ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΕΤΑΠΤΥΧΙΑΚΟ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

Transformer methods for Financial Forecasting

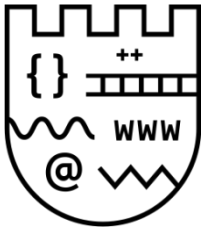
Χρήση μεθόδων Transformer για Χρηματοοικονομικές Προβλέψεις

ΑΝΑΣΤΑΣΙΟΣ ΜΠΑΡΜΠΑΚΑΣ

ΑΕΜ: 91

Επιβλέπων Καθηγητής:

Βλαχάβας Ιωάννης



Μάρτιος 2023

Περίληψη

Ένας από τους σημαντικότερους τομείς έρευνας της χρηματοοικονομικής επιστήμης είναι η πρόβλεψη των χρηματοοικονομικών τάσεων, που έχει σαν βασικό στόχο την παροχή προβλέψεων με μεγάλη ακρίβεια στους λήπτες χρηματιστηριακών αποφάσεων για τις μελλοντικές χρηματοοικονομικές τάσεις. Οι πρόσφατες εξελίξεις στη Μηχανική Μάθηση και την Τεχνητή Νοημοσύνη έχουν οδηγήσει στην ανάπτυξη πολυάριθμων τεχνικών που είναι ικανές να βελτιώσουν την ακρίβεια των οικονομικών προβλέψεων. Σε αυτή τη διατριβή, διερευνούμε τη χρήση μεθόδων Transformer για χρηματοοικονομικές προβλέψεις.

Τα Transformer είναι ένας νέος τύπος αρχιτεκτονικής Νευρωνικών Δικτύων που εισήχθη το 2017 και έκτοτε έχει γίνει ένα ισχυρό εργαλείο κυρίως για εργασίες Επεξεργασίας Φυσικής Γλώσσας. Οι μέθοδοι Transformer είναι γνωστές για την ικανότητά τους να συλλαμβάνουν αποτελεσματικά τα long-term dependencies, κάτι που είναι κρίσιμο για τις οικονομικές προβλέψεις. Σε αυτή τη διατριβή, γίνεται μια διερεύνηση της χρήσης μεθόδων Transformer για πρόβλεψη οικονομικών χρονοσειρών καθώς και σύγκριση των αποδόσεών τους με παραδοσιακές μεθόδους πρόβλεψης όπως τα LSTM.

Καθώς η χρήση μεθόδων Μηχανικής Μάθησης στην χρηματοοικονομική επιστήμη αυξάνεται με ταχείς ρυθμούς, τα ευρήματα αυτής της έρευνας μπορούν να αξιοποιηθούν και να βοηθήσουν τους υπεύθυνους λήψης αποφάσεων του χρηματοοικονομικό κλάδο που βασίζονται σε ακριβείς προβλέψεις, ενημερώνοντας τις επενδυτικές τους στρατηγικές. Η μελλοντική έρευνα θα μπορούσε να εξετάσει την πιθανή εφαρμογή μεθόδων Transformer σε άλλες εργασίες χρηματοοικονομικών προβλέψεων, όπως επίσης και να διερευνήσει τη χρήση άλλων προηγμένων τεχνικών Βαθιάς Μάθησης για περαιτέρω βελτίωση της ακρίβειας των οικονομικών προβλέψεων

Abstract

Financial forecasting has long been an important area of research in finance, with the goal of providing decision-makers with accurate predictions of future financial trends. Recent advances in Machine Learning and Artificial Intelligence have led to the development of new techniques that are capable of improving the accuracy of financial forecasting. In this thesis, we explore the use of Transformer methods for financial forecasting.

The Transformer is a type of neural network architecture that was introduced in 2017 and has since become a powerful tool for natural language processing tasks. Transformer methods are known for their ability to effectively capture long-term dependencies, which is critical for financial forecasting. In this thesis, we investigate the use of Transformer methods for financial time series forecasting, and compare their performance to traditional forecasting methods such as LSTM.

As the use of machine learning in finance continues to grow, the findings of this research have significant implications for decision-makers in the financial industry who rely on accurate predictions to inform their investment strategies. Future research could explore the application of Transformer methods in other financial forecasting tasks and investigate the use of other advanced deep learning techniques to further improve the accuracy of financial forecasting.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή μου κ. Βλαχάβα Ιωάννη, για την ευκαιρία που μου έδωσε να εκπονήσω την πτυχιακή μου εργασία, καθώς και για την αμέτρητη βοήθεια που μου προσέφερε λύνοντάς μου απορίες και κάνοντας στοχευμένες υποδείξεις, όπως και τον συν – επιβλέποντα Κουλούμπρη Ελευθέριο που με καθοδήγησε με το καλύτερο τρόπο.

Επίσης θα ήθελα να ευχαριστήσω όλους τους συμφοιτητές μου, η βοήθεια των οποίων ήταν πολλές φορές σωτήρια. Ξεχωριστά θα ήθελα να ευχαριστήσω τον πολύ καλό μου φίλο και συμφοιτητή Χατζηδάκη Ραφαήλ ο οποίος μου στάθηκε και με βοήθησε κατά την διάρκεια των σπουδών μου στο μεταπτυχιακό της Τεχνητής Νοημοσύνης

Τέλος, δεν μπορώ να παραλείψω να ευχαριστήσω την οικογένειά μου και τη σύντροφό μου, οι οποίοι μου συμπαραστάθηκαν, παρά τις οποιεσδήποτε δυσκολίες, πιστεύοντας σε εμένα και στηρίζοντας κάθε προσπάθειά μου.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Πίνακας Διαγραμμάτων

Chapter 1

Εισαγωγή

1.1 Το πρόβλημα

Η Χρηματοοικονομική Επιστήμη είναι ένας τομέας της οικονομίας που ασχολείται με τη μελέτη και ανάλυση των χρηματοοικονομικών αγορών, των επενδύσεων και των χρηματοοικονομικών προϊόντων. Μία σημαντική πτυχή της, είναι η πρόβλεψη των μελλοντικών τάσεων των χρηματοοικονομικών αγορών. Η πρόβλεψη αυτή αποτελεί ένα πολύπλοκο πρόβλημα και γίνεται όλο και πιο δύσκολη λόγω της αύξησης του όγκου των δεδομένων και της αβεβαιότητας που επικρατεί στις χρηματοοικονομικές αγορές.

1.2 Σκοπός – Στόχοι της εργασίας

Η Μηχανική Μάθηση είναι ένας τομέας της Τεχνητής Νοημοσύνης που ασχολείται με τη μελέτη και το σχεδιασμό αλγορίθμων και μοντέλων που επιτρέπουν σε ένα σύστημα να βελτιώνει την απόδοσή του σε σχέση με ένα συγκεκριμένο κριτήριο, χρησιμοποιώντας δεδομένα. Η Μηχανική Μάθηση μπορεί να χρησιμοποιηθεί για την πρόβλεψη των τάσεων των χρηματοοικονομικών αγορών καθώς υπάρχει μια πληθώρα εργασιών και ερευνών με αξιόλογα ευρήματα και τεχνικές. Τέτοιες τεχνικές μπορεί να είναι παραδοσιακά στατιστική μοντέλα μέχρι και πιο πολύπλοκες μέθοδοι Βαθιάς Μάθησης. Μεταξύ αυτών των τεχνικών, τα μοντέλα που βασίζονται σε Transformer εμφανίστηκαν πρόσφατα ως μια πολλά υποσχόμενη προσέγγιση για την πρόβλεψη χρονοσειρών, παρέχοντας state-of-the-art επιδόσεις σε μια σειρά εργασιών. Σε αυτή τη διατριβή, διερευνούμε την εφαρμογή μοντέλων που βασίζονται σε Transformers στη χρηματοοικονομική πρόβλεψη, διερευνώντας την αποτελεσματικότητά τους και τις δυνατότητές τους για τη βελτίωση της ακρίβειας των προβλέψεων.

1.3 Διάρθρωση της εργασίας

Στη συνέχεια, παρουσιάζεται η δομή την παρούσας εργασίας ανά κεφάλαιο και ανά θεματική ενότητα

- Κεφάλαιο 2: Γίνεται μια εκτενής εισαγωγή στην Μηχανική Μάθηση και τους διάφορους τύπους της όπως Μάθηση με Επίβλεψη, Μάθηση χωρίς Επίβλεψης καθώς και Ενισχυτική Μάθηση. Σε αυτό το κεφάλαιο γίνεται και η παρουσίαση της τεχνικής των Transformer αναλύοντας και τους λόγους για του οποίους επιλέχθηκαν.

- Κεφάλαιο 3: Αυτό το κεφάλαιο παρέχει μια επισκόπηση των μεθόδων Τεχνικής Ανάλυσης που χρησιμοποιούνται στις χρηματοοικονομικές προβλέψεις συμπεριλαμβανομένης της διαγραμματικής ανάλυσης και των τεχνικών δεικτών - ταλαντωτών . Η διαγραμματική ανάλυση συμπεριλαμβάνει τις Γραμμές Τάσεις, μια

σειρά από σχηματισμούς όπως Κεφάλι και Ώμοι, Κούπα Χερούλι, Διπλή Κορυφή Διπλός Πυθμένας κ.α.. Επιπλέον γίνεται μια εκτενής αναφορά στα Χάσματα και στους Όγκους Συναλλαγών. Στους τεχνικούς δείκτες περιλαμβάνεται μια πληθώρα δεικτών και ταλαντωτών όπως ο Κινητός Μέσος Όρος, ο Δείκτης Σχετικής Ισχύος, ο Ταλαντωτής Τιμών, ο Ταλαντωτής Williams κ.α..

- Κεφάλαιο 4: Εδώ παρουσιάζεται το σύνολο δεδομένων (dataset) που χρησιμοποιήθηκε καθώς και η προ - επεξεργασία που υπέστη προτού τροφοδοτηθεί στο μοντέλο Transformer. Στη συνέχεια, παρουσιάζεται το μοντέλο Transformer που χρησιμοποιήθηκε για την χρηματοοικονομική πρόβλεψη με μια εκτενή περιγραφή της αρχιτεκτονικής του. Επιπλέον, επεξηγείται η loss function, ο optimizer καθώς και η διαδικασία της εκπαίδευσης του μοντέλου. Τέλος παρουσιάζονται τα αποτελέσματα του μοντέλου και γίνεται μια σύγκριση με τα αποτελέσματα ενός παραδοσιακού LSTM Νευρωνικού Δικτύου στο ίδιο dataset.

- Κεφάλαιο 5: Στο παρόν κεφάλαιο γίνεται μια σύνοψη της εργασίας παρουσιάζοντας τις δυσκολίες που εμφανίστηκαν κατά την διάρκεια της συγγραφής της και γίνεται μια αναφορά στην πιθανή μελλοντική επέκτασή της.

Chapter 2

Μηχανική Μάθηση

Η ικανότητα του ανθρώπου να μαθαίνει θεωρείται βασική για να μπορέσει να επιβιώσει, και αναπτύσσεται από τα πρώτα χρόνια της ζωής του. Η μάθηση ως ικανότητα, έχει μελετηθεί από την επιστημονική κοινότητα διεξοδικά χωρίς όμως να καταφέρει να γίνει πλήρως κατανοητή. Αυτό όμως δεν εμπόδισε ειδικούς της επιστήμης των υπολογιστών να εκμεταλλευτούν τις υπάρχουσες γνώσεις και να δημιουργήσουν συστήματα τα οποία μιμούνται την ανθρώπινη ικανότητα μάθησης και προσαρμογής (Andreas C Müller).

Ο Tom M. Mitchell το 1997 έδωσε τον εξής ορισμό για τη Μηχανική Μάθηση:

«Ένα πρόγραμμα υπολογιστή μπορούμε να θεωρήσουμε ότι έχει την ικανότητα να μαθαίνει από εμπειρία Ε ως προς μια κλάση εργασιών Τ και με ένα μέτρο απόδοσης Ρ, αν η επίδοσή του σε εργασίες τις κλάσεις Τ, βελτιώνεται με τη χρήση της εμπειρίας Ε, σύμφωνα με το μέτρο επίδοσης Ρ.»

Η ανάγκη ύπαρξης της Μηχανικής Μάθησης, αντικατοπτρίζεται στην αδυναμία εκτέλεσης κάποιων πολύπλοκων εργασιών από απλά προγράμματα υπολογιστή. Τέτοιου είδους εργασίες είναι κατά βάση ενέργειες που πραγματοποιούν οι άνθρωποι στην καθημερινότητά τους όπως οδήγησης οχήματος, κατανόηση εικόνας και μοτίβων. Επιπλέον μεγάλη εφαρμογή έχουν συστήματα Μηχανικής Μάθησης σε εργασίες που ξεπερνούν τα όρια των ανθρώπινων δυνατοτήτων, όπως για παράδειγμα σε συστήματα πρόγνωσης καιρικών φαινομένων ή ακόμα και στις μηχανές αναζήτησης.

2.1 Είδη Μηχανικής Μάθησης

Η Μηχανική Μάθηση ως επιστημονικό πεδίο είναι ιδιαίτερα ευρύ και ως συνέπεια μπορούμε να τη διακρίνουμε σε επιμέρους πεδία ανάλογα με τον τρόπο μάθησης και τη συμμετοχή του ανθρώπινου παράγοντα σε αυτήν. Γενικώς μπορούμε να διακρίνουμε τρία σημαντικά είδη Μηχανικής Μάθησης τα οποία αναλύονται παρακάτω.

- Μάθηση με Επίβλεψη
- Μάθηση χωρίς Επίβλεψη
- Μάθηση με Ενίσχυση

2.2 Μάθηση με Επίβλεψη (Supervised Learning)

Στην Μάθηση με Επίβλεψη, το σύστημα καλείται να ανακαλύψει μοτίβα και επαναλαμβανόμενες καταστάσεις μέσα από έτοιμα κατηγοριοποιημένα δεδομένα, γνωρίζοντας τα επιθυμητά αποτελέσματα. Βασικός στόχος είναι να αντιστοιχίσει τα δεδομένα με τα αποτελέσματά τους. Τα προβλήματα που αντιμετωπίζουν αλγόριθμοι επιβλεπόμενης μηχανικής μάθησης διακρίνονται σε δύο κατηγορίες:

- Προβλήματα Ταξινόμησης. Στα συγκεκριμένα προβλήματα γίνεται προσπάθεια δημιουργίας μοντέλων πρόβλεψης διακριτών τάξεων. Δηλαδή η αντιμετώπιση αυτών των προβλημάτων έχει να κάνει με την ικανότητα διαχωρισμού και κατηγοριοποίησης αντικειμένων βάση των ειδικών χαρακτηριστικών που διαθέτουν.

- Προβλήματα Παρεμβολής. Σε αυτά τα προβλήματα καλούμαστε να προβλέψουμε κατά βάση αριθμητικές τιμές. Ένα πεδίο στο οποίο εμφανίζονται προβλήματα παρεμβολής είναι η Χρηματοοικονομική Τεχνολογία (Financial Technology, FINTECH), όπου στην ουσία εφαρμόζονται αλγόριθμοι μηχανικής μάθησης για τη πρόβλεψη των χρηματιστηριακών μετοχών.

Ακολουθούν ορισμένοι από τους πιο ευρέως χρησιμοποιούμενους αλγορίθμους Μάθησης με Επίβλεψη.

2.2.1 Δένδρα απόφασης (Decision Trees)

Τα Δένδρα Απόφασης είναι ένας από τους πιο απλούς αλγορίθμους Μάθησης με Επίβλεψη και χρησιμοποιείται κατά βάση για προβλήματα κατηγοριοποίησης (classification) αλλά υπάρχουν και επεκτάσεις που εμφανίζουν αξιόλογα αποτελέσματα και σε προβλήματα παλινδρόμησης (regression). Ο αλγόριθμος αποσκοπεί στην λήψη απόφασης για την τιμή στόχο κάποιας περίπτωσης (instance) του προβλήματος βασιζόμενο στα χαρακτηριστικά του.

Ένα Δένδρο Απόφασης αποτελείται από κόμβους και ακμές και η δομή του είναι η εξής:

- Κάθε μη τερματικός κόμβος, αντιπροσωπεύει και την ύπαρξη συνθήκης ελέγχου της τιμής κάποιου χαρακτηριστικού των περιπτώσεων του συνόλου δεδομένων
- Κάθε ακμή που ξεκινάει από κάποιον κόμβο, αντιστοιχεί σε μια διαφορετική διακριτή τιμή του χαρακτηριστικού που ελέγχθηκε από την συνθήκη του κόμβου
- Οι τερματικοί κόμβοι, γνωστοί και ως φύλλα του δένδρου, καθορίζουν την τιμή στόχο
- Κάθε μονοπάτι, από τη ρίζα μέχρι κάποιο φύλλο του δένδρου, αποτελεί ένα κανόνα ταξινόμησης ή παρεμβολής, ανάλογα με το αν η μεταβλητή στόχος είναι διακριτή ή συνεχής.

Ένα από τα βασικότερα σημεία αναφοράς των Δένδρων Απόφασης, είναι η επιλογή του κριτηρίου το οποίο θα χρησιμοποιηθεί για των διαχωρισμό ενός κόμβου σε υπό-κόμβους. Ορισμένα από τα πιο συχνώς χρησιμοποιούμενα κριτήρια είναι η Εντροπία της Πληροφορίας (Information Entropy), ο Δείκτης Gini (Gini Index) και ο δείκτης Chi-Square.

Εντροπία της Πληροφορίας: Στόχος του κριτηρίου αυτού, είναι μετά από κάθε διαχωρισμό, να επιλέγεται η μεταβλητή που οδηγεί σε όσο πιο συμπαγές δένδρο γίνεται. Σε ένα πρόβλημα δύο κατηγοριών (κλάσεων) Α και Β, η Εντροπία της Πληροφορίας δίνεται από τη σχέση:

$$E(S) = - p_A log_2 p_A - p_B log_2 p_B$$

Όπου S είναι το σύνολο των δεδομένων εκπαίδευσης στον τρέχοντα κόμβο, p_A είναι το κλάσμα των παραδειγμάτων της κλάσης A του S και p_B είναι το κλάσμα των παραδειγμάτων της κλάσης B του S.

Σαν γενικό κανόνα για C διαφορετικές κατηγορίες, η Εντροπία ορίζεται ως εξής:

$$E(S) = \sum_{i=1}^{c} - p_i \, log_2 \, p_i$$

όπου το p_i το ποσοστό των παραδειγμάτων του S που ανήκουν στην κατηγορία *i.*

Η Εντροπία της Πληροφορίας, είναι επί της ουσίας μια μετρική της ανομοιογένειας του συνόλου δεδομένων S σε σχέση με την εξαρτημένη μεταβλητή που εξετάζουμε. Σε ένα πρόβλημα δύο κλάσεων A και B, εάν το σύνολο των παραδειγμάτων είναι ομοιογενές, δηλαδή όλα τα παραδείγματα του S ανήκουν σε μία κλάση, τότε η τιμή της εντροπίας είναι μηδέν. Σε περίπτωση που τα μισά μέλη ανήκουν στην κλάση A και τα υπόλοιπα στην κλάση B, η εντροπία λαμβάνει την τιμή 1. Ωστόσο σε προβλήματα που το πλήθος των κλάσεων είναι K, οι τιμές που μπορεί να λάβει η εντροπία της πληροφορίας είναι από μηδέν μέχρι log(K).

Στην πράξη , γίνεται χρήση του του κέδρους πληροφορίας (Information Gain) το οποίο, για ένα σύνολο εκπαίδευσης S ο δείκτης του Κέρδους Πληροφορίας G(S, A) , μετρά την μείωση της εντροπίας του S εάν επιλεγεί ως παράμετρος διαχωρισμού η μεταβλητή A. Ο τύπος του Κέρδους Πληροφορίας είναι ο εξής:

$$G(S, A) = E(S) - \sum_{v \in V(A)} \frac{|S_v|}{|S|} E(S_v)$$

Όπου:

- |S| το πλήθος των παραδειγμάτων του υπό εξέταση τρέχοντος κόμβου
- E(S) η εντροπία της πληροφορίας του υπό εξέταση κόμβου
- A η ανεξάρτητη μεταβλητή την οποία και αξιολογούμε ως πιθανή επιλογή για την επόμενη διακλάδωση
- $|S_V|$ το πλήθος των παραδειγμάτων με A = V
- $E(S_V)$ η εντροπία πληροφορίας του υπό εξέταση κόμβου ως προς A = V

Δείκτης Gini: Ο δείκτης Gini μετά πόσο συχνά ένα τυχαία επιλεγμένο στοιχείο του data set θα κατηγοριοποιηθεί λανθασμένα, εάν η κατηγοριοποίησή του είχε γίνει τυχαία, σύμφωνα με την κατανομή των ετικετών του υποσυνόλου. Ο τύπος του δείκτη Gini είναι ο εξής:

$$I_G(p) = 1 - \sum_{i=1}^{K} {p_i}^2$$

Chi-Square: Ο αλγόριθμος Chi-Square, χρησιμοποιείται σε ένα Δένδρο Απόφασης για την εύρεση στατιστικά σημαντικής διαφοράς μεταξύ ενός κόμβου και των υπό-κόμβων του. Ο

τρόπος υπολογισμού του έγκειται στο άθροισμα των τετραγώνων των διαφορών μεταξύ των παρατηρούμενων και των αναμενόμενων τιμών της μεταβλητής στόχου. Γενικώς, όσο μεγαλύτερη τιμή λαμβάνει ο δείκτης Chi-Square, τόσο μεγαλύτερη είναι η στατιστική σημαντικότητα των διαφορών. Στον τύπο του δείκτη Chi-Square παρακάτω, με O_i αναπαρίστανται οι παρατηρούμενες τιμές και με E_i οι αναμενόμενες τιμές.

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

2.2.2 K – Κοντινότεροι Γείτονες (K – Nearest Neighbors)

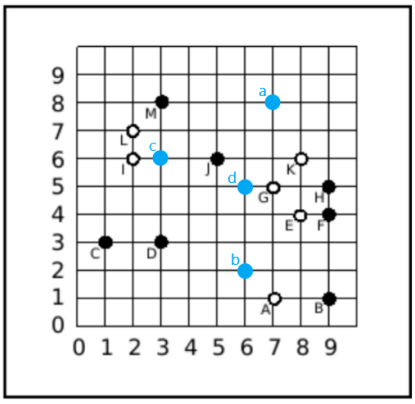
Ο αλγόριθμος των Κ Κοντινότερων Γειτόνων (K Nearest neighbors, K – NN) είναι ένας αλγόριθμος Επιβλεπόμενης Μάθησης (Supervised Learning) ο οποίος χρησιμοποιείται για κατηγοριοποίηση δεδομένων.

Επί της ουσίας, ο αλγόριθμος Κ Κοντινότερων Γειτόνων, κατηγοριοποιεί τα δεδομένα βασιζόμενος στον τύπο των δεδομένων των κοντινότερων «Κ» γειτόνων του. Με το «Κ» αναφερόμαστε στο πλήθος των κοντινότερων γειτόνων που πρόκειται να ελέγξουμε. Συνεπώς ένας αλγόριθμος Τεσσάρων Κοντινότερων Γειτόνων θα κατηγοριοποιεί τα δεδομένα με βάση την κλάση (κατηγορία) των τεσσάρων κοντινότερων γειτόνων τους. Η επιλογή του αριθμού των γειτόνων που θα ελέγξουμε (δηλαδή το Κ), γίνεται μετά από μια σειρά δοκιμών με διαφορετικά Κ κάθε φορά, όπου εν τέλει επιλέγεται ο αριθμός που μας δίνει καλύτερης ποιότητας κατηγοριοποίηση. Ωστόσο, αρκετές φορές αυτό ο αριθμός επιλέγεται αυθαίρετα.

Στο παράδειγμα της Εικόνας 2.1 έχουμε ένα σύνολο από κατηγοριοποιημένα δεδομένα, τα οποία αναπαρίστανται με κεφαλαία γράμματα και ασπρόμαυρες κουκκίδες. Τα δεδομένα αυτού του συνόλου ανήκουν σε δύο κατηγορίες: «Λευκό» για τα δεδομένα με λευκό χρώμα και «Μαύρο» για τα δεδομένα με μαύρο χρώμα.

