποίες θα συγκρίνουμε με τις πραγματικές τιμές της χρονοσειράς. Η απόδοση τους απεικονίζεται στο Σχήμα 3.10 και Σχήμα 3.11 αντιστοίχως.

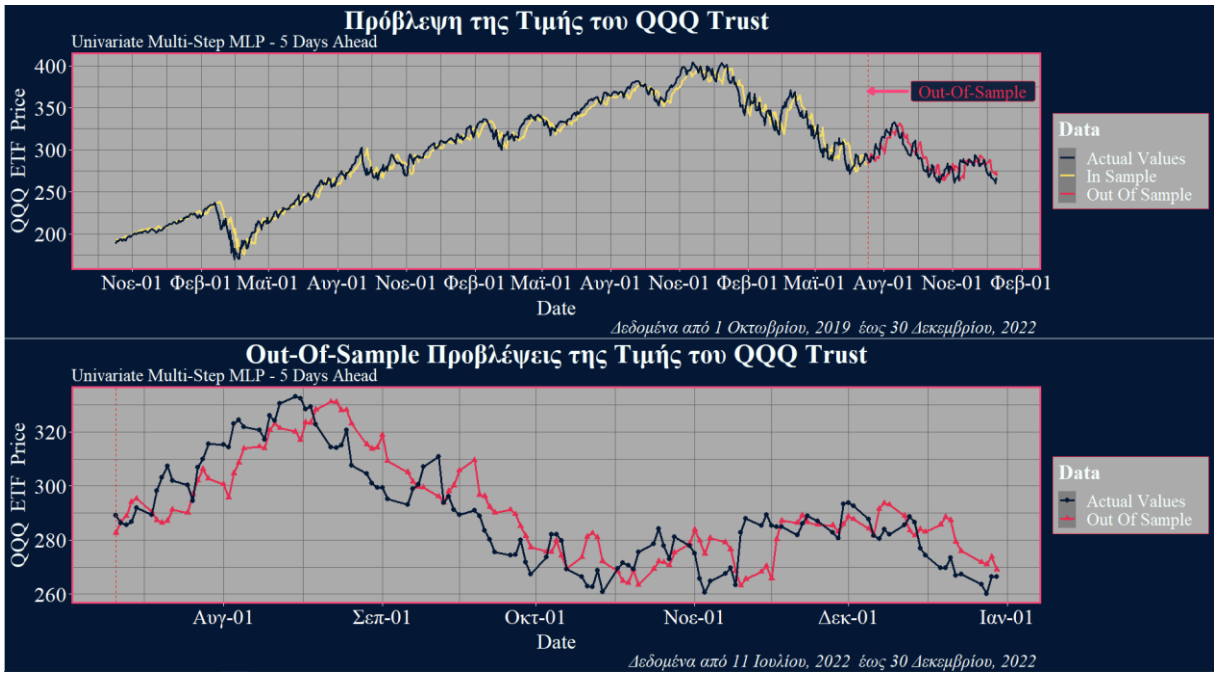
ΣΧΗΜΑ 3.10

Απόδοση Univariate Mutli Step MLP για 1 Μέρα Μπροστά



ΣΧΗΜΑ 3.11

Απόδοση Univariate Mutli Step MLP για 5 Μέρες Μπροστά



Παρατηρούμε ότι το νευρωνικό δίκτυο καταφέρνει και σε αυτή την περίπτωση να προβλέψει ικανοποιητικά την πορεία της χρονοσειράς για τις προβλέψεις ενός βήματος, ενώ οι προβλέψεις πέντε βημάτων, όπως θα περιμέναμε, εμφανίζουν μικρότερη ακρίβεια. Παρ' όλ' αυτά και στις δύο περιπτώσεις το μοντέλο καταφέρνει να ανιχνεύσει την γενικότερη τάση της χρονοσειράς.

Στον παρακάτω πίνακα παρουσιάζουμε αναλυτικά τα αποτελέσματα από τις πέντε επαναλήψεις κάθε μοντέλου. Η αξιολόγηση του γίνεται πάνω στα δεδομένα ελέγχου και για το κάθε σενάριο, στις στήλες *1* έως *5*, φαίνεται το MAE και το RMSE για τις προβλέψεις για ένα έως και πέντε χρονικά βήματα στο μέλλον. Επίσης, η στήλη *Total* αντιστοιχεί στις τιμές των μετρικών όταν λαμβάνουμε υπόψη το σφάλμα που προκύπτει και από τις πέντε μέρες που προβλέπονται.

Πίνακας 3.3 Σύγκριση Univariate Multi-Step MLP μοντέλων

Lags	Αλγόρ. Βελτιστ.	Προβλεπτική Ακρίβεια (out-of sample)											
		MAE						RMSE					
k	*optimizer*	*1*	*2*	*3*	*4*	*5*	*Total*	*1*	*2*	*3*	*4*	*5*	*Total*
3	Adam	4.372	6.179	7.566	8.672	9.130	7.184	5.509	7.567	9.042	10.30	10.82	8.860
3	Adam	4.703	6.602	7.958	8.702	9.298	7.453	5.766	7.985	9.430	10.26	10.91	9.057
3	Adam	4.679	6.212	7.611	8.537	9.279	7.264	5.679	7.573	9.221	10.11	10.87	8.889
3	Adam	4.591	6.315	7.811	8.469	9.230	7.283	5.678	7.707	9.430	10.40	11.03	9.061
3	Adam	4.541	6.210	7.600	8.335	9.605	7.258	5.539	7.556	9.032	10.05	11.21	8.901
5	Adam	4.531	6.523	7.753	8.762	9.345	7.383	5.833	7.867	9.199	10.40	10.97	9.045
5	Adam	4.250	6.272	7.593	8.293	9.228	7.127	5.376	7.610	9.127	10.10	10.95	8.854
5	Adam	4.609	6.289	8.169	9.185	9.474	7.545	5.684	7.631	9.775	11.01	11.08	9.274
5	Adam	5.073	6.603	7.632	8.844	9.292	7.489	6.151	8.047	9.168	10.55	10.92	9.135
5	Adam	4.592	6.386	7.814	8.324	9.254	7.274	5.816	7.763	9.458	9.956	10.94	8.970
10	Adam	4.626	6.480	7.641	8.283	9.142	7.234	5.708	7.814	9.122	10.12	10.84	8.907
10	Adam	4.627	6.417	7.633	8.583	9.106	7.273	5.850	7.805	9.115	10.37	10.76	8.961
10	Adam	4.568	6.671	7.731	8.828	9.259	7.411	5.961	8.133	9.447	10.74	11.01	9.245
10	Adam	4.871	6.360	8.900	8.163	9.512	7.561	6.043	7.832	10.72	9.954	11.28	9.372
10	Adam	4.569	6.543	7.882	8.440	9.288	7.344	6.082	8.004	9.379	10.25	11.00	9.110
3	RMSprop	4.672	6.137	7.793	8.519	9.486	7.321	5.698	7.536	9.194	10.13	11.10	8.942
3	RMSprop	6.186	6.742	8.089	8.716	9.301	7.807	7.794	8.367	9.900	10.77	11.10	9.675
3	RMSprop	4.579	6.320	7.730	8.318	9.061	7.202	5.564	7.703	9.202	10.04	10.70	8.835
3	RMSprop	4.705	6.428	7.649	8.460	9.336	7.316	5.704	7.717	9.263	10.28	11.18	9.039
3	RMSprop	4.523	6.550	7.739	8.396	9.498	7.341	5.682	7.998	9.287	10.21	11.06	9.044
5	RMSprop	5.408	6.396	8.745	9.288	10.25	8.018	6.661	7.734	10.32	11.01	12.06	9.772
5	RMSprop	4.553	6.308	8.137	8.595	9.253	7.369	5.670	7.673	9.910	10.25	10.91	9.093

5	RMSprop	5.047	7.011	7.883	8.917	9.534	7.679	6.197	8.522	9.446	10.88	11.20	9.425
5	RMSprop	5.736	6.229	7.655	8.865	10.60	7.818	6.847	7.618	9.127	10.63	12.42	9.543
5	RMSprop	4.434	6.313	7.820	8.493	9.190	7.250	5.595	7.679	9.407	10.13	10.99	8.969
10	RMSprop	4.320	6.303	7.375	8.332	9.003	7.067	5.494	7.765	8.964	10.09	10.67	8.792
10	RMSprop	4.913	6.214	7.420	8.364	9.356	7.253	6.094	7.609	9.056	10.09	11.04	8.952
10	RMSprop	4.525	6.363	7.681	8.301	9.322	7.238	5.726	7.728	9.229	10.12	10.94	8.940
10	RMSprop	4.659	6.660	7.716	8.711	9.503	7.450	5.928	8.040	9.278	10.41	11.16	9.151
10	RMSprop	4.548	6.829	7.398	8.091	9.041	7.181	5.747	8.469	9.071	9.894	10.68	8.934

Υπολογίζοντας την μέση τιμή των μετρικών για τις πέντε φορές που επαναλαμβάνεται κάθε σενάριο, τα αποτελέσματα συνοψίζονται στον επόμενο Πίνακα 3.4.

Πίνακας 3.4 Συνοπτική παρουσίαση συγκρίσεων των Univariate Multi-Step MLP μοντέλων

Lags	Αλγόρ. Βελτιστ.	Προβλεπτική Ακρίβεια (out-of sample)											
		MAE (Μέση Τιμή 5 Επαναλήψεων)						RMSE (Μέση Τιμή 5 Επαναλήψεων)					
k	*optimizer*	*1*	*2*	*3*	*4*	*5*	*Total*	*1*	*2*	*3*	*4*	*5*	*Total*
3	Adam	**4.577**	**6.304**	7.709	8.543	9.309	7.288	**5.634**	**7.677**	9.231	10.22	10.96	**8.953**
5	Adam	4.611	6.415	7.792	8.682	9.319	7.364	5.772	7.783	9.345	10.40	10.97	9.056
10	Adam	4.652	6.494	7.957	8.459	9.261	7.365	5.929	7.918	9.557	10.28	10.97	9.119
3	RMSprop	4.933	6.435	7.800	8.482	9.336	7.397	6.088	7.864	9.369	10.28	11.03	9.107
5	RMSprop	5.036	6.451	8.048	8.832	9.766	7.627	6.194	7.845	9.643	10.58	11.51	9.361
10	RMSprop	4.661	6.516	**7.554**	**8.367**	**9.305**	**7.281**	5.874	7.962	**9.159**	**10.12**	**10.95**	8.994

Σημείωση: Με έντονη γραφή δηλώνουμε το μοντέλο με την καλύτερη απόδοση για το συγκεκριμένο χρονικό βήμα και μετρική.

Παρατηρούμε ότι τα επικρατέστερα μοντέλα είναι αυτά με αλγόριθμο βελτιστοποίησης Adam και *k=3* (*Total MAE*: 7,288, *Total RMSE*: 8,953), καθώς και το μοντέλο με τον αλγόριθμο RMSprop και χρονική υστέρηση *k=10* (*Total MAE*: 7,281, *Total RMSE*: 8,994). Ενδιαφέρον παρουσιάζει το γεγονός ότι το πρώτο μοντέλο, που εστιάζει λιγότερο στο παρελθόν, προβλέπει με μεγαλύτερη ακρίβεια τις τιμές της χρονοσειράς για τα πρώτα δύο χρονικά βήματα, ενώ το δεύτερο μοντέλο, που εξετάζει τα δεδομένα μέχρι 10 βήματα στο παρελθόν, παρότι συγκριτικά με το πρώτο μοντέλο υστερεί στις βραχυπρόθεσμες προβλέψεις, πετυχαίνει πιο ακριβείς προβλέψεις για τους πιο μακρινούς ορίζοντες των τριών, τεσσάρων και πέντε ημερών στο μέλλον.

3.3.1.2 Multivariate MLP Μοντέλα

A) Multivariate One Step MLP Μοντέλα

Σε αυτήν την κατηγορία μοντέλων, χρησιμοποιούμε μία ή περισσότερες χρονοσειρές ως είσοδο και από αυτές εξαρτάται η (μοναδική) χρονοσειρά που παρέχεται ως έξοδος. Οι χρονοσειρές εισόδου εξελίσσονται παράλληλα στον χρόνο, καθώς οι παρατηρήσεις τους χαρακτηρίζουν τα ίδια χρονικά βήματα. Επομένως, αν θέλουμε να προβλέψουμε την τιμή της χρονοσειράς *Close_QQQ*, χρησιμοποιώντας επιπρόσθετα τις τιμές της χρονοσειράς *Close_MSFT* για την Microsoft, θα χρησιμοποιήσουμε ένα μοντέλο που λαμβάνει υπόψη και τις δύο αυτές χρονοσειρές κατά την πρόβλεψη. Για παράδειγμα, για την ακολουθία των πρώτων εφτά τιμών τους

$$Close\ QQQ \rightarrow [187.27, 184.05, 186.07, 188.81, 188.24, 185.42, 187.23, \dots,]$$

$$Close\ MSFT \rightarrow [137.07, 134.65, 136.28, 138.12, 137.12, 135.67, 138.24, \dots,]$$

αν $k = 3$, τότε τα δείγματα που θα κατασκευάσουμε θα είναι

$$[187.27, 184.05, 186.07, 137.07, 134.65, 136.28],\ [188.81]$$
$$[184.05, 186.07, 188.81, 134.65, 136.28, 138.12],\ [188.24]$$
$$[186.07, 188.81, 188.24, 136.28, 138.12, 137.12],\ [185.42]$$
$$[\ 188.81, 188.24, 185.42, 138.12, 137.12, 135.67],\ [187.23]$$
$$\vdots$$

όπου κάθε δείγμα έχει μήκος $3 \cdot 2 = 6$ στοιχεία (γινόμενο χρονικής υστέρησης με το πλήθος των διαφορετικών χρονοσειρών εισόδου). Σημειώνουμε ότι η διάταξη των στοιχείων εισόδου μέσα σε κάθε δείγμα δεν επηρεάζει τον υπολογισμό του σταθμισμένου αθροίσματος net. Ανεξάρτητα από τον τρόπο με τον οποίο οι όροι προστίθενται, το αποτέλεσμα παραμένει αμετάβλητο. Ωστόσο, από δείγμα σε δείγμα, πρέπει να διατηρείται συνοχή στη διάταξη των χαρακτηριστικών, προκειμένου τα βάρη να αντιστοιχίζονται ακριβώς στα ίδια χαρακτηριστικά κάθε φορά. Για παράδειγμα, θα μπορούσαμε εναλλακτικά να κατασκευάζαμε τα παραπάνω δείγματα ως εξής:

$$[187.27, 137.07, 184.05, 134.65, 186.07, 136.28\ ],\ [188.81]$$
$$[184.05, 134.65, 186.07, 136.28, 188.81, 138.12],\ [188.24]$$
$$[186.07, 136.28, 188.81, 138.12, 188.24, 137.12],\ [185.42]$$
$$[188.81, 138.12, 188.24, 137.12, 185.42, 135.67],\ [187.23]$$
$$\vdots$$

και τα αποτελέσματα των προβλέψεών μας δεν θα άλλαζαν με ουσιαστικό τρόπο.

Στη συνέχεια θα εκτελέσουμε τέσσερα διαφορετικά σενάρια νευρωνικών δικτύων που διαφοροποιούνται στην επιλογή των μεταβλητών που θα χρησιμοποιηθούν ως input για να προβλέψουμε την τιμή του QQQ για την επόμενη χρηματιστηριακή ημέρα. Πιο συγκεκριμένα οι μεταβλητές που θα χρησιμοποιήσουμε σε κάθε σενάριο φαίνεται παρακάτω

1. *Scenario 1*

   Close_QQQ, Open, High, Low (4 μεταβλητές)

   Τιμή κλεισίματος QQQ, τιμή ανοίγματος QQQ, μεγαλύτερη και χαμηλότερη τιμή QQQ. Αυτό θα αποτελέσει το *baseline* μοντέλο (μοντέλο αναφοράς) το οποίο θα επιχειρήσουμε να βελτιώσουμε.

2. *Scenario 2*

   Close_QQQ, Close_MSFT, Open, High, Low (5 μεταβλητές)

   Τιμή κλεισίματος QQQ και Microsoft, τιμή ανοίγματος QQQ, μεγαλύτερη και χαμηλότερη τιμή QQQ.

3. *Scenario 3*

   Close_QQQ, Close_MSFT, Close_AAPL, Close_TSLA, Open, High, Low

   Τιμή κλεισίματος QQQ, Microsoft, Apple και Tesla, τιμή ανοίγματος QQQ, μεγαλύτερη και χαμηλότερη τιμή QQQ.

4. *Scenario 4*

   Close_QQQ, Close_MSFT, Close_AAPL, Close_TSLA, Close_AMZN,

   Close_NVDA, , Close_GOOGL, Close_GOOG, Close_META, Close_AVGO, Open ,High, Low (13 μεταβλητές)

   Τιμή κλεισίματος QQQ, Microsoft, Apple, Tesla, Amazon, Nvidia, Google Class A shares, Google Class C shares, Meta και Broadcom, τιμή ανοίγματος QQQ, μεγαλύτερη και χαμηλότερη τιμή QQQ.

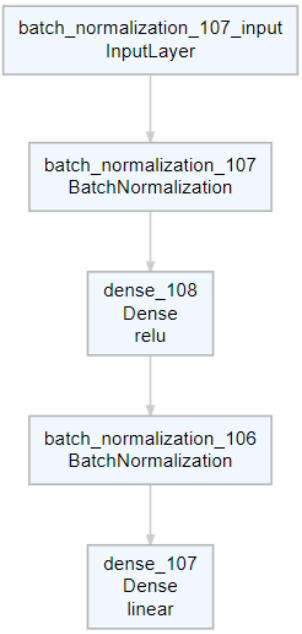
Σε κάθε σενάριο, το μοντέλο θα εκπαιδευτεί για 100 εποχές, χρησιμοποιώντας παρτίδες με μέγεθος 8. Θα πραγματοποιηθούν πέντε εκπαιδεύσεις με τον αλγόριθμο βελτιστοποίησης RMSprop και ρυθμό μάθησης 0.001. Η χρονική υστέρηση θα είναι $k = 3$, και το καλύτερο μοντέλο θα εκπαιδευτεί επίσης για $k = 10$.

Παρακάτω, ενδεικτικά παρουσιάζουμε το γράφημα που απεικονίζει την αρχιτεκτονική του δικτύου για το μοντέλο του Σεναρίου 3 (βλ. Σχήμα 3.12). Όπως βλέπουμε, το σήμα εισόδου αποτελείται από $3 \cdot 7 = 21$ τιμές. Το δίκτυο έχει μία κρυφή στιβάδα με 32 νευρώνες και συ-

νάρτηση ενεργοποίησης την ReLU και η στιβάδα που ακολουθεί μετά είναι η στιβάδα κανονικοποίησης. Επιπλέον, καθώς τώρα εισάγουμε πληροφορίες από περισσότερες από μία χρονοσειρές, και οι διακύμανση στα διανύσματα εισόδου αυξάνεται, προσθέτουμε μια επιπλέον στιβάδα κανονικοποίησης στην αρχή του δικτύου. Έτσι, κανονικοποιηούμε τα ανεπεξέργαστα δεδομένα εισόδου προτού εισαχθούν στο υπόλοιπο μοντέλο.

ΣΧΗΜΑ 3.12

Αρχιτεκτονική Multivariate One Step MLP Μοντέλου



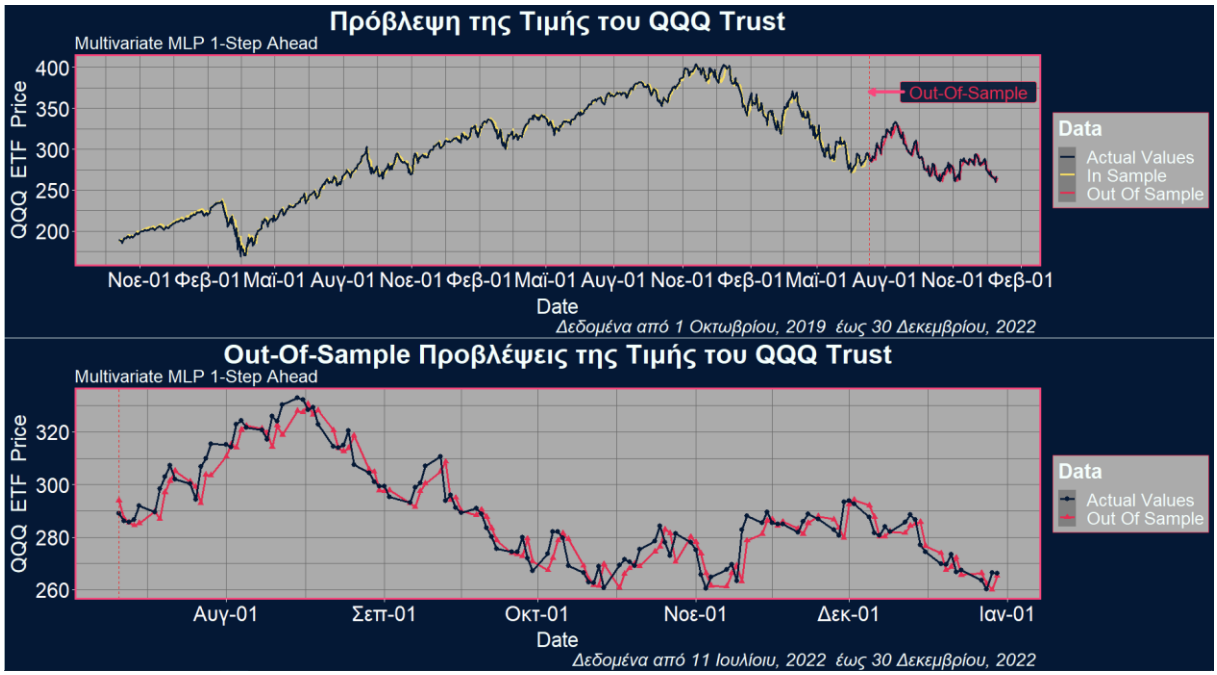
Layer (type)	Output Shape	Param #	Trainable
batch_normalization_105 (BatchNormalization)	(None, 21)	84	Y
dense_106 (Dense)	(None, 32)	704	Y
batch_normalization_104 (BatchNormalization)	(None, 32)	128	Y
dense_105 (Dense)	(None, 1)	33	Y

Total params: 949
Trainable params: 843
Non-trainable params: 106

Η απόδοση αυτού του μοντέλου φαίνεται στο παρακάτω Σχήμα 3.13.

