Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών

Εθνικό Μετσόβιο Πολυτεχνείο

Πρόβλεψη Κατανάλωσης Ενέργειας με Χρήση Μεθόδων Μηχανικής Μάθησης

Κωνσταντίνα Μαρία Καμπανέλλη

Επιβλέπων Καθηγητής: Κωνσταντίνος Κουσουρής

Αθήνα, Σεπτέμβριος 2023

Ευχαριστίες

Στο τέλος των προπτυχιακών σπουδών μου, οι οποίες ολοκληρώνονται με την εκπόνηση της διπλωματικής μου εργασίας, επιθυμώ να απευθύνω τις πιο εγκάρδιες ευχαριστίες μου στον επιβλέποντα καθηγητή μου κ. Κωνσταντίνο Κουσουρή, για την αδιάκοπη υποστήριξή του και την ουσιαστική καθοδήγηση, ο οποίος προσέφερε απλόχερα τη βοήθειά και την εμπιστοσύνη του σε όλη τη διάρκεια της συνεργασίας μας. Είμαι ευγνώμων για την ευκαιρία που μου δόθηκε να εργαστώ σε ένα τόσο ενδιαφέρον και σύγχρονο θέμα. Αυτή η διαδικασία συνέβαλε στο να ωριμάσω ακαδημαϊκά και η εμπειρία αυτή θα με συνοδεύει στα επόμενα χρόνια. Τέλος, δε θα μπορούσα να μην ευχαριστήσω την οικογένειά μου για τη στήριξη που μου παρείχαν σε υλικό και ψυχολογικό επίπεδο στην ως τώρα πορεία μου.

Κωνσταντίνα Μαρία Καμπανέλλη

Αθήνα, Σεπτέμβριος 2023

Περίληψη

Η παρούσα διπλωματική εργασία παρουσιάζει και συζητά μοντέλα πρόβλεψης ενεργειακής χρήσης συσκευών. Τα δεδομένα που χρησιμοποιούνται περιλαμβάνουν μετρήσεις θερμοκρασίας και υγρασίας 9 δωματίων ενός σπιτιού, που παρακολουθούνται με ένα ασύρματο δίκτυο αισθητήρων ZigBee. Περιλαμβάνει επίσης μετεωρολογικά και κλιματικά δεδομένα, όπως θερμοκρασία, πίεση, υγρασία, ταχύτητα ανέμου, ορατότητα και σημείο υγροποίησης που μετρήθηκαν από το αεροδρόμιο Chievres. Το σύνολο δεδομένων συλλεγόταν 4,5 μήνες. Πέντε στατιστικά μοντέλα εκπαιδεύτηκαν και αξιολογήθηκαν: (α) πολλαπλή γραμμική παλινδρόμηση, (β) λογιστική παλινδρόμηση, (γ) νευρωνικά δίκτυα, (δ) gradient boosting decision trees (GBDT) και (ε) τυχαίο δάσος. Εξετάστηκαν και συζητήθηκαν παραλλαγές του καθενός. Το καλύτερο μοντέλο ήταν το νευρωνικό δίκτυο με 7 κρυφά στρώματα.

Λέξεις κλειδιά

Πρόβλεψη κατανάλωσης ενέργειας, μηχανική μάθηση, τεχνητή νοημοσύνη, γραμμική παλινδρόμηση, λογιστική παλινδρόμηση, νευρωνικά δίκτυα, δέντρα απόφασης, gradient boosting decision trees, τυχαίο δάσος.

Abstract

This diploma thesis presents and discusses data-driven predictive models for the energy use of appliances. Data used include temperature and humidity measurements of 9 rooms in a house, monitored by a wireless ZigBee sensor network. It also includes meteorological and climatic data such as temperature, pressure, humidity, wind speed, visibility and humidity point measured from Chievres airport. The dataset was collected over 4.5 months. Five statistical models were trained and evaluated in a testing set: (a) multiple linear regression, (b) logistic regression, (c) neural networks, (d) gradient boosting decision trees (GBDT) and (e) random forest. Alterations of each were examined and discussed. The best model was the neural network with 7 hidden layers.

Key Words

Energy forecasting, machine learning, artificial intelligence, linear regression, logistic regression, neural networks, decision trees, gradient boosting decision trees, random forest.

Περιεχόμενα

1 Εισαγωγή

Στην εργασία αυτή ασχολούμαστε με ένα πρόβλημα παλινδρόμησης από τον κλάδο της ενέργειας. Βασική επιδίωξη είναι η ανάπτυξη μεθόδων και η προσαρμογή μοντέλων σε ένα σύνολο μετρήσεων που αφορούν την πρόβλεψη κατανάλωσης αυτής. Προβλήματα πρόβλεψης τιμών και γενικότερα προβλήματα που στην επίλυσή τους χρησιμοποιείται Μηχανική Μάθηση εμφανίζονται όλα και πιο συχνά στον τομέα της ενέργειας.

Στην εποχή που διανύουμε, η παραγωγή και η αξιοποίηση της ενέργειας παίζει καθοριστικότατο ρόλο στην εύρυθμη λειτουργεία της οικονομίας, την κοινωνικής και πολιτικής δραστηριότητας, αλλά και στη γενικότερη βελτίωση του βιοτικού επιπέδου. Με την πάροδο του χρόνου, οι ανάγκες όλο και αυξάνονται τόσο σε οικιακό, όσο και βιομηχανικό επίπεδο. Αυτή η αυξημένη ζήτηση οδηγεί τους επιστήμονες στο να προσπαθήσουν να βρουν πηγές άφθονης και διαθέσιμης ενέργειας με το ελάχιστο δυνατό κόστος. Οι προσπάθειες αυτές κατευθύνονται προς την ανεύρεση νέων πηγών ενέργειας, την αναβάθμιση των εγκαταστάσεων παραγωγής, τη βελτιστοποίηση των δικτύων μεταφοράς και διανομής ενέργειας και την επίτευξη καλύτερης και αποδοτικότερης χρήσης της σε επίπεδο των καταναλωτών.

Το φαινόμενο της κλιματικής αλλαγής εγείρει ανησυχίες σχετικά με τις περιβαλλοντικές και οικονομικές συνέπειες. Αυτό γεννά την ανάγκη της διαχείρισης κατανάλωσης και εδώ υπεισέρχεται ο τομέας της Μηχανικής Μάθησης. Οι δυνατότητες της και η αυξημένη υπολογιστική της ισχύς είναι σε θέση να παρέχει αξιόπιστες και προβλέψεις κατανάλωσης ενέργειας με μικρές αποκλίσεις. Αυτές είναι χρήσιμες στον σχεδιασμό και τη λήψη βέλτιστων αποφάσεων σχετικά με συστήματα ενέργειας.

Στην εργασία μας το πρόβλημα που καλούμαστε να αντιμετωπίσουμε είναι η πρόβλεψη κατανάλωσης ενέργειας ενός σπιτιού βασισμένο σε πραγματικά δεδομένα. Στο σύνολο δεδομένων υπάρχουν πληροφορίες συλλεγμένες σε διάστημα τεσσεράμισι μηνών ανά δέκα λεπτά. Οι συνθήκες θερμοκρασίας και υγρασίας του σπιτιού παρακολουθούνταν με ένα ασύρματο δίκτυο αισθητήρων ZigBee. Κάθε ασύρματος κόμβος μετέδιδε τις συνθήκες θερμοκρασίας και υγρασίας περίπου τριάμισι λεπτά. Στη συνέχεια, τα ασύρματα δεδομένα υπολογίστηκαν κατά μέσο όρο για περιόδους δέκα λεπτών. Τα ενεργειακά δεδομένα καταγράφονταν κάθε δέκα λεπτά με μετρητές ενέργειας m-bus. Ο καιρός από τον πλησιέστερο μετεωρολογικό σταθμό αεροδρομίου (αεροδρόμιο Chievres, Βέλγιο) μεταφορτώθηκε από ένα δημόσιο σύνολο δεδομένων από την Reliable Prognosis (rp5. ru) και συγχωνεύθηκε με τα πειραματικά σύνολα δεδομένων χρησιμοποιώντας τη στήλη ημερομηνίας και ώρας.

Πιο αναλυτικά, οι επεξηγηματικές μεταβλητές είναι οι εξής: η ημερομηνία (έτος-μήνας-μέρα ώρα: λεπτά: δευτερόλεπτα), η χρήση ενέργειας των φωτιστικών σωμάτων στο σπίτι σε Wh, η θερμοκρασία στον χώρο της κουζίνας σε βαθμούς Κελσίου, η υγρασία στον χώρο της κουζίνας επί τοις εκατό, η θερμοκρασία στον χώρο του καθιστικού σε βαθμούς Κελσίου, η υγρασία στον χώρο του καθιστικού επί τοις εκατό, η θερμοκρασία στον χώρο του πλυντηρίου σε βαθμούς Κελσίου, η υγρασία στον χώρο του πλυντηρίου επί τοις εκατό, η θερμοκρασία στον χώρο του γραφείου σε βαθμούς

Κελσίου, η υγρασία στον χώρο του γραφείου επί τοις εκατό, η θερμοκρασία στο μπάνιο σε βαθμούς Κελσίου, η υγρασία στο μπάνιο επί τοις εκατό, η θερμοκρασία έξω από το κτίριο (βόρεια πλευρά) σε βαθμούς Κελσίου, η υγρασία έξω από το κτίριο (βόρεια πλευρά) επί τοις εκατό, η θερμοκρασία στο δωμάτιο σιδερώματος σε βαθμούς Κελσίου, η υγρασία στο δωμάτιο σιδερώματος επί τοις εκατό, η θερμοκρασία στο δωμάτιο των εφήβων σε βαθμούς Κελσίου, η υγρασία στο δωμάτιο των εφήβων επί τοις εκατό, η θερμοκρασία στο δωμάτιο των γονέων σε βαθμούς Κελσίου, η υγρασία στο δωμάτιο των γονέων επί τοις εκατό, η θερμοκρασία έξω (από τον μετεωρολογικό σταθμό) σε βαθμούς Κελσίου, η πίεση (στον μετεωρολογικό σταθμό) σε mm Hg, η υγρασία έξω (από τον μετεωρολογικό σταθμό) επί τοις εκατό, η ταχύτητα του ανέμου (στον μετεωρολογικό σταθμό) σε μέτρα ανά ώρα, η ορατότητα (στον μετεωρολογικό σταθμό) σε χιλιόμετρα, το σημείο υγροποίησης (στον μετεωρολογικό σταθμό) σε Â°C, μια τυχαία μεταβλητή 1 (αδιάστατη) και μια ακόμα τυχαία μεταβλητή 2 (αδιάστατη). Η εξαρτώμενη μεταβλητή, την οποία καλούμαστε να προβλέψουμε, είναι η συνολική κατανάλωση ενέργειας των οικιακών συσκευών σε Wh.

Το σύνολο δεδομένων προέρχεται από την ιστοσελίδα kaggle.com και ο ακριβής σύνδεσμος είναι https://www.kaggle.com/datasets/loveall/appliances-energy-prediction . Έχουμε στη διάθεση μας 19735 γεγονότα. Συνεπώς, το αρχείο περιέχει 19735 γραμμές και 29 στήλες εκ των οποίων οι 28 είναι οι επεξηγηματικές μεταβλητές που χαρακτηρίζουν το κάθε γεγονός και η μια εξαρτώμενη μεταβλητή.

Πρώτα, εφαρμόστηκε ένας γραμμικός αλγόριθμος πολλαπλής παλινδρόμησης. Έπειτα, υλοποιήθηκαν λογιστική παλινδρόμηση, Νευρωνικά Δίκτυα και Ενδυναμωμένα Δάση (Gradient Boosting Decision Trees και Random Forest Decision Trees). Τις αποδοτικότερες προβλέψεις έδωσε το Νευρωνικό Δίκτυο με τα περισσότερα κρυφά στρώματα.

2 Μηχανική Μάθηση

Η μηχανική μάθηση είναι ένας κλάδος της τεχνητής νοημοσύνης (AI) και της επιστήμης των υπολογιστών που επικεντρώνεται στη χρήση δεδομένων και αλγορίθμων για τη μίμηση του τρόπου, με τον οποίο μαθαίνει ο άνθρωπος, βελτιώνοντας σταδιακά την ακρίβειά του. Ασχολείται με τη σχεδίαση συστημάτων που μπορούν να βελτιώσουν την απόδοση κατά την εκτέλεση συγκεκριμένων εργασιών χωρίς την ανάγκη επαναπρογραμματισμού. Ένα τέτοιο σύστημα χρησιμοποιεί την προηγούμενη γνώση και εμπειρία, για να ορίσει λογικά τις αναγκαίες και επαρκείς συνθήκες που αντιστοιχούν σε μια δεδομένη κατηγορία αντικειμένων.

Τις τελευταίες δύο δεκαετίες, η τεχνολογική πρόοδος στον τομέα της αποθήκευσης και της επεξεργαστικής ισχύος γέννησε την ανάγκη για καινοτόμα προϊόντα που βασίζονται στη μηχανική μάθηση, όπως η μηχανή συστάσεων του Netflix και τα αυτοκινούμενα αυτοκίνητα.

Η μηχανική μάθηση είναι ένα σημαντικό στοιχείο του αναπτυσσόμενου τομέα της επιστήμης των δεδομένων. Μέσω της χρήσης στατιστικών μεθόδων, οι αλγόριθμοι εκπαιδεύονται για να κάνουν ταξινομήσεις ή προβλέψεις και να αποκαλύπτουν βασικές πληροφορίες σε εργασίες εξόρυξης δεδομένων. Αυτές οδηγούν στη συνέχεια στη λήψη αποφάσεων επηρεάζοντας την ανάπτυξη, λόγου χάρη επιχειρήσεων. Καθώς τα μεγάλα δεδομένα συνεχίζουν να επεκτείνονται και να αυξάνονται, η ζήτηση της αγοράς για επιστήμονες δεδομένων θα αυξηθεί. Θα τους ζητηθεί να βοηθήσουν στον εντοπισμό των πιο σχετικών επιχειρηματικών ερωτημάτων και στην απάντησή τους.

Η μηχανική μάθηση είναι σημαντική επειδή δίνει στις επιχειρήσεις μια εικόνα των τάσεων στη συμπεριφορά των πελατών και των επιχειρησιακών λειτουργικών προτύπων, καθώς και υποστηρίζει την ανάπτυξη νέων προϊόντων. Πολλές από τις σημερινές κορυφαίες εταιρείες, όπως το Facebook, η Google και η Uber, καθιστούν τη μηχανική μάθηση κεντρικό μέρος των λειτουργιών τους. Η μηχανική εκμάθηση έχει γίνει ένα σημαντικό ανταγωνιστικό διαφοροποιητικό στοιχείο για πολλές εταιρείες.

Η κλασική μηχανική μάθηση συχνά κατηγοριοποιείται με βάση τον τρόπο με τον οποίο ένας αλγόριθμος μαθαίνει να γίνεται πιο ακριβής στις προβλέψεις του. Υπάρχουν τέσσερις βασικές προσεγγίσεις: μάθηση με επίβλεψη, μάθηση χωρίς επίβλεψη, μάθηση με ημιεπίβλεψη και μάθηση με ενίσχυση. Ο τύπος αλγορίθμου που επιλέγουν να χρησιμοποιήσουν οι επιστήμονες δεδομένων εξαρτάται από τον τύπο των δεδομένων που θέλουν να προβλέψουν.