Επιπλέον, έχουμε ένα σύνολο από μη κατηγοριοποιημένα δεδομένα τα οποία αναπαρίστανται από μπλε κουκίδες και πεζά γράμματα. Αυτά τα δεδομένα καλούμαστε να κατηγοριοποιήσουμε χρησιμοποιώντας το αλγόριθμο K-nn.

Έστω ότι θέτουμε K = 3 (3 – nn) για τη κατηγοριοποίηση των μπλε σημείων *a, b, c, d.*



Εικόνα 2.1 Το σύνολο των δεδομένων πριν την κατηγοριοποίηση

Από την Εικόνα 2.1 παρατηρούμε πως το σημείο “a” έχει ως κοντινότερους γείτονες τα σημεία :

- K (απόσταση = 3)
- G (απόσταση = 3)
- J (απόσταση = 4)
- M (απόσταση = 4)

Εμείς όμως θέλουμε να εφαρμόσουμε έναν 3-nn αλγόριθμο. Συνεπώς πρέπει να λάβουμε υπόψη μόνο τους τρεις κοντινότερους γείτονες. Εδώ θα διατηρήσουμε τα σημεία **K** και **G** και μεταξύ των σημείων **J** και **M** θα κρατήσουμε το J καθώς είναι αλφαβητικά μικρότερο. Εδώ πάλι είναι μια αυθαίρετη σύμβαση σχετικά με ποιο από τα δύο σημεία θα κρατήσουμε, καθώς σε άλλα προβλήματα ενδεχομένως να επιλεχθεί το αλφαβητικά μεγαλύτερο. Πολλές φορές εξαρτάται από τη φύση των δεδομένων η απόφαση αυτή αλλά τις περισσότερες φορές είναι ελάσσονας σημασίας.

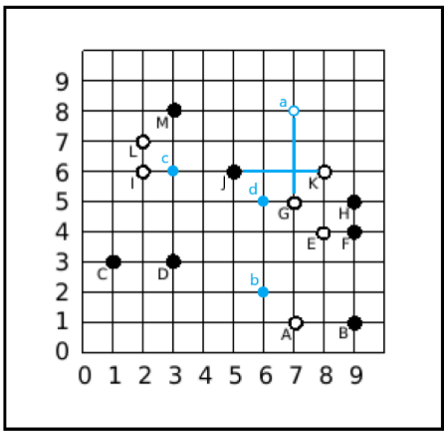
Στο παράδειγμά μας τώρα, παρατηρούμε τις κλάσεις των τριών αυτών γειτόνων οι οποίες είναι οι εξής:

K – Άσπρο

G – Άσπρο

J – Μαύρο

Παρατηρούμε πως οι περισσότεροι γείτονες ανήκουν στην κατηγορία «Άσπρο». Συνεπώς και το σημείο “a” (μπλε) θα κατηγοριοποιηθεί ως άσπρο όπως φαίνεται και στην Εικόνα 2.2

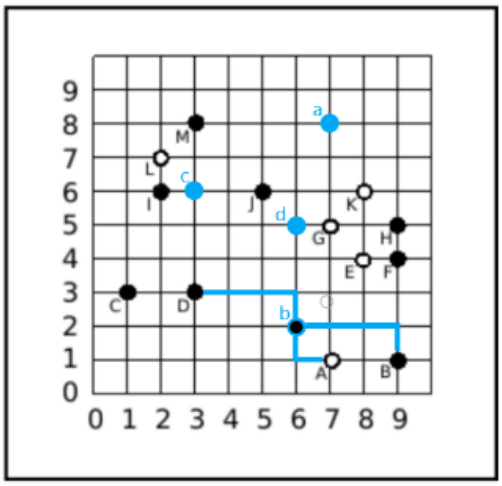


Εικόνα 2.2 Μετά την κατηγοριοποίηση του σημείου “a”.

Αντίστοιχα δουλεύουμε και για τα υπόλοιπα σημεία. Πιο συγκεκριμένα, παρατηρούμε πως το σημείο “b” έχει ως κοντινότερους γείτονες τα σημεία:

- A – Άσπρο (απόσταση = 2)
- B – Μαύρο (απόσταση = 4)
- D – Μαύρο (απόσταση = 4)

Οι περισσότεροι γείτονες ανήκουν στην κατηγορία «Μαύρο». Συνεπώς και το σημείο “b” (μπλε) θα κατηγοριοποιηθεί ως μαύρο

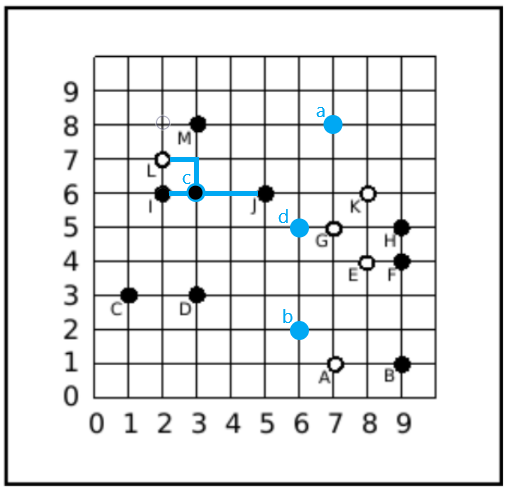


Εικόνα 2.3 Μετά την κατηγοριοποίηση του σημείου “b”.

Παρατηρούμε πως το σημείο “c” έχει ως κοντινότερους γείτονες τα σημεία:

- I – Μαύρο (απόσταση = 1)
- J – Μαύρο (απόσταση = 2)
- L – Άσπρο (απόσταση = 2)

Οι περισσότεροι γείτονες ανήκουν στην κατηγορία «Μαύρο». Συνεπώς και το σημείο “c” (μπλε) θα κατηγοριοποιηθεί ως μαύρο

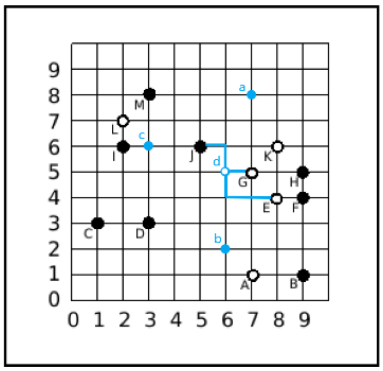


Εικόνα 2.4 Μετά την κατηγοριοποίηση του σημείου “c”.

Παρατηρούμε πως το σημείο “d” έχει ως κοντινότερους γείτονες τα σημεία:

- G – Άσπρο (απόσταση = 1)
- J – Άσπρο (απόσταση = 2)
- E – Μαύρο (απόσταση = 3)

Οι περισσότεροι γείτονες ανήκουν στην κατηγορία «Άσπρο». Συνεπώς και το σημείο “d” (μπλε) θα κατηγοριοποιηθεί ως Άσπρο



Εικόνα 2.5 Μετά την κατηγοριοποίηση του σημείου “d”.

2.2.3 Μηχανή Διανυσμάτων Υποστήριξης (Support Vector Machine, SVM)

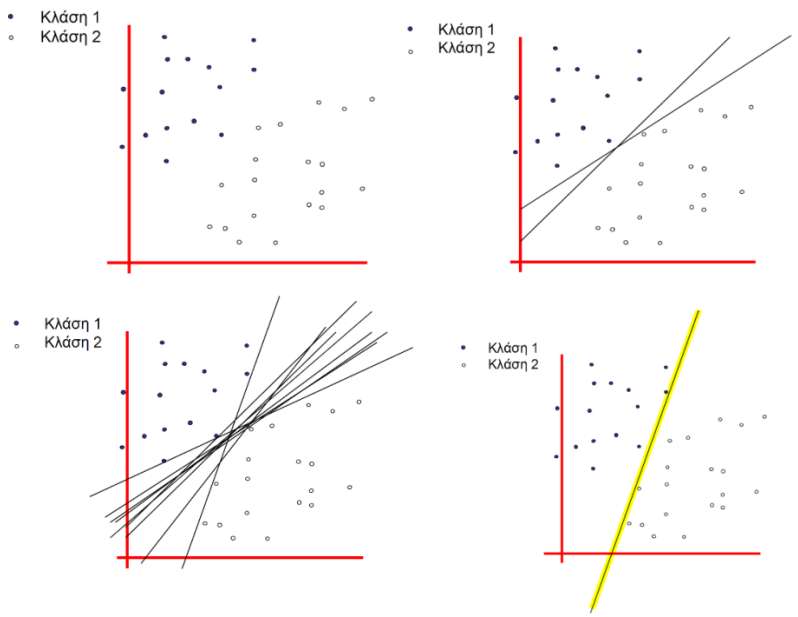
Ένα Support Vector Machine (SVM), είναι ένας αλγόριθμος που χρησιμοποιείται για προβλήματα κατηγοριοποίησης και παλινδρόμησης, έχοντας ως βάση τη θεωρία στατιστικής μάθησης (statistical learning theory). Μια πρώτη προσέγγιση των SMVs έγινε από τον Vladimir Vapnik το 1963, ο οποίος τα πρότεινε ως μια νέα μορφή γραμμικών ταξινομητών. Αργότερα όμως, υπήρξαν αξιόλογες επεκτάσεις και για επίλυση προβλημάτων παρεμβολής (Support Vector Regression).

Η βασική ιδέα πίσω από τη λειτουργία του αλγορίθμου, είναι ο καθορισμός του βέλτιστου σημείου διαχωρισμού (ή σημείου απόφασης) των δεδομένων. Συνεπώς, ένα SMV θα βρει τη νοητή γραμμή διαχωρισμού των παραδειγμάτων, βασιζόμενο στην ετικέτα (label) του κάθε παραδείγματος.

Η γραμμή διαχωρισμού που θα επιλεγεί, θα πρέπει να μεγιστοποιεί την απόσταση από το Support Vector, το οποίο είναι το σημείο των δεδομένων με τη μικρότερη απόσταση από τη γραμμή διαχωρισμού. Εδώ αξίζει να αναφέρουμε πως ανάλογα με τις διαστάσεις του προβλήματος, το διαχωριστικό σύνορο λαμβάνει και διαφορετική ονομασία:

- Γραμμή → προβλήματα δύο διαστάσεων
- Επίπεδο → προβλήματα τριών διαστάσεων
- Υπέρ-επίπεδο → προβλήματα μεγαλύτερα των τριών διαστάσεων

Η μεγαλύτερη πρόκληση του αλγορίθμου είναι η κατηγοριοποίηση των κοντινότερων, στο διαχωριστικό σύνορο, σημείων δεδομένων (support vectors), καθώς λόγω της απόστασής τους η κατηγοριοποίησής τους είναι πιο αβέβαια. Όμως, θεωρούνται τα πιο σημαντικά σημεία του συνόλου δεδομένων, καθώς έστω και ένα από αυτά εάν αλλάξει θέση θα πρέπει να δημιουργηθεί εκ νέου η γραμμή διαχωρισμού. Συνεπώς, ένα SVM δημιουργεί το βέλτιστο για τα δεδομένα σύνορο, ενώ παράλληλα προσπαθεί να μεγιστοποιήσει το περιθώριο (margin) που δεν είναι άλλο από την απόσταση μεταξύ της γραμμής διαχωρισμού και του Support Vector.



Εικόνα 2.6 Εύρεση του καλύτερου συνόρου διαχωρισμού

Τα SMVs θεωρούνται μία από τις καλύτερες μεθόδους επίλυσης διάφορων προβλημάτων Μηχανικής Μάθησης παρουσιάζοντας state of the art αποτελέσματα σε προβλήματα όπως:

- Ταξινόμηση κειμένου
- Ταξινόμηση εικόνων
- Αναγνώριση χαρακτήρων
- Αναγνώριση Βιολογικών Αλληλουχιών

Επιπλέον, λόγω της πλούσιας θεωρητικής θεμελίωσης που διαθέτουν, μπορούν να επιτύχουν την εύρεση του απόλυτου ελαχίστου σε μια διαδικασία βελτιστοποίησης, αφήνοντας πίσω ακόμα και μεθόδους που χρησιμοποιούν Νευρωνικά Δίκτυα τα οποία θεωρούν ως βέλτιστο κάποιο τοπικό ελάχιστο. Εδώ αξίζει να αναφέρουμε πως το υπολογιστικό κόστος των SVMs είναι ασύγκριτα χαμηλότερο αυτού των Νευρωνικών Δικτύων (Βλαχάβας Ιωάννης, *2020)* .

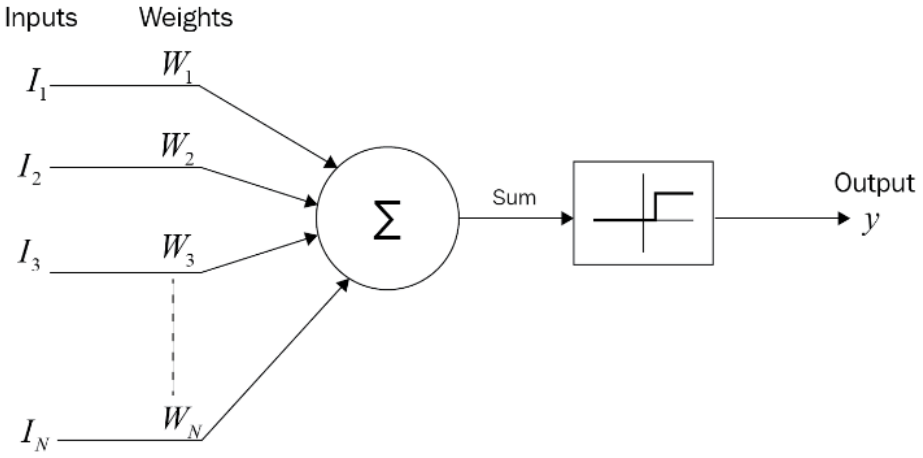
Εκτός αυτού, ένα βασικό χαρακτηριστικό τους είναι η εξαιρετική τους απόδοση σε ιδιαίτερα πολύπλοκα δεδομένα, όπως για παράδειγμα εφαρμογές επεξεργασίας κειμένου, αλλά και σε περιπτώσεις δύσκολων συνόλων δεδομένων όπου για παράδειγμα περιέχουν πολλές στήλες και σχετικά λίγες γραμμές.

Τέλος να αναφέρουμε πως αξιοσημείωτη είναι η ανθεκτικότητά τους σε περιπτώσεις υπερπροσαρμογής των δεδομένων(over-fitting).

2.2.4 Νευρωνικά Δίκτυα (Neural Networks, NN)

Τα Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ, Artificial Neural Networks, ANN) είναι μοντέλα Μηχανικής Μάθησης εμπνευσμένα από την λειτουργία του ανθρώπινου εγκεφάλου. Βασίζονται στους βιολογικούς νευρώνες και χρησιμοποιούν δομές και διαδικασίες που μιμούνται την λειτουργία του νευρώνων του ανθρώπινου εγκεφάλου. Η πιο αξιόλογη ιστορική αναφορά είναι η δημιουργία του Perceptron από τον Rosenblatt το 1957, ενός μοναδικού τεχνητού νευρώνα ο οποίος όμως δεν μπορούσε να επιλύσει μη γραμμικώς

διαχωρίσιμα προβλήματα. Παρά ταύτα, οι πρώτες αναφορές για ΤΝΔ καταγράφονται το 1943 σε μια προσπάθεια εξήγησης της μνήμης από τους McCulloch και Pitts, οι οποίοι πρότειναν και το πρώτο υπολογιστικό μοντέλο νευρώνα. Η λειτουργία του χωρίζεται σε δύο διακριτές διαδικασίες. Η πρώτη λειτουργία αποτελείται από τον αθροιστή, οποίος προσθέτει τις επηρεασμένες από τα βάρη τιμές εισόδου και η δεύτερη λειτουργία εξάγει μια συγκεκριμένη τιμή, σε σύγκριση με τη συνάρτηση ενεργοποίησης.



Διάγραμμα 2.1: Ο νευρώνας των McCulloch – Pitts

Ένα Νευρωνικό Δίκτυο αποτελείται από πολλά επίπεδα νευρώνων. Σε γενικές γραμμές, αποτελείται από το επίπεδο εισόδου (input layer), τα κρυφά ενδιάμεσα επίπεδα νευρώνων (hidden layers) και το επίπεδο εξόδου (output layer). Αξίζει να αναφέρουμε πως τα τελευταία χρόνια έχουν αναπτυχθεί Νευρωνικά Δίκτυα τα οποία περιέχουν πολύ μεγάλο αριθμό κρυφών επιπέδων νευρώνων, τα οποία στη σύγχρονη βιβλιογραφία είναι γνωστά ως *Βαθιά Νευρωνικά Δίκτυα (Deep Neural Networks)* και αποτελούν τη βάση ενός πολλά υποσχόμενου κλάδου της Μηχανικής Μάθησης, την λεγόμενη *Βαθιά Μάθηση (Deep Learning).* Κάθε επίπεδο περιέχει ένα μεγάλο αριθμό νευρώνων. Κάθε νευρώνας του επιπέδου λαμβάνει κάποια τιμή ως είσοδο και παράγει με τη σειρά του μια τιμή εξόδου. Η τιμή εισόδου είναι η τιμή εξόδου του προηγούμενου επιπέδου, εκτός των νευρώνων του πρώτου επιπέδου όπου η είσοδος τους είναι μια από τις ανεξάρτητες μεταβλητές του προβλήματος που αντιμετωπίζουμε. Σαν έξοδο, η τιμή μπορεί να είναι συνεχής, δυαδική ή κατηγορική, συμβαδίζοντας πάντα με τη φύση του προβλήματος.

Ένα από τα βασικότερα συστατικά της δομής ενός Νευρωνικού Δικτύου είναι οι Συναρτήσεις Ενεργοποίησης (Activation Functions). Μια Συνάρτηση Ενεργοποίησης χρησιμοποιείται για τον έλεγχο της τιμής εξόδου που παράγεται από έναν νευρώνα. Με αυτό τον τρόπο, λαμβάνεται η απόφαση για το εάν ο νευρώνας αυτός μπορεί να θεωρηθεί ενεργός ή όχι, με βάση πάλι τη φύση του προβλήματος. Η τιμή εξόδου της Συνάρτησης Ενεργοποίησης ανήκει στο σύνολο των πραγματικών αριθμών και προέρχεται από τον εξής τύπο:

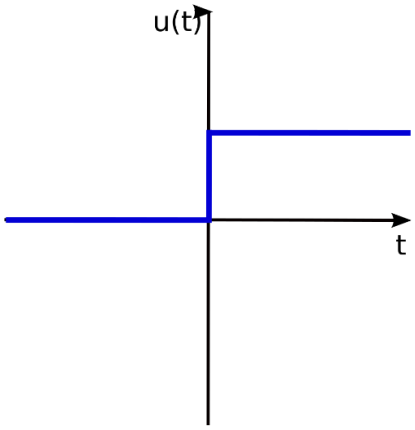
$$y = \Sigma(weight * input) + bias$$

Οι τύποι συναρτήσεων ενεργοποίησης είναι αρκετοί, ωστόσο οι πιο σημαντικοί και ευρέως χρησιμοποιούμενοι είναι οι εξής:

- Βηματική Συνάρτηση (Step Function)
- Σιγμοειδής Συνάρτηση (Sigmoid Function)
- Συνάρτηση tanh
- ΣυνάρτησηReLU (Rectified Linear Unit)

Η Βηματική Συνάρτηση χρησιμοποιείται κατά βάση για προβλήματα δυαδικής ταξινόμησης και η μορφή της είναι η εξής:

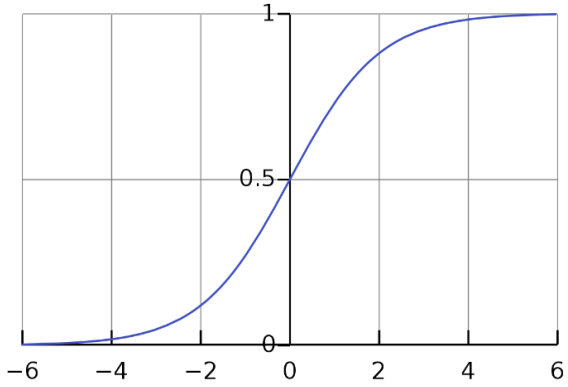
$$f(x) = \begin{cases} 1 & , \text{x>n} \\ 0 & , \text{otherwise} \end{cases}$$



Διάγραμμα 2.2: Η Βηματική Συνάρτηση Ενεργοποίησης

Η Σιγμοειδής Συνάρτηση σε γενικές λειτουργεί ως φίλτρο καταστέλλοντας τις πολύ μεγάλες τιμές. Σαν συνάρτηση είναι μη γραμμική και έχει την δυνατότητα να χρησιμοποιηθεί σε Νευρωνικά Δίκτυα πολλαπλών επιπέδων. Ο τύπος της είναι ο εξής:

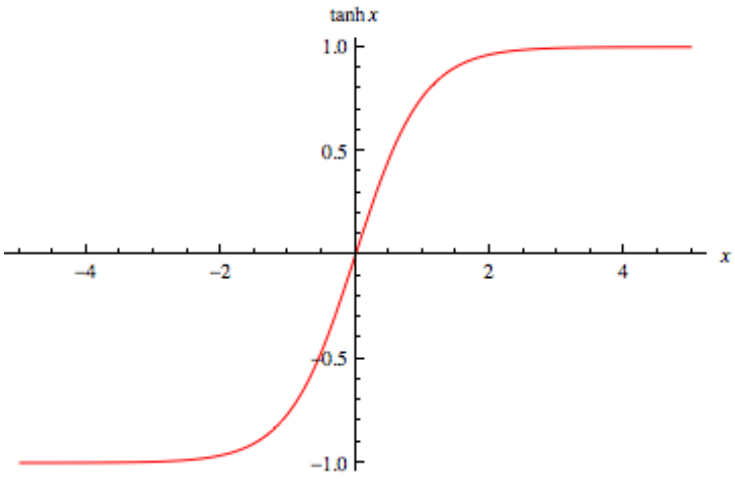
$$f(x) = \frac{1}{1+\exp^{-x}}$$



Διάγραμμα 2.3: Η Σιγμοειδής Συνάρτηση Ενεργοποίησης

Η συνάρτηση Tanh(x) ανήκει στις σιγμοειδής συναρτήσεις και ο τύπος της είναι ο εξής

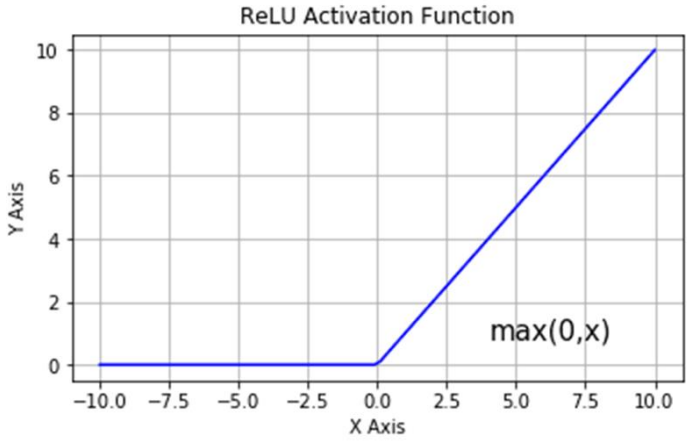
$$\tanh(x) = \frac{2}{1+\exp^{-2x}} - 1$$



Διάγραμμα 2.4: Η Συνάρτησης ενεργοποίησης Tanh

Τέλος, έχουμε την συνάρτηση ReLU (Rectified Linear Unit). Το βασικό αρνητικό της συνάρτησης ReLU είναι πως το εύρος τιμών της εξόδου της μπορεί να λάβει τιμές από μηδέν έως το άπειρο, καθιστώντας πολλές φορές την ενεργοποίηση του αντίστοιχου νευρώνα αδύνατη. Αξίζει όμως να αναφέρουμε πως στα Βαθιά Νευρωνικά Δίκτυα, η συνάρτηση ReLU είναι ιδιαίτερα δημοφιλής καθώς οι απαιτήσεις της σε μαθηματική επεξεργασία είναι αρκετά απλούστερες σε σχέση με τις Σιγμοειδείς συναρτήσεις και ως εκ τούτου, έχουν χαμηλότερο υπολογιστικό κόστος. Ο τύπος της συνάρτησης ReLU είναι ο εξής:

$$f(x) = max(0, x)$$



Διάγραμμα 2.5: Η Συνάρτηση Ενεργοποίησης ReLU

Επιπλέον, αξίζει να αναφέρουμε πως μπορούμε να διακρίνουμε αρκετούς τύπους Τεχνητών Νευρωνικών Δικτύων. Με κριτήριο διαχωρισμού την συνδεσιμότητα τους, έχουμε δύο κατηγορίες:

- Πλήρως Συνδεδεμένα ΝΔ: όπου κάθε νευρώνας ενός επιπέδου συνδέεται με κάθε ένα ξεχωριστά νευρώνα του επόμενου επιπέδου
- Μερικώς Συνδεδεμένα ΝΔ: όπου υπάρχουν κάποιοι νευρώνες ενός επιπέδου που δεν συνδέονται με κάποιους νευρώνες του επόμενου επιπέδου.