ΣΧΗΜΑ 3.13

Απόδοση Multivariate One Step MLP Μοντέλου



Από το διάγραμμα βλέπουμε πως οι προβλέψεις του μοντέλου έχουν καταφέρει να ανταναακλούν σε μεγάλο βαθμό την πραγματική πορεία της τιμής του QQQ και τις αντίστοιχες διακυμάνσεις της, τόσο στα in sample όσο και στα out of sample δεδομένα.

Στον Πίνακα 3.5 φαίνονται αναλυτικά τα αποτελέσματα για κάθε σενάριο

Πίνακάς 3.5 Σύγκριση Multivariate One-Step MLP μοντέλων

Χρονική Υστέρηση	Μοντέλο Πρόβλεψης	Προβλεπτική Ακρίβεια (out-of sample)	
k	*Scenario*	*MAE*	*RMSE*
3	1	4.724613	5.767769
3	1	4.532358	5.684462
3	1	4.384699	5.576337
3	1	4.746253	5.730711
3	1	4.340295	5.561379
3	2	4.291122	5.427513
3	2	4.99256	6.448709
3	2	4.572556	5.658131
3	2	4.768447	5.920044
3	2	4.972423	6.122662
3	3	4.255977	5.446363

3	3	4.502479	5.637379
3	3	4.576725	5.727963
3	3	4.392602	5.50625
3	3	4.571862	5.729178
3	4	5.041178	6.108118
3	4	5.209782	6.421727
3	4	5.265083	6.521289
3	4	7.204208	8.91853
3	4	5.519476	6.811254
10	3	4.421766	5.715985
10	3	4.430902	5.649654
10	3	4.47722	5.706579
10	3	4.523963	5.790923
10	3	4.60385	5.743069
Scenario 1 Mean		4.5456436	5.6641316
Scenario 2 Mean		4.7194216	5.9154118
Scenario 3 Lag-3 Mean		**4.459929**	**5.6094266**
Scenario 4 Mean		5.6479454	6.9561836
Scenario 3 Lag-10 Mean		4.4915402	5.721242

Σημείωση: Με έντονη γραφή δηλώνουμε το μοντέλο με την καλύτερη απόδοση.

Από τα παραπάνω αποτελέσματα γίνεται φανερό πως η προσθήκη επιπλέον μεταβλητών στο νευρωνικό δίκτυο δεν εγγυάται απαραίτητα και βελτιωμένα αποτελέσματα, καθώς το Σενάριο 4 που περιείχε τις περισσότερες μεταβλητές επέφερε την χειρότερη απόδοση. Το πολυμεταβλητό μοντέλο του Σεναρίου 1 που αποτελείται από χαρακτηριστικά που σχετίζονται εξολοκλήρου με την συμπεριφορά του QQQ (Close_QQQ, Open, High ,Low) επιφέρει ικανοποιητικά αποτελέσματα, αλλά η προσθήκη των μεταβλητών για τις τιμές κλεισίματος της Microsoft, Apple και Tesla στο Σενάριο 3 βελτίωσε την απόδοση του μοντέλου. Τέλος το μοντέλου του Σεναρίου 3, για χρονική υστέρηση 10 βημάτων, αν και με ικανοποιητική απόδοση, δεν καταφέρνει να ξεπεράσει την απόδοση του ίδιου μοντέλου για χρονική υστέρηση ίση με 3.

Β) Multivariate Multi Step MLP Μοντέλα

Όπως είδαμε στα Univariate Multi Step MLP μοντέλα, η διαφορά από τα μοντέλα ενός βήματος είναι η αύξηση της διάστασης του output δείγματος. Για παράδειγμα, για την ακολουθία των πρώτων εφτά τιμών του QQQ και της Microsoft που είδαμε νωρίτερα

$$Close\ QQQ \rightarrow [187.27, 184.05, 186.07, 188.81, 188.24, 185.42, 187.23, \dots,]$$

$$Close\ MSFT \rightarrow [137.07, 134.65, 136.28, 138.12, 137.12, 135.67, 138.24, \dots,]$$

αν $k = 3$ και θέλουμε να προβλέψουμε την τιμή του QQQ για τις επόμενες δύο χρηματιστηριακές ημέρες, τότε τα δείγματα που θα κατασκευάσουμε θα είναι

$$[187.27, 184.05, 186.07, 137.07, 134.65, 136.28],\ [188.81, 188.24]$$

$$[184.05, 186.07, 188.81, 134.65, 136.28, 138.12],\ [188.24, 185.42]$$

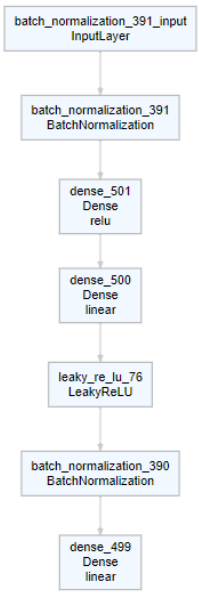
$$[186.07, 188.81, 188.24, 136.28, 138.12, 137.12],\ [185.42, 187.23]$$

$$\vdots$$

Στο Σχήμα 3.14 φαίνεται η αρχιτεκτονική του νευρωνικού δικτύου. Αποτελείται από δύο κρυφές στιβάδες, η πρώτη με 32 νευρώνες και συνάρτηση ενεργοποίησης την ReLU ενώ η δεύτερη αποτελείται από 16 νευρώνες και συνάρτηση ενεργοποίησης την LReLU με $\alpha = 0.001$. Μετά από τις δύο κρυφές στιβάδες ακολουθεί μια στιβάδα κανονικοποίησης, ενώ τα ανεπεξέργαστα δεδομένα έχουν επίσης κανονικοποιήθει στην αρχή του δικτύου.

ΣΧΗΜΑ 3.14

Αρχιτεκτονική Multivariate Multi Step MLP Μοντέλου



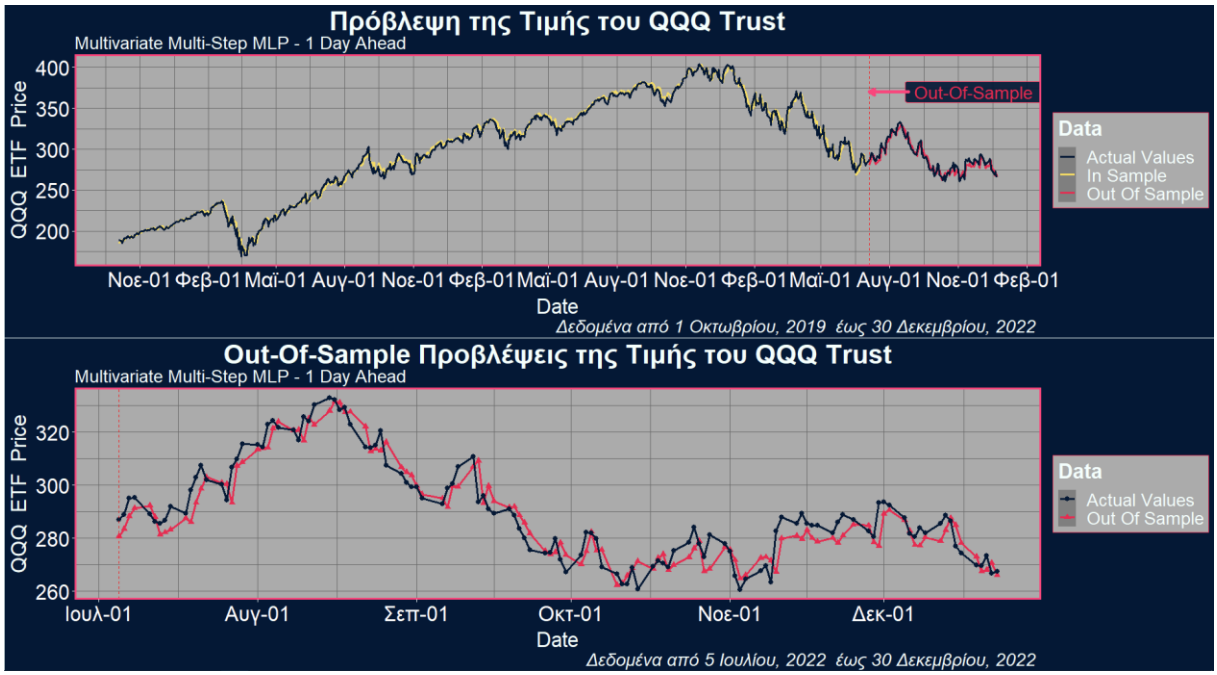
Layer (type)	Output Shape	Param #	Trainable
batch_normalization_391 (BatchNormalization)	(None, 3)	12	Y
dense_501 (Dense)	(None, 32)	128	Y
dense_500 (Dense)	(None, 16)	528	Y
leaky_re_lu_76 (LeakyReLU)	(None, 16)	0	Y
batch_normalization_390 (BatchNormalization)	(None, 16)	64	Y
dense_499 (Dense)	(None, 5)	85	Y

Total params: 817
Trainable params: 779
Non-trainable params: 38

Σημειώνουμε ότι ο αριθμός των παραμέτρων και η διάσταση του σήματος εισόδου που φαίνονται στον πίνακα του Σχήματος 3.14 αφορούν το Σενάριο 3. Η απόδοση αυτού του μοντέλου απεικονίζεται στο Σχήμα 3.15 και Σχήμα 3.16.

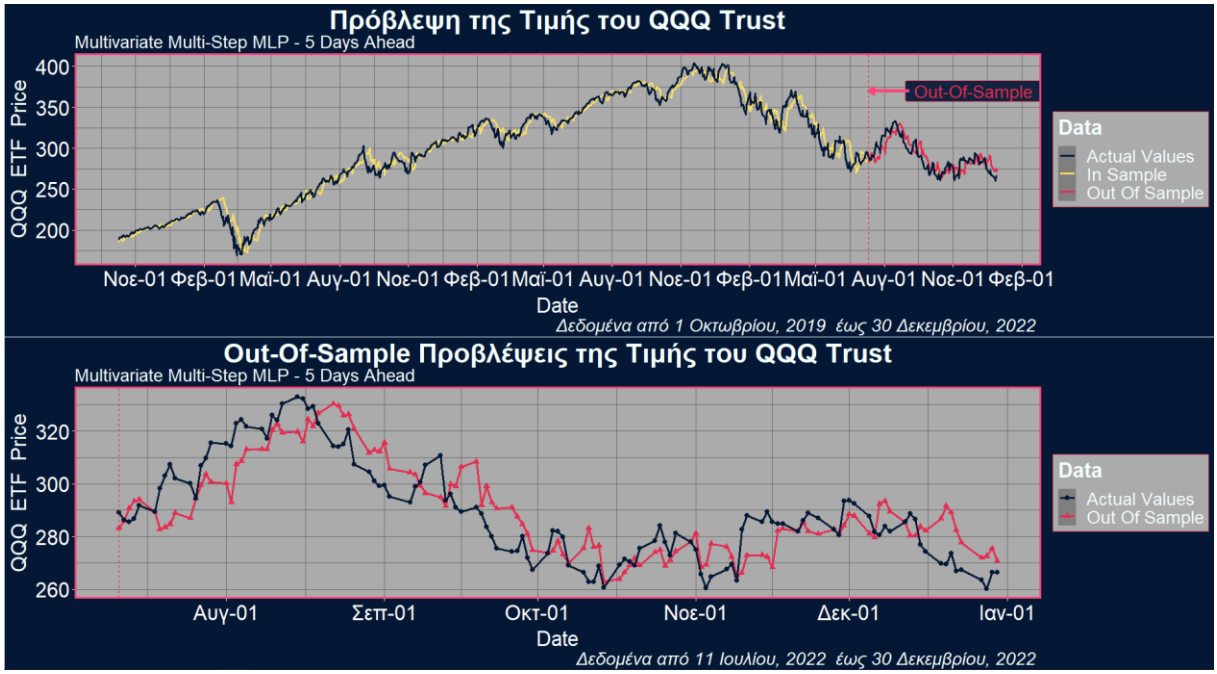
ΣΧΗΜΑ 3.15

Απόδοση Multivariate Mutli Step MLP για 1 Μέρα Μπροστά



ΣΧΗΜΑ 3.16

Απόδοση Multivariate Mutli Step MLP για 5 Μέρες Μπροστά



Στον παρακάτω πίνακα παρουσιάζουμε αναλυτικά τα αποτελέσματα από τις πέντε επαναλήψεις κάθε σεναρίου για την πρόβλεψη των επόμενων πέντε τιμών της χρονοσειράς QQQ.

Πίνακας 3.6 Σύγκριση Multivariate Multi-Step MLP μοντέλων

		Προβλεπτική Ακρίβεια (out-of sample)											
Lags	Model	MAE						RMSE					
k	*scenario*	*1*	*2*	*3*	*4*	*5*	*Total*	*1*	*2*	*3*	*4*	*5*	*Total*
3	1	4.595	6.302	7.852	8.495	9.181	7.285	5.666	7.574	9.234	10.21	10.83	8.902
3	1	4.335	6.230	7.636	8.480	9.048	7.146	5.584	7.567	9.144	10.24	10.84	8.882
3	1	4.673	6.745	8.206	8.946	10.16	7.746	5.760	8.098	9.689	10.70	12.02	9.503
3	1	4.955	6.353	7.690	8.483	9.175	7.331	6.404	7.662	9.088	10.18	10.88	8.992
3	1	4.533	6.239	8.028	9.114	9.652	7.513	5.635	7.496	9.570	10.88	11.37	9.244
3	2	4.415	6.152	7.952	8.058	8.951	7.106	5.439	7.390	9.683	9.681	10.63	8.771
3	2	4.290	7.781	7.874	8.482	9.205	7.526	5.542	9.797	9.427	10.24	10.88	9.369
3	2	4.717	6.736	8.130	8.302	9.163	7.410	5.957	8.104	9.896	9.899	10.75	9.085
3	2	4.334	6.808	7.586	8.190	9.528	7.289	5.520	8.209	9.068	9.935	11.59	9.091
3	2	4.597	6.957	9.248	8.328	9.161	7.658	5.857	8.595	11.29	10.14	10.85	9.551
3	3	4.857	6.380	7.861	9.074	9.391	7.513	6.046	7.760	9.379	10.93	11.18	9.266
3	3	4.509	6.167	7.744	8.419	9.925	7.353	5.576	7.549	9.231	10.12	11.75	9.097
3	3	4.520	6.438	7.714	8.343	9.102	7.223	5.793	7.702	9.398	10.18	10.82	8.965
3	3	4.454	6.189	7.790	8.520	9.141	7.219	5.508	7.536	9.256	10.28	10.78	8.885
3	3	4.321	6.370	7.978	8.588	9.185	7.288	5.397	7.692	9.572	10.38	10.97	9.033
3	4	5.145	7.227	8.071	8.964	9.553	7.792	6.695	9.034	9.755	10.77	11.80	9.766
3	4	5.431	7.099	8.078	9.549	9.847	8.001	6.803	8.773	10.19	11.46	11.70	9.952
3	4	5.993	7.357	8.579	9.929	9.910	8.354	7.269	8.947	10.46	12.11	11.90	10.30
3	4	7.628	9.344	9.505	10.13	10.83	9.488	9.417	11.62	11.56	12.35	13.27	11.71
3	4	6.749	10.08	10.43	10.68	11.06	9.800	8.795	12.54	12.45	12.53	12.87	11.93
10	3	5.113	7.042	7.680	8.596	9.169	7.520	6.415	8.647	9.338	10.60	10.89	9.316
10	3	4.792	6.393	7.464	8.213	8.981	7.169	6.073	7.717	8.916	9.821	10.66	8.786
10	3	4.742	6.623	7.937	8.592	9.505	7.480	5.951	7.900	9.356	10.45	11.13	9.147
10	3	5.294	6.243	7.955	9.116	9.741	7.670	6.809	7.741	9.533	10.83	11.37	9.422
10	3	4.907	6.399	7.804	8.541	9.898	7.510	5.958	7.776	9.400	10.10	11.60	9.175

Υπολογίζοντας την μέση τιμή των μετρικών για τις πέντε φορές που επαναλαμβάνεται κάθε σενάριο, τα αποτελέσματα συνοψίζονται στον επόμενο Πίνακα 3.7.

Πίνακας 3.7 Συνοπτική παρουσίαση συγκρίσεων των Multivariate Multi-Step MLP μοντέλων

		Προβλεπτική Ακρίβεια (out-of sample)											
Lags	Model.	MAE (Μέση Τιμή 5 Επαναλήψεων)						RMSE (Μέση Τιμή 5 Επαναλήψεων)					
k	*scenario*	*1*	*2*	*3*	*4*	*5*	*Total*	*1*	*2*	*3*	*4*	*5*	*Total*
3	1	4.618	6.374	7.883	8.704	9.443	7.404	5.810	7.679	9.345	10.44	11.18	9.104
3	2	**4.471**	6.887	8.158	**8.272**	**9.202**	7.398	**5.663**	8.419	9.873	**9.979**	**10.94**	9.173
3	3	4.532	**6.309**	**7.817**	8.589	9.349	**7.319**	5.664	**7.648**	9.367	10.37	11.10	**9.049**
3	4	6.189	8.221	8.933	9.851	10.24	8.687	7.796	10.18	10.88	11.84	12.30	10.73
10	3	4.969	6.540	7.768	8.612	9.459	7.470	6.241	7.956	**9.308**	10.35	11.13	9.169

Σημείωση: Με έντονη γραφή δηλώνουμε το μοντέλο με την καλύτερη απόδοση για το συγκεκριμένο χρονικό βήμα..

Από τα παραπάνω αποτελέσματα, παρατηρούμε ότι το μοντέλο του Σεναρίου 2, παρόλο που έχει το μικρότερο σφάλμα πρόβλεψης για την $1^{η}$, $4^{η}$ και $5^{η}$ μέρα, δεν είναι το βέλτιστο μοντέλο συνολικά λόγω της χαμηλής απόδοσής του για τις άλλες ημέρες. Αντίθετα, το μοντέλο του Σεναρίου 3 φαίνεται να έχει την πιο ισορροπημένη απόδοση για όλες τις ημέρες.

Επιπλέον, όταν εκτελέστηκε το μοντέλο του Σεναρίου 3 με χρονική υστέρηση ίση με 10 μέρες, παρατηρήθηκε καλύτερη απόδοση, με βάση το RMSE, που ήταν και η συνάρτηση απώλειας, για την $3^{η}$ και $4^{η}$ μέρα (για την $5^{η}$ μέρα η πρόβλεψη ήταν ελάχιστα χειρότερη) σε σύγκριση με το αντίστοιχο μοντέλο με χρονική υστέρηση $3^{ων}$ ημερών, υποδηλώνοντας πως το μοντέλο με μεγαλύτερη χρονική υστέρηση πετυχαίνει και πάλι πιο ακριβείς μακροπρόθεσμες προβλέψεις.

3.3.2 Συνελικτικά Νευρωνικά Δίκτυα

Αρχικά, θα παρουσιάσουμε τα μονομεταβλητά μοντέλα και στη συνέχεια θα εξετάσουμε τα πολυμεταβλητά μοντέλα. Τέλος, θα αναλύσουμε τα μοντέλα που χρησιμοποιούνται για την πρόβλεψη της κατεύθυνσης της τιμής του QQQ,.

3.3.2.1 Univariate CNN Μοντέλα

A) Univariate One Step CNN Μοντέλα

Τα CNN αυτού του είδους, με τις στιβάδες συνέλιξης, αναλύουν το μονοδιάστατο σήμα εισόδου με τον πυρήνα συνέλιξης. Αυτός, καθώς κινείται από τη μία άκρη των δεδομένων εισόδου προς την άλλη, εξετάζει τα δεδομένα σε μικρά τμήματα παρόμοια με τον τρόπο λειτουργίας των 2D CNN στις δισδιάστατες εικόνες. Με κάθε ανάλυση του τμήματος δημιουργείται μια ερμηνεία της εισόδου, η οποία στη συνέχεια αντιστοιχίζεται σε ένα feature map. Με αυ-

τόν τον τρόπο καταγράφονται διάφορα σημαντικά χαρακτηριστικά της ακολουθίας που παρέχεται ως είσοδος στο δίκτυο. Με άλλα λόγια, το CNN δεν βλέπει τα δεδομένα σαν μια ακολουθία που έχει χρονικά βήματα, αλλά τα αντιμετωπίζει σαν να μια "μονοδιάστατή εικόνα", πάνω στην οποία θα εκτελέσει την διαδικασία της συνέλιξης. Τα δεδομένα εισόδου αναπαρίστανται ως ένας τανυστής τριών διαστάσεων, σε αντίθεση με τους τανυστές τεσσάρων διαστάσεων που χρησιμοποιούνται στα 2D CNN. Συγκεκριμένα, η πρώτη διάσταση αντιστοιχεί στον αριθμό των δειγμάτων που έχουμε, η δεύτερη διάσταση αναφέρεται στη χρονική υστέριση, ενώ η τρίτη δηλώνει τον αριθμό των παράλληλων χρονοσειρών (δηλαδή μεταβλητές) που θα χρησιμοποιηθούν για τις προβλέψεις μας. Στην περίπτωση των μονομεταβλητών χρονοσειρών, αυτή η τρίτη διάσταση θα είναι ίση με 1, καθώς θα αξιοποιήσουμε μόνο τις τιμές κλεισίματος των προηγούμενων ημερών της χρονοσειράς QQQ για να προβλέψουμε την τιμή που θα λάβει την επόμενη χρηματιστηριακή ημέρα. Για παράδειγμα αν η χρονική υστέρηση είναι $k = 3$ τα δεδομένα εισόδου έχουν διάσταση $(samples, time\ steps, features) = (817,3,1)$ και η μορφή των input-output δειγμάτων για τις πρώτες έξι τιμές της χρονοσειράς QQQ

$$[187.27, 184.05, 186.07, 188.81, 188.24, 185.42, \dots,]$$

θα ήταν η παρακάτω

$$\begin{matrix}[[187.27] \\ [184.05] \\ [186.07]] & [188.81] \\ [[184.05] \\ [\ 186.07] \\ [188.81]] & [188.24] \\ [[186.07] \\ [188.81] \\ [188.24]] & [185.42] \\ \vdots & \end{matrix}$$

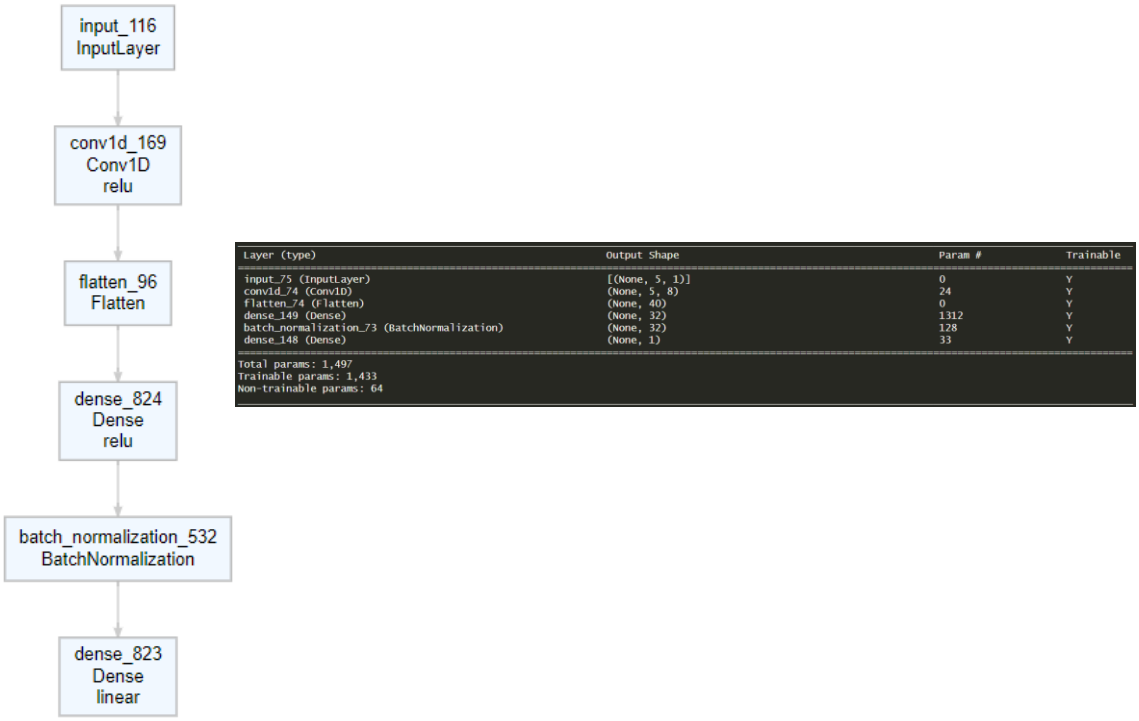
Η διάσταση των τιμών εισόδου πρέπει να δηλωθούν και στο νευρωνικό δίκτυο με το όρισμα `input_shape` $= (time\ steps, features) = (3,1)$.