Κατά τη μάθηση με επίβλεψη, κατασκευάζεται μια συνάρτηση, όπου τα δεδομένα εισόδου (σύνολο εκπαίδευσης) έχουν γνωστή επιθυμητή έξοδο και στόχος είναι η γενίκευση με άγνωστη έξοδο. Παραδείγματα τέτοιων αλγορίθμων είναι η γραμμική και λογιστική παλινδρόμηση, οι Μηχανές Διανυσμάτων Υποστήριξης (SVM), τα Δέντρα Απόφασης (Decision Trees) και τα Νευρωνικά Δίκτυα (NN).

Κατά τη μάθηση χωρίς επίβλεψη, οι αλγόριθμοι εκπαιδεύονται με δεδομένα εισόδου, των οποίων η έξοδος δεν είναι γνωστή. Ο αλγόριθμος σαρώνει σύνολα δεδομένων αναζητώντας οποιαδήποτε ουσιαστική σύνδεση. Τα δεδομένα στα οποία εκπαιδεύονται

οι αλγόριθμοι καθώς και οι προβλέψεις ή οι συστάσεις που εξάγουν είναι προκαθορισμένα.

Στη μάθηση με ημιεπίβλεψη περιλαμβάνεται ένα μείγμα των δύο προηγούμενων τύπων. Οι επιστήμονες δεδομένων μπορούν να τροφοδοτήσουν έναν αλγόριθμο κυρίως με επισημασμένα (labeled) δεδομένα εκπαίδευσης, αλλά το μοντέλο είναι ελεύθερο να εξερευνήσει τα δεδομένα μόνο του και να αναπτύξει τη δική του κατανόηση πάνω στο σύνολο δεδομένων. Παραδείγματα εφαρμογής αποτελούν η μηχανική μετάφραση (Διδασκαλία αλγορίθμων για τη μετάφραση γλώσσας βασιζόμενη σε λιγότερο από ένα πλήρες λεξικό λέξεων) και η ανίχνευση απάτης.

Η μάθηση με ενίσχυση χρησιμοποιείται συνήθως για να διδαχτεί μια μηχανή πως να ολοκληρώσει μια διαδικασία πολλών βημάτων, για την οποία υπάρχουν σαφώς καθορισμένοι κανόνες. Οι επιστήμονες προγραμματίζουν έναν αλγόριθμο, για να ολοκληρώσει μια εργασία δίνοντας του θετικές ή αρνητικές ενδείξεις, καθώς αυτός επεξεργάζεται τον τρόπο ολοκλήρωσης μιας εργασίας. Αλλά ως επί το πλείστον, ο αλγόριθμος αποφασίζει μόνος του ποια βήματα θα ακολουθήσει στην πορεία, ελαχιστοποιώντας το ρίσκο. Χαρακτηριστικές εφαρμογές αυτού του τύπου μάθησης είναι στη ρομποτική, όπου τα ρομπότ μαθαίνουν να εκτελούν εργασίες στον φυσικό κόσμο και τη διαχείριση πεπερασμένων πόρων των επιχειρήσεων και την κατανομή αυτών.

3 Προπαρασκευή Δεδομένων

Στην εργασία μας το πρόβλημα που καλούμαστε να αντιμετωπίσουμε είναι η πρόβλεψη κατανάλωσης ενέργειας ενός σπιτιού βασισμένο σε πραγματικά δεδομένα. Στο σύνολο δεδομένων υπάρχουν πληροφορίες συλλεγμένες σε διάστημα τεσσεράμισι μηνών ανά δέκα λεπτά. Οι συνθήκες θερμοκρασίας και υγρασίας του σπιτιού παρακολουθούνταν με ένα ασύρματο δίκτυο αισθητήρων ZigBee. Κάθε ασύρματος κόμβος μετέδιδε τις συνθήκες θερμοκρασίας και υγρασίας περίπου τριάμισι λεπτά. Στη συνέχεια, τα ασύρματα δεδομένα υπολογίστηκαν κατά μέσο όρο για περιόδους δέκα λεπτών. Τα ενεργειακά δεδομένα καταγράφονταν κάθε δέκα λεπτά με μετρητές ενέργειας m-bus. Ο καιρός από τον πλησιέστερο μετεωρολογικό σταθμό αεροδρομίου (αεροδρόμιο Chievres, Βέλγιο) μεταφορτώθηκε από ένα δημόσιο σύνολο δεδομένων από την Reliable Prognosis (rp5. ru) και συγχωνεύθηκε με τα πειραματικά σύνολα δεδομένων χρησιμοποιώντας τη στήλη ημερομηνίας και ώρας.

Πιο αναλυτικά, οι επεξηγηματικές μεταβλητές είναι οι εξής:

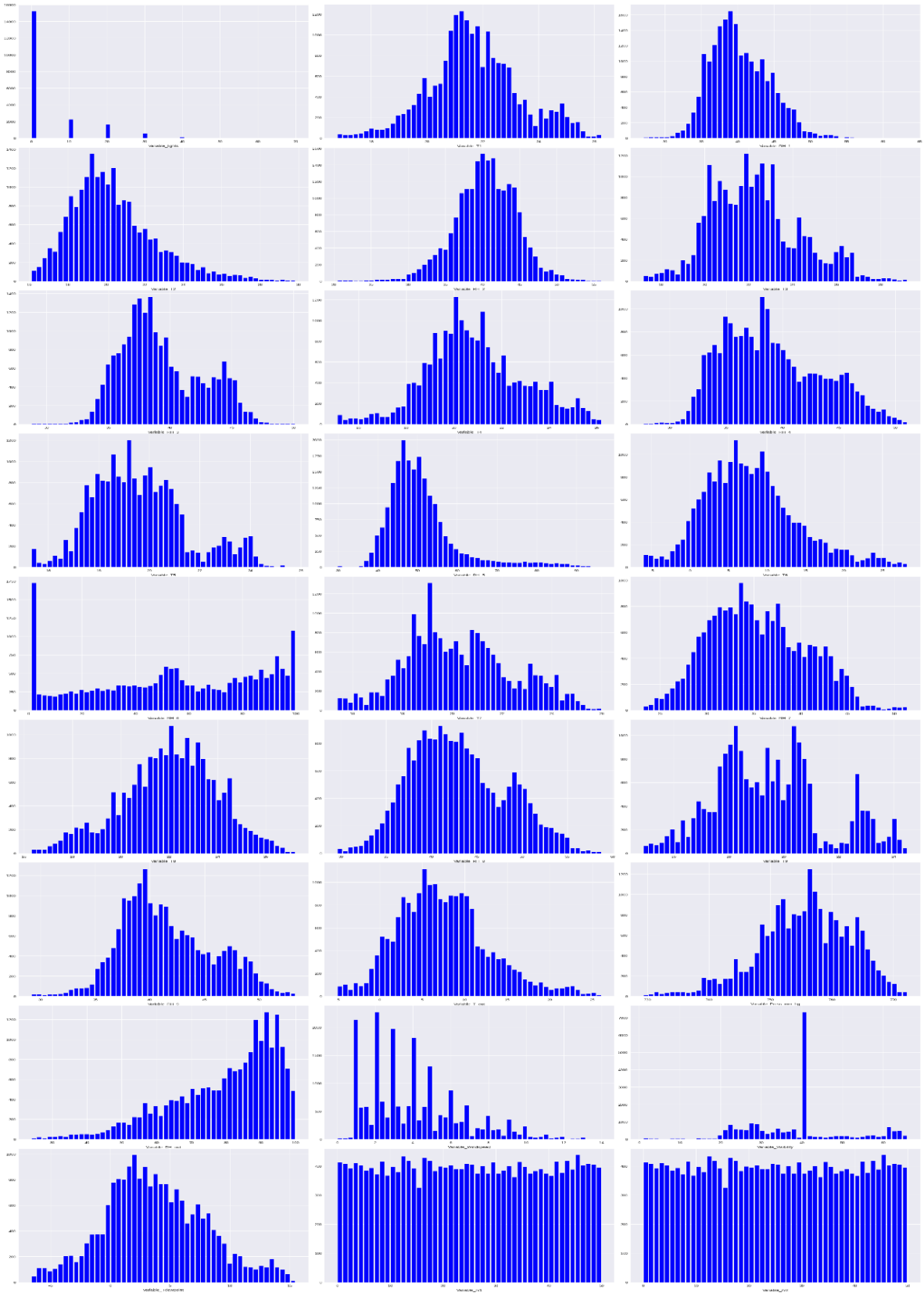
Μεταβλητή	Μονάδα Μέτρησης Μεταβλητής
Χρήση ενέργειας των φωτιστικών σωμάτων	Wh
Θερμοκρασία στον χώρο της κουζίνας	^{O}C
Υγρασία στον χώρο της κουζίνας	%
Θερμοκρασία στον χώρο του καθιστικού	^{O}C
Υγρασία στον χώρο του καθιστικού	%
Θερμοκρασία στον χώρο του πλυντηρίου	^{O}C
Υγρασία στον χώρο του πλυντηρίου	%
Θερμοκρασία στον χώρο του γραφείου	^{O}C
Υγρασία στον χώρο του γραφείου	%
Θερμοκρασία στο μπάνιο	^{O}C
Υγρασία στο μπάνιο επί τοις εκατό	%
Θερμοκρασία έξω από το κτίριο (βόρεια πλευρά)	^{O}C
Υγρασία έξω από το κτίριο (βόρεια πλευρά)	%
Θερμοκρασία στο δωμάτιο σιδερώματος	^{O}C
Υγρασία στο δωμάτιο σιδερώματος	%
Θερμοκρασία στο δωμάτιο των εφήβων	^{O}C
Υγρασία στο δωμάτιο των εφήβων	%
Θερμοκρασία στο δωμάτιο των γονέων	^{O}C
Υγρασία στο δωμάτιο των γονέων	%
Θερμοκρασία έξω (από τον μετεωρολογικό σταθμό)	^{O}C
Πίεση (στον μετεωρολογικό σταθμό)	mm Hg
Υγρασία έξω (από τον μετεωρολογικό σταθμό)	%
Ταχύτητα του ανέμου (στον μετεωρολογικό σταθμό)	m/s
Ορατότητα (στον μετεωρολογικό σταθμό)	km
Σημείο υγροποίησης (στον μετεωρολογικό σταθμό)	Â°C
Τυχαία μεταβλητή 1	αδιάστατη
Τυχαία μεταβλητή 2	αδιάστατη

Πίνακας 1: Επεξηγηματικές Μεταβλητές

Η εξαρτώμενη μεταβλητή, την οποία καλούμαστε να προβλέψουμε, είναι η συνολική κατανάλωση ενέργειας των οικιακών συσκευών σε Wh.

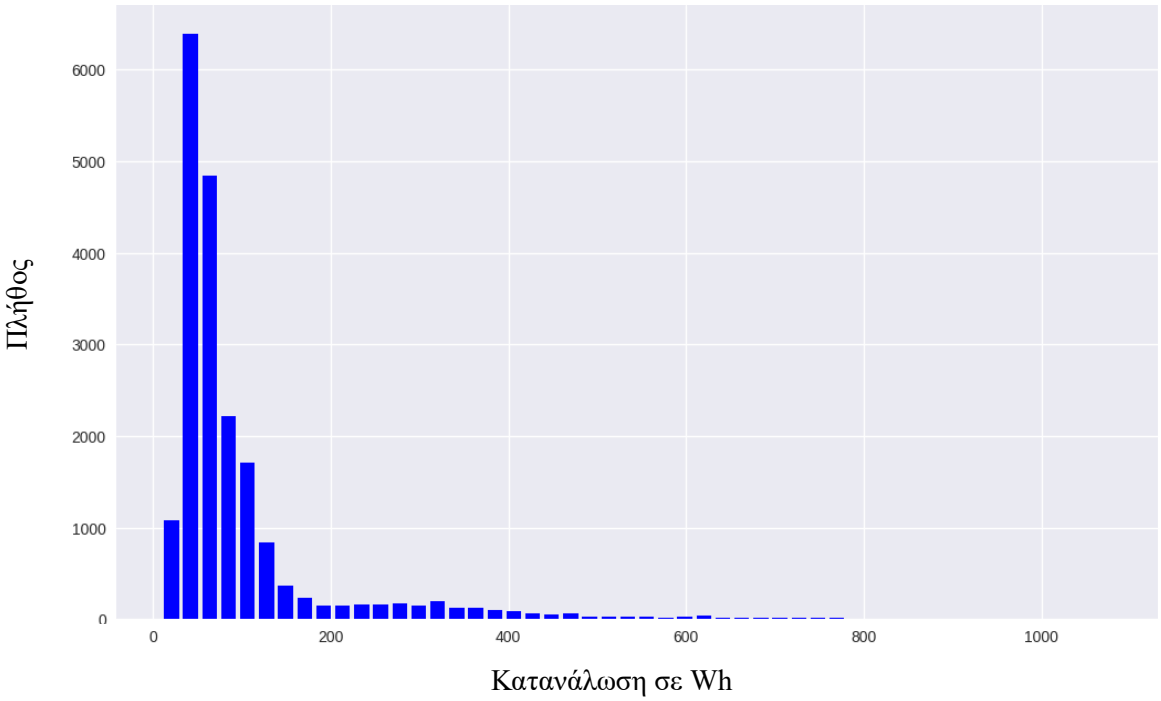
Το σύνολο δεδομένων προέρχεται από την ιστοσελίδα kaggle.com και ο ακριβής σύνδεσμος είναι https://www.kaggle.com/datasets/loveall/appliances-energy-prediction . Έχουμε στη διάθεση μας 19735 γεγονότα. Συνεπώς, το αρχείο περιέχει 19735 γραμμές και 29 στήλες εκ των οποίων οι 28 είναι οι επεξηγηματικές μεταβλητές που χαρακτηρίζουν το κάθε γεγονός και η μια εξαρτώμενη μεταβλητή.

Έπειτα οπτικοποιούμε τα δεδομένα με τη βοήθεια ιστογραμμάτων (Σχήμα 1), ώστε να έχουμε καλύτερη εποπτεία. Η σειρά, με την οποία εμφανίζονται τα ιστογράμματα συνάδει με τη σειρά των επεξηγηματικών μεταβλητών στον Πίνακα 1.