Ένα ακόμη σημαντικό κριτήριο διαχωρισμού είναι η κατεύθυνση της ροής της πληροφορίας εντός του δικτύου, όπου πάλι διακρίνουμε δύο κατηγορίες:

- Αναδρομικά ΝΔ (Recurrent Neural Networks, RNNs): όπου οι νευρώνες ενός επιπέδου συνδέονται και με νευρώνες προηγούμενου ή του ίδιου επιπέδου (ανατροφοδότηση). Με αυτόν τον τρόπο, επιτρέπεται η ανατροφοδότηση της πληροφορίας εντός του δικτύου και η επανεπεξεργασία των εξόδων μεταγενέστερων επιπέδων, παρουσιάζοντας εξαιρετικά αποτελέσματα σε εφαρμογές Βαθιάς Μάθησης
- Μη αναδρομικά ΝΔ (Feedforward Neural Networks): όπου οι νευρώνες κάθε επιπέδου συνδέονται αποκλειστικά και μόνο, με νευρώνες επόμενων επιπέδων

2.3 Μάθηση χωρίς Επίβλεψη (Unsupervised Learning)

Στην μάθηση χωρίς επίβλεψη, το σύστημα καλείται να αντιληφθεί μοτίβα μέσω παρατηρήσεων μη κατηγοριοποιημένων δεδομένων του εξωτερικού του περιβάλλοντος, χωρίς να έχει γνώση των επιθυμητών αποτελεσμάτων. Η μάθηση χωρίς επίβλεψη χρησιμοποιείται κατά βάση για τρεις διακριτές διεργασίες που είναι η Συσταδοποίηση (clustering), Κανόνες Συσχέτισης (Association Rules) και η Μείωση της διαστασιμότητας των δεδομένων (Dimensionality Reduction)

2.3.1 Ομαδοποίηση

Η Ομαδοποίηση, είναι η προσπάθεια δημιουργίας ομάδων ή αλλιώς συστάδων (clusters) βάση των κοινών χαρακτηριστικών τους. Ορισμένα παραδείγματα είναι η ομαδοποίηση εικόνων με βάση το εικονιζόμενο περιεχόμενο, η ομαδοποίηση των Tweets στο Twitter με βάση το περιεχόμενο του κάθε Tweet. Ένας από τους πιο κλασικούς αλγορίθμους μάθησης χωρίς επίβλεψη είναι η ομαδοποίηση Κ-μέσων (K -means Clustering).

Συσταδοποίηση Κ – Μέσων (K – Means Clustering)

Ο αλγόριθμος συσταδοποίησης K – Means είναι ένας αλγόριθμος Μη Επιβλεπόμενης Μάθησης, ο οποίος χρησιμοποιείται για τη συσταδοποίηση δεδομένων που δεν ανήκουν σε κάποια κατηγορία ή ομάδα. Ο αλγόριθμος στοχεύει στη δημιουργία συστάδων (ομάδων) από δεδομένα, το πλήθος των οποίων αναπαρίσταται από τον αριθμό K. Η λειτουργία του αλγορίθμου είναι επαναληπτική, καθώς σε κάθε επανάληψη τοποθετεί κάθε στοιχείο των

δεδομένων σε μία από τις K συστάδες δεδομένων, βασιζόμενος στις μεταξύ τους αποστάσεις. Ο αριθμός K συνήθως επιλέγεται τυχαία.

Κάθε συστάδα σχετίζεται με ένα κεντρικό σημείο το οποίο ονομάζεται centroid. Ο τρόπος που προκύπτει το centroid είναι ο εξής: Έστω ότι έχουμε σε μια συστάδα ορισμένα δεδομένα τα οποία αναπαρίστανται με τους αριθμούς 2, 8, 4, 6. Τότε το centroid θα προκύψει από την εξής πράξη:

$$(2 + 8 + 4 + 6) / 4 = 5$$

Συνεπώς το centroid είναι ο μέσος όρος των τιμών των δεδομένων μιας συστάδας. Έτσι λοιπόν, κάθε σημείο από τα 2, 8, 4, 6 θα σχετίζεται με το centroid 5 και κατ' επέκταση με τη συστάδα που αντιπροσωπεύει τον συγκεκριμένο centroid.

Γενικώς, το πλήθος των centroids είναι ίδιο με το πλήθος των συστάδων που θέλουμε να δημιουργήσουμε καθώς κάθε centroid αντιπροσωπεύει μια συστάδα. Συνεπώς, ο αριθμός K εκτός από τον αριθμό των συστάδων, υποδηλώνει και το πλήθος των centroids που θέλουμε να έχουμε.

Τα βήματα του αλγορίθμου είναι τα εξής:

1. Επέλεξε K αρχικά centroids (τυχαία)
2. Επανάλαβε
    - Συσχέτισε όλα τα σημεία στο κοντινότερό τους centroid και αντίστοιχα στην συστάδα που αντιπροσωπεύει
    - Υπολόγισε ξανά τα centroids της κάθε συστάδας
3. Μέχρι τα centroids να παραμένουν αμετάβλητα

Η συσταδοποίηση με τον αλγόριθμο K - means είναι αρκετά απλή και εύκολη στην υλοποίηση. Ένα ευαίσθητο σημείο, όμως, του αλγορίθμου είναι η επιλογή των αρχικών centroids. Γενικώς, διαφορετικές αρχικές θέσεις για τα centroids δίνουν πάντα διαφορετικά αποτελέσματα συσταδοποίησης. Συνήθως θεωρείται πιο σωστό να επιλέγονται centroids που απέχουν όσο γίνεται περισσότερο μεταξύ τους καθώς η συσταδοποίηση σε αυτήν την περίπτωση παρουσιάζει καλύτερα και πιο έγκυρα αποτελέσματα τις περισσότερες φορές.

Πιθανοκρατική Συσταδοποίηση (Probabilistic Clustering)

Ένα πιθανοκρατικό μοντέλο μα βοηθά να επιλύσουμε προβλήματα συσταδοποίησης με βάση την εκτίμηση της πυκνότητας των δεδομένων (Density estimation clustering). Σε μια Πιθανοκρατική Συσταδοποίηση, κάθε παράδειγμα του συνόλου των δεδομένων κατηγοριοποιείται με βάση την μέγιστη πιθανοφάνεια να ανήκει σε μια συγκεκριμένη κατανομή. Μια από τις πιο ονομαστές και συνήθεις μεθόδους πιθανοκρατικής συσταδοποίησης είναι το μοντέλο Gaussian Mixture Model (GMM).

2.3.2 Κανόνες Συσχέτισης

Ένας Κανόνας Συσχέτισης, όπως φανερώνει και το όνομα, είναι μια μέθοδος βασισμένη σε κανόνες για την εύρεση των συσχετίσεων μεταξύ των μεταβλητών ενός συνόλου δεδομένων. Τέτοιες μέθοδοι εφαρμόζονται σε μεγάλη κλίμακα σε διεργασίες «ανάλυσης καλαθιού

αγορών», επιτρέποντας σε εταιρίες να κατανοήσουν καλύτερα τις συσχετίσεις μεταξύ των διάφορων προϊόντων ενός καταστήματος, με απώτερο στόχο να οργανώσουν καλύτερες cross-selling στρατηγικές αλλά και να δημιουργήσουν πιο αποδοτικές μηχανές συστάσεων (recommendation engines). Σε αυτό το σημείο αξίζει να αναφέρουμε πως υπάρχουν αρκετοί αλγόριθμοι που παράγουν κανόνες συσχέτισης, με πιο ονομαστό και ευρέως χρησιμοποιούμενο τον αλγόριθμο Apriori

Αλγόριθμος Apriori

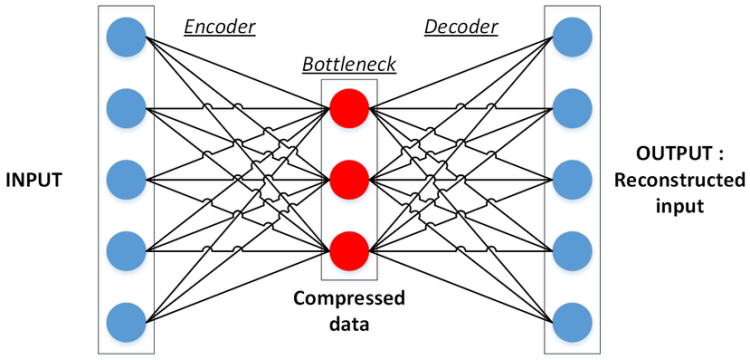
Ο αλγόριθμος Apriori χρησιμοποιείται, κατά κόρον, σε δεδομένα συναλλαγών για την εύρεση συχνών συνόλων αντικειμένων με στόχο την εύρεση της πιθανοφάνειας της αγοράς ενός προϊόντος, δεδομένης της αγοράς ενός άλλου προϊόντος.

2.3.3 Μείωση της διαστασιμότητας των δεδομένων (Dimensionality Reduction)

Σε γενικές γραμμές, όσο περισσότερα δεδομένα έχουμε σε ένα σύνολο δεδομένων, τόσο πιο ακριβές πιθανότατα θα προκύψει και το μοντέλο που χρησιμοποιούμε. Ωστόσο, δεν είναι λίγες οι φορές που ο αριθμός των χαρακτηριστικών ή των διαστάσεων ενός συνόλου δεδομένων είναι αρκετά μεγάλος έχοντας ως αποτέλεσμα τη δημιουργία ενός από τα συνηθέστερα προβλήματα που μπορεί να προκύψουν κατά την διαδικασία της ανάλυσης των δεδομένων, αυτού της υπέρ-προσαρμογής(overfitting) του μοντέλου στα δεδομένα. Οι τεχνικές της μείωσης της διαστασιμότητας των δεδομένων, έχουν ως στόχο να μειώσουν την διάσταση των δεδομένων εισόδου μετατρέποντάς τα σε πιο διαχειρίσιμη μορφή, χωρίς να αλλοιώνεται οι φύση και η σημασία των δεδομένων. Μία από τις πιο ονομαστές τεχνικές είναι οι Autoencoders.

Autoencoders

Οι Autoencoders χρησιμοποιούν τα Νευρωνικά Δίκτυα για να συμπιέσουν δεδομένα και στη συνέχεια να δημιουργήσουν μια νέα αναπαράσταση των δεδομένων εισόδου. Λόγω της χρήσης Νευρωνικών Δικτύων, η λειτουργία του αλγορίθμου χωρίζεται σε τρία επίπεδα: το επίπεδο εισόδου, ένα κρυφό επίπεδο και το επίπεδο εξόδου. Η μετάβαση από το επίπεδο εισόδου στο κρυφό επίπεδο ονομάζεται κωδικοποίηση (encoding), ενώ η μετάβαση από το κρυφό επίπεδο προς το επίπεδο εξόδου ονομάζεται αποκωδικοποίηση (decoding). Τα δεδομένα ανακατασκευάζονται κατά την μετάβαση από το κρυφό επίπεδο προς το επίπεδο εξόδου, αφού επεξεργαστούν στο κρυφό επίπεδο το οποίο λειτουργεί ως συμπιεστής της εισόδου.

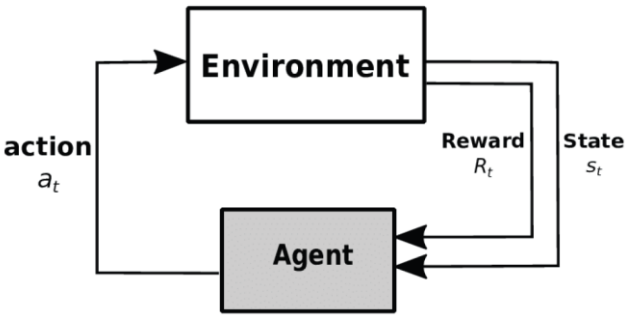


Εικόνα 2.7: Η λειτουργία των Autoencoders

2.4 Μάθηση με Ενίσχυση (Reinforcement Learning)

Στην Ενισχυτική Μάθηση (Reinforcement Learning, RL), ένα πρόγραμμα καλείται να μάθει μια στρατηγική ενεργειών αλληλοεπιδρώντας άμεσα με το εξωτερικό του περιβάλλον βασιζόμενο στη διαρκή ανατροφοδότηση, θετική ή αρνητική. Ενώ στα δύο προηγούμενα είδη το πρόγραμμα βασίζεται στα δεδομένα, στην ενισχυτική μάθηση κάνει δοκιμές, με απώτερο στόχο τα αποτελέσματα που θα προκύψουν να μεγιστοποιούν τη θετική ανταμοιβή από το εξωτερικό του περιβάλλον. Μεγάλη εφαρμογή έχουν οι αλγόριθμοι της ενισχυτικής μάθησης στον χρονοπρογραμματισμό διεργασιών σε βιομηχανικούς χώρους καθώς και σε μάθηση διάφορων παιχνιδιών όπως το σκάκι.

Συνεπώς, ο βασικός στόχος ενός μοντέλου Ενισχυτικής Μάθησης είναι η μεγιστοποίηση της θετικής ανατροφοδότησης και αντίστοιχα η ελαχιστοποίηση της αρνητικής.



Διάγραμμα 2.6: Η λειτουργία της Ενισχυτικής Μάθησης

Agent: Ο πράκτορας του διαγράμματος 2.6 αντιπροσωπεύει το σύστημα λήψης αποφάσεων του μοντέλου, δηλαδή τον ίδιο τον αλγόριθμο. Ένας τέτοιος αλγόριθμος μπορεί να είναι μια σειρά από Νευρωνικά Δίκτυα τα οποία εκτιμούν την κατάσταση του περιβάλλοντος και αναλόγως λαμβάνουν τις αποφάσεις.

Environment: Το περιβάλλον αντιπροσωπεύει το χώρο στον οποίο ο πράκτορας δρα και βρίσκεται και περιέχει μια σειρά από καταστάσεις τις οποίες χρησιμοποιεί ως είσοδο ο πράκτορας.

Actions: Οι αποφάσεις-δράσεις έχουν να κάνουν με όλες τις δυνατές κινήσεις που πιθανόν να κάνει ο πράκτορας.

State: Ο όρος State αντιπροσωπεύει την κατάσταση που βρίσκεται τη δεδομένη στιγμή το περιβάλλον και λαμβάνεται ως είσοδος στον πράκτορα.

Reward: Η συνάρτηση Reward αποτελεί την αξιολόγηση της αμέσως προηγούμενης δράσης και είναι η βασική πηγή ανατροφοδότησης του μοντέλου.

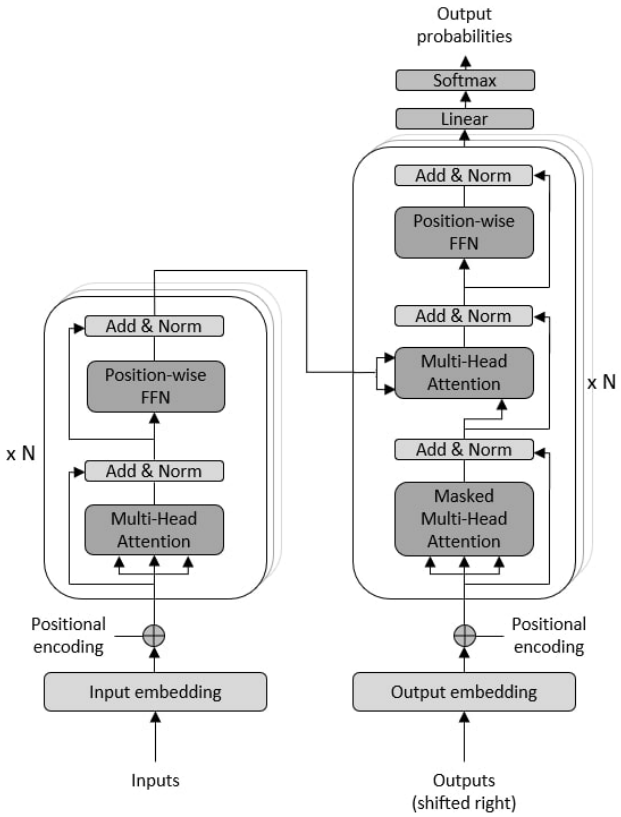
2.5 Ο αλγόριθμος που επιλέχθηκε: Transformers

Η σημαντικότερη και πιο ουσιαστική πρόοδος στην επεξεργασία ακολουθιών και ιδίως στην Επεξεργασία Φυσικής Γλώσσα (Natural Language Processing, NLP) σημειώθηκε με την ανάπτυξη των Transformer. Ιστορικά, οι Vaswani et al. (2017) έκαναν μια πρωτοποριακή δημοσίευση με τίτλο “Attention Is All You Need” παρουσιάζοντας την αρχιτεκτονική των Transformers η οποία και αποτέλεσε έναυσμα για τη δημιουργία μιας ολοκληρωτικά νέας οικογένειας μοντέλων Βαθιάς Μάθησης. Όπως είναι φανερό και από τον τίτλο, επιστρατεύεται η έννοια της «επικέντρωσης της προσοχής» (attention) για αντιμετωπιστούν πολλά από τα μειονεκτήματα των, μέχρι τότε state-of-the-art, Αναδρομικών Νευρωνικών Δικτύων. Η βασική ιδέα πίσω από αυτήν την τεχνική είναι η επικέντρωση την προσοχής σε σχετικά μέρη της εισόδου, αντιγράφοντας την βιολογική λειτουργία του ανθρώπου να επικεντρώνεται σε ότι αυτός θεωρεί πιο σημαντικό.

Η επικέντρωση της προσοχής προτάθηκε από τους Bahdanau et al. (2014) και επιτρέπει στο μοντέλο να επικεντρώνεται σε όλα τα σχετικά κομμάτια της εισόδου ταυτόχρονα. Η ικανότητα της ταυτόχρονης επικέντρωσης της προσοχής σε όλη την είσοδο αποτελεί και την ειδοποιό διαφορά των Transformers με τα Αναδρομικά Νευρωνικά Δίκτυα που εργάζονται σειριακά με τα επί μέρους κομμάτια της εισόδου. Η τεχνική αυτή δημιουργεί μια αναπαράσταση της εισόδου με βάρη έτσι ώστε να ξεχωρίσει τα πιο σημαντικά επιμέρους τμήματα της εισόδου αυξάνοντας την αντίστοιχη τιμή των βαρών τους, χρησιμοποιώντας ερωτήματα (queries), κλειδιά (keys) και τιμές (values) οι οποίες συγκρίνονται σε τελική φάση μεταξύ τους. Εδώ αξίζει να αναφέρουμε πως αυτή η τεχνική χρησιμοποιείται ευρέως από τις μηχανές αναζήτησης. Γενικώς, το εκάστοτε ερώτημα αντιστοιχίζεται με ένα συγκεκριμένο κλειδί, επιστρέφοντας μια μοναδική τιμή, η οποία αντιπροσωπεύει το συγκεκριμένο κλειδί. Η τεχνική της επικέντρωσης στους Transformers είναι παρόμοια, με την βασική διαφορά πως αντί να έχουμε μια απλή αντιστοίχιση, το ερώτημα (query) ταιριάζει με όλα τα κλειδιά σε ένα συγκεκριμένο βαθμό με βάση το εσωτερικό τους γινόμενο. Συνεπώς, το εσωτερικό γινόμενο εδώ αναλαμβάνει το ρόλο μετρικής ομοιότητας (similarity score) μεταξύ των ερωτημάτων και των κλειδιών. Τα similarity scores στη συνέχεια χρησιμοποιούνται έτσι ώστε να δοθούν νέα βάρη στην επικέντρωση της προσοχής στις αντίστοιχες τιμές ούτως ώστε να παραχθεί μία νέα αναπαράσταση της ακολουθίας εισόδου.

Παρατηρώντας το αρχικό μοντέλο των Transformers που παρουσίασαν οι Vaswani et. Al. (2017) στο διάγραμμα 2.7 που ακολουθεί, μπορούμε να πούμε πως αρχιτεκτονική

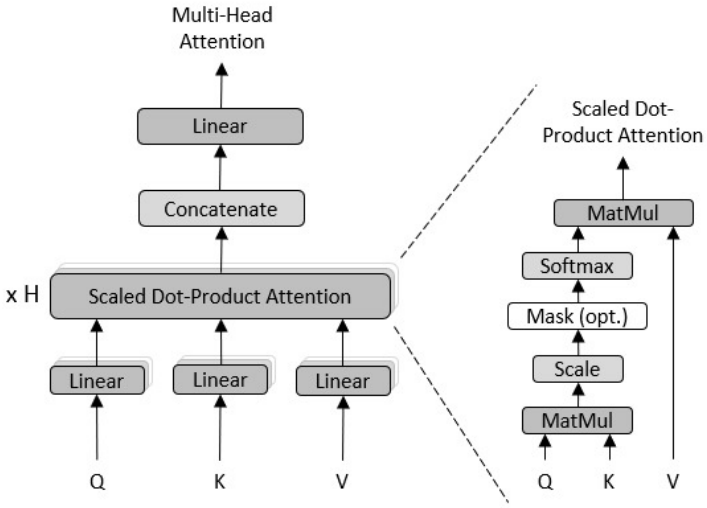
Κωδικοποιητή (Encoder) και Αποκωδικοποιητή (Decoder) που χρησιμοποιεί, είναι μια συνήθως χρησιμοποιούμενη αρχιτεκτονική σε εφαρμογές NLP όπου οι ακολουθίες εισόδου και εξόδου είναι διαφορετικών μεγεθών αλλά και γλωσσών.



Διάγραμμα 2.7: Το μοντέλο Transformer (Vaswani et al. 2017)

Ο κωδικοποιητής (στα αριστερά του διαγράμματος) παρέχει μια κωδικοποιημένη αναπαράσταση της ακολουθίας εισόδου, την οποία χρησιμοποιεί ο αποκωδικοποιητής (δεξιά) για να παραχθεί η επιθυμητή έξοδος. Η αρχιτεκτονική των περισσότερων από τα υπό-επίπεδα του συγκεκριμένου μοντέλου είναι παρόμοια με αυτήν του κωδικοποιητή – αποκωδικοποιητή. Ο κωδικοποιητής γενικώς περιέχει δύο βασικά υπό-επίπεδα, ένα multi-head self-attention επίπεδο και ένα απλό position-wise πλήρως συνδεδεμένο Feed Forward Network (FFN). Η λειτουργία του αποκωδικοποιητή είναι παρόμοια, με την βασική διαφορά πως το multi-head attention υπό-επίπεδο του αποκωδικοποιητή χρησιμοποιεί τιμές από τον κωδικοποιητή αλλά και από τον αποκωδικοποιητή. Εφόσον λοιπόν η βασική λειτουργία και των δύο μερών είναι παρόμοια, στη συνέχεια θα αναλυθεί μόνον η λειτουργία του κωδικοποιητή.

Το θεμελιώδες κομμάτι της αρχιτεκτονικής των Transformers είναι το multi-head self-attention υπό-επίπεδο, του οποίου την λειτουργία μπορούμε να την δούμε στο διάγραμμα 2.8.