Για τα μοντέλα που θα εφαρμόσουμε, θα χρησιμοποιήσουμε χρονική υστέρηση τριών, πέντε και δέκα χρονικών βημάτων και τους αλγορίθμους βελτιστοποίησης Adam και RMSprop με ρυθμό μάθησης $\eta = 0.001$. Για τις πρώτες δύο περιπτώσεις, το μέγεθος του πυρήνα συνέλιξης θα είναι $s = 2$, ενώ για τη χρονική υστέρηση δέκα βημάτων, θα είναι $s = 3$. Θα χρησι-

μοποιήσουμε μια στιβάδα συνέλιξης με 8 φίλτρα και την συνάρτηση ενεργοποίησης ReLU. Μετά από αυτή, ακολουθεί η διαδικασία του flattening, όπου το σήμα μετατρέπεται σε ένα μοναδικό διάνυσμα και περνά σαν είσοδος σε μια στιβάδα πλήρως συνδεδεμένων νευρώνων με 32 κόμβους και συνάρτηση ενεργοποίησης την ReLU. Τέλος, πριν το output layer υπάρχει μια στιβάδα κανονικοποίησης. Παρακάτω, παρουσιάζεται η αρχιτεκτονική του νευρωνικού δικτύου:

ΣΧΗΜΑ 3.17

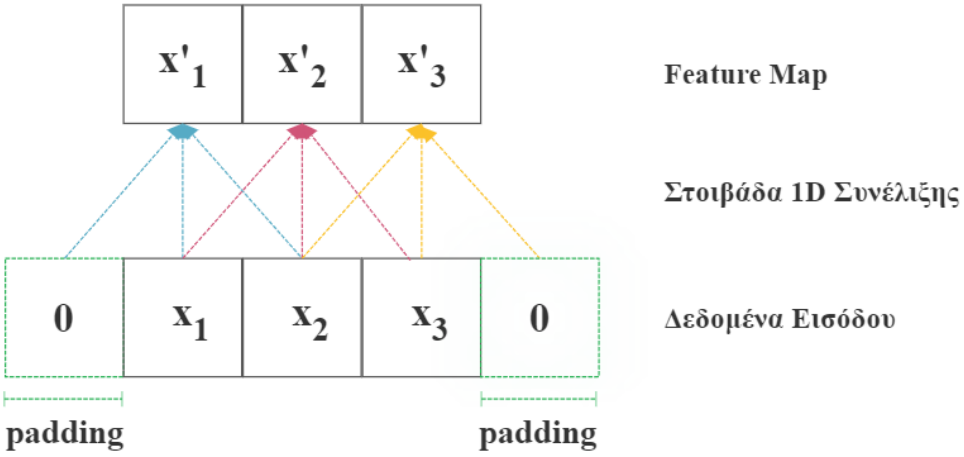
Αρχιτεκτονική Univariate One Step CNN Μοντέλων



Σημειώνουμε ότι στη στιβάδα συνέλιξής εφαρμόσαμε την τεχνική του αιτιώδους γεμίσματος (causal padding), όπου "γεμίζουμε" την ακολουθία εισόδου με $s-1$ μηδενικά στο ξεκίνημα της (όπου s είναι το μέγεθος του πυρήνα συνέλιξης). Με αυτόν τον τρόπο κάθε στοιχείο εξόδου της στιβάδας συνέλιξης εξαρτάται μόνο από τα προηγούμενα και το τρέχον στοιχείο. Αυτό διασφαλίζει ότι το δίκτυο δεν έχει πρόσβαση σε μελλοντικές πληροφορίες κατά την διάρκεια της εκπαίδευσης, διατηρώντας έτσι την αιτιότητα στα δεδομένα. Στα παρακάτω δύο σχήματα φαίνεται η διαφορά του causal padding από την κλασική μέθοδο γεμίσματος.

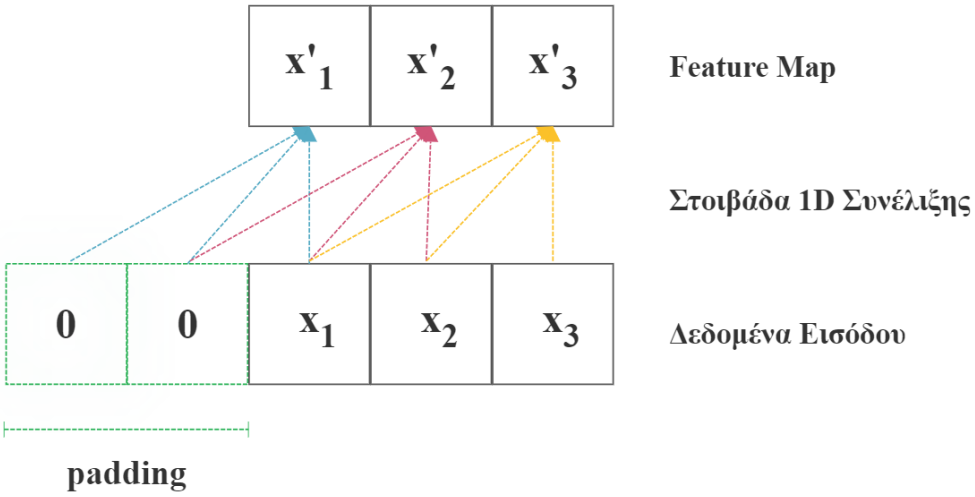
ΣΧΗΜΑ 3.18

Κλασική μέθοδος Padding



ΣΧΗΜΑ 3.19

Causal Padding



Στον παρακάτω πίνακα φαίνονται τα αποτελέσματα των μοντέλων που δοκιμάστηκαν.

Πίνακας 3.8 Σύγκριση Univariate One-Step CNN μοντέλων

Χρονική Υστέρηση	Αλγόριθμος Βελτι-στοποίησης	Προβλεπτική Ακρίβεια (out-of sample)	
k	*optimizer*	*MAE*	*RMSE*
3	Adam	4.446777	5.473499
3	Adam	4.421006	5.448093
3	Adam	4.501219	5.503738
3	Adam	4.377	5.520
3	Adam	4.685	5.713
5	Adam	4.270	5.358
5	Adam	4.483	5.468
5	Adam	4.312	5.391
5	Adam	4.585446	5.581615
5	Adam	4.450569	5.470035
10	Adam	4.470259	5.52139
10	Adam	4.560356	5.596975
10	Adam	4.519309	5.55286
10	Adam	4.346162	5.445253
10	Adam	4.291524	5.423557
3	RMSprop	4.352154	5.407429
3	RMSprop	4.751949	5.791421
3	RMSprop	4.344268	5.434042
3	RMSprop	4.609319	5.592298
3	RMSprop	4.529015	5.519909
5	RMSprop	4.068191	5.348939
5	RMSprop	4.234219	5.358335
5	RMSprop	4.257947	5.550213
5	RMSprop	4.218592	5.315744
5	RMSprop	4.315233	5.37973
10	RMSprop	4.819144	5.861154
10	RMSprop	4.240606	5.392079
10	RMSprop	4.339182	5.478175
10	RMSprop	4.35739	5.460094
10	RMSprop	4.963135	6.059522
Adam 3-Lags Mean		4.4860438	5.5318334
Adam 5-Lags Mean		4.420164	5.453666
Adam 10-Lags Mean		4.437522	5.508007
RMSprop 3-Lags Mean		4.517341	5.54902
RMSprop 5-Lags Mean		**4.218836**	**5.390592**
RMSprop 10-Lags Mean		4.543891	5.650205

Σημείωση: Με έντονη γραφή δηλώνουμε το μοντέλο με την καλύτερη απόδοση.

Από τον πίνακα βλέπουμε ότι η απόδοση όλων των μοντέλων είναι αρκετά κοντά μεταξύ τους, όμως το μοντέλο με χρονική υστέρηση 5 και τον αλγόριθμο βελτιστοποίησης RMSprop έχει την καλύτερη επίδοση. Παρατηρούμε επίσης ότι το μοντέλο με χρονική υστέρηση δέκα βημάτων, τόσο με τον αλγόριθμο Adam όσο και με τον RMSprop, εμφανίζει αρκετά καλή απόδοση σε κάποιες από τις επαναλήψεις του, και δύο εξ αυτών μάλιστα, πετυχαίνουν Μέσο Απόλυτο Σφάλμα κάτω από 4.3. Ωστόσο, φαίνεται ότι η απόδοση του είναι αρκετά ασταθής, καθώς εμφανίζει κάποιες επαναλήψεις με μεγάλα σφάλματα που επηρεάζουν την συνολική του επίδοση στο σύνολο των 5 επαναλήψεων. Για να επιβεβαιώσουμε αυτήν την παρατήρηση, θα μπορούσαμε να επαναλάβουμε το πείραμα με περισσότερες επαναλήψεις. Στο παρακάτω σχήμα φαίνεται η επίδοση του καλύτερου μοντέλου.

ΣΧΗΜΑ 3.20

Απόδοση Univariate One Step CNN Μοντέλου

Παρατηρούμε ότι το μοντέλο καταφέρνει να προβλέψει με αρκετά μεγάλη ακρίβεια τις πραγματικές τιμές της χρονοσειράς και αποτυπώνει ικανοποιητικά την συμπεριφορά και τις διακυμάνσεις της.

B) Univariate Multi Step CNN Μοντέλα

Όπως διαπιστώσαμε και νωρίτερα στη μετάβαση από τα μοντέλα Univariate One Step MLP στα Univariate Multi Step MLP, έτσι και στην περίπτωση των CNN προβλέψεων, υπάρχει μια ανάλογη διαφορά ανάμεσα στα Univariate CNN μοντέλα μιας και πολλαπλών προβλέψεων. Η διαφορά αυτή είναι ότι το output των Univariate CNN μοντέλων για πολλαπλά βήματα είναι ένα διάνυσμα που περιέχει περισσότερες από μία τιμές και αντιπροσωπεύει την πρόβλεψη για περισσότερα χρονικά βήματα. Για παράδειγμα, για την ακολουθία των πρώτων έξι τιμών της χρονοσειράς QQQ

$$[187.27, 184.05, 186.07, 188.81, 188.24, 185.42, \dots,]$$

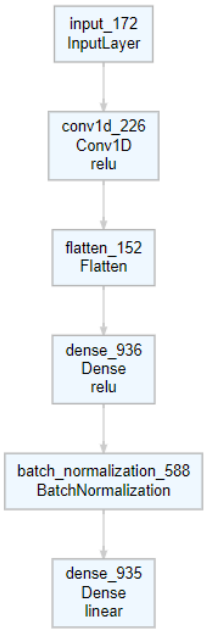
αν η $k = 3$, τότε τα δείγματα που θα κατασκευάζαμε θα ήταν

$$\begin{matrix} [[187.27] & \\ [184.05] & \\ [186.07]] & [188.81,188.24] \\ [[184.05] & \\ [\ 186.07] & \\ [188.81]] & [188.24, 185.42] \\ \vdots & \end{matrix}$$

Στο Σχήμα 3.21 παρουσιάζεται η δομή του νευρωνικού δικτύου. Οι στιβάδες που θα χρησιμοποιήσουμε είναι οι ίδιες με αυτές που βρίσκονταν στο προηγούμενο μοντέλο, που προέβλεπε μόνο την επόμενη χρηματιστηριακή μέρα, με τη διαφορά ότι τώρα αυξήσαμε τα φίλτρα στη στιβάδα συνέλιξης στα 32. Επιπλέον, σε αυτήν την περίπτωση, η στιβάδα εξόδου αποτελείται από 5 νευρώνες, έναν για κάθε επόμενη χρηματιστηριακή μέρα που προβλέπουμε. Λόγω αυτής της αλλαγής, ο αριθμός των παραμέτρων που εκπαιδεύονται στην στιβάδα εξόδου αυξάνεται σε 165 $(5 \cdot 32 + 5 \cdot 1)$, σε αντίθεση με τις 33 $(1 \cdot 32 + 1 \cdot 1)$ παραμέτρους που είχε το προηγούμενο μοντέλο.

ΣΧΗΜΑ 3.21

Αρχιτεκτονική Univariate One Step CNN Μοντέλων



Layer (type)	Output Shape	Param #	Trainable
input_172 (InputLayer)	[(None, 3, 1)]	0	Y
conv1d_226 (Conv1D)	(None, 3, 32)	96	Y
flatten_152 (Flatten)	(None, 96)	0	Y
dense_936 (Dense)	(None, 32)	3104	Y
batch_normalization_588 (BatchNormalization)	(None, 32)	128	Y
dense_935 (Dense)	(None, 5)	165	Y

Total params: 3,493
Trainable params: 3,429
Non-trainable params: 64

Στον παρακάτω πίνακα παρουσιάζονται αναλυτικά τα αποτελέσματα των μοντέλων που δοκιμάστηκαν.

Πίνακας 3.9 Σύγκριση Univariate Multi-Step CNN μοντέλων

Lags	Αλγόρ. Βελτιστ.	Προβλεπτική Ακρίβεια (out-of sample)											
		MAE						RMSE					
k	*optimizer*	*1*	*2*	*3*	*4*	*5*	*Total*	*1*	*2*	*3*	*4*	*5*	*Total*
3	Adam	4.568	6.350	7.689	8.404	9.029	7.208	5.834	7.744	9.155	10.13	10.79	8.910
3	Adam	4.406	6.281	7.745	8.499	9.445	7.275	5.453	7.618	9.188	10.20	11.24	8.975
3	Adam	4.637	6.392	7.619	8.350	9.028	7.205	5.988	7.892	9.210	10.12	10.77	8.961
3	Adam	4.473	6.216	7.603	8.409	9.171	7.174	5.539	7.549	9.048	10.09	10.84	8.819
3	Adam	4.320	6.228	7.618	8.363	9.100	7.126	5.394	7.614	9.126	10.08	10.78	8.812
5	Adam	4.354	6.193	7.666	8.316	9.017	7.109	5.667	7.585	9.134	10.01	10.81	8.833
5	Adam	4.360	6.278	7.615	8.428	9.106	7.157	5.397	7.597	9.081	10.08	10.77	8.797
5	Adam	4.288	6.218	7.632	8.335	8.957	7.086	5.445	7.574	9.163	10.09	10.71	8.802
5	Adam	4.406	6.249	7.623	8.347	8.993	7.124	5.852	7.693	9.137	10.08	10.71	8.869
5	Adam	4.563	6.260	7.615	8.359	8.996	7.159	5.917	7.756	9.199	10.06	10.73	8.901
10	Adam	4.302	6.359	7.587	8.544	9.183	7.195	5.491	7.801	9.093	10.41	10.83	8.938
10	Adam	4.362	6.133	7.475	8.319	9.009	7.060	5.725	7.531	9.019	10.06	10.62	8.774
10	Adam	4.303	6.167	7.537	8.181	9.051	7.048	5.576	7.646	9.048	9.910	10.72	8.770
10	Adam	4.719	6.452	7.606	8.297	9.072	7.229	6.084	7.881	9.160	10.04	10.78	8.944

10	Adam	4.364	6.212	7.583	8.397	9.069	7.125	5.727	7.584	9.105	10.11	10.70	8.831
3	RMSprop	4.675	6.328	7.609	8.410	9.080	7.221	5.926	7.599	9.226	10.04	10.83	8.901
3	RMSprop	4.693	6.829	8.217	8.453	9.415	7.522	5.662	8.195	9.820	10.06	11.07	9.158
3	RMSprop	4.275	6.180	7.579	8.355	9.141	7.106	5.510	7.535	9.086	10.08	10.84	8.817
3	RMSprop	4.317	6.263	7.628	8.373	9.093	7.135	5.417	7.595	9.125	10.08	10.77	8.810
3	RMSprop	4.401	6.257	7.657	8.443	9.280	7.208	5.439	7.564	9.099	10.12	10.98	8.860
5	RMSprop	4.330	6.160	7.597	8.307	9.117	7.102	5.582	7.536	9.170	10.06	10.80	8.830
5	RMSprop	4.612	6.258	7.648	8.564	9.274	7.272	5.607	7.555	9.070	10.27	10.89	8.887
5	RMSprop	4.478	6.192	7.700	8.604	9.220	7.239	5.502	7.529	9.146	10.34	10.84	8.889
5	RMSprop	4.641	6.463	7.979	8.839	9.344	7.453	5.639	7.788	9.502	10.59	10.96	9.109
5	RMSprop	4.430	6.211	7.610	8.422	9.118	7.158	5.533	7.542	9.046	10.05	10.74	8.782
10	RMSprop	4.215	6.089	7.531	8.579	9.128	7.108	5.539	7.567	9.059	10.33	10.80	8.870
10	RMSprop	4.378	6.191	7.520	8.588	9.174	7.170	5.508	7.637	9.058	10.26	10.87	8.878
10	RMSprop	4.455	6.266	7.571	8.691	9.078	7.212	5.633	7.704	9.108	10.58	10.78	8.968
10	RMSprop	4.339	6.111	7.517	8.527	9.074	7.114	5.744	7.599	9.088	10.23	10.75	8.871
10	RMSprop	4.568	6.350	7.689	8.404	9.029	7.208	5.834	7.744	9.155	10.13	10.79	8.910

Υπολογίζοντας την μέση τιμή των μετρικών για τις πέντε φορές που επαναλαμβάνεται κάθε σενάριο, τα αποτελέσματα συνοψίζονται στον επόμενο Πίνακα 3.10

Πίνακας 3.10 Συνοπτική παρουσίαση συγκρίσεων των Univariate Multi-Step CNN μοντέλων

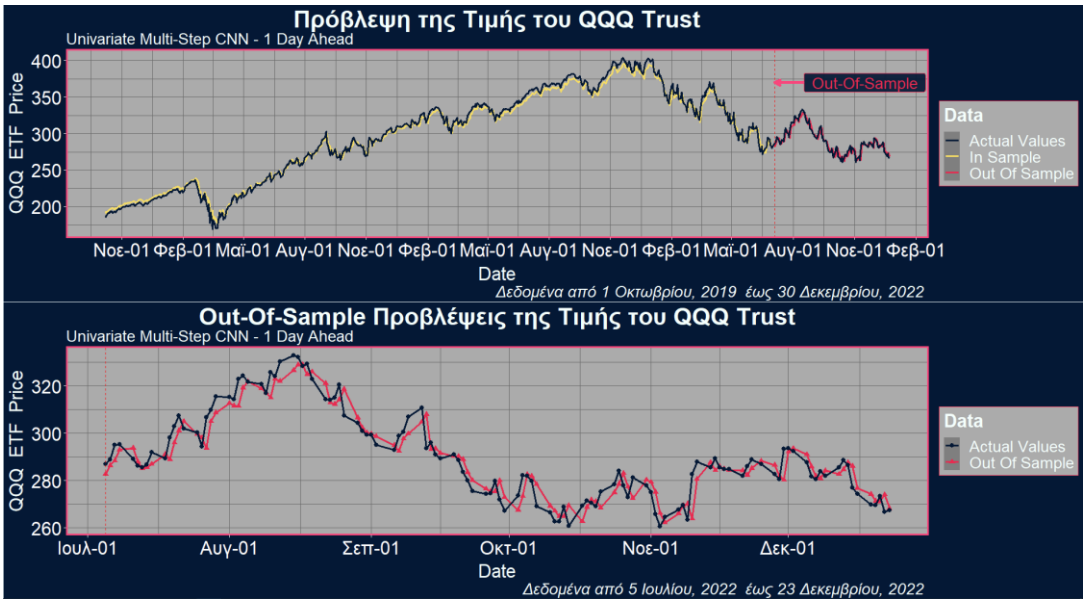
Lags	Αλγόρ. Βελτιστ.	Προβλεπτική Ακρίβεια (out-of sample)											
		MAE (Μέση Τιμή 5 Επαναλήψεων)						RMSE (Μέση Τιμή 5 Επαναλήψεων)					
k	*optimizer*	*1*	*2*	*3*	*4*	*5*	*Total*	*1*	*2*	*3*	*4*	*5*	*Total*
3	Adam	4.481	6.293	7.655	8.405	9.155	7.198	5.642	7.683	9.146	10.13	10.89	8.895
5	Adam	4.394	6.239	7.630	8.357	**9.014**	**7.127**	5.656	7.641	9.143	**10.06**	10.75	**8.840**
10	Adam	4.410	6.265	7.558	**8.348**	9.077	7.131	5.721	7.689	9.085	10.11	**10.73**	8.852
3	RMSprop	4.472	6.371	7.738	8.407	9.202	7.238	5.591	7.698	9.271	10.08	10.90	8.909
5	RMSprop	4.498	6.257	7.707	8.547	9.215	7.245	**5.573**	7.590	9.187	10.26	10.85	8.899
10	RMSprop	**4.352**	**6.176**	**7.519**	8.512	9.109	7.133	5.613	**7.625**	**9.068**	10.27	10.79	8.873

Σημείωση: Με έντονη γραφή δηλώνουμε το μοντέλο με την καλύτερη απόδοση για το συγκεκριμένο χρονικό βήμα και μετρική.