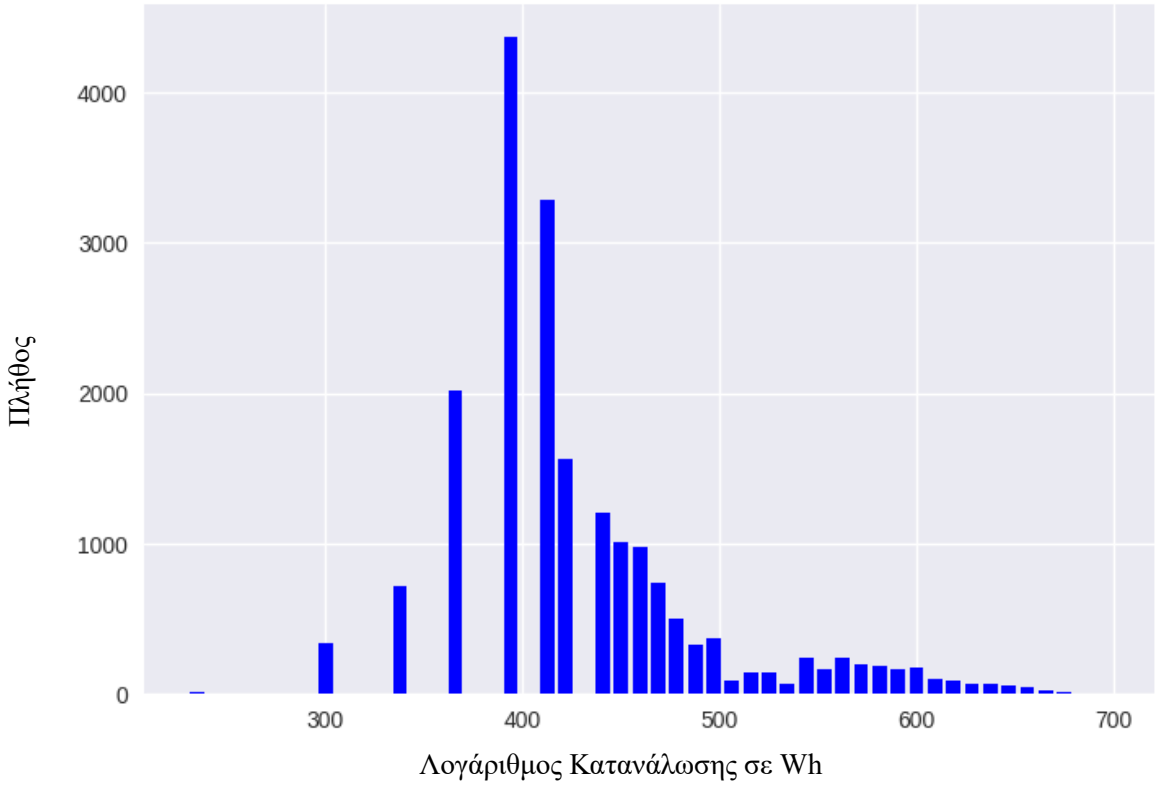


Σχήμα 1: Ιστογράμματα Χαρακτηριστικών

Στο Σχήμα 2 και 3 ασχολούμαστε με την εξαρτημένη μεταβλητή. Πιο συγκεκριμένα, στο Σχήμα 2 παρουσιάζεται το ιστόγραμμα της εξαρτημένης μεταβλητής κατανάλωσης (Appliances). Παρατηρώντας το μπορεί κάποιος να ισχυριστεί ότι θα ήταν καλή ιδέα να τη μετατρέψουμε σε λογαριθμική κλίμακα λόγω του δυναμικού εύρους της (dynamic range), όπως και κάνουμε στο Σχήμα 3.



Σχήμα 2: Ιστόγραμμα Συνολικής Κατανάλωσης



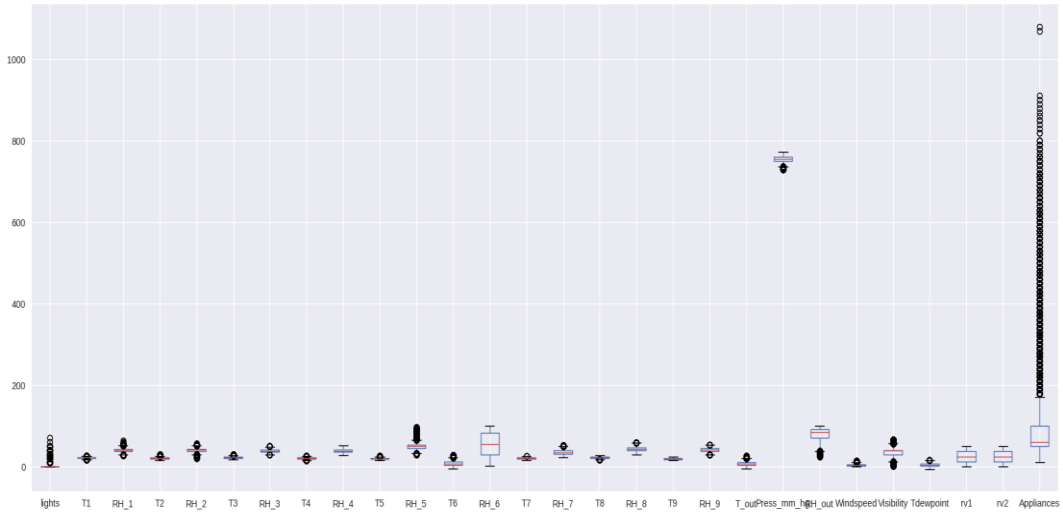
Σχήμα 3: Ιστόγραμμα Συνολικής Κατανάλωσης σε Λογαριθμική Κλίμακα

Ελέγχουμε αν λείπουν τιμές από τα δεδομένα που στην περίπτωση μας, πράγματι, δεν λείπουν. Ωστόσο, στην πράξη, ορισμένες τιμές μπορεί να λείπουν από κάποια διανύσματα χαρακτηριστικών. Τέτοιες περιπτώσεις εμφανίζονται συνήθως στις κοινωνικές επιστήμες, λόγω ελλιπών απαντήσεων κατά τη συμπλήρωση ερωτηματολογίων, ή την τηλεσκόπηση (remote sensing), στην περίπτωση που ορισμένες περιοχές καλύπτονται από ένα υποσύνολο αισθητήρων.

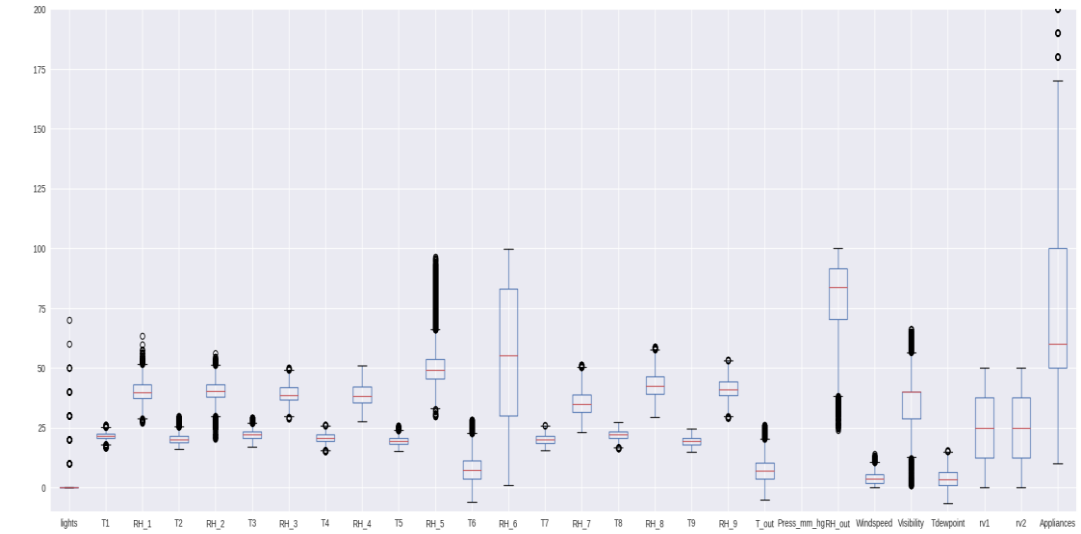
Κλασικούς τρόπους αντιμετώπισης του προβλήματος των ελλιπών τιμών (missing values) αποτελεί η «συμπλήρωσή» τους με μηδενικές τιμές, με τη μέση τιμή του χαρακτηριστικού με βάση τις υπόλοιπες μετρήσεις ή με μια τυχαία τιμή, αν γνωρίζουμε την πυκνότητα πιθανότητας της μεταβλητής. Η συμπλήρωση των μη διαθέσιμων τιμών σε ένα σύνολο δεδομένων είναι γνωστή και ως απόδοση τιμών (imputation). Εάν τα δεδομένα είναι πολλά, μέθοδο αντιμετώπισης του προβλήματος αποτελεί και η απόρριψη των ελλιπών διανυσμάτων χαρακτηριστικών, αλλά αυτό δεν είναι πάντα εφικτό, αφού τα δεδομένα είναι πολύτιμα.

Ελέγχοντας τα θηκογράμματα (Σχήμα 4 και 5) για ακραίες τιμές, διαπιστώνουμε ότι δεν υπάρχουν. Επομένως, μπορεί η μελέτη να συνεχιστεί χωρίς να χρειάζεται να αποκόψουμε παρατηρήσεις. Ένα σημείο θεωρείται ότι συνιστά ακραία τιμή (outlier) όταν βρίσκεται πολύ μακριά από την μέση τιμή της αντίστοιχης τυχαίας μεταβλητής. Η απόσταση αυτή μετριέται ως προς κάποια δοθείσα τιμή κατωφλίου που είναι συνήθως ίση με ένα πολλαπλάσιο της τυπικής απόκλισης. Για τυχαία μεταβλητή που ακολουθεί την κανονική κατανομή, απόσταση ίση με δύο φορές την τυπική απόκλιση

καλύπτει το 95% των σημείων και απόσταση ίση με τρεις φορές την τυπική απόκλιση καλύπτει το 99% των σημείων. Μετρήσεις με τιμές πολύ διαφορετικές από την μέση τιμή, επιφέρουν μεγάλα σφάλματα κατά την εκπαίδευση και μπορούν να προκαλέσουν προβλήματα, καθώς συνήθως οι συναρτήσεις κόστους βασίζονται στην τετραγωνική απόσταση. Οι συνέπειες αυτές είναι ακόμα χειρότερες όταν οι ακραίες τιμές είναι αποτέλεσμα ενθόρυβων (noisy) μετρήσεων.



Σχήμα 4: Θηκογράμματα Χαρακτηριστικών



Σχήμα 5: Μεγεθυμένα Θηκογράμματα Χαρακτηριστικών

Κατασκευάζουμε το διάγραμμα του πίνακα συνδιασποράς (Σχήμα 6) και τον πίνακα συσχέτισης (Σχήμα 7). Παρατηρούμε ότι δεν υπάρχει κάποιο ιδιαίτερο πρόβλημα, αλλά καλό θα ήταν να είμαστε επιφυλακτικοί όσον αφορά στις μεταβλητές τυχαία μεταβλητή 1 και τυχαία μεταβλητή 2. Αυτό εξ' αιτίας τόσο των κατανομών τους, όσο και την έλλειψη πληροφορίας σχετική με τη φύση των μεταβλητών αυτών.

Η συνδιασπορά δύο τυχαίων μεταβλητών X και Y συμβολίζεται ως cov(X,Y) και ορίζεται ως:

$$cov(X,Y) = E[(X - E[X]) \cdot (Y - E[Y])]$$

ή

$$cov(X,Y) = E[X \cdot Y] - E[X] \cdot E[Y]$$

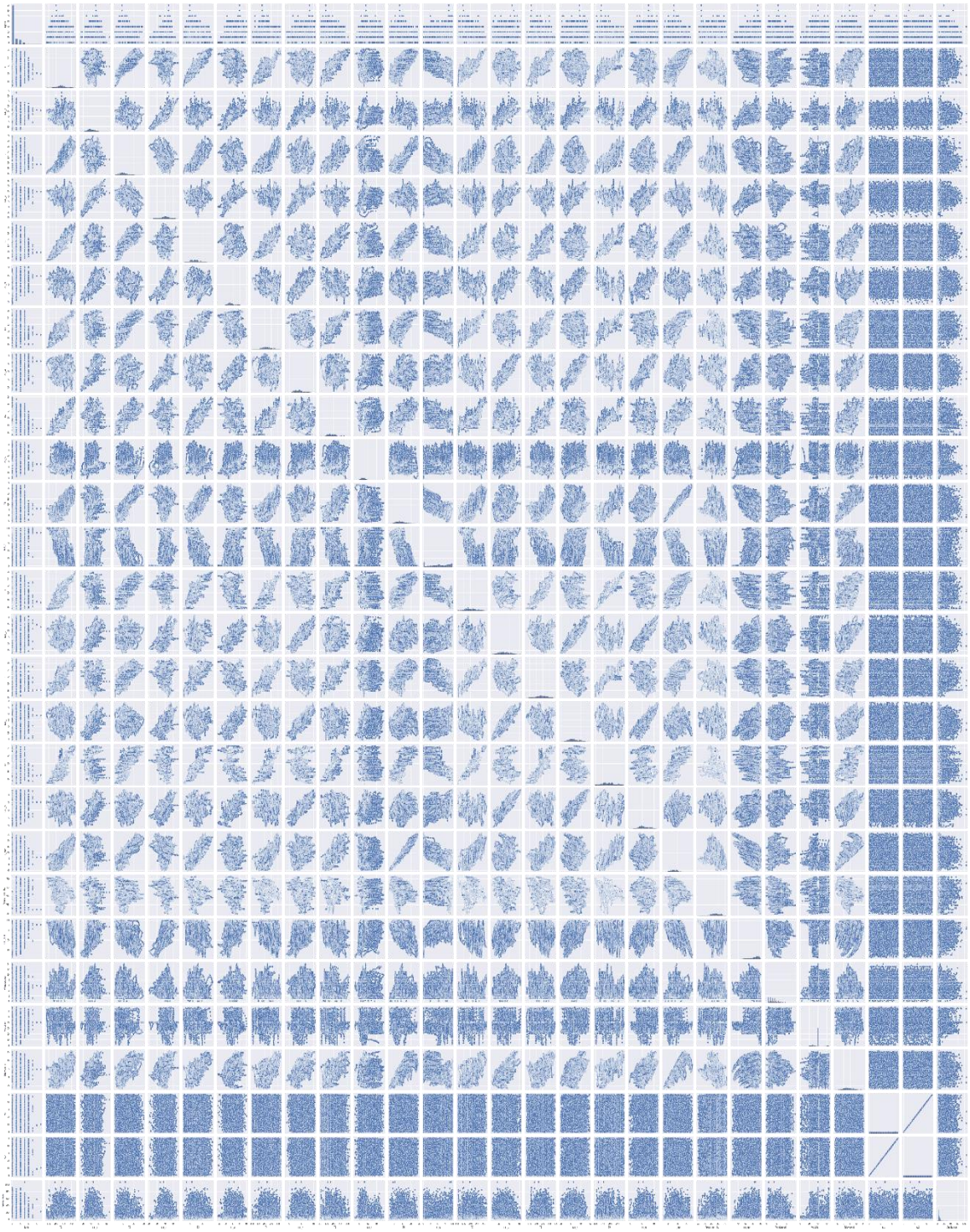
Ο πίνακας συσχέτισης R είναι ένας τετραγωνικός πίνακας που δείχνει τις συσχετίσεις μεταξύ των διάφορων μεταβλητών. Τα διαγώνια στοιχεία του είναι ίσα με 1. Ο πίνακας συσχέτισης των n τυχαίων μεταβλητών $X_1,\ldots,X_n$ είναι ο n×n συμμετρικός πίνακας, του οποίου τα στοιχεία είναι $r_{xy}=corr(X_i , X_j)$. Τα στοιχεία αυτά παίρνουν τιμές από -1 έως 1, 1 σε περίπτωση μίας τέλειας άμεσης (αύξουσας) γραμμικής συσχέτισης, -1 σε περίπτωση μίας τέλειας φθίνουσας (αντίστροφης) γραμμικής συσχέτισης (αντισυσχέτιση) και 0 όταν η τυχαίες μεταβλητές είναι ασυσχέτιστες. Ο δειγματικός συντελεστής συσχέτισης γράφεται ως εξής:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n s_x s_y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