Διάγραμμα 2.8: Το multi-head self-attention υπό-επίπεδο

Το μοντέλο Transformer χρησιμοποιεί την επικέντρωση της προσοχής με βάση το εσωτερικό γινόμενο με έναν επιπλέον βαθμό κλιμάκωσης που είναι $\sqrt{d_k}$. Η επικέντρωση της προσοχή με τη χρήση βαθμοτής κλιμάκωσης εσωτερικού γινομένου, αναπαρίσταται στο δεξί μέρος του διαγράμματος 2.8 και μπορεί να γραφεί ως εξής:

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}'}{\sqrt{d_k}}\right)\mathbf{V}$$

όπου d_k είναι η κρυμμένη διάσταση των κλειδιών (keys) και $Q \in \mathbb{R}^{T\times dk}$, $K \in \mathbb{R}^{T\times dk}$ και $V \in \mathbb{R}^{T\times dv}$ είναι οι πίνακες που αντιπροσωπεύουν τα ερωτήματα (Queries) , τα κλειδιά (Keys) και τις τιμές (Values) αντίστοιχα. Οι διαστάσεις των τιμών d_v ενδεχομένως να διαφέρουν από τις διαστάσεις των queries και keys. Επιπλέον, η softmax χρησιμοποιείται για να μετατρέψει κάθε γραμμή του similarity matrix $QK' \in \mathbb{R}^{T\times T}$ σε πιθανότητες. Καθώς η διακύμανση των εσωτερικών γινομένων του similarity matrix είναι ανάλογη της αύξησης της συνάρτησης του d_k, οι τιμές υφίστανται μια κλιμάκωση από την τετραγωνική ρίζα του d_k έτσι ώστε να διασφαλιστεί η μη εμφάνιση του φαινομένου ”vanishing gradient” το οποίο συγκαταλέγεται στα αρνητικά της χρήσης των RNNs. Επιπλέον, οι τιμές οι οποίες χρησιμοποιούνται για να σταθμίσουν το διάνυσμα των τιμών (Value vector), είναι γνωστές ως τιμές προσοχής (Attention Scores) οι οποίες μπορούν να αναπαρασταθούν σε ένα πίνακα $T \times T$ για να φανεί κάθε σύνδεση μεταξύ των τιμών της ακολουθίας εισόδου με μήκος *T.*

Σύμφωνα με τους Zhang et al. (2021), τρεις διαφορετικοί πίνακες βαρών (weights) χρησιμοποιούνται για τρεις διαφορετικούς μετασχηματισμούς της ακολουθία εισόδου. Οι τρεις πίνακες είναι οι εξής:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q$$
$$\mathbf{K} = \mathbf{X}\mathbf{W}^K$$
$$\mathbf{V} = \mathbf{X}\mathbf{W}^V,$$

όπου:

$$\mathbf{W}^Q \in \mathbb{R}^{d_{model} \times d_k}, \mathbf{W}^K \in \mathbb{R}^{d_{model} \times d_k}, \mathbf{W}^V \in \mathbb{R}^{d_{model} \times d_v}$$

Όλα αυτά τα βάρη βελτιστοποιούνται κατά τη διαδικασία του υπολογισμού του μοντέλου έτσι ώστε να παραχθεί η βέλτιστη σταθμισμένη επικέντρωση προσοχής (weighting attention).

Η λειτουργία του multi-head self-attention υπό-επιπέδου, είναι επί της ουσίας μια επέκταση ενός απλού self-attention υπολογισμού. Η βασική ιδέα είναι η χρήση πολλαπλών σημείων επικέντρωσης της προσοχής, έτσι ώστε το μοντέλο να μάθει διαφόρων ειδών συσχετίσεων από την ακολουθία εισόδου. Κάνοντας χρήση μαθηματικών, η επικέντρωση της προσοχής πολλαπλών σημείων μπορεί να διατυπωθεί ως εξής:

$$\text{MultiHead}(Q, K, V) = \text{concat}(head_1, ..., head_h)\mathbf{W}^O,$$

όπου:

$$head_i = \text{attention}(Q, K, V)$$

Έστω πως ο αριθμός των σημείων προσοχής είναι h, τότε οι έξοδοι των σημείων προσοχής $\text{head}_i \in \mathbb{R}^{\text{T} \times dv}$ αρχικά ενώνονται και στη συνέχεια για να αποκτήσουν τον αρχικό μετασχηματισμό της εισόδου πολλαπλασιάζονται με:

$$\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{model}}$$

Το επόμενο υπό-επίπεδο του κωδικοποιητή, είναι το Position-wise feed-forward επίπεδο. Αυτό πρακτικά σημαίνει πως ένα πλήρως συνδεδεμένο FNN εφαρμόζεται με τον ίδιο τρόπο σε κάθε μία θέση ξεχωριστά. Αυτό επιτυγχάνεται χρησιμοποιώντας μια συνάρτηση ενεργοποίησης τύπου ReLU μεταξύ των δύο γραμμικών μετασχηματισμών ως εξής:

$$\text{FFN}(x) = \text{ReLU}(x\mathbf{W}_1)\mathbf{W}_2$$

όπου:

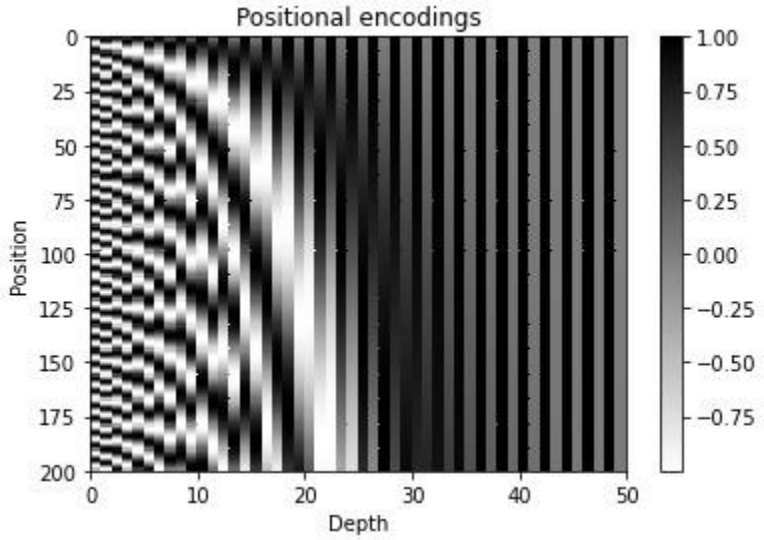
$$\mathbf{W}_1 \in \mathbb{R}^{d_{model} \times 4d_{model}}, \mathbf{W}_2 \in \mathbb{R}^{4d_{model} \times d_{model}}$$

Αξίζει σε αυτό το σημείο να πούμε πως το μόνο μειονέκτημα αυτής της αρχιτεκτονικής είναι πως δεν χρησιμοποιεί την σειρά με την οποία έρχεται η είσοδος καθώς δεν υπάρχουν αναδρομικά επίπεδα. Σύμφωνα με τον Chollet (2021), η είσοδος αντιμετωπίζεται ως ένα ενιαίο σύνολο. Εφόσον όμως τις περισσότερες φορές η σειρά της εισόδου παίζει πολύ σημαντικό ρόλο, πρέπει με κάποιον τρόπο οι πληροφορίες των θέσεων των δεδομένων εισόδου να κωδικοποιηθούν και αυτές. Για αυτό το λόγο, οι Vaswani et al. (2017) πειραματίστηκαν χρησιμοποιώντας διάφορες μεθόδους για να κωδικοποιήσουν τις πληροφορίες των θέσεων των δεδομένων εισόδου καταλήγοντας σε μια μέθοδο σταθερής ημιτονοειδούς καταγραφής της σειράς των θέσεων,

χρησιμοποιώντας διάφορες συχνότητες των συναρτήσεων του ημιτόνου και του συνημιτόνου ως εξής:

$$\mathrm{PE}_{pos,2i} = \sin(pos/10000^{2i/d_{model}})$$
$$\mathrm{PE}_{pos,2i+1} = \cos(pos/10000^{2i/d_{model}})$$

όπου pos είναι η θέση της ακολουθίας εισόδου και i η διάσταση του μοντέλου. Στη συνέχεια οι κωδικοποιήσεις των θέσεων προστίθενται στην είσοδο, $\bar{X} = X + PE(X)$. Μια απεικόνιση των κωδικοποιητών θέσεων μπορούμε να δούμε στο διάγραμμα 2.9 όπου το μήκος της ακολουθίας είναι 200 και η διάσταση του μοντέλου 50.



Διάγραμμα 2.9: Οι κωδικοποιητές θέσεων

Ένα από τα πιο προφανή μειονεκτήματα σχεδόν όλων των μοντέλων βαθιάς μάθησης, είναι η έλλειψη ερμηνευσιμότητας. Και ενώ έχουν αναπτυχθεί αρκετές μέθοδοι οι οποίες είναι model-agnostic (LIME και SHARP), σύμφωνα με τους Lim et al. (2021) δεν είναι κατάλληλες για προβλήματα ανάλυσης χρονοσειρών, διότι, πολύ απλά, δεν έχουν την δυνατότητα να υπολογίσουν χρονικές σειρές. Και εδώ να αναφέρουμε πως, σε αντίθεση με τις τυπικές αρχιτεκτονικές των RNN, όπως τα LSTMs ή τα GRUs, η αρχιτεκτονική των Transformers ενδεχομένως να διατηρεί και πληροφορίες για τη σημασία που δύνανται να έχουν διάφορα timesteps για ένα συγκεκριμένο πρόβλημα.

Chapter 3

Αγορά Χρήματος με μεθόδους Τεχνικής Ανάλυσης

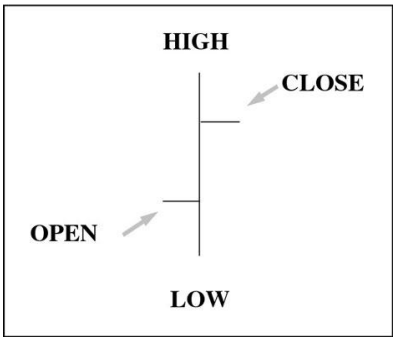
Η Τεχνική Ανάλυση είναι μια μορφή μεθοδικής μελέτης και ανάλυσης οποιουδήποτε χρεογράφου που εμπίπτει σε διαπραγμάτευση όπως μετοχές, ομόλογα, χρεόγραφα ή άλλα προϊόντα χρηματαγοράς. Ένα από τα κύρια πλεονεκτήματα της Τεχνικής Ανάλυσης είναι η δυνατότητα ανάλυσης βασιζόμενοι αποκλειστικά στην μελέτη των διαγραμμάτων των τιμών των μετοχών χωρίς να χρειάζεται να έχουμε την παραμικρή περαιτέρω πληροφορία για την εταιρία στην οποία αντιστοιχεί η εκάστοτε μετοχή.

Η Τεχνικής Ανάλυση βασίζεται στη φιλοσοφία πως έχοντας γνώση των παρελθοντικών τιμών και εφαρμόζοντας συγκεκριμένες και μεθοδικές τακτικές πάνω σε αυτές τις τιμές, μπορούμε να προβλέψουμε την κίνηση (ανοδική ή καθοδική) της εκάστοτε μετοχής στο άμεσο μέλλον. Σε αυτό το σημείο όμως θα πρέπει να σημειώσουμε πως η πρόβλεψη δεν μπορεί να γίνει με απόλυτη ακρίβεια καθώς ο επενδυτής απλώς αποκτά μια σειρά από αρκετά αξιόπιστα σήματα αγοράς και πώλησης. Κάνοντας χρήση αυτών των σημάτων αυξάνονται σημαντικά οι πιθανότητες αριστοποίησης του χαρτοφυλακίου του με συνεπαγόμενη αύξηση των κερδών.

3.1 Ανάλυση Βασιζόμενη σε Διαγράμματα

Η Τεχνική Ανάλυση χωρίζεται σε δύο βασικές κατηγορίες ανάλογα με τον τρόπο ανάλυσης που χρησιμοποιείται. Η πρώτη κατηγορία, που είναι και η πιο απλή σε εφαρμογή, είναι η Ανάλυση Βασιζόμενη σε Διαγράμματα (Διαγραμματική Ανάλυση) όπου βασική μέριμνα των αναλυτών είναι η μελέτη και ανάλυση γραφημάτων τιμών. Η δεύτερη κατηγορία είναι η λεγόμενη Αμιγής Ανάλυση η οποία βασίζεται στους τεχνικούς δείκτες και ταλαντωτές οι οποίοι χαρακτηρίζονται για το σημαντικό μαθηματικό του υπόβαθρο.

Η Ανάλυση Βασιζόμενη σε Διαγράμματα χρησιμοποιείται για την απεικόνιση της πορείας της υποκείμενης αξίας της μετοχής σε ένα συγκεκριμένο χρονικό διάστημα. Τα διαγράμματα που χρησιμοποιούνται είναι διαφόρων μορφών ανάλογα με τον τρόπο απεικόνισης αλλά και του περιεχομένου που απεικονίζουν. Μια από τις πιο συνηθισμένες μορφές διαγραμμάτων είναι τα Bar Chartsτα οποία απεικονίζουν με πολύ ευανάγνωστο τρόπο τη συμπεριφορά της μετοχής μέσα σε μία ενεργώς χρηματιστηριακή ημέρα.



Διάγραμμα 3.1: Η Μορφή των Bar Charts

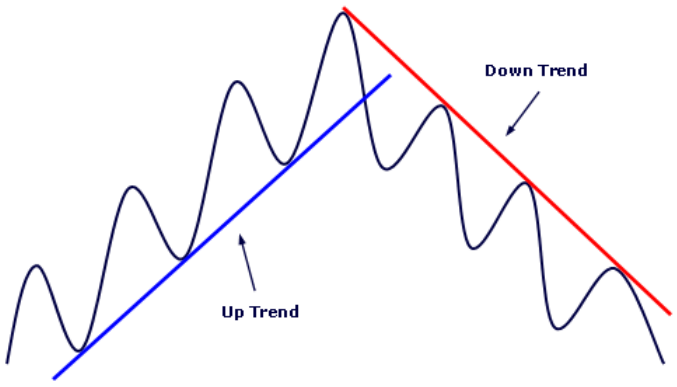
Όπως παρατηρούμε και στο Διάγραμμα 1, το διάγραμμα περιλαμβάνει την ανώτατη και την κατώτατη τιμή (Highκαι Lowαντίστοιχα) που έλαβε η μετοχή κατά τη διάρκεια της ημέρας όπως επίσης και τις τιμές ανοίγματος και κλεισίματος (Openκαι Closeαντίστοιχα).

3.1.1 Γραμμές Τάσης

Οι Γραμμές Τάσης δημιουργούνται από την ένωση των τοπικών ακροτάτων των τιμών των μετοχών. Όσο περισσότερα είναι αυτά τα τοπικά ακρότατα, τόσο πιο ισχυρή και αξιόπιστη θεωρείται η αντίστοιχη Γραμμή Τάσης. Υπάρχουν δύο τύποι Γραμμών Τάσεων

- Ανοδικές Γραμμές Τάσης: Σχηματίζονται από την ένωση των κατωτάτων επιπέδων, που ονομάζονται και πυθμένες, της μετοχής. Μια τέτοια Ανοδική Γραμμή Τάσης φαίνεται στο αριστερό μέρος του Διαγράμματος 2 με χρώμα Μπλε.

- Καθοδικές Γραμμές Τάσης: Σχηματίζονται από την ένωση των ανώτατων επιπέδων, που ονομάζονται και κορυφές, της μετοχής. Μια τέτοια Καθοδική Γραμμή Τάσης φαίνεται στο δεξί μέρος του Διαγράμματος 2 με χρώμα Κόκκινο.



Διάγραμμα 3.2: Ανοδική και Καθοδική Γραμμή Τάσης

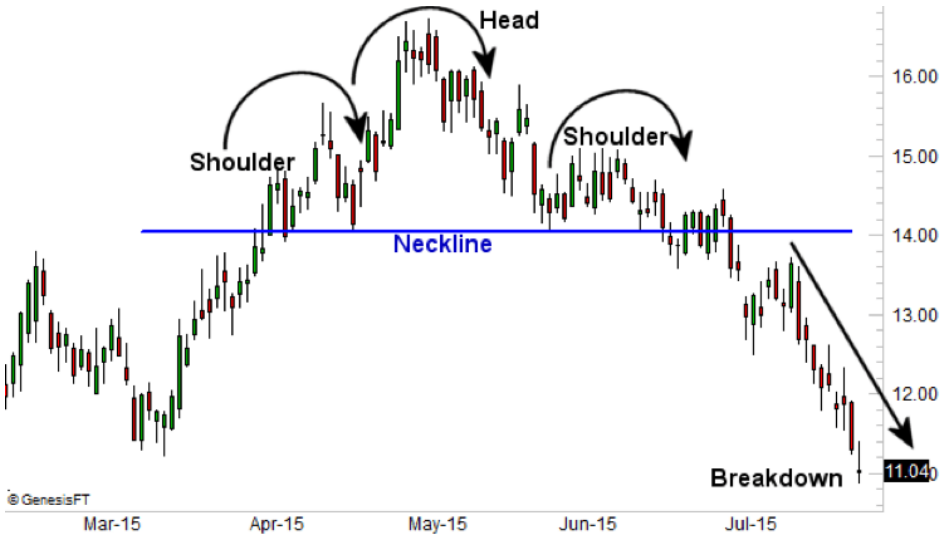
3.1.2 Σχηματισμός Κεφάλι και Ώμοι (Head and Shoulders Pattern)

Το διάγραμμα Κεφάλι και Ώμοι αποτελεί έναν από τους καλύτερους διαγραμματικούς σχεδιασμούς ο οποίος χρησιμοποιείται ευρέως από τους αναλυτές καθώς μπορεί να δώσει

εγκαίρως ενδείξεις σημαντικής ανόδου ή αντίστοιχα καθόδου. Σε μια περίοδο πτώσης της τιμής, η μετοχή ενδέχεται να παρουσιάσει ορισμένα στιγμιαία ξεσπάσματα ανόδου. Σε τέτοιες περιπτώσεις, μπορεί να έχουμε την εμφάνιση του σχηματισμού Κεφάλι και Ώμοι όταν δημιουργούνται τρία στιγμιαία ανοδικά κύματα, εκ των οποίων, το μεσαίο (Κεφάλι) είναι μεγαλύτερο των δύο ακριανών (Ώμοι).

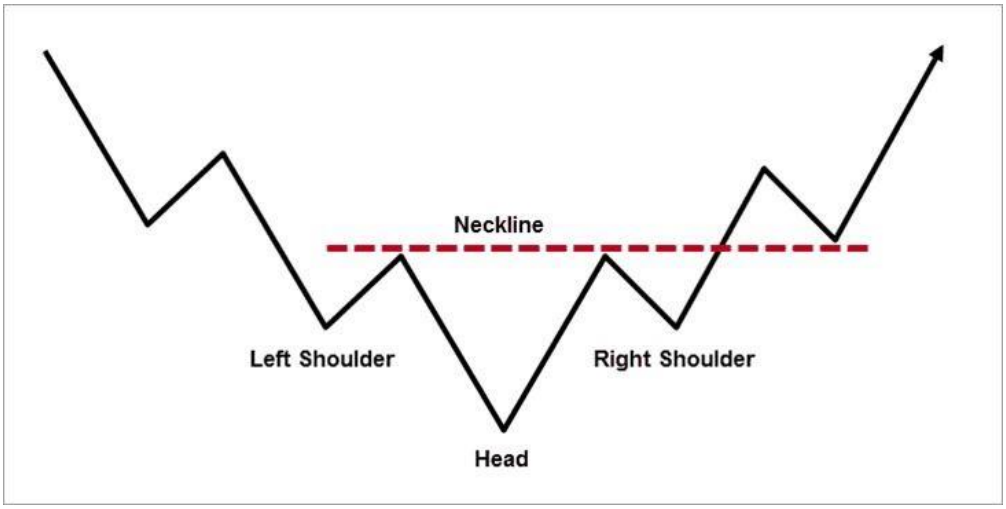
Όπως φαίνεται και στο Διάγραμμα 3.2, η τιμή της μετοχής δημιουργεί τον σχηματισμό Κεφάλι και Ώμοι. Στο τέλους του δεξιού ώμου, έρχεται αντιμέτωπη με την λεγόμενη «γραμμή λαιμού» (Neckline) η οποία δημιουργείται από την ένωση των τοπικών ελαχίστων μεταξύ του αριστερού ώμου, του κεφαλιού και του δεξιού ώμου. Σε αυτό το σημείο θα πρέπει να ελέγξουμε το σημείο τομής μεταξύ της γραμμής λαιμού και της τιμής και πιο συγκεκριμένα την φορά διάσπασης. Έτσι, εάν η τιμή διασπάσει καθοδικά την γραμμή λαιμού, χωρίς στο άμεσο επόμενο διάστημα να την διασπάει ανοδικά, τότε αναμένουμε μια έντονη πτωτική πορεία της μετοχής.

Αντιστρόφως όμως εργαζόμαστε σε περίπτωση εμφάνισης του σχηματισμού «Ανεστραμμένο Κεφάλι και Ώμοι» (Inverse Head and Shoulders)



Διάγραμμα 3.3: Ο Σχηματισμός Κεφάλι και Ώμοι

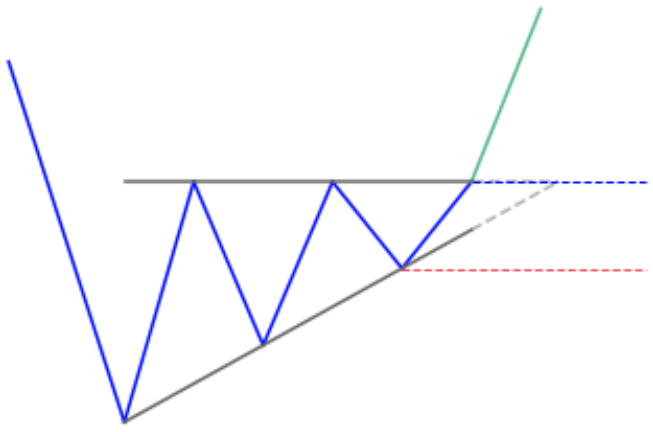
Αντιστρόφως όμως εργαζόμαστε σε περίπτωση εμφάνισης του σχηματισμού «Ανεστραμμένο Κεφάλι και Ώμοι» (Inverse Head and Shoulders). Σε αυτήν την περίπτωση, μετά το πέρας του δεξιού ώμου, εάν διασπάσει η τιμή ανοδικά την γραμμή λαιμού χωρίς στο επόμενο διάστημα να την διασπάει καθοδικά, τότε δίνεται σήμα για ενδεχόμενη σημαντική αύξηση της τιμής της μετοχής.



Διάγραμμα 3.4: Ο Σχηματισμός Ανεστραμμένο Κεφάλι και Ώμοι

3.1.3 Τριγωνικοί Σχηματισμοί (Triangular Patterns)

Οι σχηματισμοί τριγωνικής μορφής είναι μια πολύ απλή μορφή ανάλυσης. Σε αυτούς τους σχηματισμούς, συναντούμε δύο Γραμμές Τάσης οι οποίες συγκλίνουν δημιουργώντας ένα νοητό τρίγωνο. Όπως φαίνεται και στο Διάγραμμα 3.4, οι δύο Γραμμές Τάσης εφάπτονται στα τοπικά ακρότατα που δημιουργούνται στη συνάρτηση της τιμής. Αυτό που μας ενδιαφέρει σε αυτά τα διαγράμματα είναι η φορά της διάσπασης ενός εκ των δύο Γραμμών Τάσεων πριν από το σημείο τομής τους. Πιο συγκεκριμένα, η ανοδική διάσπαση της άνω Γραμμής Τάσης αποτελεί και ένδειξη συνέχισης της ανόδου της τιμής της μετοχής. Σε περίπτωση καθοδικής διάσπασης της κάτω Γραμμής Τάσης έχουμε ένδειξη συνέχισης της πτώσης της τιμής της μετοχής.



Διάγραμμα 3.5 Τριγωνικός Σχηματισμός

3.1.4 Σχηματισμός Κούπα με Χερούλι (Cup and Handle Pattern)

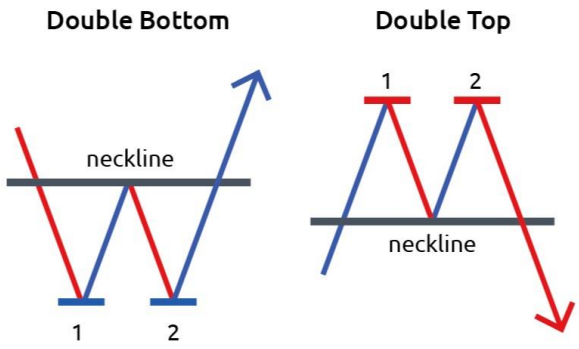
Σχηματισμός Κούπα με Χερούλι θεωρείται, σύμφωνα με την ιστοσελίδα Investopedia, ένας αρκετά αξιόπιστος σχηματισμός του οποίου η διάρκεια κυμαίνεται μεταξύ επτά και εξήντα πέντε εβδομάδες. Όπως μπορούμε να δούμε και στο Διάγραμμα 3.5, σε αυτές τις περιπτώσεις η διαγραμματική αποτύπωση της τιμής της μετοχής δημιουργεί σχηματισμούς που προσεγγίζουν την μορφή μιας κούπας με χερούλι. Το πιο ουσιαστικό σημείο αυτού του σχηματισμού βρίσκεται στο τέλους του «χερουλιού». Εάν η τιμή της μετοχής το διασπάσει ανοδικά, τότε δίνεται απευθείας σήμα αγοράς στον αναλυτή καθώς προμηνύεται αξιόλογη αύξηση της τιμής στο κοντινό μέλλον.