Παρατηρούμε ότι το μοντέλο με χρονική υστέρηση 5 βημάτων και αλγόριθμο βελτιστοποίησης Adam παρουσιάζει συνολικά την καλύτερη απόδοση. Το μοντέλο με χρονική υστέρηση 10 βημάτων και αλγόριθμο βελτιστοποίησης Adam, επίσης, αποδίδει πολύ καλά αποτελέσματα και αποτελεί το δεύτερο καλύτερο μοντέλο. Η απόδοση του βέλτιστου μοντέλου απεικονίζεται στα παρακάτω γραφήματα.

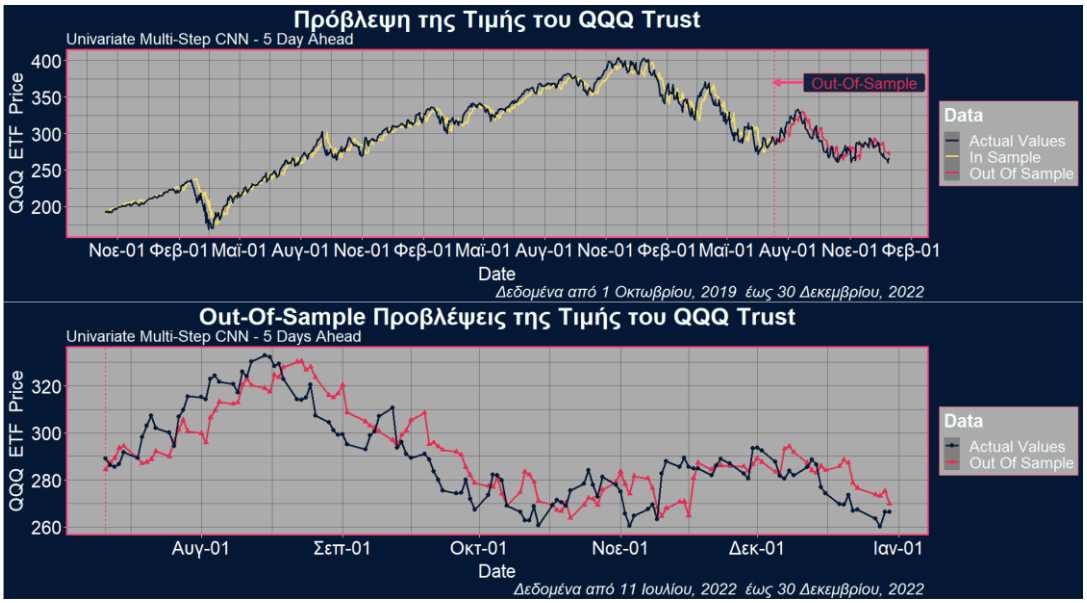
ΣΧΗΜΑ 3.22

Απόδοση Univariate Mutli Step CNN για 1 Μέρα Μπροστά



ΣΧΗΜΑ 3.23

Απόδοση Univariate Mutli Step CNN για 5 Μέρες Μπροστά



Από τα διαγράμματα παρατηρούμε ότι το μοντέλο προβλέπει τις πραγματικές τιμές του QQQ με αρκετά υψηλή ακρίβεια, ιδιαίτερα όσον αφορά τις τιμές της επόμενης χρηματιστηριακής ημέρας. Ωστόσο, για την χρηματιστηριακή ημέρα που ακολουθεί μετά από πέντε βήματα, το μοντέλο παρά την ικανότητά του να ακολουθήσει την γενική πορεία του QQQ στο χρονικό

διάστημα από την αρχή έως τα μέσα του Νοεμβρίου, αποκλίνει σημαντικά από τις πραγματικές τιμές.

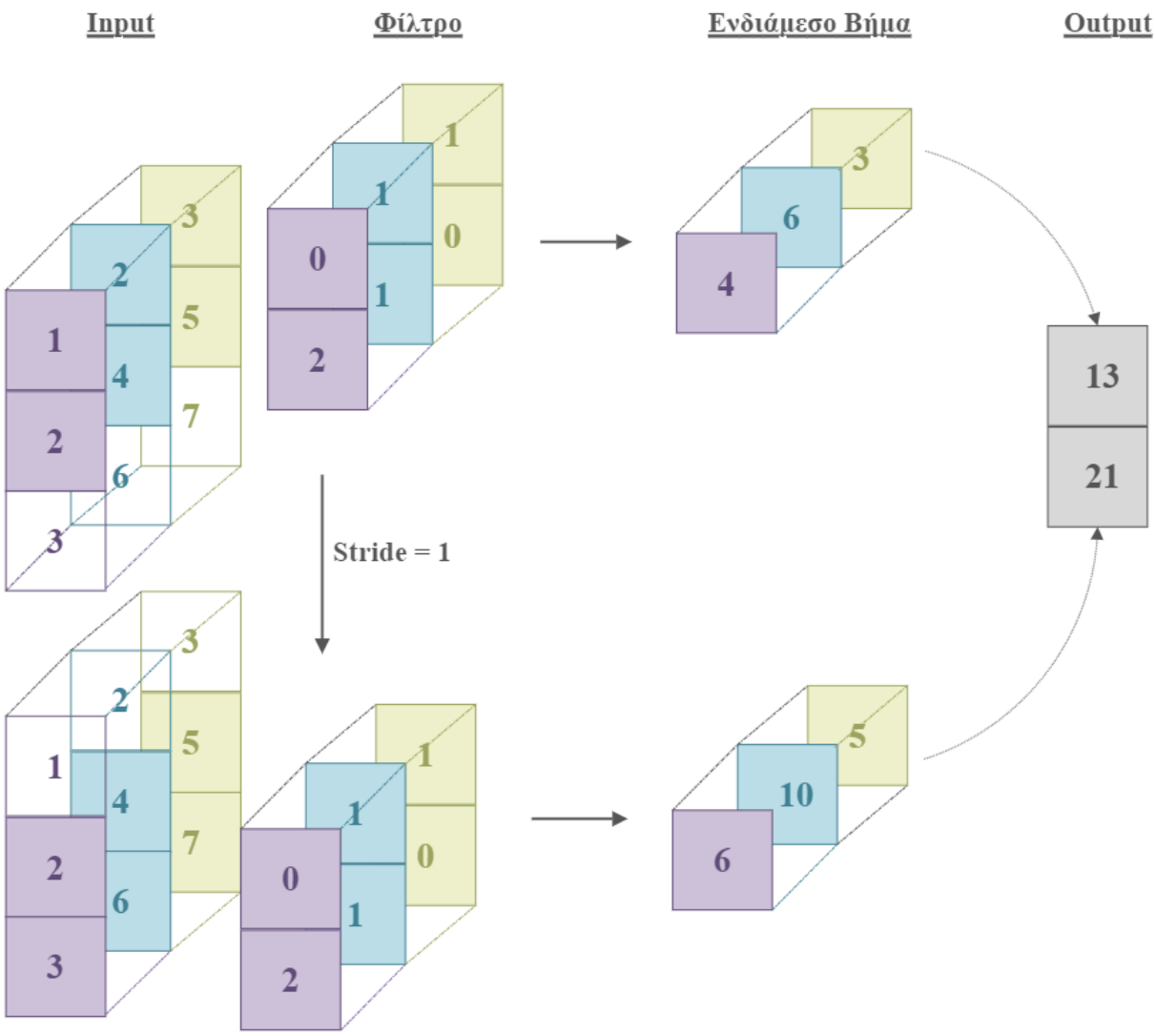
3.3.2.2 Multivariate CNN Μοντέλα

A) Multivariate One Step CNN Μοντέλα

Τα Συνελικτικά Νευρωνικά Δίκτυα όπως και τα MLP, δέχονται πολλαπλές χρονοσειρές ως είσοδο, οι οποίες εξελίσσονται παράλληλα στο χρόνο. Στα CNN αυτό επιτυγχάνεται με τη χρήση καναλιών, όπου το μοντέλο αντιλαμβάνεται κάθε χρονοσειρά ως μια διαφορετική διάσταση (κανάλι) και έτσι, αντί να αποτυπώνεται για παράδειγμα μια εικόνα με κόκκινο, πράσινο και μπλε κανάλι, όπως γίνεται στα CNN δυο διαστάσεων για την κατηγοριοποίηση εικόνων, κάθε κανάλι τώρα αντιπροσωπεύει μια μονοδιάστατη χρονοσειρά. Με αυτόν τον τρόπο το δίκτυο αποσπάει πληροφορίες από κάθε χρονοσειρά εισόδου. Στο παρακάτω σχήμα φαίνεται πως πραγματοποιείται η διαδικασία της συνέλιξης για τις τρεις παράλληλες χρονοσειρές $(1,2,3), (2,4,6)$ και $(3,5,7)$.

ΣΧΗΜΑ 3.24

Διαδικασία Συνέλιξης σε Μία Διάσταση



Όπως βλέπουμε σε κάθε κανάλι αντιστοιχεί ένας ξεχωριστός πυρήνας του φίλτρου, όπου καθένας χαρακτηρίζεται από τα δικά του βάρη. Στο ενδιάμεσο βήμα βλέπουμε τα βάρη να πολλαπλασιάζονται με το αντίστοιχο στοιχείο τους, στα δεδομένα εισόδου, και στη συνέχεια τα γινόμενα αυτά προστίθενται. Τελικά, αθροίζοντας κάθε στοιχείο στο ενδιάμεσο βήμα (που προκύπτουν από την εφαρμογή της προηγούμενης διαδικασίας σε κάθε κανάλι) παίρνουμε την τελική τιμή του στοιχείου στο output. Οι πράξεις του παραδείγματος φαίνονται παρακάτω.

$$\mathbf{0}\cdot 1+\mathbf{2}\cdot 2+\mathbf{1}\cdot 2+\mathbf{1}\cdot 4+\mathbf{1}\cdot 3+\mathbf{0}\cdot 5=4+6+3=13$$

$$\mathbf{0}\cdot 2+\mathbf{2}\cdot 3+\mathbf{1}\cdot 4+\mathbf{1}\cdot 6+\mathbf{1}\cdot 5+\mathbf{0}\cdot 7=6+10+5=21$$

Έτσι, αν θέλουμε να αξιοποιήσουμε, πέρα από τις τιμές κλεισίματος του QQQ, και τις τιμές κλεισίματος της μετοχής της Microsoft,

$$Close\ QQQ \rightarrow [187.27, 184.05, 186.07, 188.81, 188.24, 185.42, \dots,]$$

$$Close\ MSFT \rightarrow [137.07, 134.65, 136.28, 138.12, 137.12, 135.67, \dots,]$$

θα χρησιμοποιήσουμε δύο κανάλια και για χρονική υστέρηση $k = 3$ τα δεδομένα εισόδου θα έχουν διάσταση $(samples, time\ steps, features) = (817,3,2)$ και η μορφή των δειγμάτων θα ήταν η παρακάτω

$$\begin{matrix}[[187.27,137.07] \\ [184.05, 134.65] \\ [186.07,136.28]] \quad [188.81]\end{matrix}$$

$$\begin{matrix}[[184.05,134.65] \\ [\ 186.07,136.28] \\ [188.81, 138.12]] \quad [188.24]\end{matrix}$$

$$\begin{matrix}[[186.07,136.28] \\ [188.81, 138.12] \\ [188.24,137.12]] \quad [185.42]\end{matrix}$$

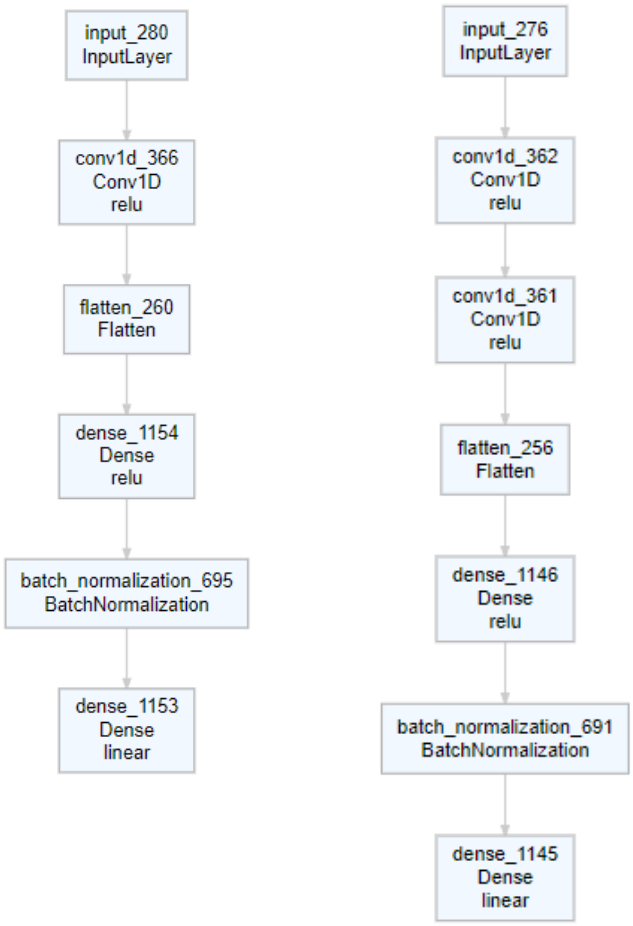
$$\vdots$$

και η διάσταση των τιμών εισόδου που θα δηλωνόταν στο νευρωνικό δίκτυο θα ήταν `input_shape=`$(time\ steps, features) = (3,2)$.

Στον πίνακα 3.11 παρουσιάζονται τα αποτελέσματα για τα σενάρια (βλ. <u>Ενότητα 3.3.1.2 - Α</u>) που αναφέραμε στα Multivariate One Step MLP μοντέλα. Αυτή τη φορά, χρησιμοποιήθηκε ο αλγόριθμος βελτιστοποίησης Adam, καθώς απέδειξε καλύτερη απόδοση στις δοκιμές μας. Η χρονική υστέρηση είναι $k = 3$ και το καλύτερο μοντέλο εκπαιδεύτηκε επίσης για $k = 10$. Για τα μοντέλα με $k = 3$, ο πυρήνας συνέλιξης έχει μήκος 2, και η στιβάδα συνέλιξης αποτελείται από 32 φίλτρα εκτός από το μοντέλο για το Σενάριο 4 όπου αποτελείται από δύο στιβάδες συνέλιξης, η πρώτη με 16 και η δεύτερη με 32 φίλτρα. Στο μοντέλο με $k = 10$ χρησιμοποιήθηκαν επίσης 2 στιβάδες συνέλιξης, με 16 και 32 φίλτρα, ενώ ο πυρήνας συνέλιξης έχει μήκος 3. Στο Σχήμα 3.25 απεικονίζονται οι δύο αυτές αρχιτεκτονικές του σεναρίου 1 για $k = 3$ και για $k = 10$ (μιας και αυτό το μοντέλο πέτυχε την καλύτερη απόδοση),

ΣΧΗΜΑ 3.25

Αρχιτεκτονική Multivariate One Step CNN Μοντέλων



Layer (type)	Output Shape	Param #	Trainable
input_280 (InputLayer)	[(None, 3, 4)]	0	Y
conv1d_366 (Conv1D)	(None, 3, 32)	288	Y
flatten_260 (Flatten)	(None, 96)	0	Y
dense_1154 (Dense)	(None, 16)	1552	Y
batch_normalization_695 (BatchNormalization)	(None, 16)	64	Y
dense_1153 (Dense)	(None, 1)	17	Y

Total params: 1,921
Trainable params: 1,889
Non-trainable params: 32

Layer (type)	Output Shape	Param #	Trainable
input_276 (InputLayer)	[(None, 10, 4)]	0	Y
conv1d_362 (Conv1D)	(None, 10, 16)	208	Y
conv1d_361 (Conv1D)	(None, 10, 32)	1568	Y
flatten_256 (Flatten)	(None, 320)	0	Y
dense_1146 (Dense)	(None, 16)	5136	Y
batch_normalization_691 (BatchNormalization)	(None, 16)	64	Y
dense_1145 (Dense)	(None, 1)	17	Y

Total params: 6,993
Trainable params: 6,961
Non-trainable params: 32

Στον Πίνακα 3.11 φαίνονται τα αποτελέσματα για κάθε σενάριο.

Πίνακας 3.11 Σύγκριση Multivariate One-Step MLP μοντέλων

Χρονική Υστέρηση *k*	Μοντέλο Πρόβλεψης *Scenario*	Προβλεπτική Ακρίβεια (out-of sample)	
		MAE	*RMSE*
3	1	4.227831	5.415875
3	1	4.355655	5.504388
3	1	4.433199	5.561895
3	1	4.282139	5.585269
3	1	4.382976	5.799642
3	2	4.436701	5.557962
3	2	4.785871	5.852935
3	2	4.277087	5.430156
3	2	4.655854	5.677106
3	2	5.500681	4.461222
3	3	4.611634	5.685312
3	3	4.69768	5.822792
3	3	5.085719	6.103173
3	3	4.692799	5.852929
3	3	4.653156	5.714885
3	4	5.1101	6.219992
3	4	4.314663	5.529416
3	4	5.388217	6.425992
3	4	5.425872	6.60337
3	4	5.574306	7.032429
10	1	4.287756	5.437489
10	1	4.463343	5.526195
10	1	4.381466	5.4426
10	1	4.666663	5.735455
10	1	4.536355	5.558387
Scenario 1 Lag-3 Mean		**4.33636**	**5.573414**
Scenario 2 Mean		4.731239	5.395876
Scenario 3 Mean		4.748198	5.835818
Scenario 4 Mean		5.162632	6.36224
Scenario 1 Lag-10 Mean		4.467117	5.540025

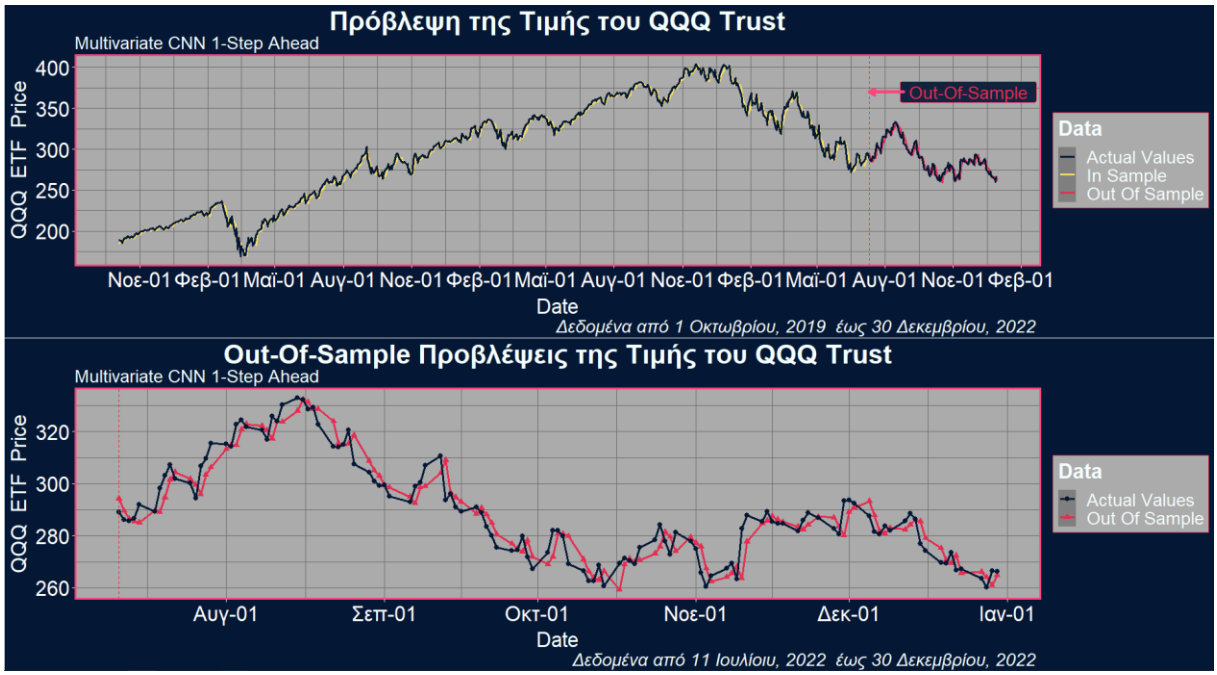
Σημείωση: Με έντονη γραφή δηλώνουμε το μοντέλο με την καλύτερη απόδοση.

Παρατηρούμε ότι για το Σενάριο 1, το μοντέλο πέτυχε το χαμηλότερο MAE και είχε επίσης αρκετά καλή απόδοση σύμφωνα με το RMSE. Το Σενάριο 2, αν και είχε ελαφρώς χαμηλότερο RMSE, είχε αρκετά υψηλό MAE, οπότε δεν θα το επιλέξουμε ως το καλύτερο μοντέλο.

Τέλος, το Σενάριο 1 με χρονική υστέρηση 10 βημάτων εμφάνισε ικανοποιητική απόδοση, και το RMSE ήταν μικρότερο από αυτό του μοντέλου με χρονική υστέρηση $3^{ων}$ βημάτων. Στο παρακάτω σχήμα φαίνεται η απόδοση του μοντέλου για το Σενάριο 1 με χρονική υστέρηση $k = 3$.

ΣΧΗΜΑ 3.26

Απόδοση Multivariate One Step CNN Μοντέλου



Από το διάγραμμα βλέπουμε πως το μοντέλο προβλέπει με αρκετή ακρίβεια της πορεία και τις διακυμάνσεις των πραγματικών τιμών της χρονοσειράς QQQ Trust.

