ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

Διπλωματική Εργασία

Εφαρμογές βαθιάς μάθησης

Φοιτητής: Λέοναρντ Τσερρίκου
Αριθμός Μητρώου: 44713

Επιβλέπων: Λέκτορας Γρηγόριος Νικολάου

ΑΘΗΝΑ- ΑΙΓΑΛΕΩ, ΟΚΤΩΒΡΙΟΣ 2021

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF INDUSTRIAL DESIGN AND PRODUCTION ENGINEERING

Diploma Thesis

Applications of deep learning

Student: Leonard Cerriku

Registration Number: 44713

Supervisor: Lecturer Grigorios Nikolaou

ATHENS-EGALEO, OCTOBER 2021

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Γ.Νικολάου , Λέκτορας	Σ. Βασιλειάδου, Επίκουρη Καθηγήτρια	Χ. Δρόσος, ΕΔΙΠ
GRIGORIOS NIKOLAOU Digitally signed by GRIGORIOS NIKOLAOU Date: 2021.12.02 12:55:37 +02'00' (Υπογραφή)	Soultana Vasileiadou Digitally signed by Soultana Vasileiadou Date: 2021.12.02 15:24:45 +02'00' (Υπογραφή)	Christos Drosos Digitally signed by Christos Drosos Date: 2021.12.02 12:18:09 +02'00' (Υπογραφή)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Λέοναρντ του Άρμπεν με αριθμό μητρώου 44713 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.

Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι …… και έπειτα από αίτησή μου στη Βιβλιοθήκη και έγκριση του επιβλέποντος καθηγητή.»

Ο Δηλών

Λέοναρντ Τσερρίκου

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ θερμά τον υπεύθυνο καθηγητή μου Γρηγόριο Νικολάου, λέκτορα του Τμήματος Βιομηχανικής Σχεδίασης & Παραγωγής του Πανεπιστημίου Δυτικής Αττικής, που μου ανέθεσε και επέβλεψε μέχρι τέλους την εκπόνηση της παρούσας διπλωματικής εργασίας. Τον ευχαριστώ, ακόμη, που πίστεψε στο έργο μου αλλά και στην πολύτιμη βοήθεια που έλαβα από εκείνον για την υλοποίηση των πειραματικών διαδικασιών, καθώς και για τη συγγραφής της εργασίας.

Εν συνεχεία, ιδιαίτερες ευχαριστίες στη Δάφνη Βρετουδάκη που βοήθησε με τη σειρά της στην συγγραφή της εν λόγω διπλωματικής εργασίας.

Τέλος, αφιερώνω το παρόν πόνημα στη μητέρα μου και τους φίλους μου, που με στηρίζουν και με ενθαρρύνουν καθημερινά σε κάθε μου προσπάθεια.

ΠΕΡΙΛΗΨΗ

Στην παρούσα διπλωματική εργασία παρουσιάζονται και εφαρμόζονται παραδείγματα χρήσης αλγορίθμων και μέθοδοι εύρεσης σφαλμάτων όπως DBSCAN, LOF, Isolation Forest και Tukey IQR, χρησιμοποιώντας συνθετικά δεδομένα. Επιπλέον, αναφέρονται παραδείγματα αντικατάστασης ελλιπών τιμών σε δεδομένα θερμοκρασίας. Μετέπειτα, παρουσιάζονται παραδείγματα χρήσης βιβλιοθηκών βαθιάς μάθησης Estimator API, Tensorflow και βιβλιοθήκης Keras για την δημιουργία μοντέλων πρόβλεψης που αφορούν την επίλυση προβλημάτων σε τραπεζικά δεδομένα, ηλεκτρική κατανάλωση, δεδομένα θερμοκρασίας, πολυταξικός ταξινομητής για δεδομένα αρίθμησης [0-9] στην νοηματική γλώσσα. Τέλος, παρουσιάζεται η πειραματική εφαρμογή μοντέλου πρόβλεψης γωνίας τιμονιού για την οδήγηση οχήματος σε πίστα μέσα σε προσομοιωτή.

ABSTRACT

In this dissertation are presented various examples regarding finding faulty values and outliers in a dataset with the use of algorithms as DBSCAN, LOF, Isolation Forest and Tukey IQR method. Moreover, there will be an example of dealing with missing values in a timeseries dataset of temperature data and dimensionality reduction usind PCA algorithm. Finally, we will show examples of solving problems using deep learning libraries like the following Estimator API, Tensorflow and Keras with datasets such as banking data, electrical consumption, multi classification of pictures of sign language numbers [0-9] and predicting steering values for the use of driving a vehicle in autonomous mode in a simulator.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1- ΠΡΟ-ΕΠΕΞΕΡΓΑΣΙΑ ΚΑΙ ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ ΣΤΗΝ PYTHON

Σε αυτό το κεφάλαιο θα αναφερθούν οι διάφοροι τύποι δεδομένων η χρησιμότητα τους και οι διάφοροι τρόποι εύρεσης σφαλμάτων σε σύνολα δεδομένων καθώς και παράδειγμα μείωσης διαστάσεων. Καθώς και παραδείγματα αντικατάστασης ελλιπών τιμών για δεδομένα θερμοκρασίας.

1.1 Δεδομένα, Διερευνητική Ανάλυση Δεδομένων-Δ.Α.Δ (EDA).

Δεδομένα ορίζονται ως η συλλογή πληροφοριών, οι οποίες μπορεί να βρίσκονται είτε σε οργανωμένη μορφή, είτε σε μη οργανωμένη. Οργανωμένη μορφή είναι όταν τα δεδομένα είναι ταξινομημένα, δηλαδή η κάθε γραμμή αντιστοιχεί σε μια μέτρηση ή παρατήρηση, και η κάθε στήλη αντιστοιχεί σε ένα χαρακτηριστικό. Αντίθετα, μη οργανωμένα είναι τα δεδομένα τα οποία δεν είναι ταξινομημένα και συνήθως υπάρχουν κενά (Missing Data). Αυτού του τύπου τα δεδομένα χρειάζονται επεξεργασία για να μετασχηματιστούν σε οργανωμένη μορφή (Ozdemir, 2017, p. 5).

Η Διερευνητική Ανάλυση Δεδομένων-Δ.Α.Δ. (Exploratory Data Analysis-E.D.A.) είναι το πρώτο βήμα για:

- Κατανόηση των δεδομένων
- Έλεγχο των δεδομένων για σφάλματα
- Διόρθωση και προετοιμασία των δεδομένων (Correct & Organize Data)
- Γραφική απεικόνιση των δεδομένων (Data Visualization)
- Αναζήτηση πιθανών σχέσεων μεταξύ των διάφορων χαρακτηριστικών και μοτίβα
- Επιλογή κατάλληλων χαρακτηριστικών

1.1.1 Στατιστικά Δεδομένα

Τα δεδομένα χωρίζονται σε δυο κύριες κατηγορίες.

Ποιοτικά (*Qualitative*)

Τα δεδομένα αυτά είναι περιγραφικά. Με άλλα λόγια, στοχεύουν στην περιγραφή ενός αντικειμένου ή ενός γεγονότος που διεξάγεται σε μη αριθμητική

μορφή. Σε αντίθεση με τα *ποσοτικά* δεδομένα τα *ποιοτικά* εκφράζονται χρησιμοποιώντας την γλώσσα.
Για παράδειγμα, «*Η μπάλα του ποδοσφαίρου είναι στρογγυλή*».

Πληροφορίες που μπορούμε να πάρουμε είναι: Πόσα και ποια είναι τα μοναδικά χαρακτηριστικά , την ελάχιστη και μέγιστη συχνότητα εμφάνισης, καθώς και την συχνότητα εμφάνισης του (Ozdemir, 2017, pp. 30-34).

Ποσοτικά (*Quantitative*)

Αυτού του τύπου τα δεδομένα είναι αριθμητικά και περιγράφουν ένα αντικείμενο ή ένα γεγονός στο οποίο γίνεται μια παρατήρηση χρησιμοποιώντας αριθμούς. Για παράδειγμα, *«Η βιβλιοθήκη μου έχει 25 βιβλία», «Καθάρισα τον κήπο μου 10 φορές αυτόν τον μηνά», «Η μια μέρα έχει 24 ώρες», «Ο ζωολογικός κήπος έχει 157 διαφορετικά ζώα»*, πληροφορίες οι οποίες εκφράζουν ποσότητα. Υπάρχει η δυνατότητα συλλογής πληροφοριών από αυτού του τύπου δεδομένα χρησιμοποιώντας μαθηματικές πράξεις. Οι πληροφορίες αυτές αφορούν το *μέσο όρο, αν η τιμή αυξάνεται ή μειώνεται ανάλογα με τον χρόνο* και ερωτήματα όπως «*Τι θα γινόταν αν αυτή η ποσότητα αυξηθεί ή μειωθεί σημαντικά;*».

Τα ποσοτικά δεδομένα χωρίζονται σε 2 τύπους:

Διακριτά (*Discrete*)

Διακριτά δεδομένα θεωρούνται εκείνα τα οποία παίρνουν μορφή μόνο ακέραιων αριθμών. Για παράδειγμα, σε ένα φεστιβάλ πόσοι άνδρες και γυναίκες υπάρχουν. Έτσι, ποτέ δεν εμφανίζονται ως δεκαδικοί αριθμοί. Είναι δεδομένα στα οποία υπάρχουν κενά μεταξύ τους.

Συνεχή (*Continuous*)

Συνεχή είναι τα δεδομένα τα οποία δεν έχουν κάποια όρια. Μπορεί να είναι ακέραιοι αριθμοί αλλά και δεκαδικοί. Λόγου χάρη, ο χρόνος και η θερμοκρασία, στα οποία υπάρχει αέναη διακύμανση τιμών.

Τα δεδομένα χωρίζονται σε τέσσερα επίπεδα (Ozdemir, 2017, pp. 35-44) τα οποία είναι:

Ονομαστικά (Nominal)

Άλλη ονομασία για αυτό το επίπεδο είναι και *κατηγορικά*. Αυτού του τύπου δεδομένα περιγράφονται αποκλειστικά από κάποιο όνομα ή κατηγορία. Για παράδειγμα τα εξής: Φύλο, Είδος, Χρώμα…κλπ. Δηλαδή δε χρησιμοποιούνται αριθμοί και κατ' επέκταση είναι αδύνατες οι μαθηματικές πράξεις. Γεγονός που τα καθιστά ποιοτικά δεδομένα.

Διατάξιμα (Ordinal)

Δεδομένα τα οποία έχουν σειρά και κλιμάκωση, συναντώνται συχνά σε ερωτήσεις που ο χρήστης έχει την δυνατότητα να επιλέξει αναμεσά σε ένα εύρος τιμών. Για παράδειγμα, «*Βαθμολόγησε αυτό το προϊόν [0-10]»* ή «*Πείτε μας για την εμπειρία σας στο .... Φεστιβάλ, [Χάλια, Καλά, Πολύ καλά, Τέλεια]*». Υπάρχει η δυνατότητα σύγκρισης μεταξύ αυτών των δεδομένων αλλά δεν επιδέχονται επιπλέον μαθηματικές πράξεις.

Διαστήματος (Interval)

Σε αυτό το επίπεδο μελετάται η σύγκριση μεταξύ των δεδομένων ίδιου τύπου. Παράδειγμα, η ημερομηνία γέννησης, η θερμοκρασία σώματος. Για παράδειγμα, «*Η θερμοκρασία στην Αθήνα είναι 30 βαθμοί κελσίου», «Πάτρα 25 βαθμοί κελσίου*». Άρα από τα παραπάνω διεξάγεται το συμπέρασμα πως η Αθήνα είναι κατά +5 βαθμούς κελσίου πιο θερμή από την Πάτρα.

Αναλογικά (Ratio)

Αυτά τα δεδομένα έχουν όλα τα χαρακτηριστικά των άλλων επιπέδων. Επιπλέον μπορεί να είναι και διακριτά ή συνεχή. Παράδειγμα: *Ηλικία, Βάρος, Αριθμός βιβλίων σε μια βιβλιοθήκη*. Επιτρέπονται οι εξής μαθηματικές πράξεις: Αφαίρεση, Πρόσθεση, Πολλαπλασιασμός, Διαίρεση.

1.1.2 Χρησιμότητα Δεδομένων στον τομέα της Τεχνητής Νοημοσύνης (A.I)

Όταν μιλάμε για «*Τεχνητή Νοημοσύνη*» στο μυαλό μας έρχεται η δυνατότητα μιας «*Μηχανής*» να σκέφτεται, να παίρνει αποφάσεις με βάση κάποια δεδομένα. Όπως γίνεται αντιληπτό, χωρίς την ύπαρξη των δεδομένων τίποτε από αυτά δεν θα ήταν δυνατό. Κανένας αλγόριθμος μηχανικής μάθησης δεν θα μπορούσε να βγάλει σωστά αποτελέσματα χωρίς τα δεδομένα να είναι:

- Σε σωστή μορφή
- Οργανωμένα
- Επαρκή

Τίποτα δεν θα έβγαζε νόημα εάν η συλλογή τους γινόταν με λανθασμένο τρόπο. Ακόμη όμως και αν η συλλογή τους γινόταν σωστά, δε θα μπορούσε να γίνει η εισαγωγή τους σε κάποιον αλγόριθμο και να περιμένουμε αποτελέσματα χωρίς πρώτα να τα ελέγξουμε. Με άλλα λόγια, η σωστή συλλογή δεδομένων αλλά και η προ-επεξεργασία τους έχει σημασία στα αποτελέσματα μας.

Τα δεδομένα είναι επίσης χρήσιμα διότι αποσκοπούν στην λήψη αποφάσεων, την πρόβλεψη μιας πιθανής κατάστασης στο μέλλον αλλά και στην κατανόηση του παρελθόντος και του παρόντος. Αυτά είναι εφικτά μετά από ανάλυση τους, και χρήση των ειδικών αλγορίθμων μηχανικής μάθησης. Από τους αλγορίθμους αυτούς, ο χρήστης είναι σε θέση να κατανοήσει καλυτέρα τα δεδομένα, να ανακαλύψει μοτίβα και «κρυμμένη» πληροφορία μέσα σε αυτά που πριν ήταν πολύ δύσκολα και χρονοβόρα να ανακαλυφθούν.

1.1.3 Δεδομένα (Datasets)

Δεδομένα διαφόρων τύπων μπορούν να ανακτηθούν από αρκετές ιστοσελίδες. Ενδεικτικές ιστοσελίδες αναφέρονται παρακάτω. Στις περισσότερες τα δεδομένα βρίσκονται σε οργανωμένη μορφή.

- http://www.image-net.org/
- https://www.kaggle.com/datasets
- https://public.enigma.com
- https://archive.ics.uci.edu/ml/datasets.html
- https://vincentarelbundock.github.io/Rdatasets/datasets.html
- https://www.quandl.com

1.2 Ανίχνευση ανωμαλιών και σφαλμάτων

Η ανίχνευση ανωμαλιών είναι η εύρεση τιμών σε ένα σύνολο δεδομένων, οι οποίες δεν είναι *σχετικές.* Αποτελεί δηλαδή την ανίχνευση τιμών οι οποίες δεν ανήκουν στο σύνολο των δεδομένων που έχουν οριστεί ως φυσιολογικά. Συνήθως τέτοιες τιμές

υποδεικνύουν ότι υπάρχει πρόβλημα, ότι κάπου υπάρχει ένα είδος σφάλματος που απαιτεί περαιτέρω διερεύνηση. Άρα μια ανωμαλία είναι ένα απρόβλεπτο συμβάν-συμπεριφορά, σε σχέση με αυτό που θεωρείται φυσιολογικό, είτε αυτό δημιουργήθηκε από σφάλμα των αισθητήρων ή του εκάστοτε προγράμματος, είτε από τη συλλογή των δεδομένων.

1.2.1 Σημασία διόρθωσης πιθανών ανωμαλιών & σφαλμάτων

Τα σφάλματα σε ένα σύνολο δεδομένων θα πρέπει να κριθούν εάν είναι κατάλληλα να συμπεριληφθούν στη μελέτη ή όχι. Να σημειωθεί ότι αυτές οι τιμές (σφάλματα) δεν είναι πάντοτε ασήμαντες, καθώς εξαρτάται από το είδος της μελέτης που εφαρμόζεται σε κάθε περίπτωση.

Είναι σημαντικό να αναφερθεί πως η διαδικασία εύρεσης ανωμαλιών γίνεται κατά την διάρκεια της προετοιμασίας των δεδομένων. Σε αυτό το βήμα πρέπει να αφαιρεθούν οποιεσδήποτε "*Λάθος*" τιμές που αλλοιώνουν ένα μοντέλο πρόβλεψης το οποίο θα εκπαιδευτεί με βάση αυτές. Το παραπάνω βήμα κρίνεται απαραίτητο προκειμένου να λειτουργήσει σωστά ο αλγόριθμος και να αποφευχθούν τυχόν λανθασμένες προβλέψεις και συμπεράσματα. Άρα δεν αρκεί μόνο η γνώση του προγραμματισμού αλλά και η απαιτούμενη γνώση στον τομέα των δεδομένων προκειμένου να είμαστε σε θέση να κρίνουμε τι είναι φυσιολογικό και μη. Το αποτέλεσμα της διαδικασίας αυτής είναι ένα καθαρό σύνολο δεδομένων που είναι έτοιμο να περάσει στην επόμενη φάση της προ-επεξεργασίας.

Σε μελέτη των *Van den Broeck et al.* σε δεδομένα του τομέα της κλινικής επιδημιολογίας αναφέρουν το λεπτό ζήτημα του «καθαρισμού» των δεδομένων. Τα περισσότερα προβλήματα σε ένα σύνολο δεδομένων συνήθως προέρχονται από ανθρώπινα λάθη, η ανακρίβεια μιας μέτρησης από διάφορα όργανα μέτρησης είναι συνήθως αποδεχτή και θα πρέπει να εστιάσουμε στα σημαντικά σφάλματα τα οποία μπορούν να επηρεάσουν κατά πολύ τα συμπεράσματα της εκάστοτε μελέτης (Van den Broeck, 2005).

Ακόμη οι συγγραφείς τονίζουν πως για την αποφυγή σφαλμάτων πρέπει να γίνει πρόληψη κατά την διάρκεια της συλλογής των δεδομένων ή αμέσως μετα την καταχώριση. Μια απλή γραφική απεικόνιση διασποράς θα μπορούσε να βοηθήσει στην εύρεση σφαλμάτων σε ένα σύνολο δεδομένων. Αναφέρουν επίσης πως ορισμένες μελέτες πρέπει να επαναληφθούν με και χωρίς τα διάφορα σφάλματα που βρέθηκαν

στο σύνολο δεδομένων της μελέτης και να μελετηθούν τα συμπεράσματα τους (Van den Broeck, 2005).

Σε μια διαφορετική μελέτη των *Pollet et al.*, σε δεδομένα τεστοστερόνης, οι συγγράφεις μελέτησαν και παρουσίασαν τα αποτελέσματα αφενός όταν είχαν αφαιρέσει τα σφάλματα που ανήκαν εκτός των ορίων ±2.5 SD και ±3 SD, αφετέρου όταν οι τιμές εκτός ± 2.5 SD και ± 3 SD συμπεριλαμβανόντουσαν στο σύνολο δεδομένων. Τα αποτελέσματα φανέρωσαν, στο 7% έως 54% των περιπτώσεων, ότι όταν τα δεδομένα εκτός των επιτρεπτών ορίων αφαιρούνταν από το σύνολο δεδομένων, τα αποτελέσματα ήταν πιο αντικειμενικά (Pollet, 2016).

1.2.2 Αλγόριθμοι ανίχνευσης ανωμαλιών

Υπάρχουν πολλά είδη αλγόριθμων (DBSCAN, LOF και άλλα) που βοηθούν στην εύρεση πιθανών σφαλμάτων που ενδέχεται να υπάρχουν στα διάφορα σύνολα δεδομένων. Παρακάτω θα γίνει συνοπτική περιγραφή τους, όπως και των κανόνων που τους διέπουν και σε επόμενο κεφάλαιο θα γίνει πειραματική εφαρμογή τους σε συνθετικά δεδομένα.

1.2.2.1 Αλγόριθμος DBSCAN

Σύμφωνα με τη μελέτη των Martin Ester et al. είναι ένας αλγόριθμος ομαδοποίησης δεδομένων με θόρυβο. Ο αλγόριθμος δεν απαιτεί να γνωρίζουμε τον αριθμό ομάδων σε ένα σύνολο δεδομένων , και όπως απέδειξε η μελέτη τους μπορεί να βρίσκει και να ομαδοποιεί δεδομένα σε αυθαίρετα σχήματα (Ester, Kriegel, Sander, & Xu, 1996).

Ο αλγόριθμος αυτός διαθέτει συνολικά 8 παραμέτρους. Συγκεκριμένα οι δυο σημαντικότεροι παράμετροι θεωρούνται οι εξής: *epsilon, min_samples*

Όπου:

- *eps:* Είναι μια τιμή η οποία ορίζει την μέγιστη απόσταση που θα πρέπει να έχουν τα σημεία μεταξύ τους για να θεωρούνται ότι είναι μέρος μιας ομάδας/συνόλου.

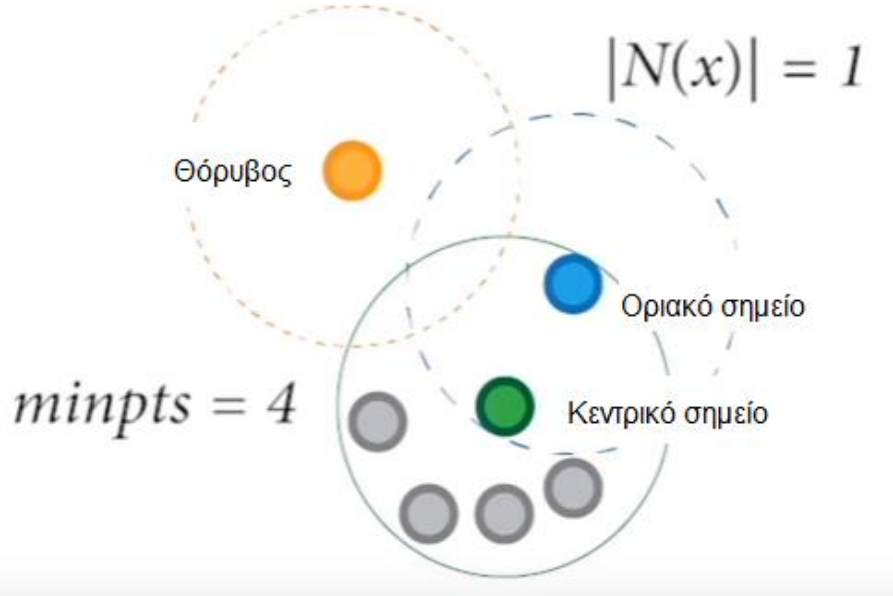
- *min_samples:* Ελάχιστος αριθμός σημείων γύρω από ένα σημείο που θα το καθορίσουν[1] ως ένα κεντρικό σημείο (Pedregosa, 2011).

Έπειτα από εφαρμογή του αλγορίθμου μπορούμε να πάρουμε μέσω model.labels_ τις πληροφορίες που χρειαζόμαστε, δηλαδή:

labels_: array, shape = [n_samples]: Επιστρέφει ίδιου μεγέθους πίνακα με τα δεδομένα. Με την διαφορά πως έχουν δοθεί τιμές από 0,1,2,3… έως n, υποδεικνύοντας σε ποια ομάδα ανήκει η κάθε τιμή. Τα δεδομένα στα οποία έχει δοθεί η τιμή -1 στο πίνακα αυτό, σημαίνει πως δεν ανήκουν σε κάποια ομάδα άρα απέχουν από αυτές και έτσι είναι «*Θόρυβος, Outliers, Anomalies*» (Pedregosa, 2011).

Ο αλγόριθμος αυτός το μόνο που χρειάζεται είναι τα δεδομένα, δεν χρειάζεται επίσης κάποια άλλη ρύθμιση από τον χρήστη εκτός από την κατάλληλη επιλογή των δυο παραμέτρων που αναφέρθηκαν παραπάνω.

[1]Τα σημεία αυτά πληρούν την παράμετρο ***eps***.



Εικόνα 1. Αναπαράσταση εφαρμογής του αλγορίθμου DBSCAN. Διακρίνεται το κεντρικό σημείο (Core point), το σημείο (Border) το οποίο βρίσκεται στα όρια της παραμέτρου «Epsilon» και τέλος την τιμή (Noise) η οποία θεωρείται "Θόρυβος" διότι βρίσκεται μακριά από όλα τα σημεία στο συγκεκριμένο σύνολο δεδομένων (Chaurasia, χ.χ.).

Ο συγκεκριμένος αλγόριθμος ομαδοποίησης, στην δυνατότητα του να μετράει την απόσταση μεταξύ δυο σημείων χρησιμοποιεί την *Ευκλείδεια* απόσταση η οποία συνιστά την πιο συνηθισμένη μέθοδο. Δουλεύει άρτια όταν υπάρχουν τρεις ή λιγότερες διαστάσεις στα δεδομένα, ενώ σε δεδομένα με πολλές διαστάσεις δηλαδή από τέσσερις και πάνω, η απόδοση του μειώνεται. Αυτό το φαινόμενο εμφανίζεται σε οποιονδήποτε αλγόριθμο χρησιμοποιεί την Ευκλείδεια απόσταση και είναι το ονομαζόμενο «*Curse of dimensionality*» (Koppen, n.d.). Επίσης ο αλγόριθμος συναντά δυσκολίες σε δεδομένα τα οποία έχουν διαφορετική πυκνότητα (Ertoz, Steinbach, & Kumar, 2003).

Μια ακόμη δυσκολία στον αλγόριθμο αυτόν είναι η σωστή ρύθμιση των δυο παραμέτρων. Αναφέρθηκε παραπάνω πως πρέπει να βρεθούν οι κατάλληλες τιμές ανάλογα με τα δεδομένα που πρόκειται να εφαρμοστούν στον αλγόριθμο.

Η εξοικείωση στον τομέα των δεδομένων μας βοηθάει να ορίσουμε καλύτερα τις δυο αυτές παραμέτρους. Άλλος τρόπος είναι να ορίσουμε κάποιο ποσοστό των δεδομένων ως μη φυσιολογικές τιμές διαλέγοντας πειραματικά τιμές για τις δυο παραμέτρους του αλγορίθμου *DBSCAN*.

1.2.2.1.1 Εφαρμογές του DBSCAN

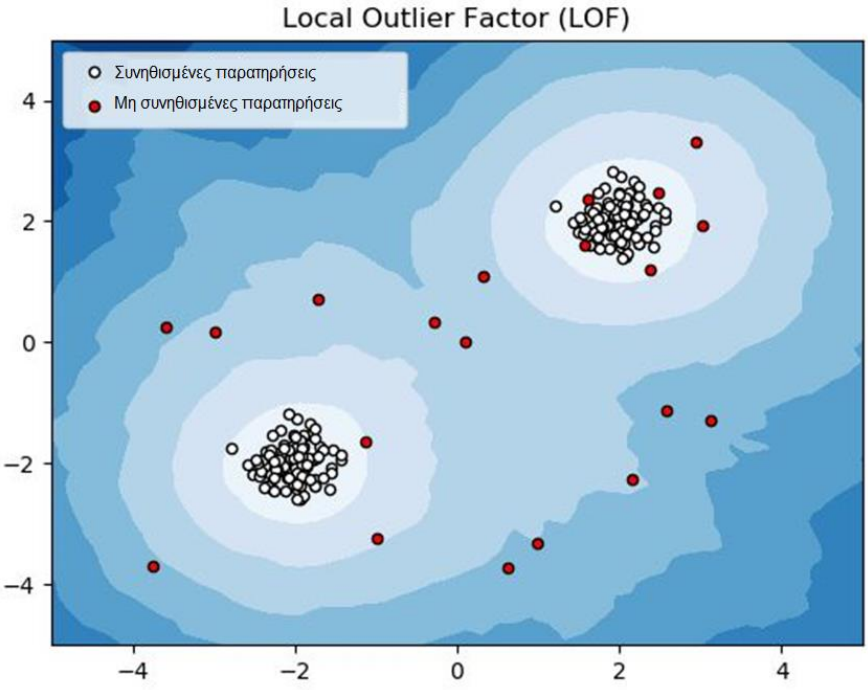
Σε μελέτη των *ÇELİK et al.* διερευνήθηκε η εύρεση σφαλμάτων και σε μηνιαία δεδομένα θερμοκρασίας για το χρονικό διάστημα από το 1975 έως το 2008 στην

Τουρκία χρησιμοποιώντας τον αλγόριθμο DBSCAN. Στην μελέτη τα δεδομένα θερμοκρασίας που παρουσίαζαν εποχικότητα (seasonality) και είναι φυσιολογικό να υφίστανται διακυμάνσεις ανάλογα την εποχή, αφαιρέθηκαν πριν εισαχθούν ή εφαρμοστούν μέθοδοι εύρεσης ανωμαλιών σε αυτά. Ακόμη, διαπιστώθηκε πως ο αλγόριθμος ήταν αποτελεσματικός να βρει ανώμαλες τιμές στα δεδομένα ακόμη κι αν αυτά δεν ήταν ακραίες τιμές (Celik, 2011).

Σε μελέτη των *Huan et al*. διερευνήθηκε η χρήση του αλγορίθμου DBSCAN για την σταθεροποίηση του θορύβου γύρω από κινούμενα αντικείμενα σε βίντεο. Μέσω πειραματικών διαδικασιών διαπιστώθηκε πως ο αλγόριθμος ήταν ικανός να ομαδοποιήσει τις γωνίες γύρω από τα κινούμενα αντικείμενα και να τα διακρίνει με αυτά του background, επίσης μπορεί να διακρίνει με ακρίβεια τις περιπτώσεις που υπάρχουν πολλά αντικείμενα, ποικίλου μεγέθους, τα οποία κινούνται στον χώρο (Huan Yu, 2013).

1.2.2.2 Αλγόριθμος *LOF*

Ο αλγόριθμος αυτός συγκρίνει ένα πυκνό σημείο στα δεδομένα με τα υπόλοιπα σημεία κοντά σε αυτό. Ο τρόπος που βρίσκει τα πυκνά σημεία μέσα στα δεδομένα είναι μέσω του αλγορίθμου K-NN χρησιμοποιώντας τις αποστάσεις που υπολογίζει. Αναγνωρίζει τα λιγότερο πυκνά σημεία στα δεδομένα και δεν τα ορίζει ως θόρυβο, αλλά ως ένα σημείο αναφοράς και ελέγχει τα κοντινότερα σημεία όπως θα έκανε εάν ήταν πυκνό σημείο. Δηλαδή λειτουργεί και με δεδομένα τα οποία έχουν ασύμμετρη κατανομή (Breunig, Kriegel, Ng, & Sander, 2000)



Εικόνα 2. Ο αλγόριθμος βρίσκει τα πυκνά σημεία με βάση το kNN, στην συνέχεια συγκρίνει τα κοντινά σημεία εάν αυτά είναι πυκνά η όχι . Εάν δεν είναι τα ορίζει ως θόρυβο. Ο αλγόριθμος είναι ικανός όχι μόνο να βρει τα τοπικά η αλλιώς τα γειτονικά σημεία αλλά και απομακρυσμένα (Global outliers) *(Breunig, Kriegel, Ng, & Sander, 2000)*

Στην περίπτωση που μια περιοχή στα δεδομένα δεν είναι τόση πυκνή όσο σε μια διαφορετική περιοχή, της συμπεριφέρεται το ίδιο όπως θα ήταν με ένα πυκνό σημείο. Έτσι ακόμη και οι αραιές περιοχές αναγνωρίζονται από τον αλγόριθμο σε αντίθεση με τον αλγόριθμο DBSCAN που δεν τις αναγνώριζε. (Breunig, Kriegel, Ng, & Sander, 2000).

Ο αλγόριθμος στην απλούστερη μορφή εφαρμογής του, ορίζεται με τις εξής παραμέτρους:

- *n_neighbors: int, optional (default=20):* Ορισμός αριθμό γειτονικών σημείων.

- *contamination: float in (0., 0.5), optional (default=0.1)*: Η παράμετρος αυτή ορίζει το ποσοστό δεδομένων το οποίο είναι και ο αριθμός σημείων θορύβου σε ένα σύνολο δεδομένων. Με μέγιστο 0.5 δηλαδή το 50% του συνόλου δεδομένων να οριστεί ως θόρυβο (Pedregosa, 2011).

Πληροφορίες έπειτα από ορισμό του αλγορίθμου (outputs):

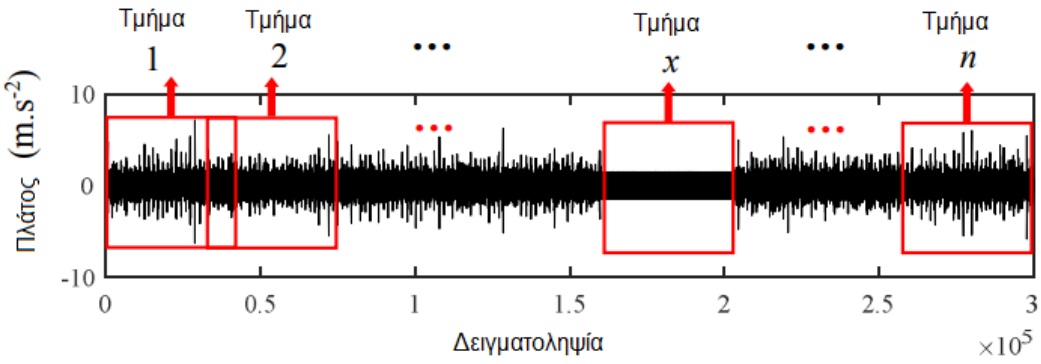
- *negative_outlier_factor_: numpy array, shape (n_samples,)*: Περιέχει αρνητικές τιμές του «LOF score», τιμές κοντά στο [-1] θεωρούνται φυσιολογικές. Τιμές μικρότερες από αυτό θεωρούνται θόρυβο.
- n_neighbors_ : Ο αριθμός των γειτονικών σημείων, όπου χρησιμοποιήθηκε όταν ορίστηκε το μοντέλο (Pedregosa, 2011).

Έπειτα από τον ορισμό του αλγορίθμου, γίνεται η προσαρμογή του στα δεδομένα (*model.fit_predict*), ως έξοδος λαμβάνεται ένας πίνακας ο οποίος περιέχει τις τιμές LOF (σκορ). Οι ετικέτες με τιμές [1] ορίζεται ως μια φυσιολογική τιμή και ετικέτες με τιμές [-1] ως ένα σημείο θορύβου (Pedregosa, 2011).

1.2.2.2.1 Εφαρμογές του LOF

Σε μελέτη των *Xuefang et al.* διερευνήθηκε η απόδοση του αλγορίθμου LOF για την εύρεση ανώμαλης συμπεριφοράς δεδομένων που αφορούσαν την λειτουργικότητα μηχανήματων. Συγκεκριμένα, στην μελέτη χρησιμοποιήθηκαν τα δεδομένα ενός επιταχυνσιόμετρου για την καταγραφή δονήσεων που αφορούσαν την λειτουργικότητα μιας ανεμογεννήτριας. Τα δεδομένα αυτά χωριστήκαν σε τμήματα, στην μελέτη τους οι συγγράφεις αναφέρουν πως το μέγεθος μιας ανώμαλης συμπεριφοράς είναι άγνωστο. Έτσι, εφάρμοσαν την τεχνική «Sliding Window».

Για τον καθορισμό του βέλτιστου εύρους της εν λόγω τεχνικής επιλέχθηκε το εύρος εκείνο το οποίο είχε μέγιστο *LOF* και όρισαν ποσοστό επικάλυψης 0.9.

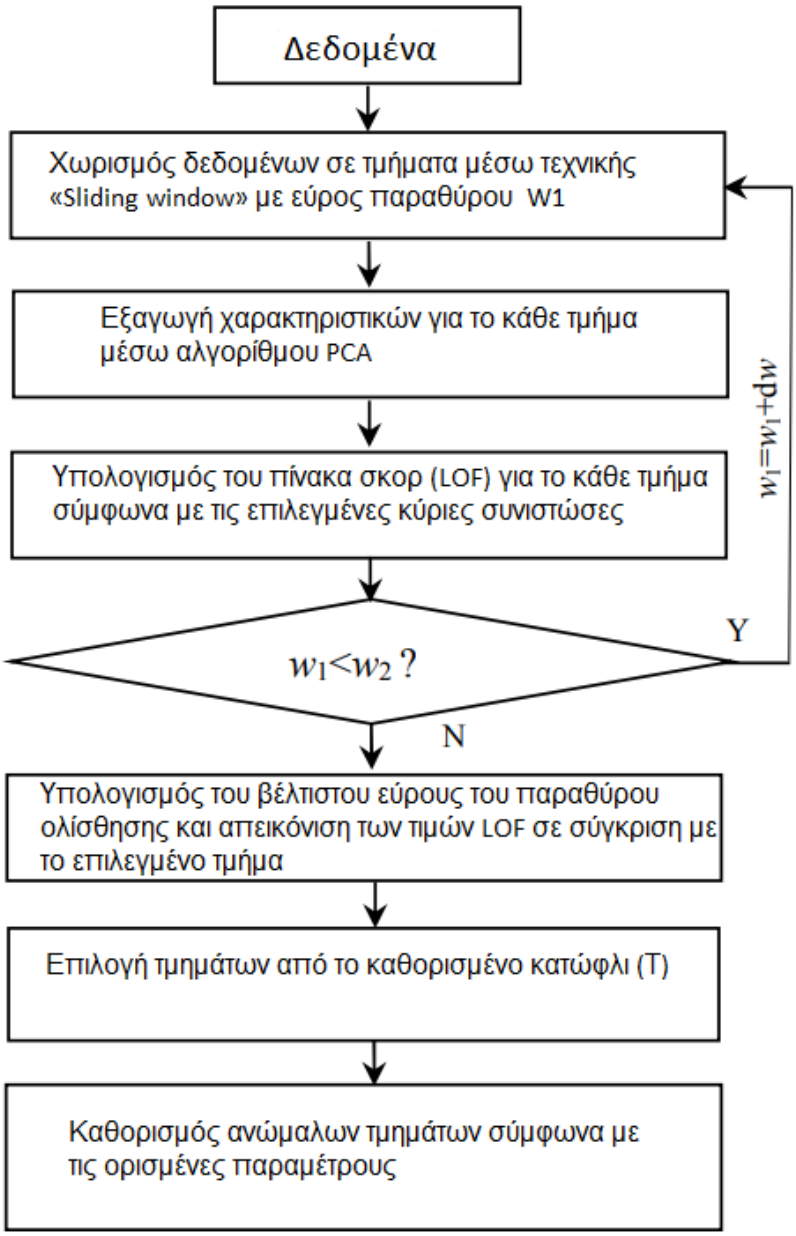


Εικόνα 3. Διαχωρισμός δεδομένων σε τμήματα χρησιμοποιώντας παράθυρο ολίσθησης (Xu, 2018)

Χρησιμοποίησαν τον αλγόριθμο μείωση διαστάσεων *PCA* (Ανάλυση Κύριων Συνιστωσών) για την εξαγωγή χαρακτηριστικών για το κάθε τμήμα και κρατήθηκαν οι κύριες συνιστώσες που είχαν συνολική διακύμανση *99.99%* και έγινε εκ νέου υπολογισμός του *LOF*. Οι συγγραφείς έθεσαν το κατώφλι (T) : Τμήμα

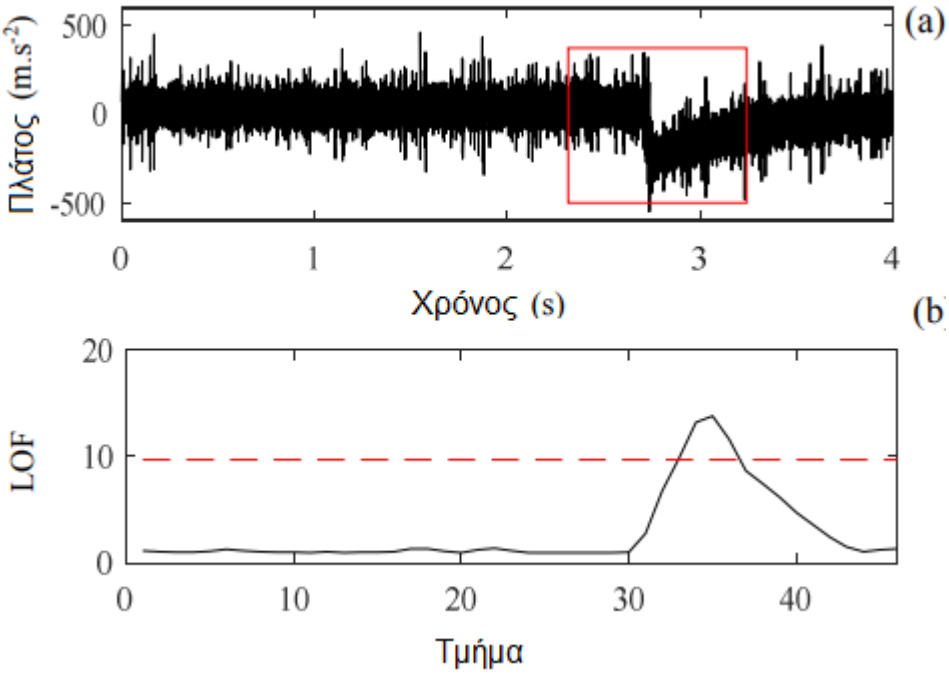
$$\boldsymbol{T = \mu + \lambda\sigma}$$

Όπου $\boldsymbol{\mu}$ και $\boldsymbol{\sigma}$ είναι ο μέσος όρος και η τυπική απόκλιση αντίστοιχα. Το $\boldsymbol{\lambda}$ αντιπροσωπεύει το βαθμό της ανώμαλης συμπεριφοράς του τμήματος. Στην μελέτη επιλέχθηκε ο βαθμός να είναι δυο. Άρα οποιαδήποτε τιμή στο *LOF* η οποία είναι εκτός ορίου *T* θα θεωρηθεί ότι το συγκεκριμένο τμήμα δεδομένων παρουσιάζει ανώμαλη συμπεριφορά. Παρακάτω είναι το διάγραμμα ροής της μελέτης αυτής.



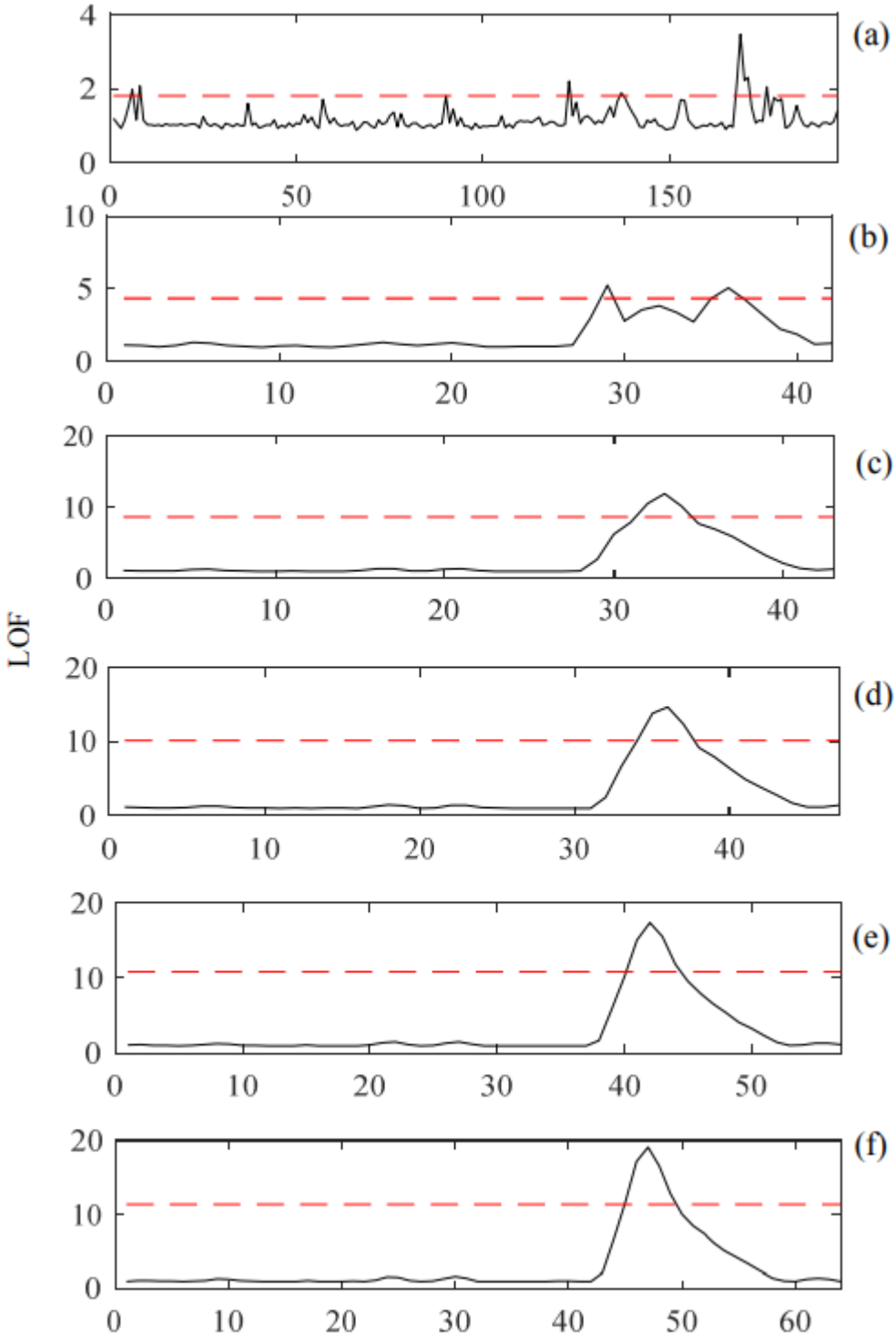
Εικόνα 4. Διάγραμμα ροής της μεθόδου (Xu, 2018)

Παρακάτω παρουσιάζεται ένα παράδειγμα των αποτελεσμάτων της μελέτης αυτής. Να σημειωθεί ότι για τη παράμετρο ***K*** επιλέχθηκε τιμή 18. Επιλογή τμημάτων από το καθορισμένο κατώφλι (T).



Εικόνα 5. Στο πάνω διάγραμμα αναπαρίστανται τα πραγματικά δεδομένα, και στο κάτω τα αποτελέσματα του αλγορίθμου. Παρατηρείται πως η προτεινόμενη μέθοδος στην μελέτη είχε καλά αποτελέσματα (Xu, 2018).

Τέλος οι συγγραφείς δίνουν έμφαση στην παράμετρο ***k***. Όταν γίνεται επιλογή μικρού ***k*** η προτεινόμενη μέθοδος δεν είναι σε θέση να εντοπίσει την ανώμαλη συμπεριφορά στα τμήματα, ενώ η επιλογή μεγάλου ***k*** αυξάνει την ακρίβεια της μεθόδου έτσι ώστε να εντοπίζονται οι ανώμαλες συμπεριφορές. Οι συγγραφείς αναφέρουν πως χρειάζεται ιδιαίτερη προσοχή στην εύρεση βέλτιστου ***k***. Παρακάτω παρουσιάζεται ένα παράδειγμα με επιλογή παραμέτρου ***k*** [5-30] και τα αποτελέσματα τις μεθόδου.