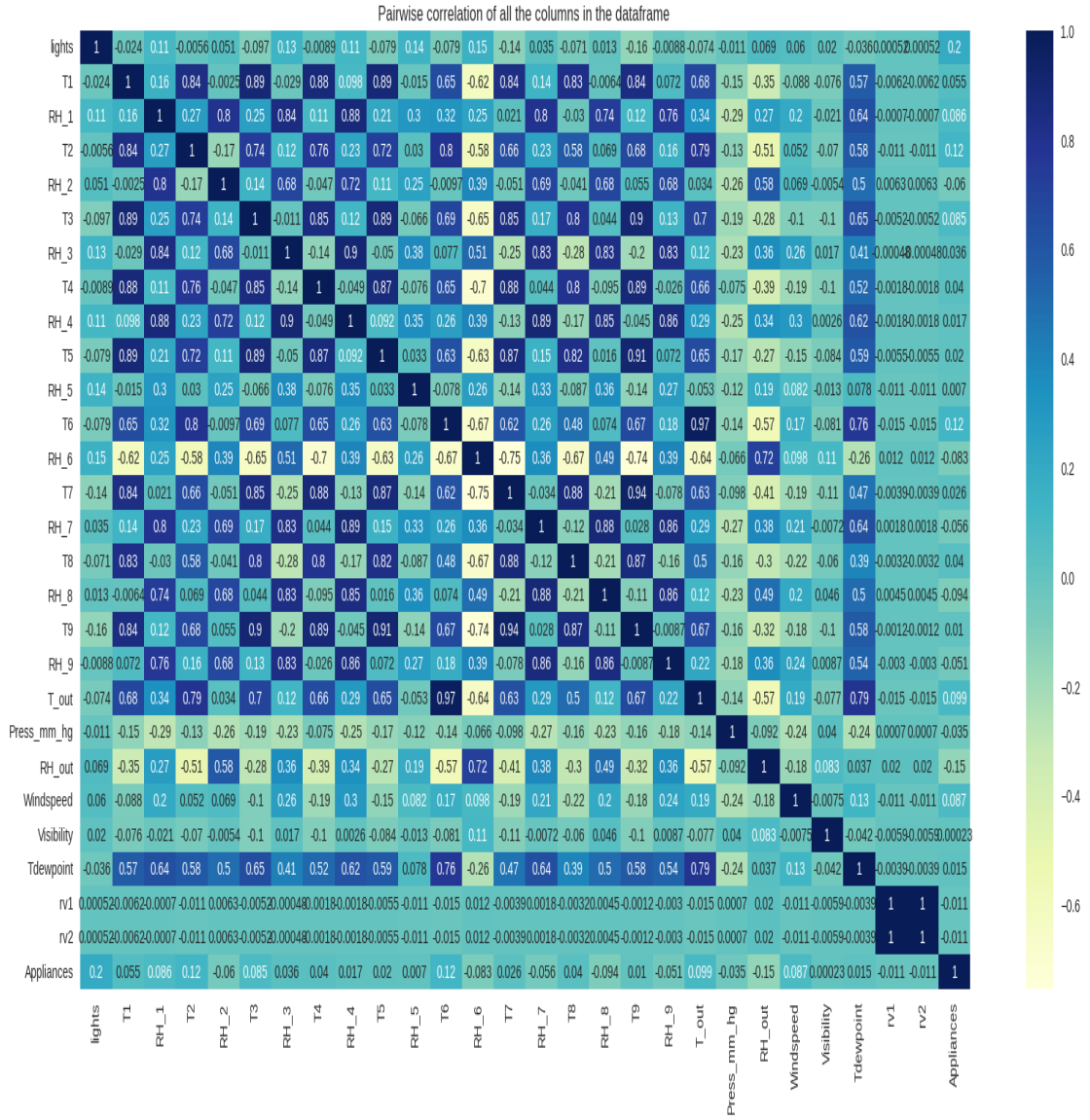
όπου x και y είναι ο δειγματικός μέσος των X και Y και s_x και s_y είναι οι δειγματικές τυπικές αποκλίσεις των X και Y.

Επίσης ισχύει η εξής σχέση:

$$corr(X,Y) = \frac{cov(X,Y)}{s_x s_y}$$



Σχήμα 6: Scatter Matrix



Σχήμα 7: Πίνακας συσχέτισης

4 Επιλογή Χαρακτηριστικών

Είναι γεγονός ότι σε πολλά προβλήματα Μηχανικής Μάθησης το πλήθος των χαρακτηριστικών μεταβλητών είναι μεγάλο και για αυτό το πρόβλημα θεωρείται περίπλοκο και υπολογιστικά δυσβάστακτο (κατάρα διαστατικότητας). Καλούμαστε, λοιπόν, να αξιολογήσουμε ποιες από τις μεταβλητές είναι πραγματικά σημαντικές στην προσαρμογή του μοντέλου παλινδρόμησης, και αν αυτό είναι εφικτό να μειώσουμε τη διάσταση του προβλήματος.

Ένας ακόμη λόγος είναι ότι, παρόλο που δύο χαρακτηριστικά μπορούν να περιγράψουν μεγαλύτερο μέρος του δείγματος, εάν αφαιρέσουμε ένα από τα δύο, το κόστος θα είναι πολύ μικρό, ενώ η πολυπλοκότητα ίσως σημαντικά μικρότερη.

Η επιλογή μεταβλητών γίνεται με βάση το ποιες περιγράφουν μεγαλύτερο μέρος της διασποράς του δείγματος, αφού τα μετασχηματίσουμε σε γραμμικά ανεξάρτητα μεταξύ τους, ώστε να αποφύγουμε φαινόμενα πολυσυγγραμμικότητας (υψηλή συσχέτιση μεταξύ των στηλών). Επιθυμούμε, δηλαδή να κρατήσουμε αυτές που μας δίνουν την περισσότερη πληροφορία στο δείγμα μας.

Επίσης, όσο μεγαλύτερος είναι ο λόγος του αριθμού των προτύπων εκπαίδευσης προς τον αριθμό των ελεύθερων παραμέτρων του μοντέλου, τόσο μεγαλύτερη είναι η ικανότητα γενίκευσης του προκύπτοντος μοντέλου. Ένας μεγάλος αριθμός χαρακτηριστικών μεταφράζεται άμεσα σε μεγάλο αριθμό παραμέτρων (παραδείγματος χάρη βάρη συνάψεων σε Νευρωνικό Δίκτυο, βάρη σε ένα γραμμικό μοντέλο κ.λ.π.).

Δεν αρκεί να σχεδιάσουμε ένα σύστημα παλινδρόμησης, αλλά είναι επιπλέον αναγκαίο να εκτιμήσουμε και την απόδοσή του. Να σημειωθεί ότι η εκτίμηση του σφάλματος βελτιώνεται καθώς ο λόγος N/l αυξάνει, όπου N το πλήθος των παρατηρήσεων και l το πλήθος των χαρακτηριστικών.

Απαραίτητη σε αυτό το σημείο είναι η κλιμάκωση (scaling) των χαρακτηριστικών προκειμένου να εξασφαλιστεί το συγκρίσιμο του εύρους των αντίστοιχων τιμών τους. Πολύ συχνά, τα χαρακτηριστικά λαμβάνουν τιμές σε διαφορετικά διαστήματα. Σε τέτοιες περιπτώσεις, χαρακτηριστικά με μεγάλες τιμές μπορεί να έχουν μεγαλύτερη επίδραση στη συνάρτηση κόστους, σε σύγκριση με χαρακτηριστικά που λαμβάνουν μικρές τιμές, μολονότι αυτό δεν αντικατοπτρίζει κατ' ανάγκη τη σημαντικότητα του καθενός στη σχεδίαση του μοντέλου παλινδρόμησης.

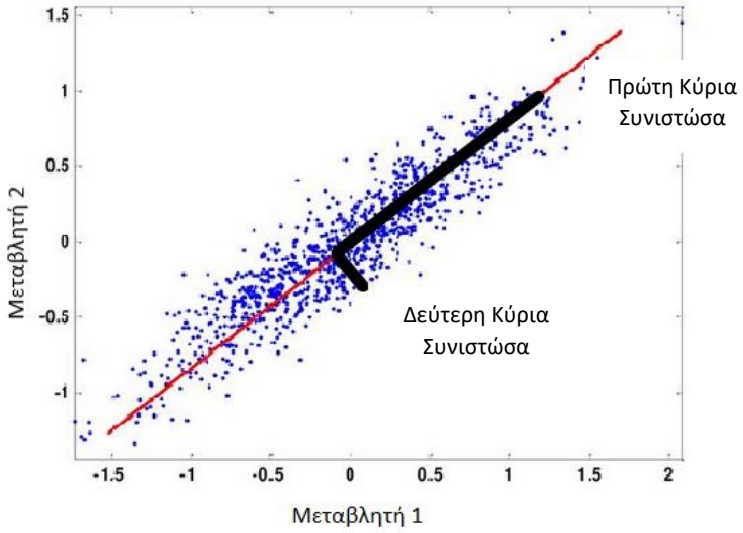
Μπορούμε να αντιμετωπίσουμε το πρόβλημα αυτό κανονικοποιώντας τα χαρακτηριστικά, έτσι ώστε να λαμβάνουν τιμές σε παρόμοια πεδία. Μια προφανής τεχνική είναι η κανονικοποίηση μέσω των αντίστοιχων εκτιμήσεων μέσης τιμής και διασποράς. Για διαθέσιμα δεδομένα του k-οστού χαρακτηριστικού έχουμε:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_{ik}, k = 1, 2, \ldots, l$$

$$\sigma_k^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_{ik} - \bar{x}_k)^2$$

$$\hat{x} = \frac{x_{ik} - \bar{x}_k}{\sigma_\kappa}$$

Ανάμεσα σε πολλές μεθόδους μείωσης διαστάσεων που χρησιμοποιούνται στη Μηχανική Μάθηση διαλέγουμε αυτή της Ανάλυσης Κύριων Συνιστωσών (PCA). Η PCA είναι μια μη επιβλεπόμενη (unsupervised) μέθοδος που κατά κύριο λόγο χρησιμοποιείται για μείωση διαστάσεως ενός προβλήματος. Τα αποτελέσματα της PCA δίνουν μια μικρών διαστάσεων εικόνα της δομής των δεδομένων και τους κυρίαρχους μη σχετιζόμενους παράγοντες που καθορίζουν τη διασπορά των δεδομένων. Στόχος είναι να βρούμε τις διευθύνσεις (Principal Components) εκείνες κατά μήκος των οποίων η διασπορά του συνόλου είναι μεγαλύτερη. Ο αριθμός των κυρίων συνιστωσών αποτελεί εσωτερική διάσταση του συνόλου (intrinsic dimension). Ουσιαστικά είναι ένα πρόβλημα εύρεσης ιδιοτιμών και ιδιοδιανυσμάτων. Για παράδειγμα, το παρακάτω σχήμα αναφέρεται σε ένα πρόβλημα δύο κύριων συνιστωσών.



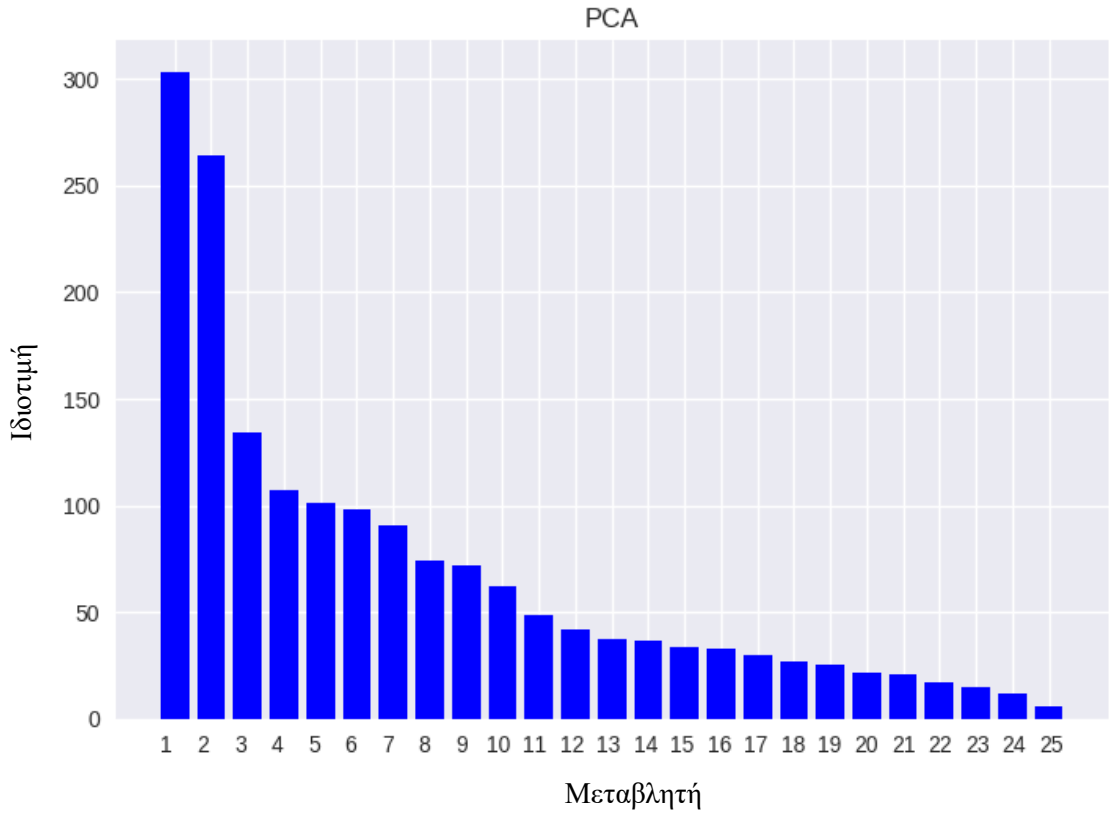
Σχήμα 8: Γράφημα PCA

Ο αλγόριθμος PCA λειτουργεί ως εξής:

1) Υπολογίζουμε τη μέση τιμή και τη διασπορά κάθε μεταβλητής.
2) Υπολογίζουμε τον d×d πίνακα διασποράς X^TX των τυποποιημένων μεταβλητών $x \to \frac{x-\mu}{\sigma}$.
3) Υπολογίζουμε τις ιδιοτιμές λ_i , όπου $i \leq d$, και τα αντίστοιχα ιδιοδιανύσματα e_i του πίνακα διασποράς.