Β) Multivariate Multi Step CNN Μοντέλα

Όμοια με τις κατηγορίες Multi step μοντέλων που είδαμε νωρίτερα, η διαφορά από τα μοντέλα ενός βήματος είναι η αύξηση της διάστασης του output δείγματος. Για παράδειγμα, για την ακολουθία των πρώτων έξι τιμών του QQQ και της Microsoft που είδαμε νωρίτερα

$$Close\ QQQ \rightarrow [187.27, 184.05, 186.07, 188.81, 188.24, 185.42, \dots,]$$

$$Close\ MSFT \rightarrow [137.07, 134.65, 136.28, 138.12, 137.12, 135.67, \dots,]$$

αν $k = 3$ και θέλουμε να προβλέψουμε την τιμή του QQQ για τις επόμενες δύο χρηματιστηριακές ημέρες, τότε τα δείγματα που θα κατασκευάσουμε θα είναι

$$\begin{matrix}[[187.27,137.07] \\ [184.05, 134.65] \\ [186.07,136.28]] \end{matrix} \quad [188.81,188.24]$$

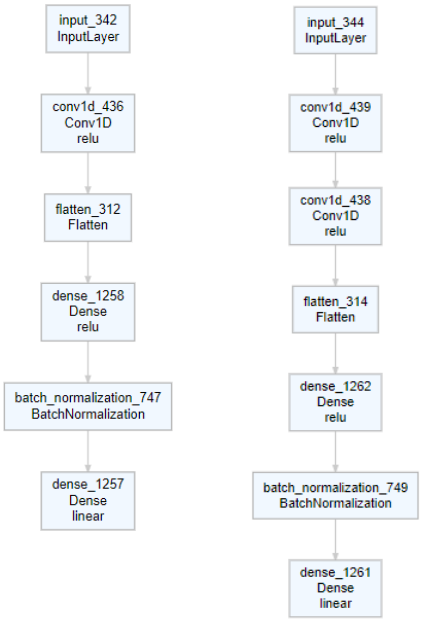
$$\begin{matrix}[[184.05,134.65] \\ [\ 186.07,136.28] \\ [188.81, 138.12]] \end{matrix} \quad [188.24,185.42]$$

$$\vdots$$

Για την κατασκευή των νευρωνικών δικτύων χρησιμοποιήθηκαν τα γνωστά σενάρια (βλ. Ενότητα 3.3.1.2 - Α) και η ίδια αρχιτεκτονική με τα Multivariate One Step CNN μοντέλα, με τη διαφορά ότι, στις δύο περιπτώσεις όπου χρησιμοποιήθηκαν δύο στιβάδες συνέλιξης (Σενάριο 4 και Σενάριο με χρονική υστέρηση $k = 10$), κάθε μία από αυτές είχε 32 φίλτρα (αντί για 16 και 32). Στο παρακάτω σχήμα φαίνεται η αρχιτεκτονική του Σεναρίου 2 για $k = 3$ και $k = 10$.

ΣΧΗΜΑ 3.27

Αρχιτεκτονική Multivariate Multi Step CNN Μοντέλων



Layer (type)	Output Shape	Param #	Trainable
input_342 (InputLayer)	[(None, 3, 5)]	0	Y
conv1d_436 (Conv1D)	(None, 3, 32)	352	Y
flatten_312 (Flatten)	(None, 96)	0	Y
dense_1258 (Dense)	(None, 16)	1552	Y
batch_normalization_747 (BatchNormalization)	(None, 16)	64	Y
dense_1257 (Dense)	(None, 5)	85	Y

Total params: 2,053
Trainable params: 2,021
Non-trainable params: 32

```
Layer (type)                                 Output Shape              Param #   Trainable
=================================================================
input_344 (InputLayer)                       [(None, 10, 5)]           0         Y
conv1d_439 (Conv1D)                          (None, 10, 32)            512       Y
conv1d_438 (Conv1D)                          (None, 10, 32)            2080      Y
flatten_314 (Flatten)                        (None, 320)               0         Y
dense_1262 (Dense)                           (None, 16)                5136      Y
batch_normalization_749 (BatchNormalization) (None, 16)                64        Y
dense_1261 (Dense)                           (None, 5)                 85        Y
=================================================================
Total params: 7,877
Trainable params: 7,845
Non-trainable params: 32
_________________________________________________________________
```

Στον παρακάτω πίνακα παρουσιάζουμε αναλυτικά τα αποτελέσματα από τις πέντε επαναλήψεις κάθε σεναρίου για την πρόβλεψη των επόμενων πέντε τιμών της χρονοσειράς QQQ.

Πίνακας 3.12 Σύγκριση Multivariate Multi-Step MLP μοντέλων

Lags	Model	Προβλεπτική Ακρίβεια (out-of sample)											
		MAE						RMSE					
k	*scenario*	*1*	*2*	*3*	*4*	*5*	*Total*	*1*	*2*	*3*	*4*	*5*	*Total*
3	1	4.383	6.240	7.628	8.512	9.127	7.178	5.565	7.500	9.065	10.14	10.81	8.821
3	1	4.394	6.302	7.605	8.405	9.265	7.194	5.644	7.610	9.123	10.16	10.98	8.908
3	1	4.577	6.482	7.772	8.393	9.104	7.266	5.689	7.982	9.138	10.04	10.78	8.907
3	1	4.359	6.330	7.649	8.491	9.154	7.197	5.456	7.667	9.076	10.11	10.83	8.836
3	1	4.632	6.359	7.608	8.318	9.023	7.188	6.054	7.696	9.237	9.993	10.70	8.895
3	2	4.245	7.680	7.680	8.322	9.525	7.229	5.583	7.644	9.321	10.04	11.30	9.001
3	2	4.310	6.240	7.628	8.333	9.096	7.121	5.787	7.538	9.195	10.10	10.79	8.867
3	2	4.445	6.362	7.679	8.254	8.974	7.143	5.500	7.656	9.306	9.997	10.61	8.809
3	2	4.209	6.274	7.602	8.388	9.036	7.102	5.447	7.610	9.055	10.07	10.69	8.778
3	2	4.187	6.248	7.527	8.353	9.005	7.064	5.320	7.598	8.982	10.06	10.71	8.748
3	3	4.592	6.028	7.210	7.887	8.858	6.915	5.938	7.451	8.844	9.668	10.52	8.639
3	3	4.528	6.379	7.709	8.249	9.171	7.207	5.692	7.693	9.188	9.985	10.91	8.884
3	3	4.606	6.355	7.802	8.425	9.040	7.245	5.733	7.649	9.322	10.00	10.69	8.862
3	3	5.264	6.714	7.885	8.922	9.647	7.686	6.681	8.152	9.421	10.74	11.46	9.449
3	3	5.685	7.176	8.348	9.163	10.41	8.157	6.799	8.537	9.917	11.03	12.19	9.877
3	4	5.916	7.231	8.103	8.792	9.780	7.964	7.478	8.903	9.796	10.72	11.82	9.857
3	4	5.476	7.963	8.930	9.076	9.662	8.221	6.845	9.983	11.16	10.91	11.62	10.25
3	4	5.879	6.886	8.467	8.694	9.602	7.906	7.476	8.515	10.23	10.73	11.58	9.822
3	4	5.342	6.856	9.603	8.410	11.20	8.282	6.560	8.305	11.50	10.24	13.35	10.27
3	4	5.024	6.966	7.858	8.647	9.786	7.656	6.256	8.306	9.312	10.40	11.86	9.420
10	2	4.633	6.149	7.576	8.193	8.907	7.092	6.059	7.539	9.111	10.02	10.68	8.843
10	2	4.207	6.076	7.611	8.273	9.118	7.057	5.303	7.411	9.013	9.991	10.90	8.751
10	2	4.949	6.884	7.807	8.295	9.174	7.422	6.345	8.782	9.358	10.02	10.96	9.225
10	2	4.423	6.205	7.566	8.147	8.969	7.062	5.716	7.746	9.000	9.885	10.70	8.784
10	2	4.718	6.219	7.579	8.261	9.172	7.190	6.091	7.630	9.059	9.987	10.80	8.875

Υπολογίζοντας την μέση τιμή των μετρικών για τις πέντε φορές που επαναλαμβάνεται κάθε σενάριο, τα αποτελέσματα συνοψίζονται στον επόμενο Πίνακα 3.13.

Πίνακας 3.13 Συνοπτική παρουσίαση συγκρίσεων των Multivariate Multi-Step MLP μοντέλων

		Προβλεπτική Ακρίβεια (out-of sample)											
Lags	Model.	MAE (Μέση Τιμή 5 Επαναλήψεων)						RMSE (Μέση Τιμή 5 Επαναλήψεων)					
k	*scenario*	*1*	*2*	*3*	*4*	*5*	*Total*	*1*	*2*	*3*	*4*	*5*	*Total*
3	1	4.469	6.342	7.652	8.424	9.135	7.204	5.682	7.691	9.128	10.09	10.82	8.874
3	2	**4.279**	6.561	**7.623**	8.330	9.127	**7.132**	**5.527**	**7.609**	9.172	10.05	10.82	**8.841**
3	3	4.935	6.530	7.791	8.529	9.426	7.442	6.168	7.896	9.338	10.28	11.15	9.142
3	4	5.528	7.180	8.592	8.724	10.01	8.006	6.923	8.802	10.40	10.60	12.05	9.924
10	3	4.586	**6.307**	7.628	**8.234**	**9.068**	7.164	5.903	7.822	**9.108**	**9.981**	**10.81**	8.895

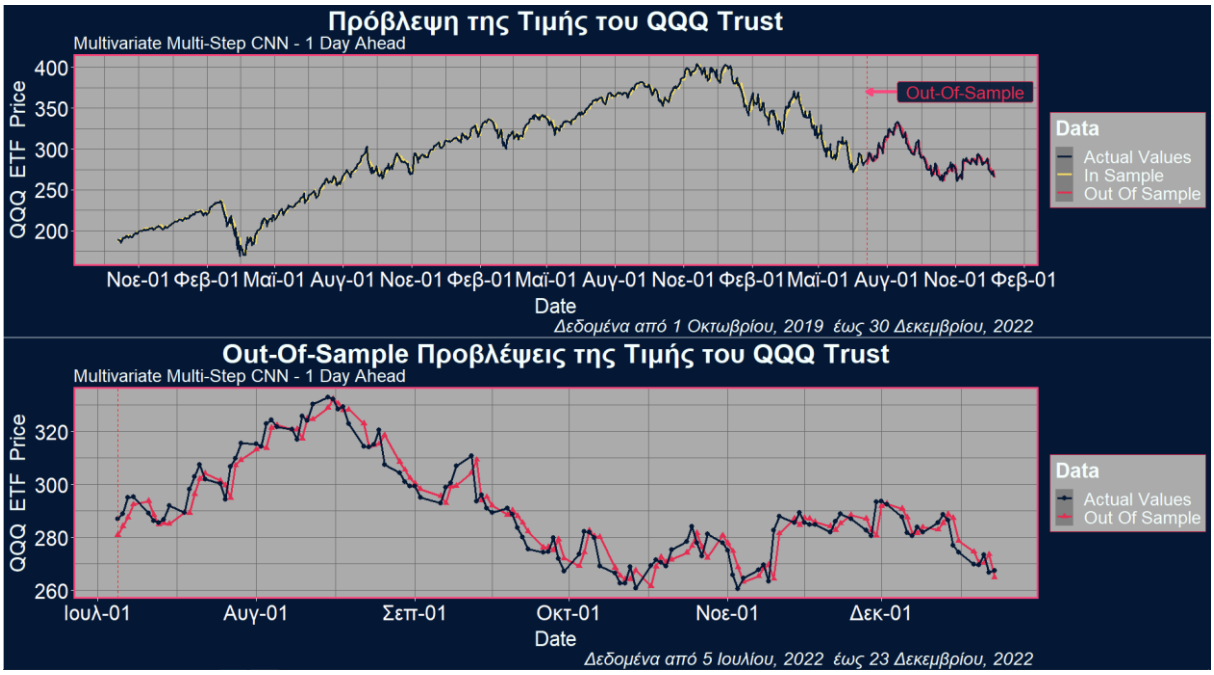
Σημείωση: Με έντονη γραφή δηλώνουμε το μοντέλο με την καλύτερη απόδοση για το συγκεκριμένο χρονικό βήμα.

Από τον πίνακα, διαπιστώνουμε ότι το μοντέλο του Σεναρίου 2 έχει την καλύτερη απόδοση. Αξίζει να σημειωθεί πως, για ακόμη μία φορά, παρά το γεγονός ότι το μοντέλο με χρονική υστέρηση 10 βημάτων δεν έχει την ίδια ακρίβεια με το αντίστοιχο μοντέλο με χρονική υστέρηση $3^{ων}$ βημάτων στις προβλέψεις για σύντομο χρονικό ορίζοντα, παρατηρούμε ότι οι μακροπρόθεσμες προβλέψεις του είναι ακριβέστερες.

Στη συνέχεια δίνονται τα διαγράμματα για την απόδοση του μοντέλου στο να προβλέπει την τιμή του QQQ μετά από μία και πέντε χρηματιστηριακές ημέρες

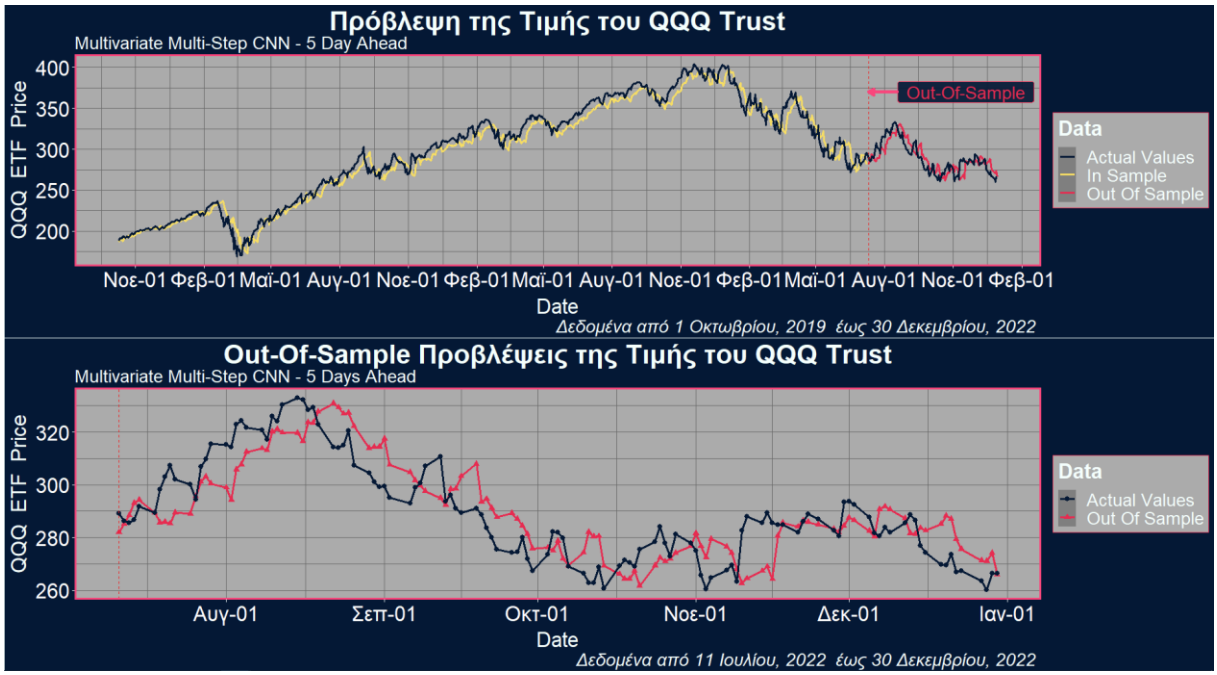
ΣΧΗΜΑ 3.28

Απόδοση Multivariate Mutli Step CNN για 1 Μέρα Μπροστά



ΣΧΗΜΑ 3.29

Απόδοση Multivariate Mutli Step CNN για 5 Μέρες Μπροστά



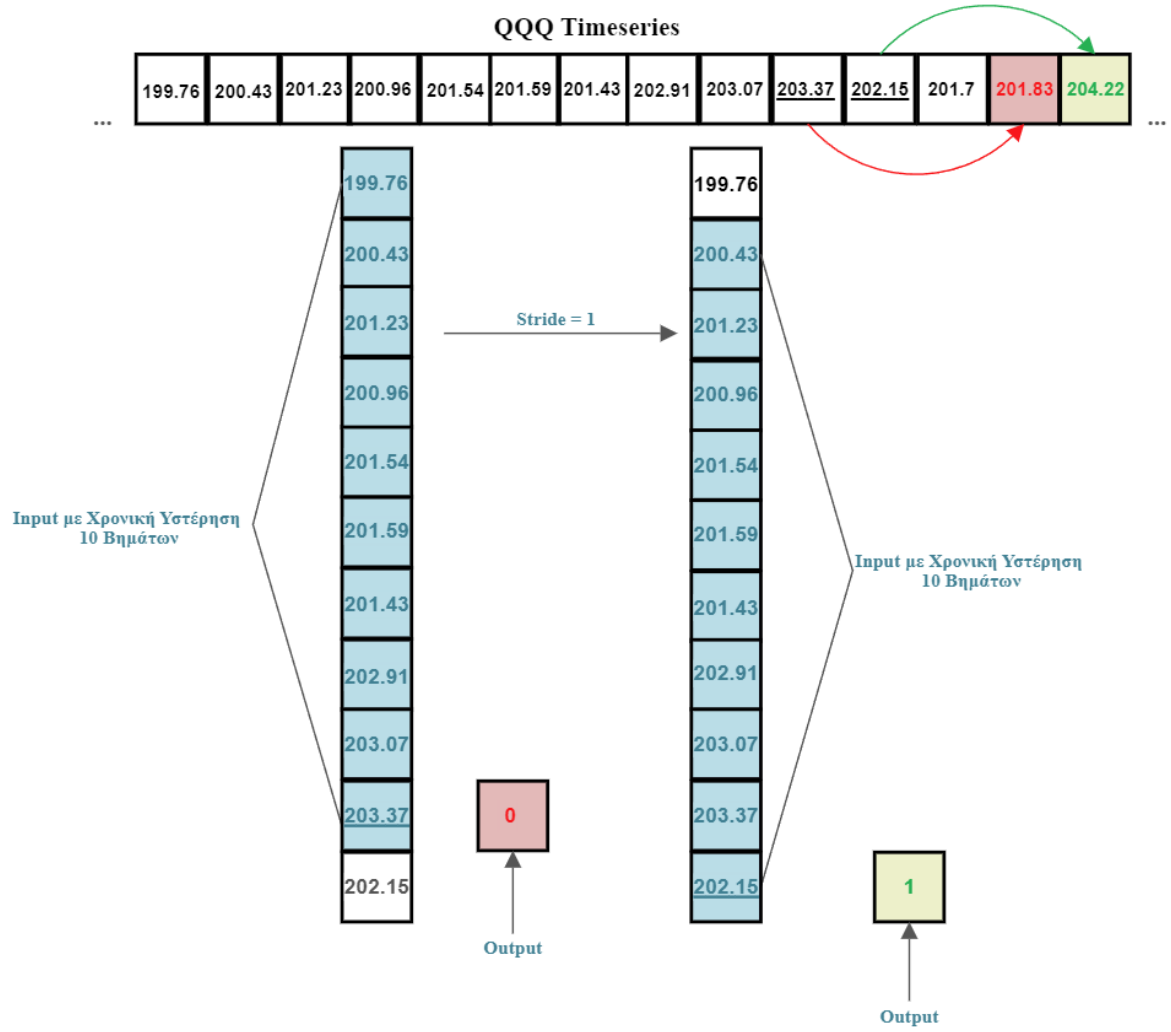
Παρατηρούμε ότι το μοντέλο καταφέρνει να αντιληφθεί τη συμπεριφορά της χρονοσειράς των πραγματικών τιμών, τόσο για τις προβλέψεις μίας ημέρας, όσο και για τις προβλέψεις πέντε ημερών μπροστά. Ωστόσο, παρατηρούμε ότι στη δεύτερη περίπτωση δεν καταφέρνει να εντοπίσει με την ίδια ακρίβεια τη διακύμανση των πραγματικών τιμών.

3.3.2.3 Classification CNN Μοντέλα

Σε αυτήν την ενότητα, θα εφαρμόσουμε μοντέλα κατηγοριοποίησης προκειμένου να προβλέψουμε την κίνηση της τιμής κλεισίματος του QQQ μετά από τρεις ημέρες. Για να το πετύχουμε αυτό, δημιουργήσαμε μια συνάρτηση που συγκρίνει, σε κάθε βήμα της χρονοσειράς, την τρέχουσα ημέρα με την ημέρα που βρίσκεται τρεις ημέρες αργότερα. Όταν η μελλοντική τιμή είναι μεγαλύτερη από την τρέχουσα τιμή, τότε η μεταβλητή παίρνει την τιμή 1 και όταν είναι μικρότερη, παίρνει την τιμή 0. Η χρονική υστέρηση σε όλα τα μοντέλα θα είναι $k = 10$. Χρησιμοποιώντας 14 τιμές από το dataset μας, παρουσιάζουμε παρακάτω δύο παραδείγματα ζευγαριών εισόδου-εξόδου, που παράγονται με τη διαδικασία που περιγράψαμε.

ΣΧΗΜΑ 3.30

Ζευγάρια εισόδου-εξόδου Μοντέλου Κατηγοριοποίησης

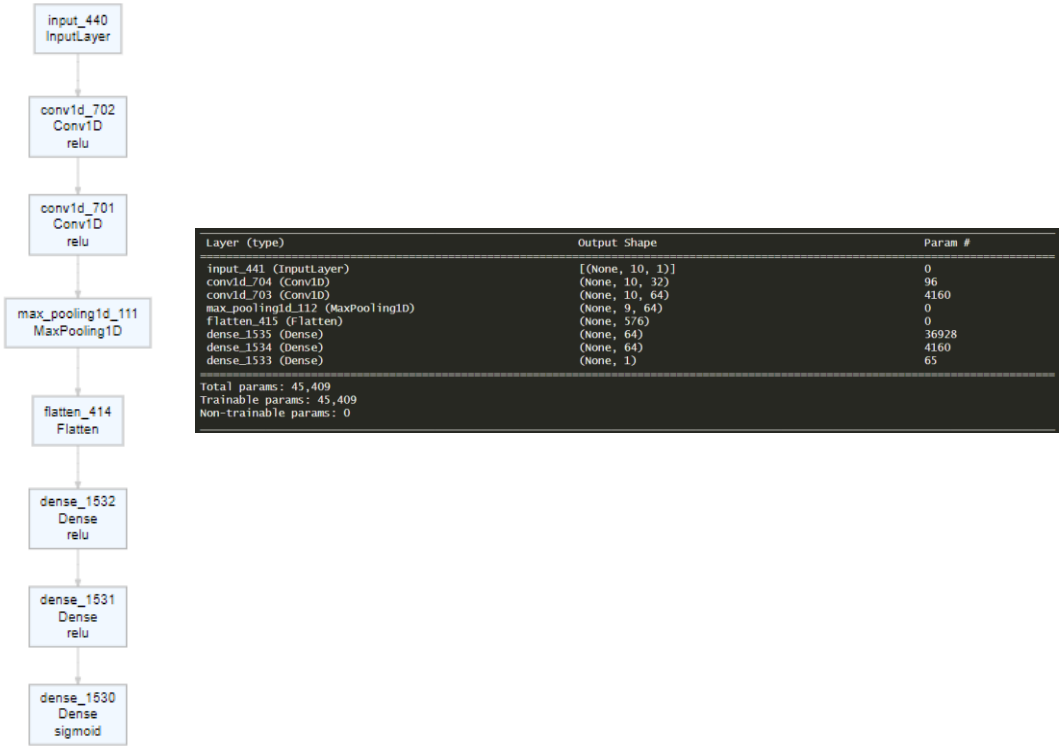


A) Univariate CNN Classification

Το πρώτο μοντέλο που θα εφαρμόσουμε αξιοποιεί μόνο την τιμή κλεισίματος QQQ και έχει την παρακάτω αρχιτεκτονική.