Διάγραμμα 3.6: Ο Σχηματισμός Κούπα Χερούλι

3.1.5 Σχηματισμός Διπλή Κορυφή και Διπλός Πυθμένας (Double Top and Bottom Pattern)

Ο σχηματισμός αυτός προκύπτει όταν η διαγραμματική απεικόνιση της τιμής σχηματίσει δύο κορυφές. Πιο συγκεκριμένα, στην περίπτωση του διαγράμματος της Διπλής Κορυφής, το διάγραμμα πραγματοποιεί μια άνοδο σχηματίζοντας την πρώτη κορυφή ακολουθούμενη από μια πτώση της τιμής. Το τοπικό ελάχιστο που θα φτάσει αυτή η κάθοδος της τιμής, σηματοδοτεί και την γραμμή λαιμού του διαγράμματος. Στη συνέχεια, πραγματοποιείται και δεύτερη άνοδος η κορυφή της οποία δεν ξεπερνά κατά τουλάχιστον 3% το ύψος της πρώτης κορυφής ακολουθούμενη από μία υποχώρηση της τιμής. Εάν η τελευταία κάθοδος διασπάσει καθοδικά την γραμμή λαιμού, τότε έχουμε μια ισχυρή ένδειξη έντονης πτώσης της τιμής στο άμεσο μέλλον.

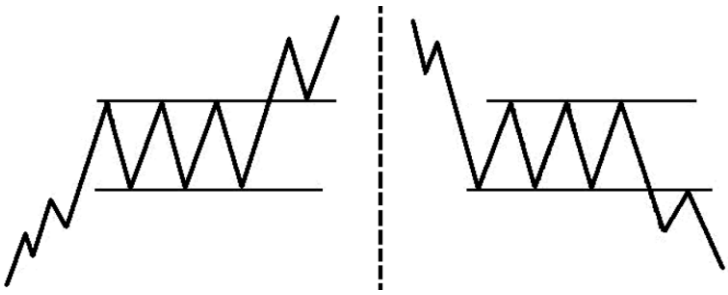
Αντιστρόφως εργαζόμαστε στην περίπτωση τους διαγράμματος του Διπλού Πυθμένα όπου εάν η τελευταία άνοδος της τιμής διαπεράσει ανοδικά την γραμμή λαιμού, η τιμής της μετοχής θα κινηθεί έντονα ανοδικά στο άμεσο μέλλον.



Διάγραμμα 3.7: Ο Σχηματισμός Διπλή Κορυφή και Διπλός Πυθμένας

3.1.6 Σχηματισμός Παραλληλογράμμου

Ο διαγραμματικός σχηματισμός Παραλληλογράμμου, αντιπροσωπεύει την παύση της ήδη υπάρχουσας τάσης. Ο σχηματισμός αυτός προκύπτει από την ύπαρξη δύο παράλληλων ευθειών μεταξύ των οποίων παρουσιάζεται μια πλευρική κίνηση των τιμών χωρίς να διαφαίνεται κάποια σαφή κατεύθυνση. Σε γενικές γραμμές, δεν μπορούμε να πούμε με ευκολία εάν ο σχηματισμός του Παραλληλογράμμου θα δώσει σήμα συνέχισης της τάσης, καθώς για να συμβεί αυτό θα πρέπει η γραμμή των τιμών να διασπάσει το Παραλληλόγραμμο. Από τη διάσπαση αυτή, αναλόγως σε ποια πλευρά του Παραλληλογράμμου έγινε, δίνεται και σήμα συνέχισης της πορείας της τιμής προς εκείνη την κατεύθυνση.

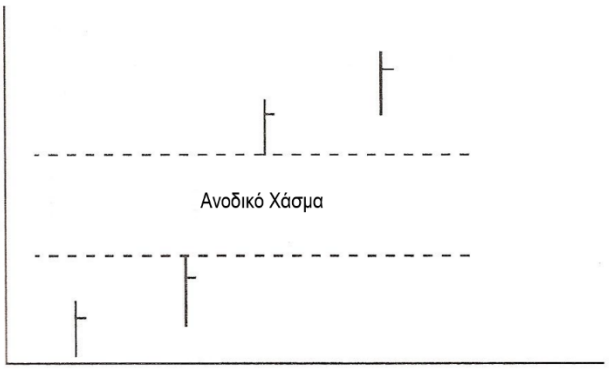


Διάγραμμα 3.8: Ο Σχηματισμός του Παραλληλογράμμου.

3.1.7 Χάσματα

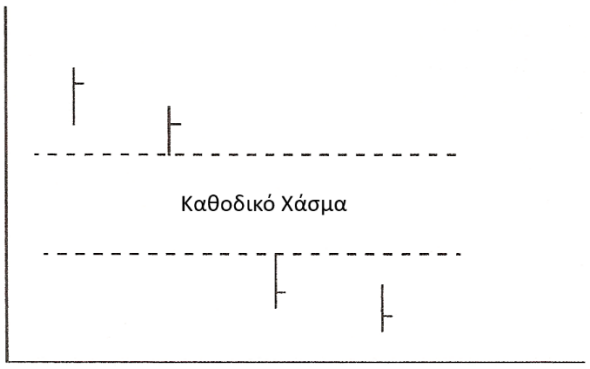
Ένα χάσμα αποτελεί ένα εύρος τιμών μέσα στο οποίο δεν παρατηρούνται χρηματιστηριακές διαπραγματεύσεις. Κάτι τέτοιο μπορεί να οφείλεται σε αρκετά σημαντικές προσυνεδριακές ειδήσεις.

Ανοδικό Χάσμα παρατηρείται όταν ανάμεσα σε δύο διαδοχικές χρηματιστηριακές συνεδριάσεις και σε περίοδο έντονης ανοδικής τάσης, δημιουργείται ένα κενό (χάσμα) μεταξύ της υψηλότερης τιμής της προηγούμενης ημέρας και την χαμηλότερη τιμή της επόμενης ημέρας.



Διάγραμμα 3.9: Ανοδικό Χάσμα

Καθοδικό Χάσμα αντιθέτως, έχουμε σε περιόδους έντονης καθοδικής τάσης όπου η κατώτερη τιμής της προηγούμενης ημέρας είναι υψηλότερη από την ανώτερη της επόμενης ημέρας.



Διάγραμμα 3.10: Καθοδικό Χάσμα

Σε αυτό το σημείο αξίζει να σημειώσουμε πως σε καταστάσεις που παρατηρείται μια ενδεχόμενη αντιστροφή της τάσης, τα Χάσματα μπορούν να βοηθήσουν στην επιβεβαίωση ή της απόρριψη αυτής της αντιστροφής. Έστω για παράδειγμα πως βρισκόμαστε στο τέλος μιας ήδη υπάρχουσας ανοδικής πορείας με έντονα στοιχεία ανάκαμψης. Εάν στο διάγραμμα των τιμών παρατηρηθεί ένα Καθοδικό Χάσμα όπως αυτό τους Διαγράμματος 3.9, τότε η συνέχιση της πτώσης πρέπει να θεωρηθεί δεδομένη χωρίς πιθανή διορθωτική κάμψη στο άμεσο μέλλον.

3.1.8 Όγκος Συναλλαγών

Ο Όγκος Συναλλαγών αντιπροσωπεύει το ύψος του επενδυτικού ενδιαφέροντος στην αγορά. Με λίγα λόγια, ο Όγκος Συναλλαγών είναι ο αριθμός των μετοχών που διαπραγματεύονται κατά τη διάρκεια μιας χρονικής περιόδου διαπραγμάτευσης, είτε αυτή είναι ημερήσια, είτε σε εκτενέστερη χρονική περίοδο.

Με αυτόν τον τρόπο, παρατηρώντας την τάση της τιμής σε συνδυασμό με τον Όγκο Συναλλαγών της μετοχής που μας ενδιαφέρει, μπορούμε να εξάγουμε συμπεράσματα για ενδεχόμενα σήματα αγοράς και πώλησης σύμφωνα με τον παρακάτω πίνακα:

Τάση της Τιμής	**Όγκος Συναλλαγών**	**Σήμα Αγοράς / Πώλησης**
Ανοδική Τάση	Ανοδική Τάση	Ένδειξη Αγοράς
Ανοδική Τάση	Καθοδική Τάση	Ένδειξη Πώλησης
Καθοδική Τάση	Ανοδική Τάση	Ένδειξη Πώλησης
Καθοδική Τάση	Καθοδική Τάση	Ένδειξη Αγοράς

Πίνακας 3.1: Σήματα Αγοράς / Πώλησης με βάση τον Όγκο Συναλλαγών και την τάση της τιμής

3.2 Τεχνικοί Δείκτες και Ταλαντωτές (Technical Indicators and Oscillators)

Οι Τεχνικοί Δείκτες και οι Ταλαντωτές χρησιμοποιούνται ευρέως από τους τεχνικούς αναλυτές, καθώς αποτελούν ένα από τα πιο σημαντικά εργαλεία διαπραγμάτευσης στο χρηματιστήριο. Ένα από τα βασικότερα χαρακτηριστικά των Τεχνιτών Δεικτών είναι η δυνατότητα παρακολούθησης των τιμών μιας μετοχής ή του ακόμα και του Γενικού Δείκτη του χρηματιστηρίου με την προϋπόθεση πως δεν θα υπάρξουν κάποια έντονα γεγονότα, είτε χρηματιστηριακής φύσεως είτε όχι, τα οποία θα έχουν οποιαδήποτε επιρροή στην πορεία των τιμών.

Οι τιμές των μετοχών από μόνες τους αλλά και σε συνδυασμό με τον όγκο συναλλαγών, συσχετίζονται άμεσα με τους Τεχνικούς Δείκτες κάνοντας ακόμα πιο εύκολη την διαδικασία της ανάλυσης σε σχέση με την διαγραμματική ανάλυση. Στις περισσότερες περιπτώσεις, η ανάλυση βασιζόμενη σε Τεχνικούς Δείκτες θεωρείται αρκετά πιο αποτελεσματική και αντικειμενική σε σχέση με την διαγραμματική ανάλυση η οποία είναι αρκετά υποκειμενική και η αποτελεσματικότητά της είναι δύσκολο να διαπιστωθεί.

Ένα ακόμα χαρακτηριστικό των Τεχνικών Δεικτών, είναι πως χρησιμοποιούνται με μεγάλη αποτελεσματικότητα σε διαστήματα όπου παρατηρείται μια συγκεκριμένη πορεία της μετοχής. Υπάρχουν όμως και περίοδοι όπου η πορεία της μετοχής δε διατηρεί κάποια συγκεκριμένη πορεία και διασπά συνεχώς τους δείκτες τάσης. Αυτό οδηγεί τους αναλυτές σε εσφαλμένα συμπεράσματα κάνοντας ζημιογόνες αγοραπωλησίες.

Σε αυτό το σημείο έρχονται οι Ταλαντωτές, οι οποίοι επιλύουν το παραπάνω πρόβλημα και χρησιμοποιούνται ευρέως ιδίως σε περιόδους που οι τιμές κινούνται πλευρικά. Βασικό χαρακτηριστικό τους είναι πως ταλαντώνονται εντός ενός ορίου τιμών εκατέρωθεν μιας γραμμής ισορροπίας η οποία συνήθως είναι το μηδέν ή το εκατό.

3.2.1 Κινητός Μέσος Όρος (Moving Average)

Ο απλούστερος αλλά και πολύ αποτελεσματικός Τεχνικός Δείκτης είναι ο Κινητός Μέσος Όρος ή απλούστερα ΚΜΟ. Βασικό χαρακτηριστικό αυτού του δείκτη, είναι η δυνατότητα απάλειψης των ιδιαίτερα έντονων διακυμάνσεων που μπορεί να παρουσιάσει η τιμή της μετοχής. Όπως

βλέπουμε και στο Διάγραμμα 3.10, αυτό έχει ως αποτέλεσμα την ομαλοποίηση της γραμμής της μετοχής κάνοντάς την πιο ευανάγνωστη στον αναλυτή.

Για μια τιμή μετοχής μπορούμε να ορίσουμε διάφορους ΚΜΟ ανάλογα την χρονική περίοδο που θέλουμε να εξετάσουμε. Έτσι μπορούμε να έχουμε ΚΜΟ των πέντε, δέκα, τριάντα ημερών. Για τον υπολογισμό του απλώς βρίσκουμε τον μέσο όρο των τιμών που έλαβε η μετοχή το συγκεκριμένο διάστημα.

Διάγραμμα 3.11: Κινητός Μέσος Όρος

Σε περιόδους ανοδικής τάσης, ο ΚΜΟ κάθε ημέρας λαμβάνει μικρότερη τιμή από την αντίστοιχη τιμή κλεισίματος έχοντας ως αποτέλεσμα να βρίσκεται συνεχώς κάτω από την καμπύλη των τιμών της μετοχής. Σε αντίθετη περίπτωση, όταν δηλαδή παρατηρείται καθοδική τάση των τιμών, ο ΚΜΟ βρίσκεται επάνω από την καμπύλη των τιμών των μετοχών.

Σήματα αγοροπωλησιών με βάση τον ΚΜΟ έχουμε στις εξής περιπτώσεις:

- Όταν η καμπύλη του ΚΜΟ διασπάται ανοδικά από την καμπύλη των τιμών, έχουμε σήμα αγοράς καθώς διαφαίνεται έναρξη ανοδικής τάσης.

- Όταν η καμπύλη του ΚΜΟ διασπάται καθοδικά από την καμπύλη των τιμών, έχουμε σήμα πώλησης καθώς διαφαίνεται έναρξη καθοδικής τάσης.

- Όταν παρατηρείται αλλαγή της κλίσης της καμπύλης του ΚΜΟ, έχουμε ένδειξη αντιστροφής της τάσης. Έτσι για παράδειγμα εάν βρισκόμαστε σε ανοδική τάση και η καμπύλη του ΚΜΟ αντιστραφεί, αυτό αποτελεί σήμα πώλησης καθώς η τιμή θα ακολουθήσει και αυτή καθοδική πορεία στο μέλλον.

3.2.2 Σταθμισμένος Κινητός Μέσος Όρος (Weighted Moving Average)

Ένα μειονέκτημα του απλού ΚΜΟ είναι η μεγάλη χρονική υστέρηση που εμφανίζει, καθώς αντιμετωπίζει όλες τις τιμές με την ίδια ακριβώς βαρύτητα.

Αντίθετα, ο Σταθμισμένος Κινητός Μέσος Όρος, δίνει μεγαλύτερη σημασία στις νεότερες τιμές, έχοντας ως αποτέλεσμα την γρηγορότερη αλλαγή κατεύθυνσης. Αυτό δίνει στους αναλυτές μεγαλύτερη ακρίβεια αλλά καλύτερο χρόνο αντίδρασης σε απότομες αλλαγές των τιμών. Πολλοί αναλυτές πιστεύουν πως η παρατήρηση του Σταθμισμένου έναντι του απλού ΚΜΟ είναι προτιμότερη.

Διάγραμμα 3.12: Σταθμισμένος Κινητός Μέσος Όρος (μπλε) έναντι του Απλού Κινητού Μέσου Όρου

3.2.3 Ο Δείκτης Όγκου Ισορροπίας (On Balance Volume, OBV)

Ο Δείκτης Όγκου Ισορροπίας (OBV) δημιουργεί μια συσχέτιση μεταξύ του όγκου των συναλλαγών και των μεταβολών των τιμών. Για να δημιουργήσουμε τον OBV ακολουθούμε τα εξής βήματα:

- Αν η τιμή κλεισίματος της ημέρας Κ είναι υψηλότερη από την τιμή κλεισίματος της Κ-1 ημέρας, τότε ο όγκος συναλλαγών της ημέρας Κ προστίθεται στον OBV της ημέρας Κ-1.

- Σε αντίθετη περίπτωση, ο όγκος συναλλαγών της Κ ημέρα αφαιρείται από τον OBV της Κ-1 ημέρας.

Σαν ερμηνεία του δείκτη OBV, μπορούμε να πούμε πως οι αλλαγές στην κατεύθυνσή του πρέπει να προηγούνται των αλλαγών των τιμών. Σε αντίθετη περίπτωση, εάν για παράδειγμα η τιμή αρχίσει να ανεβαίνει χωρίς να συμβαίνει το ίδιο και με τον OBV, έχουμε ένδειξη πως η τιμή έχεις φτάσει στα ανώτατο σημείο της. Αντίστροφα, εάν η τιμή αρχίσει να εμφανίζει πτωτική πορεία χωρίς να συμβαίνει το ίδιο και με τον OBV, έχουμε ένδειξη πως η τιμή έχεις φτάσει στα κατώτατο σημείο της.

Η παραγωγή των σημάτων αγοράς και πώλησης που αποτυπώνεται στον Πίνακα 3.2, μας δείχνει πως για να έχουμε σήμα αγοράς πρέπει ο OBV να έχει ανοδική τάση, ανεξάρτητα από την τάση

της μετοχής. Αντίστοιχα, για να έχουμε πώληση, ο OBV πρέπει να παρουσιάζει καθοδική τάση και πάλι ανεξάρτητα από την τάση της μετοχής.

Τάση της Τιμής	**OBV**	**Σήμα Αγοράς / Πώλησης**
Ανοδική Τάση	Ανοδική Τάση	Ένδειξη Αγοράς
Ανοδική Τάση	Καθοδική Τάση	Ένδειξη Πώλησης
Καθοδική Τάση	Ανοδική Τάση	Ένδειξη Αγοράς
Καθοδική Τάση	Καθοδική Τάση	Ένδειξη Πώλησης

Πίνακας 3.2: Ο Δείκτης Όγκου Ισορροπίας και τα αντίστοιχα σήματα αγοράς και πώλησης.

3.2.4 Δείκτης Σχετικής Ισχύος (Relative Strength Index – RSI)

Ο Δείκτης Σχετικής Ισχύος (RSI), είναι ένας από τους πιο δημοφιλής δείκτες. Ο υπολογισμός του σύμφωνα με τους δημιουργούς του Wilder και Since, πρέπει να βασίζεται σε χρονικές περιόδους των δεκατεσσάρων ημερών και να λαμβάνει τιμές από μηδέν έως εκατό.

Πιο συγκεκριμένα για τον υπολογισμό του ακολουθούμε τα εξής βήματα:

- Αρχικά υπολογίζουμε το Μέσο Κέρδος αθροίζοντας όλα και κέρδη των τιμών που έχουν σημειωθεί τις τελευταίες δεκατέσσερεις ημέρες και διαιρώντας με το δεκατέσσερα.

- Στη συνέχεια υπολογίζουμε την Μέση Απώλεια αθροίζοντας όλες τις επιμέρους απώλειες που έχουν σημειωθεί τις τελευταίες δεκατέσσερεις ημέρες και διαιρώντας με το δεκατέσσερα. Εδώ να σημειωθεί πως η απώλεια είναι πάντα θετικός αριθμός καθώς αντιπροσωπεύει μια ποσότητα.

- Υπολογίζουμε την σχετική ισχύ RS, η οποία ισούται με το Μέσο Κέρδος διά την Μέση Απώλεια (RS = Average Gain / Average Loss)

- Τέλος υπολογίζουμε τον RSI με τον εξής τύπο: RSI = 100 – (100 / (1+RS))

Εάν τον δούμε διαγραμματικά, οι Wilder και Since όρισαν πως όταν ο δείκτης RSI λάβει τιμή μεγαλύτερη από εβδομήντα, τότε μας δίδεται η ένδειξη πως η μετοχή έχει αρχίσει να υπερτιμάται. Αυτό μπορεί να οφείλεται στη μεγάλη ζήτηση της μετοχής, προειδοποιώντας για ενδεχόμενη διορθωτική πορεία στο μέλλον, δηλαδή για πτώση. Εδώ να πούμε πως η διορθωτική αυτή πορεία μπορεί να μην είναι στο άμεσο μέλλον καθώς ενδεχομένως για ένα μικρό χρονικό διάστημα η τιμή να κινηθεί πλευρικά.

Στην αντίθετη περίπτωση, όταν δηλαδή ο δείκτης λάβει τιμή μικρότερη των τριάντα, τότε αυτό αποτελεί ένδειξη υποτίμησης της μετοχής. Αυτή μπορεί να οφείλεται σε πολύ μεγαλύτερες σε

σχέση με το πρόσφατο παρελθόν πωλήσεις και αποτελεί ένδειξη πιθανής ανάκαμψής στο μέλλον.

Διάγραμμα 3.13: Ο Δείκτης Σχετικής Ισχύος RSI

3.2.5 Ο Ταλαντωτής Τιμών (Price Oscillator)

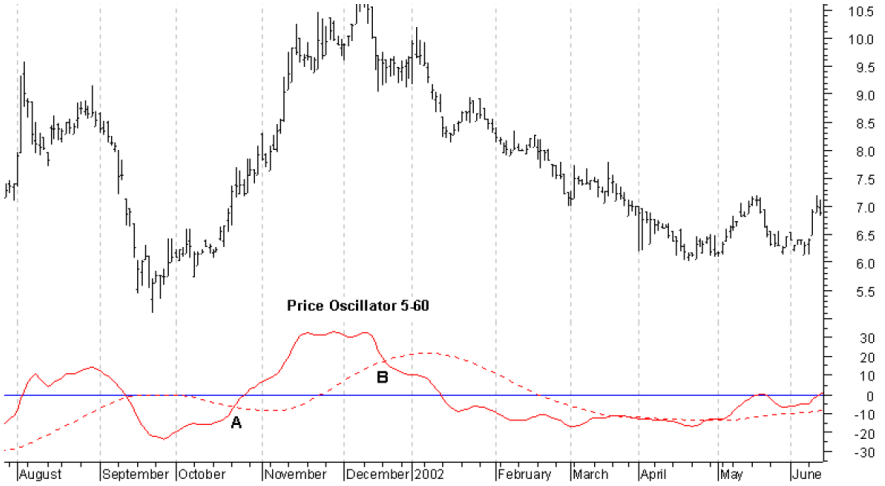
Ο Ταλαντωτής Τιμών, μπορούμε να τον υπολογίσουμε αφαιρώντας την ποσοστιαία διαφορά ενός βραχυχρόνιου ΚΜΟ και ενός μακροχρόνιου ΚΜΟ. Συνήθως, οι πλειονότητα των αναλυτών προτιμούν τους ΚΜΟ πέντε ημερών και εξήντα ημερών για βραχυχρόνιο και μακροχρόνιο ΚΜΟ αντίστοιχα. Ο υπολογισμός του γίνεται σύμφωνα με τον ακόλουθο τύπο:

$$Pr.Osc_{5\text{-}60} = = \frac{KMO_5 - KMO_{60}}{KMO_{60}} \times 100$$

Εξίσωση 3.1

Σήματα αγοράς και πώλησης μπορούμε να έχουμε σε περιπτώσεις διάσπασης του ΚΜΟ των σαράντα πέντε ημερών του $Pr.Osc_{5\text{-}60}$ από τον ίδιο τον $Pr.Osc_{5\text{-}60}$. Πιο συγκεκριμένα:

- Όταν ο $Pr.Osc_{5\text{-}60}$ διασπάσει ανοδικά τον KMO_{45} έχουμε σήμα αγοράς
- Αντίθετα, όταν ο $Pr.Osc_{5\text{-}60}$ διασπάσει καθοδικά τον KMO_{45} έχουμε σήμα πώλησης



Διάγραμμα 3.14: Ο Ταλαντωτής Τιμών

3.2.6 Λωρίδες Bollinger (Bollinger Bands)

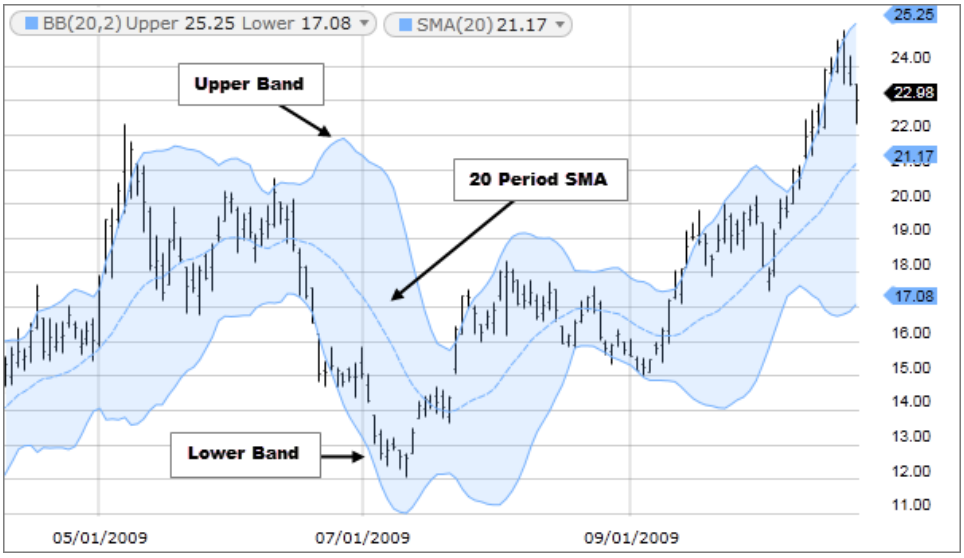
Σε αρκετές περιπτώσεις μπορεί να παρατηρηθούν έντονες διακυμάνσεις στις τιμές της μετοχής. Από αυτές, μπορούμε να βγάλουμε εύκολα συμπεράσματα για σήματα αγοράς και πώλησης εάν τις οριοθετήσουμε μεταξύ δύο περιοχών μέσα στις οποίες αναμένετε να ταλαντωθούν. Αυτό αποτέλεσε και το έναυσμα για τον John Bollinger να δημιουργήσει Bollinger Bands οι οποίες αποτελούνται από έναν $ΚΜΟ_{20}$ που περιβάλλεται από δύο λωρίδες. Η άνω λωρίδα προκύπτει από το άθροισμα του $ΚΜΟ_{20}$ και δύο τυπικών αποκλίσεων των τιμών, ενώ η κάτω εάν αφαιρέσουμε δύο τυπικές αποκλίσεις των τιμών από των $ΚΜΟ_{20}$.