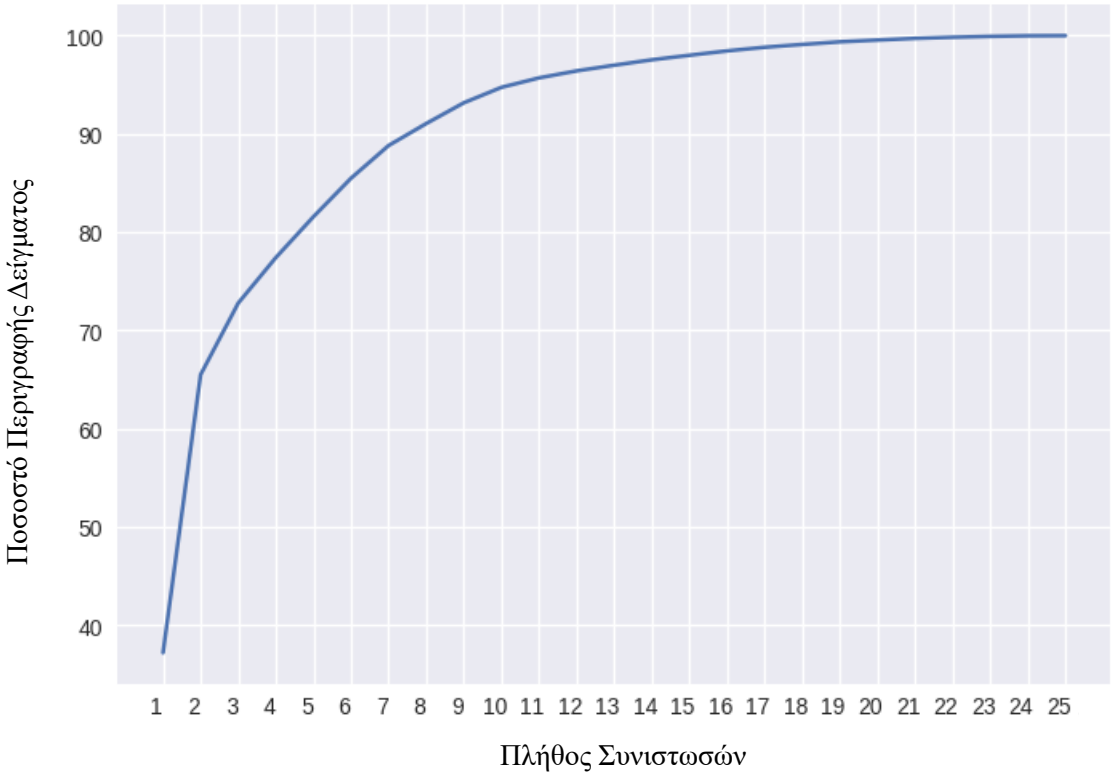
4) Κατατάσσουμε τις ιδιοτιμές σε φθίνουσα σειρά. Να σημειωθεί εδώ ότι ο πίνακας είναι θετικά ημιορισμένος, ως πίνακας διασποράς. Τα πρώτα m ιδιοδιανύσματα συνιστούν τις κύριες διευθύνσεις με τη μεγαλύτερη διασπορά.
5) Κατασκευάζουμε τον πίνακα $A_{d\times m}$ με στήλες τα m κανονικοποιημένα διανύσματα e_i.
6) Ο μετασχηματισμός $y_{m\times 1}= A^T_{d\times m}\, x_{d\times 1}$ προβάλλει το σύνολο x στον χώρο διάστασης m των κύριων κατευθύνσεων, όπου οι νέες μεταβλητές είναι γραμμικώς ανεξάρτητες.

Παρουσιάζουμε τα αποτελέσματα για το πρόβλημα μας:



Σχήμα 9: Ιδιοτιμές Πίνακα Συνδιασποράς

Στο πρόβλημα επιλέγουμε να κρατήσουμε τις πρώτες 8 μεταβλητές, οι οποίες περιγράφουν το 91,05% της διασποράς του δείγματος. Ο λόγος που δεν επιλέγουμε λιγότερες π.χ. 4 είναι επειδή ο χρόνος προσαρμογής του μοντέλου δεν αλλάζει σημαντικά για αντισταθμίσει το 13% της ανάλυσης διασποράς.



Σχήμα 10: Ποσοστό περιγραφής δείγματος σε σχέση με το πλήθος μεταβλητών

5 Πολλαπλή Γραμμική Παλινδρόμηση

Στην πραγματικότητα, το γραμμικό μοντέλο δεν είναι το πιο αποτελεσματικό πάντα. Ωστόσο, χρησιμοποιείται πολύ συχνά, λόγω της απλότητας του. Το μοντέλο πολλαπλής γραμμικής παλινδρόμησης αποτελεί επέκταση του απλού γραμμικού μοντέλου, στην οποία λαμβάνεται υπόψιν η ταυτόχρονη ύπαρξη περισσότερων από μίας επεξηγηματικών μεταβλητών, έστω p το πλήθος. Κατά την εκπαίδευση του γραμμικού μοντέλου υπολογίζονται τα βάρη (weights) b_i και ο σταθερός όρος (stable coefficient) b_0 από τα δεδομένα εκπαίδευσης.

Σε κάθε πρόβλημα p διαστάσεων ορίζουμε μια επιπλέον μεταβλητή $x_0=1$, η οποία είναι σταθερή και ίση με 1, ώστε ο σταθερός όρος b_0 να ενσωματωθεί στο δίανυσμα των βαρών.

Το γενικό γραμμικό μοντέλο δίνεται από τη σχέση $y_i=b_o+b_1x_{i1}+b_2x_{i2}+\ldots+b_px_{ip}+\varepsilon_i$, όπου

- y_i : οι τιμές των παρατηρήσεων της εξαρτημένης μεταβλητής y
- x_{ij} : οι τιμές για την i-οστή παρατήρηση των επεξηγηματικών μεταβλητών x_j (i=1,2,…,n, j=1,2, … , p)
- $b_0,b_1,\ldots,b_p$: οι άγνωστες παράμετροι του μοντέλου
- ε_i : τα τυχαία σφάλματα, τα οποία υποθέτουμε ότι είναι τυχαίες μεταβλητές που ακολουθούν κανονική κατανομή $N(0,\sigma^2)$ και ότι είναι ανεξάρτητα μεταξύ τους.

Η εν λόγω σχέση μπορεί να γραφτεί με τη βοήθεια πινάκων ως εξής: $\mathbf{y}=\mathbf{X}b+\boldsymbol{\varepsilon}$, όπου

$$y=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X=\begin{bmatrix} 1 & X_{11} & X_{12} & \ldots & X_{1p} \\ 1 & X_{21} & X_{22} & \ldots & X_{2p} \\ \vdots & \vdots & \ddots & & \vdots \\ 1 & X_{n1} & X_{n2} & \ldots & X_{np} \end{bmatrix}, b=\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{bmatrix}, \varepsilon=\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Ο υπολογισμός γίνεται σε βήματα (iteratively), όπου σε κάθε βήμα διορθόνονται οι τιμές των βαρών λαμβάνοντας υπόψιν τα διανύσματα εκπαίδευσης. Αυτή η διόρθωση καθορίζεται από μια συνάρτηση κόστους (cost function), η οποία χαρακτηρίζει κάθε αλγόριθμο.

Η μέθοδος ελαχίστων τετραγώνων για την εκτίμηση των παραμέτρων **b** βασίζεται στην ελαχιστοποίηση της παράστασης

$$J(b)=\sum_{i=1}^{n}(y_i-x_i^Tb)^2$$

Οπότε παραγωγίζοντας ως προς **b** έχουμε:

$$\frac{\partial J}{\partial b}=\sum_{i=1}^{n}x_i(y_i-x_i^T\hat{b})$$

και έτσι θέτωντας την παραπάνω σχέση ίση με το μηδέν, καταλήγουμε στη σχέση

$$\left(\sum_{i=1}^{n} x_i x_i^T\right)\hat{b} = \sum_{i=1}^{n} x_i y_i$$

.

Επομένως:

$$\mathbf{(X^T X)\,\hat{b} = X^T y}$$

Αν ο πίνακας $\mathbf{X^T X}$ αντιστρέφεται, τότε η εκτιμήτρια ελαχίστων τετραγώνων (OLS) του διανύσματος **b** δίνεται από τη σχέση

$$\mathbf{\hat{b} = (X^T X)^{-1} (X^T y)}\ ,$$

Οπότε και το μοντέλο που εκτιμούμε δίνεται από τη σχέση

$$\mathbf{\hat{y} = X\hat{b}}$$

Και τα υπόλοιπα (residuals) υπολογίζονται ως

$$\boldsymbol{\varepsilon = y - \hat{y}.}$$

Κάθε μια από τις $\hat{b}_j$ (j=1,2,…,p) εκφράζει την αναμενόμενη μεταβολή της y για μία μονάδα αύξησης ή μείωσης της αντίστοιχης επεξηγηματικής μεταβλητής x_j, δεδομένου ότι οι άλλες επεξηγηματικές μεταβλητές παραμένουν σταθερές. Η $\hat{b}_0$ εκφράζει την αναμενόμενη τιμή της y όταν όλες οι επεξηγηματικές μεταβλητές πάρουν τη τιμή μηδέν.

Το προσαρμοσμένο μοντέλο (fitted model) μπορεί να γρεφεί ως

$$\mathbf{\hat{y} = X\hat{b}}$$

$$\mathbf{= X(X^T X)^{-1} X^T y}$$

$$\mathbf{= Hy}\ ,$$

Όπου $\mathbf{H = X(X^T X)^{-1} X^T y}$ ο πίνακας προβολής, ο οποίος μάλιστα είναι συμμετρικός και ταυτοδύναμος.

Η βηματική (iterative) ελαχιστοποίηση της συνάρτησης κόστους και επομένως η εύρεση του βέλτιστου διανύσματος βαρών γίνεται με τη χρήση της μεθόδου απότομης καθόδου (steepest descent method):

$$b_k(t+1) = b_k(t) - \rho(t)\frac{\partial J(\vec{b})}{\partial b}$$

$$= b_k(t) - \rho(t) X^T [Xb(t) - y]$$

όπου ρ(t) μια αλληλουχία θετικών αριθμών.

Ο αλγόριθμος λειτουργεί ως εξής:

- Αρχικοποιούμε τυχαία τα b(0) και ρ(0).
- Υπολογίζουμε τους πίνακες X, X^T και y.

- Σε κάθε βήμα διορθώνουμε την τιμή του b(t) και ενδεχομένως προσαρμόζουμε το ρ(t).
- Όταν η συνάρτηση κόστους έχει πάρει την ελάχιστη τιμή ή όταν |b(t+1)-b(t)|<ε, δηλαδή εάν δεν αλλάζει σημαντικά το διάνυσμα b, τότε τερματίζουμε τη μέθοδο.

Παρακάτω θα αναλύσουμε τη μέθοδο Ho – Kashyap, η οποία αποτελεί επέκταση της μεθόδου ελαχίστων τετραγώνων που θεωρεί το Y μεταβλητό, με αποτέλεσμα να προσδιορίζεται και αυτό βηματικά, όπως και τα βάρη. Είναι προφανές ότι το υπολογιστικό κόστος εδώ είναι μεγαλύτερο, αλλά συγκλίνει πάντα.

Η συνάρτηση κόστους είναι:

$$J(b,Y)=\frac{1}{2}||Xb-Y||^2$$

Παραγωγίζοντας:

$$\frac{\partial J}{\partial b}=X^T(Xb-Y),\frac{\partial J}{\partial Y}=Y-Xb$$

Ελαχιστοποιώντας:

$$\frac{\partial J}{\partial b}=0\Rightarrow b=(X^TX)^{-1}X^TY$$

Η κεντρική ιδέα είναι να ακολοθήσουμε βήματα όπου (1) κρατάμε το Y σταθερό και ελαχιστοποιούμε το J ως προς b, (2) κρατάμε το b σταθερό και ελαχιστοποιούμε το J ως προς Y.

Για την πρώτη ελαχιστοποίηση προκύπτει ότι $b=(X^T X)^{-1}X^T Y$. Για τη δεύτερη ελαχιστοποίηση θα πρέπει επιπλέον να λάβουμε υπόψη τον περιορισμό ότι τα στοιχεία του Y δε θα πρέπει να αλλάζουν πρόσημο. Οπότε θα χρησιμοποιήσουμε τον τροποποιημένο κανόνα πτώσης:

$$Y(t+1)=Y(t)-\rho(t)[\frac{\partial J}{\partial Y}-|\frac{\partial J}{\partial Y}|]$$

$$=Y(t)+\rho(t)[e(t)+|e(t)|],e(t)=Xb(t)-Y(t)$$

Επομένως σε κάθε βήμα:

(1) $Y(t+1)=Y(t)+\rho(t)[e(t)+|e(t)|]$
(2) $b(t+1)=(X^T X)^{-1}X^T Y(t+1)$.

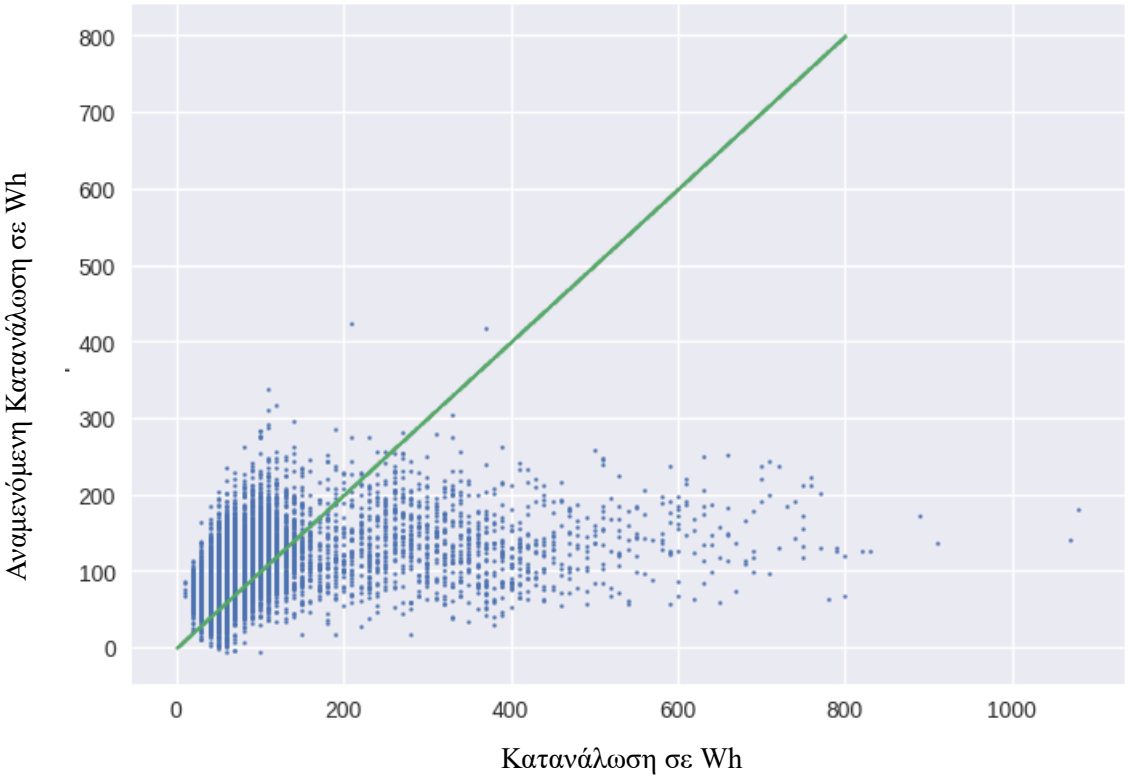
Εφαρμόζοντας το γραμμικό μοντέλο στο σύνολο εκπαίδευσης καλατήγουμε στο μοντέλο, το οποίο περιγράφεται ως εξής:

Τα βάρη για τις 25 μεταβλητές είναι:

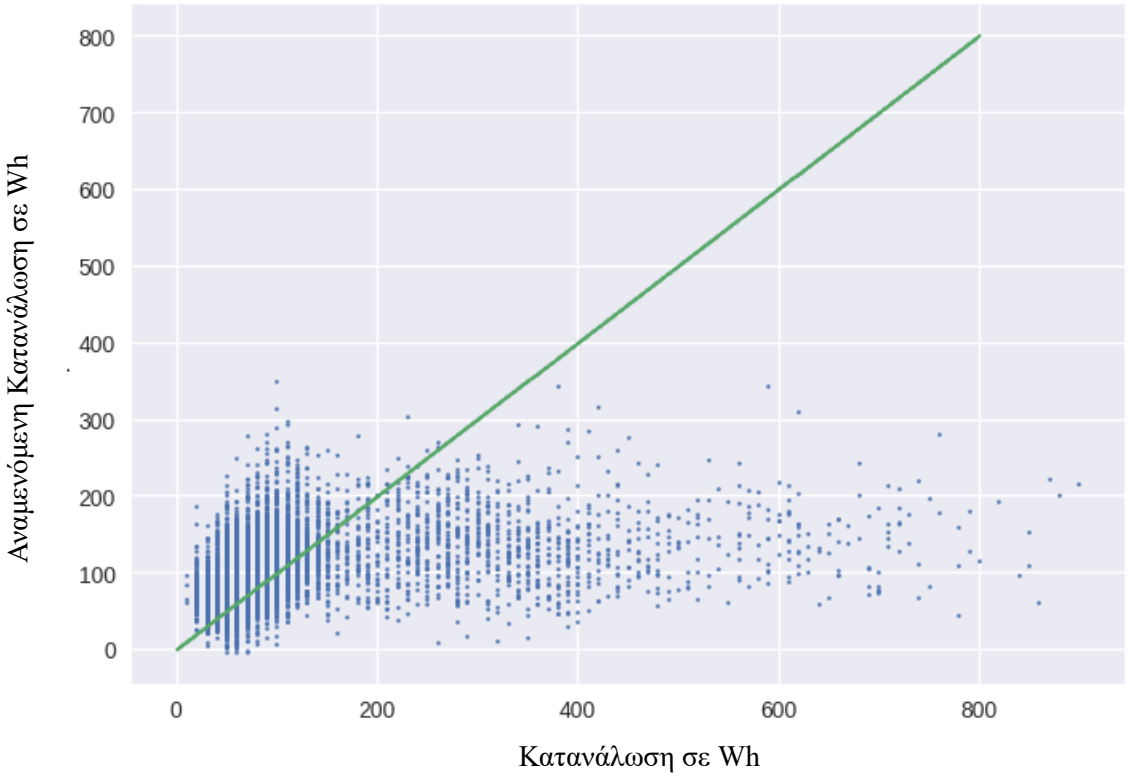
[15.89	-0.235	59.86	-35.76	-52.65
50.67	15.90	-9.44	-2.55	3.81
0.62	45.86	9.16	6.91	-8.04

14.94 -23.95 -37.63 -3.95 -63.99
1.59 -19.65 4.08 1.88 27.53]
και του σταθερού όρου:
15.89

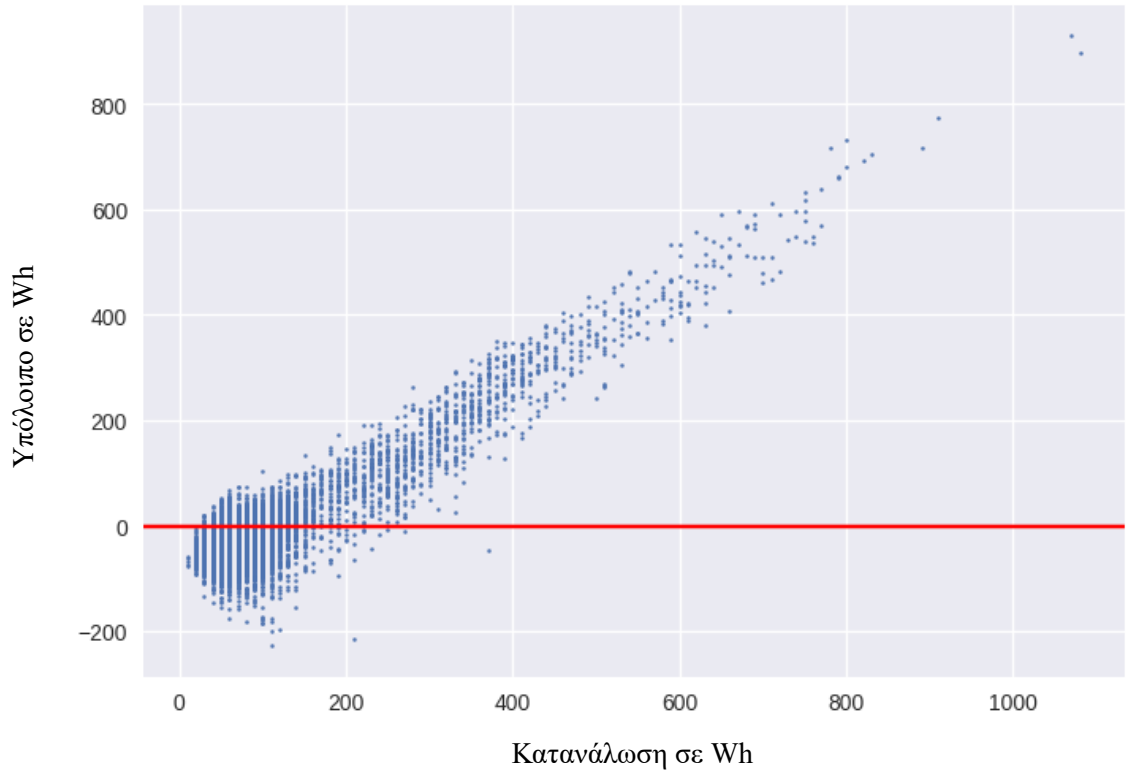
Στα παρακάτω σχήματα η πράσινη ευθεία (όπου η πραγματική τιμή ισούται με την αναμενόμενη τιμή) και η κόκκινη (όπου τα υπόλοιπα πραγματικής μείον αναμενόμενης τιμής είναι μηδενικά) είναι βοηθητικές οπτικά.



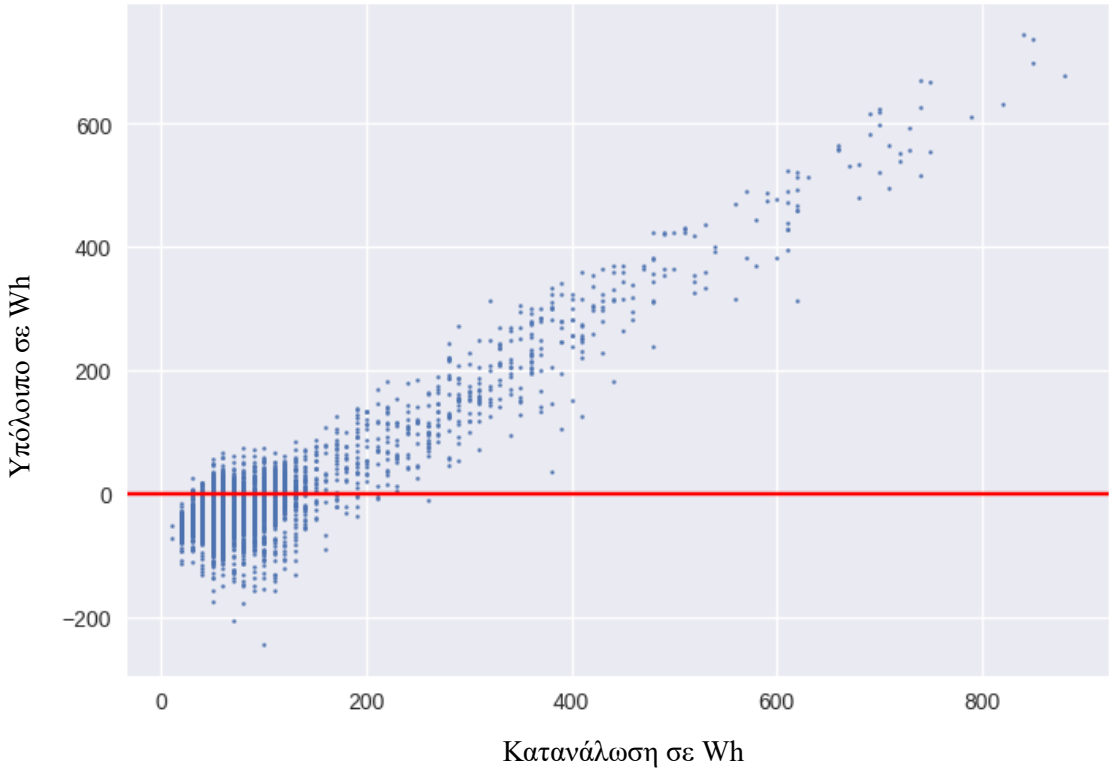
Σχήμα 11: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης



Σχήμα 12: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης



Σχήμα 13: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο εκπαίδευσης

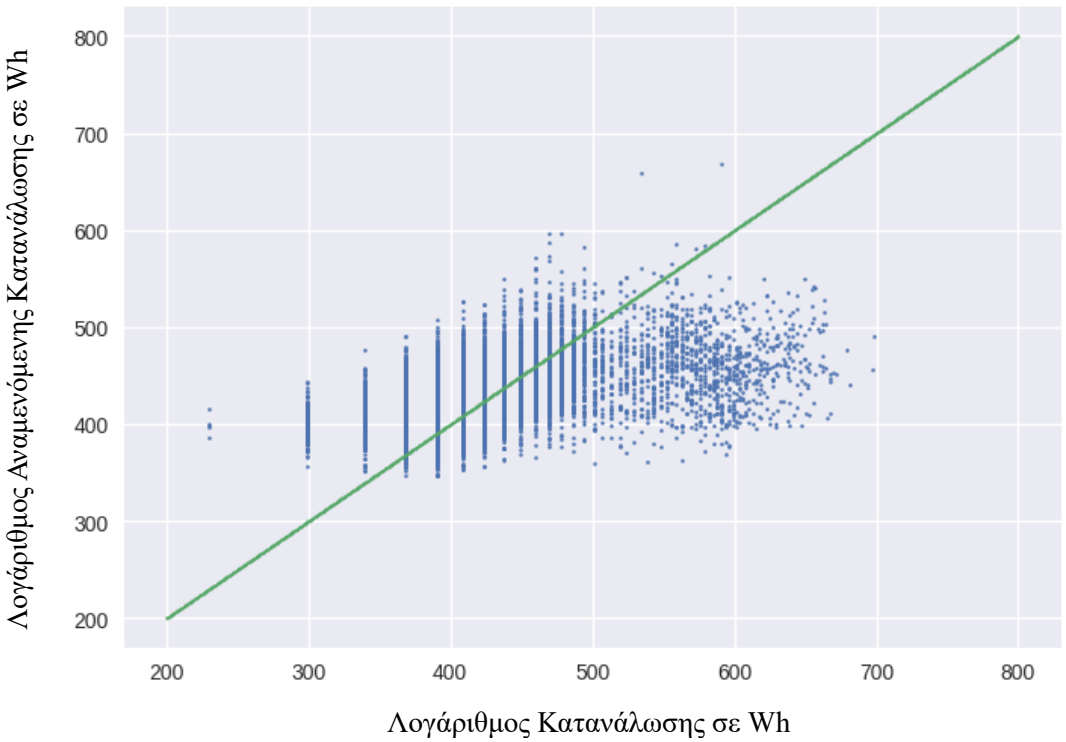


Σχήμα 14: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο αξιολόγησης

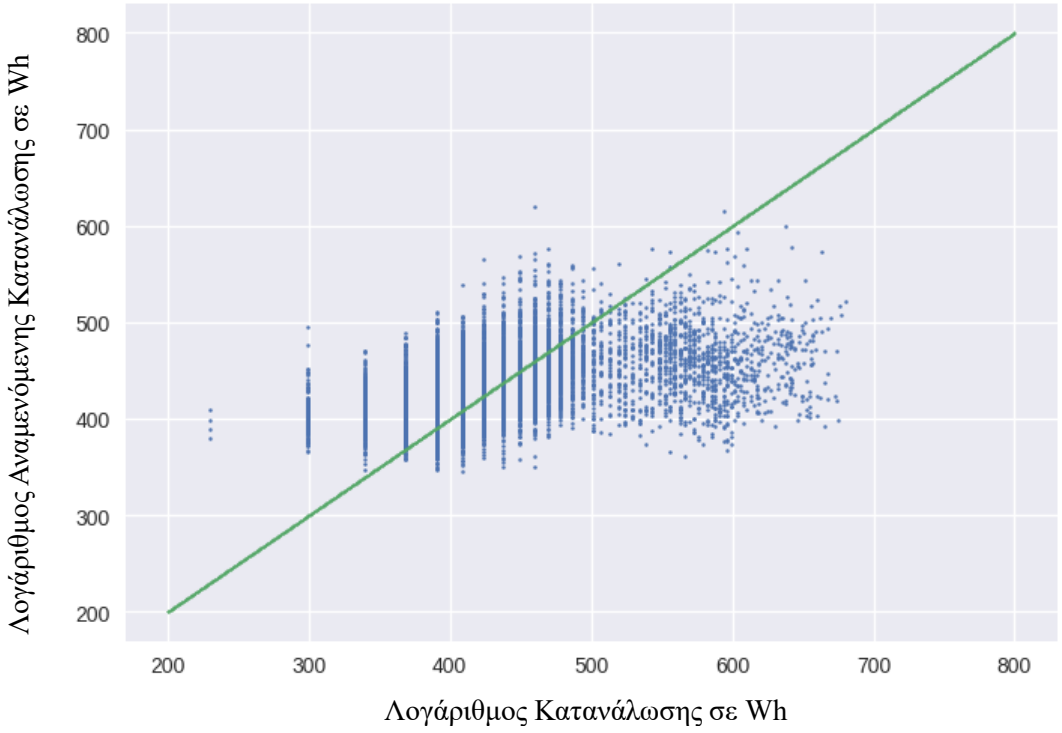
Παρατηρούμε ότι τα αποτελέσματα δεν είναι πολύ καλά, αλλά μας δίνουν μια γενική εικόνα. Όπως ειπώθηκε και στην αρχή, το δυναμικό εύρος (dynamic range) της εξαρτώμενης μεταβλητής (κατανάλωση) μας εγείρουν την περιέργεια να εξετάσουμε αντίστοιχα και για λογαριθμική κλίμακα. Αυτό το μοντέλο ονομάζεται λογαριθμοκανονική παλινδρόμηση (Lognormal Regression). Η λογαριθμοκανονική παλινδρόμηση βασίζεται στην υπόθεση ότι τα στοιχεία που περιέχονται στη βάση δεδομένων είναι μη αρνητικά, ο φυσικός λογάριθμος της ανεξάρτητης μεταβλητής ακολουθεί την κανονική κατανομή και ο αριθμητικός μέσος είναι σχετικά μεγάλος.