ΣΧΗΜΑ 3.31

Αρχιτεκτονική Univariate CNN Classification Μοντέλου



Το μοντέλο αποτελείται από δύο στιβάδες συνέλιξης. Η πρώτη στιβάδα έχει 32 φίλτρα και η δεύτερη 64. Το μήκος του πυρήνα συνέλιξης και για τις δύο στιβάδες είναι 2, και το stride είναι 1. Στη συνέχεια, ακολουθεί ένα max pooling στρώμα με μέγεθος παραθύρου ίσο με 2. Τέλος, αφού γίνει η διαδικασία της συνέλιξης και το output πάρει την μορφή διανύσματος (διαδικασία flattening), το σήμα επεξεργάζεται από δύο πλήρως συνδεδεμένες στιβάδες με 64 νευρώνες η καθεμία. Για να πάρουμε την πιθανότητα ανόδου της QQQ, το τελικό output θα υπολογιστεί από την συνάρτηση ενεργοποίησης Sigmoid. Αλγόριθμος βελτιστοποίησης θα είναι ο Adam με ρυθμό μάθησης $\eta = 0.0001$. Για συνάρτηση απώλειας χρησιμοποιήθηκε η Binary Cross-Entropy και το μοντέλο εκπαιδεύτηκε για 200 εποχές με μέγεθος παρτίδας 8. Επίσης, σημειώνουμε ότι οι περιπτώσεις καθόδου και ανόδου του QQQ μετά από τρεις μέρες είναι 349 και 468 αντίστοιχα, οπότε οι δύο κλάσεις είναι αρκετά ισορροπημένες μεταξύ τους. Για την αξιολόγηση της απόδοσης του μοντέλου, θα χρησιμοποιήσουμε δύο μετρικές: την Ακρίβεια (*Accuracy*) και το *F1-Score*. Η Ακρίβεια αξιολογεί το μοντέλο λαμβάνοντας υπόψη μόνο τον αριθμό των σωστών προβλέψεων, ενώ το F1-Score λαμβάνει υπόψη και τις

λανθασμένες προβλέψεις στον υπολογισμό του, προσφέροντας μια πιο συνολική εικόνα για την απόδοση του μοντέλου. Παρακάτω παραθέτουμε τον Πίνακας Σύγχυσης (*Confusion Matrix*) μαζί με τις τιμές των μετρικών.

Πίνακας 3.14 Πίνακας Σύγχυσης Univariate CNN Classification Μοντέλου

Confusion Matrix		
	Πραγματικές Τιμές	
Πρόβλεψη	0	1
0	55	36
1	12	21
Accuracy:	0.6129032	
F1-Score:	0.6962025	

Στον Πίνακα 3.14 βλέπουμε πως το μοντέλο έχει Ακρίβεια 61.29%, δηλαδή περίπου το 61.3% των προβλέψεών του είναι σωστές. Το F1-Score είναι περίπου 70% πράγμα που σημαίνει πως το μοντέλο πετυχαίνει μια αρκετά καλή γενικότερη απόδοση.

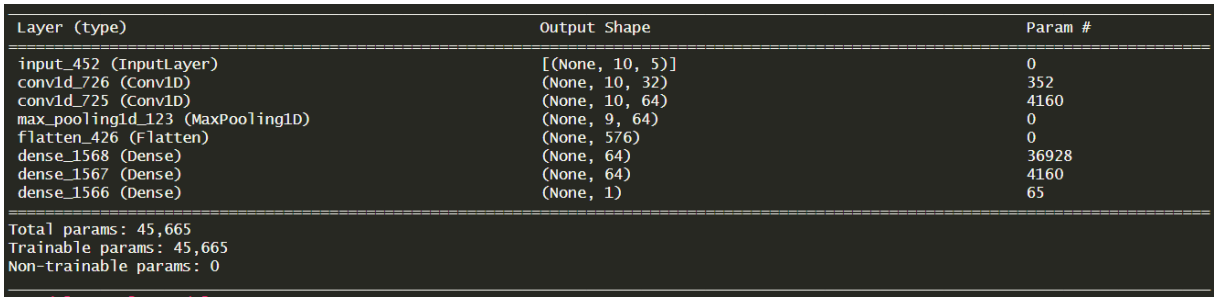
Το παραπάνω μοντέλο θα αποτελέσει το μοντέλο αναφοράς για τα νευρωνικά δίκτυα κατηγοριοποίησης που θα εφαρμόσουμε. Με τη χρήση διάφορων τεχνικών και βελτιώσεων, θα προσπαθήσουμε να πετύχουμε μια καλύτερη απόδοση με τις κατηγορίες μοντέλων που ακολουθούν.

B) Multivariate CNN Classification

Στη κατηγορία αυτή όπως και στα Multivariate CNN μοντέλα παλινδρόμησης ο τανυστής που θα δίνεται ως είσοδο στο δίκτυο θα αποτελείται και από άλλες μεταβλητές, πέρα από την τιμή κλεισίματος του QQQ. Για την κατασκευή του μοντέλου, θα χρησιμοποιήσουμε τις μεταβλητές που περιγράψαμε στο Σενάριο 2 (Close_QQQ, Close_MSFT, Open, High, Low), αφού αυτό το μοντέλο φάνηκε ότι επιτυγχάνει την καλύτερη απόδοση στις μακροπρόθεσμες προβλέψεις, όταν εφαρμόζουμε CNN μοντέλα. Η αρχιτεκτονική του δικτύου είναι ίδια με αυτή που φαίνεται στο Σχήμα 3.31, με τη διαφορά ότι τώρα στη στιβάδα εισόδου δίνονται δείγματα διάστασης $(timesteps, features) = (10,5)$ (`InputLayer`), όπως βλέπουμε και στο παρακάτω σχήμα.

ΣΧΗΜΑ 3.32

Σύνοψη Αρχιτεκτονικής Multivariate CNN Classification Μοντέλου



Layer (type)	Output Shape	Param #
input_452 (InputLayer)	[(None, 10, 5)]	0
conv1d_726 (Conv1D)	(None, 10, 32)	352
conv1d_725 (Conv1D)	(None, 10, 64)	4160
max_pooling1d_123 (MaxPooling1D)	(None, 9, 64)	0
flatten_426 (Flatten)	(None, 576)	0
dense_1568 (Dense)	(None, 64)	36928
dense_1567 (Dense)	(None, 64)	4160
dense_1566 (Dense)	(None, 1)	65

Total params: 45,665
Trainable params: 45,665
Non-trainable params: 0

Στη συνέχεια δίνεται ο πίνακας σύγχυσης, για τα δεδομένα ελέγχου, τον οποίο θα χρησιμοποιήσουμε για να αξιολογήσουμε την απόδοση του μοντέλου.

Πίνακας 3.15 Πίνακας Σύγχυσης Multivariate CNN Classification Μοντέλου

Confusion Matrix		
	Πραγματικές Τιμές	
Πρόβλεψη	0	1
0	62	46
1	5	11
Accuracy:	0.5887097	
F1-Score:	0.7085714	

Παρατηρούμε ότι, αν και το F1-Score είναι ελαφρώς υψηλότερο, η Ακρίβεια είναι αρκετά μικρότερη από αυτή που πετύχαμε νωρίτερα (βλ. Πίνακας 3.14), οπότε η προσθήκη των επιπλέον μεταβλητών στο μοντέλο φαίνεται ότι δεν κατάφερε να βελτιώσει την απόδοση του.

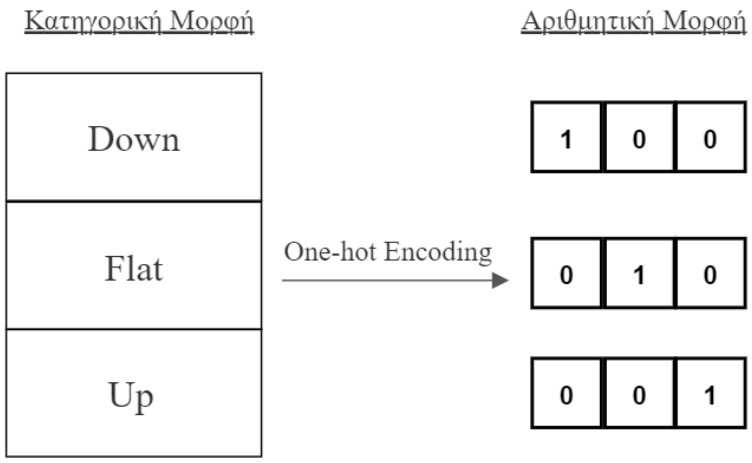
Γ) Univariate CNN Multi Class Classification

Σε αυτήν την περίπτωση, θα κατηγοριοποιήσουμε διαφορετικά τις περιπτώσεις, όπου η τιμή του QQQ αυξάνεται ή μειώνεται κατά ένα μικρό ποσοστό. Έτσι τα δεδομένα θα κατηγοριοποιηθούν σε τρεις κατηγορίες: "*Up*", "*Flat*" και "*Down*". Πιο συγκεκριμένα, όταν η τιμή κλεισίματος αυξηθεί κατά ποσοστό μεγαλύτερο του 1% μετά από τρεις χρηματιστηριακές

ημέρες, τότε η τρέχουσα ημέρα κατηγοριοποιείται ως "*Up*". Αντίστοιχα, αν η τιμή μειωθεί κατά 1% ή περισσότερο, κατηγοριοποιείται ως "*Down*". Σε περίπτωση που δεν συμβεί τίποτα από τα δύο, δηλαδή δεν παρατηρήθηκε σημαντική αλλαγή μετά από τρεις ημέρες, η ημέρα κατηγοριοποιείται ως "*Flat*". Η αναπαράσταση των κατηγορικών δεδομένων σε αριθμητική μορφή θα γίνει με τη μέθοδο *one-hot-encoding*. Με αυτήν τη μέθοδο, όπως αναφέρθηκε και στην Υπο-Ενοτητα 1.3.2, κάθε διαφορετική κατηγορία αντιστοιχίζεται σε μία ξεχωριστή στήλη ενός πίνακα. Η αντίστοιχη στήλη για κάθε κατηγορία λαμβάνει την τιμή 1, όταν η γραμμή ανήκει σε αυτήν τη κατηγορία, ενώ όλες οι υπόλοιπες στήλες παίρνουν την τιμή 0. Αυτό γίνεται για κάθε γραμμή του πίνακα, ανάλογα με την κατηγορία που ανήκει η γραμμή. Η διαδικασία αυτή απεικονίζεται στο παρακάτω σχήμα.

ΣΧΗΜΑ 3.33

Διαδικασία One-Hot Encoding

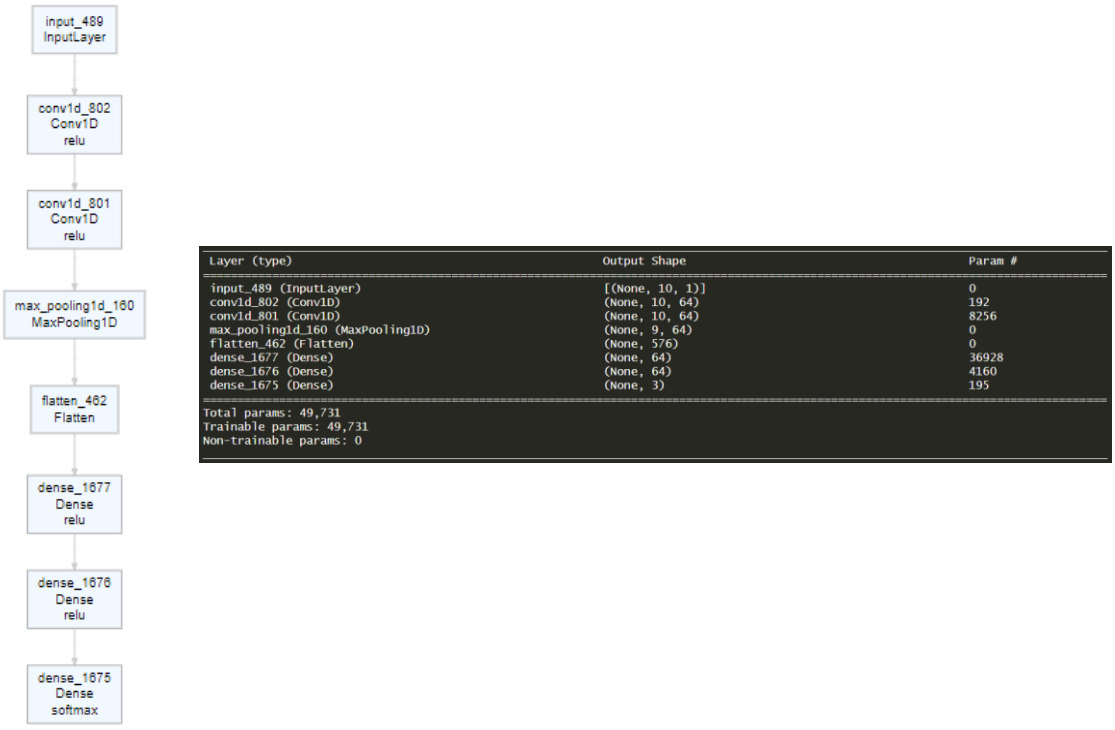


Μετά αυτήν τη μετατροπή των δεδομένων, έχουμε 316 περιπτώσεις ανόδου του QQQ μετά από τρεις μέρες, 231 περιπτώσεις καθόδου, και 270 περιπτώσεις όπου θεωρούμε ότι η τιμή παρέμεινε σταθερή.

Η αρχιτεκτονική του δικτύου, όπως φαίνεται στο Σχήμα 3.34, είναι παρόμοια με αυτήν του Σχήματος 3.31, αλλά με μερικές διαφορές. Και οι δύο στιβάδες συνέλιξης τώρα αποτελούνται από 64 φίλτρα, και επιπλέον, η στιβάδα εξόδου χρησιμοποιεί τη συνάρτηση ενεργοποίησης Softmax. Τέλος, ο αλγόριθμος βελτιστοποίησης είναι η (κατηγορική) διασταυρούμενη εντροπία (Categorical Cross-Entropy).

ΣΧΗΜΑ 3.34

Αρχιτεκτονική Univariate Multi Class Classification Μοντέλου



Στη συνέχεια δίνεται ο πίνακας σύγχυσης, για τα δεδομένα ελέγχου.

Πίνακας 3.16 Πίνακας Σύγχυσης Univariate CNN Multi Class Classification Μοντέλου

Confusion Matrix			
	Πραγματικές Τιμές		
Πρόβλεψη	Down	Flat	Up
Down	18	6	13
Flat	6	2	2
Up	28	16	33
Accuracy:	0.4274		

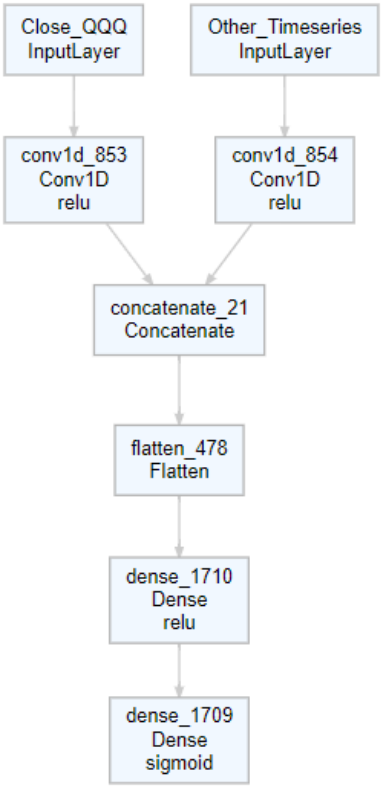
Όπως είναι φανερό από τον πίνακα, το μοντέλο δεν καταφέρνει να αναγνωρίσει με μεγάλη ακρίβεια την κάθε κατηγορία .

Δ) Multivariate CNN Classification με Υπο-Μοντέλα

Στη κατηγορία αυτή θα χρησιμοποιήσουμε τις μεταβλητές του Σεναρίου 2 και θα κατασκευάσουμε το παρακάτω νευρωνικό δίκτυο με υπο-μοντέλα.

ΣΧΗΜΑ 3.35

Αρχιτεκτονική Multivariate CNN Classification με Υπο-Μοντέλα



Layer (type)	Output Shape	Param #	Connected to
Close_QQQ (InputLayer)	[(None, 10, 1)]	0	[]
Other_Timeseries (InputLayer)	[(None, 10, 4)]	0	[]
conv1d_853 (Conv1D)	(None, 10, 32)	96	['Close_QQQ[0][0]']
conv1d_854 (Conv1D)	(None, 10, 32)	288	['Other_Timeseries[0][0]']
concatenate_21 (Concatenate)	(None, 20, 32)	0	['conv1d_853[0][0]', 'conv1d_854[0][0]']
flatten_478 (Flatten)	(None, 640)	0	['concatenate_21[0][0]']
dense_1710 (Dense)	(None, 128)	82048	['flatten_478[0][0]']
dense_1709 (Dense)	(None, 1)	129	['dense_1710[0][0]']

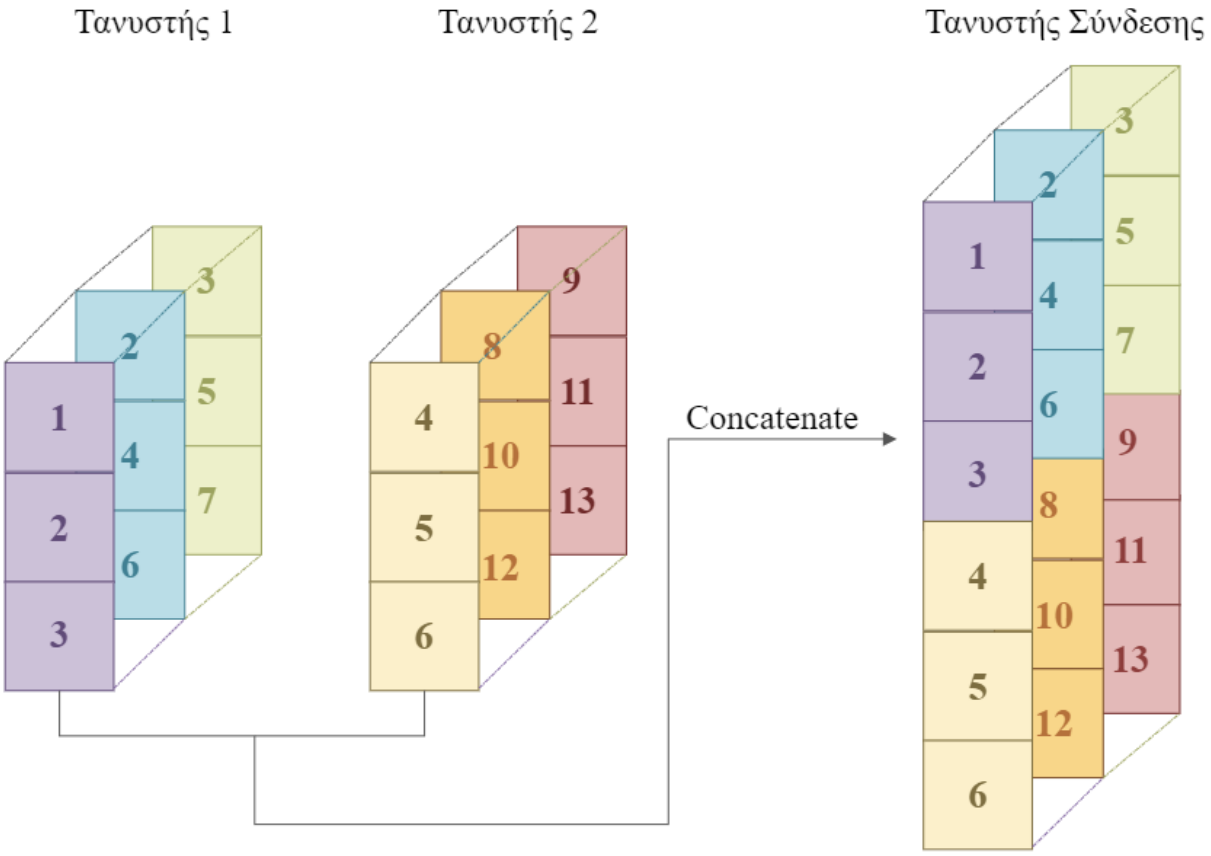
Total params: 82,561
Trainable params: 82,561
Non-trainable params: 0

Το δίκτυο αυτό, σε αντίθεση με τα μοντέλα που είδαμε μέχρι τώρα, αποτελείται από δύο στιβάδες εισόδου. Στην αριστερή κολώνα (*Close_QQQ*) είσοδος είναι η τιμή κλεισίματος του QQQ και επεξεργάζεται μόνο αυτή από την στιβάδα συνέλιξης που ακολουθεί. Στη δεξιά κο-

λώνα (*Other Timeseries*), χρησιμοποιούνται οι μεταβλητές *Open*, *Low*, *High*, και *Close_MSFT*, που επίσης επεξεργάζονται από το δίκτυο σε μια ξεχωριστή στιβάδα συνέλιξης με 32 φίλτρα. Τα σήματα από τις δύο κολόνες συνδυάζονται στη στιβάδα σύνδεσης (*Concatenate layer*) και προωθούνται σε μια πλήρως συνδεδεμένη στιβάδα με 128 νευρώνες. Στο παρακάτω σχήμα απεικονίζεται ο τρόπος λειτουργίας της στιβάδας σύνδεσης πάνω σε δύο τανυστές.

ΣΧΗΜΑ 3.36

Εφαρμογή Στιβάδας Σύνδεσης



Το τελικό αποτέλεσμα, παράγεται στη στιβάδα εξόδου με χρήση της συνάρτησης ενεργοποίησης Sigmoid. Επίσης, το μοντέλο εκπαιδεύτηκε για 100 μόνο εποχές, δηλαδή για τον μισό αριθμό εποχών σε σχέση με τα προηγούμενα μοντέλα που εξετάστηκαν σε αυτήν την ενότητα, αλλά ο αριθμός των παραμέτρων (κυρίως λόγω της στιβάδας σύνδεσης πριν τη πλήρως

συνδεδεμένη στιβάδα) είναι σχεδόν διπλάσιος (82,561 έναντι 45,665 παραμέτρων που είχε το Multivariate CNN μοντέλο).