Με αυτόν τον τρόπο, ελέγχονται δυναμικά οι διακυμάνσεις καθώς όσο αυξάνονται αυτές άλλο τόσο θα αυξάνονται και οι τυπικές αποκλίσεις και αντίστοιχα και το πλάτος των λωρίδων. Οι τελευταίες, βοηθούν στην στήριξη και στην αντίσταση της τιμής των μετοχών καθώς όταν η τιμή προσεγγίσει την άνω λωρίδα πιθανότατα θα βρει αντίσταση προχωρώντας σε διορθωτική πορεία προς τα κάτω. Αντίστοιχα, όταν προσεγγίσει την κάτω λωρίδα θα βρει στήριξη με πιθανή ανοδική πορεία στη συνέχεια.

Τα σήματα αγοράς και πώλησης δίδονται στις εξής δύο περιπτώσεις:

- Όταν η τιμή της μετοχής διασπάσει ανοδικά την άνω λωρίδα, τότε έχουμε σήμα αγοράς

- Όταν η τιμή της μετοχής διασπάσει καθοδικά την κάτω λωρίδα, τότε έχουμε σήμα πώλησης.



Διάγραμμα 3.15: Οι λωρίδες Bollinger

3.2.7 Ο Ταλαντωτής Williams (Williams Oscillator)

Ο Ταλαντωτής Williams %R λαμβάνει την τιμή κλεισίματος της τρέχουσας χρηματιστηριακής συνεδρίας και τη συγκρίνει με την υψηλότερη τιμή κλεισίματος της επιθυμητής χρονικής περιόδου. Αυτή η περίοδος συνήθως αντιπροσωπεύει δεκατέσσερις χρηματιστηριακές συνεδρίες.

Ο τύπος υπολογισμού του είναι ο ακόλουθος:

$$\text{Wiliams } \%R = \frac{\text{Highest High} - \text{Close}}{\text{Highest High} - \text{Lowest Low}}$$

Εξίσωση 3.2

Όπου:

- Highest High = Η μεγαλύτερη τιμή κλεισίματος της περιόδου
- Close = Η τρέχουσα τιμή κλεισίματος
- Lowest Low = Η μικρότερη τιμή κλεισίματος της περιόδου

Ο Ταλαντωτής Williams %R λαμβάνει τιμές μεταξύ μείον εκατό και μηδέν, μεταξύ των οποίων μια μετοχή θεωρείται υπερτιμημένη όταν ο δείκτης βρίσκεται στο διάστημα μεταξύ μηδέν και μείον είκοσι. Αντιθέτως, όταν ο δείκτης βρίσκεται μεταξύ του μείον ογδόντα και του μείον εκατό, η μετοχή θεωρείται πως είναι υποτιμημένη.

Τα σήματα αγοράς και πώλησης μπορούν να εξαχθούν ως εξής:

- Όταν ο Ταλαντωτής διασπάσει ανοδικά την γραμμή του μείον ογδόντα έχουμε σήμα αγοράς.

- Όταν ο Ταλαντωτής διασπάσει καθοδικά την γραμμή του μείον είκοσι έχουμε σήμα πώλησης.

Αυτό που αξίζει να σημειώσουμε επιπλέον για τον Ταλαντωτή Williams %R, είναι η ιδιότητά του να σχηματίζονται στην υπερτιμημένη περιοχή κορυφές και στην υποτιμημένη πυθμένες λίγες ημέρες πριν αυτές σχηματιστών στην καμπύλη των τιμών. Αυτό μπορεί να χρησιμοποιηθεί ως σήμα αγοράς ή πώλησης καθώς διαφαίνεται η αντιστροφή της τάσης πριν αυτή αρχίσει να υφίσταται.

Διάγραμμα 3.16: Ο Ταλαντωτής Williams %R

3.2.8 Ο Ταλαντωτής του Ρυθμού Μεταβολής (Rate of Change Oscillator, ROC)

Ο Ταλαντωτής του Ρυθμού Μεταβολής (ROC) μετράει την διαφορά μεταξύ της πιο πρόσφατης τιμής κλεισίματος της μετοχής και της τιμής της K ημέρες πριν. Ο τύπος του Ταλαντωτής ROC για K = 5 είναι ο εξής:

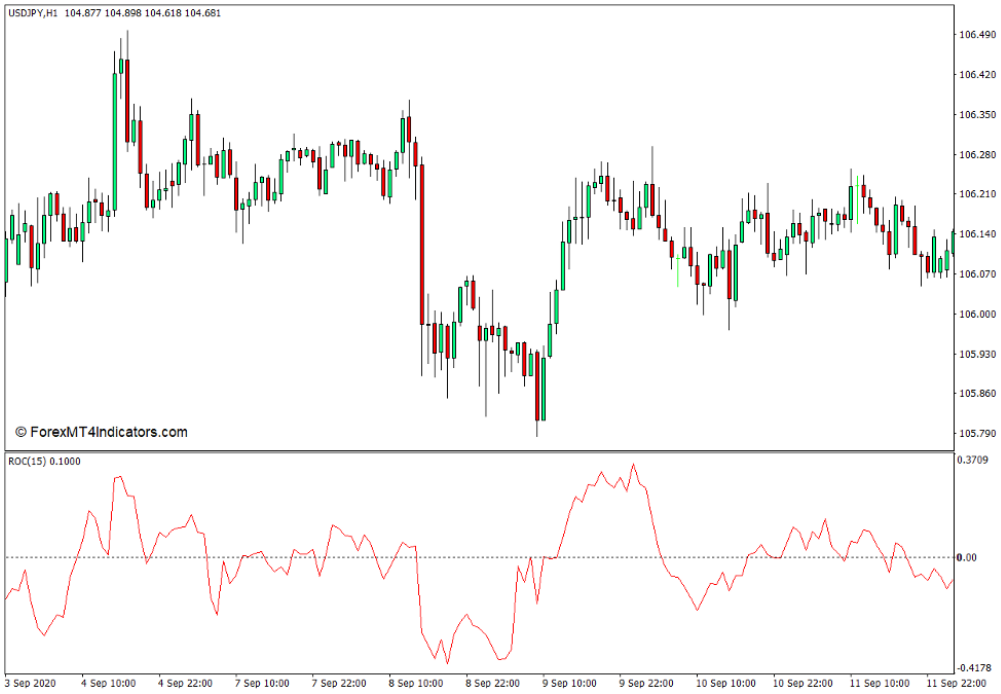
$$ROC = \left(\frac{Price_t}{Price_{t-K}}\right) - 1 \times 100$$

Εξίσωση 3.3

Όπως είναι φανερό και στο τύπο, αρχικά διαιρούμε την τρέχουσα τιμή κλεισίματος $Price_t$ με την τιμή κλεισίματος K ημέρες πριν $Price_{t-K}$. Στη συνέχεια αφαιρούμε από τον κλάσμα την μονάδα

και πολλαπλασιάζουμε με το εκατό. Αυτό έχει ως αποτέλεσμα την δημιουργία ενός ποσοστού που μα βοηθάει να αντιληφθούμε εύκολα την ποσοστιαία μεταβολή της τιμής σε αυτή την χρονική περίοδο των Κ ημερών.

Βασικό του πλεονέκτημα είναι η απλότητα που τον χαρακτηρίζει. Αντίθετα όμως, στα μειονεκτήματα συγκαταλέγεται η αδυναμία εξομάλυνσης έχοντας ως αποτέλεσμα η καμπύλη του Ταλαντωτή ROC να χαρακτηρίζεται από έντονες ακμές και απότομες μεταβολές στην κατεύθυνσή του. Επιπλέον, σε αντίθεση με άλλους αρκετά πιο περίπλοκους Ταλαντωτές, δεν αξιοποιεί άλλες πληροφορίες για το χρονικό διάστημα t με t-K.



Διάγραμμα 3.17: Ο Ταλαντωτής του Ρυθμού Μεταβολής

3.2.9 Commodity Channel Index (CCI)

Ο Ταλαντωτής Commodity Channel Index (CCI), χρησιμοποιείται για την μέτρηση της απόκλισης της τιμής μιας μετοχής από το στατικό μέσο όρο της. Ο Donald Lampert που δημιούργησε τον Ταλαντωτή, το είχε σχεδιάσει για το χρηματιστήριο εμπορευμάτων. Ωστόσο, δουλεύει εξίσου καλά και για την ανάλυση στο χρηματιστήριο μετοχών.

Ο τύπος υπολογισμού του Ταλαντωτή CCI είναι ο εξής:

$$CCI = \frac{T_1 - M}{0.015D}$$

Εξίσωση 3.4

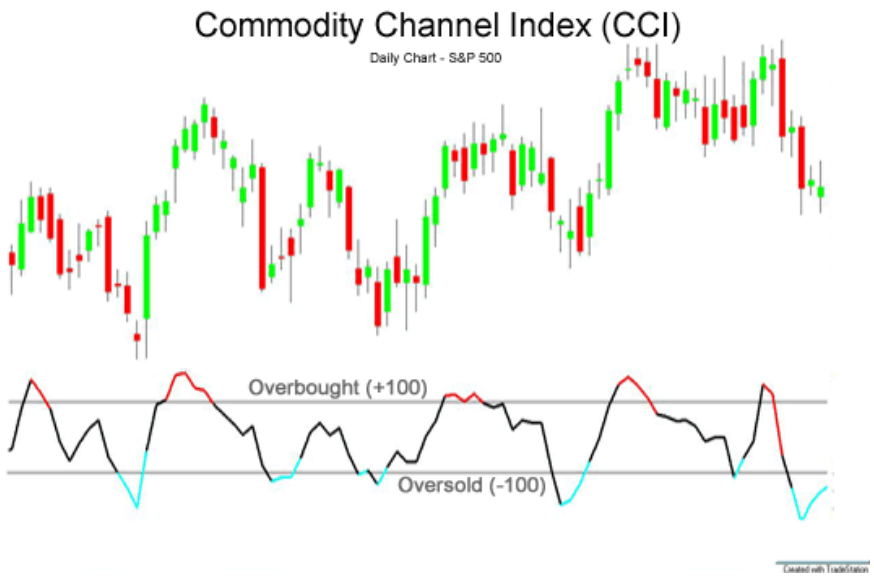
Όπου:

- T_1 είναι ο μέσος όρος της μέγιστης, της ελάχιστης και της τιμής κλεισίματος της ημέρας
- Μ είναι ο Κινητός Μέσος Όρος των T_i της περιόδου που μας ενδιαφέρει
- D είναι η μέση απόκλιση των T_i της περιόδου που μας ενδιαφέρει

Στον Ταλαντωτή CCI δεν έχουμε κάποια συγκεκριμένα όρια μέσα στα οποία μπορεί να ταλαντωθεί αλλά η πλειονότητα των τιμών που λαμβάνει βρίσκεται μεταξύ του μείον εκατό και του εκατό. Αυτές οι τιμές καθορίζουν και τις υπερπωλημένες και υπεραγορασμένες ζώνες. Πιο συγκεκριμένα, κάτω από την τιμών των μείον εκατό βρίσκεται η υπερπωλημένη περιοχή ενώ πάνω από το εκατό η υπεραγορασμένη.

Σήματα αγοράς και πώλησης έχουμε στις εξής περιπτώσεις:

- Όταν η γραμμή του CCI διασπάσει ανοδικά την τιμή μείον εκατό, έχουμε σήμα αγοράς
- Όταν η γραμμή του CCI διασπάσει καθοδικά την τιμή εκατό, έχουμε σήμα πώλησης



Διάγραμμα 3.18: Ο Ταλαντωτής Commodity Chanel Index

3.2.10 Ο Ταλαντωτής Chaikin A/D (CAD)

Ο Ταλαντωτής Chaikin A/D (CAD) βασίζεται στη θεώρηση της συσσώρευσης / διανομής του όγκου συναλλαγών. Γενικώς, σε περιόδους όπου η τιμή κλεισίματος είναι μεγαλύτερη από την μέση ημερήσια τιμή ο CAD θεωρεί πως συσσωρεύεται όγκος, δηλαδή έχουμε συσσώρευση κεφαλαίων το οποίο είναι και σημάδι ανόδου της τιμής στο άμεσο μέλλον. Σε αντίθετη περίπτωση, ό όγκος των συναλλαγών διανέμεται δηλαδή απομακρύνονται επενδυτικά κεφάλαια, το οποίο αποτελεί και σημάδι πτώσης της τιμής στο κοντινό μέλλον.

Ο τύπος υπολογισμού του CAD είναι ο εξής:

$$CAD = \left(\frac{2p - Pmin - Pmax}{Pmax - Pmin}\right) \times V$$

Εξίσωση 3.5

Όπου:

- p = Η σημερινή τιμή κλεισίματος
- Pmin = Η ελάχιστη τιμή της σημερινής συνεδρίασης
- Pmax = Η μέγιστη τιμή της σημερινής συνεδρίασης
- V = Ο όγκος των συναλλαγών της σημερινής συνεδρίασης

Σαν γραμμή ισορροπίας στον Ταλαντωτή CAD έχουμε την γραμμή μηδέν όπου πάνω από αυτήν βρίσκεται η υπεραγορασμένη περιοχή και κάτω η υπερπωλημένη. Σήμα αγοράς έχουμε όταν ο CAD διασπάσει ανοδικά την γραμμή ισορροπίας ενώ πώλησης όταν την διασπάσει καθοδικά.

Διάγραμμα 3.19: Ο Ταλαντωτής Chaikin A/D

Chapter 4

Υλοποίηση

Στόχος του πρακτικού μέρους της εργασίας ήταν η πρόβλεψη τιμών μετοχών χρησιμοποιώντας αρχιτεκτονική Transformer Νευρωνικού Δικτύου και η μετέπειτα χρήση των προβλεπόμενων από αυτό το μοντέλο τιμών για την παραγωγή σημάτων αγοράς, διατήρησης ή πώλησης μιας μετοχής. Αυτά τα σήματα θα χρησιμοποιηθούν στην συνέχεια για την δημιουργία ενός trading-bot, μέσω του οποίου θα αγοράζονται ή θα πωλούνται μετοχές. Επιπλέον, για λόγους πληρότητας της πειραματικής διαδικασίας, δημιουργήθηκε και LSTM Νευρωνικό Δίκτυο, με σκοπό την παροχή συγκριτικών αποτελεσμάτων.

Όσο αφορά την γλώσσα υλοποίησης του πειραματικού μέρους, είναι η Python, η οποία ενδείκνυται για δημιουργία μοντέλων μηχανικής μάθησης και για επεξεργασία μεγάλου όγκου δεδομένων.

Επιπλέον, το εργαλείο ανάπτυξης της πειραματικής διαδικασία ήταν το Jupyter Notebook, το οποίο αποτελεί επίσης ένα από τα πλέον διαδεδομένα εργαλεία ανάπτυξης υλοποιήσεων σε γλώσσα Python.

Σημαντικές βιβλιοθήκες, οι οποίες βοήθησαν στην πραγματοποίηση της πειραματικής διαδικασίας είναι οι παρακάτω:

- pandas
- numpy
- tensorflow
- matplotlib

Το μοντέλο Transformer αξιολογήθηκε ως προς τις μετρικές:

- Loss
- MAE (Mean Absolute Error)
- MAPE (Mean Absolute Percentage Error)

Όσο αφορά, το σύνολο δεδομένων που χρησιμοποιήθηκε, αφορούσε στοιχεία της μετοχής Apple. Ωστόσο, το προτεινόμενο μοντέλο θα μπορούσε να χρησιμοποιηθεί για οποιοδήποτε άλλη μετοχή.

Στην συνέχεια του κεφαλαίου παρουσιάζονται σταδιακά τμήματα κώδικα τα οποία χρησιμοποιήθηκαν, καθώς και τα καταγραφόμενα αποτελέσματα, όπως και τις αντίστοιχες οπτικοποιήσεις για καλύτερη κατανόηση των αποτελεσμάτων.

4.1 Χρησιμοποιούμενο σύνολο δεδομένων

Όπως αναφέρθηκε και στην εισαγωγική ενότητα, χρησιμοποιήθηκε σαν σύνολο δεδομένων ένα αρχείο με 6081 εγγραφές (καλύπτουν τα έτη από 1984 έως 2008) της μετοχής Apple. Κάθε εγγραφή περιέχει τις παρακάτω στήλες:

- **Date**: ημερομηνία
- **Open**: τιμή ανοίγματος
- **High**: υψηλότερη τιμή
- **Low**: χαμηλότερη τιμή
- **Close**: τιμή κλεισίματος
- **Volume**: όγκος συναλλαγών
- **Adjclose**: τιμή της μετοχής μετά την εξόφληση των μερισμάτων

Στο διάγραμμα 4.1 μπορούμε να δούμε ένα μέρος του χρησιμοποιούμενο σύνολο δεδομένων:

```
Date,Open,High,Low,Close,Volume,Adj Close
2008-10-14,116.26,116.40,103.14,104.08,70749800,104.08
2008-10-13,104.55,110.53,101.02,110.26,54967000,110.26
2008-10-10,85.70,100.00,85.00,96.80,79260700,96.80
2008-10-09,93.35,95.80,86.60,88.74,57763700,88.74
2008-10-08,85.91,96.33,85.68,89.79,78847900,89.79
2008-10-07,100.48,101.50,88.95,89.16,67099000,89.16
2008-10-06,91.96,98.78,87.54,98.14,75264900,98.14
2008-10-03,104.00,106.50,94.65,97.07,81942800,97.07
2008-10-02,108.01,108.79,100.00,100.10,57477300,100.10
2008-10-01,111.92,112.36,107.39,109.12,46303000,109.12
2008-09-30,108.25,115.00,106.30,113.66,58095800,113.66
2008-09-29,119.62,119.68,100.59,105.26,93581400,105.26
```

Διάγραμμα 4.1: Μέρος του χρησιμοποιούμενο σύνολο δεδομένων

4.2 Εισαγωγή Δεδομένων

Πρώτο βήμα στην πειραματική διαδικασία ήταν η εισαγωγή των δεδομένων, η αντικατάσταση των μηδενικών στην περίπτωση της στήλης Volume (έτσι ώστε να μην υπάρχει πρόβλημα διαίρεσης με το μηδέν), η ταξινόμηση των εγγραφών σε αύξουσα σειρά ως προς την ημερομηνία, η εφαρμογή κινητού μέσου (moving average) με μέγεθος παραθύρου 10 ημερών σε όλες τις στήλες και η αφαίρεση όλων των γραμμών που περιείχαν έστω και μια NaN τιμή:

```
df = pd.read_csv('AAPL.csv', delimiter=',', usecols=['Date', 'Open', 'High', 'Low', 'Close', 'Volume'])
```

```
# Replace 0 to avoid dividing by 0 later on

df['Volume'].replace(to_replace=0, method='ffill', inplace=True)

df.sort_values('Date', inplace=True)



# Apply moving average with a window of 10 days to all columns

df[['Open', 'High', 'Low', 'Close', 'Volume']] = df[['Open', 'High', 'Low', 'Close', 'Volume']].rolling(10).mean()



# Drop all rows with NaN values

df.dropna(how='any', axis=0, inplace=True)

print(df.head())
```

Κώδικας 1

4.3 Οπτικοποίηση Δεδομένων

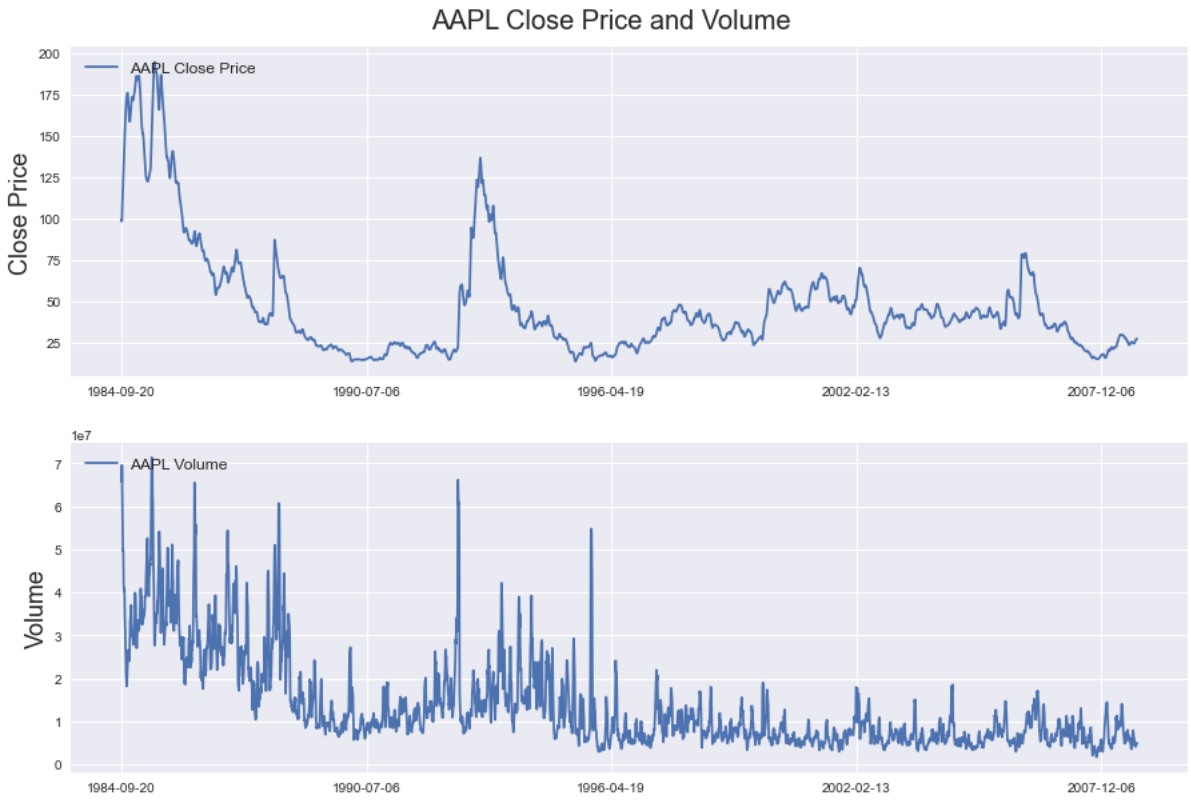
Στην συνέχεια, παρατίθεται κώδικας, ο οποίος χρησιμοποιείται για την δημιουργία οπτικοποίησης των δεδομένων, όσο αφορά τις στήλες Close Price και Volume:

```
fig = plt.figure(figsize=(15,10))

st = fig.suptitle(“AAPL Close Price and Volume”, fontsize=20)

st.set_y(0.92)



ax1 = fig.add_subplot(211)

ax1.plot(df[‘Close’], label=’AAPL Close Price’)

ax1.set_xticks(range(0, df.shape[0], 1464))

ax1.set_xticklabels(df[‘Date’].loc[::1464])
```

```
ax1.set_ylabel('Close Price', fontsize=18)

ax1.legend(loc="upper left", fontsize=12)


ax2 = fig.add_subplot(212)

ax2.plot(df['Volume'], label='AAPL Volume')

ax2.set_xticks(range(0, df.shape[0], 1464))

ax2.set_xticklabels(df['Date'].loc[::1464])

ax2.set_ylabel('Volume', fontsize=18)

ax2.legend(loc="upper left", fontsize=12)
```