Εφαρμόζοντας το γραμμικό μοντέλο στο σύνολο εκπαίδευσης για λογαριθμική κλίμακα καλατήγουμε στο μοντέλο, το οποίο περιγράφεται ως εξής:

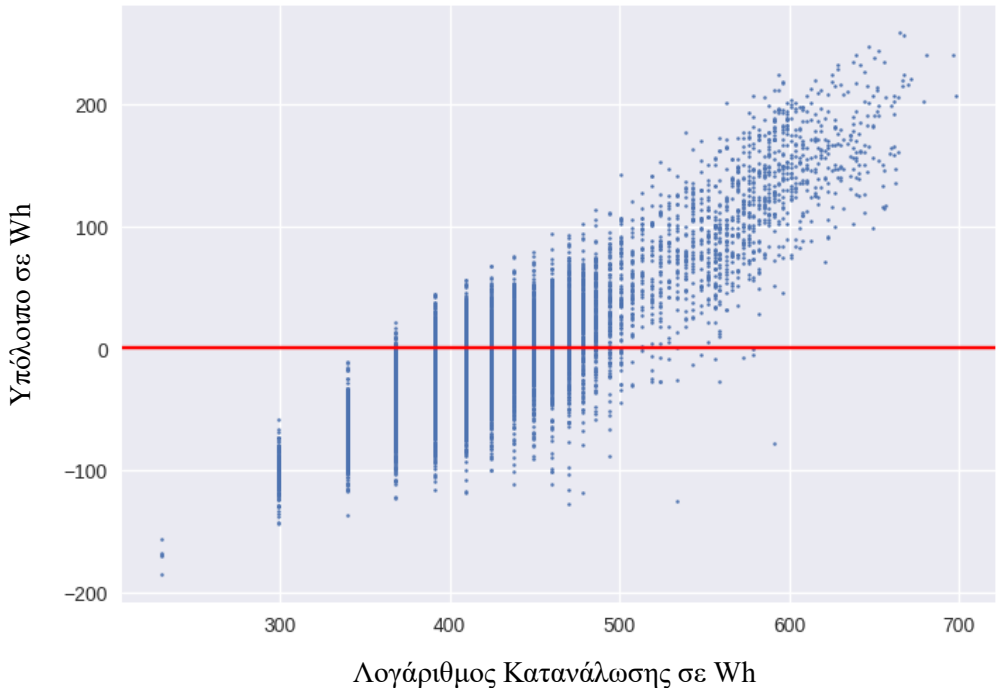
Τα βάρη για τις 25 μεταβλητές είναι:
[1.32e+01 1.94e+00 4.18e+01 -2.10e+01 -3.30e+01 3.27e+01 8.31e+00
 -5.75e+00 -8.05e-01 2.44e+00 2.69e+00 3.33e+01 7.29e+00 -3.46e+00
 -2.42e+00 2.04e+01 -2.44e+01 -2.72e+01 -5.30e+00 -4.20e+01 3.16e-02
 -1.34e+01 3.12e+00 1.09e+00 1.88e+01]
και του σταθερού όρου:
13.17



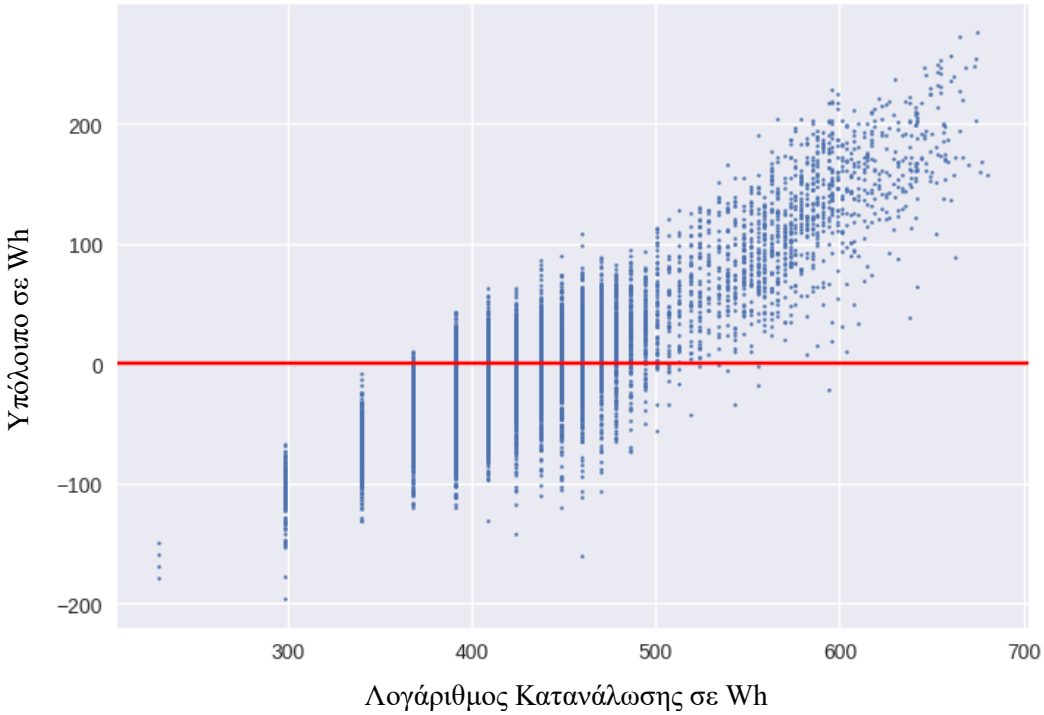
Σχήμα 15: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης



Σχήμα 16: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης



Σχήμα 17: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο εκπαίδευσης



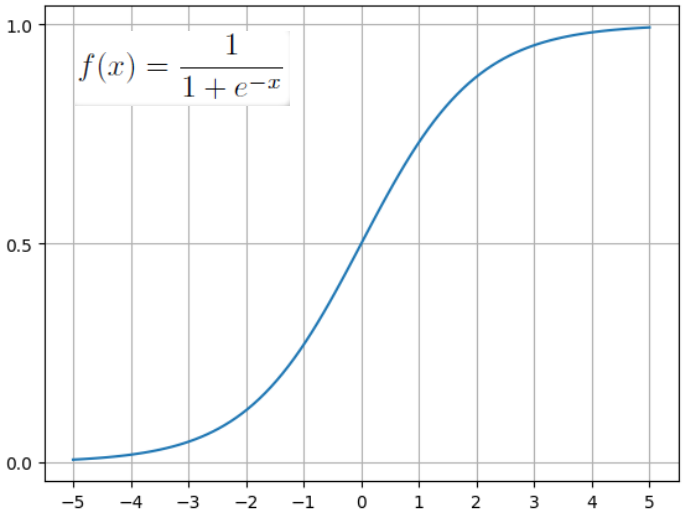
Σχήμα 18: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο αξιολόγησης

Εδώ σημειώνουμε ότι για τις μεγαλύτερες παρατηρήσεις καλύτερα λειτουργεί το μοντέλο με τη λογαριθμιμένη κλίμακα, ωστόσο στις πιο μικρές λειτουργεί καλύτερα το

πρώτο. Ξέρουμε ότι οι περισσότερες παρατηρήσεις είναι μικρές, οπότε το πρώτο μοντέλο που εφαρμόσαμε μας εξυπηρετεί καλύτερα.

6 Λογιστική Παλινδρόμηση

Η λογιστική παλινδρόμηση (Logistic Regression) αποτελεί κατά βάση μοντέλο ταξινόμησης των τιμών μιας μεταβολής απόκρισης y. Στο μοντέλο αυτό η μεταβλητή y συνήθως έχει δυαδικό (binary) χαρακτήρα σήματος ή υποβάθρου. Ωστόσο, μπορεί να δουλέψει και για διακριτές τιμές (non continuous data). Η βασική διαφορά μεταξύ γραμμικής και λογιστικής παλινδρόμησης είναι ότι ενώ κατά τη γραμμική παλινδρόμηση η εκτίμηση των παραμέτρων $\hat{b}$ γίνεται με τη μέθοδο των ελάχιστων τετραγώνων, κατά τη λογιστική η εκτίμηση των παραμέτρων γίνεται με τη μέθοδο του λόγου πιθανοφάνειας. Τα μοντέλα λογιστικής παλινδρόμησης υπολογίζουν την καμπυλόγραμμη σχέση αναμεσά στην y και στις μεταβλητές Xi, οι οποίες μπορεί να είναι συνεχείς ή διακριτές. Η καμπύλη της λογιστικής παλινδρόμησης είναι προσεγγιστικά γραμμική στις μεσαίες τιμές και λογαριθμική στις ακραίες τιμές.



Σχήμα 19: Καμπύλη της Λογιστικής Παλινδρόμησης

Ο λογάριθμος των λόγων πιθανοφάνειας μοντελοποιείται μέσω γραμμικών συναρτήσεων. Δηλαδή,

$$\ln \frac{P(\omega_i|x)}{P(\omega_M|x)} = w_{i,0} + w_i{}^T x \text{ , i=1,2,...,M-1}$$

Από πιθανότητες έχουμε ότι:

$$\sum_{i=1}^{M} P(\omega_i|x) = 1$$

Συνδυάζοντας τα παραπάνω καταλήγουμε στο ότι αυτό το είδος γραμμικής μοντελοποίησης είναι ισοδύναμο με μια εκθετική μοντελοποίηση των εκ των υστέρων πιθανοτήτων:

$$P(\omega_{\mathrm{M}}|x) = \frac{1}{1+\sum_{i=1}^{M-1}\exp(w_{i,0}+{w_i}^T x)}$$

$$P(\omega_i|x) = \frac{\exp(w_{i,0}+{w_i}^T x)}{1+\sum_{i=1}^{M-1}\exp(w_{i,0}+{w_i}^T x)}, i=1,2,\ldots,M-1$$

Για να εκτιμηθεί το σύνολο των άγνωστων παραμέτρων, συνήθως υιοθετείται μια προσέγγιση μέγιστης πιθανοφάνειας. Η βελτιστοποίηση πραγματοποιείται ως προς όλες τις παραμέτρους, τις οποίες μπορούμε να θεωρήσουμε ως στοιχεία ενός διανύσματος παραμέτρων **θ** έστω x_k , k=1,2,…,N είναι τα διανύσματα χαρακτηριστικών εκπαίδευσης, με γνωστές κατανομές. Η συνάρτηση λογαριθμικής πιθανοφάνειας που θα βελτιωθεί δίνεται από την σχέση:

$$L(\theta) = ln\{\prod_{k=1}^{N_1} p(x_k^{(1)}|\omega_1;\theta)\prod_{k=1}^{N_2} p(x_k^{(2)}|\omega_2;\theta)\ldots\prod_{k=1}^{N_{\mathrm{M}}} p(x_k^{(\mathrm{M})}|\omega_{\mathrm{M}};\theta)\}$$

Λαμβάνοντας υπόψη ότι

$$p(x_k^{(m)}|\omega_m;\theta) = \frac{p(x_k^{(m)}P(\omega_m|x_k^{(m)};\theta)}{P(\omega_m)}$$

Προκύπτει:

$$L(\theta) = \sum_{k=1}^{N_1} lnP(\omega_1|x_k^{(1)}) + \sum_{k=1}^{N_2} lnP(\omega_2|x_k^{(2)}) + \ldots + \sum_{k=1}^{N_M} lnP(\omega_M|x_k^{(M)}) + C$$

Όπου η ρητή εξάρτηση από το **θ** έχει καταργηθεί για λόγους απλότητας, ενώ το C είναι η παράμετρος ανεξάρτητη του **θ** και ίση με

$$C = ln\frac{\prod_{k=1}^{N} p(x_k)}{\prod_{m=1}^{M} P(\omega_m)^{N_m}}$$

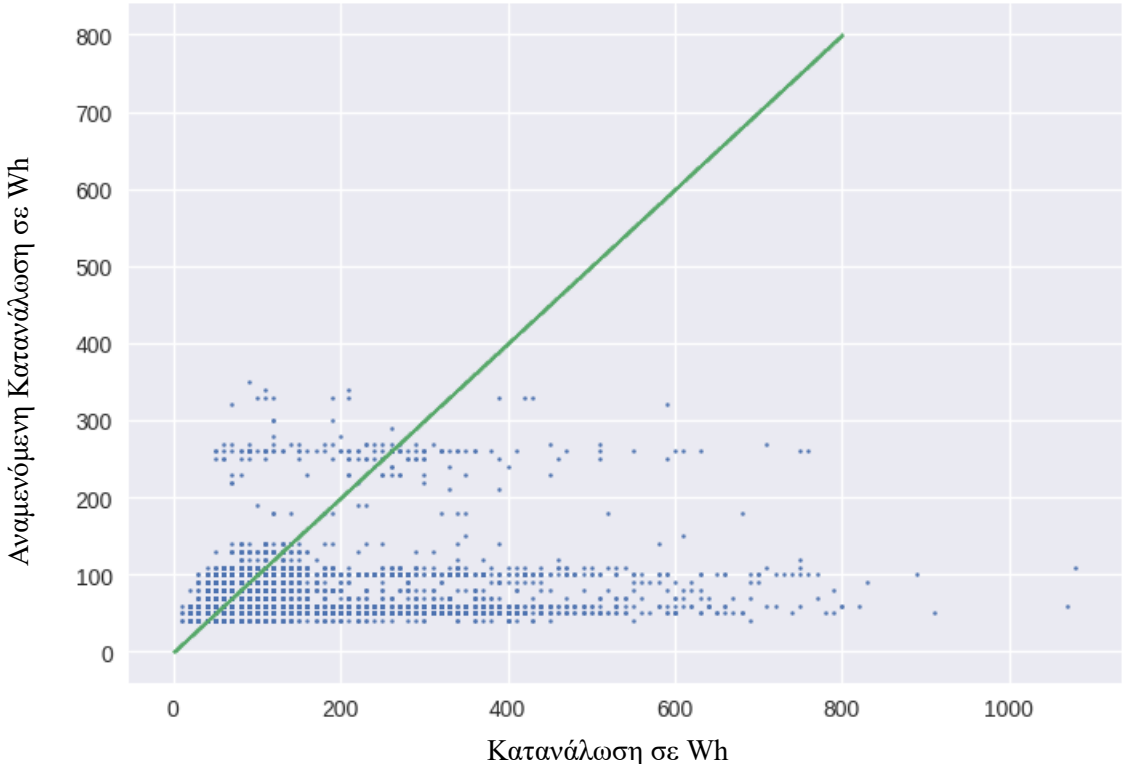
.