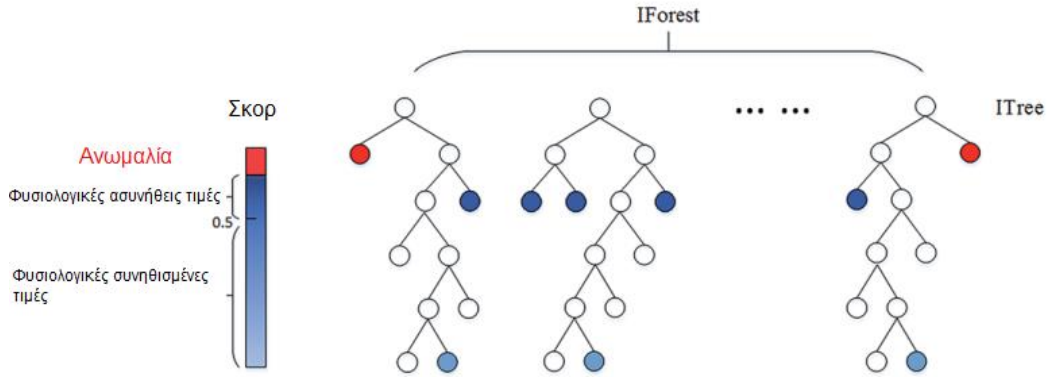


Εικόνα 6. Για (a) k=5, (b) k=10,(c) k=15,(d) k=20,(e) k=25,(f) k=30 (Xu, 2018)

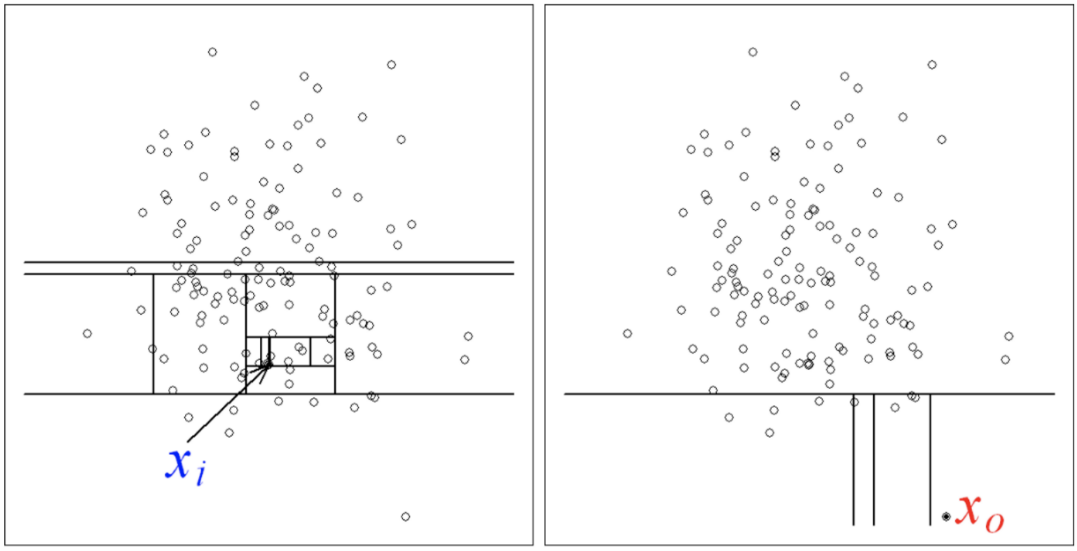
1.2.2.3 Αλγόριθμος Isolation Forest

Ο αλγόριθμος αυτός δεν χρησιμοποιεί κάποια μέθοδο των παραπάνω αλγορίθμων, δηλαδή δεν λειτουργεί με βάση την πυκνότητα ή την απόσταση των σημείων. Χρησιμοποιεί την λογική από *Random Decision Forest*, έτσι το *iForest* δημιουργεί δενδροδιάγραμμα απόφασης επιλέγοντας τυχαία χαρακτηριστικά από το σύνολο δεδομένων (Ho, 1995).

Η μέθοδος του αλγορίθμου στηρίζεται στην τυχαία επιλογή ενός χαρακτηριστικού από το σύνολο δεδομένων και διαλέγει μια τιμή χωρισμού (*Split value*) μεταξύ της μέγιστης και της ελάχιστης τιμής για το συγκεκριμένο χαρακτηριστικό. Η διαδικασία συνεχίζεται έως ότου η επιλεγμένη τιμή να απομονωθεί. Η κάθε προσπάθεια του αλγορίθμου να απομονώσει την εν λόγω τιμή έχει ως αποτέλεσμα την αύξηση (σε μέγεθος) του δενδροδιαγραμμάτων απόφασης (βλέπε Εικόνα 7) (Liu, Ting, & Zhou, 2008).



Εικόνα 7. Τα κόκκινα σημεία είναι τα σημεία εκείνα όπου ο αλγόριθμος κατάφερε να απομονώσει χωρίς μεγάλη προσπάθεια, ενώ με μπλε είναι εκείνα που θα θεωρηθούν ως φυσιολογικές τιμές (Chen, Yun, Wen, Lu, & Zhang, 2016).



Εικόνα 8. Στο πρώτο διάγραμμα φαίνεται ότι εάν ένα σημείο χρειάζεται αρκετές προσπάθειες για την απομόνωση του σημαίνει πως έχει κι άλλα σημεία κοντά του, δηλαδή βρίσκεται σε μια πυκνή περιοχή και άρα θεωρείται φυσιολογική τιμή. Στο δεύτερο διάγραμμα φαίνεται ότι εάν ένα σημείο ή ένα επιλεγμένο χαρακτηριστικό σε ένα σύνολο δεδομένων απομονώνεται εύκολα σημαίνει πως το σημείο αυτό είναι θόρυβος. (Liu, Ting, & Zhou, 2008).

Το μέγεθος του κάθε δενδροδιαγράμματος απόφασης μεταφράζεται σε ένα πίνακα σκορ. Οι τιμές του πίνακα σκορ είναι μεταξύ *[-0.5 έως +0.5]*. Τιμές μικρότερες από το 0 σύμφωνα με τις ρυθμίσεις του αλγορίθμου θεωρούνται σφάλματα, ενώ μεγαλύτερες από 0 είναι φυσιολογικές (Liu, Ting, & Zhou, 2008).

Χρησιμοποιώντας την ιδιότητα *fit_predict(Dataset)* με παράμετρο *contamination = 'auto'*, ο αλγόριθμος επιστρέφει έναν πίνακα ο οποίος θεωρεί τις τιμές [-1] ως ανώμαλα σημεία, και τιμές [+1] ως φυσιολογικά σημεία (Liu, Ting, & Zhou, 2008).

Λειτουργεί επίσης με δεδομένα πάνω από τρεις διαστάσεις, διότι όπως αναφέρθηκε παραπάνω δεν χρησιμοποιεί μεθόδους απόστασης ή πυκνότητας και δεν επηρεάζεται από το *"The Curse of Dimensionality"*(Liu, Ting, & Zhou, 2008).

Παράμετροι αλγορίθμου *Isolation Forest* :

- n_estimators: *int, optional (default=100)*: Αριθμός δενδροδιαγραμμάτων απόφασης (Pedregosa, 2011).
- *max_samples: int or float, optional (default=”auto”)*: Αριθμός χαρακτηριστικών/σημείων που θα επιλεχθούν τυχαία από τον αλγόριθμο
- *contamination: float in (0., 0.5), optional (default=0.1)*: Ορισμός ποσοστού ανωμαλιών στα δεδομένα (Pedregosa, 2011).
- *max_features: int or float, optional (default=1.0)*: Αριθμός χαρακτηριστικών για το κάθε δενδροδιάγραμμα απόφασης (Pedregosa, 2011).

Ιδιότητες :

- *offset_: float*: Αριθμός αντιστάθμισης για τον υπολογισμό του πίνακα σκορ. Έχει τιμή 0.5 στην περίπτωση που η παράμετρος contamination ισούται με “*auto”*, διαφορετικά η τιμή του είναι ίδια με την τιμή της παραμέτρου *contamination* (Pedregosa, 2011).
- *score_samples(Dataset)*: Πίνακας των σκορ με βάση τον αριθμό χωρισμάτων (Isolation Splits) που χρειάστηκαν να απομονωθεί ένα συγκεκριμένο σημείο. Τιμές μικρότερες από [-0.5] θεωρούνται ανωμαλίες ενώ τιμές μεγαλύτερες από [-0.5] θεωρούνται φυσιολογικές (Pedregosa, 2011).
- *fit_predict (Dataset, y = None)*: Μέθοδος για «Unsupervised Anomaly Detection» η οποία επιστρέφει πίνακα σκορ στον οποίο οι τιμές [-1] θεωρούνται ανωμαλίες και τιμές [+1] θεωρούνται φυσιολογικές (Pedregosa, 2011)

1.2.2.3.1 Εφαρμογές Isolation Forest

Σε μελέτη των *Zhang et al*. (Zhang, Kang, & Li, 2019) που μελετήθηκε ο αλγόριθμος για την ικανότητα και την απόδοση του αναφορικά με την εύρεση ανωμαλιών σε υπερφασματικα δεδομένα[2]. Οι συγγράφεις στην μέθοδο που προτείνουν χρησιμοποιούν τον αλγόριθμο για να πάρουν τον πίνακα ανωμαλιών $\hat{S}$ της ολόκληρης πρώτης υπερφασματικης εικόνας, αλλά παρατήρησαν πως αρκετά μικρά αντικείμενα στην εικόνα και λόγω της πολυπλοκότητας της, τείνουν να έχουν αρκετά μεγάλο σκορ ανωμαλίας. Επιπλέον, διαπιστώθηκε ότι όταν το σκορ σε ένα εικονοστοιχείο (pixel) είναι κοντά στο 0.5 είναι δύσκολο να εκτιμηθεί εάν αυτό προκύπτει λόγω του background ή είναι όντως μια ανωμαλία. Για την αντιμετώπιση αυτού του προβλήματος οι συγγράφεις προτείνουν τη δημιουργία ενός τοπικού iForest για την βελτίωση των σκορ για τα σημεία αυτά. Τα επόμενα βήματα τους είναι να μετατρέψουν τον πίνακα αυτόν $\hat{S}$ σε δυαδική μορφή:

$$\boldsymbol{\delta = THRESH(\hat{S})}$$

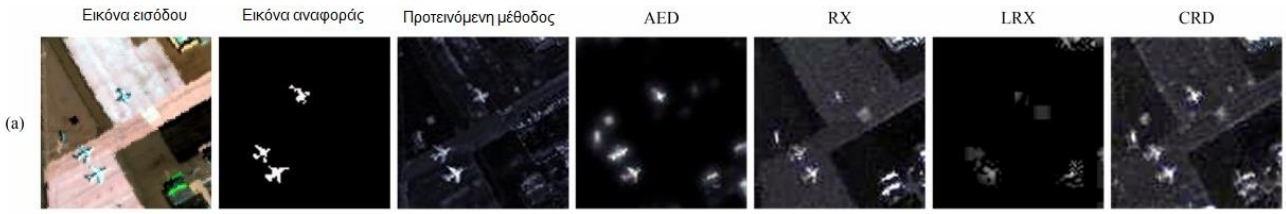
όπου $\boldsymbol{\delta}$ είναι ένα κατώφλι το οποίο υπολογίζεται αυτόματα με τη χρήση της μεθόδου Otsu στον πίνακα ανωμαλιών $\hat{S}$ (Liu D. &., 2009). Οπότε ο δυαδικός πίνακας προκύπτει από:

$$\widehat{S}_b = \begin{cases} 1, & if\ \hat{S} > \delta \\ 0, & if\ \hat{S} \leq \delta \end{cases}$$

Τέλος οι συγγραφείς για την εύρεση ανώμαλων αντικειμένων επισημαίνουν πως εν γένει οι ανωμαλίες παρουσιάζονται κυρίως σε μικρές περιοχές, οπότε έχοντας τον πίνακα $\hat{\boldsymbol{S}}_{\boldsymbol{b}}$ μπορούν να εξάγουν όλα εκείνα τα «ανώμαλα» σημεία που ορίζουν ένα αντικείμενο. Ακόμη αναφέρουν πως εάν η περιοχή/επιφάνεια του αντικειμένου είναι μεγαλύτερη ενός δεδομένου $\boldsymbol{\alpha}$ (κατώφλι) τότε θα δημιουργηθεί ένας τοπικός αλγόριθμος iForest ο οποίος θα περιέχει τα μισά υπερφασματικα εικονοστοιχεία. Κατόπιν, θα γίνεται επικαιροποίηση του αρχικού πίνακα σκορ $\hat{\boldsymbol{S}}_{\boldsymbol{b}}$. Ως προϋπόθεση για τον τερματισμό της μεθόδου αυτής, ορίστηκε όταν πλέον δεν υπάρχουν αλλά

[2] http://xudongkang.weebly.com/data-sets.html

αντικείμενα τα οποία να έχουν μεγαλύτερη επιφάνεια από το α. Παρακάτω τα αποτελέσματα τους (Zhang, Kang, & Li, 2019).



Εικόνα 9. Στην πρώτη εικόνα δίνεται η αρχική κάτοψη ενός αεροδρομίου. Στις επόμενες διακρίνονται οι δοκιμές που πραγματοποιήθηκαν με την μέθοδο που πρότειναν οι συγγραφείς και έγινε σύγκριση των αποτελεσμάτων με άλλες μεθόδους (Zhang, Kang, & Li, 2019).

Δεδομένα	Προτεινόμενη μέθοδος	AED [9]	RX [4]	LRX [4]	CRD [11]
Airport	**0.9902**	0.9846	0.9403	0.9678	0.9781

Εικόνα 10. Παρουσιάζονται οι επίδοσης AUC (Area Under The Curve) (Narkhede, 2018) τις προτεινόμενης μεθόδου σε σύγκριση με άλλες μεθόδους (Zhang, Kang, & Li, 2019).

1.2.2.4 Μέθοδος Tukey Box Plot & Tukey outliers (IQR)

Η μέθοδος αυτή κάνει χρήση του διατεταρτημοριακού διαστήματος. Το κατώτατο τεταρτημόριο Q_1 είναι αφορά στο 25% των δεδομένων, και το ανώτατο Q_3 στο 75% των δεδομένων (Tukey, 1977) (Seo, 2002).

Αρά χρησιμοποιώντας αυτές τις πληροφορίες η τιμή του διατεταρτημοριακού διαστήματος είναι:

$$IQR = Q_3 - Q_1$$

Ορισμός των ορίων:

$$Κατωτατοορισ = Q_1 - (IQR * 1{,}5)$$

$$Ανωτατοορισ = Q_3 + (IQR * 1{,}5)$$

Οι τιμές οι οποίες βρίσκονται εκτός των ορίων αυτών είναι πιθανά ανώμαλα σημεία σε ένα σύνολο δεδομένων. Η μέθοδος αυτή είναι χρήσιμη στην εύρεση ακραίων τιμών μέσα σε ένα σύνολο δεδομένων, αλλά επίσης και για πιθανές ανώμαλες τιμές οι οποίες εάν βρεθούν πρέπει να ερευνηθούν περαιτέρω.

1.3 Ελλείποντα στοιχεία

Η ποιότητα των δεδομένων μπορεί να επηρεάσει αρνητικά τα αποτελέσματα μιας μελέτης. Αλγόριθμοι μηχανικής αλλά και βαθιάς μάθησης βασίζονται σε μεγάλο βαθμό στην ποιότητα των δεδομένων και της προ-επεξεργασίας τους για να αποφέρουν καλά αποτελέσματα (Raschka & Mirjalili, 2017).

Σχεδόν σε όλα τα συστήματα συλλογής δεδομένων, είτε αυτά συλλέγονται αυτόματα είτε από τον άνθρωπο, παρατηρείται το φαινόμενο των ελλειπόντων δεδομένων. Με άλλα λόγια, μέσα σε ένα σύνολο δεδομένων να υπάρχουν «θέσεις» οι οποίες δεν διαθέτουν κάποια τιμή (Raschka & Mirjalili, 2017).

1.3.1 Λόγοι και τρόποι αντιμετώπισης

Ελλείποντα στοιχεία προκύπτουν από διάφορες αιτίες. Μερικές από αυτές είναι:

1. Αποτυχία συστημάτων συλλογής δεδομένων
   Συμβαίνει όταν τα συστήματα αυτά τίθενται εκτός λειτουργίας και έτσι σταματάει η διαδικασία συλλογής με αποτέλεσμα να χαθεί πολύτιμος χρόνος και δεδομένα (Altman, 2007).

2. Αποτυχία αισθητήρων
   Αποτελεί την πιο συνηθισμένη αιτία στην οποία ένας ή περισσότεροι αισθητήρες οι οποίοι «παρακολουθούν» μια διαδικασία *φθείρονται ή δυσλειτουργούν*, αποτυγχάνοντας κατά αυτόν τον τρόπο να πραγματοποιήσουν μετρήσεις. Οι λόγοι που οδηγούν σε μια τέτοια κατάσταση αφορούν την κακή διαχείριση/ εγκατάσταση, την καθυστέρηση μιας προγραμματιζόμενης συντήρησης τους, ή τα κατασκευαστικά σφάλματα. Προκειμένου να προληφθεί αυτό το ανεπιθύμητο συμβάν, συνιστάται έλεγχος της λειτουργία των αισθητήρων πριν εκείνα εγκατασταθούν, ενώ συνήθως για μια μέτρηση προτείνεται να υπάρχουν δυο ή παραπάνω αισθητήρες για σύγκριση τιμών (RAMBO, 2018).

3. Προγραμματιστικά σφάλματα
   Αυτά συμβαίνουν σε ένα κακογραμμένο πρόγραμμα, που δεν έχουν υπολογιστεί όλες οι πιθανές καταστάσεις, που μπορεί να συμβεί κάτι λάθος. Συγκεκριμένα, στην περίπτωση που αποτυγχάνει ένας αισθητήρας δεν υπάρχει κάποια δικλείδα ασφάλειας μέσα στο πρόγραμμα με την οποία να συλλέγονται και να

αποθηκεύονται τα δεδομένα για να ενημερώνονται οι υπεύθυνοι. Ως αποτέλεσμα να χαθούν τα δεδομένα από τον εν λόγω αισθητήρα.

4. Ανθρωπινά λάθη

   Κατά τη συλλογή δεδομένων (*Ερωτηματολόγια* και άλλα) υπάρχει περίπτωση τα άτομά του δείγματος να μην έχουν συμπληρώσει σωστά όλα τα πεδία του ερωτηματολογίου με αποτέλεσμα αυτά τα πεδία να παραμένουν κενά. Δεδομένα που βρίσκονται τυπωμένα σε χαρτί όπως *ερωτηματολόγια* είναι απαραίτητη η μεταφορά τους σε ψηφιακή μορφή. Κατά την διαδικασία αυτή, είναι πολύ πιθανό να λόγω ανθρώπινης παράβλεψης να μη γίνει σωστή καταγραφή τους (Altman, 2007).

Η αντιμετώπιση των παραπάνω προβλημάτων κρίνεται μείζονος σημασίας. Η δημιουργία μοντέλων πάνω στην μηχανική και βαθιά μάθηση που χρησιμοποιεί δεδομένα με ελλείποντα στοιχεία έχει ως συνέπεια χαμηλή ακρίβεια και λάθος συμπεράσματα. Ακόμη, τα ελλείποντα στοιχεία προσθέτουν μεροληψία (*Bias*) στο μοντέλο, οδηγώντας το σε χαμηλή απόδοση και αυξημένο σφάλμα. Ακόμη και στην περίπτωση που χρησιμοποιούνται τεχνικές αφαίρεσης ή συμπλήρωσης των κενών θέσεων, εξακολουθούν να παρατηρούνται τέτοιο είδους προβλήματα ενώ ταυτόχρονα δεν υπάρχει αξιοπιστία. (Dillon, 2018)

1.4 Μείωση Διαστάσεων, PCA (Ανάλυση Κύριων Συνιστωσών)

Η συγκεκριμένη μέθοδος χρησιμοποιείται κυρίως για τη μείωση των διαστάσεων ενός συνόλου δεδομένων. Λόγοι που χρησιμοποιείται ο αλγόριθμος αυτός:

- Μείωση χρόνου εκπαίδευσής για αλγορίθμους μηχανικής και βαθιάς μάθησης
- Απεικόνιση πολυδιάστατου συνόλου δεδομένων
- Εύρεση πιθανών ομαδοποιήσεων
- Εξαγωγή σημαντικών χαρακτηριστικών (*Feature extraction*)

Το PCA, όπως αναφέρθηκε μειώνει τις διαστάσεις ενός συνόλου δεδομένων. Η διαδικασία επιτυγχάνεται μέσω γραμμικών συνδυασμών σε όλα τα χαρακτηριστικά (features/variables), από το αρχικό σύνολο δεδομένων. Διατηρεί τις μεταβλητές εκείνες

οι οποίες έχουν τη μεγαλύτερη διακύμανση (Smith, 2002) (Jonathon, Systems Neurobiology Laboratory, & Science, 2005) (Williams, 2016).

Το μεγαλύτερο μειονέκτημα του είναι ότι λαμβάνει υπόψη όλα τα χαρακτηριστικά για να βρει τους γραμμικούς συνδυασμούς και για αυτόν τον λόγο δεν είναι κατάλληλος να λειτουργεί με δεδομένα δυαδικής λογικής[3] (Zou, Hastie, & Tibshirani, 2006) (Hsu, Huang, & Chen, 2014) (Williams, 2016).

[3] Σύνολα δεδομένων τα οποία έχουν στόχο [0,1/ Είναι, Δεν είναι]

ΚΕΦΑΛΑΙΟ 2- ΒΙΒΛΙΟΘΗΚΗ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ *TENSORFLOW*

Η *TensorFlow* (TF), είναι μια βιβλιοθήκη λογισμικού ανοιχτού κώδικα για υψηλούς αριθμητικούς υπολογισμούς. Αναπτύχθηκε από την ομάδα τεχνητής νοημοσύνης *Google Brain*. Υποστηρίζει εφαρμογές *μηχανικής* αλλά και *βαθιάς μάθησης*. Η ευέλικτη αρχιτεκτονική του επιτρέπει να λειτουργεί σε οποιαδήποτε μονάδα επεξεργασίας (*CPU, GPU, TPU*) και λειτουργικά συστήματα (Raschka & Mirjalili, 2017).

2.1 Estimator Api, Keras, TensorBoard

Το *Estimator API* απλοποιεί της εφαρμογές για *μηχανική μάθηση*. Καθιστά εύκολη την *εκπαίδευση, εκτίμηση, πρόβλεψη, εξαγωγή και προετοιμασία του μοντέλου για την παραγωγή* σε σχέση με την χρήση χαμηλού επιπέδου *TF API*. Παρέχει ήδη ορισμένα μοντέλα[4] όπως τα *LinearRegressor, LinearClassifier, DNNRegressor και DNNClassifier*, αλλά δίνεται και η δυνατότητα για τη δημιουργία ενός νέου μοντέλου[5].

Το *Keras* όπως και το *Estimator API*, παρέχει μεγάλη ευκολία στην δημιουργία μοντέλων. Είναι γρήγορη όσον αφορά την δημιουργία πρωτότυπων μοντέλων για χρήση στην έρευνα και την παραγωγή. Το *Keras* χρησιμοποιείται κυρίως για τη β*αθιά μάθηση*, καθώς παρέχει μεγάλη διευκόλυνση στους χρήστες για την ανάπτυξη τεχνητών νευρωνικών δικτύων.

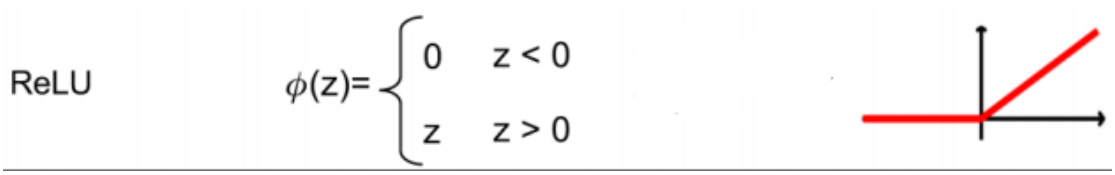
Το *TensorBoard*[6] χρησιμοποιείται για την απεικόνιση μετρήσεων όπως την ακρίβεια και το ποσοστό μάθηση ενός μοντέλου, και γενικές πληροφορίες για το μοντέλο και το υπολογιστικό γράφημα που δημιουργήθηκε από την TensorFlow. Χρησιμοποιώντας αυτές τις χρήσιμες πληροφορίες, δίνεται η δυνατότητα βελτίωσης του μοντέλου.

[4] https://www.tensorflow.org/api_docs/python/tf/estimator
[5] https://www.tensorflow.org/guide/custom_estimators
[6] https://www.tensorflow.org/guide/summaries_and_tensorboard

2.2 Βασικές Μη Γραμμικές Συναρτήσεις Ενεργοποίησης

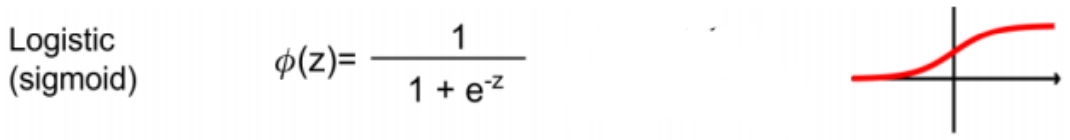
Στα νευρωνικά δίκτυα η χρήση μη γραμμικών συναρτήσεων ενεργοποίησης εξυπηρετεί την εύρεση πιθανών πολύπλοκων μοτίβων στα δεδομένα (Raschka & Mirjalili, 2017).

Η συνάρτηση ενεργοποίησης *ReLU*, όταν λαμβάνει αρνητικές τιμές εισόδου επιστρέφει μηδέν και όταν λαμβάνει θετικές τιμές επιστρέφει την ίδια τιμή εισόδου (Raschka & Mirjalili, 2017).



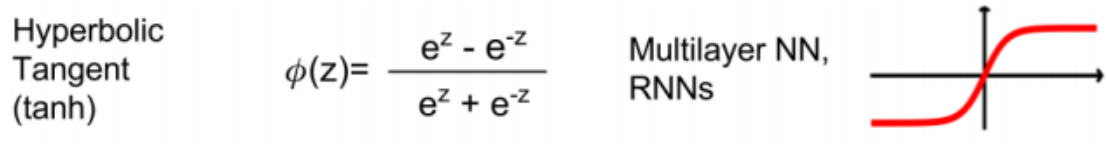
Εικόνα 11. Συνάρτηση ενεργοποίησης ReLU (Raschka & Mirjalili, 2017)

Συνάρτηση ενεργοποίησης *Sigmoid*, η συνάρτηση κατανέμει τα δεδομένα εισόδου σε ένα συγκεκριμένο διάστημα [0,1] (Raschka & Mirjalili, 2017).



Εικόνα 12. Συνάρτηση ενεργοποίησης ReLU (Raschka & Mirjalili, 2017)

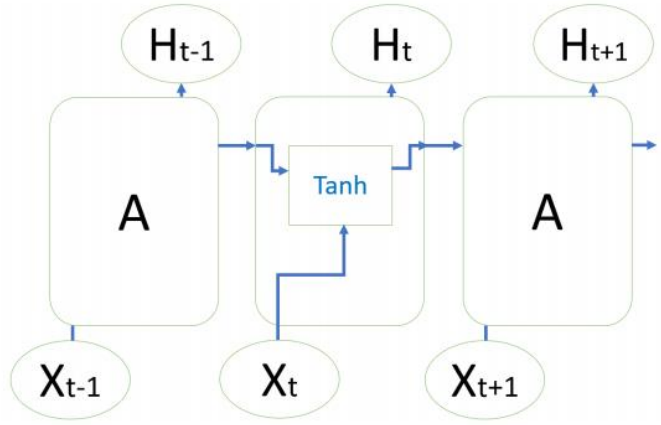
Συνάρτηση ενεργοποίησης *Tanh*, η συνάρτηση αυτή κατανέμει τα δεδομένα εισόδου σε ένα συγκεκριμένο διάστημα [-1,1] (Raschka & Mirjalili, 2017).



Εικόνα 13. Συνάρτηση ενεργοποίησης ReLU (Raschka & Mirjalili, 2017)

2.3 RNN-LSTM (Long Short-Term Memory)

Το επαναλαμβανόμενο νευρωνικό δίκτυο (Recurrent neural network) είναι μια μορφή δικτύου η οποία κάνει χρήση λογικής RNN.



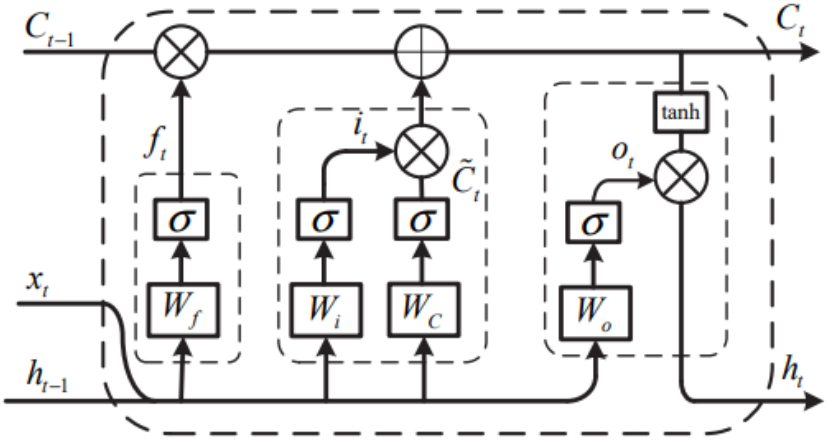
Εικόνα 14. Το επαναλαμβανόμενο στοιχείο RNN (Istiake Sunny, 2020).

Οι είσοδοι σε ένα τέτοιο δίκτυο είναι διαδοχικά δεδομένα όπως ήχος και κείμενο και γενικά δεδομένα τα οποία μεταβάλλονται με την διάρκεια του χρόνου.

Η αρχιτεκτονική του δικτύου δίνει την δυνατότητα βραχυπρόθεσμης μνήμης, αλλά αδυνατεί να διατηρήσει την πληροφορία μακροπρόθεσμα. Η μορφή αυτή του δικτύου παρουσιάζει επίσης προβλήματα. Ένα από αυτά είναι ο ρυθμός μάθησης[7], είτε όταν ελαχιστοποιείται καθώς μεταβάλλονται τα βάρη προς τις πρώτες στρώσεις του δικτύου είτε αποκτώντας τεράστιες τιμές (Hochreiter & Schmidhuber, 1997).

Τις αδυναμίες αυτές του δικτύου τύπου RNN τις ξεπερνάει το LSTM. Το νευρωνικό δίκτυο LSTM προσφέρει την δυνατότητα μακράς αλλά και βραχείας μνήμης, η οποίες δίνουν στο δίκτυο την ικανότητα να «θυμάται» αρκετά μεγάλες ακολουθίες. Ακόμη παρέχουν στο δίκτυο τη δυνατότητα να παραβλέπει η να ανακαλεί σημαντικές πληροφορίες καθώς εκπαιδεύεται (Hochreiter & Schmidhuber, 1997).

[7] Αριθμός, ο οποίος χρησιμοποιείται από αλγόριθμο βελτιστοποίησης για να ανανεώνει τα βάρη κατά τη διάρκεια της εκπαίδευσης του μοντέλου.



Εικόνα 15. Μονάδα LSTM, όπου πίνακας εισόδου, πίνακας κατάστασης, πίνακας μνήμης, πύλη

Η πύλη μνήμης $\boldsymbol{f_t}$ (Forget Gate):

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Αποφασίζει πόση πληροφορία εισόδου θα διατηρηθεί και αποθηκεύει την έξοδο (ποσοστό) στην κατάσταση του $\boldsymbol{c_t}$ (Li, 2018).

Η πύλη εισόδου $\boldsymbol{i_t}$ (Input Gate):

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Αποφασίζει πόση πληροφορία από την είσοδο και από την προηγούμενη κατάσταση του θα διατηρηθεί για να περάσει στην κατάσταση του στοιχείου (Li, 2018).

Τέλος, η πύλη εξόδου $\boldsymbol{O_t}$ (Output Gate):

Η έξοδος της πύλης θα είναι ένας πίνακας για τη νέα κατάσταση του στοιχείου

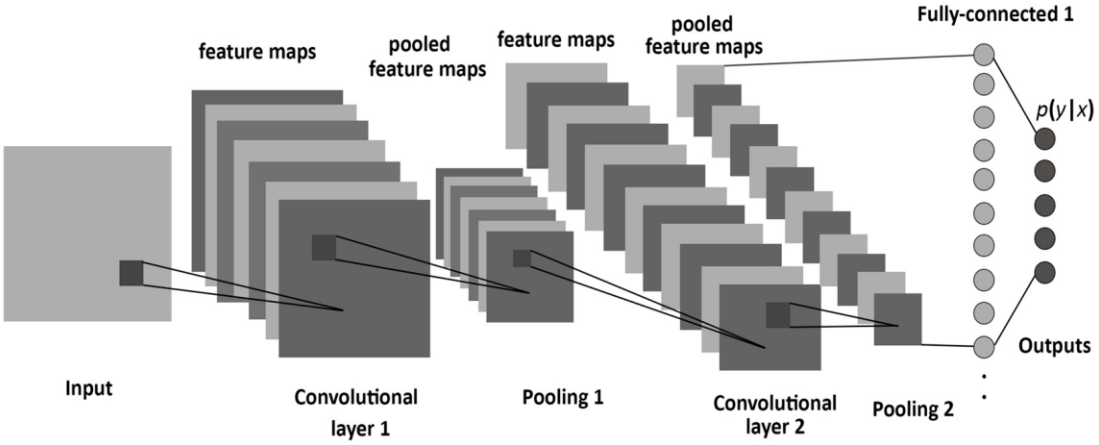
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

(Li, 2018).

2.4 CNN (Convolutional Neural Network)

Τα μοντέλα CNN αφορούν δεδομένα όπως εικόνες, είτε αυτές είναι δυο διαστάσεων δηλαδή εικόνες χωρίς χρώματα *(Σε κλίμακα του γκρι),* είτε με χρώματα τριών διαστάσεων συμπεριλαμβάνοντας τα 3 κανάλια *(BGR-RGB).* (Sarkar, Bali, & Sharma, 2018)

Η είσοδος σε ένα τέτοιου είδους μοντέλο, είναι η εικόνα χωρίς κάποια επεξεργασία[8]. Στην συνέχεια στα δεδομένα της εικόνας εφαρμόζονται κάποια φίλτρα *(Convolution Filter)* ο αριθμός και μέγεθος των οποίων ορίζεται από τον χρήστη. Το επόμενο βήμα είναι η περαιτέρω συμπίεση της πληροφορίας της εικόνας *(MaxPooling)*



Εικόνα 16. Δίκτυο CNN (Sarkar, Bali, & Sharma, 2018)

και η επανάληψη της διαδικασίας (Sarkar, Bali, & Sharma, 2018).

Πίνακας Εισόδου

2	5	2	1	8	9
8	9	6	3	2	1
9	8	4	1	1	0
2	5	0	1	5	6
0	6	0	7	1	9
7	8	9	5	7	8

(6x6)

Φίλτρο

	1	0	-1
(Συνέλιξη)	1	0	-1
(*)	1	0	-1
(Convolution)			

(3x3)

=

	A	B	C	D
1	7	17	1	-5
2	9	17	2	-2
3	7	10	-3	-6
4	0	6	-4	-10

ΘΕΣΗ: A1= ((1*2)+(1*8)+(1*9))+ ((0*5)+(0*9)+(0*8))+((-1*2)+(-1*6)+(-1*4))
ΘΕΣΗ: B1= ((1*5)+(1*9)+(1*8))+ ((0*2)+(0*6)+(0*4))+((-1*1)+(-1*3)+(-1*1))
ΘΕΣΗ: C1= ((1*2)+(1*6)+(1*4))+ ((0*1)+(0*3)+(0*1))+((-1*8)+(-1*2)+(-1*1))
ΘΕΣΗ: D1= ((1*1)+(1*3)+(1*1))+ ((0*8)+(0*2)+(0*1))+((-1*9)+(-1*1)+(-1*0))

Εικόνα 17. Η πρώτη διαδικασία θεωρείται η συνέλιξη μεταξύ πίνακα εισόδου και του πίνακα του φίλτρου. Δηλαδή το κάθε φίλτρο διατρέχει τον πίνακα εισόδου και πολλαπλασιάζεται με τις τιμές του και το νέο σημείο στον πίνακα εξόδου (4x4) *είναι το άθροισμα των τιμών αυτών.*

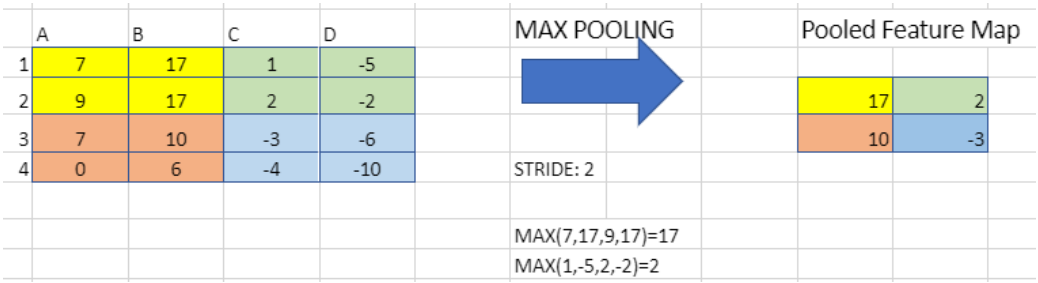
[8] Χωρίς επεξεργασία διότι αναφερόμαστε σε Βαθιά Ν.Δ. Το μοντέλο θα βρίσκει τα χαρακτηριστικά από μόνο του χωρίς κάποια υπόδειξη.

2	5	2
8	9	6
9	8	4

Εικόνα 18. Το φίλτρο στο συγκεκριμένο παράδειγμα είναι ένας πίνακας (3x3) *και η είσοδος είναι ένας πίνακας* (6x6). *Το φίλτρο ξεκινάει και διατρέχει τον πίνακα εισόδου από την αρχή του όπως φαίνεται στην εικόνα. Και κάθε φορά μεταφέρεται δεξιά κατά μια θέση, μέχρι να βρεθεί στο τέλος του φτιάχνοντας έτσι την πρώτη γραμμή του πίνακα εξόδου* [-5, -4, 0, 8]. *Συνεχίζει κατά μια θέση κάτω από την προηγουμένη πρώτη θέση κ.ο.κ.*

0	0	0	0	0	0	0	0
0	2	5	2	1	8	9	0
0	8	9	6	3	2	1	0
0	9	8	4	1	1	0	0
0	2	5	0	1	5	6	0
0	0	6	0	7	1	9	0
0	7	8	9	5	7	8	0
0	0	0	0	0	0	0	0

Εικόνα 19. Η πληροφορία που χάνεται από την συνέλιξη είναι οι τιμές στις άκρες του πίνακα εισόδου, και για αυτόν τον λόγο υπάρχει η δυνατότητα να προστεθεί η τιμή 0 *στις θέσεις γύρω από τις άκρες του πίνακα εισόδου*



Εικόνα 20. Υστέρα από την δημιουργία του πίνακα εξόδου (4x4) *εφαρμόζεται ένα επιπλέον φίλτρο που ονομάζεται "MaxPooling". Το οποίο δεν κάνει τίποτε άλλο από το να εξάγει την μέγιστη ή την μέση τιμή σε έναν χώρο στον πίνακα εισόδου λόγου χάρη* (2x2) *και ακολουθεί την ίδια λογική με το προηγούμενο βήμα όπου σαρώνει τον πίνακα εξόδου. Ως αποτέλεσμα να έχουμε έναν καινούργιο πίνακα (2x2) "Pooled feature map". Η διαδικασία αυτή μας βοηθάει να γενικευτούν τα χαρακτηριστικά που αφορούν τον πίνακα εισόδου ή την εικόνα εισόδου. Παρατήρηση η παράμετρος «Stride» ορίζει τα βήματα του φίλτρου.*

Εικόνα 21. Για να συνδεθούν οι πίνακες "Pooled Feature Map" με κάποιο πυκνό στρώμα νευρωνικού δικτύου πρέπει πρώτα να μετατραπούν σε άλλη μορφή πίνακα, μια ιδιότητα η οποία λέγεται "Flattening". Δηλαδή ο πίνακας στην εικόνα από (2x2) θα μετατραπεί σε (4x1).

2.5 Δυνατότητες και λύσεις υπερπροσαρμογής

Τα μοντέλα CNN έχουν την δυνατότητα να αναγνωρίζουν χαρακτηριστικά όπως κάθετες ή οριζόντιες ευθείες, γωνίες δηλαδή χαμηλού επιπέδου χαρακτηριστικά κυρίως στα πρώτα επίπεδα. Καθώς τα επίπεδα αυξάνονται, το μοντέλο δύναται να αναγνωρίσει, ανάλογα με τα δεδομένα εισόδου, περισσότερα πολύπλοκα χαρακτηριστικά (Shafkat, 2018).

Για την επίλυση προβλημάτων υπερπροσαρμογής του μοντέλου στα δεδομένα εισόδου, επιλέγεται η προσθήκη ενός επιπλέον στρώματος “Dropout", το οποίο ρυθμίζεται από τον χρήστη και λαμβάνει τιμές από 0 έως 1 που μεταφράζονται σε ποσοστό επί τοις εκατό (%). Κατά την διάρκεια της μάθησης, απενεργοποιείται τυχαία ένα ποσοστό νευρώνων/φίλτρων και δεν μεταβάλλονται τα βάρη σε αυτά (Wu & Gu) (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

Επίσης, εφαρμόζονται τεχνικές στα δεδομένα εισόδου οι οποίες έχουν στόχο τη μείωση υπερπροσαρμογής του μοντέλου και την αύξηση των δεδομένων, όπως η αλλαγή χρώματος, η μεγέθυνση και η περιστροφή γύρω από έναν άξονα *(Vertical/Horizontal Flip).* Η διαδικασία αυτή βοηθάει το μοντέλο να εκπαιδευτεί για να αναγνωρίζει καλυτέρα τα χαρακτηριστικά και να μην επικεντρώνεται σε μερικά από αυτά (Ruizendaal, 2017).

ΚΕΦΑΛΑΙΟ 3- ΠΕΙΡΑΜΑΤΙΚΕΣ ΕΦΑΡΜΟΓΕΣ ΑΛΓΟΡΙΘΜΩΝ

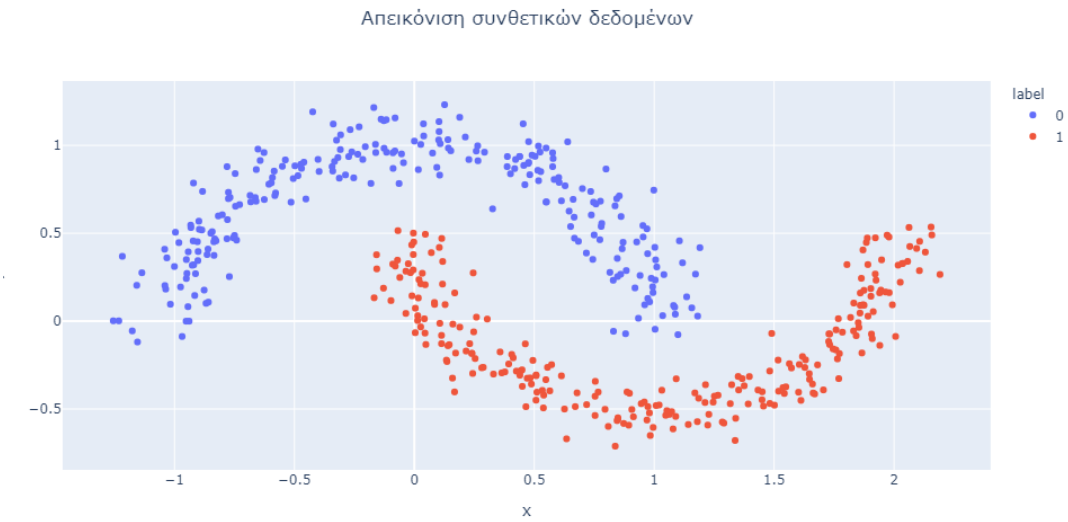
Σε αυτό το τμήμα της διπλωματικής εργασίας θα μελετηθούν οι εφαρμογές των αλγορίθμων που αναφέρθηκαν παραπάνω στην βιβλιογραφική ανασκόπηση. Οι αλγόριθμοι αυτοί πραγματεύονται την εύρεση ανωμαλιών και ελλειπόντων στοιχείων καθώς και την εφαρμογή σε μοντέλα βαθιάς μάθησης.

3.1 Εφαρμογές αλγορίθμων για ανίχνευση σφαλμάτων

Παρακάτω θα γίνουν εφαρμογές των αλγορίθμων χρησιμοποιώντας απλά σύνολα δεδομένων για την ανίχνευση ανωμαλιών/σφαλμάτων (DBSCAN, LOF, Isolation Forest, Tukey IQR), την ανίχνευση ελλειπόντων στοιχείων και συμπλήρωση αυτών. Τέλος θα γίνει μια απλή εφαρμογή μείωσης διαστάσεων ενός συνόλου δεδομένων (PCA).

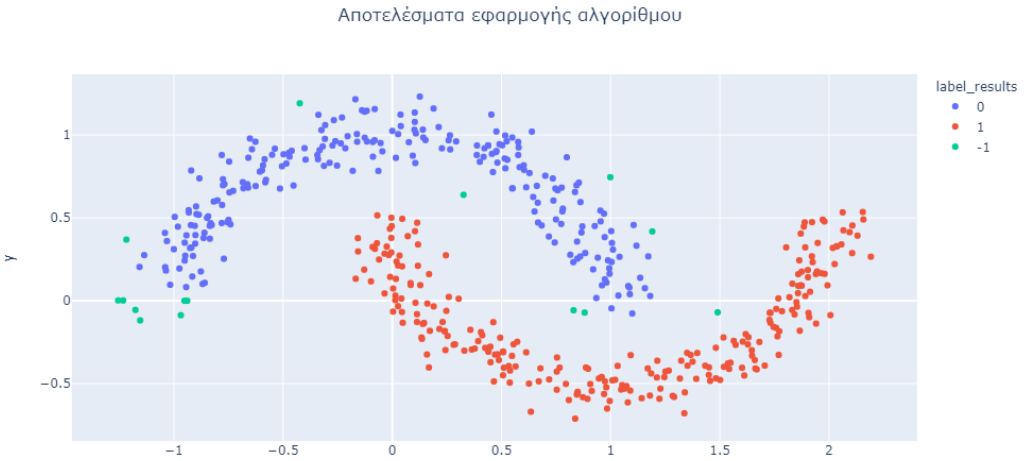
3.1.1 Εφαρμογή αλγορίθμου DBSCAN

Για το παράδειγμα του αλγορίθμου *DBSCAN* θα δημιουργηθεί ένα σύνολο συνθετικών δεδομένων χρησιμοποιώντας το *make_moons* από *Sklearn* βιβλιοθήκη.