Στη συνέχεια δίνεται ο πίνακας σύγχυσης, για τα δεδομένα ελέγχου.

Πίνακας 3.17 Πίνακας Σύγχυσης Multivariate CNN Classification με Υπο-Μοντέλα

Confusion Matrix		
	Πραγματικές Τιμές	
Πρόβλεψη	0	1
0	57	36
1	10	21
Accuracy:	0.6290323	
F1-Score:	0.7125	

Από τον πίνακα παρατηρούμε ότι το δίκτυο με τα υπο-μοντέλα όχι μόνο βελτίωσε την απόδοσή του σε σχέση με το αντίστοιχο μοντέλο που δεχόταν όλες τις μεταβλητές του Σεναρίου 2 σε μία είσοδο (βλ. Πίνακας 3.15), αλλά κατάφερε επίσης να ξεπεράσει και την απόδοση του μοντέλου αναφοράς (βλ. Πίνακας 3.14).

3.3.3 Συγκρίσεις Καλύτερων Μοντέλων

Στην ενότητα αυτή θα συγκρίνουμε μεταξύ τους όλες τις μεθόδους της one-step και multi-step κατηγορίας, επιλέγοντας από κάθε μέθοδο το μοντέλο που εμφάνισε την καλύτερη απόδοση στην πρόβλεψη των τιμών του test set. Αξίζει να σημειωθεί πως, αν και επαναλάβαμε κάθε μοντέλο πέντε φορές (για να αποκτήσουμε μια πιο αντιπροσωπευτική εικόνα σχετικά με την απόδοση του), τα αποτελέσματα μπορεί ακόμη να επηρεάζονται από την τυχαιότητα και αν θα θέλαμε να είμασταν περισσότερο σίγουροι θα έπρεπε να επαναλάβουμε κάθε μοντέλο πολλές περισσότερες φορές. Στους παρακάτω πίνακες 3.18 και 3.19 δίνονται συγκεντρωτικά τα αποτελέσματα για τα καλύτερα one-step και multi-step μοντέλα παλινδρόμησης. Θυμίζουμε ότι το Σενάριο 1 των multivariate μοντέλων περιλάμβανε τις μεταβλητές Close_QQQ, Open, High και Low, το Σενάριο 2 αξιοποιούσε όλες τις μεταβλητές του Σεναρίου 1 και επιπλέον την μεταβλητή Close_MSFT για τις τιμές κλεισίματος της Microsoft, ενώ το Σενάριο

3 αποτελούνταν από τις μεταβλητές Close_QQQ, Close_MSFT, Close_AAPL, Close_TSLA, Open, High και Low.

Πίνακας 3.18 Πίνακας Σύγκρισης One step μοντέλων

One-Step Forecasts			
Κατηγορία Μοντέλου	Κατηγορία Μεθόδου	Προβλεπτική Ακρίβεια (Out-of Sample)	
Model	*Method*	*MAE*	*RMSE*
MLP	Univariate (Adam 3-Lags)	4.39442	5.45157
MLP	Multivariate (Scenario 3 RMSprop Lag-3)	4.45993	5.60943
CNN	**Univariate (RMSprop 5-Lags)**	**4.21884**	**5.39059**
CNN	Multivariate (Scenario 1 RMSprop Lag-3)	4.33636	5.57341
ARIMA	Univariate (One step test set forecasts)	4.28004	5.394

Σημείωση: Με έντονη γραφή δηλώνουμε το μοντέλο με την καλύτερη απόδοση.

Πίνακας 3.19 Πίνακας Σύγκρισης Multi step μοντέλων

Multi-Step Forecasts			
Κατηγορία Μοντέλου	Κατηγορία Μεθόδου	Προβλεπτική Ακρίβεια (Out-of Sample)	
Model	*Method*	*MAE*	*RMSE*
MLP	Univariate (RMSprop 10-Lags)	7.281	8.994
MLP	Multivariate (Scenario 3 RMSprop Lag-3)	7.319	9.049
CNN	**Univariate (Adam 5-Lags)**	**7.127**	**8.84**
CNN	Multivariate (Scenario 2 RMSprop Lag-3)	7.132	8.841

Σημείωση: Με έντονη γραφή δηλώνουμε το μοντέλο με την καλύτερη απόδοση.

Από τους παραπάνω πίνακες φαίνεται ότι τα μοντέλα CNN, τόσο τα univariate όσο και τα multivariate, ξεπερνούν σε απόδοση τα MLP μοντέλα τόσο στην κατηγορία των one-step προβλέψεων όσο και των multi-step προβλέψεων. Τα univariate CNN πετυχαίνουν την καλύτερη απόδοση και στις δύο κατηγορίες, με την multivariate εκδοχή τους να επιτυγχάνει επίσης αρκετά καλή απόδοση. Έτσι για να διαπιστωθεί ποιο από τα δύο μοντέλα είναι το επικρατέστερο, απαιτούνται περισσότερες επαναλήψεις για πιο αξιόπιστη αξιολόγηση. Σημειώνεται επίσης, ότι το μοντέλο ARIMA, όταν εφαρμόζεται με τη μέθοδο *one-step test set forecasts* και με βάση το RMSE, επιδεικνύει απόδοση πολύ κοντά σε αυτήν του univariate

CNN. Σχετικά με τους αλγορίθμους βελτιστοποίησης Adam και RMSprop, δεν φαίνεται να υπάρχει σαφής διαφορά στην απόδοση μεταξύ τους και κανένας από τους δύο δεν επιδεικνύει συνεπώς καλύτερα αποτελέσματα. Καλυτέρα αποτελέσματα φαίνεται ότι πετυχαίνουν τα μοντέλα με χρονική υστέρηση $3^{ων}$ βημάτων και, ειδικά στα Univariate CNN, αποτελεσματικά ήταν επίσης και τα μοντέλα με χρονική υστέρηση 5 βημάτων.

ΠΑΡΑΡΤΗΜΑΤΑ

Παρακάτω δίνεται ενδεικτικά μέρος του κώδικα που χρησιμοποιήθηκε στην R.

Multivariate MLP Multi-Step

```
Data_Complete <-
read.csv("C:/Users/Kon/Documents/RStudio_Import_Files/QQQ_With_Macro_and_To
p10.csv", header = TRUE, stringsAsFactors = FALSE)
Data_Complete <- Data_Complete %>% select(-c(Date,Adj.Close))  %>%
as.data.frame()

#1st Scenario
input <- Data_Complete %>%
  select(c( Close_QQQ, Open, High, Low)) %>%
  mutate(across(.cols = everything(), .fns = list(lag1 = ~ lag(., n = 1),
                                                  lag2 = ~ lag(., n = 2),
                                                  lag3 = ~ lag(., n = 3)
  ))) %>%
  select(matches("lag"))

# 2nd Scenario
input <- Data_Complete %>%
  select(c( Close_QQQ, Open, High, Low,Close_MSFT)) %>%
  mutate(across(.cols = everything(), .fns = list(lag1 = ~ lag(., n = 1),
                                                  lag2 = ~ lag(., n = 2),
                                                  lag3 = ~ lag(., n = 3)
  ))) %>%
  select(matches("lag"))

# 3rd Scenario
input <- Data_Complete %>%
  select(c( Close_QQQ, Open, High, Low,Close_MSFT,Close_AAPL,Close_TSLA))
%>%
  mutate(across(.cols = everything(), .fns = list(lag1 = ~ lag(., n = 1),
lag2 = ~ lag(., n = 2),
                                                  lag3 = ~ lag(., n = 3))))
%>%
  select(matches("lag"))

#4th scenario
input <- Data_Complete %>%
  select(c( Close_QQQ, Open, High,
Low,Close_AAPL,Close_MSFT,Close_AMZN,Close_NVDA,Close_TSLA,
            Close_GOOGL,Close_GOOG,Close_META,Close_AVGO)) %>%
  mutate(across(.cols = everything(), .fns = list(lag1 = ~ lag(., n = 1),
                                                  lag2 = ~ lag(., n = 2),
                                                  lag3 = ~ lag(., n = 3)
  ))) %>%
  select(matches("lag"))

#5th scenario
```

```
input <- Data_Complete %>%
  select(c( Close_QQQ, Open, High, Low,Close_MSFT,Close_AAPL,Close_TSLA))
%>%
  mutate(across(.cols = everything(), .fns = list(lag1 = ~ lag(., n = 1),
lag2 = ~ lag(., n = 2),
                                                  lag3 = ~ lag(., n = 3),
lag4 = ~ lag(., n = 4),
                                                  lag5 = ~ lag(., n = 5),
lag6 = ~ lag(., n = 6),
                                                  lag7 = ~ lag(., n = 7),
lag8 = ~ lag(., n = 8),
                                                  lag9 = ~ lag(., n = 9),
lag10 = ~ lag(., n = 10)))) %>%
  select(matches("lag"))


output <- Data_Complete["Close_QQQ"] %>%
  mutate(across(.cols = everything(), .fns = list(lead1 = ~ lead(., n = 1),
                                                  lead2 = ~ lead(., n = 2),
                                                  lead3 = ~ lead(., n = 3),
                                                  lead4 = ~ lead(., n =
4))))



input_output_mx<- cbind(input,output) %>% na.omit

# Train Val Test Split
all_x <- input_output_mx %>% as.data.frame %>% select(matches("lag")) %>%
as.matrix
all_y <- input_output_mx %>% as.data.frame %>% se-
lect(Close_QQQ:Close_QQQ_lead4) %>% as.matrix

colnames(all_x)
colnames(all_y)

num_train <- round(nrow(input_output_mx) * 0.7)
num_val <- round(nrow(input_output_mx) * 0.15)
num_test <- nrow(input_output_mx) -  num_train - num_val



train_x <- all_x[seq(num_train), ] %>% as.matrix
train_y <- all_y[seq(num_train), ] %>% as.matrix

val_x <- all_x[seq(from = nrow(train_x) + 1,length.out = num_val), ] %>%
as.matrix
val_y <- all_y[seq(from = nrow(train_y) + 1,length.out = num_val), ] %>%
as.matrix

test_x <- all_x[seq(to = nrow(all_x),length.out = num_test), ] %>%
as.matrix
test_y <- all_y[seq(to = nrow(all_x),length.out = num_test), ] %>%
as.matrix

tail(val_y)
head(test_y)
```

```
nrow(train_x);nrow(val_x);nrow(test_x);nrow(train_y);nrow(val_y);nrow(test_
y)

#'[######### Model Training and Evaluation #########]

model <- keras_model_sequential()
model %>%
  layer_batch_normalization(input_shape = dim(train_x)[2]) %>%
  layer_dense(units = 32, activation = 'relu') %>%
  layer_dense(units = 16) %>%
  layer_activation_leaky_relu(alpha = 0.001) %>%
  layer_batch_normalization() %>%
  layer_dense(units = dim(train_y)[2])



# Compile
model %>% compile(loss = 'mse',
                  # optimizer = optimizer_adam(learning_rate=0.001),
                  optimizer = optimizer_rmsprop(learning_rate=0.001),
                  metrics = 'mae')
summary(model)

model %>% plot_model()


checkpoint <- list(
  callback_model_checkpoint(
    filepath =
"C:/Users/Kon/Documents/ANN_Models/multivariate_mlp_5steps_4.keras",
    save_best_only = TRUE,
    monitor = "val_loss",
    verbose = 1)
)



# Fit Model with CheckPoint
history <- model %>%
  fit(train_x,
      train_y,
      epochs = 100,
      batch_size = 8,
      validation_data = list(val_x,val_y),
      callbacks = checkpoint)

plot(history)

best_model<-
load_model_tf("C:/Users/Kon/Documents/ANN_Models/multivariate_mlp_5steps_4.
keras")


# Evaluate
model %>% evaluate(test_x, test_y)
best_model %>% evaluate(test_x, test_y)
pred_multi_mpl_5steps <- best_model %>% predict(test_x)
```

```
mae(pred_multi_mpl_5steps[,1],test_y[,1])
mae(pred_multi_mpl_5steps[,2],test_y[,2])
mae(pred_multi_mpl_5steps[,3],test_y[,3])
mae(pred_multi_mpl_5steps[,4],test_y[,4])
mae(pred_multi_mpl_5steps[,5],test_y[,5])
mae(pred_multi_mpl_5steps,test_y)


rmse(pred_multi_mpl_5steps[,1],test_y[,1])
rmse(pred_multi_mpl_5steps[,2],test_y[,2])
rmse(pred_multi_mpl_5steps[,3],test_y[,3])
rmse(pred_multi_mpl_5steps[,4],test_y[,4])
rmse(pred_multi_mpl_5steps[,5],test_y[,5])
rmse(pred_multi_mpl_5steps,test_y)
```

Multivariate CNN Multi-Step

```
sampling_rate <- 1
time_steps <- 10
pred_days <- 5
batch_size <- 1000

# Scenario 1
input_data_colnames <- Data_Complete %>% select(Close_QQQ,Open,High,Low)
%>% colnames()

# Scenario 2
input_data_colnames <- Data_Complete %>% select(c( Close_QQQ, Open, High,
Low,Close_MSFT)) %>% colnames()

#Scenario 3
input_data_colnames <- Data_Complete %>%   select(c( Close_QQQ, Open, High,
Low,Close_MSFT,Close_AAPL,Close_TSLA)) %>% colnames()

# Scenario 4
input_data_colnames <- Data_Complete %>%  select(c( Close_QQQ, Open, High,
Low,Close_AAPL,Close_MSFT,Close_AMZN,Close_NVDA,Close_TSLA,Close_GOOGL,Clos
e_GOOG,Close_META,Close_AVGO)) %>% colnames()

df_to_inputs_and_targets <- function(df) {
  inputs <- df[input_data_colnames] %>%
    as.matrix()
  targets <- df$Close_QQQ %>% as.matrix %>% embed(pred_days) %>%
as.data.frame %>% rev %>% as.matrix
   list (
    head(inputs, - pred_days),
    tail(targets, -time_steps)
  )
}
make_dataset <- function(df)
{ c(inputs, targets) %<-%
    df_to_inputs_and_targets(df)
  timeseries_dataset_from_array(
    inputs, targets,
```

```
    sampling_rate = sampling_rate,
    sequence_length = time_steps,
    shuffle = FALSE,
    batch_size = batch_size
  )
}


# Scenario 1
data_cnn<-Data_Complete %>% select(Close_QQQ,Open,High,Low) %>%
make_dataset

# Scenario 2
data_cnn<-Data_Complete %>% select(c( Close_QQQ, Open, High,
Low,Close_MSFT)) %>% make_dataset

#Scenario 3
data_cnn<-Data_Complete %>% select(c( Close_QQQ, Open, High,
Low,Close_MSFT,Close_AAPL,Close_TSLA)) %>% make_dataset

# Scenario 4
data_cnn<-Data_Complete %>% select(c( Close_QQQ, Open, High,
Low,Close_AAPL,Close_MSFT,Close_AMZN,Close_NVDA,Close_TSLA,Close_GOOGL,Clos
e_GOOG,Close_META,Close_AVGO)) %>% make_dataset


data_cnn<-c(inputs, targets) %<-%
  iter_next(as_iterator(data_cnn))

##########3 steps#####################

train_x <- data_cnn[[1]][1:569,,]
train_y <-data_cnn[[2]][1:569,]

#val_set
val_x <-data_cnn[[1]][570:691,,]
val_y <-data_cnn[[2]][570:691,]

#test_set
test_x <-data_cnn[[1]][692:813,,]
test_y <-data_cnn[[2]][692:813,]


################ 5 steps #############

train_x <- data_cnn[[1]][1:568,,]
train_y <-data_cnn[[2]][1:568,]

#val_set
val_x <-data_cnn[[1]][569:689,,]
val_y <-data_cnn[[2]][569:689,]

#test_set
test_x <-data_cnn[[1]][690:811,,]
test_y <-data_cnn[[2]][690:811,]

################ 10 steps #############
train_x <- data_cnn[[1]][1:564,,]
train_y <-data_cnn[[2]][1:564]
```

```
#val_set
val_x <-data_cnn[[1]][565:685,,]
val_y <-data_cnn[[2]][565:685,]

#test_set
test_x <-data_cnn[[1]][686:806,,]
test_y <-data_cnn[[2]][686:806,]

##############################

length(input_data_colnames)
inputs <- layer_input(shape = c(time_steps,length(input_data_colnames)))
#n_steps, n_features

outputs <- inputs %>%

  layer_conv_1d(filters = 32, kernel_size = 3, stride=1, padding =
'causal', activation = "relu") %>%
  layer_conv_1d(filters = 32, kernel_size = 2, stride=1, padding =
'causal', activation = "relu") %>%
  layer_flatten() %>%
  layer_dense(units = 16, activation = 'relu') %>%
  layer_batch_normalization() %>%
  layer_dense(5)

# create and compile model
model <- keras_model(inputs = inputs,
                     outputs = outputs) %>% compile(
                       loss = "mse",
                       optimizer = optimizer_adam(learning_rate = 0.001),
                       # optimizer = optimizer_rmsprop(learning_rate =
0.001),

                       # optimizer = optimizer_rmsprop(learning_rate =
0.0001, decay = 1e-6),
                       metrics = "mae"
                     )
summary(model)
model %>% plot_model

checkpoint <- list(
  callback_model_checkpoint(
    # filepath =
"C:/Users/Kon/Documents/ANN_MOdels/best_{epoch:04d}_{val_loss:.3f}.keras",
    filepath =
"C:/Users/Kon/Documents/ANN_Models/multivariate_cnn_5steps.keras",
    save_best_only = TRUE,
    monitor = "val_loss",
    verbose = 1)
)

history <- model %>%
  fit(train_x,
      train_y,
      epochs = 100,
      batch_size = 8,
      validation_data = list(val_x,val_y),
      callbacks = checkpoint
  )
```

```
model %>% evaluate(test_x,test_y)

plot(history)
best_model<-
load_model_tf("C:/Users/Kon/Documents/ANN_Models/multivariate_cnn_5steps.ke
ras")

best_model %>% evaluate(test_x,test_y)
pred_multi_cnn_5steps <- best_model %>% predict(test_x)

mae(pred_multi_cnn_5steps[,1],as.array(test_y[,1]))
mae(pred_multi_cnn_5steps[,2],as.array(test_y[,2]))
mae(pred_multi_cnn_5steps[,3],as.array(test_y[,3]))
mae(pred_multi_cnn_5steps[,4],as.array(test_y[,4]))
mae(pred_multi_cnn_5steps[,5],as.array(test_y[,5]))
mae(pred_multi_cnn_5steps,as.array(test_y))

rmse(pred_multi_cnn_5steps[,1],as.array(test_y[,1]))
rmse(pred_multi_cnn_5steps[,2],as.array(test_y[,2]))
rmse(pred_multi_cnn_5steps[,3],as.array(test_y[,3]))
rmse(pred_multi_cnn_5steps[,4],as.array(test_y[,4]))
rmse(pred_multi_cnn_5steps[,5],as.array(test_y[,5]))
rmse(pred_multi_cnn_5steps,as.array(test_y))
```

Univariate CNN Multi Class Classification

```
Data_Movement <- Data_Complete

# Down Flat Up
Data_Movement$Movement <- NA
for(i in 1:(nrow(Data_Movement)-3)){
  if(Data_Movement$Close_QQQ[i+3] > (Data_Movement$Close_QQQ[i] + Da-
ta_Movement$Close_QQQ[i]*0.01)){
    Data_Movement$Movement[i] <- 2

  }
  else if (Data_Movement$Close_QQQ[i+3] < (Data_Movement$Close_QQQ[i] - Da-
ta_Movement$Close_QQQ[i]*0.01)){
    Data_Movement$Movement[i] <- 0

  }
  else {
    Data_Movement$Movement[i] <- 1
  }
}

Data_Movement <- na.omit(Data_Movement)


sampling_rate <- 1
time_steps <- 10
batch_size <- 1000
pred_days <- 1
```

```r
# Scenario 0
input_data_colnames <- Data_Movement %>% select(Close_QQQ) %>% colnames()


# Scenario 1
input_data_colnames <- Data_Movement %>% select(Close_QQQ,Open,High,Low)
%>% colnames()

# Scenario 2
input_data_colnames <- Data_Scaled %>% select(c( Close_QQQ, Open, High,
Low,Close_MSFT)) %>% colnames()

#Scenario 3
input_data_colnames <- Data_Scaled %>%   select(c( Close_QQQ, Open, High,
Low,Close_MSFT,Close_AAPL,Close_TSLA)) %>% colnames()

# Scenario 4
input_data_colnames <- Data_Scaled %>%  select(c( Close_QQQ, Open, High,
Low,Close_AAPL,Close_MSFT,Close_AMZN,Close_NVDA,Close_TSLA,Close_GOOGL,Clos
e_GOOG,Close_META,Close_AVGO)) %>% colnames()


df_to_inputs_and_targets <- function(df) {
  inputs <- df[input_data_colnames] %>%
    # normalize_input_data() %>%
    as.matrix()
  targets <- Data_Movement$Movement %>% as.matrix %>% embed(pred_days) %>%
as.data.frame %>% rev %>% as.matrix
  list (
    inputs,
    tail(targets, (-time_steps+1))
  )
}


make_dataset <- function(df)
{ c(inputs, targets) %<-%
    df_to_inputs_and_targets(df)
  timeseries_dataset_from_array(
    inputs, targets,
    sampling_rate = sampling_rate,
    sequence_length = time_steps,
    shuffle = FALSE,
    batch_size = batch_size
  )
}

# # Scenario 0
data_cnn<-Data_Movement %>% select(Close_QQQ) %>% make_dataset

# Scenario 1
data_cnn<-Data_Movement %>% select(Close_QQQ,Open,High,Low) %>%
make_dataset

# Scenario 2
data_cnn<-Data_Movement %>% select(c( Close_QQQ, Open, High,
Low,Close_MSFT)) %>% make_dataset

#Scenario 3
```

```
data_cnn<-Data_Movement %>% select(c( Close_QQQ, Open, High,
Low,Close_MSFT,Close_AAPL,Close_TSLA)) %>% make_dataset

# Scenario 4
data_cnn<-Data_Movement %>% select(c( Close_QQQ, Open, High,
Low,Close_AAPL,Close_MSFT,Close_AMZN,Close_NVDA,Close_TSLA,Close_GOOGL,Clos
e_GOOG,Close_META,Close_AVGO)) %>% make_dataset


data_cnn<-c(inputs, targets) %<-%
  iter_next(as_iterator(data_cnn))

######################## 5 days ahead - 10 lag - WITH FLAT
#################

train_x <- data_cnn[[1]][1:560,,]
train_y <-data_cnn[[2]][1:560,]

#val_set
val_x <-data_cnn[[1]][561:680,,]
val_y <-data_cnn[[2]][561:680,]

#test_set
test_x <-data_cnn[[1]][681:806,,]
test_y_og <-data_cnn[[2]][681:806,]


train_y <-to_categorical(train_y)
val_y <- to_categorical(val_y)
test_y <- to_categorical(test_y_og)
############################################################

length(input_data_colnames)
inputs <- layer_input(shape = c(time_steps,length(input_data_colnames)))

outputs <- inputs %>%
  layer_conv_1d(filters = 64, kernel_size = 2, stride=1, padding =
'causal', activation = "relu") %>%
  layer_conv_1d(filters = 64, kernel_size = 2, stride=1, padding =
'causal', activation = "relu") %>%
  layer_max_pooling_1d(pool_size = 2L, stride=1) %>%
  layer_flatten() %>%
  layer_dense(units = 64, activation = 'relu') %>%
  layer_dense(units = 64, activation = 'relu') %>%
layer_dense(units = 3, activation = 'softmax')


# create and compile model
model <- keras_model(inputs = inputs,
                     outputs = outputs) %>% compile(
                       loss = "categorical_crossentropy",
                       # optimizer = optimizer_rmsprop(learning_rate =
0.0001),
                       optimizer = optimizer_adam(learning_rate = 0.0001),

                       metrics = "accuracy"
                     )

summary(model)
```

```
model %>% plot_model


checkpoint <- list(
  callback_model_checkpoint(
    # filepath =
"C:/Users/Kon/Documents/ANN_MOdels/best_{epoch:04d}_{val_loss:.3f}.keras",
    filepath =
"C:/Users/Kon/Documents/ANN_Models/uni_cnn_movement_6.keras", #
    save_best_only = TRUE,
    monitor = "val_loss",
    verbose = 1)
)


history <- model %>%
  fit(train_x,
      train_y,
      epochs = 200,
      batch_size = 8,
      validation_data = list(val_x,val_y),
      callbacks = checkpoint
  )


model %>% evaluate(test_x,test_y)
plot(history)
best_model<-
load_model_tf("C:/Users/Kon/Documents/ANN_Models/uni_cnn_movement_6.keras")
best_model %>% evaluate(test_x,test_y)

#' [Down Flat Up]
pred_uni_movement <- best_model %>% predict(test_x)

pred_binary <- ifelse(pred_uni_movement[,1] > 0.33, "0",
                      ifelse(pred_uni_movement[,2] >= 0.33, "1", "2"))

pred_binary <- max.col(pred_uni_movement)-1

conf_mat <- confusionMatrix(factor(pred_binary,levels = c(0, 1, 2)),
as.factor(as.vector(test_y_og)));conf_mat
```