Κώδικας 2

Ακολουθούν οι αντίστοιχες γραφικές παραστάσεις για το Close Price και το Volume:



Διάγραμμα 4.2: Οι γραφικές παραστάσεις των Close Price και Volume

4.4 Προ - επεξεργασία Δεδομένων

Ακολουθεί το τμήμα του κώδικα, που χρησιμοποιήθηκε για την προ - επεξεργασία των δεδομένων. Αρχικά χρησιμοποιήθηκε η pct_change() προκειμένου να υπολογιστεί ή επί τοις 100 μεταβολή μεταξύ διαδοχικών τιμών της κάθε στήλης (τα αποκαλούμενα returns). Στην περίπτωση μας, θα ασχοληθούμε πλέον με τα returns των στηλών. Άλλη χαρακτηριστική λειτουργία που υλοποιήθηκε είναι αυτή της κανονικοποίησης των τιμών των στηλών χρησιμοποιώντας την μέθοδο MinMax. Και τέλος, πραγματοποιήθηκε διαχωρισμός του συνόλου δεδομένων σε train set (σύνολο εκπαίδευσης), validation set (σύνολο επικύρωσης) και test set (σύνολο ελέγχου).

```
df['Open'] = df['Open'].pct_change() # Create arithmetic returns column

df['High'] = df['High'].pct_change() # Create arithmetic returns column

df['Low'] = df['Low'].pct_change() # Create arithmetic returns column

df['Close'] = df['Close'].pct_change() # Create arithmetic returns column

df['Volume'] = df['Volume'].pct_change()


df.dropna(how='any', axis=0, inplace=True) # Drop all rows with NaN values


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

'''Create indexes to split dataset'''


times = sorted(df.index.values)

last_10pct = sorted(df.index.values)[-int(0.1*len(times))] # Last 10% of series

last_20pct = sorted(df.index.values)[-int(0.2*len(times))] # Last 20% of series


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

'''Normalize price columns'''

#

min_return = min(df[(df.index < last_20pct)][['Open', 'High', 'Low', 'Close']].min(axis=0))
```

```
max_return = max(df[(df.index < last_20pct)][['Open', 'High', 'Low', 'Close']].max(axis=0))


# Min-max normalize price columns (0-1 range)
df['Open'] = (df['Open'] - min_return) / (max_return - min_return)
df['High'] = (df['High'] - min_return) / (max_return - min_return)
df['Low'] = (df['Low'] - min_return) / (max_return - min_return)
df['Close'] = (df['Close'] - min_return) / (max_return - min_return)


#############################################################################
'''Normalize volume column'''


min_volume = df[(df.index < last_20pct)]['Volume'].min(axis=0)
max_volume = df[(df.index < last_20pct)]['Volume'].max(axis=0)


# Min-max normalize volume columns (0-1 range)
df['Volume'] = (df['Volume'] - min_volume) / (max_volume - min_volume)


#############################################################################
'''Create training, validation and test split'''
df_train = df[(df.index < last_20pct)]  # Training data are 80% of total data
df_val = df[(df.index >= last_20pct) & (df.index < last_10pct)]
df_test = df[(df.index >= last_10pct)]


# Remove date column
df_train.drop(columns=['Date'], inplace=True)
df_val.drop(columns=['Date'], inplace=True)
```

```
df_test.drop(columns=['Date'], inplace = True)


train_data = df_train.values

val_data = df_val.values

test_data = df_test.values

print('Training data shape: {}'.format(train_data.shape))

print('Validation data shape: {}'.format(val_data.shape))

print('Test data shape: {}'.format(test_data.shape))
```

Κώδικας 3

Όσο αφορά τις διαστάσεις των χρησιμοποιούμενων συνόλων δεδομένων, χαρακτηριστικός είναι ο παρακάτω πίνακας:

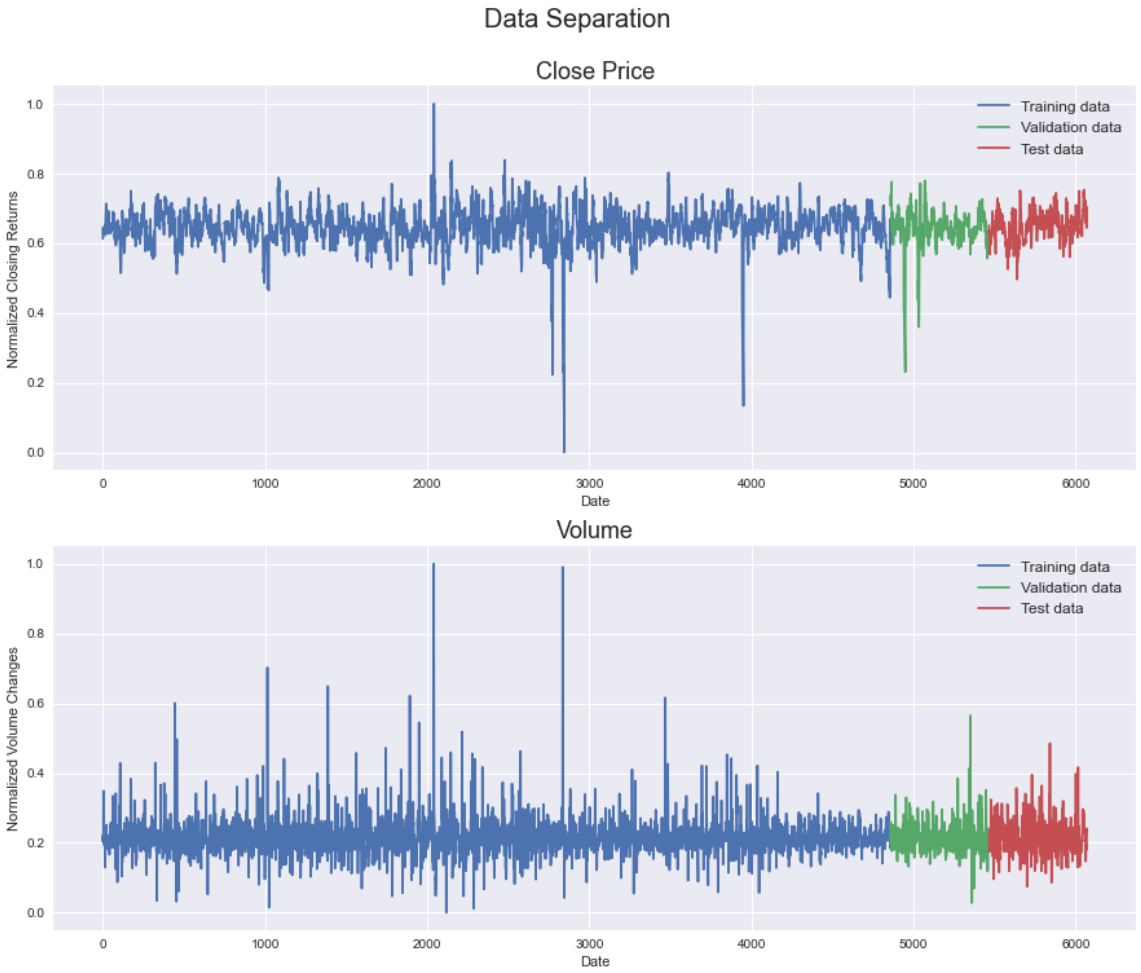
Σύνολο Δεδομένων	Διαστάσεις
Train Set	(4857,5)
Validation Set	(607,5)
Test Set	(607,5)

Όπως γίνεται αντιληπτό από τον παραπάνω πίνακα, το 80% του συνόλου των δεδομένων χρησιμοποιήθηκε για εκπαίδευση, το 10% για επικύρωση και το υπόλοιπο 10% για έλεγχο.

Στο σημείο αυτό, αξίζει να σημειωθεί ότι δοκιμάστηκαν και άλλες αναλογίες ποσοστών, όμως η συγκεκριμένη που παραθέσαμε παραπάνω μας έδωσε τα καλύτερα δυνατά αποτελέσματα ως προς τις μετρικές που χρησιμοποιήσαμε για την αξιολόγηση του μοντέλου.

4.5 Οπτικοποίηση Κανονικοποιημένων Συνόλων Δεδομένων

Στην συνέχεια, παρατίθενται αποτελέσματα, τα οποία προέκυψαν από την οπτικοποίηση του Close Price και του Volume όσο αφορά τα τρία σύνολα δεδομένων:



Διάγραμμα 4.3 Τα Close Price και του Volume για τα τρία σύνολα δεδομένων

4.6 Μοντέλο Transformer

Στο παραπάνω τμήμα κώδικα, το οποίο χρησιμοποιήθηκε για ην υλοποίηση των λειτουργιών του Transformer μοντέλου, χρησιμοποιούνται 3 βασικές κλάσεις:

- Κλάση **Time2Vector**: Χρησιμοποιείται για την δημιουργία του Time2Vectorεπιπέδου του νευρωνικού δικτύου.
- Κλάση **SingleAttention**: Χρησιμοποιείται για την δημιουργία του SingleAttention επιπέδου του νευρωνικού δικτύου.
- Κλάση **MultiAttention**: Χρησιμοποιείται για την δημιουργία του MultiAttention επιπέδου του νευρωνικού δικτύου..

```
class Time2Vector(Layer):
 def __init__(self, seq_len, **kwargs):
  super(Time2Vector, self).__init__()

  self.seq_len = seq_len

 def build(self, input_shape):
  '''Initialize weights and biases with shape (batch, seq_len)'''
  self.weights_linear = self.add_weight(name='weight_linear',
                        shape=(int(self.seq_len),),
                        initializer='uniform',
                        trainable=True)

  self.bias_linear = self.add_weight(name='bias_linear',
                        shape=(int(self.seq_len),),
                        initializer='uniform',
                        trainable=True)

  self.weights_periodic = self.add_weight(name='weight_periodic',
                        shape=(int(self.seq_len),),
                        initializer='uniform',
                        trainable=True)

  self.bias_periodic = self.add_weight(name='bias_periodic',
                        shape=(int(self.seq_len),),
                        initializer='uniform',
```

```
                        trainable=True)


  def call(self, x):
    '''Calculate linear and periodic time features'''
    x = tf.math.reduce_mean(x[:,:,:4], axis=-1)
    time_linear = self.weights_linear * x + self.bias_linear # Linear time feature
    time_linear = tf.expand_dims(time_linear, axis=-1) # Add dimension (batch, seq_len, 1)

    time_periodic = tf.math.sin(tf.multiply(x, self.weights_periodic) + self.bias_periodic)
    time_periodic = tf.expand_dims(time_periodic, axis=-1) # Add dimension (batch, seq_len, 1)
    return tf.concat([time_linear, time_periodic], axis=-1) # shape = (batch, seq_len, 2)

  def get_config(self): # Needed for saving and loading model with custom layer
    config = super().get_config().copy()
    config.update({'seq_len': self.seq_len})
    return config




class SingleAttention(Layer):
  def __init__(self, d_k, d_v):
    super(SingleAttention, self).__init__()
    self.d_k = d_k
    self.d_v = d_v

  def build(self, input_shape):
```

```
  self.query = Dense(self.d_k,
                     input_shape=input_shape,
                     kernel_initializer='glorot_uniform',
                     bias_initializer='glorot_uniform')

  self.key = Dense(self.d_k,
                   input_shape=input_shape,
                   kernel_initializer='glorot_uniform',
                   bias_initializer='glorot_uniform')

  self.value = Dense(self.d_v,
                     input_shape=input_shape,
                     kernel_initializer='glorot_uniform',
                     bias_initializer='glorot_uniform')

def call(self, inputs): # inputs = (in_seq, in_seq, in_seq)
  q = self.query(inputs[0])
  k = self.key(inputs[1])

  61ttn._weights = tf.matmul(q, k, transpose_b=True)
  61ttn._weights = tf.map_fn(lambda x: x/np.sqrt(self.d_k), 61ttn._weights)
  61ttn._weights = tf.nn.softmax(61ttn._weights, axis=-1)

  v = self.value(inputs[2])
  61ttn._out = tf.matmul(61ttn._weights, v)
  return 61ttn._out
```

```
class MultiAttention(Layer):
  def __init__(self, d_k, d_v, n_heads):
    super(MultiAttention, self).__init__()
    self.d_k = d_k
    self.d_v = d_v
    self.n_heads = n_heads
    self.attn_heads = list()

  def build(self, input_shape):
    for n in range(self.n_heads):
      self.attn_heads.append(SingleAttention(self.d_k, self.d_v))

    # input_shape[0]=(batch, seq_len, 7), input_shape[0][-1]=7
    self.linear = Dense(input_shape[0][-1],
                        input_shape=input_shape,
                        kernel_initializer='glorot_uniform',
                        bias_initializer='glorot_uniform')

  def call(self, inputs):
    62ttn. = [self.attn_heads[i](inputs) for I in range(self.n_heads)]
    concat_attn = tf.concat(62ttn., axis=-1)
    multi_linear = self.linear(concat_attn)
    return multi_linear
```

```
class TransformerEncoder(Layer):
 def __init__(self, d_k, d_v, n_heads, ff_dim, dropout=0.1, **kwargs):
  super(TransformerEncoder, self).__init__()
  self.d_k = d_k
  self.d_v = d_v
  self.n_heads = n_heads
  self.ff_dim = ff_dim
  self.attn_heads = list()
  self.dropout_rate = dropout

 def build(self, input_shape):
  self.attn_multi = MultiAttention(self.d_k, self.d_v, self.n_heads)
  self.attn_dropout = Dropout(self.dropout_rate)
  self.attn_normalize = LayerNormalization(input_shape=input_shape, epsilon=1e-6)

  self.ff_conv1D_1 = Conv1D(filters=self.ff_dim, kernel_size=1, activation='relu')
  # input_shape[0]=(batch, seq_len, 7), input_shape[0][-1] = 7
  self.ff_conv1D_2 = Conv1D(filters=input_shape[0][-1], kernel_size=1)
  self.ff_dropout = Dropout(self.dropout_rate)
  self.ff_normalize = LayerNormalization(input_shape=input_shape, epsilon=1e-6)

 def call(self, inputs): # inputs = (in_seq, in_seq, in_seq)
  63ttn._layer = self.attn_multi(inputs)
  63ttn._layer = self.attn_dropout(63ttn._layer)
```

```
  64ttn._layer = self.attn_normalize(inputs[0] + 64ttn._layer)


  ff_layer = self.ff_conv1D_1(64ttn._layer)
  ff_layer = self.ff_conv1D_2(ff_layer)
  ff_layer = self.ff_dropout(ff_layer)
  ff_layer = self.ff_normalize(inputs[0] + ff_layer)
  return ff_layer


 def get_config(self): # Needed for saving and loading model with custom layer
  config = super().get_config().copy()
  config.update({'d_k': self.d_k,
                 'd_v': self.d_v,
                 'n_heads': self.n_heads,
                 'ff_dim': self.ff_dim,
                 'attn_heads': self.attn_heads,
                 'dropout_rate': self.dropout_rate})
  return config
```

Κώδικας 4

4.7 Δημιουργία Μοντέλου

Ακολουθεί ο κώδικας για την δημιουργία του μοντέλου. Όπως φαίνεται και από τον παρακάτω κώδικα, χρησιμοποιείται ένα time layer και τρία attention layers. Στο σημείο αυτό αξίζει να αναφέρουμε ότι το χαρακτηριστικό το οποίο θα έπρεπε να προβλεφθεί είναι το Closing Return.

```
def create_model():
  '''Initialize time and transformer layers'''
  time_embedding = Time2Vector(seq_len)
  attn_layer1 = TransformerEncoder(d_k, d_v, n_heads, ff_dim)
```

```
  attn_layer2 = TransformerEncoder(d_k, d_v, n_heads, ff_dim)
  attn_layer3 = TransformerEncoder(d_k, d_v, n_heads, ff_dim)

  '''Construct model'''
  in_seq = Input(shape=(seq_len, 5))
  x = time_embedding(in_seq)
  x = Concatenate(axis=-1)([in_seq, x])
  x = attn_layer1((x, x, x))
  x = attn_layer2((x, x, x))
  x = attn_layer3((x, x, x))
  x = GlobalAveragePooling1D(data_format='channels_first')(x)
  x = Dropout(0.1)(x)
  x = Dense(64, activation='relu')(x)
  x = Dropout(0.1)(x)
  out = Dense(1, activation='linear')(x)

  model = Model(inputs=in_seq, outputs=out)
  model.compile(loss='mse', optimizer='adam', metrics=['mae', 'mape'])
return model
```

4.8 Εκπαίδευση μοντέλου

Ακολουθεί το μέρος της πειραματικής διαδικασίας, το οποίο απαίτησε το μεγαλύτερο χρονικό διάστημα για την εκτέλεσή του. Χαρακτηριστικά, θα αναφέρουμε ότι στην πειραματική μας διαδικασία, η διαδικασία εκπαίδευσης πραγματοποιήθηκε σε 20 epochs και με batch_size = 16.

Για μία τέτοια διαδικασία, στο υπολογιστικό σύστημα, στο οποίο πραγματοποιήθηκε η εκπαίδευση, απαιτήθηκαν περίπου 9 ώρες. Επομένως, συμπεραίνουμε ότι για περισσότερες δοκιμές τόσο όσο

αφορά τον συνδυασμό των παραμέτρων αλλά και σε διαφορετικά σύνολα δεδομένων (πχ διαφορετικές μετοχές), θα απαιτούνται πανίσχυρα υπολογιστικά συστήματα.

```
Model = create_model()
model.summary()


callback = tf.keras.callbacks.ModelCheckpoint('Transformer+TimeEmbedding.hdf5',
                                monitor='val_loss',
                                save_best_only=True,
                                verbose=1)


history = model.fit(X_train, y_train,
            batch_size=batch_size,
            epochs=20,
            callbacks=[callback],
            validation_data=(X_val, y_val))


model = tf.keras.models.load_model('Transformer+TimeEmbedding.hdf5',
                        custom_objects={'Time2Vector': Time2Vector,
                                        'SingleAttention': SingleAttention,
                                        'MultiAttention': MultiAttention,
                                        'TransformerEncoder': TransformerEncoder})


'''Calculate predictions and metrics'''


#Calculate predication for training, validation and test data
train_pred = model.predict(X_train)
```

```
val_pred = model.predict(X_val)

test_pred = model.predict(X_test)



#Print evaluation metrics for all datasets

train_eval = model.evaluate(X_train, y_train, verbose=0)

val_eval = model.evaluate(X_val, y_val, verbose=0)

test_eval = model.evaluate(X_test, y_test, verbose=0)

print(' ')

print('Evaluation metrics')

print('Training Data – Loss: {:.4f}, MAE: {:.4f}, MAPE: {:.4f}'.format(train_eval[0], train_eval[1], train_eval[2]))

print('Validation Data – Loss: {:.4f}, MAE: {:.4f}, MAPE: {:.4f}'.format(val_eval[0], val_eval[1], val_eval[2]))

print('Test Data – Loss: {:.4f}, MAE: {:.4f}, MAPE: {:.4f}'.format(test_eval[0], test_eval[1], test_eval[2]))



fig = plt.figure(figsize=(15,20))

st = fig.suptitle("Moving Average – Transformer + TimeEmbedding Model", fontsize=22)

st.set_y(0.92)
```

Κώδικας 5

Στην συνέχεια, παραθέτουμε ενδεικτικό screen-shot, το οποίο προέκυψε κατά την διάρκεια πραγματοποίησης της πειραματικής διαδικασίας εκπαίδευσης:

```
Epoch 1/20
296/296 [==============================] - 611s 2s/step - loss: 0.0752 - mae: 0.1743 - mape: 27.5331 - val_loss: 0.0037 - val_m
ae: 0.0467 - val_mape: 7.8165

Epoch 00001: val_loss improved from inf to 0.00375, saving model to Transformer+TimeEmbedding.hdf5
Epoch 2/20
296/296 [==============================] - 507s 2s/step - loss: 0.0062 - mae: 0.0594 - mape: 38496.7483 - val_loss: 0.0022 - va
l_mae: 0.0321 - val_mape: 5.3327

Epoch 00002: val_loss improved from 0.00375 to 0.00222, saving model to Transformer+TimeEmbedding.hdf5
Epoch 3/20
296/296 [==============================] - 514s 2s/step - loss: 0.0051 - mae: 0.0536 - mape: 155655.2631 - val_loss: 0.0022 - v
al_mae: 0.0319 - val_mape: 5.2306

Epoch 00003: val_loss improved from 0.00222 to 0.00218, saving model to Transformer+TimeEmbedding.hdf5
Epoch 4/20
296/296 [==============================] - 514s 2s/step - loss: 0.0053 - mae: 0.0526 - mape: 532042.6669 - val_loss: 0.0025 - v
al_mae: 0.0346 - val_mape: 5.8016
```

Εικόνα 4.1

Παραθέτουμε και ένα ενδεικτικό screen-shot, στο οποίο συνοψίζονται οι πληροφορίες του μοντέλου Transformer, το οποίο χρησιμοποιήθηκε:

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 128, 5)]     0
__________________________________________________________________________________________________
time2_vector (Time2Vector)      (None, 128, 2)       512         input_1[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 128, 7)       0           input_1[0][0]
                                                                 time2_vector[0][0]
__________________________________________________________________________________________________
transformer_encoder (Transforme (None, 128, 7)       99114       concatenate[0][0]
                                                                 concatenate[0][0]
                                                                 concatenate[0][0]
__________________________________________________________________________________________________
transformer_encoder_1 (Transfor (None, 128, 7)       99114       transformer_encoder[0][0]
                                                                 transformer_encoder[0][0]
                                                                 transformer_encoder[0][0]
__________________________________________________________________________________________________
transformer_encoder_2 (Transfor (None, 128, 7)       99114       transformer_encoder_1[0][0]
                                                                 transformer_encoder_1[0][0]
                                                                 transformer_encoder_1[0][0]
__________________________________________________________________________________________________
global_average_pooling1d (Globa (None, 128)          0           transformer_encoder_2[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 128)          0           global_average_pooling1d[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 64)           8256        dropout[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 64)           0           dense[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 1)            65          dropout_1[0][0]
==================================================================================================
Total params: 306,175
Trainable params: 306,175
Non-trainable params: 0
```

Εικόνα 4.2: Πληροφορίες του χρησιμοποιούμενου μοντέλου Transformer

4.9 Αποτελέσματα Αξιολόγησης Μοντέλου Transformer

Τέλος, παρατίθενται τα τελικά αποτελέσματα, τα οποία προέκυψαν για την αξιολόγηση του μοντέλου Transformer. Θυμίζουμε ότι το μοντέλο αξιολογήθηκε ως προς τις παρακάτω μετρικές:

- Loss
- MAE (Mean Absolute Error)
- MAPE (Mean Absolute Percentage Error)

Όσο αφορά την μετρική Loss, ελήφθησαν τα παρακάτω αποτελέσματα:

Σύνολο Δεδομένων	Loss
Train Set	0.0008
Validation Set	0.0006
Test Set	0.0005

Όσο αφορά την μετρική MAE, ελήφθησαν τα παρακάτω αποτελέσματα:

Σύνολο Δεδομένων	MAE
Train Set	0.0202
Validation Set	0.0171
Test Set	0.0179

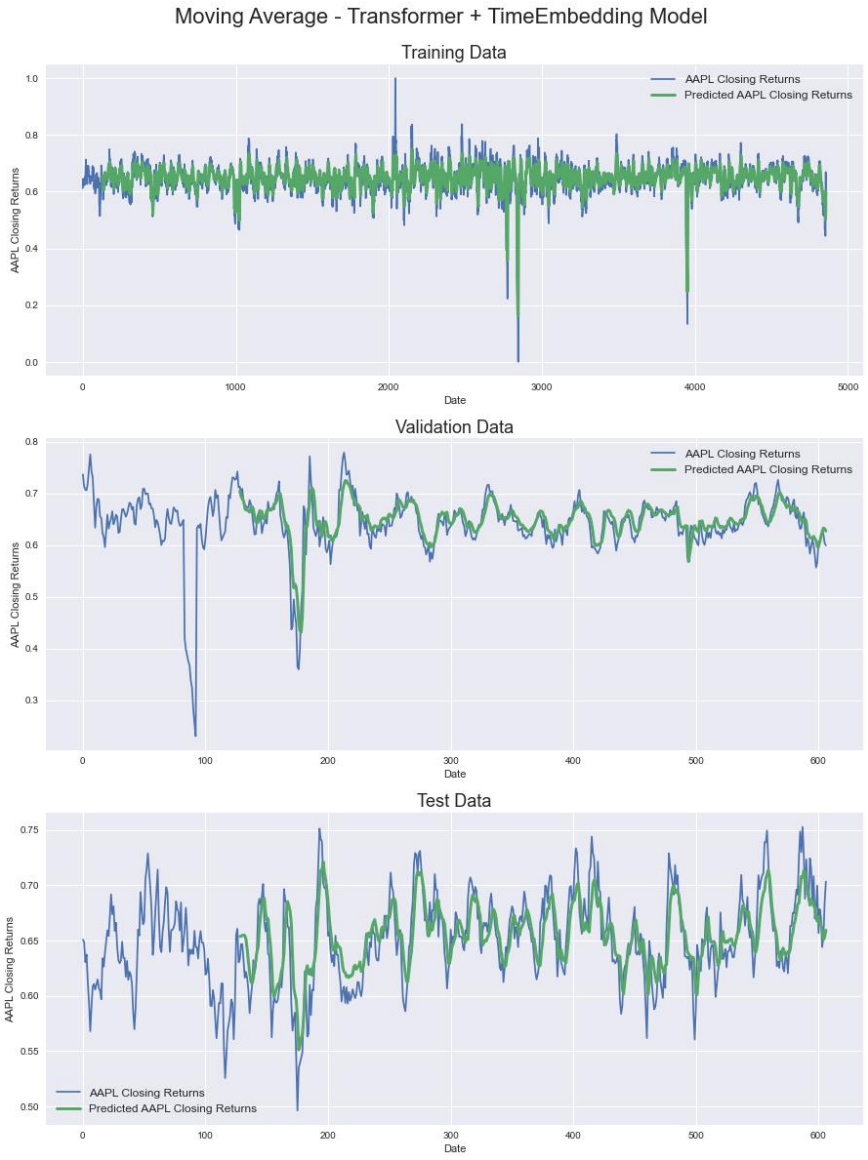
Τέλος, όσο αφορά την μετρική MAPE, ελήφθησαν τα παρακάτω αποτελέσματα:

Σύνολο Δεδομένων	MAPE
Train Set	34479.2266
Validation Set	2.8001
Test Set	2.7650

Από τα παραπάνω διαπιστώνουμε ότι τα αποτελέσματα, τα οποία λάβαμε ως προς τιμές και των 3 μετρικών και όσο αφορά και τα 3 σύνολα δεδομένων, ήταν άκρως ικανοποιητικά (με εξαίρεση την περίπτωση της μετρικής MAPEγια την περίπτωση του Train Set).

Άποψή μας είναι ότι χρήση ακόμα μεγαλύτερου batch size (για παράδειγμα 32) θα οδηγούσε σε ακόμη καλύτερα αποτελέσματα. Χρήση όμως μεγαλύτερου batch size, καθυστερούσε ακόμη περισσότερο την πειραματική διαδικασία, καθιστώντας απαγορευτικούς τους αντίστοιχους χρόνους.

Οι ισχυρισμοί μας με βάση τα παραπάνω αποτελέσματα, επιβεβαιώνονται και από τις παρακάτω γραφικές παραστάσεις, οι οποίες δείχνουν μεγάλου βαθμού ταύτιση μεταξύ των πραγματικών και των προβλεπόμενων τιμών όσο αφορά το Closing Return (που αποτελούσε και το χαρακτηριστικό, το οποίο θα έπρεπε να προβλεφθεί):



Διάγραμμα 4.4: Τα αποτελέσματα του μοντέλου

4.10 Μοντέλο LSTM

Όπως αναφέρθηκε και στην αρχή του κεφαλαίου για την παροχή συγκριτικών αποτελεσμάτων, δημιουργήθηκε μοντέλο LSTM, το οποίο επίσης χρησιμοποιείται με σκοπό την πρόβλεψη στην περίπτωση της μετοχής APPLE. Ακολουθεί ο κώδικας, ο οποίος υλοποιήθηκε στην συγκεκριμένη περίπτωση:

```
import math

import numpy as np

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, LSTM

import matplotlib.pyplot as plt


df = pd.read_csv('AAPL.csv', delimiter=',', usecols=['Date', 'Open', 'High', 'Low', 'Close', 'Volume'])

data = df.filter(['Close'])

dataset = data.values

training_data_len = math.ceil(len(dataset) * .8)


scaler = MinMaxScaler(feature_range=(0,1))

scaled_data = scaler.fit_transform(dataset)


train_data = scaled_data[0:training_data_len, : ]

x_train = []

y_train = []

for i in range(60 , len(train_data)):

  x_train.append(train_data[i-60:i, ])

  y_train.append(train_data[i, 0])
```

```
x_train , y_train = np.array(x_train), np.array(y_train)

x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))


model = Sequential()

model.add(LSTM(50, return_sequences =True, input_shape = (x_train.shape[1], 1)))

model.add(LSTM(50, return_sequences = False))

model.add(Dense(25))

model.add(Dense(1))

model.compile(optimizer = 'adam', loss ='mean_squared_error')


model.fit(x_train, y_train, batch_size=1, epochs=1)


test_data = scaled_data[training_data_len - 60: , : ]

x_test = []

y_test = dataset[training_data_len: , : ]


for i in range(60, len(test_data)):

  x_test.append(test_data[i-60 :  i, 0])

x_test = np.array(x_test)

x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))


predictions = model.predict(x_test)

predictions = scaler.inverse_transform(predictions)
```

Κώδικας 6

4.11 Αποτελέσματα Αξιολόγησης Μοντέλου LSTM

Όπως και το μοντέλο Transformer, έτσι και το μοντέλο LSTM αξιολογήθηκε ως προς τις παρακάτω μετρικές:

- Loss
- MAE
- MAPE

Ακολουθεί ο αντίστοιχος κώδικας, που υλοποιήθηκε:

```
def mae(act, pred):
    diff = pred - act
    abs_diff = np.absolute(diff)
    mean_diff = abs_diff.mean()
    return mean_diff


def mape(actual, pred):
    actual, pred = np.array(actual), np.array(pred)
    return np.mean(np.abs((actual - pred) / actual)) * 100




from sklearn.metrics import mean_squared_error


loss = mean_squared_error(y_test, predictions, squared=False)


print("Loss : " + str(loss))
print("MAE : "+ str(mae(y_test,predictions)))
print("MAPE : " + str(mape(y_test,predictions)))
```

Κώδικας 7

Και τα αντίστοιχα αποτελέσματα των μετρικών όσο αφορά το test set:

Μετρική	Τιμή
Loss	2.2176
MAE	1.6313
MAPE	4.9857

Αυτό λοιπόν, το οποίο παρατηρούμε, πραγματοποιώντας σύγκριση μεταξύ των δύο μοντέλων, είναι ότι όσο αφορά και τις τρεις μετρικές, στην περίπτωση του μοντέλου Transformer λάβαμε πολύ καλύτερα αποτελέσματα. Ακολουθεί ο παρακάτω συγκριτικός πίνακας για την διαπίστωση των λεγομένων μας:

Μετρική	Τιμή (LSTM)	Τιμή (Transformer)
Loss	2.2176	0.0005
MAE	1.6313	0.0179
MAPE	4.9857	2.7650

Επομένως, το μοντέλο Transformer συμπεριφέρεται καλύτερα, και άρα θα αποτελέσει την βάση για την συνέχεια της πειραματικής διαδικασίας (υλοποίηση trading bot).

4.12 Δημιουργία Σημάτων Αγοράς – Πώλησης - Διατήρησης

Οι τιμές που έχουν προβλεφθεί από την παραπάνω διαδικασία, μπορούν να αξιοποιηθούν από την συνάρτηση suggest_actions(), η οποία υλοποιήθηκε και η οποία λαμβάνει τις παρακάτω εισόδους:

- predictions: προβλέψεις
- interval: χρονικό παράθυρο
- threshold: κατώφλι μεταβολής

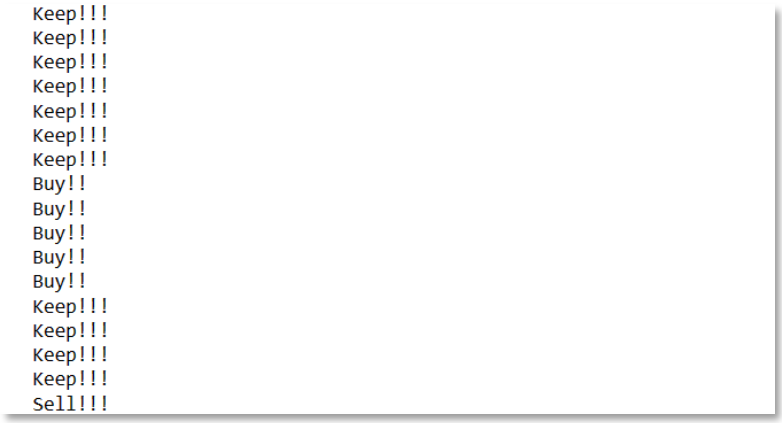
```
def suggest_actions(predictions,interval, threshold):
    num_keeps=0
    num_buys=0
```

```
num_sells = 0


for i in range(len(predictions)-interval):
    dif_percent = ((predictions[i+interval] - predictions[i]) / predictions[i])*100


    if (dif_percent >=threshold):
        num_buys = num_buys+1
        print("Buy!!")


    elif (-dif_percent >=threshold):
        num_sells = num_sells+1
        print("Sell!!!")


    else:
        num_keeps = num_keeps+1
        print("Keep!!!")


print("Total number of buy suggestions:"+str(num_buys))
print("Total number of sell suggestions:"+str(num_sells))
print("Total number of keep suggestions:"+str(num_keeps))
```

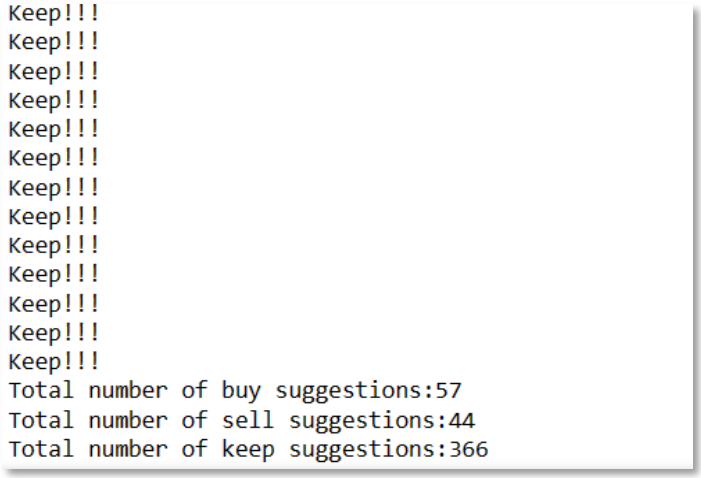
Κώδικας 8

Η συνάρτηση αυτή ελέγχει με βάση τις τιμές που έχουν προβλεφθεί και τις αντίστοιχες παραμέτρους που έχουν δοθεί σε κάθε χρονική στιγμή (ημέρα), εάν προτείνεται στον επενδυτή να αγοράσει, να διατηρήσει ή να πουλήσει τις μετοχές του. Για παράδειγμα εάν δοθεί σαν είσοδος time_interval = 10 και threshold = 5, ελέγχεται για κάθε χρονική στιγμή η τρέχουσα προβλεφθείσα τιμή και η αντίστοιχη προβλεφθείσα τιμή για μετά από 10 ημέρες. Αν παρατηρείται άνοδος από 5% και πάνω, τότε προτείνεται αγορά, ενώ αν παρατηρείται πτώση από το 5% και πάνω, προτείνεται

πώληση. Σε διαφορετική περίπτωση προτείνεται διατήρηση της μετοχής. Παρατίθενται χαρακτηριστικά screen-shot κατά την έναρξη και λήξη της εκτέλεσης:



Εικόνα 4.3: Τα αποτελέσματα της συνάρτησης suggest_actions() κατά την έναρξη της εκτέλεσης



Εικόνα 4.4: Τα αποτελέσματα της συνάρτησης suggest_actions() κατά την λήξη της εκτέλεσης

Στο τέλος, εμφανίζεται και το συνολικό πλήθος προτάσεων διατήρησης, αγοράς και πώλησης της μετοχής.

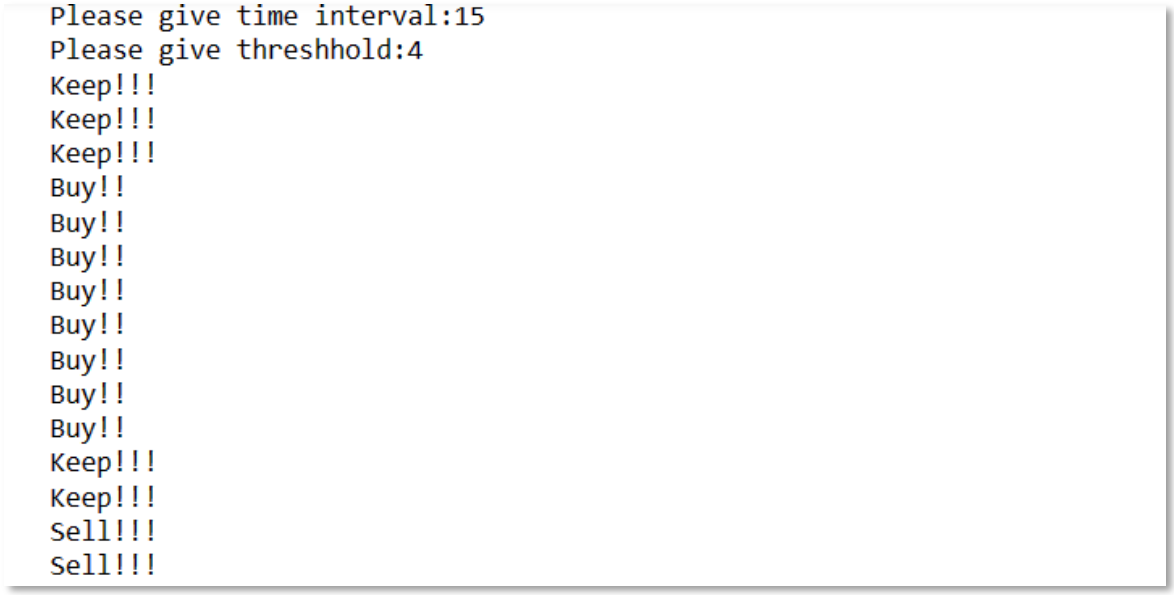
4.13 Δημιουργία Trading Bot

Τέλος, δημιουργήθηκε η παρακάτω συνάρτηση για την υλοποίηση της λειτουργίας του trading bot, η οποία αξιοποιεί την παραπάνω συνάρτηση και λαμβάνει τις τιμές των παραμέτρων από τον χρήστη:

```
def trading_bot(predictions):


   time_interval=int(input("Please give time interval:"))


   threshold=float(input("Please give threshhold:"))


   suggest_actions(predictions,time_interval,threshold)
```

Κώδικας 9

Ακολουθεί, ενδεικτικό screen-shot λειτουργίας του trading bot:



Εικόνα 4.5: Η λειτουργία του trading bot

Chapter 5

Συμπεράσματα, Μελλοντική Εξέλιξη και Δυσκολίες

5.1 Συμπεράσματα

Σε αυτήν την εργασία προσεγγίστηκε η δυνατότητα δημιουργίας προβλέψεων για χρηματιστηριακές τιμές με την βοήθεια ενός μοντέλου Transformer. Τα αποτελέσματα της σύγκρισης με ένα τυπικό μοντέλο LSTM Νευρωνικού Δικτύου, επιβεβαιώνουν την αρχική πεποίθηση πως το Transformer μοντέλο είναι αρκετά πιο ακριβές στις προβλέψεις, έχοντας καλύτερες επιδόσεις όσον αφορά τις μετρικές LOSS MAE και MAPE. Αυτό με τη σειρά του συνεπάγεται πως το Transformer μοντέλο είναι πιο αποδοτικό στην αντίληψη των πολύπλοκων μοτίβων που χαρακτηρίζουν τις μετοχές συγκριτικά με ένα LSTM μοντέλο.

Επιπρόσθετα, δημιουργήθηκε και ένα σύστημα Trading Bot το οποίο βασίστηκε στο προαναφερθέν Transformer μοντέλο. Αυτό με τη σειρά του αξιολόγησε και επιβεβαίωσε την αποτελεσματικότητα του μοντέλου σε πραγματικές συνθήκες. Έτσι μπορούμε να συμπεράνουμε πως το μοντέλο έχει τη δυνατότητα να χρησιμοποιηθεί από τους επενδυτές για τις καθημερινές προβλέψεις του χαρτοφυλακίου τους καθώς εγγυάται αρκετά καλές αποδώσεις.

5.2 Μελλοντική Εξέλιξη

Παρόλο που τα αποτελέσματα της εργασίας είναι πολλά υποσχόμενα, υπάρχουν ακόμα αρκετά σημεία τα οποία απαιτούν επιπλέον έρευνα τα οποία παρουσιάζονται παρακάτω.

- Ανθεκτικότητα: Το Trading Bot που αναπτύχθηκε σε αυτή την εργασία βασίζεται σε ιστορικά δεδομένα και η απόδοσή του μπορεί να διαφέρει σε διαφορετικές συνθήκες της αγοράς. Ως εκ τούτου, είναι απαραίτητο να αξιολογηθεί η ανθεκτικότητα του μοντέλου σε διαφορετικές συνθήκες αγοράς, όπως οι αγορές Bull and Bear.

- Επεξηγησιμότητα: Το μοντέλο Transformer είναι ένα black box μοντέλο και είναι δύσκολο να ερμηνεύσουμε τις προβλέψεις του. Η μελλοντική έρευνα μπορεί να επικεντρωθεί στην ανάπτυξη μεθόδων για να καταστεί το μοντέλο πιο εξηγήσιμο.

- Συμπίεση του Μοντέλου: Το μοντέλο μπορεί να γίνει πιο αποδοτικό και η χρήση του να απαιτεί λιγότερους υπολογιστικούς πόρους. Με αυτόν τον τρόπο θα καταστεί πιο εύκολη η χρήση του από το ευρύ κοινό χωρίς να υπάρχουν απαιτήσεις ισχυρών υπολογιστικών συστημάτων.

Σαν συμπέρασμα, μπορούμε να πούμε πως η παρούσα εργασία απέδειξε την υπεροχή του μοντέλου Transformer έναντι του LSTM σε συνθήκες χρηματιστηριακών προβλέψεων. Επιπλέον, το Trading Bot που βασίζεται στο μοντέλο Transformer, έδειξε πολλά υποσχόμενα αποτελέσματα στη παροχή σωστών προτάσεων αγοράς και πώλησης. Ωστόσο, εξακολουθούν να υπάρχουν αρκετοί τομείς που απαιτούν περαιτέρω έρευνα για τη βελτίωση της απόδοσης και της δυνατότητας εφαρμογής του μοντέλου σε σενάρια πραγματικού κόσμου. Ελπίζουμε ότι η μελέτη αυτή θα εμπνεύσει μελλοντική έρευνα σε αυτόν τον τομέα και θα συμβάλει στην ανάπτυξη πιο ακριβών και αξιόπιστων μοντέλων πρόβλεψης για το χρηματιστήριο.

5.3 Δυσκολίες

Κατά τη διάρκεια της παρούσας μεταπτυχιακής διατριβής, αντιμετωπίστηκαν αρκετές δυσκολίες που επιβράδυναν την ερευνητική πρόοδο. Μία από τις κύριες προκλήσεις ήταν η διαθεσιμότητα ποιοτικών δεδομένων. Τα στοιχεία της χρηματιστηριακής αγοράς είναι εξαιρετικά ασταθή και υπόκεινται σε διάφορους εξωτερικούς παράγοντες, όπως το new sentiment και οι οικονομικοί δείκτες. Ως εκ τούτου, ήταν απαραίτητο να διασφαλιστεί ότι τα δεδομένα που χρησιμοποιήθηκαν για την εκπαίδευση των μοντέλων ήταν υψηλής ποιότητας και αντιπροσωπευτικά της συμπεριφοράς του χρηματιστηρίου. Επιπλέον, η εκπαίδευση του μοντέλου Transformer ήταν υπολογιστικά ακριβή και απαιτούσε σημαντική ποσότητα μνήμης, γεγονός που έθετε προκλήσεις στην ανάπτυξη και βελτιστοποίηση του μοντέλου. Τέλος, η ανάπτυξη του Trading Bot απαιτούσε ολοκληρωμένη κατανόηση των στρατηγικών συναλλαγών στο χρηματιστήριο και των αρχών διαχείρισης κινδύνου, κάτι που απαιτούσε σημαντική προσπάθεια και έρευνα. Οι προκλήσεις αυτές ξεπεράστηκαν με μεθοδικότητα και επιμονή, παρουσιάζοντας πολλά υποσχόμενα αποτελέσματα, τα οποία μπορούν να συμβάλουν στην ανάπτυξη πιο ακριβών και αξιόπιστων μοντέλων πρόβλεψης για το χρηματιστήριο.

Βιβλιογραφία

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Brownlee, J. (2019). Time series forecasting with the long short-term memory network in Python. Machine Learning Mastery.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning (Vol. 1). MIT press.

Huang, Z., Li, Z., & Chen, H. (2014). Stock trend prediction by using back propagation neural network based on imperialist competitive algorithm. In International Conference on Neural Information Processing (pp. 413-420). Springer.

Shang, Y., Sun, Y., & Liu, J. (2019). Stock price prediction based on LSTM model. IEEE Access, 7, 119074-119083.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814).

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR).

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of econometrics, 31(3), 307-327.

Zou, Y., & Shi, Y. (2019). Short-term stock price forecasting based on LSTM and GRU neural networks. Journal of Intelligent & Fuzzy Systems, 37(4), 5651-5659.

Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. Scientific reports, 8(1), 6085.

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159-175.

Jiang, Y., Chen, C., & Feng, X. (2017). Stock price trend forecast based on LSTM. In 2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC) (pp. 223-226). IEEE.

Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. O'Reilly Media, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice.

Brown, A. G., & Martínez, M. L. (2018). Deep learning for time series forecasting: A review. Journal of business research, 88, 423-434.

Lipton, Z. C., Kale, D. C., & Wetzel, R. (2018). Learning to diagnose with LSTM recurrent neural networks. arXiv preprint arXiv:1511.03677.

Vargas, J. F., & López, L. A. (2018). Financial time series forecasting with deep learning: A systematic literature review. Expert Systems with Applications, 109, 1-16.

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271.

Guo, S., Li, Y., Zhang, Z., & Li, H. (2019). Stock prediction based on social media sentiment analysis: A review. IEEE Access, 7, 66786-66799.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02), 107-116.

Lashgari, A., Shahmoradi, S., & Amirian, P. (2020). Stock price prediction by integrating sentiment analysis and time series forecasting. Expert Systems with Applications, 153, 113434.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157-166.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one, 12(7), e0180944.