Εικόνα 22. Απεικόνιση συνθετικών δεδομένων για 500 σημεία. Με ποσοστό θορύβου 10%.

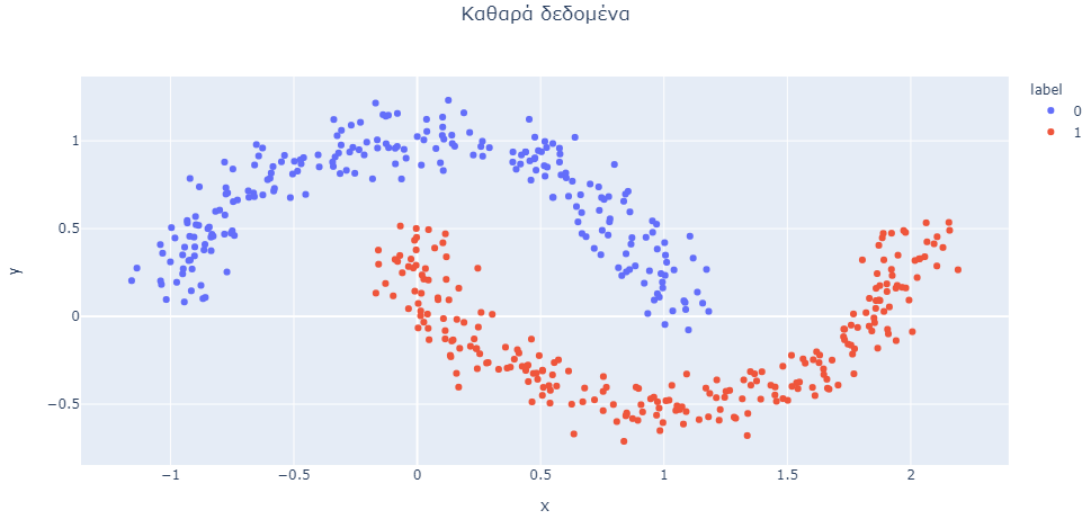
Οι παράμετροι που χρησιμοποιήθηκαν για τον αλγόριθμο είναι *eps: 0.17* και *min_samples: 15*. Στα αποτελέσματα όπως θα παρατηρήσουμε παρακάτω ο αλγόριθμος κατέγραψε 15 σημεία ως θόρυβο, το οποίο σημαίνει ότι τα σημεία θορύβου είναι μόλις 3% του συνολικού ποσοστού των δεδομένων.



Εικόνα 23. Αποτελέσματα αλγορίθμου

Ομάδα	Πλήθος
-1	15
0	236
1	249

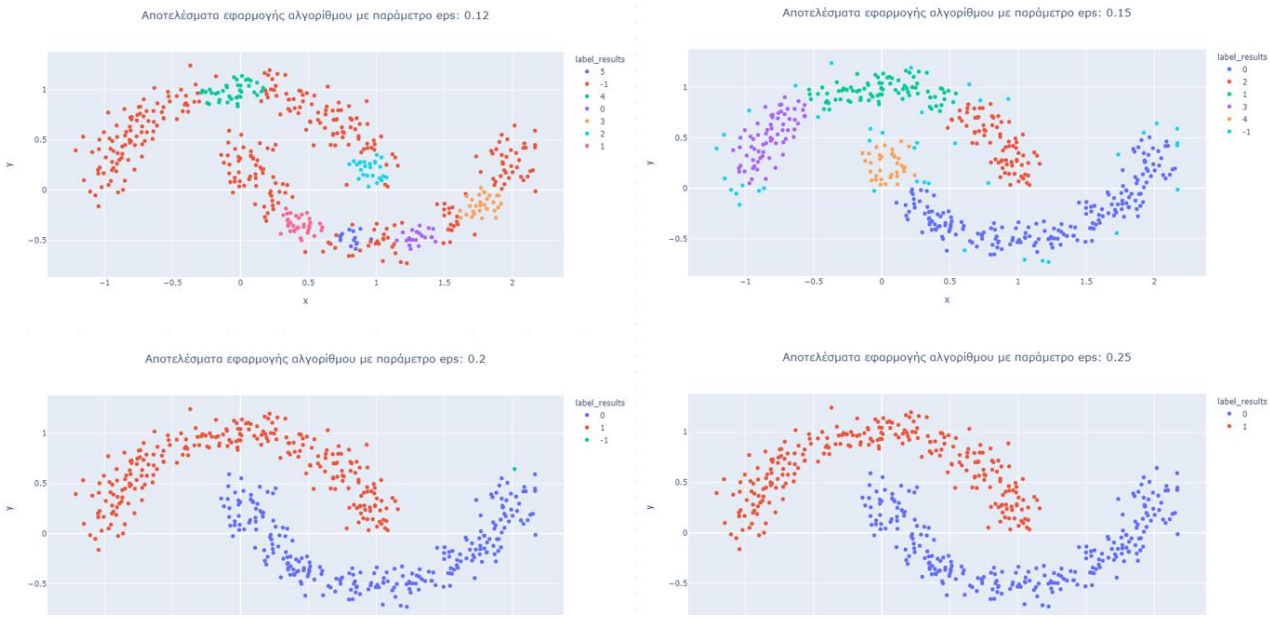
Εικόνα 24. Πλήθος και ομάδες που εντόπισε ο αλγόριθμος.



Εικόνα 25. Αφαίρεση σημείων θορύβου και απεικόνιση καθαρών δεδομένων.

Για το συγκεκριμένο παράδειγμα δεδομένων αφαιρέθηκε το 3% των αρχικών δεδομένων. Η πληροφορία αυτή μπορεί να χρησιμοποιηθεί σε περίπτωση που αφαιρεθεί ένα συγκεκριμένο ποσοστό τιμών θορύβου. Έτσι παρατηρείται η απόδοση του αλγορίθμου με τις επιλεγμένες τιμές παραμέτρων.

Σε νέα συνθετικά δεδομένα τα οποία δημιουργήθηκαν με τις τιμές παραμέτρου να είναι *eps* : 0.17 και *min_samples* : 15 βρέθηκαν 15 τιμές τις οποίες ο αλγόριθμος τις αναγνώρισε ως θόρυβο. Παρακάτω εξετάζεται το ενδεχόμενο αύξησης της τιμής *eps* διατηρώντας την τιμή *min_samples*.

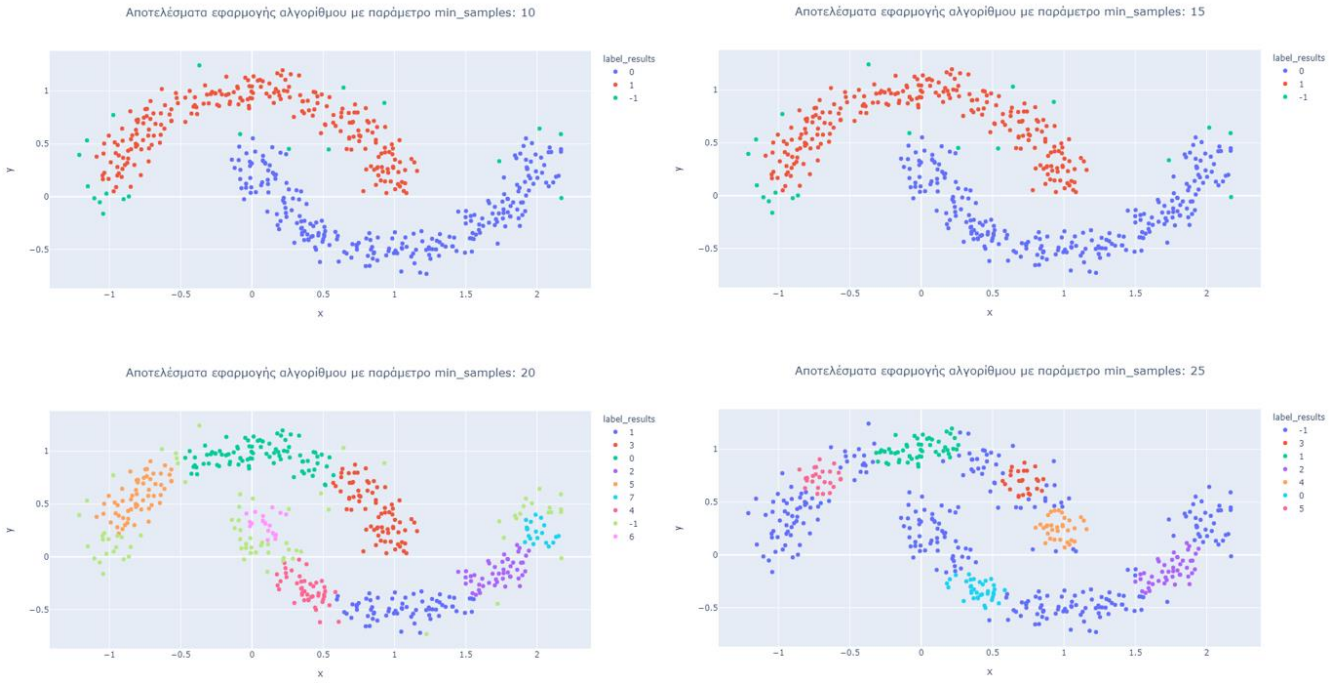


Εικόνα 26. Στα παραπάνω 4 γραφήματα εξετάζεται η διασπορά με τιμές eps 0.12 , 0.15, 0.2, 0.25, αντίστοιχα. Ο αλγόριθμος δεν κατάφερε να κάνει σωστή αναγνώριση των δεδομένων για τιμή eps 0.12. Κατά συνέπεια τα χώρισε σε μικρές ομάδες θεωρώντας ένα μεγάλο σύνολο τιμών στα δεδομένα ως θόρυβο. Για τιμή 0.15 παρατηρείται ότι ο αλγόριθμος χώρισε τα δεδομένα σε 5 ομάδες. Για τιμή 0.17 ο αλγόριθμος πρόβλεψε σωστά τις δυο ομάδες στα δεδομένα και αναγνώρισε επίσης, όπως και τις προηγούμενες φορές, τιμές θορύβου. Για τιμή 0.25 ο αλγόριθμος ομαδοποίησε ολόκληρο το σύνολο δεδομένων. Το τελευταίο έγινε διότι η τιμή απόστασης ήταν αρκετά μεγάλη έτσι ώστε ο αλγόριθμος να ομαδοποιήσει ολόκληρο το σύνολο δεδομένων.

eps: 0.12				eps: 0.15			
	Ομάδα	Πλήθος	Ποσοστό		Ομάδα	Πλήθος	Ποσοστό
	-1	335	67%		-1	41	8%
	0	21	4%		0	193	39%
	1	34	7%		1	85	17%
	2	25	5%		2	76	15%
	3	30	6%		3	69	14%
	4	40	8%		4	36	7%
	5	15	3%				
eps: 0.2				eps: 0.25			
	Ομάδα	Πλήθος	Ποσοστό		Ομάδα	Πλήθος	Ποσοστό
	-1	77	15%		0	250	50%
	0	87	17%		1	250	50%
	1	70	14%				
	2	48	10%				
	3	74	15%				
	4	45	9%				
	5	59	12%				
	6	20	4%				
	7	20	4%				

Εικόνα 27. Αποτελέσματα ομάδων και θορύβου για τις 4 διαφορετικές δοκιμές που πραγματοποιήθηκαν.

Αυτή την φορά θα διατηρηθεί η τιμή του *eps* : *0.17*, αλλάζοντας κάθε φορά την τιμή του *min_samples*.



Εικόνα 28. Για τιμή min_samples: 10 ο αλγόριθμος προέβλεψε σωστά τις δυο ομάδες στα δεδομένα και αναγνώρισε τις τιμές θορύβου. Για τιμή 15 παρατηρήθηκε ότι η συμπεριφορά του αλγορίθμου παρέμεινε σταθερή σε σχέση με την προηγούμενη τιμή (min_samples: 10). Για τις τιμές 20 και 25 ο αλγόριθμος χώρισε τα δεδομένα, που κανονικά θα έπρεπε να χωρίσει σε δυο ομάδες, σε πολλές μικρές ομάδες. Επιπλέον, τις περιοχές τιμών με αυξημένη διασπορά τις έχει αναγνωρίσει ως θόρυβο.

min_samples: 10			
	Ομάδα	Πλήθος	Ποσοστό
	-1	20	4%
	0	244	49%
	1	236	47%

min_samples: 15			
	Ομάδα	Πλήθος	Ποσοστό
	-1	20	4%
	0	244	49%
	1	236	47%

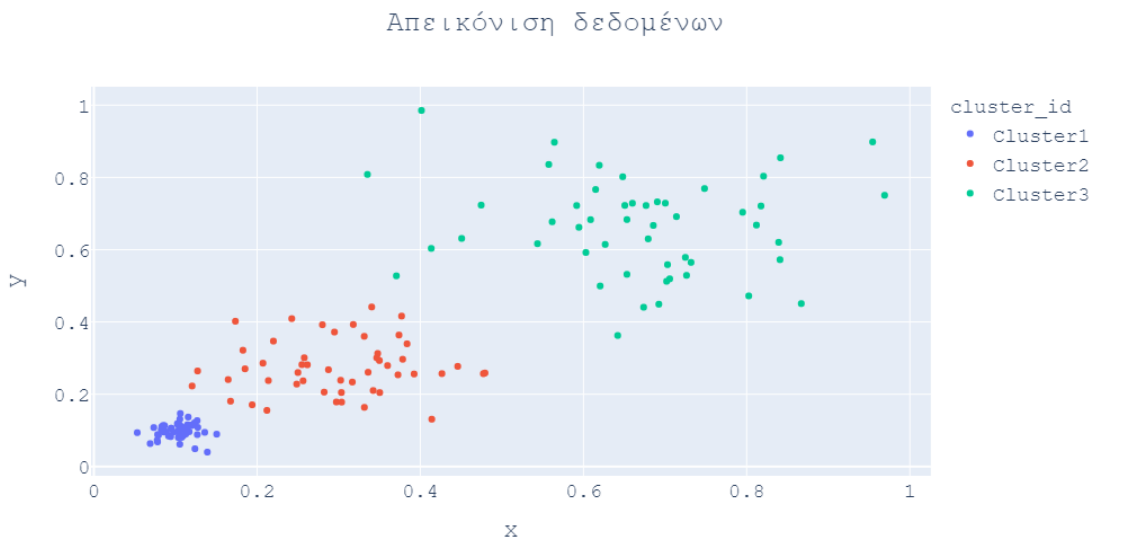
min_samples: 20			
	Ομάδα	Πλήθος	Ποσοστό
	-1	77	15%
	0	87	17%
	1	70	14%
	2	48	10%
	3	74	15%
	4	45	9%
	5	59	12%
	6	20	4%
	7	20	4%

min_samples: 25			
	Ομάδα	Πλήθος	Ποσοστό
	-1	283	57%
	0	36	7%
	1	53	11%
	2	45	9%
	3	25	5%
	4	33	7%
	5	25	5%

Εικόνα 29. Αποτελέσματα ομάδων και θορύβου για τις 4 διαφορετικές δοκιμές που πραγματοποιήθηκαν.

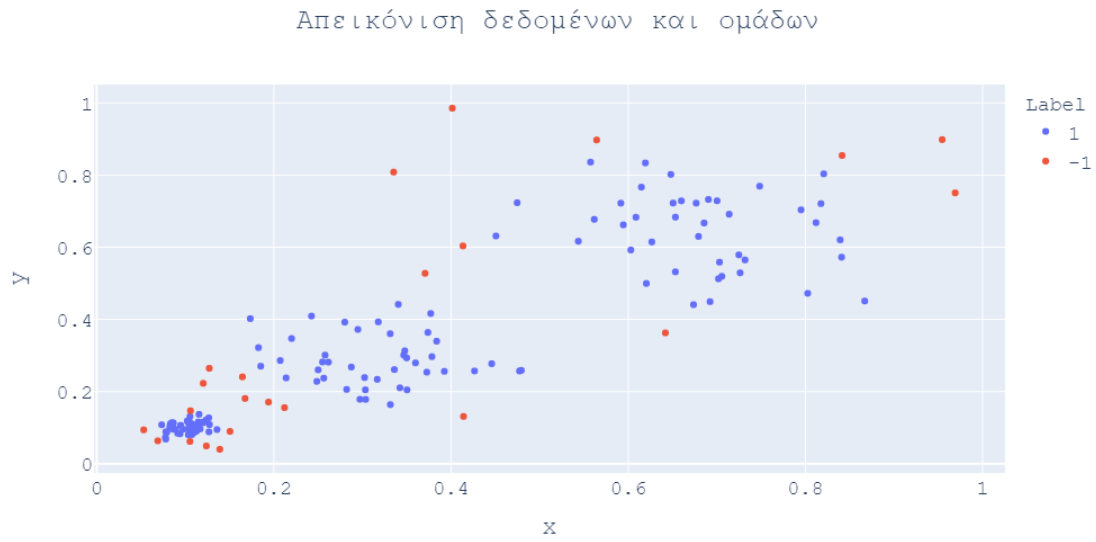
3.1.2 Εφαρμογή αλγορίθμου *LOF*

Τα δεδομένα[9] που θα χρησιμοποιηθούν για το παράδειγμα είναι τρεις ομάδες συνθετικών δεδομένων με διαφορετική πυκνότητα και συνολικά 150 τιμές.



Εικόνα 30. Απεικόνιση δεδομένων.

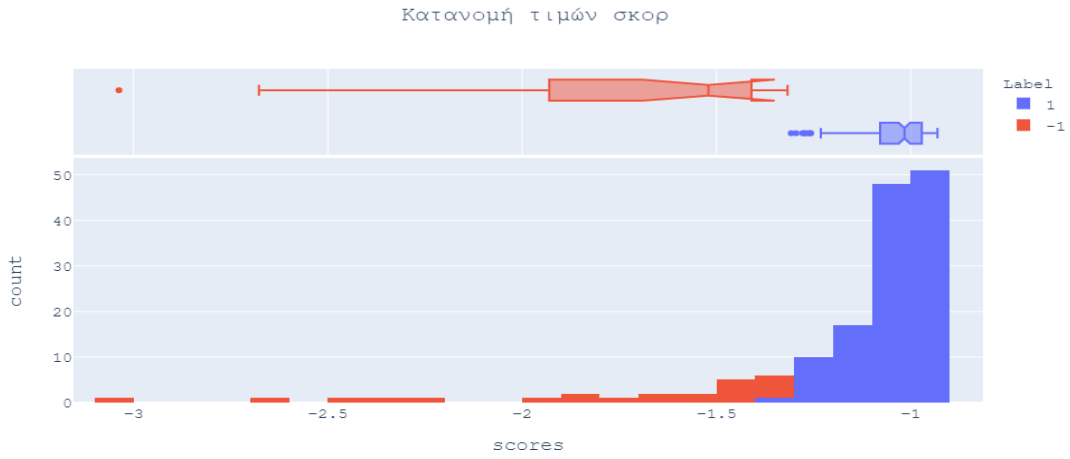
Έγινε εφαρμογή του αλγορίθμου με τις τιμές παραμέτρων να είναι *n_neighbors: 20* και *contamination: 0.15* (15%).



Εικόνα 31. Απεικόνιση αποτελεσμάτων αλγορίθμου LOF.

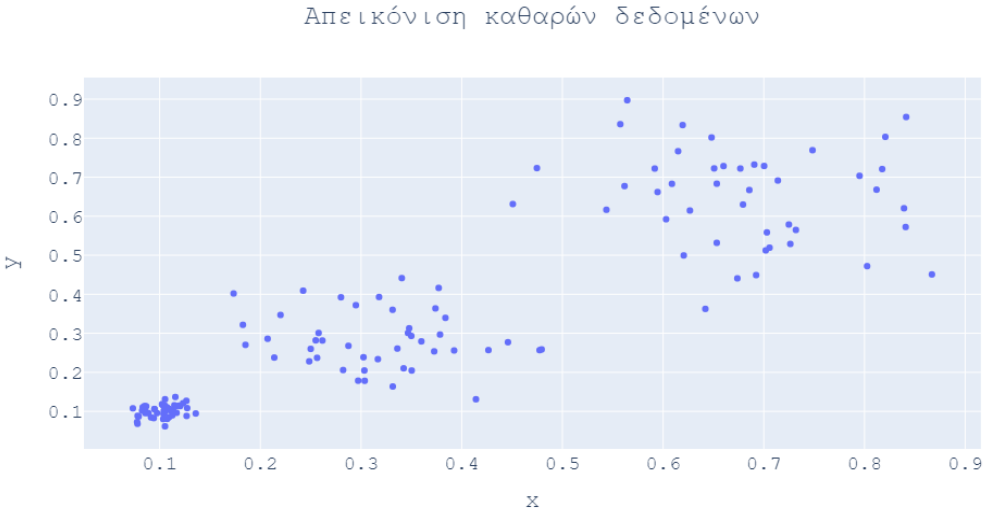
[9] Τα δεδομένα βρίσκονται στο : https://github.com/elki-project/elki/blob/master/data/synthetic/ABC-publication/pov.csv

Ο αλγόριθμος πρόβλεψε 23 τιμές στο σύνολο δεδομένων ως σφάλματα, το οποίο είναι το 15.3% του συνόλου, και υπήρχε μια μικρή απόκλιση από την παράμετρο *contamination*, η οποία όμως κυμαίνεται στα επιτρεπτά όρια. Για την καλύτερη κατανόηση επίλογων του αλγορίθμου θα διερευνηθεί η κατανομή του πίνακα των σκορ (model.negative_outlier_factor_).



Εικόνα 32. Απεικόνιση κατανομή τιμών σκορ. Το κόκκινο χρώμα αντιπροσωπεύει τα σημεία θορύβου και το μπλε τα φυσιολογικά.

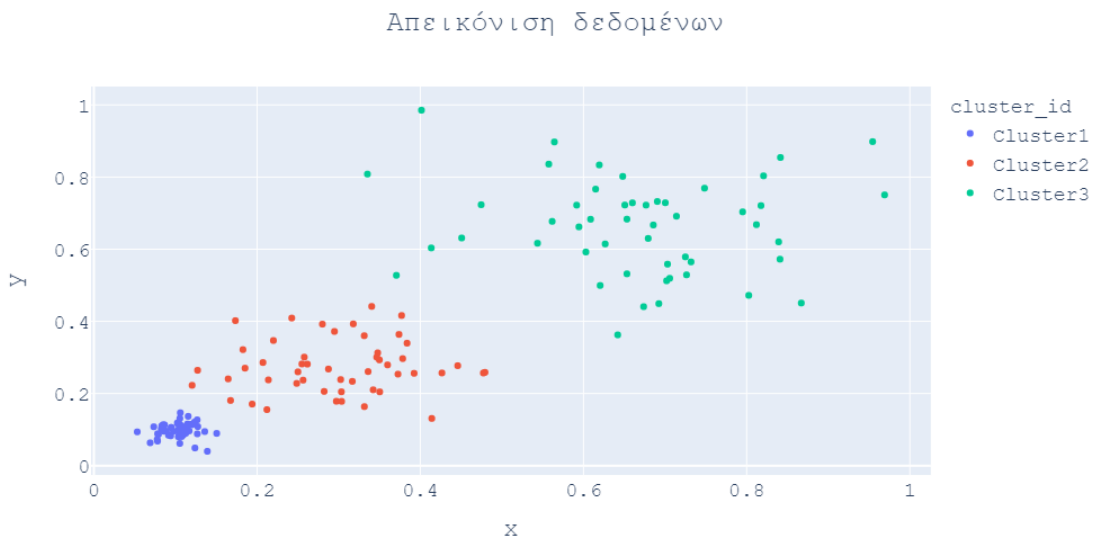
Σύμφωνα με τον πίνακα των σκορ, δεν παρατηρούνται σφάλματα στην λειτουργία του αλγορίθμου. Τέλος, στην περίπτωση που ο χρήστης είναι ικανοποιημένος για τις προβλέψεις του αλγορίθμου τότε μένει μόνο η αφαίρεση των προβλεπόμενων σφαλμάτων.



Εικόνα 33. Απεικόνιση των καθαρών τιμών έπειτα από αφαίρεση εκείνων με τιμές σκορ μικρότερες από -1.4.

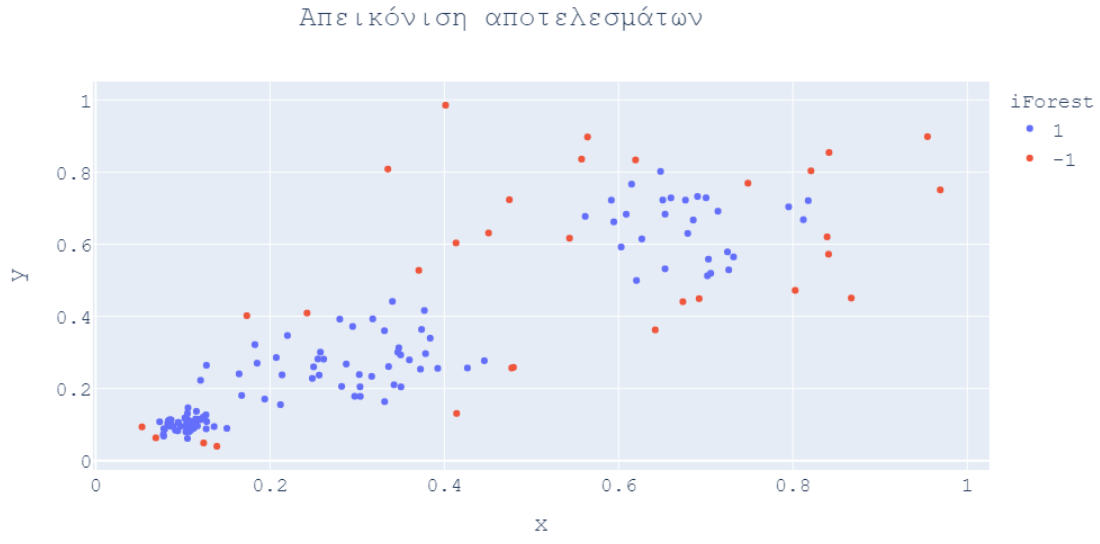
3.1.3 Εφαρμογή αλγορίθμου Isolation Forest

Για την εφαρμογή του αλγορίθμου χρησιμοποιήθηκαν τα ίδια δεδομένα με τον αλγόριθμο LOF.



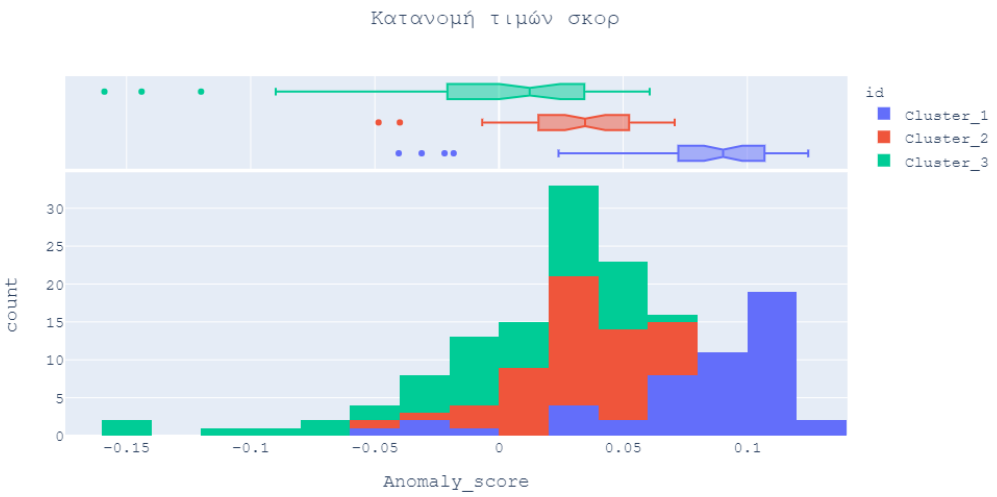
Εικόνα 34. Απεικόνιση δεδομένων.

Για την εφαρμογή μη επιβλεπόμενη προσέγγισης του αλγορίθμου η παράμετρος *contamination* ορίστηκε ως "*auto*", δηλαδή ο αλγόριθμος θα προσπαθήσει μόνος του να βρει τα ανώμαλα σημεία. Για τις παραμέτρους n_estimators: 100 και max_samples: 150.



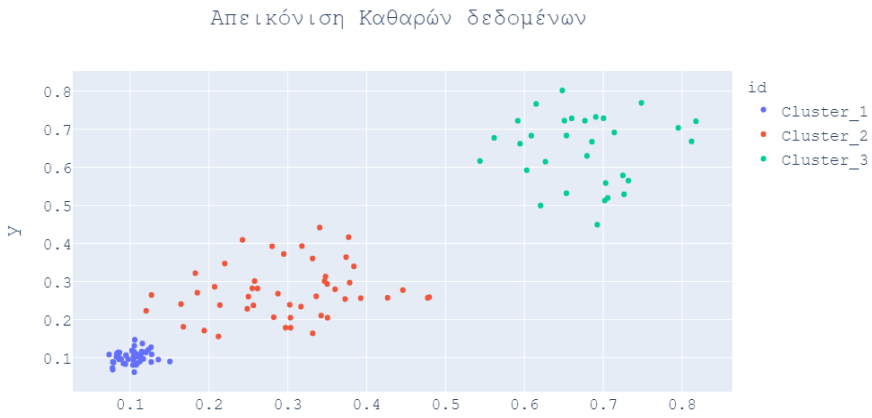
Εικόνα 35. Απεικόνιση αποτελεσμάτων του αλγορίθμου..

Ο αλγόριθμος πρόβλεψε 119 τιμές ως φυσιολογικές και 31 ως σφάλματα. Παρακάτω θα δούμε την επιβλεπόμενη προσέγγιση η οποία θα δώσει την δυνατότητα να ορίσει ο χρήστης πόσα και ποια σημεία θα αφαιρεθούν από το σύνολο δεδομένων με βάση τον πίνακα σκορ.



Εικόνα 36. Απεικόνιση κατανομή τιμών σκορ σε σχέση με την ομάδα που ανήκουν.

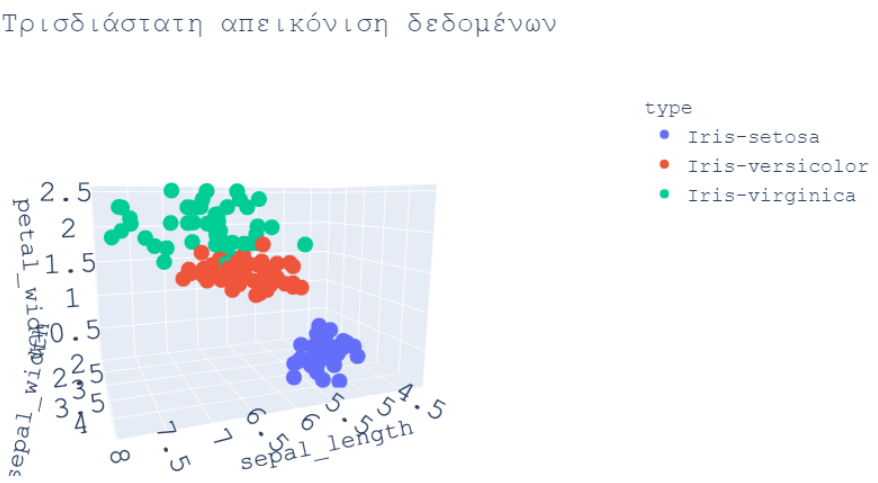
Σύμφωνα με την παραπάνω εικόνα μας δίνεται η δυνατότητα να ορίσουμε τα όρια των τιμών σκορ που θα θεωρηθούν σφάλματα σε κάθε ομάδα των δεδομένων. Δηλαδή για την πρώτη ομάδα (*Cluster_1*) θα μπορούσαμε να ορίσουμε το μηδέν και ότι είναι μικρότερο από αυτό να θεωρηθεί ως σφάλμα , για την δεύτερη ομάδα (*Cluster_2*) -0.025 ή -0.03 και για την τρίτη ομάδα (*Cluster_3*) -0.05. Έτσι ο χρήστης έχει την δυνατότητα επιλογής των σημείων που θα αφαιρεθούν. Παρακάτω θα γίνει η αφαίρεση τιμών για τιμές σκορ μικρότερο από -0.01 για όλες τις ομάδες.



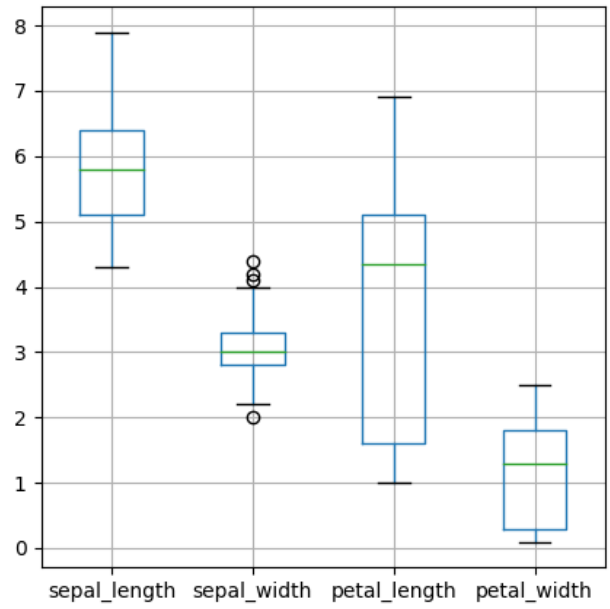
Εικόνα 37. Απεικόνιση των καθαρών τιμών έπειτα από αφαίρεση εκείνων με τιμές σκορ μικρότερες από -0.01.

3.1.4 Εφαρμογές αλγορίθμου Tukey, IQR

Παρακάτω θα γίνει η εφαρμογή της μεθόδου αυτού , και μέσω των Tukey Box Plot και μέσω την χρήση του διατεταρτημοριακού διαστήματος (IQR). Θα χρησιμοποιηθεί το σύνολο δεδομένων «*Iris Dataset*[10]».



Εικόνα 38. Απεικόνιση δεδομένων.



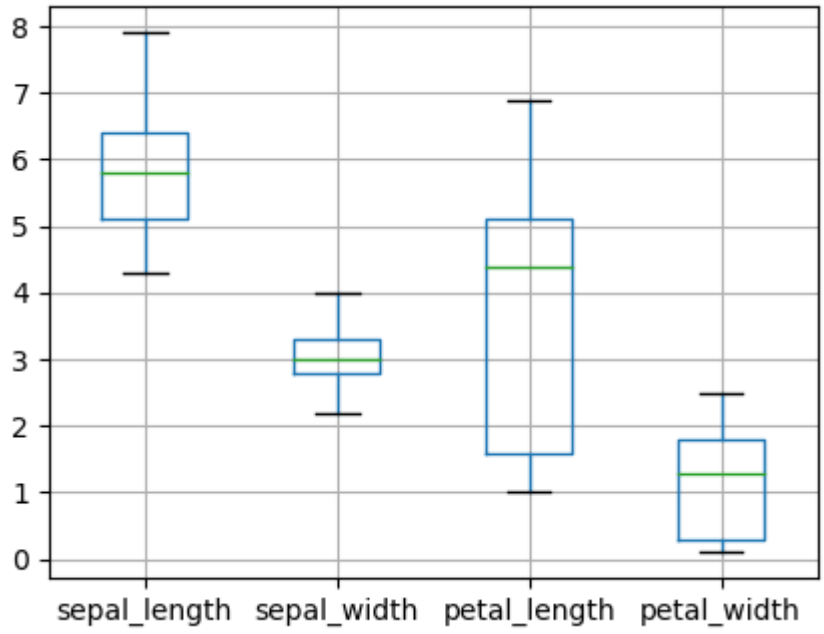
Εικόνα 39. Απεικόνιση διαγράμματος του Box-Plot.

[10] Iris Dataset: https://archive.ics.uci.edu/ml/datasets/iris

Όπως παρατήσουμε παραπάνω η στήλη *sepal_width* σύμφωνα με το *box-plot* έχει σημεία εκτός ορίων. Παρακάτω θα εφαρμοστεί φίλτρο μέσω *Tukey IQR*, με τιμή κατώτατου ορίου (*Lower Threshold*): 2.05 και ανώτατου (*Upper Threshold*): 4.05.

	sepal_length	sepal_width	petal_length	petal_width	type
15	5.7	4.4	1.5	0.4	Iris-setosa
32	5.2	4.1	1.5	0.1	Iris-setosa
33	5.5	4.2	1.4	0.2	Iris-setosa
60	5.0	2.0	3.5	1.0	Iris-versicolor

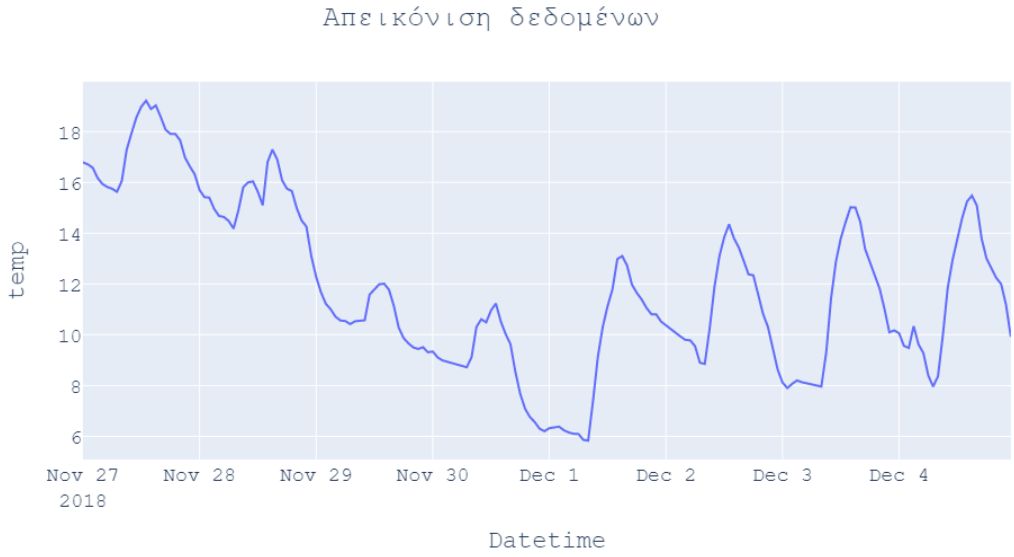
Εικόνα 40. Πίνακας με τα στοιχεία που βρέθηκαν από το φίλτρο.



Εικόνα 41. Απεικόνιση διαγράμματος του Box-Plot με τα καθαρά δεδομένα.

3.1.5 Παράδειγμα αντιμετώπισης ελλειπόντων στοιχείων χρονοσειράς

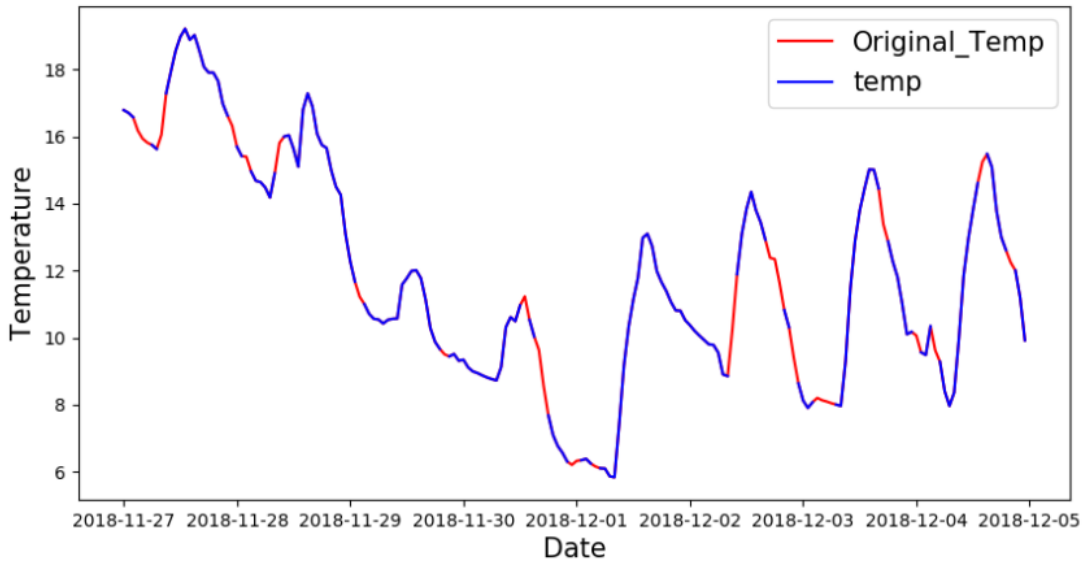
Θα διερευνηθούν οι διάφοροι τρόποι αντιμετώπισης ελλειπόντων στοιχειών. Τα δεδομένα[11] αφορούν τιμές θερμοκρασίας για το διάστημα *[27-11-2018 – 05-12-2018]* στον νομό Αττικής.



Εικόνα 42. Απεικόνιση δεδομένων.

Στα δεδομένα θα γίνει η αφαίρεση ενός τυχαίου πλήθους τιμών και στην συνέχεια θα γίνει προσπάθεια αντικατάστασης τους με διάφορες μεθόδους.

[11] https://www.meteoblue.com/en/weather/archive/export/athens_greece_264371



Εικόνα 43. Απεικόνιση δεδομένων, έγινε η αφαίρεση 27 τυχαίων τιμών στα δεδομένα θερμοκρασίας. Το κόκκινο χρώμα αντιπροσωπεύει τα σημεία που αφαιρέθηκαν και το μπλε τα πραγματικά.

Μέθοδος	Μέσο απόλυτο σφάλμα
ffill (Forward Fill)	6.85%
bfill (Backward fill)	6.50%
Rolling Mean	3.69%
Interpolate	2.33%
Interpolate (pchip)	2.08%
Interpolate (2nd Order)	10.62%
Interpolate (3rd Order)	2.17%
Interpolate (Quadratic)	2.12%
Interpolate (Cubic)	2.17%

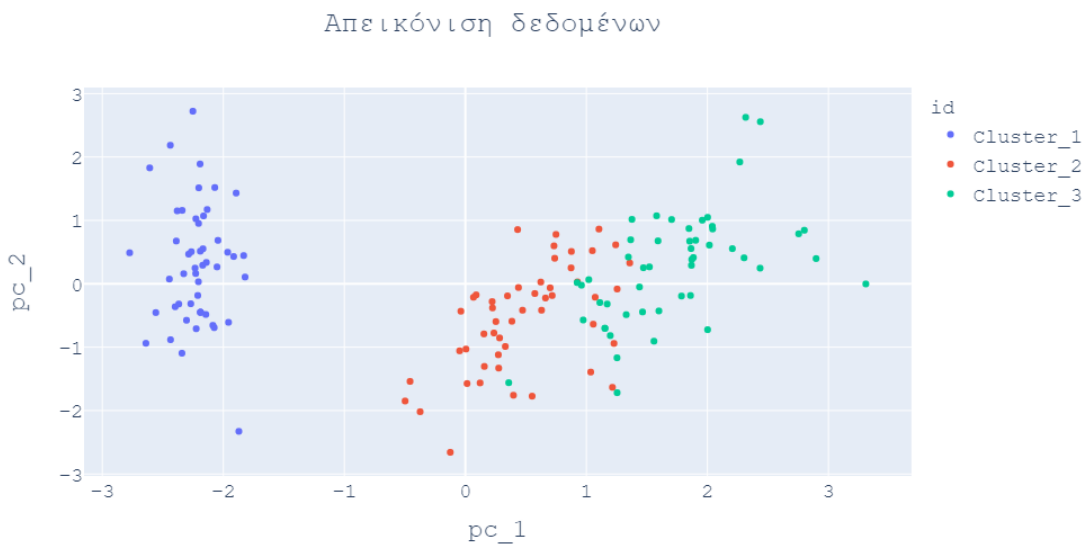
Εικόνα 44. Αποτελέσματα διαφόρων μεθόδων για την αντικατάσταση των τιμών που αφαιρέθηκαν.

3.1.6 Εφαρμογή αλγορίθμου PCA

Εφαρμογή *PCA* στο σύνολο δεδομένων *Iris_Dataset*[12]. Στο παράδειγμα αρχικά θα εφαρμοστεί η μέθοδος χωρίς την βοήθεια τις βιβλιοθήκης *sklearn*, και έπειτα με την βοήθεια της βιβλιοθήκης (Usman, 2018).

Τα βήματα που ακολουθήθηκαν για την εύρεση των κύριων συνιστωσών (*PC*):

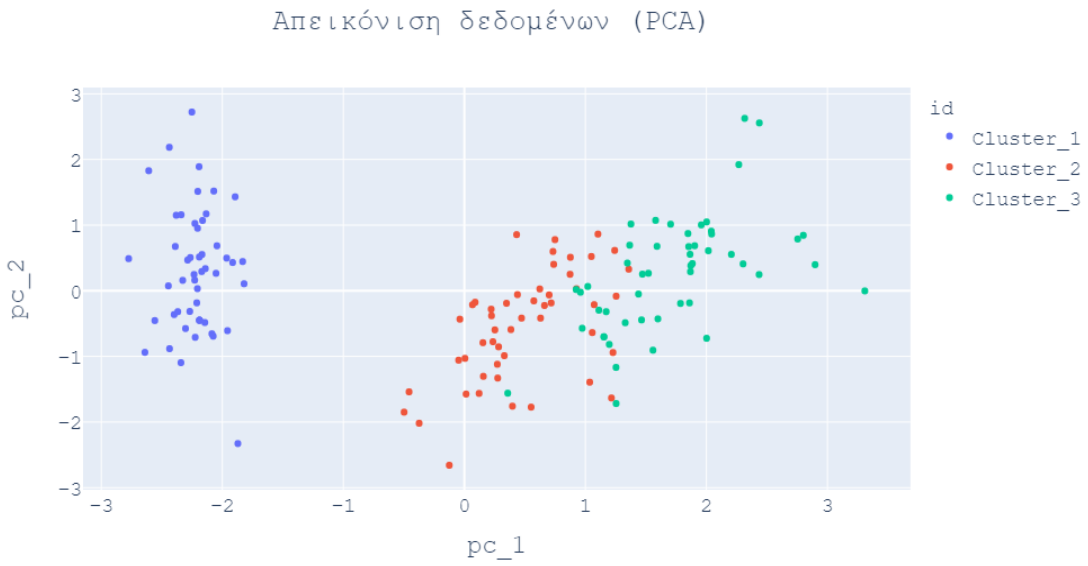
1. Κανονικοποίηση δεδομένων μέσω StandardScaler()
2. Εύρεση ανάστροφου πίνακα
3. Εύρεση πίνακα συνδιακύμανσης
4. Εύρεση ιδιοτιμών και ιδιοδιανυσμάτων
5. Υπολογισμός νέου συνόλου δεδομένων, χρησιμοποιώντας 2 PC



Εικόνα 45. Απεικόνιση δεδομένων χρησιμοποιώντας 2 PC.

[12] https://archive.ics.uci.edu/ml/datasets/iris

Εφαρμογή με την βοήθεια της βιβλιοθήκης *sklearn* με παράμετρο n_*components*: 2 (δυο *PC*).