Multivariate CNN Classification με Υπο-Μοντέλα

```
#' [SUB MODELS]
Data_Movement <- Data_Complete

# Up Down
Data_Movement$Movement <- NA
for(i in 1:(nrow(Data_Movement)-3)){
  if(Data_Movement$Close_QQQ[i+3] > Data_Movement$Close_QQQ[i]){
    Data_Movement$Movement[i] <- 1
  }
  else {
    Data_Movement$Movement[i] <- 0
  }
}

Data_Movement <- na.omit(Data_Movement)
Data_Movement %>% tail(15)
table(Data_Movement$Movement)


sampling_rate <- 1
time_steps <- 10
batch_size <- 1000
pred_days <- 1


# Scenario 2
input_data_colnames_tower_1 <- Data_Scaled %>% select(Close_QQQ) %>%
colnames()
input_data_colnames_tower_2 <- Data_Scaled %>% select(c(Open, High,
Low,Close_MSFT)) %>% colnames()


df_to_inputs_and_targets_1 <- function(df) {
  inputs <- df[input_data_colnames_tower_1] %>%
    as.matrix()
  targets <- Data_Movement$Movement %>% as.matrix %>% embed(pred_days) %>%
as.data.frame %>% rev %>% as.matrix
   list (
    inputs,
    tail(targets, (-time_steps+1))
  )
}


make_dataset_1 <- function(df)
{ c(inputs, targets) %<-%
    df_to_inputs_and_targets_1(df)
  timeseries_dataset_from_array(
    inputs, targets,
    sampling_rate = sampling_rate,
    sequence_length = time_steps,
    shuffle = FALSE,
    batch_size = batch_size
  )
}

df_to_inputs_and_targets_2 <- function(df) {
```

```
  inputs <- df[input_data_colnames_tower_2] %>%
    as.matrix()
  targets <- Data_Movement$Movement %>% as.matrix %>% embed(pred_days) %>%
as.data.frame %>% rev %>% as.matrix
  list (
    inputs,
    tail(targets, (-time_steps+1))
  )
}


make_dataset_2 <- function(df)
{ c(inputs, targets) %<-%
    df_to_inputs_and_targets_2(df)
  timeseries_dataset_from_array(
    inputs, targets,
    sampling_rate = sampling_rate,
    sequence_length = time_steps,
    shuffle = FALSE,
    batch_size = batch_size
  )
}


data_cnn_tower_1<-Data_Movement %>% select(Close_QQQ) %>% make_dataset_1
data_cnn_tower_2<-Data_Movement %>% select(Open,High,Low,Close_MSFT) %>%
make_dataset_2



data_cnn_tower_1 <- c(inputs, targets) %<-%
  iter_next(as_iterator(data_cnn_tower_1))

data_cnn_tower_2 <- c(inputs, targets) %<-%
  iter_next(as_iterator(data_cnn_tower_2))


######################## 3 days ahead - 10 lag - WITH FLAT
#########################
# train set
train_x_tw1 <- data_cnn_tower_1[[1]][1:564,,]
train_y_tw1 <-data_cnn_tower_1[[2]][1:564,]

#val_set
val_x_tw1 <-data_cnn_tower_1[[1]][561:684,,]
val_y_tw1 <-data_cnn_tower_1[[2]][561:684,]

#test_set
test_x_tw1 <-data_cnn_tower_1[[1]][685:808,,]
test_y_tw1 <-data_cnn_tower_1[[2]][685:808,]


# train set
train_x_tw2 <- data_cnn_tower_2[[1]][1:564,,]
train_y_tw2 <-data_cnn_tower_2[[2]][1:564,]

#val_set
val_x_tw2 <-data_cnn_tower_2[[1]][561:684,,]
val_y_tw2 <-data_cnn_tower_2[[2]][561:684,]
```

```
#test_set
test_x_tw2 <-data_cnn_tower_2[[1]][685:808,,]
test_y_tw2 <-data_cnn_tower_2[[2]][685:808,]

#########################################################
QQQ_data <- layer_input(shape =
c(time_steps,length(input_data_colnames_tower_1)), name = 'Close_QQQ'
)


Other_data <- layer_input(shape =
c(time_steps,length(input_data_colnames_tower_2)), name = 'Oth-
er_Timeseries'
)

tower_1 <- QQQ_data %>%
  layer_conv_1d(filters = 32, kernel_size = 2, padding='causal', activa-
tion='relu')


tower_2 <- Other_data %>%
  layer_conv_1d(filters = 32, kernel_size = 2, padding='causal', activa-
tion='relu')

output <- layer_concatenate(c(tower_1, tower_2), axis = 1) %>%
  layer_flatten() %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dense(units = 1, activation = 'sigmoid')



checkpoint <- callback_model_checkpoint(
  # filepath =
"C:/Users/Kon/Documents/AAPL_ANN/AAPL_model_{epoch:04d}_{val_loss:.3f}.h5",
  filepath = "C:/Users/Kon/Documents/ANN_Models/cnn_sub_move_3.keras",
  save_best_only = TRUE,
  monitor = "val_loss",
  verbose = 1
)



# create and compile model
model <- keras_model(inputs = c(QQQ_data, Other_data),
                     outputs = output) %>% compile(
                       loss = "binary_crossentropy",
                       optimizer = optimizer_adam(learning_rate = 0.0001),
                       metrics = "accuracy"
                     )

summary(model)
model %>% plot_model

history <- model %>%
  fit(x = list(QQQ_data = train_x_tw1 ,
               Other_data = train_x_tw2),
      y = train_y_tw1,
      epochs = 100,
```

```
      batch_size = 8,
      validation_data = list(x_val= c(val_x_tw1 , val_x_tw2), y_val =
val_y_tw1),
      # validation_split = 0.2,
      callbacks = checkpoint
  )


best_model <-
load_model_tf("C:/Users/Kon/Documents/ANN_Models/cnn_sub_move_3.keras")



model %>% evaluate(x = list(QQQ_data = test_x_tw1,
                            Other_data = test_x_tw2),
                   y = test_y_tw1
)


best_model %>% evaluate(x = list(QQQ_data = test_x_tw1,
                            Other_data = test_x_tw2),
                   y = test_y_tw1
)


plot(history)


pred_sub_movement <- best_model %>% predict(list(test_x_tw1 ,test_x_tw2 ))
pred_binary <- ifelse(pred_sub_movement> 0.5, 1, 0)
conf_mat <- confusionMatrix(factor(pred_binary,levels = c(0, 1)),
as.factor(as.vector(test_y_tw1)));conf_mat

precision <- conf_mat$byClass["Precision"]
recall <- conf_mat$byClass["Recall"]
f1_score <- conf_mat$byClass["F1"]
accuracy <- conf_mat$overall["Accuracy"]

cat("Accuracy:", accuracy, "\n")
cat("Precision:", precision, "\n")
cat("Recall:", recall, "\n")
cat("F1 Score:", f1_score, "\n")
```

Παράδειγμα Γραφήματος GGplot (Multi-Step Multivariate MLP)

```
dates <- as.Date(rownames(Data_Complete))
input <- Data_Complete %>%
  select(Close_QQQ) %>%
  mutate(across(.cols = everything(), .fns = list(lag1 = ~ lag(., n = 1),
                                                  lag2 = ~ lag(., n = 2),
                                                  lag3 = ~ lag(., n = 3))))
%>%
  select(-Close_QQQ)


#
output <- Data_Complete["Close_QQQ"] %>%
  mutate(across(.cols = everything(), .fns = list(lead1 = ~ lead(., n = 1),
                                                  lead2 = ~ lead(., n = 2),
                                                  lead3 = ~ lead(., n = 3),
                                                  lead4 = ~ lead(., n =
4))))


input_output_mx<- cbind(input,output) %>% na.omit()

input_matrix <- input %>% na.omit %>% as.matrix
pred_multi_mpl_total <- best_model %>% predict(input_matrix)
pred_multi_mpl_total<- head(pred_multi_mpl_total,-4)

#' [GGplots]
dates <- as.Date(rownames(Data))
dates_5step = tail(dates,-7)
dates_1step = tail(head(dates,-4),-3)


#' [Actual vs Predicted]
# 5th day prediction
df_pred <- data.frame(
  QQQ_Price = pred_multi_mpl_total[,5],
  Date = dates_5step,
  Data = c(rep("In Sample",691),rep("Out Of Sample",122))
)

df_actual <- data.frame(
  QQQ_Price = input_output_mx$Close_QQQ_lead4,
  Date = dates_5step
  Data =rep("Actual Values",813)
)

# next day prediction
df_pred <- data.frame(
  QQQ_Price = pred_multi_mpl_total[,1],
  Date = dates_1step,
  Data = c(rep("In Sample",691),rep("Out Of Sample",122))
)

df_actual <- data.frame(
  QQQ_Price = input_output_mx$Close_QQQ,
  Date = dates_1step,
```

```
  Data =rep("Actual Values",813)
)


fig_all <- ggplot() +
   aes(Date,QQQ_Price) + geom_line(data = df_pred, aes(color = Da-
ta,linetype = Data),show.legend = FALSE,size = 1) +
  geom_line(data = df_actual, aes(Date,QQQ_Price, color = Data,linetype =
Data), size=1)+
  geom_vline(xintercept = as.numeric(dates_1step[692]), color = "red",
linetype = "dashed") +

  #Notation for out of samople
  annotate("label", x = as.Date(dates_1step[790]), y=370,
           alpha = .92, label =" Out-Of-Sample ", color = "#ED254EFF", fill
= "#011936FF",family = "Times New Roman",size=6)+

  # Arrow for out of sample
  geom_segment(aes(x = as.Date(dates_1step[730]) , xend =
as.Date(dates_1step[692]), y = 370, yend = 370),lineend = "round",linejoin
= "round", size =1.5, arrow = arrow(length = unit(0.2, "cm")), color =
"#FF427A")

  # Customize the legend
  scale_color_manual(
    values = c("#011936FF", "#F9DC5CFF","#ED254EFF"),#
  ) +
  scale_linetype_manual(
    values = c("solid", "solid", "solid"),
  ) +
  labs(title="Πρόβλεψη της Τιμής του QQQ Trust",
       subtitle = "Multivariate Multi-Step MLP - 1 Day Ahead",
       caption = "Δεδομένα από 1 Οκτωβρίου, 2019  έως 30 Δεκεμβρίου, 2022
",
       x = "Date",y = "QQQ  ETF  Price") +
  theme_dark() +

  #Theme colors
  theme(text = element_text(family ="Times New Roman"),
        title = element_text(color = "#F4FFFDFF"),
        plot.subtitle = element_text(color = "#F4FFFDFF",size=17),
        plot.caption = element_text(color = "#F4FFFDFF", face = "ital-
ic",size = 17),
        panel.background = element_rect(fill = "gray67",
                                        color = "#F4FFFDFF"),
        panel.border = element_rect(colour ="#FF427A",fill=NA, size = 1.5
),
        plot.background = element_rect(fill = "#011936FF",
                                       color = "#F4FFFDFF"),
        axis.text = element_text(color = "white",size=20),
        axis.ticks =element_line(color = "#F4FFFDFF"),
        axis.title.y = element_text(margin = margin(t=0, r=10, b=0,
l=0),size=20),
        axis.title.x = element_text(margin = margin(t=10, r=0, b=0,
l=0),size=20),
        plot.title = element_text(hjust = 0.5, face = "bold", size = 25),
        legend.background = element_rect(fill = "gray67",
                                         color ="#ED254EFF"),
        legend.text = element_text(color = "#F4FFFDFF",size=17),
```

```
        legend.title = element_text(face = "bold",color="#F4FFFDFF", size =
20),
        legend.position = "right") +
  scale_x_date(date_labels="%b-%d",date_breaks  ="3 month");fig_all

##############################################################################
#' [Actual vs Predicted]
fig_out <- ggplot() +
  aes(Date,QQQ_Price) + geom_line(data = df_pred[692:nrow(df_pred), ],
aes(color = Data,linetype = Data),show.legend = FALSE,size = 1) +
  geom_point(data = df_pred[692:nrow(df_pred), ],aes(color = Da-
ta,shape=Data),show.legend = TRUE,size=2) +
  geom_line(data = df_actual[692:nrow(df_actual), ], aes(Date,QQQ_Price,
color = Data,linetype = Data), size=1)+
  geom_point(data = df_actual[692:nrow(df_actual), ],aes(color = Da-
ta,shape=Data),show.legend = TRUE,size=2) +
  geom_vline(xintercept = as.numeric(dates_1step[692]), color = "red",
linetype = "dashed") +
   # Customize the legend
  scale_color_manual(
    values = c("#011936FF","#ED254EFF"),#
  ) +
  scale_linetype_manual(
    values = c("solid",  "solid"),
  ) +
  labs(title="Out-Of-Sample Προβλέψεις της Τιμής του QQQ Trust",
       subtitle = "Multivariate Multi-Step MLP - 1 Day Ahead",
       caption = "Δεδομένα από 5 Ιουλίου, 2022  έως 30 Δεκεμβρίου, 2022 ",
       x = "Date",y = "QQQ  ETF  Price") +
  theme_dark() +

  #Theme colors
  theme(text = element_text(family ="Times New Roman"),
        title = element_text(color = "#F4FFFDFF"),
        plot.subtitle = element_text(color = "#F4FFFDFF",size=17),
        plot.caption = element_text(color = "#F4FFFDFF", face = "ital-
ic",size = 17),
        panel.background = element_rect(fill = "gray67",
                                        color = "#F4FFFDFF"),
        panel.border = element_rect(colour ="#FF427A",fill=NA, size = 1.5
),
        plot.background = element_rect(fill = "#011936FF",
                                       color = "#F4FFFDFF"),
        axis.text = element_text(color = "white",size=20),
        axis.ticks =element_line(color = "#F4FFFDFF"),
        axis.title.y = element_text(margin = margin(t=0, r=10, b=0,
l=0),size=20),
        axis.title.x = element_text(margin = margin(t=10, r=0, b=0,
l=0),size=20),
        plot.title = element_text(hjust = 0.5, face = "bold", size = 25),
        legend.background = element_rect(fill = "gray67",
                                         color ="#ED254EFF"),
        legend.text = element_text(color = "#F4FFFDFF",size=17),
        legend.title = element_text(face = "bold",color="#F4FFFDFF", size =
20),
        legend.position = "right") +
  scale_x_date(date_labels="%b-%d",date_breaks  ="1 month");fig_out

ggdraw() +
```

```
  draw_plot(fig_all, 0, .5, 1, .5) +
  draw_plot(fig_out, 0, 0, 1, .5)

# ggsave("mlp_multi_5step_ggplot.png",width = 480,height = 360,units =
"mm")
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Abanda, A., Mori, U. and Lozano, J. (2018). A review on distance based time series classification. *Data Mining and Knowledge Discovery*, 33, 378–412

[2] Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*, Springer Cham, Switzerland

[3] Atkinson, P. M., and Tatnall A. R. L. (1997). Introduction Neural networks in remote sensing, *International Journal of Remote Sensing*. 18, Issue 4, 699-709

[4] Atwan, T. A. (2022). *Time Series Analysis with Python Cookbook: Practical recipes for exploratory data analysis, data preparation, forecasting, and model evaluation*, Packt Publishing, Birmingham.

[5] Boutsikas, M. (2020). *Εισαγωγή στην Ανάλυση Χρονοσειρών*, Ανάλυση Χρονολογικών Σειρών, Πανεπιστήμιο Πειραιώς, Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης, Πειραιάς.

[6] Box, G. E. P., Jenkins, G. M. and Reinsel G. C. (2008). *Time Series Analysis: Forecasting and Control*, 4th ed., John Wiley, New York.

[7] Brownlee, J. (2019). *Better Deep Learning Train Faster, Reduce Overfitting and Make Better Predictions*, Machine Learning Mastery.

[8] Burkov, A. (2019). *The Hundred Page Machine Learning Book*, Andriy Burkov.

[9] Chatfield, C. (2003). *The Analysis of Time Series An Introduction*, 6th ed., Chapman and Hall/CRC, New York.

[10] Cybenco, G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Control Signal Systems* 2, 303–314.

[11] Dogo, E. M., Afolabi, O. J., Nwulu, N. I., Twala B. and Aigbavboa, C. O. (2018). A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks, *International Conference on Computational Techniques, Electronics and Mechanical Systems* (CTEMS), India.

[12] Dozie, K. C. N, Mbachu. H. I. and Raymond, M. C. (2020). The Analysis of Mixed Model in Time Series Decomposition, *International Journal of Probability and Statistics*.

[13] Feng, J. and Lu S. (2019). *Performance Analysis of Various Activation Functions in Artificial Neural Networks*, Journal of Physics Conference Series.

[14] Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed., O'Reilly Media, California.

[15] Goodfelow, I., Bengio Y. and Courville A. (2016). *Deep Learning*, MIT Press, Massachusetts.

[16] Granger, C. W. J. and Newbold P. (1986). *Forecasting Economic Time Series*, 2nd ed., Academic Press, California.

[17] Graves, A. (2019). *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer Berlin, Heidelberg.

[18] Grazzini, J. (2012). Analysis of the Emergent Properties: Stationarity and Ergodicity, *Journal of Artificial Societies and Social Simulation* vol. 15(2), pages 1-7.

[19] Haji, S H. and Abdulazeez A. M. (2021). Comparison of Optimization Techniques Based on the Gradient Descent Algorithm: A Review, *Palarch's Journal Of Archaeology Of Egypt / Egyptology*.

[20] Haykin, S. (1999). *Neural Networks a Comprehensive Foundation*, 2nd ed., Prentice Hall International, New Jersey.

[21] Hyndman, R. J. and Athanasopoulos G. (2018). *Forecasting: principles and practice*, 2nd ed., OTexts, Australia.

[22] Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, *Proceedings of the 32nd International Conference on International Conference on Machine Learning*.

[23] Lewis, N. D. (2017). *Neural Networks for Time Series Forecasting with R: An Intuitive Step by Step Blueprint for Beginners*, CreateSpace Independent Publishing Platform, South Carolina.

[24] Magaia, N., Mastorakis G., Mavromoustakis, C., Pallis. E. and Markakis E. K. (2021). *Intelligent Technologies for Internet of Vehicles*, Springer Nature, Switzerland.

[25] Meira, Wanger Jr and Zaki, M. J. (2020). *Data Mining and Machine Learning: Fundamental Concepts and Algorithms*, 2nd ed., 25, 637-670, Cambridge University Press, Cambridge

[26] Patterson, J. and Gibson A. (2017). *Deep LearningA Practitioner's Approach*, O'Reilly Media, California.

[27] Reed, R. and Marks II, R. J. (1999). *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, MIT Press, Massachusetts.

[28] Ruder, S. (2016). *An overview of gradient descent optimization algorithms*, Insight Centre for Data Analytics, NUI Galway

[29] Shanmugamani, R. (2018). *Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras*, Packt Publishing, Birmingham.

[30] Shmueli, G. and Lichtendahl, Jr. K. C. (2016). *Practical Time Series Forecasting with R: A Hands-On Guide*, 2nd ed., Axelrod Schnall Publishers, Florida.

[31] Shumway, R. H and Stoffer, D. S. (2014). *Time Series Analysis and Its Applications: With R Examples*, 3rd ed., Springer, New York.

[32] Shumway, R. H and Stoffer, J. (2020). *Introduction to Time Series Forecasting With Python: How to Prepare Data and Develop Models to Predict the Future*, Machine Learning Mastery.

[33] Tang, Y., Huang, Y., Wu, Z., Meng, H., Xu, M. and Cai L. (2016). Question detection from acoustic features using recurrent neural network with gated recurrent unit, *IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), China.

[34] Wang, Q., Ma Y., Zhao K. and Tian Y. (2020). *A Comprehensive Survey of Loss Functions in Machine Learning*, Springer-Verlag, Germany.