Εικόνα 46. Απεικόνιση αποτελεσμάτων με την χρήση βιβλιοθήκης sklearn. Το συνολικό ποσοστό για τα 2 PC είναι 95.8% .

3.2 Εφαρμογές αλγορίθμων βαθιάς μάθησης

Στο πρώτο παράδειγμα θα γίνει μια εφαρμογή του **Estimator API** της βιβλιοθήκης **Keras**. Στην συνέχεια θα χρησιμοποιηθούν τα απλοποιημένα **LSTM** κελιά, **SimpleRNN** και **Bidirectional LSTM**.

3.2.1 Παράδειγμα του Estimator API για γραμμική παλινδρόμηση

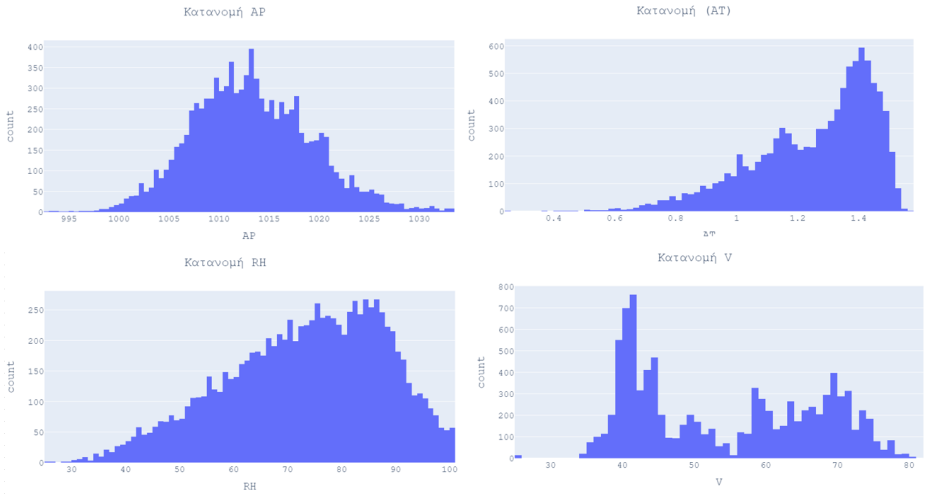
Τα δεδομένα: https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant (Tüfekci, 2014).

Πληροφορίες για τα δεδομένα (ωριαίες):

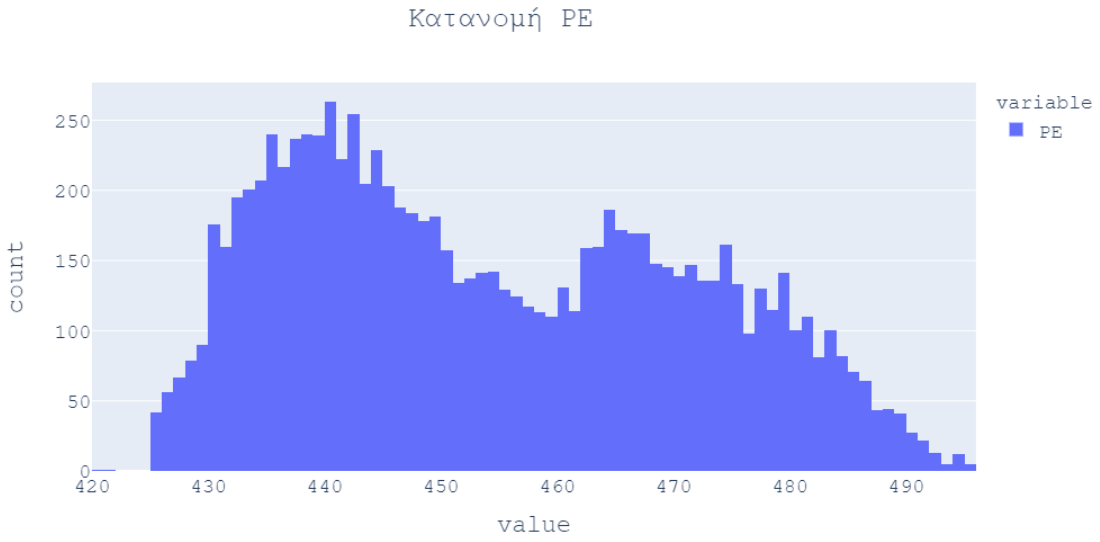
- *Θερμοκρασία (T) στο διάστημα [1,81°C - 37,11°C],*
- *Πίεση περιβάλλοντος (AP) στο διάστημα [992.89-1033.30] millibar,*
- *Σχετική υγρασία (RH) στο εύρος 25,56% έως 100,16%*
- *Εξάτμιση κενού (V) στο διάστημα [25,36-81,56] cm Hg*
- *Καθαρή ωριαία έξοδος ηλεκτρικής ενέργειας (EP) στο διάστημα [420,26-495,76]MW*

Οι μέσοι όροι λαμβάνονται από διάφορους αισθητήρες που βρίσκονται γύρω από το εργοστάσιο και καταγράφουν τις μεταβλητές περιβάλλοντος κάθε δευτερόλεπτο. Οι μεταβλητές δίνονται χωρίς κανονικοποίηση.

Στόχος του μοντέλου είναι η πρόβλεψη κατανάλωσης καθαρής ηλεκτρικής ενέργειας (*EP*). Στα δεδομένα δεν παρατηρήθηκαν ελλιπείς τιμές, παρακάτω η απεικόνιση κατανομής των δεδομένων.



Εικόνα 47. Απεικόνιση κατανομής.



Εικόνα 48. Απεικόνιση κατανομής στόχου (target).

Τα βήματα για την υλοποίηση του μοντέλου:

- Έλεγχος κατανομής και συσχέτισης δεδομένων
- Χωρισμός δεδομένων σε 70% για χρήση προς την εκπαίδευση και 30% για την δοκιμή του μοντέλου μέσω *train_test_split().*
- Κανονικοποίηση δεδομένων εκπαίδευσης μέσω *StandarScaler*()
- Δημιουργία και συγχώνευση των *'Feature Columns'*
- Ορισμός της συνάρτησης εκπαίδευσης *'Input Training Function'*
- Ορισμός της συνάρτησης πρόβλεψης *'Prediction Function'*
- Ορισμός μοντέλου *'Estimator Model'*
- Εκπαίδευση του μοντέλου
- Προβλέψεις
- Σύγκριση προβλεπόμενων τιμών με των πραγματικών
- Σφάλματα

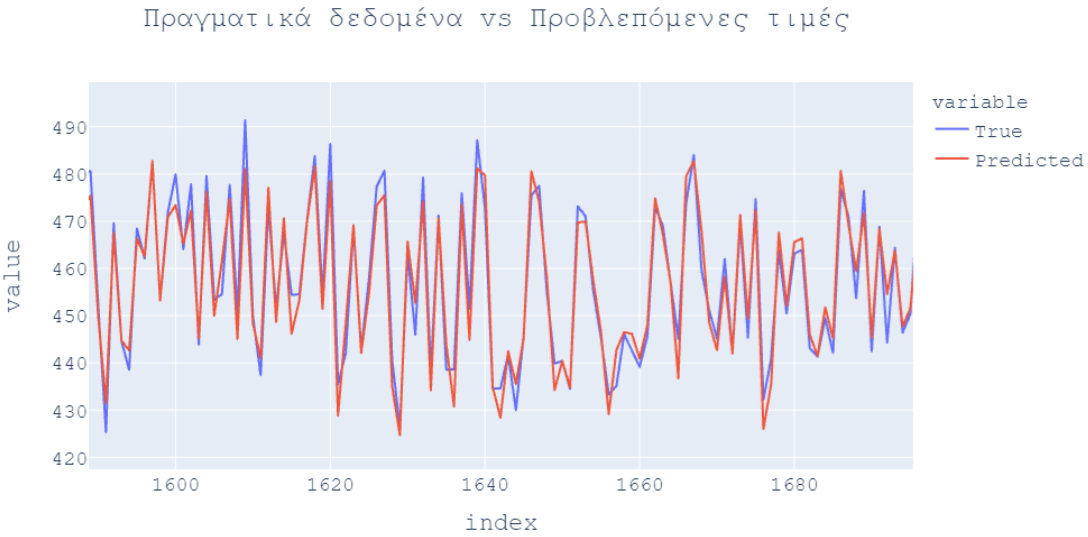
Το μοντέλο είχε μέσο τετραγωνικό σφάλμα (*RMSE*): 4.5 και $\mathbf{R}^2$ σκορ: 91%.

	AT	V	AP	RH	PE
count	9568.000000	9568.000000	9568.000000	9568.000000	9568.000000
mean	19.651231	54.305804	1013.259078	73.308978	454.365009
std	7.452473	12.707893	5.938784	14.600269	17.066995
min	1.810000	25.360000	992.890000	25.560000	420.260000
25%	13.510000	41.740000	1009.100000	63.327500	439.750000
50%	20.345000	52.080000	1012.940000	74.975000	451.550000
75%	25.720000	66.540000	1017.260000	84.830000	468.430000
max	37.110000	81.560000	1033.300000	100.160000	495.760000

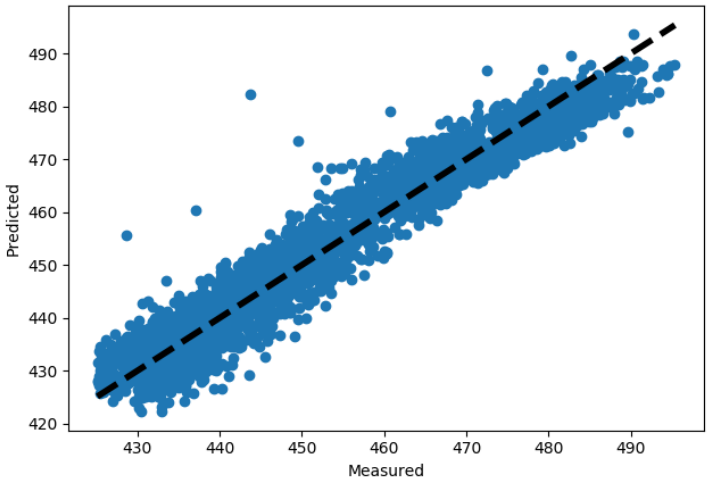
Εικόνα 49. Στατιστικές πληροφορίες των δεδομένων.

	AT	V	AP	RH	PE
AT	1.000000	0.844107	-0.507549	-0.542535	-0.948128
V	0.844107	1.000000	-0.413502	-0.312187	-0.869780
AP	-0.507549	-0.413502	1.000000	0.099574	0.518429
RH	-0.542535	-0.312187	0.099574	1.000000	0.389794
PE	-0.948128	-0.869780	0.518429	0.389794	1.000000

Εικόνα 50. Πίνακας συσχέτισης (Pearson).



Εικόνα 51. Απεικόνιση αποτελεσμάτων του μοντέλου.



Εικόνα 52. Απεικόνιση του γραμμικού μοντέλου

3.2.2 Παράδειγμα χρήσης του *Keras* για μοντέλο ταξινόμησης.

Τα δεδομένα: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

(Moro, 2014)

Πληροφορίες για τα δεδομένα:

Δεδομένα τράπεζας του πελάτη:

1. Age (αριθμητική)
2. job: τύπος εργασίας (κατηγορηματικός: «διαχειριστής», «γραφείου», «επιχειρηματίας», «υπηρέτρια», «διαχείριση», «συνταξιούχος», «αυτοαπασχολούμενος», «γενικές υπηρεσίες», «φοιτητής», «τεχνικός», «άνεργος », «άγνωστο»)
3. marital: οικογενειακή κατάσταση (κατηγορηματική: «διαζευγμένη», «παντρεμένη», «ανύπαντρη», «άγνωστη» · σημείωση: «διαζευγμένη» σημαίνει διαζευγμένη ή χήρα)
4. education: (κατηγορική: 'basic.4y', 'basic.6y', 'basic.9y', ‘λύκειο’, 'αναλφάβητοι', 'professional.course', 'university.degree', 'άγνωστο')
5. default: η πίστωση είναι προεπιλεγμένη; (κατηγορηματικό: «όχι», «ναι», «άγνωστο»)
6. housing: έχει στεγαστικό δάνειο; (κατηγορηματικό: «όχι», «ναι», «άγνωστο»)
7. loan: έχει προσωπικό δάνειο; (κατηγορηματικό: «όχι», «ναι», «άγνωστο»)

Σχετικές πληροφορίες με την τελευταία επαφή της τρέχουσας καμπάνιας:

8. contact: τύπος επικοινωνίας επαφής (κατηγορηματικός: «κινητό», «τηλέφωνο»)
9. month: τελευταίος μήνας επαφής του έτους (κατηγορηματικός: «jan», «feb», «mar»…, «nov», «dec»)
10. day_of_week: τελευταία ημέρα επαφής (κατηγορηματική: 'mon', 'tue', 'wed', 'thu', 'fri')
11. duration: τελευταία διάρκεια επαφής, σε δευτερόλεπτα (αριθμητικά). Σημαντική σημείωση: αυτό το χαρακτηριστικό επηρεάζει ιδιαίτερα τον στόχο εξόδου (π.χ. εάν η διάρκεια = 0 τότε y = 'όχι'). Ωστόσο, η διάρκεια δεν είναι γνωστή πριν πραγματοποιηθεί μια κλήση. Επίσης, μετά το τέλος της κλήσης το y είναι προφανώς γνωστό.

Άλλα γενικά χαρακτηριστικά:

12. campaign: αριθμός επαφών που πραγματοποιήθηκαν κατά τη διάρκεια αυτής της καμπάνιας και για αυτόν τον πελάτη (αριθμητικός, περιλαμβάνει την τελευταία επαφή)
13. pdays: αριθμός ημερών που πέρασαν μετά την τελευταία επικοινωνία με τον πελάτη από προηγούμενη καμπάνια (αριθμητικό; 999 σημαίνει ότι ο πελάτης δεν είχε επικοινωνήσει προηγουμένως
14. previous: αριθμός επαφών που πραγματοποιήθηκαν πριν από αυτήν την καμπάνια και για αυτόν τον πελάτη (αριθμητικό)
15. poutcome: αποτέλεσμα της προηγούμενης καμπάνιας μάρκετινγκ (κατηγορηματικό: «αποτυχία», «ανύπαρκτο», «επιτυχία»)

Χαρακτηριστικά κοινωνικού και οικονομικού πλαισίου:

16. emp.var.rate: ποσοστό διακύμανσης της απασχόλησης-τριμηνιαίος δείκτης (αριθμητικός)
17. cons.price.idx: δείκτης τιμών καταναλωτή - μηνιαίος δείκτης (αριθμητικός)
18. cons.conf.idx: δείκτης εμπιστοσύνης καταναλωτή -μηνιαίος δείκτης (αριθμητικός)
19. euribor3m: επιτόκιο 3 μηνών - ημερήσιος δείκτης (αριθμητικός)
20. nr.employed: αριθμός υπαλλήλων - τριμηνιαίος δείκτης (αριθμητικός)

Μεταβλητή εξόδου (επιθυμητός στόχος):

21. y - ο πελάτης έχει εγγράψει προθεσμιακή κατάθεση; (δυαδικό: «ναι», «όχι»)

Σύμφωνα με Moro et al., που διερευνήθηκαν οι επιδόσεις μοντέλων προβλέψεις για τα συγκεκριμένα δεδομένα, παρατηρήθηκε πως η στήλη "*duration*" που αφορούσε την διάρκεια των κλήσεων μεταξύ υπαλλήλων της τράπεζας με τους πελάτες είχε άμεση συσχέτιση με τον στόχο του μοντέλου. Για αυτόν τον λόγο θα γίνει δοκιμή με και χωρίς την στήλη αυτή και θα ελέγχουν τα αποτελέσματα. Να σημειωθεί ότι οι ερευνητές βρήκαν μοντέλο με σκορ ***0.926*** *AUC*.

Γενικές παρατηρήσεις για τα δεδομένα:

- Στήλη: *'previous'* έχει ~86% μηδενικές τιμές
- Στήλη: *'pdays'* Συχνότητα τιμής :'999' 39673
- Στήλη: *'default'* { 'no': 32588 , 'unknown': 8597 'yes': 3 }
- Στήλη: *'poutcome'* Συχνότητα 'nonexistent': 35563
- Στήλη: *'y' Target*/*Label* Είναι ασύμμετρη με συχνότητα (0: 36548, 1: 4640)

Όπως παρατηρήθηκε τα δεδομένα στόχου έχουν ασύμμετρη κατανομή, για αυτόν τον λόγο θα γίνει υπολογισμός πίνακα βαρών για την κάθε μια από αυτές[13] για να βοηθήσουμε[14] το μοντέλο να κάνει σωστές προβλέψεις. Επίσης στον διαχωρισμό των δεδομένων (*train_test_split*()) θα κρατηθεί το 50% για την εκπαίδευση και 50% για την δοκιμή του και το 30% για την εσωτερική επικύρωση (*Validation*) στην διαρκεί της εκπαίδευσης. Τέλος θα κρατηθούν όλα τα δεδομένα παρά τις παραπάνω παρατηρήσεις.

Τα βήματα υλοποίησης του μοντέλου:

- Έλεγχος δεδομένων για κενά, συχνότητα εμφάνισης κ.λπ.
- Διαχωρισμός στήλης στόχου (*y*) με τα υπόλοιπα δεδομένα
- Μετατροπή κατηγορικών μεταβλητών μέσω get_dummies()
- Χωρισμός του 50% των δεδομένων για χρήση προς την εκπαίδευση και του άλλου 50% για την δοκιμή του μοντέλου μέσω *train_test_split*().
- Κανονικοποίηση δεδομένων εκπαίδευσης μέσω *StandarScaler*()
- Ορισμός μοντέλου με δυο κρυφά επίπεδα από 128 νευρώνες αντίστοιχα με συνάρτηση ενεργοποίησης *ReLU* και *Dropout*: 0.3 (30%), με συνάρτηση

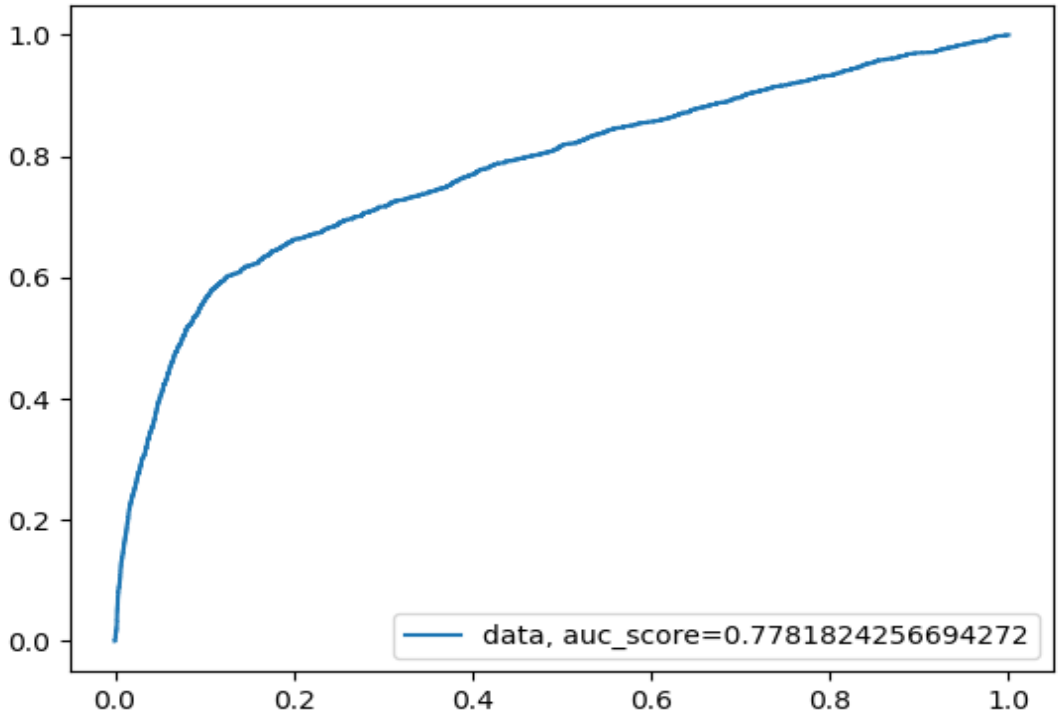
[13] Ο στόχος (y) έχει δυαδική μορφή 0 ή 1.

[14] Να δώσει περισσότερη προσοχή (επιβράβευση/αλλαγή βαρών του νευρωνικού δικτύου) όταν βρίσκει σωστά [1], και όχι τόσο όταν προβλέπει [0]

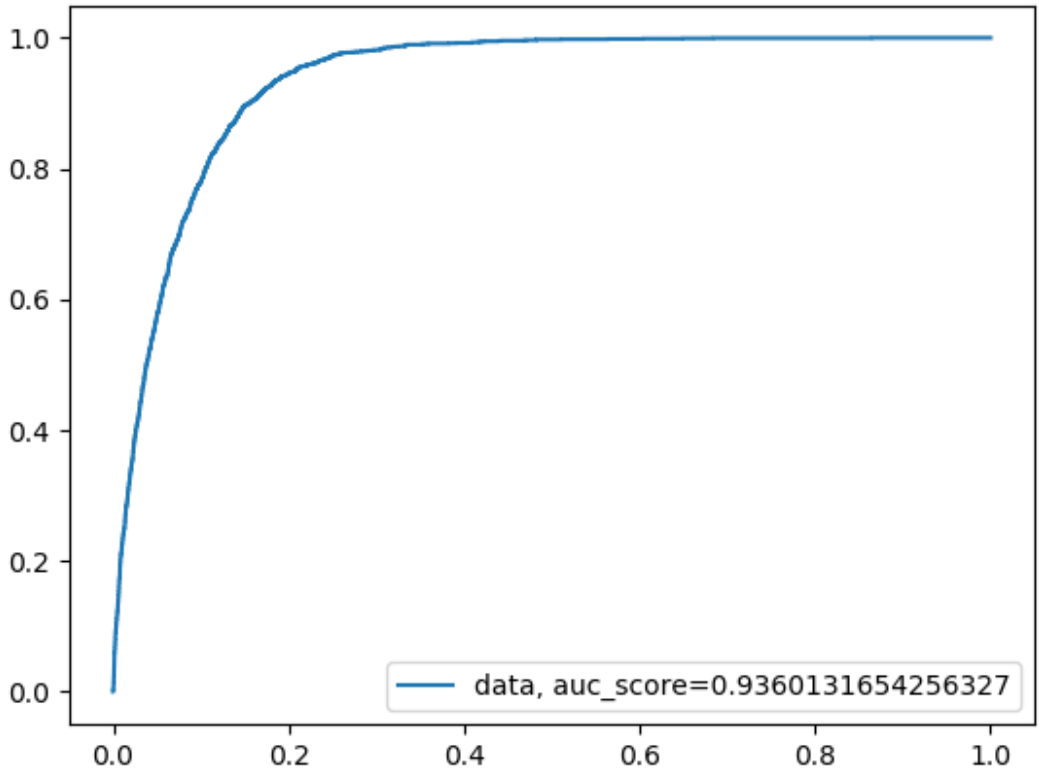
ενεργοποίησης στον τελευταίο νευρώνα *sigmoid*. Ως συνάρτηση σφάλματος ορίστηκε *binary_crossentropy* και αλγόριθμος βελτιστοποίησης *Nadam* με ρυθμό μάθησης 0.01

- Υπολογισμός βαρών για την ασύμμετρη κατανομή στόχου ορίστηκε ***[{0: 0.5643119416890447, 1: 4.387302939923306}]*** με την βοήθεια της βιβλιοθήκης *sklearn* με την μέθοδο *class_weight*
- Ορισμοί παραμέτρων εκπαίδευσης: *shuffle: True, validation_split: 0.3, epochs:10 και batch_size: 128*

Για τη δοκιμή με τη στήλη *duration* στα δεδομένα τα βάρη για τον στόχο ήταν: ***{0: 0.5637249534654549, 1: 4.423109965635739}.*** Σε καμία περίπτωση δεν υποστηρίζεται ότι τα αποτελέσματα σχετικά με τη δεύτερη δοκιμή είναι καλυτέρα από αυτά που βρήκαν οι ερευνητές *Moro et al*.



Εικόνα 53. AUC σκορ για την δοκιμή χωρίς την στήλη **duration**.



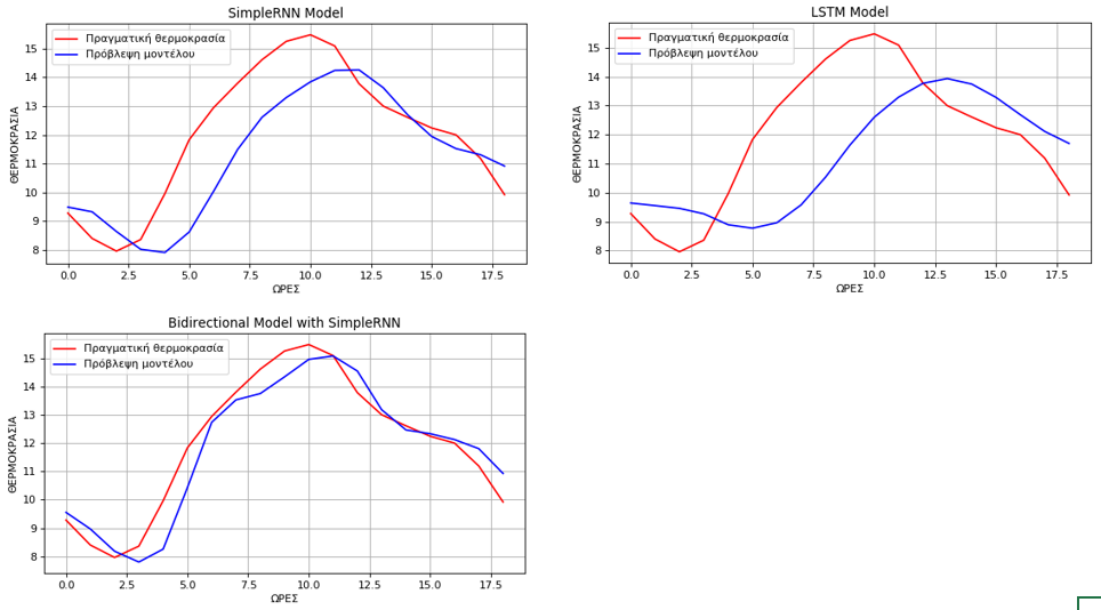
Εικόνα 54. AUC σκορ με την στήλη duration.

3.2.3 Παράδειγμα χρονοσειράς, πρόβλεψη θερμοκρασίας.

Για το παράδειγμα αυτό θα χρησιμοποιηθούν τα ίδια δεδομένα που χρησιμοποιήθηκαν στο κεφάλαιο ***3.1.5*** που αφορούσαν δεδομένα θερμοκρασίας στο νομό Αττικής.

Θα χρησιμοποιηθούν οι τελευταίες 24 τιμές για τη δοκιμή των μοντέλων και τα υπόλοιπα δεδομένα θα χρησιμοποιηθούν για την εκπαίδευση. Επίσης έγινε χρήση της λογικής *Sliding Window* με βήμα πέντε, και η έκτη τιμή ως τιμή στόχου κάθε φορά.

Παρακάτω τα αποτελέσματα των τριών πειραμάτων.



Εικόνα 55. Αποτελέσματα προβλέψεων.

Παράμετρος	Πρώτη δοκιμή (SimpleRNN)	Δεύτερη δοκιμή (LSTM)	Τρίτη δοκιμή (Bidirectional(SimpleRNN))
Επίπεδο εισόδου	*128*	*128*	*128*
Συνάρτηση ενεργοποίησης εισόδου	*ReLU*	*ReLU*	*ReLU*
Dropout	*0.25*	*0.25*	*0.25*
Κρυφό επίπεδο	*256*	*256*	*256*
Συνάρτηση ενεργοποίησης	*ReLU*	*ReLU*	*ReLU*
Dropout	*0.25*	*0.25*	*0.25*
Συνάρτηση ενεργοποίησης εξόδου	*linear*	*linear*	*linear*
Συνάρτηση σφάλματος	*mean_squared_error*	*mean_squared_error*	*mean_squared_error*
Αλγόριθμος βελτιστοποίησης	*adam*	*adam*	*adam*
Συνολικό σφάλμα (mean_squared_error)	*2.26*	*5.13*	*0.51*

Εικόνα 56. Αποτελέσματα δοκιμών για τις τελευταίες 24 τιμές..

Οι δοκιμές έγιναν για την διερεύνηση των διαφορετικών ειδών του τύπου *RNN*. Από τις δοκιμές όπως φαίνονται στον παραπάνω πίνακα η δοκιμή με *Bidirectional* και με τη χρήση απλού τύπου RNN (SimpleRNN) είχε τα καλυτέρα αποτελέσματα για την πρόβλεψη θερμοκρασίας στις τελευταίες 24 τιμές.

3.2.4 Πρώτο παράδειγμα εφαρμογής CNN (Πολυταξικός ταξινομητής)

Τα δεδομένα[15] για το παράδειγμα αυτό αφορούν αριθμούς στο διάστημα [0-9] στην νοηματική γλώσσα. Το μοντέλο εκπαιδεύτηκε στα δεδομένα (συνολικά 2062 εικόνες) και στόχος του μοντέλου τέθηκε να είναι η πρόβλεψη αριθμών στην νοηματική γλώσσα λαμβάνοντας εικόνες από κάμερα σε πραγματικό χρόνο.

Τα δεδομένα βρίσκονται ήδη στην σωστή μορφή (μέσα σε φακέλους για την κάθε κλάση σε φάκελο με ονομασία (*Dataset*)). Παρακάτω τα βήματα υλοποίησης του μοντέλου.

Επεξεργασία δεδομένων χρησιμοποιώντας το ImageDataGenerator από το Keras:

- Κανονικοποίηση δεδομένων: "*rescale* = 1./255"
- Αλλάζοντας την γωνία: "*rotation_range*=15" , "*shear_range*=0.01"
- Μετατόπιση: "*width_shift_range*=0.1, *height_shift_range*=0.1"
- Μεγέθυνση: "*zoom_range*=[0.9, 1.25]"
- Περιστροφή εικόνας κατά τον οριζόντιο άξονα: "*horizontal_flip=True"*
- Αλλαγή φωτεινότητας: "*brightness_range*=[0.9, 1.5]"
- Οι αλλαγές αυτές βοηθάνε στο να παραχθούν περισσότερα δεδομένα για την εκπαίδευση

Ορισμός φακέλου που βρίσκονται τα δεδομένα *"directory=r"./Dataset"* " και δημιουργία γεννήτριας για την εκπαίδευση με παραμέτρους:

- Το μέγεθος τις εικόνας για εκπαίδευση: "*target_size*=(64, 64)"¶
- Αλλαγή σε κλίμακα του γκρι: "*color_mode="grayscale"* "
- Το είδος της πρόβλεψης: "*class_mode="categorical"* "
- Και τέλος το μέγεθος των δεδομένων για ένα κάλεσμα του: "*batch_size*=32"

Ορισμός μοντέλου με τις εξής παραμέτρους:

- Για το επίπεδο εισόδου έγινε χρήση 36 φίλτρων μεγέθους (3x3) με συνάρτηση ενεργοποίησης *ReLU*, και *MaxPooling*: (2x2).

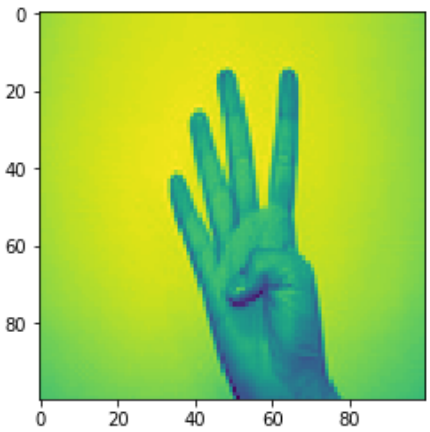
[15] https://github.com/ardamavi/Sign-Language-Digits-Dataset

- Για το κρυφό επίπεδο έγινε χρήση 24 φίλτρων μεγέθους (3x3) με συνάρτηση ενεργοποίησης *ReLU*, και *MaxPooling*: (4x4).
- Χρήση ισοπεδωτικής συνάρτησης (*Flatten*).
- Για το πυκνό επίπεδο έγινε χρήση 256 νευρώνων με συνάρτηση ενεργοποίησης *ReLU* και *Dropout*: 0.5 (50%).
- Για το επίπεδο εξόδου έγινε χρήση 10 νευρώνων με συνάρτηση ενεργοποίησης *softmax*.
- Ορισμός συνάρτησης βελτιστοποίησης βαρών *Adam* με ρυθμό μάθησης *0.001* και συνάρτηση σφάλματος *binary_crossentropy*.
- Εκπαίδευση του μοντέλου με παραμέτρους *steps_per_epoch*: 64 και *epochs*: 20 και αποθήκευση του στην ολοκλήρωση της εκπαίδευσης.

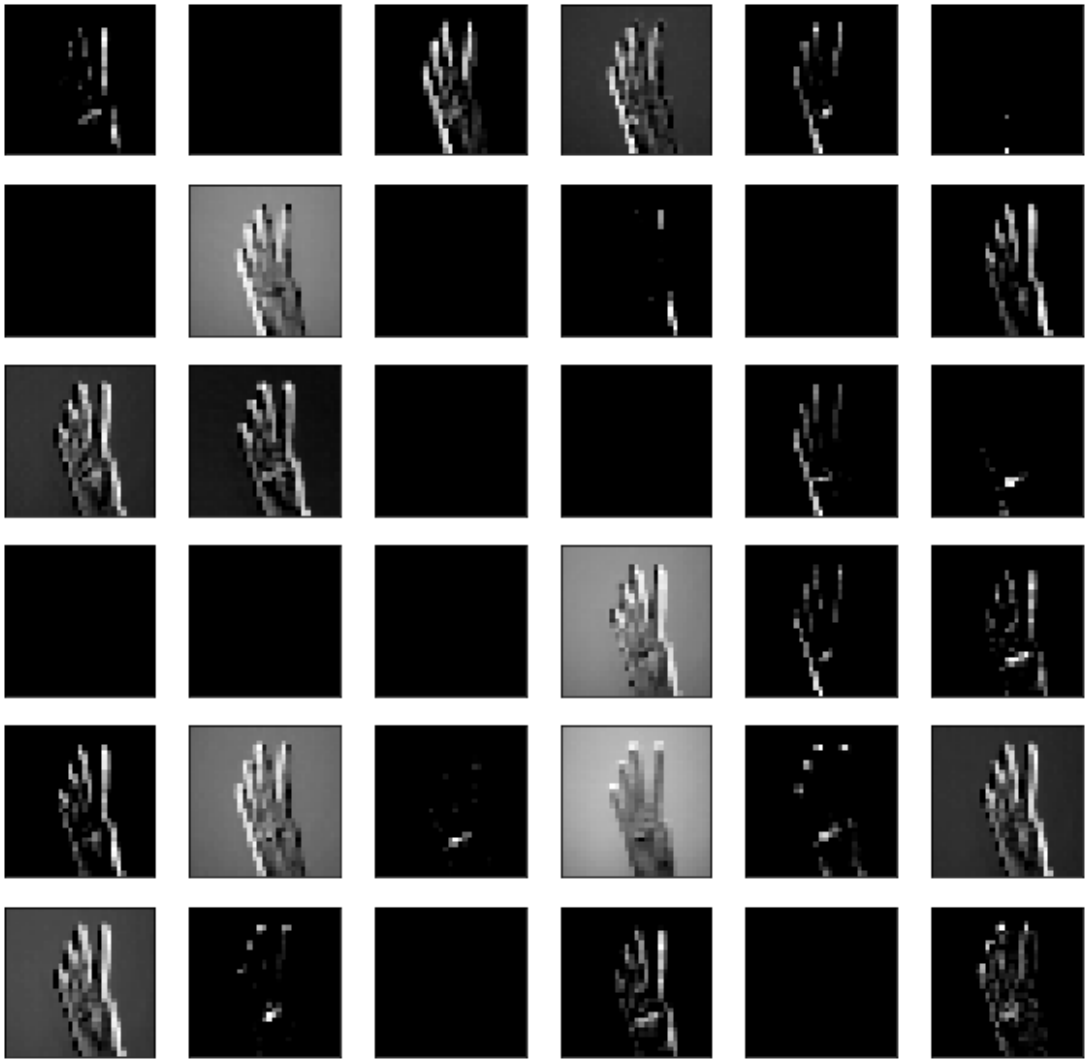
  Η ακρίβεια (*accuracy*) του μοντέλου για 20 επαναλήψεις (*epochs*) ήταν ***0.9565***.

Η επιλογή του αριθμού των φίλτρων και παραμέτρων έγινε μετά αρκετές δοκιμές. Επιλέχθηκαν αυτές οι ρυθμίσεις για το μοντέλο καθώς αυτές ήταν οι σχετικά καλύτερες για το συγκεκριμένο πρόβλημα. Υπάρχουν τεχνικές κατά τις οποίες είναι δυνατό να βρεθούν οι βέλτιστες ρυθμίσεις των υπερ-παραμέτρων, οι οποίες δεν θα καλυφθούν στην διπλωματική αυτή.

Για για την παρακάτω εικόνα θα παρουσιαστούν οι ενεργοποιήσεις των φίλτρων στο δεύτερο επίπεδο του δικτύου.



Εικόνα 57. Εικόνα για δοκιμή

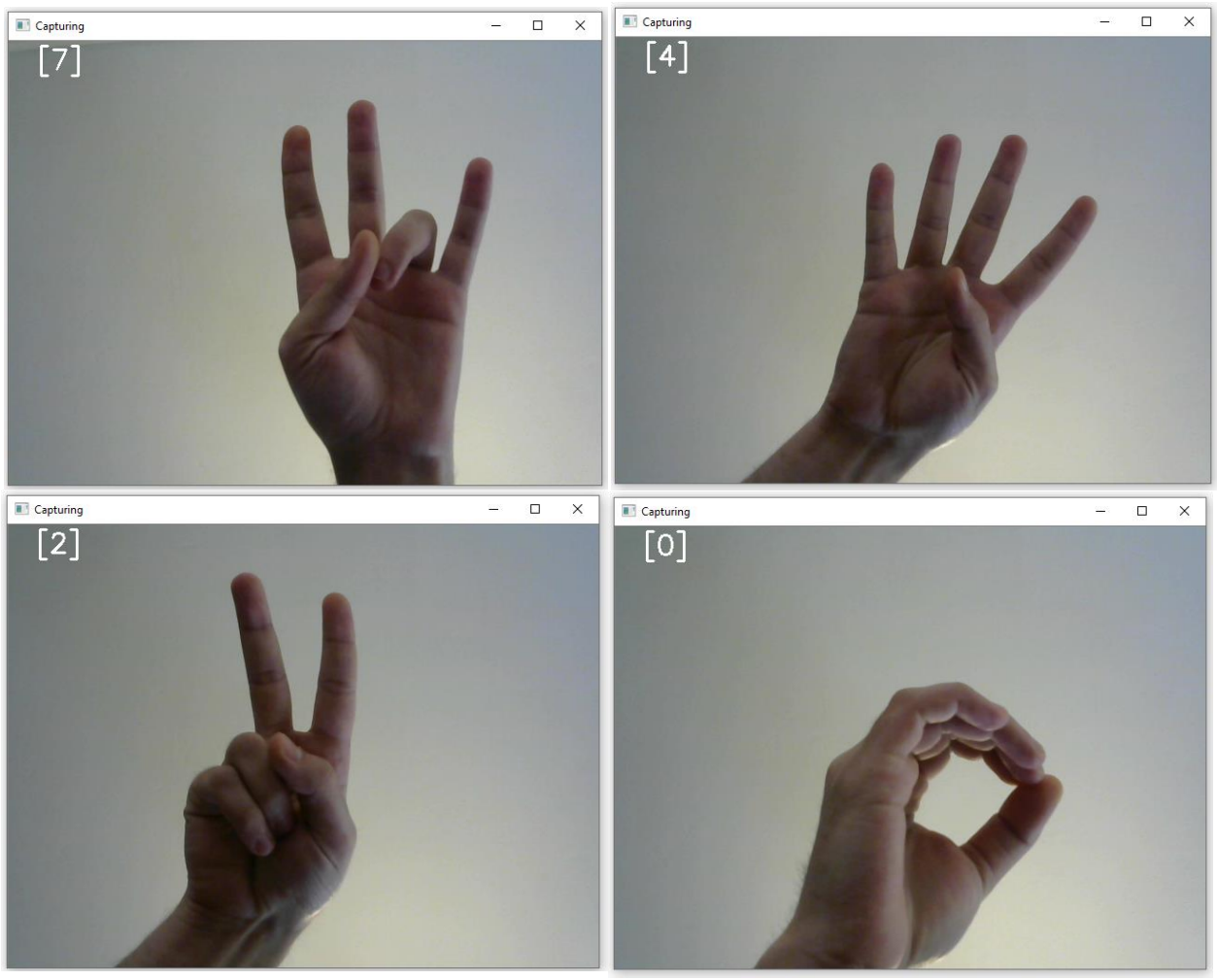


Εικόνα 58. Οι ενεργοποιήσεις των φίλτρων στο δεύτερο επίπεδό (κρυφό).

Αφού το μοντέλο εκπαιδεύτηκε, είναι δυνατή η χρήση μιας κάμερας για την πρόβλεψη αριθμών στην νοηματική γλώσσα σε πραγματικό χρόνο. Για την πρόβλεψη χρησιμοποιώντας δεδομένα από κάμερα τα βήματα που ακολουθήθηκαν είναι τα εξής:

- Χρήση *opencv* βιβλιοθήκης για την εισαγωγή και επεξεργασία εικόνων από κάμερα
- Δημιουργία γεννήτριας ίδια με αυτήν στην εκπαίδευση του μοντέλου
- Φόρτωση μοντέλου που εκπαιδεύτηκε
- Πρόβλεψη και απεικόνιση αποτελεσμάτων του μοντέλου

Μερικές προβλέψεις:



Αν και το μοντέλο κατάφερε να προβλέψει σωστά όλους τους αριθμούς, δυσκολευόταν να κάνει σωστή πρόβλεψη όταν υπήρχε αλλαγή στην είσοδο. Δηλαδή το χέρι όπως φαίνεται στις παραπάνω φωτογραφίες ήταν πιο μακριά από την κάμερα, ή βρισκόταν σε διαφορετική γωνία από αυτήν (±90 μοίρες). Σε αυτές τις περιπτώσεις το μοντέλο δεν μπορούσε να προβλέψει σωστά. Για την διόρθωση αυτού του θέματος πρέπει να συλλεχθούν περισσότερα δεδομένα σε διαφορετικές αποστάσεις και γωνίες από την κάμερα από αυτές που χρησιμοποιήθηκαν για την εκπαίδευση του μοντέλου.

3.2.5 Δεύτερο παράδειγμα CNN (Παλινδρόμηση)

Για αυτό το παράδειγμα θα χρησιμοποιηθούν δεδομένα[16] από προσομοιωτή ο οποίος δημιουργήθηκε από: https://www.udacity.com/course/self-driving-car-

[16] https://s3.amazonaws.com/video.udacity-data.com/topher/2016/December/584f6edd_data/data.zip

engineer-nanodegree--nd013. Ο κώδικας και το πρότζεκτ του προσομοιωτή είναι διαθέσιμος για κάθε χρήση στο: https://github.com/udacity/self-driving-car-sim

Ο στόχος του μοντέλου είναι να οδηγήσει το αυτοκίνητο στην πρώτη πίστα του προσομοιωτή. Δηλαδή να προβλέψει την γωνία με την οποία πρέπει να στρίψει έχοντας ως είσοδο μόνο την εικόνα από την κεντρική κάμερα.

Κατά την διάρκεια της συλλογής δεδομένων συλλέχθηκαν εικόνες από τρεις κάμερες (*Αριστερή, Κεντρική, Δεξιά*). Αλλά μόνο μια κάμερα χρησιμοποιήθηκε για την πρόβλεψη της γωνίας για το τελικό μοντέλο.

Η υλοποίηση του μοντέλου ήταν μια διαφοροποίηση του μοντέλου που πρότειναν οι ερευνητές (Gundling, 2017) (Bojarski, et al., 2016)

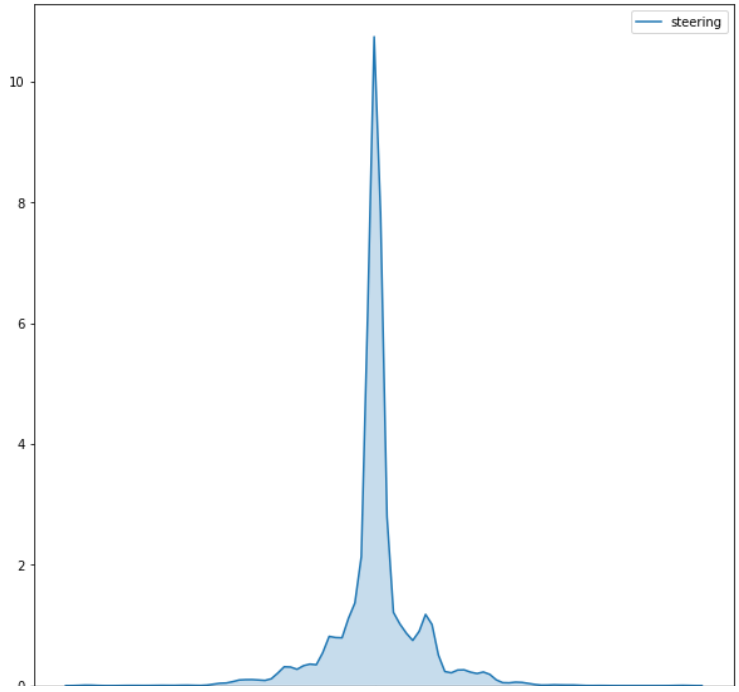
Όπως παρατηρούμε το αρχείο ***.csv*** στις πρώτες τρεις στήλες έχει πληροφορίες για το πού και ποιά είναι η εικόνα που αντιστοιχεί για τις άλλες τέσσερεις στήλες που ο προσομοιωτής έχει καταγράψει. Για το παράδειγμα αυτό θα χρησιμοποιηθεί μόνο η στήλη που έχει τις τιμές για την γωνία του τιμονιού (steering) και οι τρεις εικόνες.

Άρα κατά την διάρκεια της εκπαίδευσης του μοντέλου θα χρησιμοποιηθούν και οι τρεις εικόνες και ως «στόχος» θα είναι η αντίστοιχη γωνία που είχε το αυτοκίνητο εκείνη την στιγμή.

Διάγραμμα 1. Είσοδοι και στόχος του μοντέλου κατά τη διάρκεια της εκπαίδευσης.

Παρατηρείται επίσης πως οι σύνδεσμοι για την αριστερή και δεξιά εικόνα έχουν ένα κενό. Για αυτόν τον λόγο, ακολουθεί η διόρθωση των δεδομένων για την αποφυγή λαθών κατά την προ-επεξεργασία τους.

Η στήλη *steering* περιέχει 124 μοναδικές τιμές οι οποίες είναι στο διάστημα [1.0, -0.94] και γενικά τα δεδομένα δεν περιέχουν ελλείποντα στοιχεία.



Εικόνα 59.Απεικόνιση κατανομής στήλης steering

Όπως παρατηρείται, η στήλη *steering* περιέχει πάρα πολλές τιμές με μηδενική τιμή, κάτι το οποίο σημαίνει πως κατά την διάρκεια της καταγραφής, το όχημα παρέμεινε σταθερό δηλαδή δεν υπήρχαν αλλαγές ως προς την γωνία του τιμονιού.

Αυτό έχει ως αποτέλεσμα, τα δεδομένα να είναι μη ισορροπημένα (*Unbalanced Dataset*), και να μην είναι χρήσιμα για την εκπαίδευση κάποιου μοντέλου. Αν για παράδειγμα σε ένα σύνολο δεδομένων με 80% οι τιμές στόχου να είναι μηδενικές και το μοντέλο προβλέψει πάντα μηδέν τότε θα έχει ακρίβεια 80%.

Η διόρθωση αυτού του προβλήματος, στο συγκεκριμένο παράδειγμα, έγινε με την δημιουργία μιας γεννήτριας στην Python.

Μέσα στον κώδικα της γεννήτριας:

- Η κεντρική εικόνα μαζί με την αντίστοιχη γωνία έμειναν όπως έχει.
- Η κεντρική εικόνα άλλαξε πλευρά (Flip) και η γωνία πήρε αρνητικό πρόσημο.
- Η αριστερή εικόνα άλλαξε πλευρά (Flip) και στην τιμή της γωνίας προστέθηκε +.45.
- Η δεξιά εικόνα άλλαξε πλευρά (Flip) και στην τιμή της γωνίας προστέθηκε η τιμή -.45

Έγινε κοπή και αλλαγή μεγέθους για την καλύτερη παρατήρηση του μοντέλου των ορίων της πίστας/δρόμου. Αποκλείοντας έτσι από την εκπαίδευση μη χρήσιμες πληροφορίες από την εικόνα όπως ο ουρανός, το περιβάλλον και το μπροστινό μέρος του οχήματος. Ύστερα, έγινε αλλαγή από κλίμακα «*BGR-RGB*» σε κλίμακα του γκρι, διότι δεν είναι χρήσιμη η πληροφορία του χρώματος στην εικόνα και επίσης το μοντέλο εκπαιδεύεται πιο γρήγορα.

Τέλος, η γεννήτρια επιστρέφει έναν πίνακα με τις εικόνες και έναν πίνακα με τις αντίστοιχες νέες πλέον γωνίες. Η κανονικοποίηση της εικόνας έγινε διαιρώντας τα εικονοτοιχεία (pixels) με *255*.

Για την εκπαίδευση επιλέχθηκε ως αριθμός για την κάθε παρτίδα (*Batch_size*) το 32. Δηλαδή η γεννήτρια θα επιστρέψει συνολικά 128 [32*4] εικόνες και γωνίες.

Όπως και στο προηγούμενο παράδειγμα *CNN* μοντέλου κρατήθηκε η λογική του ενός κρυφού επιπέδου, διότι όπως και πριν δεν ζητάμε από το μοντέλο να μάθει υψηλά χαρακτηριστικά. Αυτό που μας ενδιαφέρει στο συγκεκριμένο παράδειγμα είναι το μοντέλο να αναγνωρίζει τα όρια του δρόμου.

Ορισμός μοντέλου:

- Επίπεδο εισόδου: 32 φίλτρα με μέγεθος (5x5), με συνάρτηση ενεργοποίησης *elu*, *MaxPooling* (4x4)
- Δεύτερο επίπεδο (κρυφό): 16 φίλτρα με μέγεθος (3x3), με συνάρτηση ενεργοποίησης *elu*, *MaxPooling* (3x3)
- Χρήση ισοπεδωτικής συνάρτησης (*Flatten*)
- Για το πρώτο πυκνό επίπεδο χρήση 256 νευρώνων με συνάρτηση ενεργοποίησης *elu* και *Dropout*: 0.5 (50%)
- Για το δεύτερο πυκνό επίπεδο χρήση 10 νευρώνων με συνάρτηση ενεργοποίησης *elu*
- Για το επίπεδο εξόδου χρήση 1 νευρώνα
- Ορισμός συνάρτησης βελτιστοποίησης βαρών Adam με ρυθμό μάθησης *0.001*, συνάρτηση σφάλματος *mse* (*mean_squared_error*)
- Εκπαίδευση του μοντέλου με παραμέτρους *steps_per_epoch*: 251 και *epochs*: 6 και αποθήκευση του στην ολοκλήρωση της εκπαίδευσης

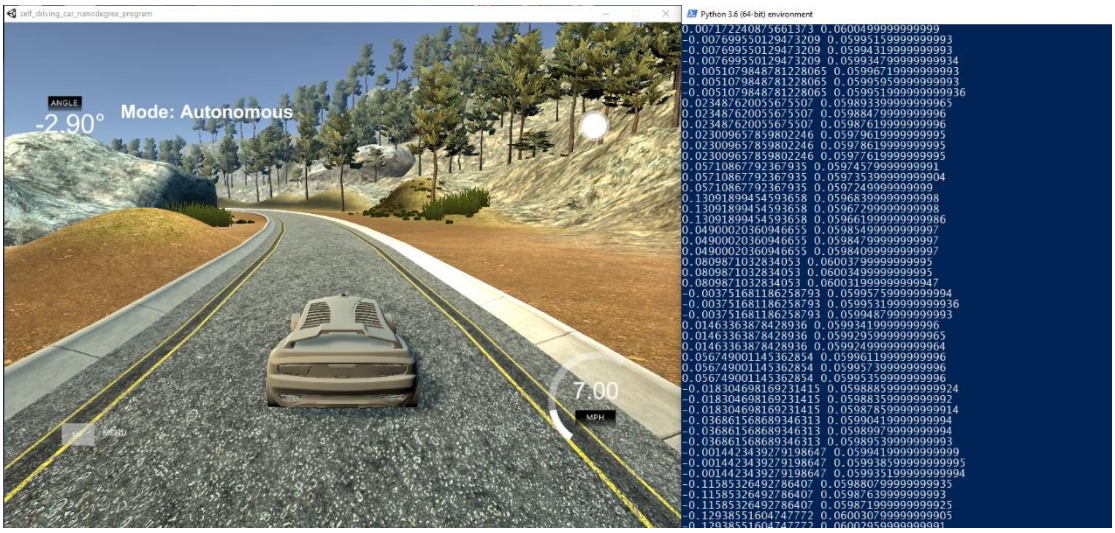
Έγινε χρήση της συνάρτησης ενεργοποίησης *elu* διότι υπάρχουν αρνητικές τιμές στην είσοδο αλλά και στην έξοδο που θα προσπαθήσει το μοντέλο να προβλέψει τις τιμές στόχου.

Τέλος μετα από την εκπαίδευση του μοντέλου η ακρίβεια ήταν στο ***0.2693***, κάτι το οποίο δεν πρέπει να προκαλέσει προβληματισμούς. Από τα δεδομένα για την δοκιμή προβλέψεων του μοντέλου παρατηρήθηκε πως τα όρια (*min-max*) ήταν στα [-0.57588226, 0.6488244].

Έγιναν αλλαγές στον κώδικα του αρχείου «***Drive.py***[17]» και προσαρμόστηκε για το μοντέλο. Το αρχείο αυτό αποτελεί ουσιαστικά τον τρόπο με τον οποίο το μοντέλο

17 Έγινε χρήση αυτής της έκδοσης https://github.com/harveenchadha/Udacity-CarND-Behavioral-Cloning-Final/blob/master/drive.py

θα επικοινωνήσει με τον (**API**) προσομοιωτή για τον έλεγχο του οχήματος. Η ταχύτητα του οχήματος παραμένει σταθερή (***7***) με την βοήθεια ενός «***PID***».



Εικόνα 60. Στιγμιότυπο κατά τη διάρκεια αυτόνομης οδήγησης από το μοντέλο.

Βίντεο[18] με το αυτοκίνητο να οδηγεί αυτόνομα την πίστα κάνοντας χρήση του μοντέλου: https://www.youtube.com/watch?v=lg6LRFp7PFE

Το μοντέλο ήταν ικανό να ολοκληρώσει την πρώτη πίστα χωρίς να βγει εκτός τα όρια της πίστας. Μετα από πειράματα, παρατηρήθηκε πως με μόλις τρία περάσματα (epochs) σε όλα τα δεδομένα ήταν και πάλι ικανό να ολοκληρώσει την πίστα. Ακόμη και όταν το όχημα τοποθετήθηκε στην ανάποδη πλευρά της πίστας δεν αντιμετώπισε προβλήματα στην ολοκλήρωση.

[18] Το βίντεο δημιουργήθηκε χρησιμοποιώντας τις εικόνες της κεντρικής κάμερας του οχήματος κατά τη διάρκεια αυτόνομης οδήγησης.

ΚΕΦΑΛΑΙΟ 4- ΣΥΜΠΕΡΑΣΜΑΤΑ

Αλγόριθμος DBSCAN: Όπως παρατηρήσαμε στα πειράματα που έγιναν τέσσερις δοκιμές για την παράμετρο *min_samples* και τέσσερις για την *eps*, η σωστή επιλογή των δυο παραμέτρων είναι υψίστης σημασίας. Είναι σημαντικό να αναφερθεί πως τα πειράματα και η εφαρμογή του αλγορίθμου έγιναν σε συνθετικά δεδομένα και όχι σε πραγματικά, οπότε χρειάζεται μεγάλη προσοχή στην εύρεση κατάλληλων παραμέτρων για να θεωρηθούν αποδεκτά τα αποτελέσματα.

Αλγόρίθμος LOF: Αν και ο αλγόριθμος θεωρητικά δούλεψε σωστά στα συγκεκριμένα δεδομένα και σε σύγκριση με τον αλγόριθμο DBSCAN δεν είχε πρόβλημα με δεδομένα που είχαν διαφορετική πυκνότητα, αυτό δεν σημαίνει πως θα συμβεί το ίδιο με πραγματικά δεδομένα. Σε πραγματικά δεδομένα ο χρήστης δεν μπορεί να γνωρίζει τον πραγματικό αριθμό σφαλμάτων και χρειάζεται μεγάλη προσοχή στην χρήση του αλγορίθμου.

Αλγόριθμος Isolation Forest: Διαπιστώνεται πως ήταν ο πιο εύκολος στην χρήση του μιας και ο αλγόριθμος δίνει την δυνατότητα να βρει ένα ποσοστό σημείων/τιμών αυτόνομα σε ένα σύνολο δεδομένων ως σφάλματα. Έτσι δίνεται η δυνατότητα μιας αρχικής εκτίμησης των σφαλμάτων σε ένα σύνολο δεδομένων. Όπως επίσης και η μέθοδος Tukey IQR παρέχει μια εύκολη και γρήγορη εκτίμηση για τις ακραίες τιμές σε ένα σύνολο δεδομένων.

Ελλείποντα στοιχεία - μέθοδος Interpolate (pchip): Είχε τα καλύτερά αποτελέσματα για δεδομένα χρονοσειρών. Αυτό δεν σημαίνει πως πρέπει να χρησιμοποιηθεί για όλα τα είδη δεδομένων χρονοσειρών. Όπως αναφέρθηκε στους αλγορίθμους εύρεσης σφαλμάτων, χρειάζεται ιδιαίτερη προσοχή διότι αυτά είναι στοιχεία που αλλάζουν τα συμπεράσματα σε μια μελέτη. Επίσης δεν διερευνήθηκαν δεδομένα με διαφορετικά χαρακτηριστικά όπως κατηγορηματικά, που εκεί η εύρεση κατάλληλων τιμών γεμίσματος δεν είναι τόσο απλή. Προτείνεται πως ένας αλγόριθμος ομαδοποίησης θα μπορούσε να κάνει μια εκτίμηση για ένα ελλείπον στοιχείο, εάν γίνει ομαδοποίηση στο σύνολο δεδομένων αυτών. Ακόμη μια μέθοδος για την επίλυση του προβλήματος αυτού είναι η δημιουργία ενός μοντέλου που θα εκπαιδευτεί με τα χαρακτηριστικά που δεν έχουν ελλιπείς τιμές και ως στόχος να τεθεί η στήλη που υπάρχουν ελλιπείς τιμές. Κατά αυτόν τον τρόπο θα προβλέψει τις τιμές που λείπουν.

Αλγόριθμος PCA: Υπάρχουν δύο τρόποι χρήσης του. Ο πρώτος αποτελεί τη μείωση διαστάσεων ενός συνόλου δεδομένων με στόχο την μείωση χρόνου

εκπαίδευσης για ένα μοντέλο πρόβλεψης. Ο δεύτερος στοχεύει και εκείνος στη μείωση διαστάσεων αλλά αυτή την φορά για απεικόνιση η οποία θα οδηγήσει στην καλύτερη κατανόηση στα δεδομένα που ερευνώνται.

Estimator API: Σε σύγκριση με την έρευνα που πραγματοποίησαν οι συγγραφείς Tüfekci et al., και με το πειραματικό μοντέλο στο παράδειγμα αυτό μπορούμε να αναφέρουμε πως τα αποτελέσματα που βρέθηκαν χρησιμοποιώντας το Estimator API είναι αρκετά ικανοποιητικά.

Keras μοντέλο ταξινόμησης: Σε σύγκριση με τον Moro et al., όπου έγινε εκτενής έρευνα για τη διερεύνηση επιδόσεων πρόβλεψης χρησιμοποιώντας μεθόδους μηχανικής μάθησης και με το πειραματικό μοντέλο που δημιουργήθηκε για το παράδειγμα δεν μπορούμε να είμαστε σε θέση να σχολιάσουμε τα αποτελέσματα. Διότι δεν υπάρχει επάρκεια γνώσεων στον τομέα που βρίσκονται τα δεδομένα και αυτός ήταν ο λόγος να διατηρηθούν τα ίδια δεδομένα (χαρακτηριστικά) που πρότειναν οι συγγράφεις. Τέλος στο παράδειγμα αυτό παρουσιαστήκαν τρόποι επεξεργασίας και αντιμετώπισης σε δεδομένα στόχου (target) με ασύμμετρη κατανομή.

Παραδείγματα CNN : Παρουσιάστηκαν δυο μοντέλα. Το πρώτο αφορούσε την πρόβλεψη σωστής κλάσης [0-9] χρησιμοποιώντας πραγματικά δεδομένα εικόνας με τη χρήση κάμερας. Για το συγκεκριμένο παράδειγμα, τα δεδομένα δεν ήταν επαρκή αλλά το μοντέλο ήταν ικανό να αποφέρει ικανοποιητικά αποτελέσματα.

Το δεύτερο αφορούσε τη σωστή πρόβλεψη γωνίας τιμονιού για την οδήγηση του οχήματος μέσα στα όρια του δρόμου καθ’ όλη τη διάρκεια της πίστας (στον προσομοιωτή) χρησιμοποιώντας μόνο την κεντρική εικόνα του οχήματος. Παρουσιάστηκαν προβλήματα σχετικά με την αντιμετώπιση μηδενικών τιμών στα δεδομένα στόχου για τα οποία έγινε προσπάθεια επίλυσης, προσθέτοντας συνθετικές τιμές για τις θέσεις αυτές. Αυτό όμως σημαίνει πως έγινε τροποποίηση τιμών στόχου προς όφελος μας, για να ολοκληρώσει το όχημα την πίστα. Το τελευταίο δεν έπρεπε να συμβεί διότι οι μηδενικές τιμές ήταν φυσιολογικές αλλά κρίθηκε απαραίτητο για την επίτευξη του στόχου του παραδείγματος.

ΚΕΦΑΛΑΙΟ 5- ΠΑΡΑΡΤΗΜΑ

5.1 Κώδικας DBSCAN

DBSCAN make_moons Dataset

Εισαγωγή βιβλιοθηκών

```
import matplotlib.pyplot as plt
%matplotlib notebook
from sklearn.cluster import DBSCAN
import pandas as pd
import numpy as np
from sklearn import cluster, datasets
```

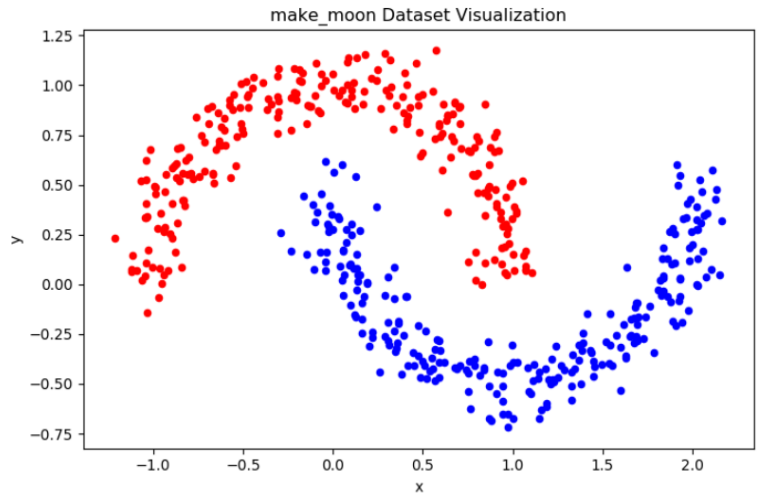
Δημιουργία δεδομένων και αποθήκευση τους ως Pandas DataFrame

```
samples=500

X, y =datasets.make_moons(n_samples=samples, noise=0.1)

# Οπου X ειναι τα δεδομένα, χωρισμένα σε δυο στήλες. Οπου y ειναι μια επιπλέον στήλη στην οποία αποθηκεύουμε σε
# ποια απο τις δυο ομάδες ανήκει η κάθε σειρά.

df = pd.DataFrame(dict(x=X[:,0], y=X[:,1], label=y))

# Μετατροπή των δεδομένων σε Pandas DataFrame και ορίζουμε μια 3η στήλη για τις δυο ομάδες.
```

Απεικόνιση των δεδομένων

```
colors = {0:'red', 1:'blue'}

# Δημιουργούμε ενα λεξικό το οποία θα χρησημοποιήσουμε στην συνέχεια για να απεικονίσουμε τα δεδομένα.

fig, ax = plt.subplots()
grouped = df.groupby('label')
```

```
for key, group in grouped:
    group.plot(ax=ax, kind='scatter', x='x', y='y', color=colors[key])
    plt.title("make_moon Dataset Visualization")

# Απεικόνιση δεδομένων ως ενα διάγραμμα διασποράς χρησιμοποιώντας το λεξικό για τον ορισμό των χρωμάτων
```



Δεν χρειάζεται η 3η στήλη αλλά μόνο τα δεδομένα για την εφαρμογή του αλγορίθμου

```
df2 = (df.drop(['label'], axis=1))

# Διαγραφή 3ης στήλης αποθηκεύοντας τα τροποποιημένα δεδομενα σε μια νεα μεταβλητη, αφηνοντας ετσι την κυρια αθικτη.

model = DBSCAN(eps=0.17, min_samples=15).fit(df2)

# Εφαρμογη αλγοριθμου DBSCAN στα δεδομενα "df2".
```

```
label = model.labels_

# Αποθηκευση πινακα με τις πληροφοριες επειτα απο εφαρμογη του αλγοριθμου.
# Οπου 0,1,2,3... αριθμηση ομαδων και με δεικτη -1 οι ανωμαλιες η αλλιως θορυβος που βρεθηκαν στα δεδομενα.

print(label)

# Απο την εκτυπωση, καταλαβαινουμε πως εχουμε δυο ομαδες [0, 1].
```

```
[ 0  1  1  1  0  0  1  1  1  0  1  1  0  0  0  1  1  1  1  0  1  1  1  1
  0  0  0  0  1  0  1  1  0  0  1  0  0  0  0  0  1  1  1  0  0  1  1  0
  0  0  0  0  0  0  0  1  1  1  1  1  0  0  1  1  1  0 -1  1  1  0  1  1  1
  0  0  0  1  1  1  0  1  1  1  0  1  0  0  0  0  1  0  1  1  0  0  0  0
  0  1  1  1  0  0  0  1  0  0 -1  0  1  0  1  1  0  0  0  1  1  0  1  1
  0  0  1  0  0  0  1  1  1  1  1  1  1  0  0  0  1  0  0  1  0  0  0  0
  1  0  1  1  1  1  0  0 -1  0  0  0  0  0  0  0 -1  0  0  1  0  1  0  1
  1  0  1  1  1  0  1  0  1  1  0  1  1  0  0  0  1  0 -1  0  1  1  1  1
  1  1  0  0  1  1  0  0  0  0  1  1  1  0  1  0  0  0  1  0  1  1  1  0
  0  1  0  0  0  1  0  0  1  0  1  0  0  0  1  0  1  0  0  1  0  1  0  0
  0  1  1  1  0  0  1  1  1  0  0  0  1  1  0  0  0  1  0  1  0  0  0 -1
  0  1  1  1 -1  1  1  1  1  0  1  0  1  1  0  0  0  0  0  1  0  0  0  1
  0  1  1  1  1  1  0  1  0  0  1  0  0  0  0  1  0  1  1  0  1  1  0  0
 -1  1  0  0  1  1  0  1  0  1  0  0  0  1  1  0  1  0  0  1  1  1  1  0
  0  1  1  0  1  1  0  0  1  0  0  1  0  0  1  1  1  0  0  0  0  1  0  0
  1  1  0  0  1  1  1  0  1  0  1  1  0  0  1  0  1  1  0  1  0 -1  1  0
  1  0  1  0  1  1  0  0  0  1  0  1  0  1  0  1  0  1 -1  1  0  1  1  1
  0  0  0  1  0  0  1  0  1  1  0  1  0  0  1  1  1  1  1  0  0  1  1  1
  0  0  1  1  0  1  0  0  1  0  1  1  0  1  1  1  1  0  1  0  0  1  1  0
  1  1  1  0  1  0  1 -1  0  1  0  0  0  1  0  0  1  0  0  1  0  1  0  1
  0  1  1  1  0  1  1  0  1  1  1  1  0  0  1  0  0  1  1  0]
```

Προσθήκη σε μια νέα στήλη των δεδομένων τις πληροφορίες που πήραμε απο τον αλγόριθμο "label = model.label_"

```
print(df2.head())
print(df2.info())
```

```
          x         y
0  0.843801  0.344454
1  1.253232 -0.467189
2  0.580611 -0.280110
3  0.105290  0.301315
4  0.797731  0.741202
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 2 columns):
x    500 non-null float64
y    500 non-null float64
dtypes: float64(2)
memory usage: 7.9 KB
None
```

```
df2['groups'] = label
print(df2.head())
print(df2.info())
```

```
          x         y  groups
0  0.843801  0.344454       0
1  1.253232 -0.467189       1
2  0.580611 -0.280110       1
3  0.105290  0.301315       1
4  0.797731  0.741202       0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 3 columns):
x         500 non-null float64
y         500 non-null float64
groups    500 non-null int64
dtypes: float64(2), int64(1)
memory usage: 11.8 KB
None
```

Μεταφορά των σειρών οι οποίες δεν έχουν τιμή [-1] στην νέα στήλη "groups" ¶
Καταμέτρηση συχνότητας εμφάνισης ανωμαλιών και ομάδων

```
unique, count = np.unique([label], return_counts=True)
outliers_groups_info = np.asarray((unique, count))
```

```
print(outliers_groups_info)
```

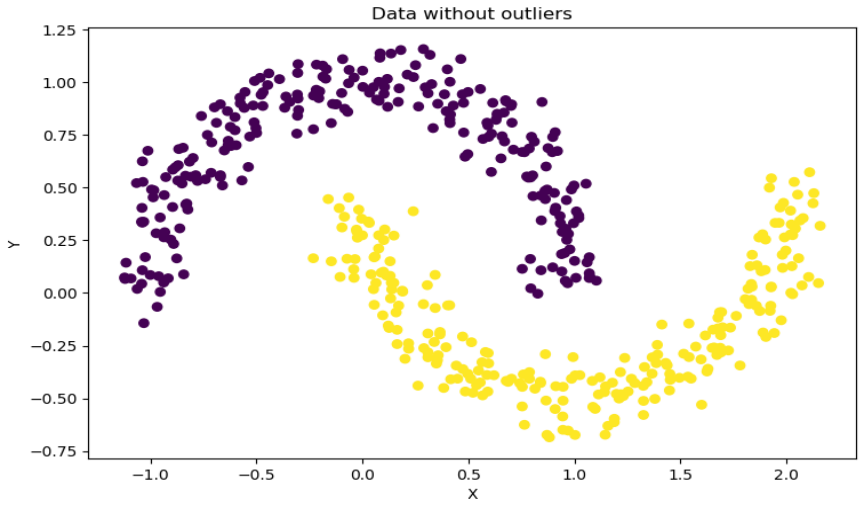
```
[[ -1   0   1]
 [ 11 247 242]]


Int64Index: 489 entries, 0 to 499
Data columns (total 3 columns):
x         489 non-null float64
y         489 non-null float64
groups    489 non-null int64
dtypes: float64(2), int64(1)
memory usage: 15.3 KB
None
```

Ποσοστό των δεδομένων που αφαιρέθηκαν απο τα αρχικά δεδομένα

```
n_samples = 500
# Έστω, για 500 τιμές για την παράμετρο στο make_moons.

num_outliers = outliers_groups_info[1][0]
# Επιλέγουμε απο τον πίνακα την τιμή των "ανωμαλιών" που βρέθηκαν απο τον αλγόριθμο DBSCAN.

val_left = n_samples - num_outliers
# Αφαιρούμε τις "ανώμαλες" τιμές απο το αρχικό πλήθος τιμών των δεδομένων, για να δούμε πόσες τιμές θα μείνουν στο καθα
# σύνολο δεδομένων.

pct_drop = ((val_left-n_samples)/n_samples)*100
```

```
print("Percentage drop: ",round(pct_drop),"% ")
```

```
Percentage drop:  -2.0 %
```



5.2 Κώδικας *LOF*

Αλγόριθμος LOF (Local outlier factor)

Εισαγωγή βιβλιοθηκών

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib notebook
from sklearn.neighbors import LocalOutlierFactor as lof
```

Εισαγωγή δεδομένων

```
df = pd.read_csv("g_data.csv", delim_whitespace=True, header=None, names=["x", "y", "cluster_id"])
df.head()
```

	x	y	cluster_id
0	0.109393	0.085409	Cluster1
1	0.082571	0.101796	Cluster1
2	0.084990	0.113641	Cluster1
3	0.114611	0.115524	Cluster1
4	0.097356	0.095484	Cluster1

Δημιουργία μιας στήλης η οποία περιέχει ονόματα χρωμάτων ανάλογα με την ομάδα

```
df['id']=df.cluster_id.map({'Cluster1':'red', 'Cluster2':'green', 'Cluster3':'blue'})
df.head()
```

Ορισμος μοντελου

```
model = lof(n_neighbors=20, contamination=0.07)
model
```

```
LocalOutlierFactor(algorithm='auto', contamination=0.07, leaf_size=30,
          metric='minkowski', metric_params=None, n_jobs=None,
          n_neighbors=20, novelty=False, p=2)
```

Διαγραφή στήλης "id", αποθηκεύοντας στην "df3" μόνο τα καθαρά δεδομένα

```
df3 = (df2.drop(['id'], axis=1))
df3.head()
```

	x	y
0	0.109393	0.085409
1	0.082571	0.101796
2	0.084990	0.113641
3	0.114611	0.115524
4	0.097356	0.095484

	x	y	cluster_id	id
0	0.109393	0.085409	Cluster1	red
1	0.082571	0.101796	Cluster1	red
2	0.084990	0.113641	Cluster1	red
3	0.114611	0.115524	Cluster1	red
4	0.097356	0.095484	Cluster1	red

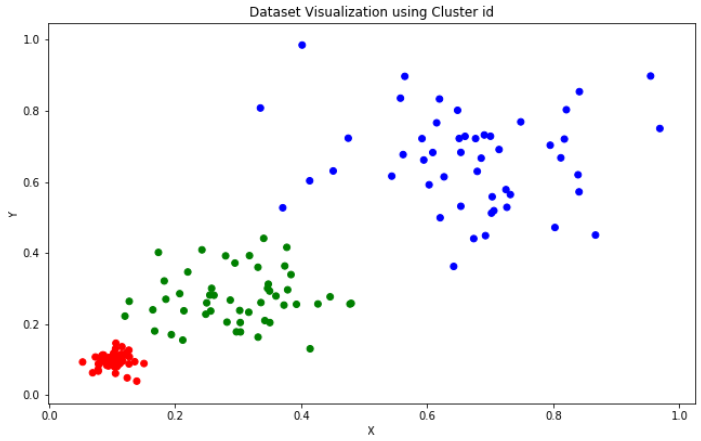
Διαγραφή στήλης "cluster_id"

```
df2 = (df.drop(['cluster_id'], axis=1))
df2.head()
```

	x	y	id
0	0.109393	0.085409	red
1	0.082571	0.101796	red
2	0.084990	0.113641	red
3	0.114611	0.115524	red
4	0.097356	0.095484	red

Απεικόνιση δεδομένων

```
plt.scatter(df2.x , df2.y , c = df2.id)
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Dataset Visualization using Cluster id");
```



Εφαρμογή αλγορίθμου στα καθαρά δεδομένα "df3"

```
label_outliers= (model.fit_predict(df3))
label_outliers
```

Τιμές κοντά στο -1 σύμφωνα με το σκορ είναι πιο πιθανές να είναι φυσιολογικές τιμές, ενώ τιμές μικρότερες από αυτό ως θόρυβος

```
model.negative_outlier_factor_
```

```
array([-1.00380763, -1.01683533, -1.02967782, -1.00734166, -0.96706344,
       -1.01434349, -0.9590511 , -1.2952214 , -0.97791989, -1.15755079,
       -0.95178085, -1.04973236, -1.1265852 , -1.49238866, -0.93987277,
       -0.97037049, -1.50400888, -1.02387354, -1.21419193, -1.25846419,
       -1.02380191, -0.95437123, -1.18896265, -1.13167501, -1.00002339,
       -1.67451729, -0.95731137, -0.97064429, -1.23165039, -1.80104619,
       -1.00031292, -0.99023152, -1.32966878, -0.97030526, -1.026465  ,
       -0.95316358, -0.95614898, -1.02714542, -1.11396408, -1.02456282,
       -1.05333996, -1.48579501, -0.99368454, -1.04142847, -1.01250529,
       -0.98843505, -1.11634346, -1.07923538, -0.99919899, -2.24381852,
       -1.766151  , -1.27449632, -1.12692904, -0.97432168, -1.27820199,
       -0.9704888 , -0.96644738, -1.02276772, -1.00755715, -0.94016282,
       -1.02579493, -0.99646179, -1.06202271, -0.97717106, -1.00981192,
       -1.07288422, -1.19450986, -1.08357173, -1.30769533, -0.95396644,
       -2.44356974, -1.06522261, -0.98646074, -0.94642842, -0.97270135,
       -2.67707631, -0.99173991, -1.09097198, -2.34775775, -0.95388594,
       -0.99906433, -1.03971753, -0.9700834 , -3.03743706, -1.11575885,
       -1.06741243, -0.96697889, -0.99082866, -0.97111909, -0.97526495,
       -1.07242664, -1.04272896, -1.12153734, -1.1137742 , -1.05013032,
       -0.97112131, -1.4272727 , -1.26455718, -1.20675952, -1.317114  ,
       -0.98307188, -0.94522896, -1.06303694, -0.96647195, -0.99623378,
       -1.02067965, -0.93137777, -1.3546493 , -1.06642321, -0.94061323,
       -1.15630081, -1.00202244, -1.17132663, -1.1663236 , -1.97220861,
       -0.97161699, -1.27222577, -1.02487489, -1.34256453, -1.01113613,
       -1.00167803, -0.98643156, -1.00090614, -1.0586435 , -0.97105627,
       -1.06866088, -1.40387724, -0.94676137, -1.52086529, -1.13501291,
       -0.96821254, -1.04148667, -1.07445501, -1.4704694 , -1.09334461,
       -1.80499025, -1.07648163, -1.66995817, -1.18884186, -1.02819638,
       -0.9451795 , -0.9327948 , -1.02155218, -1.3468348 , -0.96165528,
       -1.01644607, -1.25768173, -0.97240975, -1.15003272, -0.98542731])
```

Φιλτράρισμα δεδομένων με βάση το σκορ

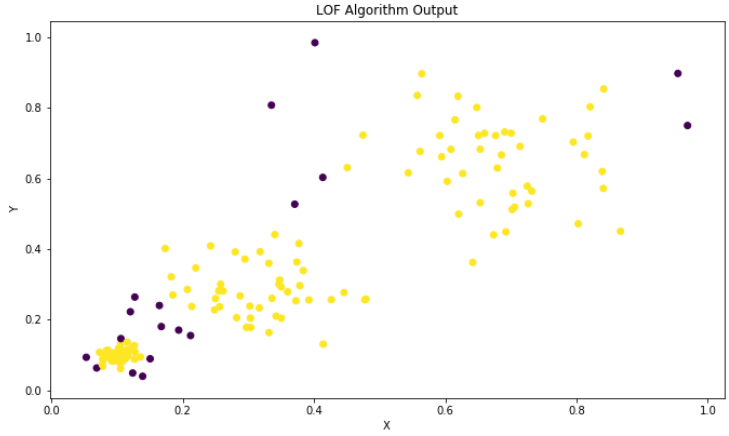
```
def filter_scores(x):
    if x > -1.4:
        return 1
    else:
        return -1
df3['outliers_based_on_score'] = df3.apply(lambda row: filter_scores(row['scores']), axis=1)
```

Δημιουργία στήλης "scores" η οποία έχει τα σκορ για όλα τα σημεία στα δεδομένα

```
df3['scores'] = model.negative_outlier_factor_
df3.head()
```

	x	y	scores
0	0.109393	0.085409	-1.003808
1	0.082571	0.101796	-1.016835
2	0.084990	0.113641	-1.029678
3	0.114611	0.115524	-1.007342
4	0.097356	0.095484	-0.967063

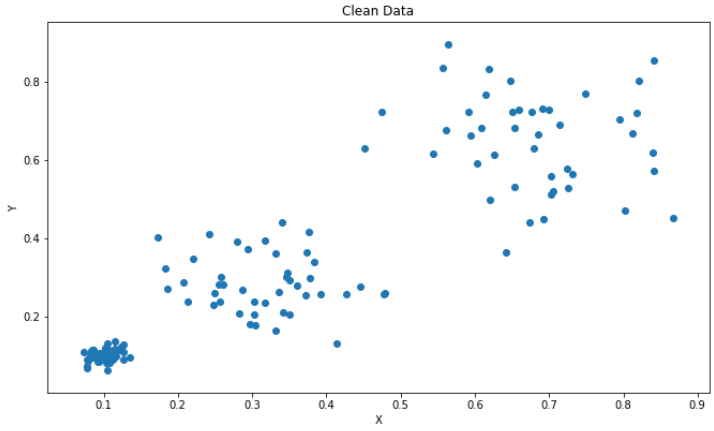
```
plt.scatter(df3.x, df3.y, c=df3.outliers_based_on_score)
plt.xlabel("X")
plt.ylabel("Y")
plt.title("LOF Algorithm Output");
```



Αφαίρεση σημείων οι οποίες έχουν σκορ μικρότερο από -1.4, και απεικόνιση καθαρών δεδομένων

```
filtered_data_based_on_score = df3.loc[df3['scores'] > -1.4]

clean_data = (filtered_data_based_on_score.drop(['scores', 'outliers_based_on_score'], axis=1))
```

```
plt.scatter(clean_data.x, clean_data.y,)
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Clean Data");
```



5.3 Κώδικας Isolation Forest

Αλγόριθμος Isolation Forest

Εισαγωγή βιβλιοθηκών ¶

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from sklearn.ensemble import IsolationForest as isfo
from sklearn import datasets
```

Θα χρησιμοποιηθούν τα ίδια δεδομένα με αυτά στο παράδειγμα του αλγορίθμου LOF

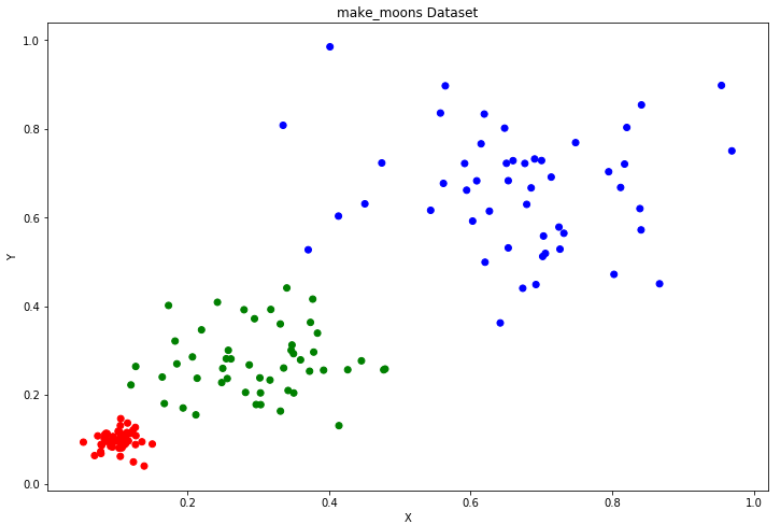
```
df = pd.read_csv("g_data.csv", delim_whitespace=True, header=None, names=["x", "y", "cluster_id"])
df.head()
```

	x	y	cluster_id
0	0.109393	0.085409	Cluster1
1	0.082571	0.101796	Cluster1
2	0.084990	0.113641	Cluster1
3	0.114611	0.115524	Cluster1
4	0.097356	0.095484	Cluster1

```
df['id']=df.cluster_id.map({'Cluster1':'red', 'Cluster2':'green', 'Cluster3':'blue'})
df = df.drop(['cluster_id'], axis=1)
df.head()
```

	x	y	id
0	0.109393	0.085409	red
1	0.082571	0.101796	red
2	0.084990	0.113641	red
3	0.114611	0.115524	red
4	0.097356	0.095484	red

```
plt.figure(figsize=(12, 8))
plt.scatter(df.x, df.y, c=df.id)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('make_moons Dataset');
```



```
df = df.drop(['id'], axis=1)
df.head()
```

	x	y
0	0.109393	0.085409
1	0.082571	0.101796
2	0.084990	0.113641
3	0.114611	0.115524
4	0.097356	0.095484

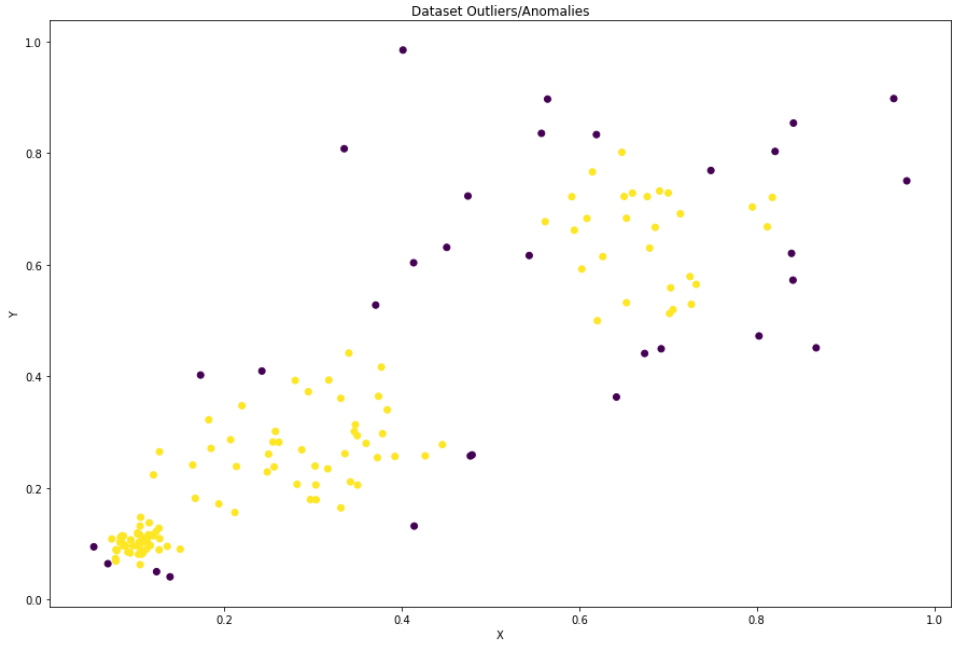
Για Unsupervized προσέγγιση, θα ορίσουμε την παράμετρο "contamination = 'auto' "

```
rng = np.random.RandomState(50)
model = isfo(max_samples=150, random_state=rng, contamination='auto', behaviour="new", n_estimators=100)
rng.rand()
```

```
0.49460164553802144
```

Χρησιμοποιώντας την ιδιότητα "fit_predict()" για την εφαρμογή του αλγορίθμου στα δεδομένα μας, μας επιστρέφει έναν πίνακα, οι τιμές [1] είναι σύμφωνα με τον αλγόριθμο και τις ρυθμίσεις του μοντέλου , φυσιολογικές τιμές. Ενώ [-1] ανώμαλα σημεία.

```
isfo_outliers = model.fit_predict(df)
df['iso_forest_outliers']= isfo_outliers
df.head(10)
```



	x	y	iso_forest_outliers
0	0.109393	0.085409	1
1	0.082571	0.101796	1
2	0.084990	0.113641	1
3	0.114611	0.115524	1
4	0.097356	0.095484	1
5	0.086612	0.113084	1
6	0.107348	0.110213	1
7	0.078011	0.068105	1
8	0.102780	0.116474	1
9	0.073398	0.107834	1

Απεικόνιση ανωμάλων σημείων

```
plt.figure(figsize=(15, 10))
plt.scatter(df.x, df.y, c=df.iso_forest_outliers)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Dataset Outliers/Anomalies');
```

Μέτρηση φυσιολογικών σημείων και μη στα δεδομένα

```
print('Normal points in Dataset:',np.count_nonzero(isfo_outliers > 0))
print('Abnormal Points in Dataset:',np.count_nonzero(isfo_outliers < 0))
```

```
Normal points in Dataset: 119
Abnormal Points in Dataset: 31
```

Supervised Προσέγγιση

```
df = df.drop(['iso_forest_outliers'], axis=1)
anomaly_score = model.decision_function(df)
df['anomaly_scores']= anomaly_score
df.head()
```

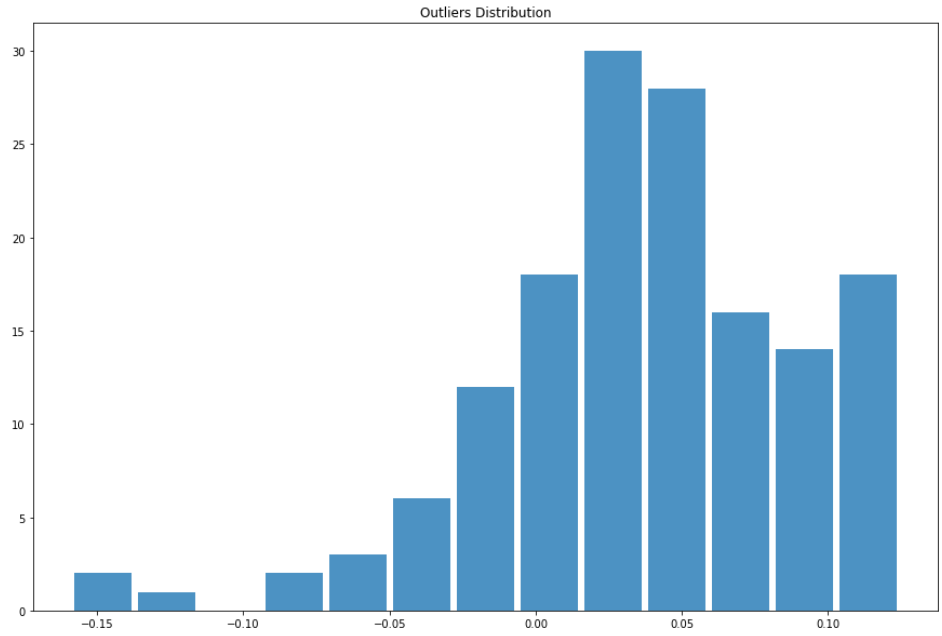
	x	y	anomaly_scores
0	0.109393	0.085409	0.097515
1	0.082571	0.101796	0.098587
2	0.084990	0.113641	0.089018
3	0.114611	0.115524	0.107727
4	0.097356	0.095484	0.113270

```
print('Maximum normality score:', max(anomaly_score), 'Minimum normality score:',min(anomaly_score))
```

```
Maximum normality score: 0.12456592379108061 Minimum normality score: -0.15891546767154807
```

Απεικόνιση διαγράμματος κατανομής των τιμών σκορ

```
plt.figure(figsize=(15, 10))
plt.hist(df.anomaly_scores, bins='auto', alpha=0.8,
         rwidth=0.9, align='mid',range=(min(anomaly_score), max(anomaly_score)));
plt.title('Outliers Distribution');
```



Φιλτράρισμα με βάση την τιμή σκορ

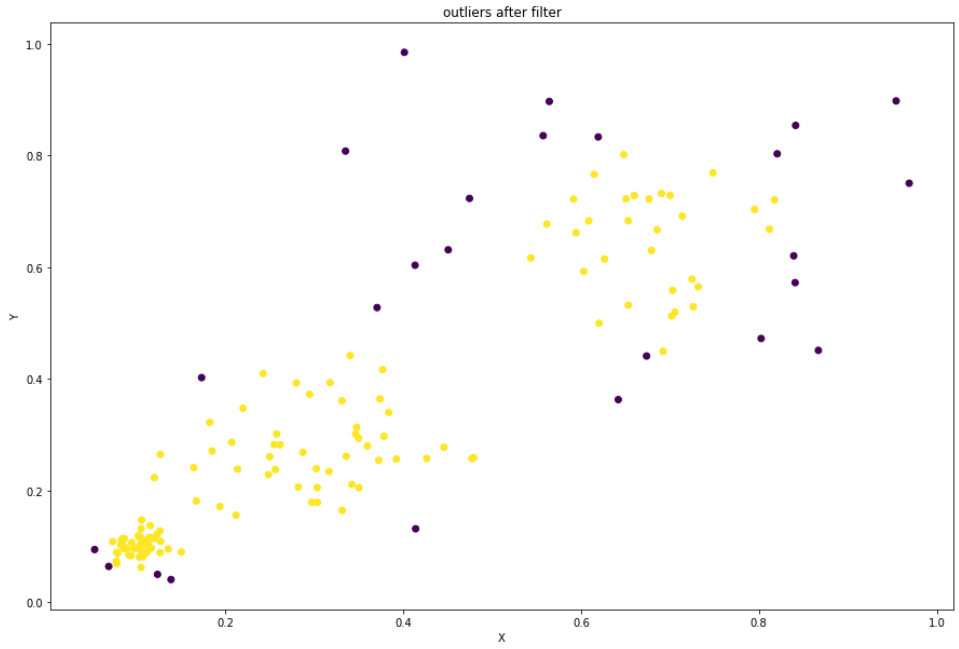
```
def filter_scores(x):
    if x < -0.01:
        return -1
    else:
        return 1
df['outliers_based_on_score'] = df.apply(lambda row: filter_scores(row['anomaly_scores']), axis=1)
```

```
df.head(15)
```

	x	y	anomaly_scores	outliers_based_on_score
0	0.109393	0.085409	0.097515	1
1	0.082571	0.101796	0.098587	1
2	0.084990	0.113641	0.089018	1
3	0.114611	0.115524	0.107727	1
4	0.097356	0.095484	0.113270	1
5	0.086612	0.113084	0.089852	1
6	0.107348	0.110213	0.114370	1
7	0.078011	0.068105	0.024021	1
8	0.102780	0.116474	0.106935	1
9	0.073398	0.107834	0.054991	1
10	0.113255	0.104047	0.113597	1
11	0.127328	0.108365	0.090176	1
12	0.105341	0.131101	0.085624	1
13	0.068981	0.063469	-0.021937	-1
14	0.104145	0.101859	0.124566	1

Απεικόνιση διαγράμματος διασποράς μετά το φιλτράρισμα

```
plt.figure(figsize=(15, 10))
plt.scatter(df.x, df.y, c=df.outliers_based_on_score)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('outliers after filter');
```



```
clean_data = df.loc[df['outliers_based_on_score'] != -1]
clean_data = clean_data.drop(['anomaly_scores','outliers_based_on_score'], axis=1)
clean_data.head()
```

	x	y
0	0.109393	0.085409
1	0.082571	0.101796
2	0.084990	0.113641
3	0.114611	0.115524
4	0.097356	0.095484

```
print('Original Data length:',len(df)
      ,'\nRemaining Data After Filtering:'
      ,len(clean_data),'\nPercentage drop (Round):'
      ,(round(((len(clean_data)-len(df))/len(df))*100)),'%')
```

```
Original Data length: 150
Remaining Data After Filtering: 125
Percentage drop (Round): -17 %
```

5.4 Κώδικας Tukey, IQR

Tukey Box Plot & IQR Method

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

Εισαγωγή συνόλου δεδομένων "Iris Dataset"

```
df = pd.read_csv("iris_dataset.txt", header=None, names=['sepal_length', 'sepal_width', 'petal_length'
                                                         ,'petal_width', 'type'])
```

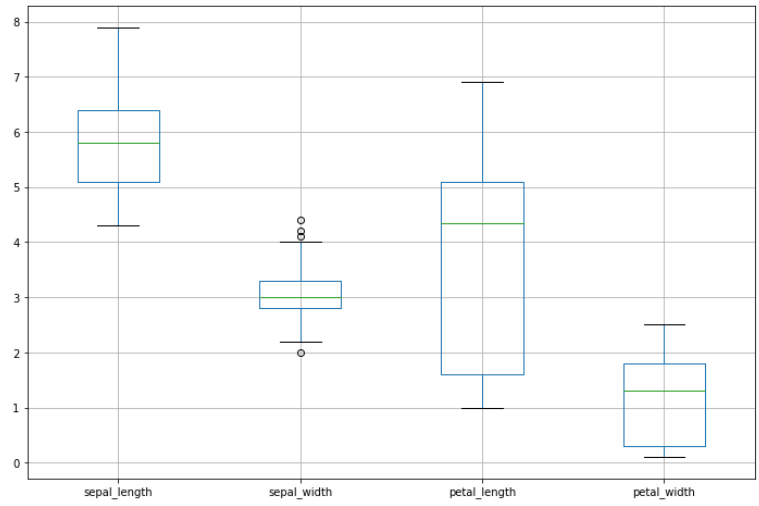
Περιγραφή των δεδομένων

```
#pd.options.display.float_format='{:.3f}'.format
df.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Απεικόνιση μέσω Box Plot ¶

```
plt.figure(figsize=(12, 8))
df.boxplot(return_type='dict')

plt.plot();
```



Παρατηρούμε πως στην στήλη "sepal_width" υπάρχουν ~4 τιμές εκτός ορίων

	sepal_length	sepal_width	petal_length	petal_width	type	outliers_based_on_IQR
0	5.1	3.5	1.4	0.2	Iris-setosa	Normal
1	4.9	3.0	1.4	0.2	Iris-setosa	Normal
2	4.7	3.2	1.3	0.2	Iris-setosa	Normal
3	4.6	3.1	1.5	0.2	Iris-setosa	Normal
4	5.0	3.6	1.4	0.2	Iris-setosa	Normal
5	5.4	3.9	1.7	0.4	Iris-setosa	Normal
6	4.6	3.4	1.4	0.3	Iris-setosa	Normal
7	5.0	3.4	1.5	0.2	Iris-setosa	Normal
8	4.4	2.9	1.4	0.2	Iris-setosa	Normal
9	4.9	3.1	1.5	0.1	Iris-setosa	Normal
10	5.4	3.7	1.5	0.2	Iris-setosa	Normal
11	4.8	3.4	1.6	0.2	Iris-setosa	Normal
12	4.8	3.0	1.4	0.1	Iris-setosa	Normal
13	4.3	3.0	1.1	0.1	Iris-setosa	Normal
14	5.8	4.0	1.2	0.2	Iris-setosa	Normal
15	5.7	4.4	1.5	0.4	Iris-setosa	Outlier
16	5.4	3.9	1.3	0.4	Iris-setosa	Normal
17	5.1	3.5	1.4	0.3	Iris-setosa	Normal
18	5.7	3.8	1.7	0.3	Iris-setosa	Normal
19	5.1	3.8	1.5	0.3	Iris-setosa	Normal

Μέθοδος Tukey IQR

```
# Εύρεση του διατεταρτημοριακού διαστήματος.

q1 = df["sepal_width"].quantile(0.25)
q3 = df["sepal_width"].quantile(0.75)
iqr = q3-q1

upper_threshold = q3+(iqr*1.5)
lower_threshold = q1-(iqr*1.5)
```

```
print( 'Lower Threshold:'+str(lower_threshold)+'\nUpper Threshold:'+str(upper_threshold))
```

```
Lower Threshold:2.05
Upper Threshold:4.05
```

```
outliers= df.loc[(df['sepal_width'] > upper_threshold) | (df['sepal_width'] < lower_threshold)]
```

```
outliers
```

	sepal_length	sepal_width	petal_length	petal_width	type
15	5.7	4.4	1.5	0.4	Iris-setosa
32	5.2	4.1	1.5	0.1	Iris-setosa
33	5.5	4.2	1.4	0.2	Iris-setosa
60	5.0	2.0	3.5	1.0	Iris-versicolor

```
def filter_iqr(x):
    if (x < 2.05 or x > 4.05):
        return "Outlier"
    else:
        return "Normal"
df['outliers_based_on_IQR'] = df.apply(lambda row: filter_iqr(row['sepal_width']), axis=1)
```

```
df.head(20)
```

5.5 Κώδικας αντιμετώπισης ελλειπόντων στοιχείων χρονοσειράς

Παράδειγμα αντιπετώπισης ελλειπόντων στοιχείων

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from datetime import datetime
import random
```

```
df = pd.read_csv("history_export_2018-12-04T18_30_11.csv",skiprows=12, header=None,
                 delimiter=';', names=['year',
                                       'month',
                                       'day',
                                       'hour',
                                       'minute',
                                       'temp',
                                       'shortwave_rad'])
```

```
df['Datetime']=pd.to_datetime(df[['year','month','day','hour']])
df.head()
```

	year	month	day	hour	minute	temp	shortwave_rad	Datetime
0	2018	11	27	0	0	16.79	0.0	2018-11-27 00:00:00
1	2018	11	27	1	0	16.71	0.0	2018-11-27 01:00:00
2	2018	11	27	2	0	16.58	0.0	2018-11-27 02:00:00
3	2018	11	27	3	0	16.18	0.0	2018-11-27 03:00:00
4	2018	11	27	4	0	15.94	0.0	2018-11-27 04:00:00

```
df.drop(columns=['year', 'month', 'day', 'hour', 'minute'], axis=1, inplace=True)
```

```
df.head()
```

	temp	shortwave_rad	Datetime
0	16.79	0.0	2018-11-27 00:00:00
1	16.71	0.0	2018-11-27 01:00:00
2	16.58	0.0	2018-11-27 02:00:00
3	16.18	0.0	2018-11-27 03:00:00

```
df.set_index('Datetime', inplace=True)
```

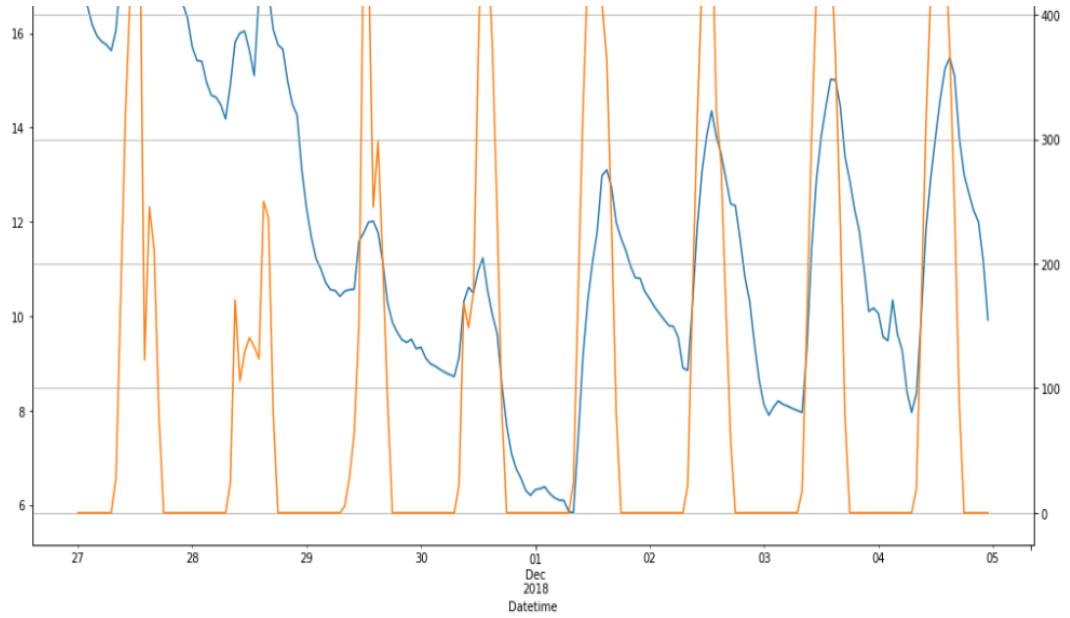
```
df.head()
```

Datetime	temp	shortwave_rad
2018-11-27 00:00:00	16.79	0.0
2018-11-27 01:00:00	16.71	0.0
2018-11-27 02:00:00	16.58	0.0
2018-11-27 03:00:00	16.18	0.0
2018-11-27 04:00:00	15.94	0.0

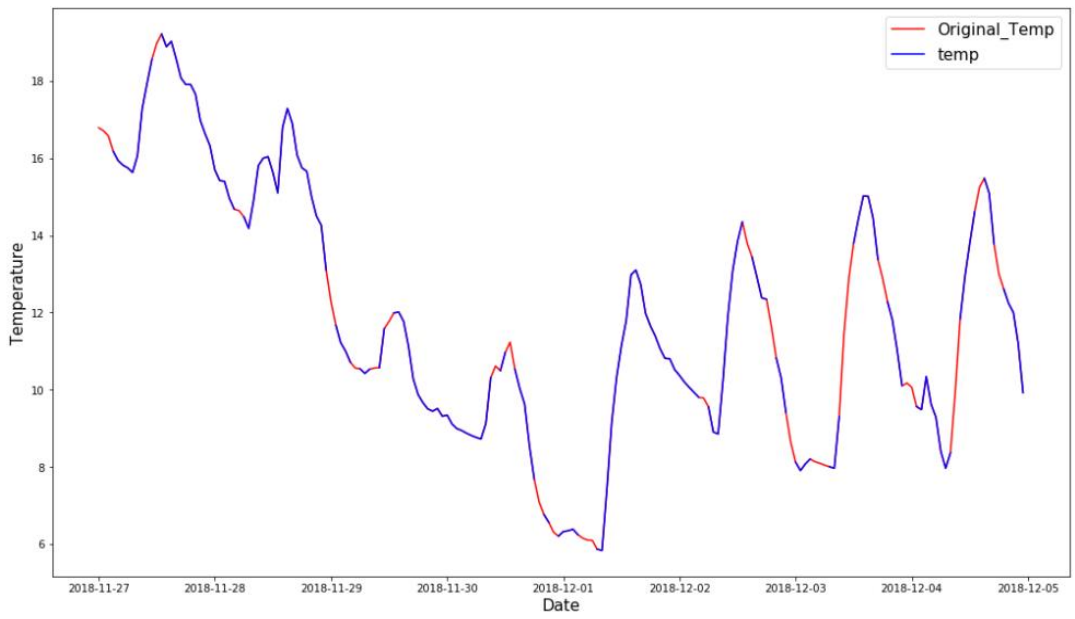
	temp
Datetime	
2018-11-27 00:00:00	16.79
2018-11-27 01:00:00	16.71
2018-11-27 02:00:00	16.58
2018-11-27 03:00:00	16.18
2018-11-27 04:00:00	15.94

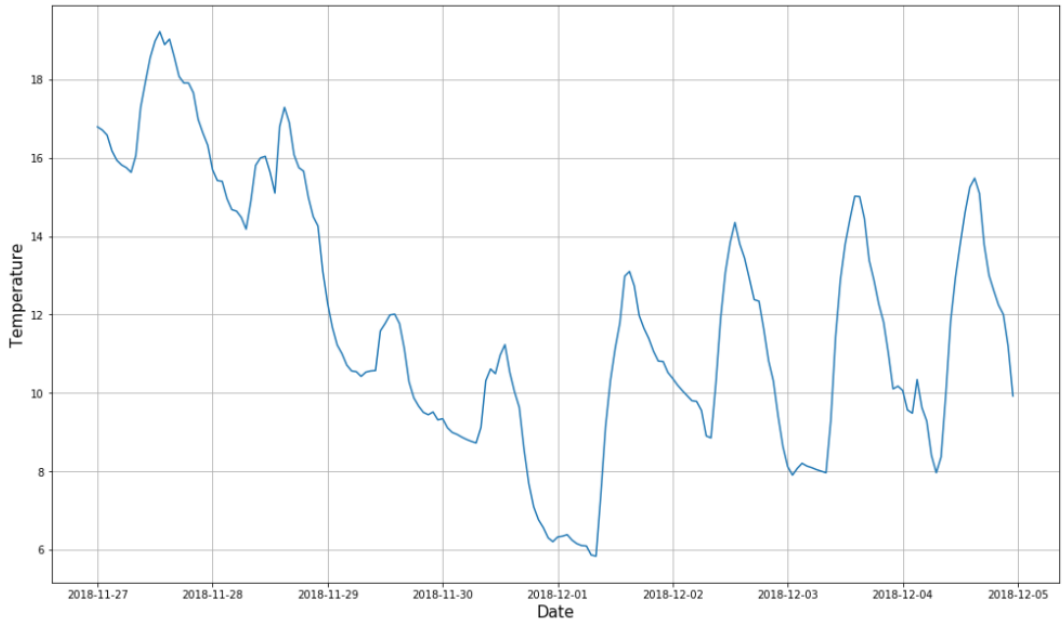
```
#df.index = pd.to_datetime(df.index)
```

```
plt.figure(figsize=(17, 10))
plt.plot(df)
plt.xlabel('Date', fontsize=15)
plt.ylabel('Temperature', fontsize=15)
plt.grid(True)
```




```
df.drop('shortwave_rad', axis=1, inplace=True)
df.head()
```






```
df_mv = df.copy()
```

Τυχαία επιλογή [~30] τιμών θερμοκρασίας και αντικατάσταση τους με Nan

```
for k in range(0,30):
        df_mv.temp[random.randint(0, 192)] = np.nan
```

Πλήθος τιμών όπου αντικαταστάθηκε

```
df_mv.temp.isnull().sum()
```

```
28
```

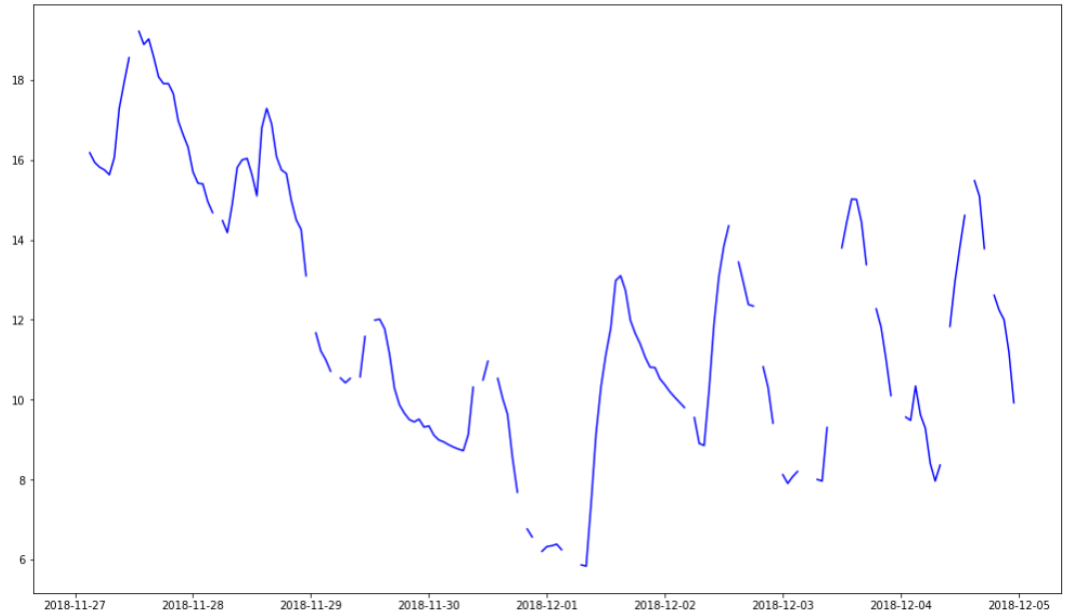
Δημιουργία 3ης στήλης όπου περιέχει τις πραγματικές τιμές θερμοκρασίας, και απεικόνιση.

```
df_mv['Original_Temp']= df['temp']
```

```
plt.figure(figsize=(17, 10))
plt.plot(df_mv.Original_Temp, c='r')
plt.plot(df_mv.temp, c='b')
plt.xlabel('Date', fontsize=15)
plt.ylabel('Temperature', fontsize=15)
plt.legend(loc='upper right', fontsize=15);
```

Τα σημεία με χρώμα μπλε είναι ότι έχει απομείνει έπειτα από αφαίρεση τυχαίων τιμών, η κόκκινη δείχνει τις πραγματικές τιμές στα κενά αυτά. ¶

```
plt.figure(figsize=(17, 10))
plt.plot(df_mv.temp, c='b');
```




```
df_mv.head(20)
```

Datetime	temp	Original_Temp
2018-11-27 00:00:00	NaN	16.79
2018-11-27 01:00:00	16.71	16.71
2018-11-27 02:00:00	NaN	16.58
2018-11-27 03:00:00	16.18	16.18
2018-11-27 04:00:00	15.94	15.94
2018-11-27 05:00:00	15.82	15.82
2018-11-27 06:00:00	15.75	15.75
2018-11-27 07:00:00	15.63	15.63
2018-11-27 08:00:00	16.06	16.06
2018-11-27 09:00:00	17.29	17.29
2018-11-27 10:00:00	17.94	17.94
2018-11-27 11:00:00	18.56	18.56
2018-11-27 12:00:00	NaN	18.98
2018-11-27 13:00:00	19.22	19.22
2018-11-27 14:00:00	18.89	18.89
2018-11-27 15:00:00	19.03	19.03
2018-11-27 16:00:00	18.58	18.58
2018-11-27 17:00:00	18.08	18.08
2018-11-27 18:00:00	17.91	17.91
2018-11-27 19:00:00	17.91	17.91

Μέθοδος "fillna()"

```
df_mv_copy = df_mv.copy()
```

Αντικατάσταση με έναν αριθμό

```
df_mv_copy = df_mv_copy.fillna(1)
```

```
df_mv_copy.head(10)
```

Datetime	temp	Original_Temp
2018-11-27 00:00:00	1.00	16.79
2018-11-27 01:00:00	16.71	16.71
2018-11-27 02:00:00	1.00	16.58
2018-11-27 03:00:00	16.18	16.18
2018-11-27 04:00:00	15.94	15.94
2018-11-27 05:00:00	15.82	15.82
2018-11-27 06:00:00	15.75	15.75
2018-11-27 07:00:00	15.63	15.63
2018-11-27 08:00:00	16.06	16.06
2018-11-27 09:00:00	17.29	17.29

Αντικατάσταση χρησιμοποιώντας λεξικό

```
df_mv_copy = df_mv.fillna({
    'temp': -9999
})
#Επιλογή στήλης που θέλουμε να αντικαταστήσουμε τα σημεία στα οποία λείπουν δεδομένα.
```

```
df_mv_copy.head(10)
```

Datetime	temp	Original_Temp
2018-11-27 00:00:00	-9999.00	16.79
2018-11-27 01:00:00	16.71	16.71
2018-11-27 02:00:00	-9999.00	16.58
2018-11-27 03:00:00	16.18	16.18
2018-11-27 04:00:00	15.94	15.94
2018-11-27 05:00:00	15.82	15.82
2018-11-27 06:00:00	15.75	15.75
2018-11-27 07:00:00	15.63	15.63
2018-11-27 08:00:00	16.06	16.06
2018-11-27 09:00:00	17.29	17.29

Μέθοδος "ffill", αντικαθιστά την τιμή που λείπει με βάση την προηγουμένη τιμή

```
df_mv_copy = df_mv.copy()
df_mv_copy = df_mv.fillna(method='ffill')

# df_mv_copy = df_mv.fillna(method='bfill', axis='columns') Αντικαθιστά οριζόντια,
# χρησιμοποιώντας τις τιμές που έχουν οι στήλες αριστερά.
```

```
df_mv_copy.head(8)
```

Datetime	temp	Original_Temp
2018-11-27 00:00:00	NaN	16.79
2018-11-27 01:00:00	16.71	16.71
2018-11-27 02:00:00	16.71	16.58
2018-11-27 03:00:00	16.18	16.18
2018-11-27 04:00:00	15.94	15.94
2018-11-27 05:00:00	15.82	15.82
2018-11-27 06:00:00	15.75	15.75
2018-11-27 07:00:00	15.63	15.63

Μέθοδος "bfill", αντικαθιστά την τιμή που λείπει με βάση την επόμενη τιμή

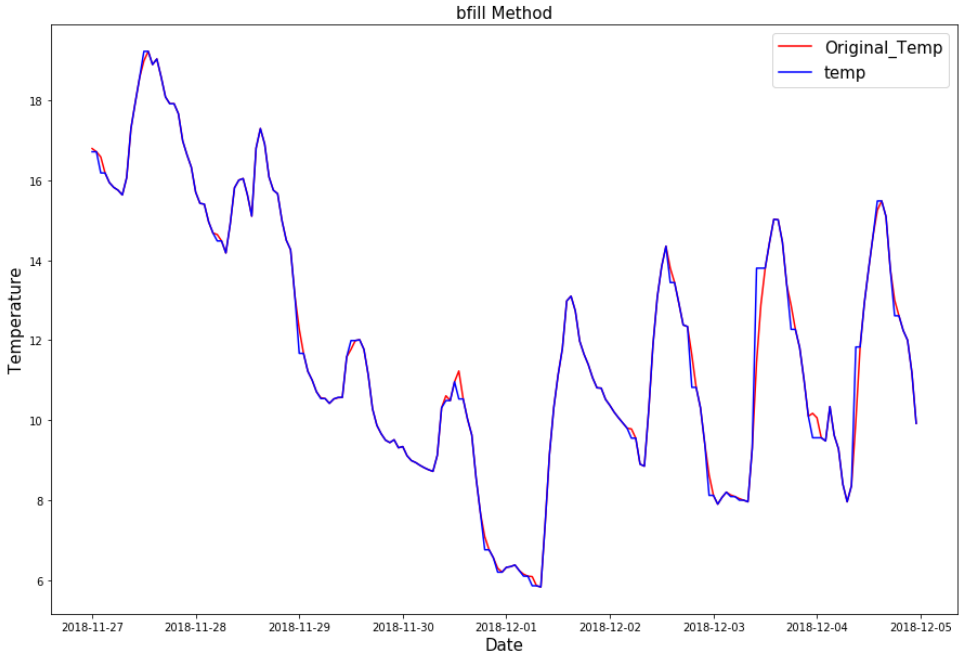
```
df_mv_copy = df_mv.copy()
```

```
df_mv_copy = df_mv.fillna(method='bfill')

# df_mv_copy = df_mv.fillna(method='bfill', axis='columns') Αντικαθιστά οριζόντια,
# χρησιμοποιώντας τις τιμές που έχουν οι στήλες δεξιά.
```

```
df_mv_copy.head(10)
```

Datetime	temp	Original_Temp
2018-11-27 00:00:00	16.71	16.79
2018-11-27 01:00:00	16.71	16.71
2018-11-27 02:00:00	16.18	16.58
2018-11-27 03:00:00	16.18	16.18
2018-11-27 04:00:00	15.94	15.94
2018-11-27 05:00:00	15.82	15.82
2018-11-27 06:00:00	15.75	15.75
2018-11-27 07:00:00	15.63	15.63
2018-11-27 08:00:00	16.06	16.06
2018-11-27 09:00:00	17.29	17.29

```
plt.figure(figsize=(15, 10))
plt.plot(df_mv_copy.Original_Temp, c='r')
plt.plot(df_mv_copy.temp, c='b')
plt.xlabel('Date', fontsize=15)
plt.ylabel('Temperature', fontsize=15)
plt.legend(loc='upper right', fontsize=15)
plt.title('bfill Method', fontsize=15);
```



Διαγραφή γραμμών όπου λείπουν τα δεδομένα

```
df_mv_copy = df_mv.copy()
```

```
df_mv_copy = df_mv_copy.dropna()
```

```
df_mv_copy.head(10)
```

Datetime	temp	Original_Temp
2018-11-27 01:00:00	16.71	16.71
2018-11-27 03:00:00	16.18	16.18
2018-11-27 04:00:00	15.94	15.94
2018-11-27 05:00:00	15.82	15.82
2018-11-27 06:00:00	15.75	15.75
2018-11-27 07:00:00	15.63	15.63
2018-11-27 08:00:00	16.06	16.06
2018-11-27 09:00:00	17.29	17.29
2018-11-27 10:00:00	17.94	17.94
2018-11-27 11:00:00	18.56	18.56

Αντικατάσταση με το μεσο όρο της στήλης

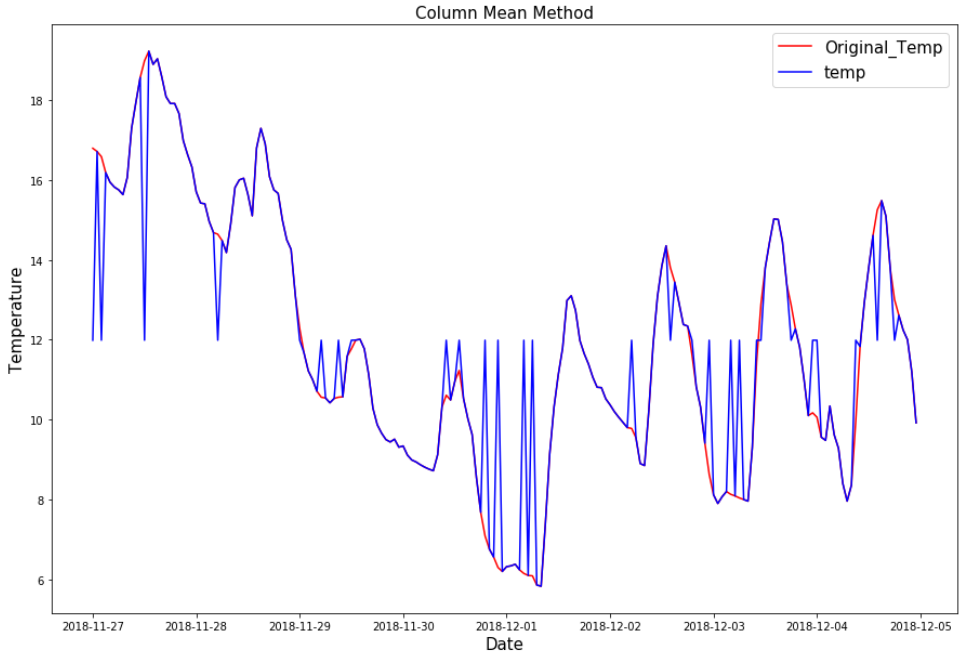
```
df_mv_copy = df_mv.copy()
```

```
df_mv_copy['temp'].fillna((df_mv_copy['temp'].mean()), inplace=True)
df_mv_copy['temp'].mean()
```

```
11.986280487804876
```

```
df_mv_copy.head(10)
```

```
plt.figure(figsize=(15, 10))
plt.plot(df_mv_copy.Original_Temp, c='r')
plt.plot(df_mv_copy.temp, c='b')
plt.xlabel('Date', fontsize=15)
plt.ylabel('Temperature', fontsize=15)
plt.legend(loc='upper right', fontsize=15)
plt.title('Column Mean Method', fontsize=15);
```



Προφανώς στην συγκεκριμένη περίπτωση χρησιμοποιώντας τον γενικό μέσο όρο δεν έχει καλά αποτελέσματα. Παρακάτω η μέθοδος "Rolling Mean" όπου αντικαθίσταται οι Nan με βάση τον τοπικό μέσο όρο

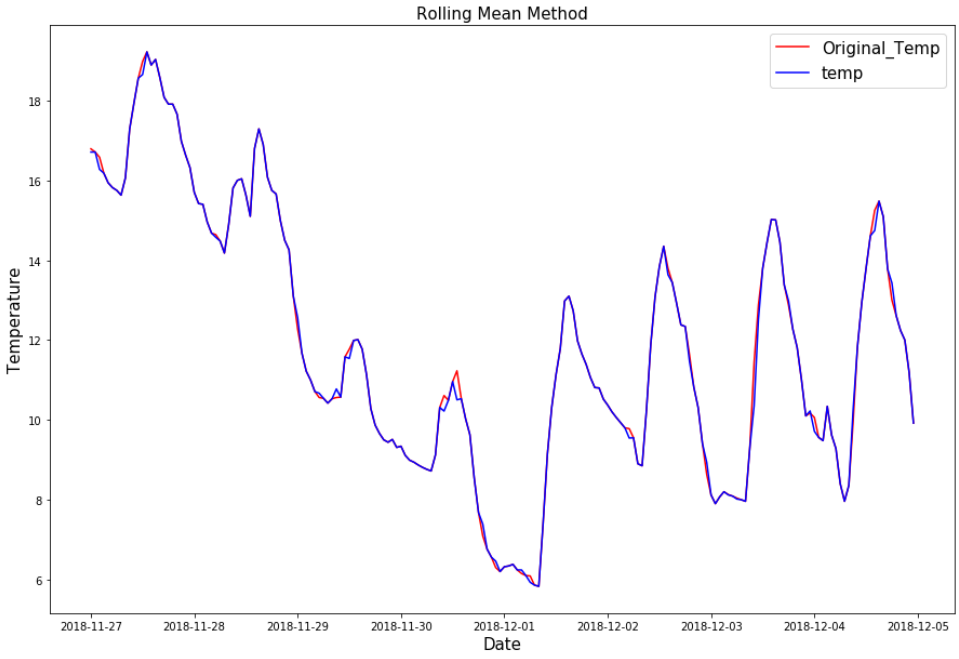
```
df_mv_copy = df_mv.copy()
```

```
df_mv_copy_2 = df_mv_copy.fillna(df_mv_copy.rolling(min_periods=1, center=True, window=5).mean());
df_mv_copy.update(df_mv_copy_2)
```

```
df_mv_copy.head()
```

Datetime	temp	Original_Temp
2018-11-27 00:00:00	16.710000	16.79
2018-11-27 01:00:00	16.710000	16.71
2018-11-27 02:00:00	16.276667	16.58
2018-11-27 03:00:00	16.180000	16.18
2018-11-27 04:00:00	15.940000	15.94

```
plt.figure(figsize=(15, 10))
plt.plot(df_mv_copy.Original_Temp, c='r')
plt.plot(df_mv_copy.temp, c='b')
plt.xlabel('Date', fontsize=15)
plt.ylabel('Temperature', fontsize=15)
plt.legend(loc='upper right', fontsize=15)
plt.title('Rolling Mean Method', fontsize=15);
```



Μέθοδος "Interpolate()"

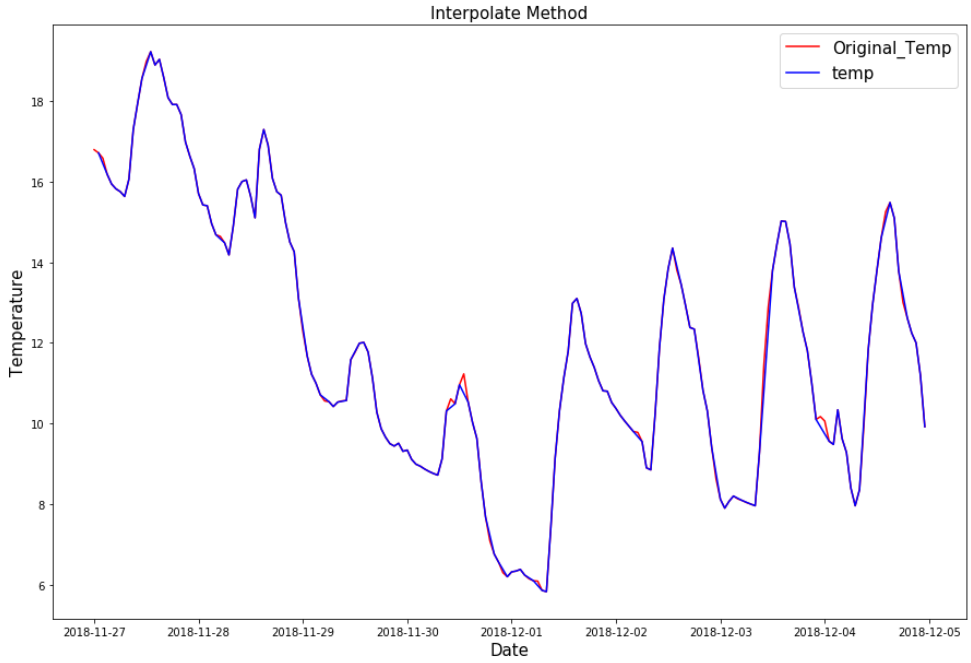
```
df_mv_copy = df_mv.copy()
```

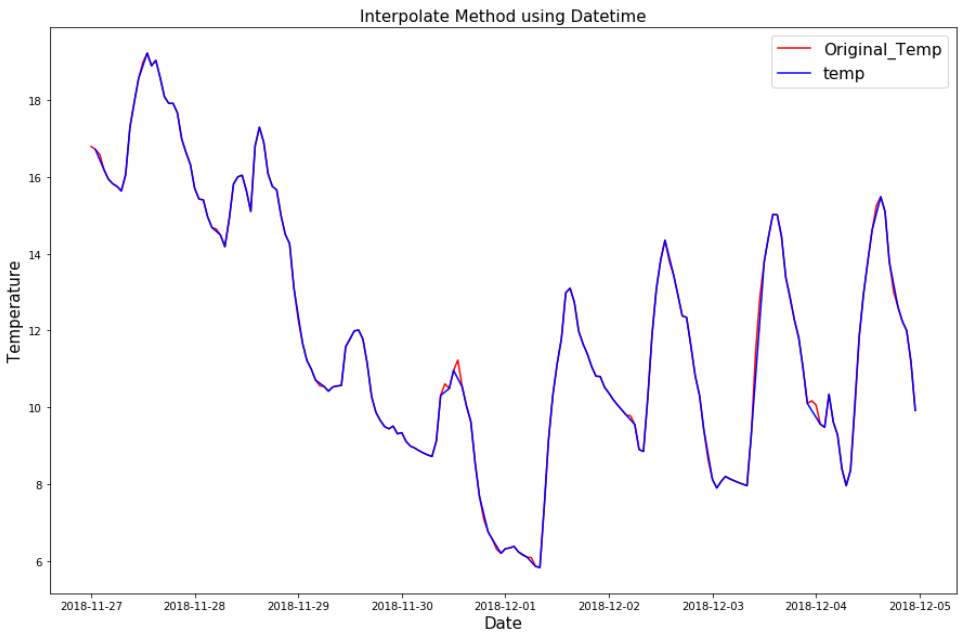
```
pd.options.display.float_format = '{:,.2f}'.format
df_mv_copy = df_mv_copy.interpolate()
#Default, η μεθοδος interpolate() βρισκει τις τιμες που λειπουν γραμμικα.
```

```
df_mv_copy.head(10)
```

Datetime	temp	Original_Temp
2018-11-27 00:00:00	nan	16.79
2018-11-27 01:00:00	16.71	16.71
2018-11-27 02:00:00	16.45	16.58
2018-11-27 03:00:00	16.18	16.18
2018-11-27 04:00:00	15.94	15.94
2018-11-27 05:00:00	15.82	15.82
2018-11-27 06:00:00	15.75	15.75
2018-11-27 07:00:00	15.63	15.63
2018-11-27 08:00:00	16.06	16.06
2018-11-27 09:00:00	17.29	17.29

```
plt.figure(figsize=(15, 10))
plt.plot(df_mv_copy.Original_Temp, c='r')
plt.plot(df_mv_copy.temp, c='b')
plt.xlabel('Date', fontsize=15)
plt.ylabel('Temperature', fontsize=15)
plt.legend(loc='upper right', fontsize=15)
plt.title('Interpolate Method', fontsize=15);
```


Interpolate Method
Original_Temp
temp
Temperature
Date
18
16
14
12
10
8
6
2018-11-27
2018-11-28
2018-11-29
2018-11-30
2018-12-01
2018-12-02
2018-12-03
2018-12-04
2018-12-05

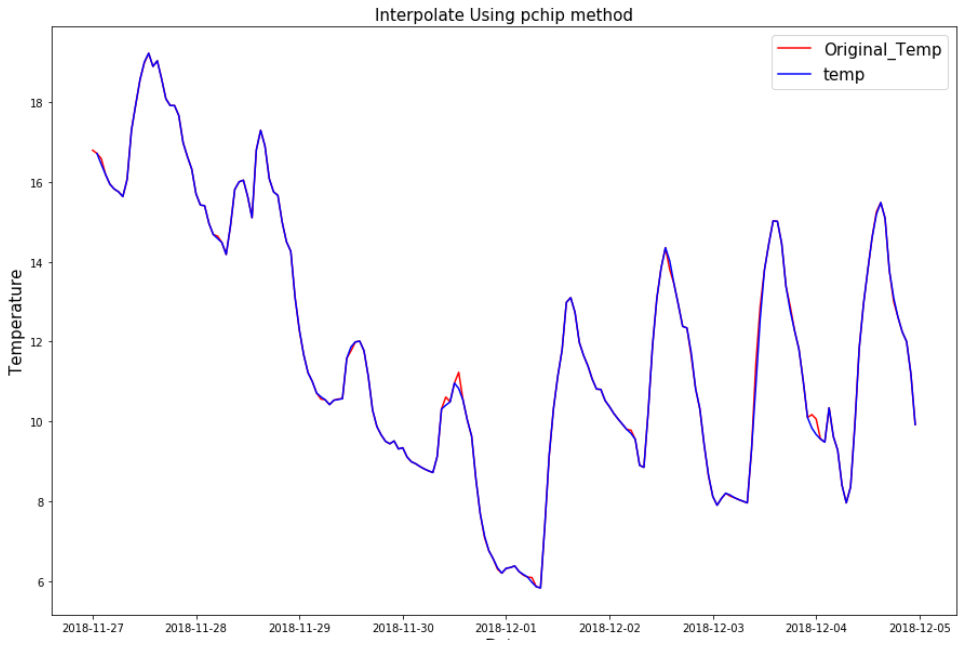

Interpolate Method using Datetime
Original_Temp
temp
Temperature
Date
18
16
14
12
10
8
6
2018-11-27
2018-11-28
2018-11-29
2018-11-30
2018-12-01
2018-12-02
2018-12-03
2018-12-04
2018-12-05

```
df_mv_copy = df_mv.copy()
```

```
df_mv_copy.interpolate(method='pchip', inplace=True)
df_mv_copy.head()
```

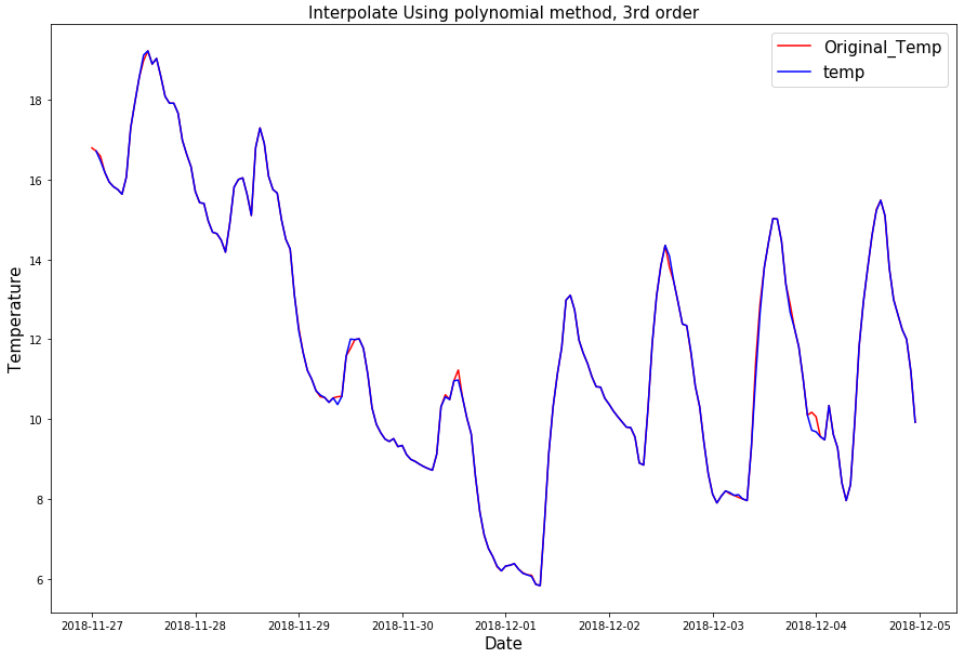
Datetime	temp	Original_Temp
2018-11-27 00:00:00	nan	16.79
2018-11-27 01:00:00	16.71	16.71
2018-11-27 02:00:00	16.44	16.58
2018-11-27 03:00:00	16.18	16.18
2018-11-27 04:00:00	15.94	15.94

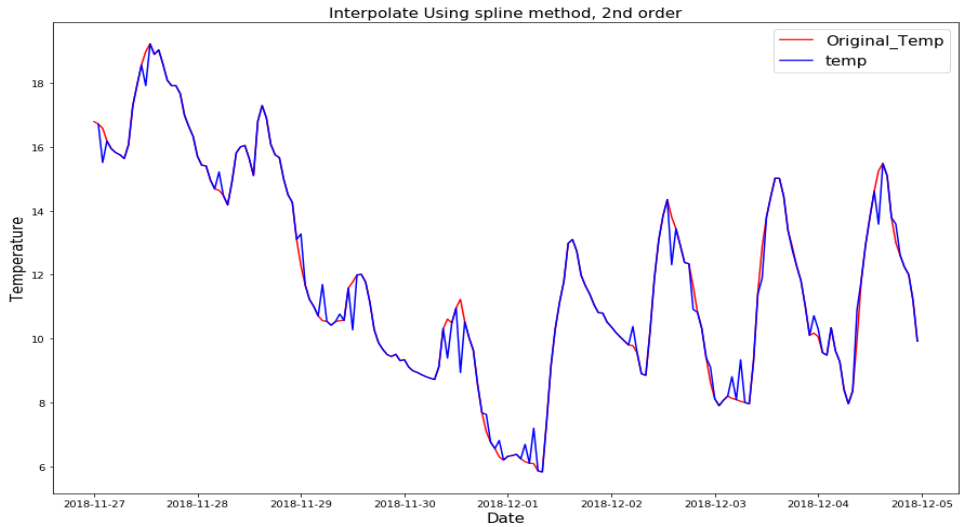
```
plt.figure(figsize=(15, 10))
plt.plot(df_mv_copy.Original_Temp, c='r')
plt.plot(df_mv_copy.temp, c='b')
plt.xlabel('Date', fontsize=15)
plt.ylabel('Temperature', fontsize=15)
plt.legend(loc='upper right', fontsize=15);
plt.title('Interpolate Using ''pchip'' method', fontsize=15);
```



Datetime	temp	Original_Temp
2018-11-27 00:00:00	nan	16.79
2018-11-27 01:00:00	16.71	16.71
2018-11-27 02:00:00	16.47	16.58
2018-11-27 03:00:00	16.18	16.18
2018-11-27 04:00:00	15.94	15.94

```
plt.figure(figsize=(15, 10))
plt.plot(df_mv_copy.Original_Temp, c='r')
plt.plot(df_mv_copy.temp, c='b')
plt.xlabel('Date', fontsize=15)
plt.ylabel('Temperature', fontsize=15)
plt.legend(loc='upper right', fontsize=15);
plt.title('Interpolate Using polynomial method, 3rd order', fontsize=15);
```





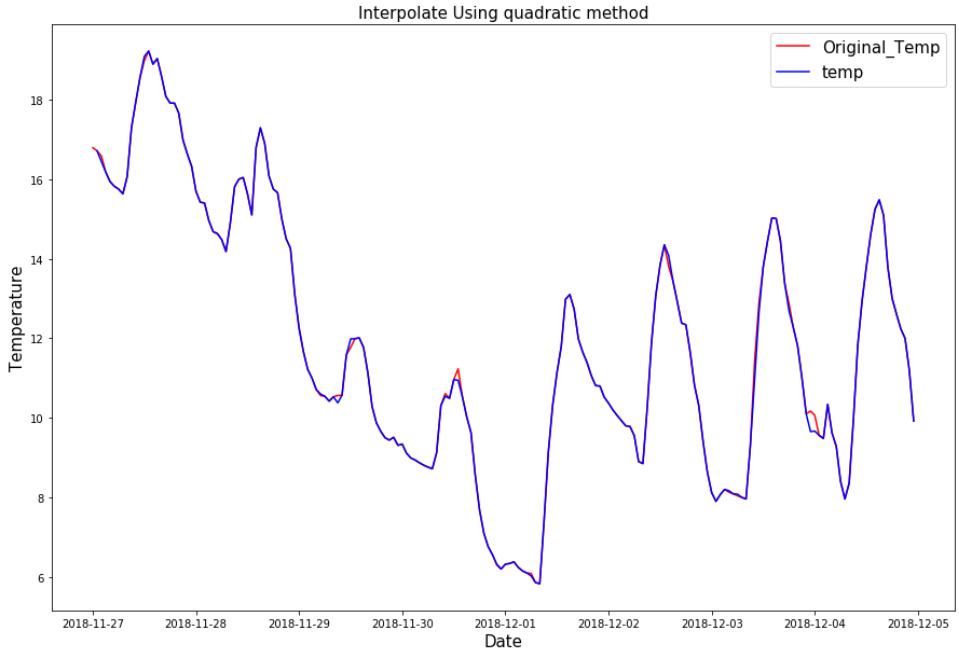
```
df_mv_copy = df_mv.copy()
```

```
df_mv_copy.interpolate(method='quadratic', inplace=True)
```

```
df_mv_copy.head()
```

Datetime	temp	Original_Temp
2018-11-27 00:00:00	nan	16.79
2018-11-27 01:00:00	16.71	16.71
2018-11-27 02:00:00	16.44	16.58
2018-11-27 03:00:00	16.18	16.18
2018-11-27 04:00:00	15.94	15.94

```
plt.figure(figsize=(15, 10))
plt.plot(df_mv_copy.Original_Temp, c='r')
plt.plot(df_mv_copy.temp, c='b')
plt.xlabel('Date', fontsize=15)
plt.ylabel('Temperature', fontsize=15)
plt.legend(loc='upper right', fontsize=15);
plt.title('Interpolate Using quadratic method', fontsize=15);
```



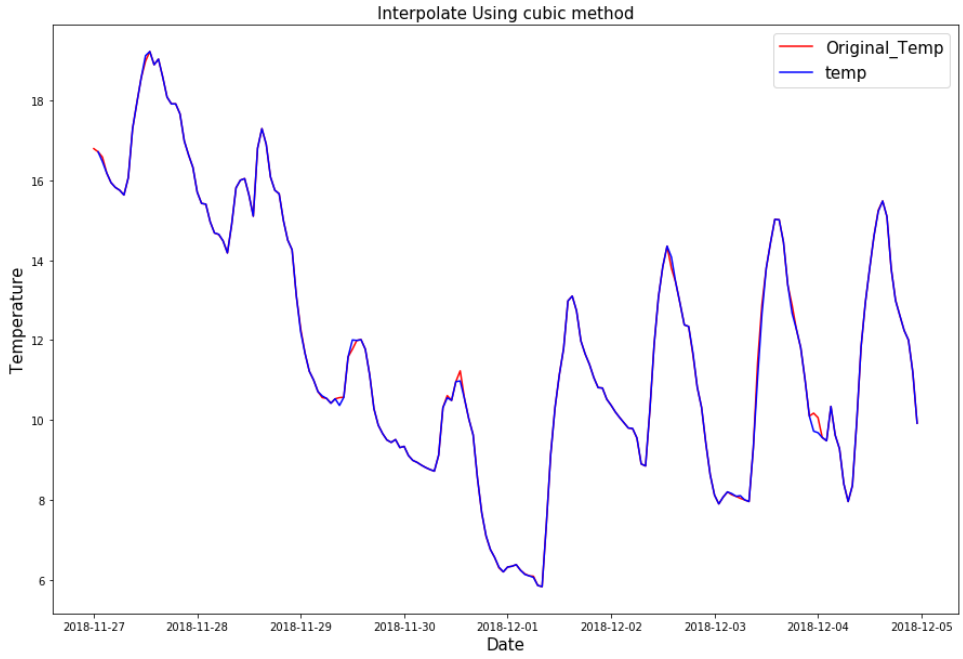
```
df_mv_copy = df_mv.copy()
```

```
df_mv_copy.interpolate(method='cubic', order=6, inplace=True)
```

```
df_mv_copy.head()
```

Datetime	temp	Original_Temp
2018-11-27 00:00:00	nan	16.79
2018-11-27 01:00:00	16.71	16.71
2018-11-27 02:00:00	16.47	16.58
2018-11-27 03:00:00	16.18	16.18
2018-11-27 04:00:00	15.94	15.94

```
plt.figure(figsize=(15, 10))
plt.plot(df_mv_copy.Original_Temp, c='r')
plt.plot(df_mv_copy.temp, c='b')
plt.xlabel('Date', fontsize=15)
plt.ylabel('Temperature', fontsize=15)
plt.legend(loc='upper right', fontsize=15);
plt.title('Interpolate Using cubic method', fontsize=15);
```



5.6 Κώδικας PCA

```
import matplotlib.pyplot as plt
%matplotlib notebook
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from numpy import linalg as LA
from mpl_toolkits.mplot3d import Axes3D
```

Εισαγωγή συνόλου δεδομένων

```
df = pd.read_csv('iris_dataset.txt', header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
                                                         'type'])
```

```
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	type
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Μετατροπή στήλης "type" σε χρώματα για λόγους απεικόνισης

```
df['type']=df.type.map({'Iris-setosa':'red', 'Iris-versicolor':'green', 'Iris-virginica':'blue'})
```

```
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	type
0	5.1	3.5	1.4	0.2	red
1	4.9	3.0	1.4	0.2	red
2	4.7	3.2	1.3	0.2	red
3	4.6	3.1	1.5	0.2	red
4	5.0	3.6	1.4	0.2	red

```
df_target=df['type']
df_target.head()
```

```
0    red
1    red
2    red
3    red
4    red
Name: type, dtype: object
```

```
df.drop(['type'], axis=1, inplace=True)
```

```
df.shape
```

```
(150, 4)
```

```
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Κανονικοποίηση των δεδομένων

```
df_std = StandardScaler().fit_transform(df)
```

Ανάστροφος

```
df_std_trans = df_std.T
```

Πίνακας συνδιακύμανσης. Για σύνολο δεδομένων με 4 χαρακτηριστικά ο πίνακας συνδιακύμανσης θα ειναι (4x4) πίνακας.

Όπου οι διαγώνιες τιμές του είναι οι διακυμάνσεις, και οι μη-διαγώνιες είναι οι συνδιακύμανσης μεταξύ τους.

```
df_std_trans_cov = np.cov(df_std_trans)
```

```
df_std_trans_cov
```

```
array([[ 1.00671141, -0.11010327,  0.87760486,  0.82344326],
       [-0.11010327,  1.00671141, -0.42333835, -0.358937  ],
       [ 0.87760486, -0.42333835,  1.00671141,  0.96921855],
       [ 0.82344326, -0.358937  ,  0.96921855,  1.00671141]])
```

Εύρεση ιδιοτιμών και ιδιοδιανυσμάτων

```
eig_val, eig_vec = np.linalg.eig(df_std_trans_cov)
```

```
eig_vec
```

```
array([[ 0.52237162, -0.37231836, -0.72101681,  0.26199559],
       [-0.26335492, -0.92555649,  0.24203288, -0.12413481],
       [ 0.58125401, -0.02109478,  0.14089226, -0.80115427],
       [ 0.56561105, -0.06541577,  0.6338014 ,  0.52354627]])
```

```
eig_val
```

```
array([2.93035378, 0.92740362, 0.14834223, 0.02074601])
```

Υπολογισμός νέου συνόλου δεδομένων, χρησιμοποιώντας 2 PC

```
pro_x = df_std.dot(eig_vec.T[0])
```

```
pro_y = df_std.dot(-eig_vec.T[1])
```

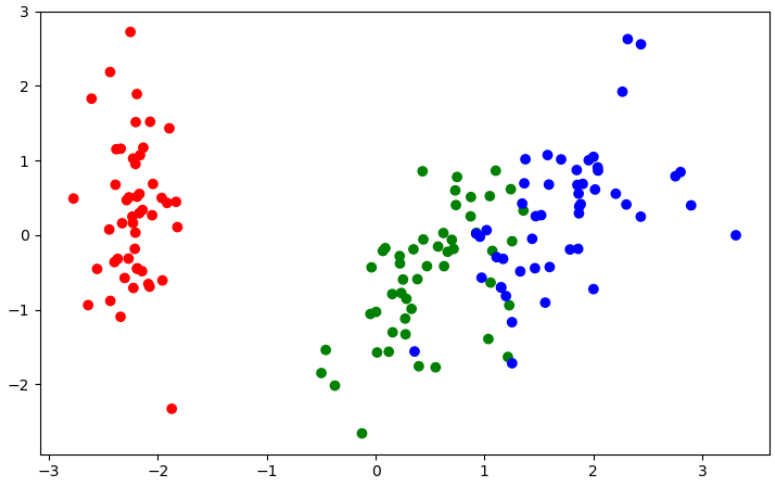
```
result = pd.DataFrame({'pc_1': pro_x.T, 'pc_2':pro_y.T}, columns=['pc_1', 'pc_2'])
```

```
result.head(10)
```

	pc_1	pc_2
0	-2.264542	0.505704
1	-2.086426	-0.655405
2	-2.367950	-0.318477
3	-2.304197	-0.575368
4	-2.388777	0.674767
5	-2.070537	1.518549
6	-2.445711	0.074563
7	-2.233842	0.247614
8	-2.341958	-1.095146
9	-2.188676	-0.448629

Απεικόνιση του νέου συνόλου δεδομένων

```
plt.scatter(x=result.pc_1, y=result.pc_2, c=df_target);
```

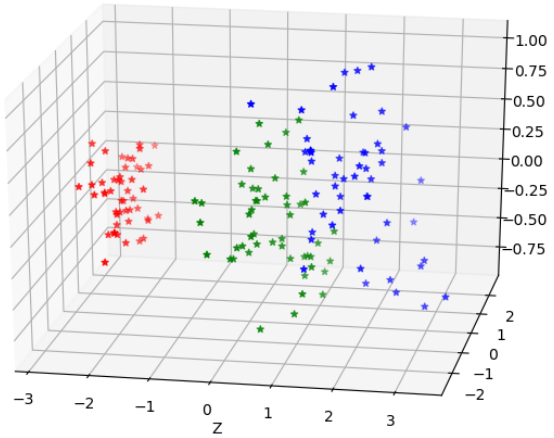


Χρήση 3 PC και τρισδιάστατη απεικόνιση των δεδομένων

```
pro_z = df_std.dot(eig_vec.T[2])
```

```
result_3d = pd.DataFrame({'pc_1': pro_x.T, 'pc_2':pro_y.T, 'pc_3': pro_z.T}, columns=['pc_1', 'pc_2', 'pc_3'])
```

```
fig = plt.figure();
ax = fig.add_subplot(111, projection='3d');
ax.scatter(result_3d.pc_1, result_3d.pc_2, result_3d.pc_3, c=df_target, marker='*')
ax.set_xlabel('X');
ax.set_xlabel('Y');
ax.set_xlabel('Z');
plt.show()
```



Εφαρμογή PCA με sklearn

```
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Ορισμός PCA

```
pca = PCA(n_components=2)
#pca = PCA(.95) Όπου διαλέγονται όσα PC χρειάζονται για την (...) 95%
```

Κανονικοποίηση των δεδομένων

```
scaled_df = StandardScaler().fit_transform(df)
```

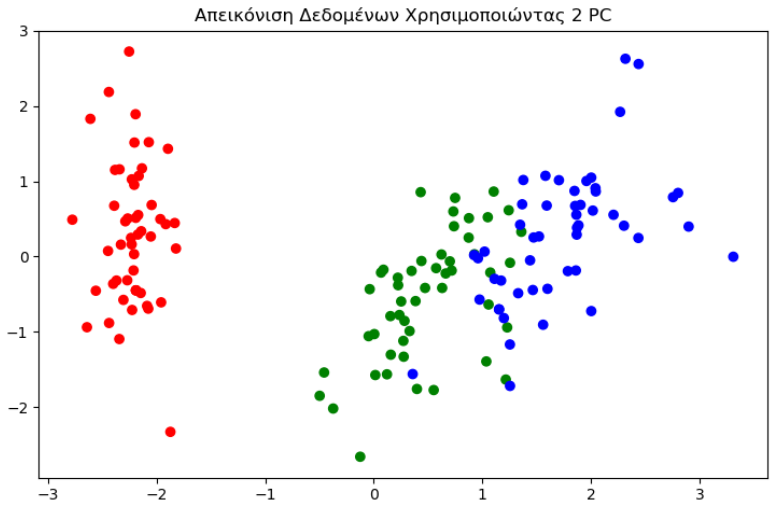
Εφαρμογή αλγορίθμου PCA

```
pca.fit(scaled_df)
```

```
PCA(copy=True, iterated_power='auto', n_components=2, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)
```

```
princ_comp = pca.transform(scaled_df)
```

```
np.set_printoptions(suppress=True)
```



```
pca.get_covariance()
```

```
array([[ 0.9779242 , -0.10104477,  0.87069468,  0.86134879],
       [-0.10104477,  1.00395722, -0.41916911, -0.37286994],
       [ 0.87069468, -0.41916911,  1.04639367,  0.93676197],
       [ 0.86134879, -0.37286994,  0.93676197,  0.99857055]])
```

```
pca.get_precision()
```

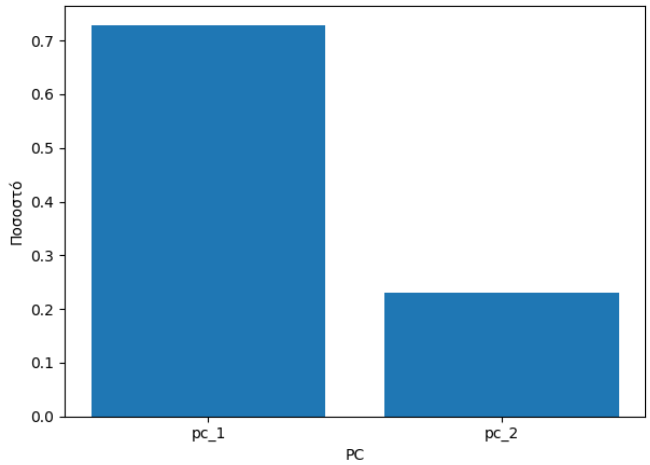
```
array([[ 7.20353574, -2.12418013, -3.57219978, -3.65572451],
       [-2.12418013,  1.82253795,  1.54848326,  1.06018486],
       [-3.57219978,  1.54848326,  7.94244356, -3.79130572],
       [-3.65572451,  1.06018486, -3.79130572,  8.10730497]])
```

Ποσοστό του κάθε PC

```
pc_var = pca.explained_variance_ratio_
```

```
labels = ['pc_' + str(x) for x in range(1, len(pc_var)+1)]
```

```
plt.bar(x=range(1,len(pc_var)+1), height=pc_var, tick_label=labels)
plt.ylabel('Ποσοστό')
plt.xlabel('PC')
plt.title('');
plt.show()
```




```
sum(pc_var)
# Συνολικό ποσοστό
```

```
0.9580097536148199
```

5.7 Κώδικας Estimator API για γραμμική παλινδρόμηση

```
import pandas as pd
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib notebook
import sys
sys.path.append('./python')
from generic_feature_statistics_generator import GenericFeatureStatisticsGenerator
import base64
from sklearn.metrics import mean_squared_error
from collections import Counter
from IPython.core.display import display, HTML
import time
```

```
df = pd.read_excel("Folds5x2_pp.xlsx")
```

```
df.head()
```

	AT	V	AP	RH	PE
0	14.96	41.76	1024.07	73.17	463.26
1	25.18	62.96	1020.04	59.08	444.37
2	5.11	39.40	1012.16	92.14	488.56
3	20.86	57.32	1010.24	76.64	446.48
4	10.82	37.50	1009.23	96.62	473.90

```
x_data= df.drop('PE', axis=1).copy()
```

```
x_data.head()
```

	AT	V	AP	RH
0	14.96	41.76	1024.07	73.17
1	25.18	62.96	1020.04	59.08
2	5.11	39.40	1012.16	92.14
3	20.86	57.32	1010.24	76.64
4	10.82	37.50	1009.23	96.62

```
target = df['PE'].copy()
```

Έλεγχος δεδομένων για κενά, συχνότητα εμφάνισης κ.λπ. χρησιμοποιώντας το «facets: https://github.com/PAIR-code/facets»

```
X_train_temp, X_test_temp, y_train_temp, y_test_temp = train_test_split(x_data,target,test_size=0.3)
```

```
gfsg = GenericFeatureStatisticsGenerator()
proto = gfsg.ProtoFromDataFrames([{'name': 'train', 'table': X_train_temp},
                                  {'name': 'test', 'table': X_test_temp}])
protostr = base64.b64encode(proto.SerializeToString()).decode("utf-8")
```

```
HTML_TEMPLATE = """<link rel="import"
          href="https://raw.githubusercontent.com/PAIR-code/facets/master/facets-dist/facets-jupyter.html" >
        <facets-overview id="elem"></facets-overview>
        <script>
          document.querySelector("#elem").protoInput = "{protostr}";
        </script>"""
html = HTML_TEMPLATE.format(protostr=protostr)
display(HTML(html))
```

	AT	V	AP	RH
4661	0.789275	1.579995	0.378377	0.514742
5908	-0.082696	-0.507311	-1.205371	0.147173
9146	0.912882	1.226335	-1.786191	0.690955
249	-1.701686	-1.432814	1.440401	0.746021
4080	-1.798423	-1.020867	-1.825025	1.055082

Δημιουργία των 'Feature Columns'

```
at = tf.feature_column.numeric_column('AT')
v = tf.feature_column.numeric_column('V')
ap = tf.feature_column.numeric_column('AP')
rh = tf.feature_column.numeric_column('RH')
```

```
feat_cols = [at,v,ap,rh]
```

```
feat_cols
```

```
[_NumericColumn(key='AT', shape=(1,), default_value=None, dtype=tf.float32, normalizer_fn=None),
 _NumericColumn(key='V', shape=(1,), default_value=None, dtype=tf.float32, normalizer_fn=None),
 _NumericColumn(key='AP', shape=(1,), default_value=None, dtype=tf.float32, normalizer_fn=None),
 _NumericColumn(key='RH', shape=(1,), default_value=None, dtype=tf.float32, normalizer_fn=None)]
```

```
batch_size = 32
learning_rate = 0.001
epochs_num = 10
```

Ορισμός της συνάρτησης εκπαίδευσης 'Input Training Function'

```
train_func = tf.estimator.inputs.pandas_input_fn(x=X_train, y=y_train,
                                                 batch_size=batch_size,num_epochs=epochs_num,
                                                 shuffle=True)
```

Χωρισμός συνόλου δεδομένων 70/30

```
X_train, X_test, y_train, y_test = train_test_split(x_data,target,test_size=0.3)
```

Κανονικοποίηση μεσω StandarScaler()

```
scaler = StandardScaler()
```

```
scaler.fit(X_train)
```

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

```
X_train = pd.DataFrame(scaler.transform(X_train),
                               columns=X_train.columns,
                               index=X_train.index)
```

```
X_test = pd.DataFrame(scaler.transform(X_test),
                              columns=X_test.columns,
                              index=X_test.index)
```

```
X_train.head()
```

	AT	V	AP	RH
5400	-0.163310	-0.087487	0.003546	0.668240
8509	1.095606	1.056987	-0.982498	-0.967235
9016	-0.300353	-0.570324	0.011988	1.697295
6191	0.910195	0.441035	-0.008273	0.010884
4892	0.555495	1.016029	-1.016267	1.151448

```
X_test.head()
```

Ορισμός της συνάρτησης πρόβλεψης 'Prediction Function'

```
predictions
```

```
[{'predictions': array([436.96036], dtype=float32)},
 {'predictions': array([456.68124], dtype=float32)},
 {'predictions': array([435.05585], dtype=float32)},
 {'predictions': array([483.0683], dtype=float32)},
 {'predictions': array([481.08517], dtype=float32)},
 {'predictions': array([436.00824], dtype=float32)},
 {'predictions': array([462.56784], dtype=float32)},
 {'predictions': array([427.82587], dtype=float32)},
 {'predictions': array([429.13223], dtype=float32)},
 {'predictions': array([450.43292], dtype=float32)},
 {'predictions': array([435.04648], dtype=float32)},
 {'predictions': array([439.99136], dtype=float32)},
 {'predictions': array([475.1091], dtype=float32)},
 {'predictions': array([473.7741], dtype=float32)},
 {'predictions': array([470.41403], dtype=float32)},
 {'predictions': array([435.3899], dtype=float32)},
 {'predictions': array([466.25296], dtype=float32)},
 {'predictions': array([455.5597], dtype=float32)},
 {'predictions': array([439.65237], dtype=float32)},
```

```
y_test.head(19)
```

```
4661    435.12
5908    454.35
9146    428.60
249     486.37
4080    485.08
6545    430.89
9045    462.12
5663    426.07
8165    430.11
5455    447.64
2393    431.93
7760    446.50
4579    471.56
7441    469.12
6517    474.79
7435    440.93
7359    465.00
156     455.86
4324    434.57
Name: PE, dtype: float64
```

```
final_preds = []
for pred in predictions:
    final_preds.append(pred['predictions'])
```

Σφάλμα

```
mean_squared_error(y_test,final_preds)**0.5
```

```
4.503233821436102
```

Προβλέψεις

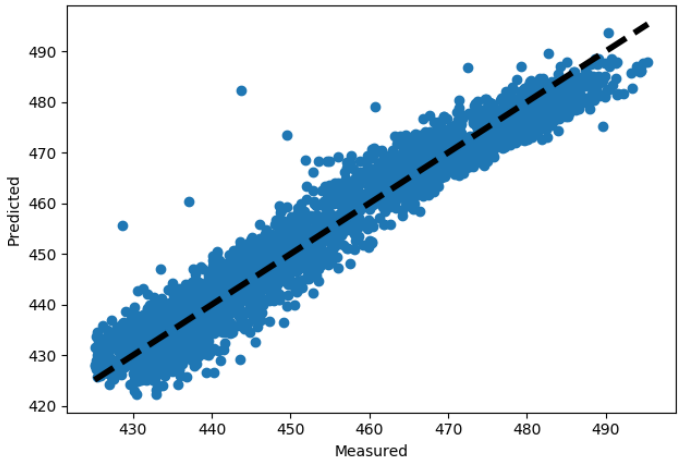
```
pred_gen = model.predict(predict_input_func)
```

```
predictions = list(pred_gen)
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from model/estimator_1549035078\model.ckpt-500
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
```

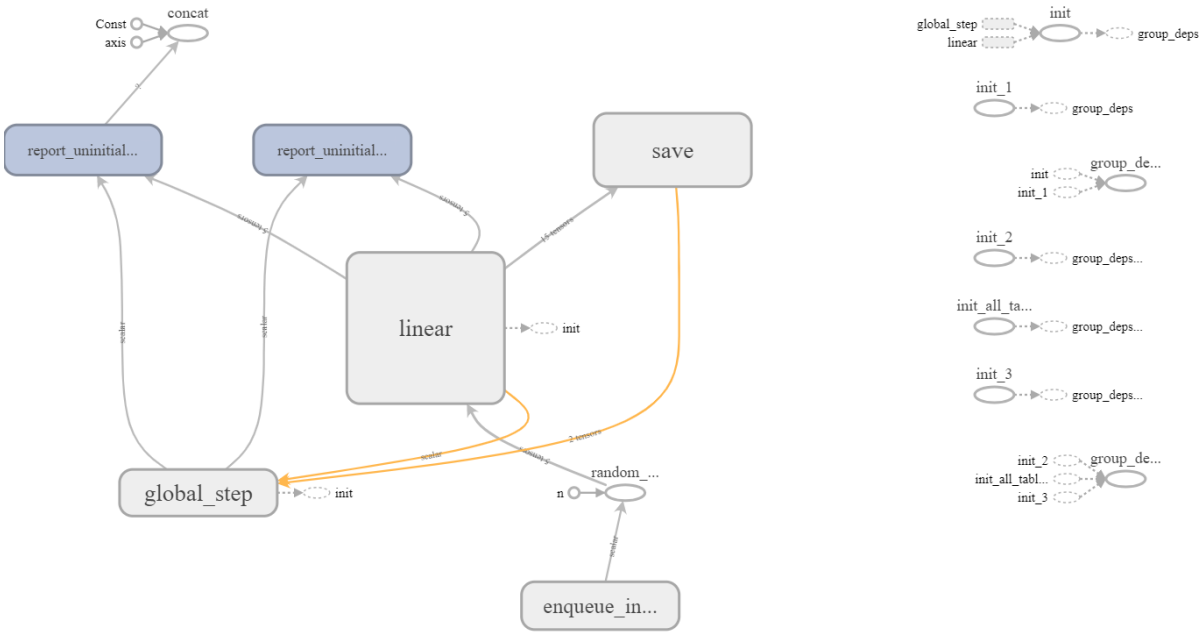
Απεικόνιση του γραμμικού μοντέλου

```
fig, ax = plt.subplots()
ax.scatter(y_test, final_preds)
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.show()
# https://scikit-learn.org/0.18/auto_examples/plot_cv_predict.html
```



Εκτέλεση του TensorBoard στον φάκελο όπου αποθηκεύτηκε το μοντέλο:

Υπολογιστικό γράφημα:



5.8 Κώδικας χρήσης του *Keras* για μοντέλο ταξινόμησης.

```
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras import optimizers
from sklearn.metrics import confusion_matrix
import tensorflow as tf
import keras
from tensorflow.keras.callbacks import TensorBoard
from sklearn.utils import class_weight
from collections import Counter
%matplotlib notebook
import base64
import sys
sys.path.append('./python')
from generic_feature_statistics_generator import GenericFeatureStatisticsGenerator
from IPython.core.display import display, HTML
import time
```

```
Using TensorFlow backend.
```

Εισαγωγή δεδομένων

```
df = pd.read_csv('bank-additional-full.csv', delimiter=';')
```

```
df = df.drop('duration', axis=1)
```

```
target = df['y'].replace(['yes', 'no'], (1,0))
```

```
df_overview = df.copy()
```

```
df_overview['y']= target
```

```
df_overview.head()
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	pdays	previous	poutcome	emp.var.rate	cons.pr
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	1	999	0	nonexistent	1.1	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	1	999	0	nonexistent	1.1	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	1	999	0	nonexistent	1.1	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	1	999	0	nonexistent	1.1	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	1	999	0	nonexistent	1.1	

Έλεγχος δεδομένων για κενά, συχνότητα εμφάνισης κ.λπ. χρησιμοποιώντας το «facets: https://github.com/PAIR-code/facets»

```
X_train_temp, X_test_temp, y_train_temp, y_test = train_test_split(df_overview, target, test_size=0.5)
```

```
gfsg = GenericFeatureStatisticsGenerator()
proto = gfsg.ProtoFromDataFrames([{'name': 'train', 'table': X_train_temp},
                                  {'name': 'test', 'table': X_test_temp}])
protostr = base64.b64encode(proto.SerializeToString()).decode("utf-8")
```

```
HTML_TEMPLATE = """<link rel="import"
                href="https://raw.githubusercontent.com/PAIR-code/facets/master/facets-dist/facets-jupyter.html" >
        <facets-overview id="elem"></facets-overview>
        <script>
          document.querySelector("#elem").protoInput = "{protostr}";
        </script>"""
html = HTML_TEMPLATE.format(protostr=protostr)
display(HTML(html))
```

ΠΑΡΑΤΗΡΗΣΕΙΣ:

Στήλη: 'previous' έχει ~86% μηδενικές τιμές

Στήλη: 'pdays' Συχνότητα τιμής :'999' 39673

Στήλη: 'default' { 'no': 32588 , 'unknown': 8597 'yes': 3 }

Στήλη: 'poutcome' Συχνότητα 'nonexistent': 35563

Στήλη: 'y' Target/Label Είναι ασύμμετρη με συχνότητα (0: 36548, 1: 4640) . Η ακρίβεια δεν πρέπει να χρησιμοποιηθεί για την αξιολόγηση του μοντέλου. F1 , recall πρέπει να ελεγχθούν

```
print('Class imbalance, Positives (1):',(100*(target.value_counts()[1] / (sum(Counter(target).values())))),'Percent')
```

```
Class imbalance, Positives (1): 11.265417111780131 Percent
```

```
Counter(target)
```

```
Counter({0: 36548, 1: 4640})
```

```
Counter(df['pdays'])
```

```
Counter({999: 39673,
         6: 412,
         4: 118,
         3: 439,
         5: 46,
         1: 26,
         0: 15,
         10: 52,
         7: 60,
         8: 18,
         9: 64,
         11: 28,
         2: 61,
         12: 58,
         13: 36,
         14: 20,
         15: 24,
         16: 11,
         21: 2,
```

```
Counter(df['poutcome'])
```

```
Counter({'nonexistent': 35563, 'failure': 4252, 'success': 1373})
```

```
Counter(df['default'])
```

```
Counter({'no': 32588, 'unknown': 8597, 'yes': 3})
```

Με βάση τα παραπάνω:

```
x_data = df.drop(['y','default', 'previous','pdays','poutcome'], axis=1)
```

```
target = df['y'].replace(['yes', 'no'], (1,0))
```

Μετατροπή κατηγορικών μεταβλητών μέσω get_dummies()

```
# Save column names if they are an object.
cat_cols = []
for c in x_data.columns:
    if x_data[c].dtype == 'object':
        cat_cols.append(c)
```

```
cat_cols
```

```
['job',
 'marital',
 'education',
 'housing',
 'loan',
 'contact',
 'month',
 'day_of_week']
```

```
for col in cat_cols: #
    x_data = pd.get_dummies(x_data, columns=[col], drop_first=True)
```

```
x_data.head()
```

	age	campaign	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	job_blue-collar	job_entrepreneur	job_housemaid	...	month_jun	month_mar
0	56	1	1.1	93.994	-36.4	4.857	5191.0	0	0	1	...	0	0
1	57	1	1.1	93.994	-36.4	4.857	5191.0	0	0	0	...	0	0
2	37	1	1.1	93.994	-36.4	4.857	5191.0	0	0	0	...	0	0
3	40	1	1.1	93.994	-36.4	4.857	5191.0	0	0	0	...	0	0
4	56	1	1.1	93.994	-36.4	4.857	5191.0	0	0	0	...	0	0

5 rows × 46 columns

Χωρισμός συνόλου δεδομένων 50/50

```
X_train, X_test, y_train, y_test = train_test_split(x_data, target, test_size=0.5)
```

Κανονικοποίηση μεσω StandarScaler()

```
scaler = StandardScaler()
```

```
scaler.fit(X_train)
```

```
c:\program files (x86)\microsoft visual studio\shared\python36_64\lib\site-packages\sklearn\preprocessing\data.py:625: DataConversionWarning: Data with input dtype uint8, int64, float64 were all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
```

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

```
X_train = pd.DataFrame(scaler.transform(X_train),
                               columns=X_train.columns,
                               index=X_train.index)
```

```
c:\program files (x86)\microsoft visual studio\shared\python36_64\lib\site-packages\ipykernel_launcher.py:1: DataConversionWarning: Data with input dtype uint8, int64, float64 were all converted to float64 by StandardScaler.
  """Entry point for launching an IPython kernel.
```

```
X_train.head()
```

	age	campaign	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	job_blue-collar	job_entrepreneur	job_housemaid	...	month_jun	n
7591	-0.290596	-0.204672	0.644636	0.717202	0.870195	0.713409	0.329873	-0.536516	-0.195348	-0.16174	...	-0.389106	
34911	-0.769046	0.149501	-1.194173	-1.174920	-1.239625	-1.371713	-0.945589	-0.536516	-0.195348	-0.16174	...	-0.389106	
20668	0.187853	-0.558846	0.834857	-0.228000	0.934781	0.771681	0.844776	-0.536516	-0.195348	-0.16174	...	-0.389106	
37038	-0.673356	-0.558846	-1.891652	-1.903585	1.473000	-1.490566	-1.263413	-0.536516	-0.195348	-0.16174	...	-0.389106	
19267	-0.386286	-0.558846	0.834857	-0.228000	0.934781	0.772835	0.844776	-0.536516	-0.195348	-0.16174	...	-0.389106	

5 rows × 46 columns

```
X_test = pd.DataFrame(scaler.transform(X_test),
                               columns=X_test.columns,
                               index=X_test.index)
```

```
c:\program files (x86)\microsoft visual studio\shared\python36_64\lib\site-packages\ipykernel_launcher.py:1: DataConversionWarning: Data with input dtype uint8, int64, float64 were all converted to float64 by StandardScaler.
  """Entry point for launching an IPython kernel.
```

```
X_test.head()
```

	age	campaign	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	job_blue-collar	job_entrepreneur	job_housemaid	...	month_jun	n
28996	0.570613	-0.558846	-1.194173	-0.862144	-1.433384	-1.282284	-0.945589	-0.536516	-0.195348	-0.161740	...	-0.389106	
32422	-0.769046	-0.204672	-1.194173	-1.174920	-1.239625	-1.335364	-0.945589	-0.536516	-0.195348	-0.161740	...	-0.389106	
30974	-0.769046	2.274542	-1.194173	-1.174920	-1.239625	-1.317479	-0.945589	-0.536516	-0.195348	-0.161740	...	-0.389106	
574	0.474923	-0.204672	0.644636	0.717202	0.870195	0.709370	0.329873	-0.536516	-0.195348	6.182772	...	-0.389106	
5919	-0.003527	-0.204672	0.644636	0.717202	0.870195	0.709370	0.329873	1.863877	-0.195348	-0.161740	...	-0.389106	

5 rows × 46 columns

Sequential Model

```
model = Sequential()
model.add(Dense(46, input_dim=46, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
```

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 46)                2162
_________________________________________________________________
dropout_1 (Dropout)          (None, 46)                0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                1504
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_3 (Dense)              (None, 32)                1056
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 33
=================================================================
Total params: 4,755
Trainable params: 4,755
```

Ορισμός μοντέλου

```
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.adam(lr=0.01),
              metrics=['accuracy'])
```

Ορισμοί παραμέτρων εκπαίδευσης

```
keras.backend.get_session().run(tf.global_variables_initializer())
```

```
NAME = 'test-{}'.format(int(time.time()))
tensorboard = TensorBoard(log_dir='.logs/10_3232_{}'.format(NAME))
model.fit(X_train, y_train,
          shuffle=True,
          class_weight=wght,
          validation_split=0.3,
          epochs=10,
          batch_size=128,
          verbose=2,
          callbacks=[tensorboard])
```

```
Epoch 2/10
 - 0s - loss: 0.5552 - acc: 0.8180 - val_loss: 0.5247 - val_acc: 0.8411
Epoch 3/10
 - 0s - loss: 0.5565 - acc: 0.8181 - val_loss: 0.5399 - val_acc: 0.8388
Epoch 4/10
 - 0s - loss: 0.5553 - acc: 0.8300 - val_loss: 0.5330 - val_acc: 0.8395
Epoch 5/10
 - 0s - loss: 0.5561 - acc: 0.8148 - val_loss: 0.5294 - val_acc: 0.8217
Epoch 6/10
 - 0s - loss: 0.5500 - acc: 0.8148 - val_loss: 0.5437 - val_acc: 0.8162
Epoch 7/10
 - 0s - loss: 0.5564 - acc: 0.8231 - val_loss: 0.5354 - val_acc: 0.8501
Epoch 8/10
 - 0s - loss: 0.5493 - acc: 0.8238 - val_loss: 0.5389 - val_acc: 0.8283
Epoch 9/10
 - 0s - loss: 0.5487 - acc: 0.8251 - val_loss: 0.5339 - val_acc: 0.8335
Epoch 10/10
 - 0s - loss: 0.5504 - acc: 0.8239 - val_loss: 0.5304 - val_acc: 0.8391

<keras.callbacks.History at 0x27cab160dd8>
```

Αξιολογιση μοντελου

```
score = model.evaluate(X_test, y_test, batch_size=128)
```

```
20594/20594 [==============================] - 0s 6us/step
```

```
print('Test accuracy:', score[1])
```

```
Test accuracy: 0.8367000096247383
```

Προβλέψεις

```
final_preds = pd.DataFrame(model.predict(X_test), index=y_test.index)
# Για προβλέψεις χωρίς ορισμό κατωφλίου:
#         final_preds = pd.DataFrame(model.predict_classes(X_test), index=y_test.index)
```

```
final_preds[final_preds<0.4]=0
```

```
final_preds[final_preds>=0.4]=1
```

```
final_preds=final_preds.astype('int64')
```

```
cm = confusion_matrix(y_true=y_test, y_pred=final_preds)
```

```
cm
```

```
array([[15090,  3160],
       [  843,  1501]], dtype=int64)
```

```
print(classification_report(y_test, final_preds))
```

```
              precision    recall  f1-score   support

           0       0.95      0.83      0.88     18250
           1       0.32      0.64      0.43      2344

   micro avg       0.81      0.81      0.81     20594
   macro avg       0.63      0.73      0.66     20594
weighted avg       0.88      0.81      0.83     20594
```

5.9 Κώδικας πρόβλεψη θερμοκρασίας.

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib notebook
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, LSTM, SimpleRNN, Dropout, Flatten, Bidirectional,Activation
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error
from IPython.display import Image
```

Θα χρησιμοποιηθούν τα ίδια δεδομένα από το Κεφάλαιο 1: 1.3.2.

```
df = pd.read_csv("history_export_2018-12-04T18_30_11.csv",skiprows=12, header=None,
                 delimiter=';', names=['year',
                                       'month',
                                       'day',
                                       'hour',
                                       'minute',
                                       'temp',
                                       'shortwave_rad'])
```

```
df['Datetime']=pd.to_datetime(df[['year','month','day','hour']])
df.head()
```

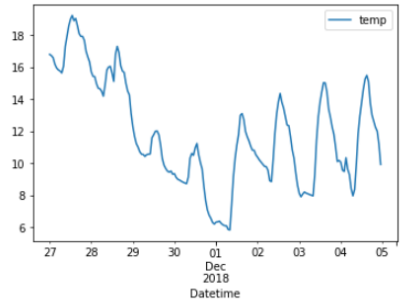
	year	month	day	hour	minute	temp	shortwave_rad	Datetime
0	2018	11	27	0	0	16.79	0.0	2018-11-27 00:00:00
1	2018	11	27	1	0	16.71	0.0	2018-11-27 01:00:00
2	2018	11	27	2	0	16.58	0.0	2018-11-27 02:00:00
3	2018	11	27	3	0	16.18	0.0	2018-11-27 03:00:00
4	2018	11	27	4	0	15.94	0.0	2018-11-27 04:00:00

```
df.drop(columns=['year', 'month', 'day', 'hour', 'minute','shortwave_rad'], axis=1, inplace=True)
```

```
df.set_index('Datetime', inplace=True)
```

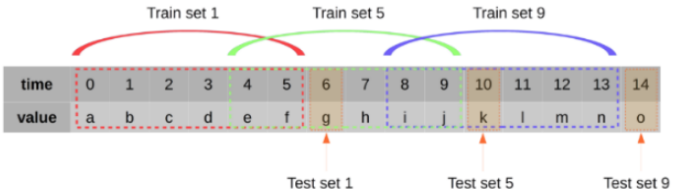
```
df.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fd64ce8e80>
```




```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 192 entries, 2018-11-27 00:00:00 to 2018-12-04 23:00:00
Data columns (total 1 columns):
temp    192 non-null float64
dtypes: float64(1)
memory usage: 3.0 KB
```



Window	Time steps in train set	Time steps in test set
1	0, 1, 2, 3, 4, 5	6
2	1, 2, 3, 4, 5, 6	7
3	2, 3, 4, 5, 6, 7	8
...	...	...
...	...	...
7	6, 7, 8, 9, 10, 11	12
8	7, 8, 9, 10, 11, 12	13
9	8, 9, 10, 11, 12, 13	14

Parameter	Value
Window size	6
horizon	1
Fixed window	True

```
df.reset_index(drop=True, inplace=True)
```

```
df.shape[0]
```

```
192
```

```
df.isnull().sum()
```

```
temp    0
dtype: int64
```

Παράδειγμα λογικής των "Sliding Windows"

Image link: https://i.stack.imgur.com/rfSAF.png

```
Image("pic_rnn/rfSAF.png")
```

Εφαρμογή παράθυρων με βήμα 5

```
timestep = 5
x_test = []
y_test = []


for i in range(timestep,x_test_scaled.shape[0]):
    x_test.append(x_test_scaled[i-timestep:i,0])
    y_test.append(x_test_scaled[i,0])

x_test,y_test=np.array(x_test),np.array(y_test)
x_test=x_test.reshape(x_test.shape[0],x_test.shape[1],1)
print("x_test shape= ",x_test.shape)
print("y_test shape= ",y_test.shape)
```

```
x_test shape=  (19, 5, 1)
y_test shape=  (19,)
```

```
y_test=np.array(y_test)
y_test=y_test.reshape(len(y_test),1)
```

```
y_test=scaler.inverse_transform(y_test)
```

```
timestep = 5
x_train = []
y_train = []


for i in range(timestep,train_scaled.shape[0]):
    x_train.append(train_scaled[i-timestep:i,0])
    y_train.append(train_scaled[i,0])

x_train,y_train=np.array(x_train),np.array(y_train)
x_train=x_train.reshape(x_train.shape[0],x_train.shape[1],1)
print("x_train shape= ",x_train.shape)
print("y_train shape= ",y_train.shape)
```

```
x_train shape=  (163, 5, 1)
y_train shape=  (163,)
```

Ορισμός δεδομένων για εκπαίδευση και πρόβλεψη

Για την εκπαίδευση θα κρατηθούν όλα τα δεδομένα εκτός από τις τελευταίες 24ωρες

Για την πρόβλεψη/εκτίμηση θα χρησιμοποιηθούν οι τελευταίες 24 ώρες

```
train=df.iloc[:len(df)-24]
test=df.iloc[len(train):]
```

```
train=np.array(train)
train=train.reshape(train.shape[0],1)
```

```
x_test=df[len(df)-len(test):]
x_test=x_test.values.reshape(-1,1)
```

Κανονικοποίηση δεδομένων μέσω "MinMaxScaler"

```
scaler=MinMaxScaler(feature_range=(0,1))
scaler.fit(train)
```

```
MinMaxScaler(copy=True, feature_range=(0, 1))
```

```
train_scaled=scaler.transform(train)
```

```
x_test_scaled=scaler.transform(x_test)
```

Όλες οι τιμές τις ακολουθίας κανονικοποιημένες

```
df_scaled = scaler.transform(df)
```

```
df_scaled
```

```
array([[0.81852128],
       [0.81254668],
       [0.80283794],
       [0.7729649 ],
       [0.75504108],
       [0.74607916],
       [0.74085138],
       [0.73188947],
       [0.76400299],
       [0.85586258],
       [0.90440627],
       [0.95070948],
       [0.98207618],
       [1.        ],
       [0.97535474],
       [0.98581031],
       [0.95220314],
       [0.91486184],
       [0.9021658 ],
       [0.9021658 ],
```

Τα μοντέλα rnn/lstm παίρνουν ως είσοδο (Μέγεθος ακολουθίας, Χαρακτηριστικά)

Παρακάτω θα χρησιμοποιηθούν 3 διαφορετικά μοντέλα

```
array([0.74607916, 0.74085138, 0.73188947, 0.76400299, 0.85586258,
       0.90440627, 0.95070948, 0.98207618, 1.        , 0.97535474,
       0.98581031, 0.95220314, 0.91486184, 0.9021658 , 0.9021658 ,
       0.88349515, 0.83271098, 0.80657207, 0.78342046, 0.73711725,
       0.71620612, 0.71471247, 0.68185213, 0.660941  , 0.6579537 ,
       0.64600448, 0.6235997 , 0.67737117, 0.74533234, 0.75952203,
       0.76250934, 0.73188947, 0.69230769, 0.81926811, 0.85586258,
       0.82673637, 0.76549664, 0.74085138, 0.73412995, 0.68409261,
       0.64749813, 0.62957431, 0.54294249, 0.48170276, 0.43614638,
       0.40253921, 0.38610904, 0.36445108, 0.35324869, 0.35175504,
       0.34279313, 0.35100822, 0.35324869, 0.35399552, 0.42942494,
       0.44361464, 0.46004481, 0.46153846, 0.44361464, 0.39581777,
       0.33233757, 0.3017177 , 0.28603435, 0.27408514, 0.26960418,
       0.27483196, 0.25989544, 0.26213592, 0.24495892, 0.23599701,
       0.23226288, 0.2270351 , 0.22255414, 0.21882001, 0.21583271,
```

Πρώτο μοντέλο χρησιμοποιώντας SimpleRNN

```
model_srnn=Sequential()

model_srnn.add(SimpleRNN(128,activation="relu",return_sequences=True,input_shape=(x_train.shape[1],1)))
model_srnn.add(Dropout(0.25))
model_srnn.add(SimpleRNN(256,activation="relu",return_sequences=True))
model_srnn.add(Dropout(0.25))

model_srnn.add(Flatten())
model_srnn.add(Dense(1, activation='linear'))

model_srnn.compile(loss="mean_squared_error",optimizer="adam")
```

```
model_srnn.fit(x_train,y_train,epochs=10,batch_size=36)
```

```
Epoch 1/10
163/163 [==============================] - 1s 3ms/step - loss: 0.1185
Epoch 2/10
163/163 [==============================] - 0s 367us/step - loss: 0.0285
Epoch 3/10
163/163 [==============================] - 0s 306us/step - loss: 0.0258
Epoch 4/10
163/163 [==============================] - 0s 392us/step - loss: 0.0172
Epoch 5/10
163/163 [==============================] - 0s 337us/step - loss: 0.0168
Epoch 6/10
163/163 [==============================] - 0s 337us/step - loss: 0.0151
Epoch 7/10
163/163 [==============================] - 0s 300us/step - loss: 0.0141
Epoch 8/10
163/163 [==============================] - 0s 337us/step - loss: 0.0116
Epoch 9/10
163/163 [==============================] - 0s 392us/step - loss: 0.0113
Epoch 10/10
163/163 [==============================] - 0s 337us/step - loss: 0.0113

<keras.callbacks.History at 0x1fd6a05f0b8>
```

Δεύτερο μοντέλο χρησιμοποιώντας LSTM

```
model_lstm=Sequential()

model_lstm.add(LSTM(128,input_shape=(x_train.shape[1],1),activation="relu",return_sequences=True))
model_lstm.add(Dropout(0.25))
model_lstm.add(LSTM(256,activation="relu",return_sequences=False))
model_lstm.add(Dropout(0.25))
model_lstm.add(Dense(1, activation='linear'))

model_lstm.compile(loss="mean_squared_error",optimizer="adam")
```

```
model_lstm.fit(x_train,y_train,epochs=10,batch_size=32)
```

```
Epoch 1/10
163/163 [==============================] - 2s 10ms/step - loss: 0.2276
Epoch 2/10
163/163 [==============================] - 0s 997us/step - loss: 0.1073
Epoch 3/10
163/163 [==============================] - 0s 1ms/step - loss: 0.0379
Epoch 4/10
163/163 [==============================] - 0s 1ms/step - loss: 0.0279
Epoch 5/10
163/163 [==============================] - 0s 1ms/step - loss: 0.0295
Epoch 6/10
163/163 [==============================] - 0s 1ms/step - loss: 0.0256
Epoch 7/10
163/163 [==============================] - 0s 1000us/step - loss: 0.0198
Epoch 8/10
163/163 [==============================] - 0s 1ms/step - loss: 0.0210
Epoch 9/10
163/163 [==============================] - 0s 1ms/step - loss: 0.0159
Epoch 10/10
163/163 [==============================] - 0s 1ms/step - loss: 0.0171

<keras.callbacks.History at 0x1fd6a278908>
```

```
predict_srnn=model_srnn.predict(x_test)
predict_srnn=scaler.inverse_transform(predict_srnn)
```

```
plt.figure(figsize=(8,4), dpi=80, facecolor='w', edgecolor='k')
plt.plot(y_test,color="r",label="Πραγματική θερμοκρασία")
plt.plot(predict_srnn,color="b",label="Πρόβλεψη μοντέλου")
plt.legend()
plt.xlabel("ΩΡΕΣ")
plt.ylabel("ΘΕΡΜΟΚΡΑΣΙΑ")
plt.title("SimpleRNN Model")
plt.grid(True)
```

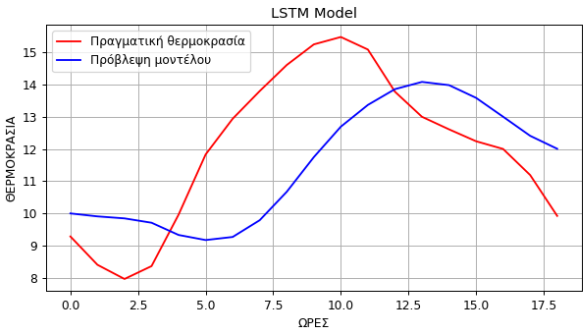
```
predict_lstm=model_lstm.predict(x_test)
predict_lstm=scaler.inverse_transform(predict_lstm)
```

```
mse_lstm = mean_absolute_error(y_true=y_test, y_pred=predict_lstm)
mse_lstm
```

```
1.9251190848099562
```

```
plt.figure(figsize=(8,4), dpi=80, facecolor='w', edgecolor='k')
plt.plot(y_test,color="r",label="Πραγματική θερμοκρασία")
plt.plot(predict_lstm,color="b",label="Πρόβλεψη μοντέλου")
plt.legend()
plt.xlabel("ΩΡΕΣ")
plt.ylabel("ΘΕΡΜΟΚΡΑΣΙΑ")
plt.title("LSTM Model")
plt.grid(True)
```

```
<IPython.core.display.Javascript object>
```



Τρίτο μοντέλο χρησιμοποιώντας Biderectional με SimpleRNN

```
model_bi = Sequential()

model_bi.add(Bidirectional(SimpleRNN(128, return_sequences=True), input_shape=(x_train.shape[1],1)))
model_bi.add(Dropout(0.25))
model_bi.add(Activation('relu'))
model_bi.add(Bidirectional(SimpleRNN(256, return_sequences=False)))
model_bi.add(Dropout(0.25))
model_bi.add(Activation('relu'))
model_bi.add(Dense(1, activation='linear'))

model_bi.compile(loss='mean_squared_error', optimizer='adam')
```

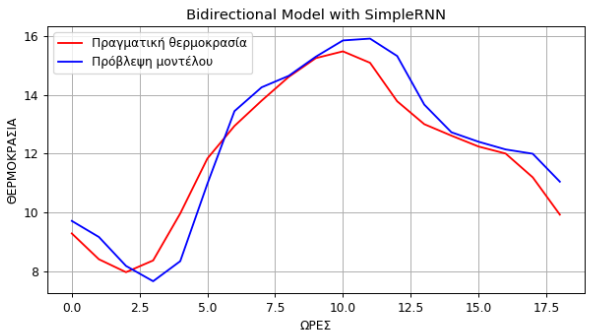
```
model_bi.fit(x_train,y_train,epochs=10,batch_size=32)
```

```
Epoch 1/10
163/163 [==============================] - 1s 7ms/step - loss: 0.2827
Epoch 2/10
163/163 [==============================] - 0s 563us/step - loss: 0.0216
Epoch 3/10
163/163 [==============================] - 0s 587us/step - loss: 0.0130
Epoch 4/10
163/163 [==============================] - 0s 581us/step - loss: 0.0053
Epoch 5/10
163/163 [==============================] - 0s 618us/step - loss: 0.0045
Epoch 6/10
163/163 [==============================] - 0s 575us/step - loss: 0.0048
Epoch 7/10
163/163 [==============================] - 0s 538us/step - loss: 0.0027
Epoch 8/10
163/163 [==============================] - 0s 542us/step - loss: 0.0023
Epoch 9/10
163/163 [==============================] - 0s 639us/step - loss: 0.0018
Epoch 10/10
163/163 [==============================] - 0s 624us/step - loss: 0.0015

<keras.callbacks.History at 0x1fd71f2eb00>
```

```
predict_bi=model_bi.predict(x_test)
predict_bi=scaler.inverse_transform(predict_bi)
```

```
plt.figure(figsize=(8,4), dpi=80, facecolor='w', edgecolor='k')
plt.plot(y_test,color="r",label="Πραγματική θερμοκρασία")
plt.plot(predict_bi,color="b",label="Πρόβλεψη μοντέλου")
plt.legend()
plt.xlabel("ΩΡΕΣ")
plt.ylabel("ΘΕΡΜΟΚΡΑΣΙΑ")
plt.title("Bidirectional Model with SimpleRNN")
plt.grid(True)
```

```
<IPython.core.display.Javascript object>
```




```
mse_bi = mean_absolute_error(y_true=y_test, y_pred=predict_bi)
mse_bi
```

```
0.5989769935607911
```

5.10 Δεύτερο παράδειγμα χρονοσειράς, πρόβλεψη τιμών Bitcoin

Δεδομένα: https://finance.yahoo.com/quote/BTC-USD/history?p=BTC-USD

Απο-Εως Feb 04, 2018 - Feb 04, 2019

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib notebook
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error
from keras.models import Sequential
from keras.layers import Dense, SimpleRNN, Dropout, Flatten, Bidirectional, Activation
# from keras.preprocessing.sequence import pad_sequences
```

```
df = pd.read_excel("Bitcoin_usd.xlsx", index_col='Date')
```

```
df.head()
```

Date	Open	High	Low	Close	Adj_Close	Volume
2019-02-04	3450.03	3470.80	3429.21	3436.52	3436.52	45471364
2019-01-25	3598.52	3607.43	3539.41	3582.89	3582.89	116674680
2019-01-24	3572.05	3616.33	3546.50	3598.52	3598.52	113155246
2019-01-23	3602.04	3631.15	3543.96	3572.05	3572.05	135864247
2019-01-22	3571.92	3635.69	3473.77	3602.04	3602.04	191355241

```
df.sort_index(ascending=True, inplace=True)
```

```
df.head()
```

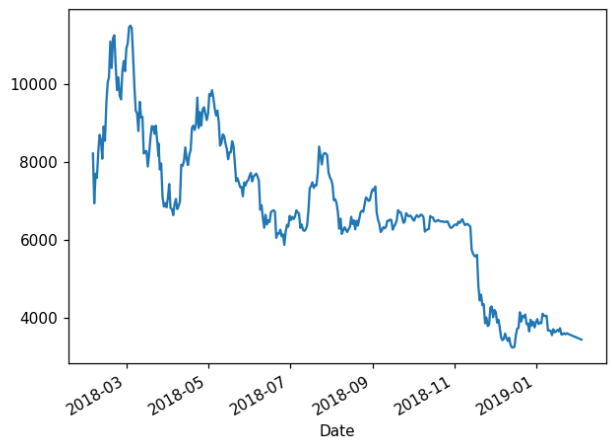
Date	Open	High	Low	Close	Adj_Close	Volume
2018-02-04	9251.27	9400.99	7889.83	8218.05	8218.05	1413207410
2018-02-05	8218.05	8391.29	6627.31	6937.08	6937.08	2534149181
2018-02-06	6936.43	7932.38	5968.36	7701.25	7701.25	3397596513
2018-02-07	7701.25	8572.68	7208.86	7592.72	7592.72	2159765331
2018-02-08	7593.78	8643.94	7590.48	8260.69	8260.69	1594673973

```
max(df.index)
```

```
Timestamp('2019-02-04 00:00:00')
```

```
min(df.index)
```

```
Timestamp('2018-02-04 00:00:00')
```



```
df.isnull().sum()
```

```
Open          0
High          0
Low           0
Close         0
Adj_Close     0
Volume        0
dtype: int64
```

```
data = df.Close.copy()
```

```
data.head()
```

```
Date
2018-02-04    8218.05
2018-02-05    6937.08
2018-02-06    7701.25
2018-02-07    7592.72
2018-02-08    8260.69
Name: Close, dtype: float64
```

```
data.reset_index(drop=True, inplace=True)
```

```
data.head()
```

```
0    8218.05
1    6937.08
2    7701.25
3    7592.72
4    8260.69
Name: Close, dtype: float64
```

```
data.isnull().sum()
```

```
0
```

Ορισμός δεδομένων για εκπαίδευση και πρόβλεψη

Για την εκπαίδευση θα κρατηθούν όλα τα δεδομένα εκτός από τις τελευταίες 30 μέρες

Για την πρόβλεψη/εκτίμηση θα χρησιμοποιηθούν οι τελευταίες 30 μέρες

```
train=data.iloc[:len(df)-30]
test=data.iloc[len(train):]
```

```
train=np.array(train)
train=train.reshape(train.shape[0],1)
```

```
x_test=data[len(data)-len(test):]
x_test=x_test.values.reshape(-1,1)
```

Κανονικοποίηση δεδομένων μέσω "MinMaxScaler"

```
scaler=MinMaxScaler(feature_range=(0,1))
scaler.fit(train)
```

```
MinMaxScaler(copy=True, feature_range=(0, 1))
```

```
train_scaled=scaler.transform(train)
```

```
x_test_scaled=scaler.transform(x_test)
```

Εφαρμογή παράθυρων με βήμα 5

```
timestep=5
x_test=[]
y_test=[]

for i in range(timestep,x_test_scaled.shape[0]):
    x_test.append(x_test_scaled[i-timestep:i,0])
    y_test.append(x_test_scaled[i,0])

x_test,y_test=np.array(x_test),np.array(y_test)
x_test=x_test.reshape(x_test.shape[0],x_test.shape[1],1) #reshaped for RNN
print("x_test shape= ",x_test.shape)
print("y_test shape= ",y_test.shape)
```

```
x_test shape=  (25, 5, 1)
y_test shape=  (25,)
```

```
y_test= np.array(y_test)
y_test=y_test.reshape(-1, 1)
y_true=scaler.inverse_transform(y_test)
```

```
x_train=[]
y_train=[]

for i in range(timestep,train_scaled.shape[0]):
    x_train.append(train_scaled[i-timestep:i,0])
    y_train.append(train_scaled[i,0])

x_train,y_train=np.array(x_train),np.array(y_train)
x_train=x_train.reshape(x_train.shape[0],x_train.shape[1],1)
print("x_train shape= ",x_train.shape)
print("y_train shape= ",y_train.shape)
```

```
x_train shape=  (322, 5, 1)
y_train shape=  (322,)
```

Ορισμος μοντελου

```
model = Sequential()
model.add(Bidirectional(SimpleRNN(128, return_sequences=True), input_shape=(x_train.shape[1],1)))
model.add(Activation('relu'))
model.add(Bidirectional(SimpleRNN(256, return_sequences=False)))
model.add(Activation('relu'))
model.add(Dense(1, activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam')
```

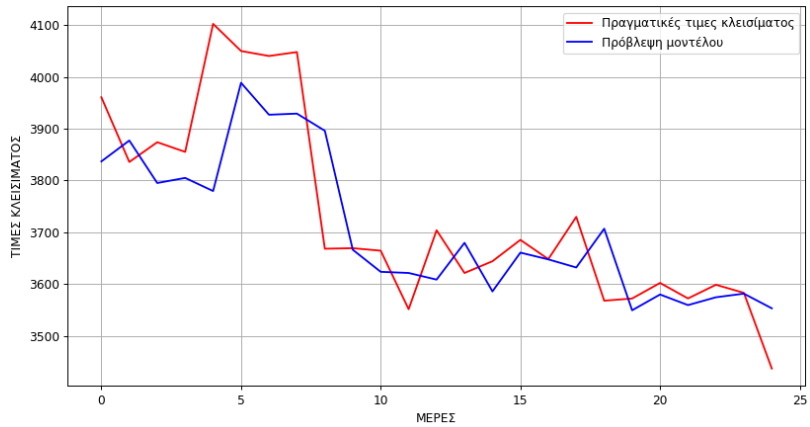
```
model.fit(x_train,y_train,epochs=10,batch_size=32)
```

```
Epoch 1/10
322/322 [==============================] - 1s 4ms/step - loss: 0.0978
Epoch 2/10
322/322 [==============================] - 0s 616us/step - loss: 0.0232
Epoch 3/10
322/322 [==============================] - 0s 629us/step - loss: 0.0083
Epoch 4/10
322/322 [==============================] - 0s 613us/step - loss: 0.0050
Epoch 5/10
322/322 [==============================] - 0s 618us/step - loss: 0.0021
Epoch 6/10
322/322 [==============================] - 0s 768us/step - loss: 0.0015
Epoch 7/10
322/322 [==============================] - 0s 723us/step - loss: 0.0013
Epoch 8/10
322/322 [==============================] - 0s 658us/step - loss: 0.0022
Epoch 9/10
322/322 [==============================] - 0s 689us/step - loss: 0.0013
Epoch 10/10
322/322 [==============================] - 0s 646us/step - loss: 0.0012

<keras.callbacks.History at 0x21ce209f438>
```

Προβλεψεις

```
predict_bi=model.predict(x_test)
predict_bi=scaler.inverse_transform(predict_bi)
```

```
plt.figure(figsize=(8,4), dpi=80, facecolor='w', edgecolor='k')
plt.plot(y_true,color="r",label="Πραγματικές τιμες κλεισίματος")
plt.plot(predict_bi,color="b",label="Πρόβλεψη μοντέλου")
plt.legend()
plt.xlabel("ΜΕΡΕΣ")
plt.ylabel("ΤΙΜΕΣ ΚΛΕΙΣΙΜΑΤΟΣ")
plt.grid(True)
```




```
mse = mean_absolute_error(y_pred=predict_bi, y_true=y_true)
mse
```

```
77.12474296875006
```

5.11 Κώδικας εφαρμογής CNN (Πολυταξικός ταξινομητής)

Δεδομένα: https://github.com/ardamavi/Sign-Language-Digits-Dataset

```
import os
import cv2
%matplotlib inline

import matplotlib.pyplot as plt
from numpy import random
import numpy as np

import keras
import tensorflow as tf
from keras import models
from tensorflow.keras.callbacks import TensorBoard

from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers import BatchNormalization, Flatten, Dropout, Dense
from keras.preprocessing.image import ImageDataGenerator
```

Τα δεδομένα βρίσκονται στον φάκελο "Dataset", όπου μέσα σε αυτόν τον φάκελο υπάρχουν φάκελοι που είναι χωρισμένα τα δεδομένα με όνομα την κατηγορία στην οποία ανήκουν.

Δηλαδή \Dataset\0 \Dataset\1 \Dataset\2 κ.ο.κ

```
data_path = ".\Dataset"
labels = ["0","1","2","3","4","5","6","7","8","9"]
```

Ρυθμίσεις για αποθήκευση των πληροφοριών σφάλματος και ακρίβειας του μοντέλου κατά την διάρκεια της εκπαίδευσης για το TensorBoard

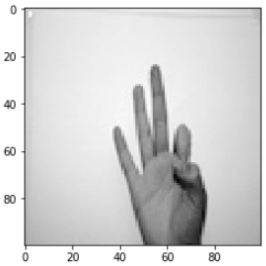
```
NAME = "sign_numbers_22"
```

```
tensorboard = TensorBoard(log_dir="logs/{}".format(NAME))
```

Τυχαία εμφάνιση μιας εικόνας

```
rand_img = random.choice(list(labels))
print("Class selection:",rand_img)
data_path1 = os.path.join(data_path, rand_img)

for img in os.listdir(data_path1):
    rand_img = random.choice(list(labels))
    image = cv2.imread(os.path.join(data_path1, img), cv2.IMREAD_GRAYSCALE)
    print("File from:",os.path.join(data_path1, img))
    print("Image Dims:", np.shape(image))
    plt.imshow(image, cmap="gray")
    break
```

```
Class selection: 9
File from: .\Dataset\9\IMG_1127.JPG
Image Dims: (100, 100)
```



Ορισμός του μοντέλου

```
multi_class_model = Sequential()

multi_class_model.add(Conv2D(36, (3, 3),input_shape = (64,64,1), activation='relu'))
multi_class_model.add(MaxPooling2D(pool_size = (2, 2)))

multi_class_model.add(Conv2D(24, (3, 3),activation='relu'))
multi_class_model.add(MaxPooling2D(pool_size = (4, 4)))

multi_class_model.add(Flatten())

multi_class_model.add(Dense(256, activation = 'relu'))
multi_class_model.add(Dropout(0.5))
multi_class_model.add(BatchNormalization())

multi_class_model.add(Dense(10, activation = 'softmax'))

multi_class_model.compile(optimizer=Adam(lr=0.001),loss = 'binary_crossentropy', metrics = ['accuracy'])
```

```
multi_class_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 62, 62, 36)        360
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 36)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 29, 29, 24)        7800
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 24)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1176)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               301312
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 313,066
Trainable params: 312,554
Non-trainable params: 512
_________________________________________________________________
```

Εκπαίδευση χρησιμοποιώντας το ImageDataGenerator απο Keras

Επεξεργασία δεδομένων:

Κανονικοποίηση δεδομένων "rescale = 1./255"

Αλλάζοντας την γωνία "rotation_range=15" , "shear_range=0.01"

Μετατόπιση "width_shift_range=0.1, height_shift_range=0.1"

Μεγέθυνση "zoom_range=[0.9, 1.25]"

Περιστροφή εικόνας κατά τον οριζόντιο άξονα "horizontal_flip=True"

Αλλαγή φωτεινότητας "brightness_range=[0.9, 1.5]"

Οι αλλαγές αυτές βοηθάνε στο να παραχθούν περισσότερα δεδομένα για την εκπαίδευση

```
train_datagen = ImageDataGenerator(rescale = 1./255,rotation_range=15,
                                   width_shift_range=0.1,
                                   height_shift_range=0.1,
                                   shear_range=0.01,
                                   zoom_range=[0.9, 1.25],
                                   horizontal_flip=True,
                                   vertical_flip=False,
                                   fill_mode='reflect',
                                   brightness_range=[0.9, 1.5])
```

Ορισμός φακέλου όπου βρίσκονται τα δεδομένα "directory=r"./Dataset" "

Το μέγεθος τις εικόνας για εκπαίδευση "target_size=(64, 64)"

Αλλαγή σε κλίμακα του γκρι "color_mode="grayscale" "

Το είδος της πρόβλεψης "class_mode="categorical" "

Και τέλος το μέγεθος των δεδομένων για ένα κάλεσμα του "batch_size=32"

```
train_generator = train_datagen.flow_from_directory(
    directory=r"./Dataset",
    target_size=(64, 64),
    color_mode="grayscale",
    batch_size=32,
    class_mode="categorical",
    shuffle=True,
    seed=42
)
```

```
Found 2062 images belonging to 10 classes.
```

Εκπαίδευση του μοντέλου

```
keras.backend.get_session().run(tf.global_variables_initializer())
```

```
multi_class_model.fit_generator(train_generator,
                    steps_per_epoch = 64,
                    epochs = 20, callbacks=[tensorboard])
```

```
Epoch 1/20
64/64 [==============================] - 48s 745ms/step - loss: 0.3258 - acc: 0.8996
Epoch 2/20
64/64 [==============================] - 10s 157ms/step - loss: 0.2713 - acc: 0.9045
Epoch 3/20
64/64 [==============================] - 10s 150ms/step - loss: 0.2380 - acc: 0.9121
Epoch 4/20
64/64 [==============================] - 10s 150ms/step - loss: 0.2176 - acc: 0.9185
Epoch 5/20
64/64 [==============================] - 10s 151ms/step - loss: 0.2001 - acc: 0.9244
Epoch 6/20
64/64 [==============================] - 10s 149ms/step - loss: 0.1903 - acc: 0.9259
Epoch 7/20
64/64 [==============================] - 10s 150ms/step - loss: 0.1780 - acc: 0.9313
Epoch 8/20
64/64 [==============================] - 10s 150ms/step - loss: 0.1731 - acc: 0.9331
Epoch 9/20
64/64 [==============================] - 10s 150ms/step - loss: 0.1621 - acc: 0.9374
Epoch 10/20
64/64 [==============================] - 10s 152ms/step - loss: 0.1562 - acc: 0.9397
Epoch 11/20
64/64 [==============================] - 10s 154ms/step - loss: 0.1516 - acc: 0.9416
Epoch 12/20
64/64 [==============================] - 9s 148ms/step - loss: 0.1472 - acc: 0.9430
Epoch 13/20
64/64 [==============================] - 10s 149ms/step - loss: 0.1462 - acc: 0.9410
Epoch 14/20
64/64 [==============================] - 9s 148ms/step - loss: 0.1382 - acc: 0.9473
Epoch 15/20
64/64 [==============================] - 10s 149ms/step - loss: 0.1338 - acc: 0.9493
Epoch 16/20
64/64 [==============================] - 10s 154ms/step - loss: 0.1258 - acc: 0.9491
Epoch 17/20
64/64 [==============================] - 10s 151ms/step - loss: 0.1249 - acc: 0.9511
Epoch 18/20
64/64 [==============================] - 10s 149ms/step - loss: 0.1210 - acc: 0.9533
```

Αποθήκευση του μοντέλου

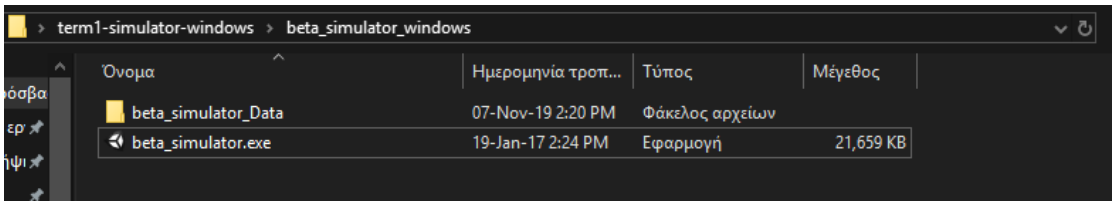
```
multi_class_model.save(r'./Model/model_20_epoch.h5')
```

Πρόβλεψη χρησιμοποιώντας δεδομένα από κάμερα

```
import cv2
import numpy as np
from PIL import Image
from keras import models

font = cv2.FONT_HERSHEY_SIMPLEX

model = models.load_model(r'./Model/model_20_epoch.h5')

video = cv2.VideoCapture(0)

def train_target_gen(image):

    image_array = np.asarray(image)
    image_array = cv2.resize(image, (64,64))
    image_array= (image_array/255)
    image_array = image_array.reshape(-1,64,64,1)

    yield (image_array)

while True:

    _, frame = video.read()
    im=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    img_array = np.array(im)
    prediction = model.predict_generator(train_target_gen((img_array)), steps=1)
    y_classes = prediction.argmax(axis=-1)

    font = cv2.FONT_HERSHEY_SIMPLEX
    bottomLeftCornerOfText = (30,30)
    fontScale = 1
    fontColor = (255,255,255)
    lineType = 2
    cv2.putText(frame,str(y_classes),
    bottomLeftCornerOfText,
    font,
    fontScale,
    fontColor,
    lineType)

    cv2.imshow("Capturing", frame)
    key=cv2.waitKey(1)
    if key ==ord('q'):
        break
video.release()
cv2.destroyAllWindows()
```

5.12 Προσομοιωτής και τρόπος συλλογής νέων δεδομένων.

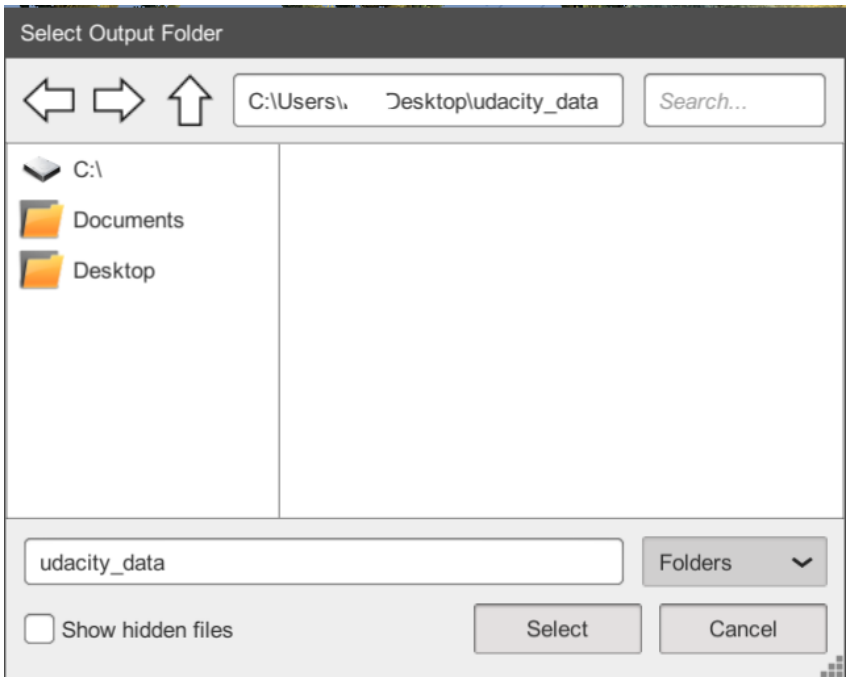
Αφού κατεβάσουμε τα αρχεία του προσομοιωτή, εκτελούμε το πρόγραμμα.



Για την συλλογή νέων δεδομένων πατάμε στην επιλογή «Εκπαίδευση» (*Training Mode*).

Τότε βλέπουμε το αυτοκίνητο στην πίστα έτοιμο να οδηγηθεί από τον χρήστη, όπως και επίσης την επιλογή για την καταγραφή των δεδομένων (*Record*).

Πατώντας «Καταγραφή» ο προσομοιωτής ζητά από τον χρήστη σε ποιον φάκελο θα αποθηκευτούν τα δεδομένα.

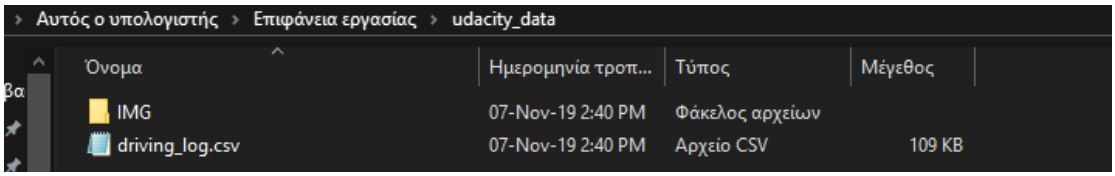


Ολοκληρώνοντας και αυτό το τελευταίο βήμα, ο χρήστης είναι πλέον έτοιμος να οδηγήσει ελεύθερα το αυτοκίνητο στην πρώτη πίστα του προσομοιωτή.

Για την καλή ποιότητα και ποσότητα των δεδομένων πρέπει να ολοκληρωθεί τουλάχιστον τρεις φορές αυτή η πίστα και το όχημα να μετατοπίζεται όσο το δυνατό περισσότερο μέσα στα όρια τις πίστας. Έτσι ώστε να υπάρχει ποικιλία στις γωνίες (*Diverse Data*).

Όταν πλέον έχουμε τελειώσει με την οδήγηση πατάμε ξανά «Καταγραφή» (*Record*) ο προσομοιωτής μαζεύει όλα τα δεδομένα και τα αποθηκεύει στον φάκελο που επιλέχθηκε παραπάνω.

Ο φάκελος περιέχει έναν φάκελο που έχουν αποθηκευτεί όλες οι εικόνες κατά την διάρκεια της εκπαίδευσης από τις τρεις κάμερες «Αριστερή, Κεντρική, Δεξιά», και ένα αρχείο ***.csv*** το οποίο περιέχει όλες τις τιμές των αισθητήρων όπως γωνία τιμονιού, ταχύτητα κλπ.



5.12.1 Κωδικας CNN (Παλινδρόμηση)

```
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense, BatchNormalization
from keras.layers.convolutional import MaxPooling2D, Conv2D
from keras.optimizers import Adam
```

```
Using TensorFlow backend.
```

Έκδοση του Tensorflow, Keras για το παράδειγμα αυτό

```
import keras
print(keras.__version__)
```

```
2.2.4
```

```
import tensorflow
print(tensorflow.__version__)
```

```
1.8.0
```

Δεδομένα

```
df = pd.read_csv("driving_log.csv")
```

```
max(df["steering"])
```

```
1.0
```

```
min(df["steering"])
```

```
-0.9426954
```

```
print("Unique Values:",df.steering.unique())
print("Unique Numbers:",len(df.steering.unique()))
```

```
Unique Values: [ 0.          0.0617599   0.05219137  0.3679529   0.5784606   0.1670138
  0.08089697  0.0904655   0.1574452   0.1765823  -0.0787459   0.38709
  0.3583844  -0.03127411 -0.05975719 -0.04076847  0.01391724  0.1287396
  0.2148564  -0.08824026 -0.2306556  -0.1547008   0.04262284  0.1478767
  0.2531306  -0.135712   -0.2781274  -0.3255992  -0.2591387   0.03305431
  0.243562    0.2626991  -0.06925154  0.3488158  -0.1167233  -0.09773462
  0.02348577  0.100034    0.2339935   0.00434871  0.1191711   0.1957194
  0.2722676  -0.05026283  0.1383082   0.3105417  -0.0122854   0.07132844
 -0.02177976  0.2818362   0.4157956   0.4540697   0.406227    0.1861508
  0.1096026  -0.2211613  -0.1831838  -0.2686331  -0.2116669  -0.1452064
 -0.373071   -0.3920597  -0.401554   -0.3825653  -0.2876218  -0.1262177
 -0.1736895  -0.09279104  0.3009732   0.5114809  -0.3066105  -0.6294187
 -0.6863848   0.6358718   0.6454403  -0.2971161   0.2052879   0.3775214
  0.3201102   0.3296788  -0.1641951  -0.107229    0.2914047  -0.1926782
 -0.2021725   0.224425   -0.24015    -0.4964977  -0.505992   -0.6009356
  0.4349326   0.5880292  -0.771834   -0.9237437  -0.8098114   0.4636382
  0.4732068   0.3392473   0.4827753   0.4923438  -0.3161048   1.
  0.7315571  -0.4205428  -0.4110484   0.6550089   0.6263033  -0.9332381
 -0.9426954  -0.4395315  -0.4490258  -0.2496443  -0.3540823  -0.4680146
 -0.7243622   0.4445012  -0.3445879  -0.4775089  -0.4300371  -0.3350936
 -0.4585202  -0.5249807   0.4253641  -0.5344751 ]
Unique Numbers: 124
```

```
df.isnull().values.any()
```

```
False
```

Εμφάνιση των 5 πρώτον σειρών

```
df.head()
```

	center	left	right	steering	throttle	brake	speed
0	IMG/center_2016_12_01_13_30_48_287.jpg	IMG/left_2016_12_01_13_30_48_287.jpg	IMG/right_2016_12_01_13_30_48_287.jpg	0.0	0.0	0.0	22.148290
1	IMG/center_2016_12_01_13_30_48_404.jpg	IMG/left_2016_12_01_13_30_48_404.jpg	IMG/right_2016_12_01_13_30_48_404.jpg	0.0	0.0	0.0	21.879630
2	IMG/center_2016_12_01_13_31_12_937.jpg	IMG/left_2016_12_01_13_31_12_937.jpg	IMG/right_2016_12_01_13_31_12_937.jpg	0.0	0.0	0.0	1.453011
3	IMG/center_2016_12_01_13_31_13_037.jpg	IMG/left_2016_12_01_13_31_13_037.jpg	IMG/right_2016_12_01_13_31_13_037.jpg	0.0	0.0	0.0	1.438419
4	IMG/center_2016_12_01_13_31_13_177.jpg	IMG/left_2016_12_01_13_31_13_177.jpg	IMG/right_2016_12_01_13_31_13_177.jpg	0.0	0.0	0.0	1.418236

Έλεγχος των δεδομένων

```
df['center'][0]
```

```
'IMG/center_2016_12_01_13_30_48_287.jpg'
```

```
df['left'][0]
```

```
' IMG/left_2016_12_01_13_30_48_287.jpg'
```

```
df['right'][0]
```

```
' IMG/right_2016_12_01_13_30_48_287.jpg'
```

Διόρθωση ' IMG/right_2016_12_01_13_30_48_287.jpg'

```
df["left"] = df["left"].str.lstrip()
```

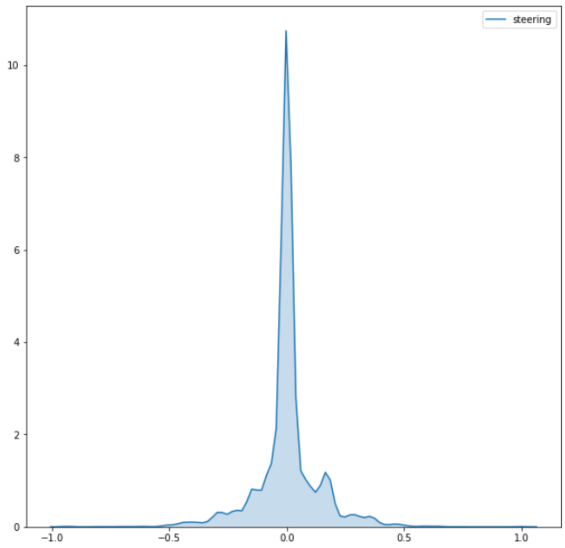
```
df["right"] = df["right"].str.lstrip()
```

Έλεγχος της στήλης γωνίας (steering)

Κατανομή στήλης *steering*:

```
import seaborn as sns
```

```
plt.figure(figsize=(10,10))
sns.kdeplot(df.steering, shade=True);
```



```
def preprop(image):
    image = cv2.cvtColor(cv2.resize(image[90:140,:], (75,75)), cv2.COLOR_BGR2RGB)
    image = cv2.cvtColor(cv2.resize(image[:-20,:,], (75,75)), cv2.COLOR_BGR2RGB)
    return image
```

Κώδικας γεννήτριας και προ-επεξεργασία

```
def train_gen(batch_size):
    while True:

        steps=0
        angle = []
        img =[]
        for k,row in df.iterrows():

            steps = steps+1

            current_angle =np.float32(row['steering'])

            #Center image
            cent_0 = cv2.imread(row['center'],-1)
            cent_0=preprop(cent_0)

            img.append(np.array(cent_0))
            angle.append(np.array(current_angle))

            img.append(np.fliplr(cent_0))
            angle.append(np.array(-current_angle))

            # Left image
            left_0 = cv2.imread(row['left'],-1)
            left_0 = preprop(left_0)
            img.append(np.array(left_0))
            angle.append(current_angle+0.45)

            # Right image
            right_0 = cv2.imread(row['right'],-1)
            right_0 = preprop(right_0)

            img.append(np.array(right_0))
            angle.append(current_angle-0.45)
            img_reshaped = np.array(img)
            img_reshaped = img_reshaped.reshape(-1,75,75,3)

            if steps == batch_size:
                img=np.array(img)
                yield (np.array(img/255), np.array(angle))
                steps = 0
                angle = []
                img =[]
```

Οι 3 εικόνες από τις κάμερες του αυτοκινήτου

```
center=plt.imread(df["Center"][0])
```

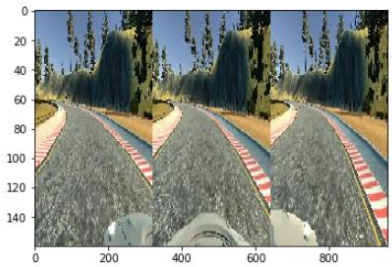
```
left=plt.imread(df["Left"][0])
```

```
right=plt.imread(df["Right"][0])
```

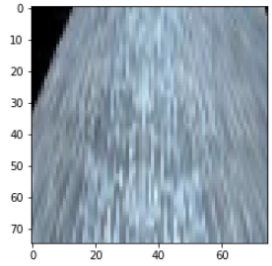
```
whole_img = np.concatenate((left,center, right),axis=1)
```

```
plt.imshow(whole_img, cmap='gray', aspect=4)
```

```
<matplotlib.image.AxesImage at 0x14b2cf87f98>
```



```
plt.imshow(image[2]);
# Left image
```



```
target[2]
# Left image target
```

```
0.45
```

Έλεγχος γεννήτριας για μια παρτίδα και τα δεδομένα

```
train_generator = train_gen(32)
```

```
image, target = next(train_generator)
```

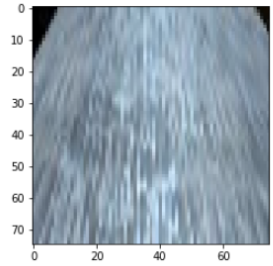
```
np.shape(image)
```

```
(128, 75, 75, 3)
```

```
np.shape(target)
```

```
(128,)
```

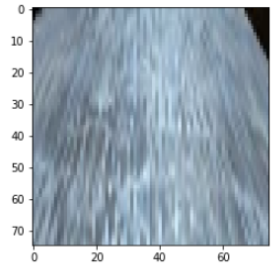
```
plt.imshow(image[0]);
# Center image
```



```
target[0]
# Center target
```

```
0.0
```

```
plt.imshow(image[3]);
# Right image
```

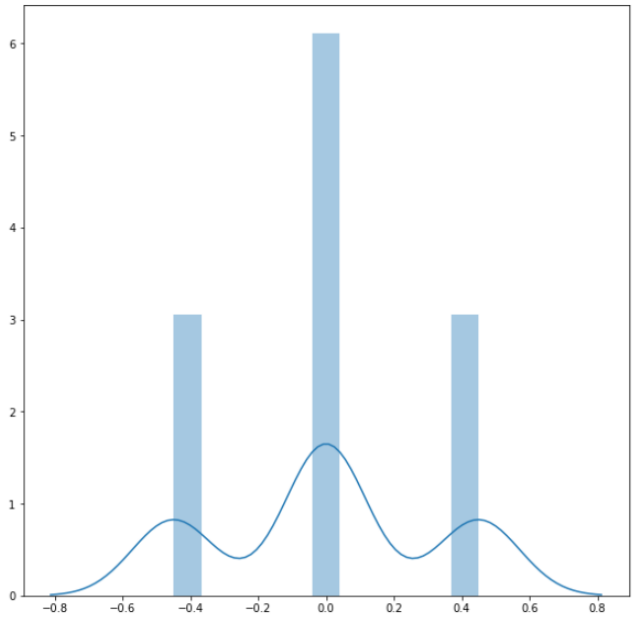


```
target[3]
# Right image target
```

```
-0.45
```

Η γεννήτρια επιστρέφει σωστά τα δεδομένα και η προ-επεξεργασία έγινε χωρίς λάθη.

```
plt.figure(figsize=(10,10));
sns.distplot(target);
```



Απεικόνιση κατανομής πίνακα στόχου (*target*[19]):

Η τιμή «***±.45***» επιλέχθηκε εμπειρικά.

Ορισμός μοντέλου.

```
model = Sequential()
```

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 36, 36, 32)        2432
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 9, 9, 32)          0
_________________________________________________________________
batch_normalization_1 (Batch (None, 9, 9, 32)          128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 7, 7, 16)          4624
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 16)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 144)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               37120
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_2 (Batch (None, 256)               1024
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 11
=================================================================
Total params: 47,909
Trainable params: 47,333
Non-trainable params: 576
_________________________________________________________________
```

[19] Πίνακας ο οποίος είναι αποθηκευμένες οι τιμές γωνίας.

Εκπαίδευση του μοντέλου:

```
model.fit_generator(generator=train_generator, steps_per_epoch= 251, epochs= 6, verbose=1)
```

```
Epoch 1/6
251/251 [==============================] - 27s 108ms/step - loss: 0.2919 - acc: 0.2081
Epoch 2/6
251/251 [==============================] - 29s 115ms/step - loss: 0.0725 - acc: 0.2634
Epoch 3/6
251/251 [==============================] - 26s 104ms/step - loss: 0.0593 - acc: 0.2678
Epoch 4/6
251/251 [==============================] - 27s 106ms/step - loss: 0.0546 - acc: 0.2689
Epoch 5/6
251/251 [==============================] - 26s 104ms/step - loss: 0.0513 - acc: 0.2695
Epoch 6/6
251/251 [==============================] - 26s 104ms/step - loss: 0.0494 - acc: 0.2693
```

Ελέγχοντας τις προβλέψεις στην παρακάτω εικόνα/κώδικα θα παρατηρήσουμε πως το μοντέλο «***έμαθε***» να κάνει σχετικά καλές προβλέψεις[20].

```
preds =model.predict_generator(generator=train_generator, steps=7)
```

```
preds
```

```
array([[-1.12754196e-01],
       [ 4.70524579e-02],
       [ 2.96085536e-01],
       [-2.90649354e-01],
       [ 1.48337698e-02],
       [-1.13654673e-01],
       [ 3.72639894e-01],
       [-2.58558869e-01],
       [ 8.23116004e-02],
       [-4.97572273e-02],
       [ 3.62937987e-01],
       [-2.68470705e-01],
       [ 3.41031849e-02],
       [-1.14294648e-01],
       [ 3.49711359e-01],
       [-2.35471711e-01],
       [ 6.27167672e-02],
       [-1.45753607e-01],
       [ 3.67643267e-01],
       [-2.40741476e-01],
```

```
max(preds)
```

```
array([0.6488244], dtype=float32)
```

```
min(preds)
```

```
array([-0.57588226], dtype=float32)
```

[20] Εννοώντας πως οι τιμές που πρόβλεψε είναι «ΛΟΓΙΚΕΣ» γωνίες.

Αποθήκευση μοντέλου:

```
model.save('model_final.h5')
```

Αλλαγή κώδικα του αρχείου «***Drive.py***[21]» και προσαρμογή του για το μοντέλο.

[21] Έγινε χρήση αυτής της έκδοσης https://github.com/harveenchadha/Udacity-CarND-Behavioral-Cloning-Final/blob/master/drive.py

Το αρχείο αυτό αποτελεί ουσιαστικά τον τρόπο με τον οποίο το μοντέλο θα επικοινωνήσει με τον προσομοιωτή. Η ταχύτητα του οχήματος παραμένει σταθερή με την βοήθεια ενός «***PID***»

```
class SimplePIController:
    def __init__(self, Kp, Ki):
        self.Kp = Kp
        self.Ki = Ki
        self.set_point = 0.
        self.error = 0.
        self.integral = 0.

    def set_desired(self, desired):
        self.set_point = desired

    def update(self, measurement):
        # proportional error
        self.error = self.set_point - measurement

        # integral error
        self.integral += self.error

        return self.Kp * self.error + self.Ki * self.integral


controller = SimplePIController(0.1, 0.002)
set_speed = 9
controller.set_desired(set_speed)
```

Στο αρχείο προσθέτουμε μια νέα συνάρτηση με βάση το μοντέλο που θα χρησιμοποιήσουμε:

```
def train_target_gen(image):
    image_array = np.asarray(image)
    image_array=cv2.cvtColor(image_array, cv2.COLOR_BGR2RGB)

    image_array = image_array[90:140,:,:]
    image_array = image_array[:-20,:,:]
    image_array = cv2.resize(image_array, (75,75))

    image_array = image_array.reshape(-1, 75, 75 , 3)

    yield (image_array/255)
```

Μέσα στην συνάρτηση «***telemetry***»:

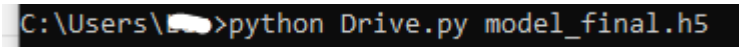
```
@sio.on('telemetry')
def telemetry(sid, data):
    if data:
        # The current steering angle of the car
        steering_angle = data["steering_angle"]
        # The current throttle of the car
        throttle = data["throttle"]
        # The current speed of the car
        speed = data["speed"]
        # The current image from the center camera of the car
        imgString = data["image"]
        image = Image.open(BytesIO(base64.b64decode(imgString)))
```

Προσθέτουμε:

```
steering_angle = float(model.predict_generator(train_target_gen(image), steps=1))
```

Τέλος ανοίγουμε ξανά τον προσομοιωτή και αυτήν την φορά επιλέγουμε την αυτόνομη λειτουργία.

Τρέχουμε το αρχείο «*Drive.py*» σε γραμμή εντολών υποδεικνύοντας την τοποθεσία του μοντέλου που θα χρησιμοποιηθεί. Όπως για παράδειγμα εάν έχει ανοιχτεί γραμμή εντολών στο ίδιο αρχείο που έχει το *μοντέλο* και το *Drive.py* μαζί:



ΒΙΒΛΙΟΓΡΑΦΙΑ

Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000). *LOF: Identifying Density-Based Local Outliers*. Retrieved from http://www.dbs.ifi.lmu.de: http://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf

Chaurasia, K. (n.d.). *DBSCAN Clustering Algorithm*. Retrieved from practice2code.blogspot: https://practice2code.blogspot.com/2017/07/dbscan-clustering-algorithm.html

Dillon, N. (2018, July 9-15). *Safe handling instructions for missing data.* Retrieved from http://conference.scipy.org: http://conference.scipy.org/proceedings/scipy2018/pdfs/dillon_niederhut.pdf

Ertoz, L., Steinbach, M., & Kumar, V. (2003, February 20). *Finding Clusters of Different Sizes, Shapes, and Densities in Noisy,.* Retrieved from users.cs.umn: https://www-users.cs.umn.edu/~kumar001/papers/SIAM_snn.pdf

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). *AAAI.* Retrieved from www.aaai.org: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. Retrieved 2 13, 2019, from https://www.bioinf.jku.at/publications/older/2604.pdf

Hsu, Y.-L., Huang, P.-Y., & Chen, D.-T. (2014, May 6). *Sparse principal component analysis in cancer research.* Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4692276/

Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv: Computation and Language*. Retrieved 2 13, 2019, from https://arxiv.org/pdf/1508.01991.pdf

Jonathon, S., Systems Neurobiology Laboratory, S. I., & Institute for Nonlinear Science, U. o. (2005, December 10). *A Tutorial on Principal Component Analysis.* Retrieved from https://www.cs.cmu.edu/~elaw/papers/pca.pdf

Koppen, M. (n.d.). *The curse of dimensionality.* Retrieved from yaroslavvb: http://www.yaroslavvb.com/papers/koppen-curse.pdf

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). *Isolation Forest.* Retrieved from cs.nju.edu.cn: https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf

Ozdemir, S. (2017). *Principles Of Data Science.* Birmingham,UK: Packt Publishing Ltd.

Seo, S. (2002). *A Review and Comparison of Methods for Detecting Outliers.* Retrieved from http://d-scholarship.pitt.edu/7948/1/Seo.pdf

sklearn.neighbors.LocalOutlierFactor. (n.d.). Retrieved from scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html#sklearn.neighbors.LocalOutlierFactor.kneighbors

Smith, L. I. (2002). *A tutorial on Principal Components Analysis*. Retrieved 1 7, 2019, from http://cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

Tukey, J. W. (1977). *Exploratory Data Analysis.* Addison-Wesley. Retrieved 11 30, 2018

Usman, M. (2018, May 10). *Implementing PCA in Python with Scikit-Learn.* Retrieved from https://stackabuse.com/implementing-pca-in-python-with-scikit-learn/

Williams, A. (2016, March 27). *Everything you did and didn't know about PCA*. Retrieved from http://alexhwilliams.info/itsneuronalblog/2016/03/27/pca/

Zou, H., Hastie, T., & Tibshirani, R. (2006). *Sparse Principal Component Analysis.* Retrieved 1 8, 2019, from https://web.stanford.edu/~hastie/Papers/spc_jcgs.pdf