Βασικό μειονέκτημα του λογιστικού μοντέλου αποτελεί το γεγονός ότι κατά την κατασκευή του πρέπει να αποφεύγεται η χρήση μεταβλητών που παρουσιάζουν μεγάλη συσχέτιση. Επίσης, παρόλο που δεν απαιτεί οι μεταβλητές να ακολουθούν την κανονική κατανομή, αν αυτές χαρακτηρίζονται από ακραία μη κανονικότητα (extreme non-normality), τότε τα αποτελέσματα του μοντέλου ενδέχεται να μην είναι ικανοποιητικά.

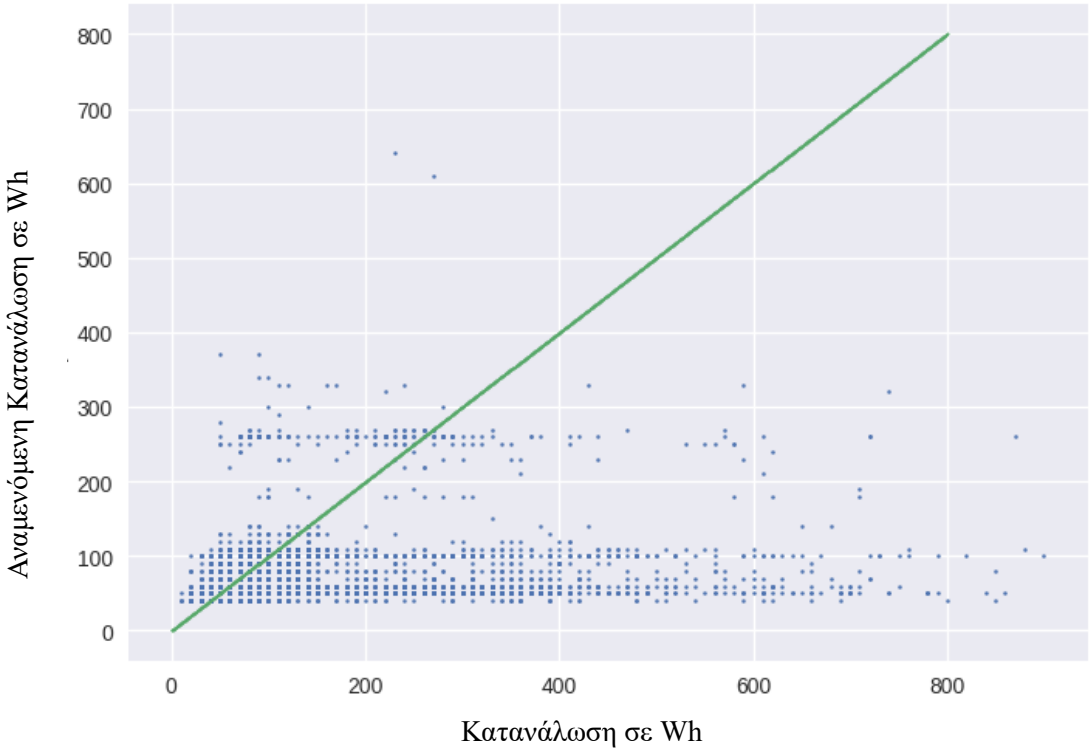
Ένα από τα πλεονεκτήματα του λογιστικού υποδείγματος είναι ότι δεν υπόκειται σε στατιστικούς περιορισμούς, όπως η διακριτική ανάλυση, επομένως μπορεί κατά την κατασκευή του να χρησιμοποιηθούν και ποιοτικές μεταβλητές. Ουσιαστικά όμως και σε αυτό παρατηρούνται στατιστικές υποθέσεις, οι οποίες μπορεί να μην αναφέρονται στα εξεταζόμενα δεδομένα, αλλά υφίστανται στο εξαγόμενο αποτέλεσμα.

Εφαρμόζοντας το μοντέλο της λογιστικής παλινδρόμησης στο σύνολο εκπαίδευσης καλατήγουμε στα εξής:

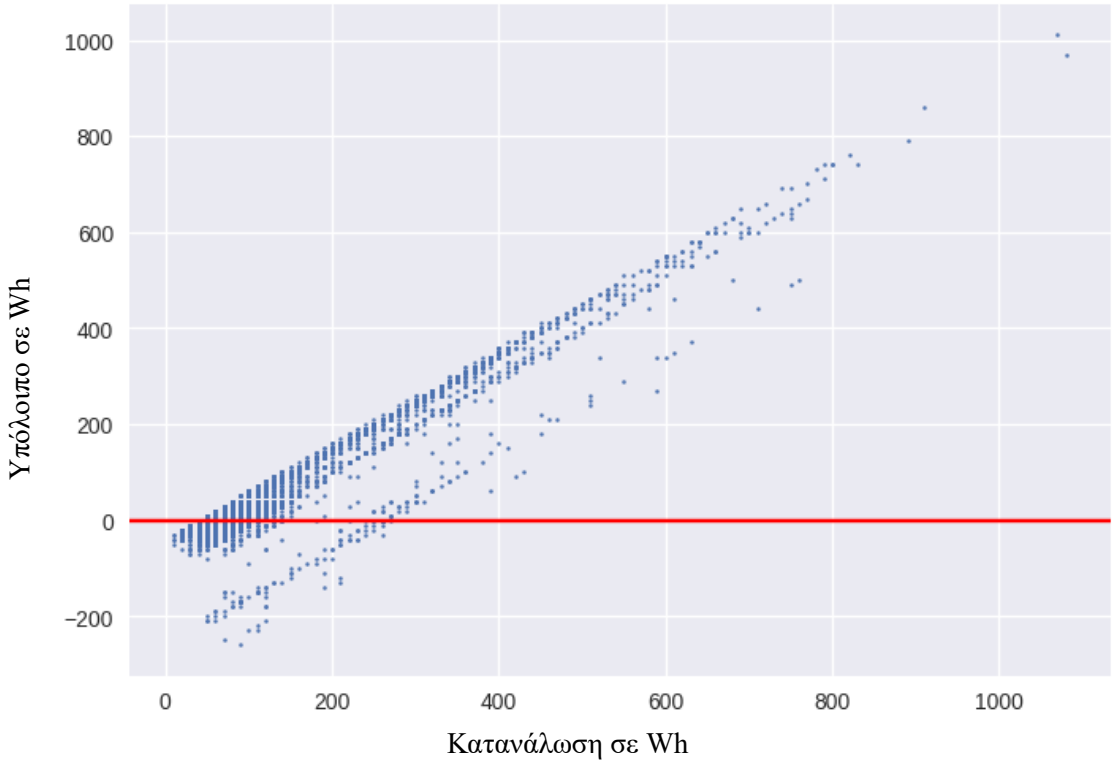
Στα παρακάτω σχήματα η πράσινη ευθεία (όπου η πραγματική τιμή ισούται με την αναμενόμενη τιμή) και η κόκκινη (όπου τα υπόλοιπα πραγματικής μείον αναμενόμενης τιμής είναι μηδενικά) είναι βοηθητικές οπτικά.



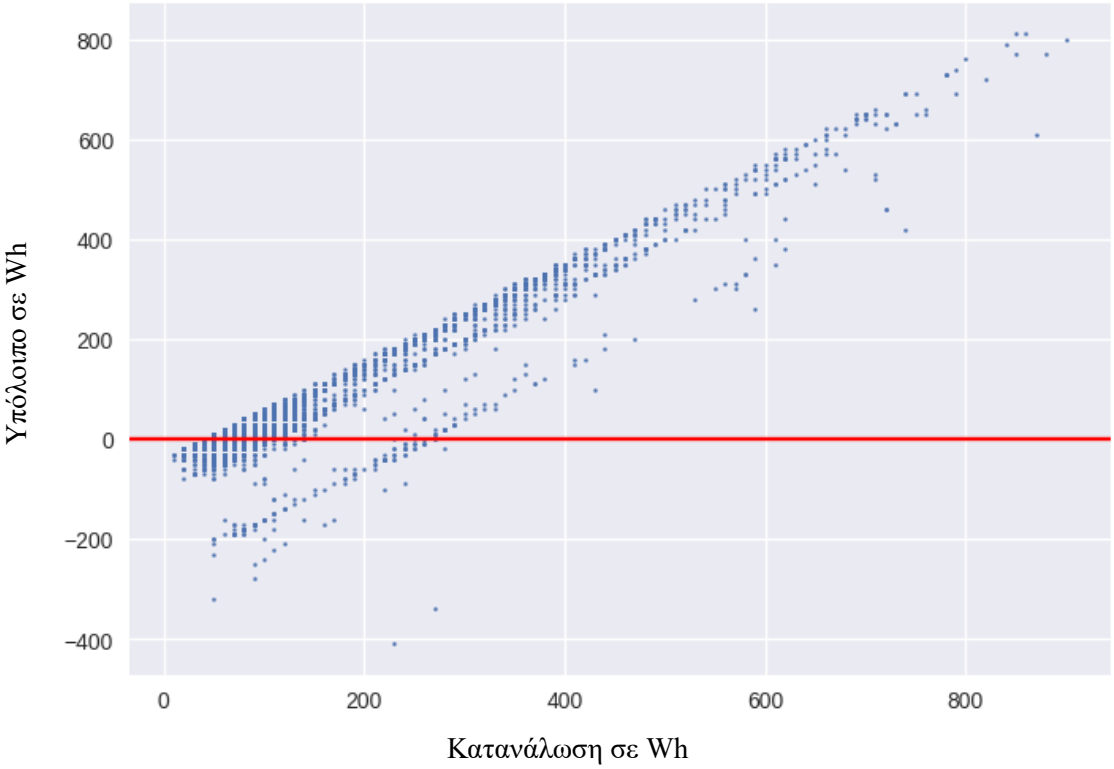
Σχήμα 20: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης



Σχήμα 21: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης

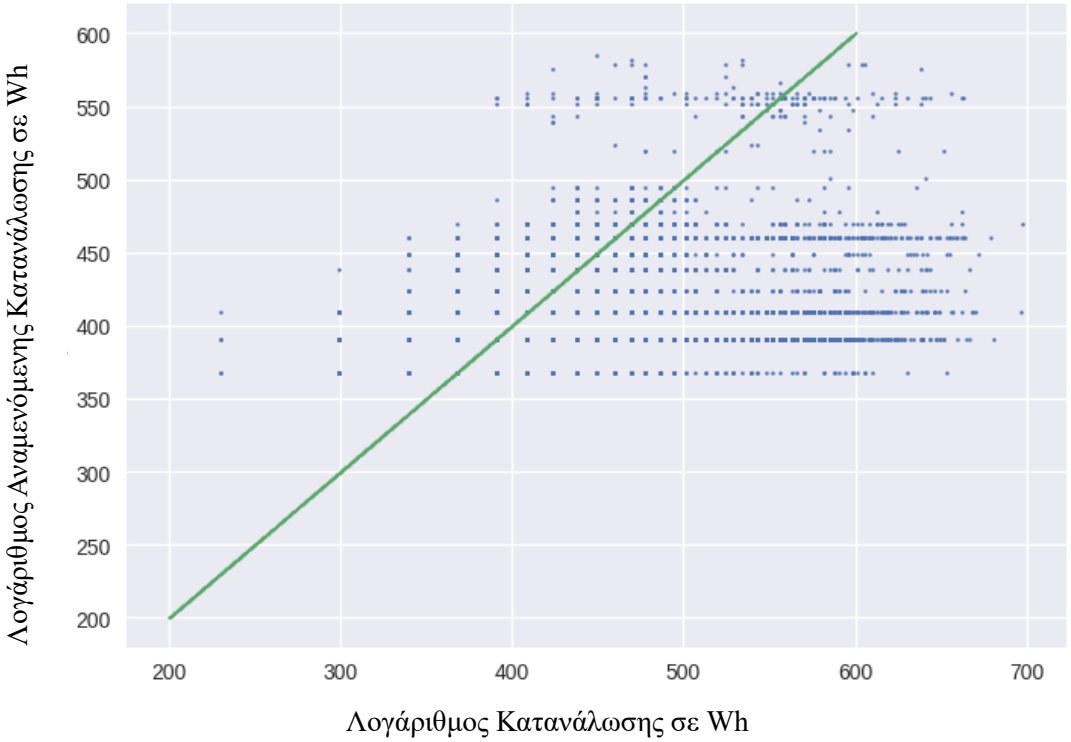


Σχήμα 22: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο εκπαίδευσης

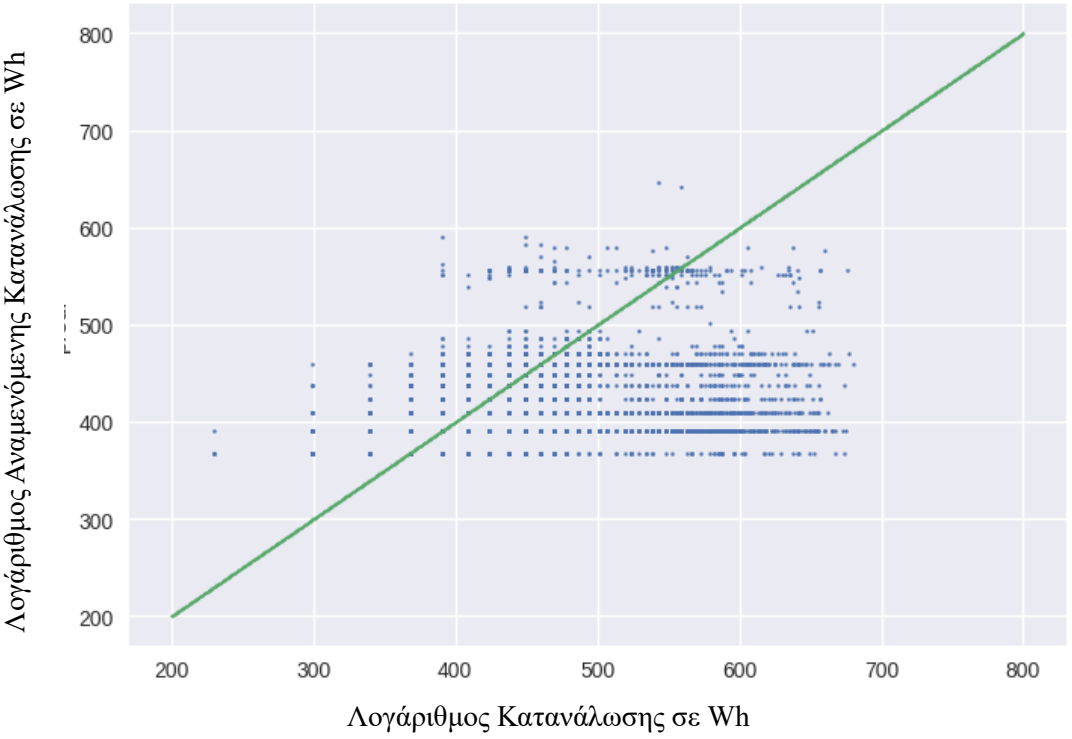


Σχήμα 23: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο αξιολόγησης

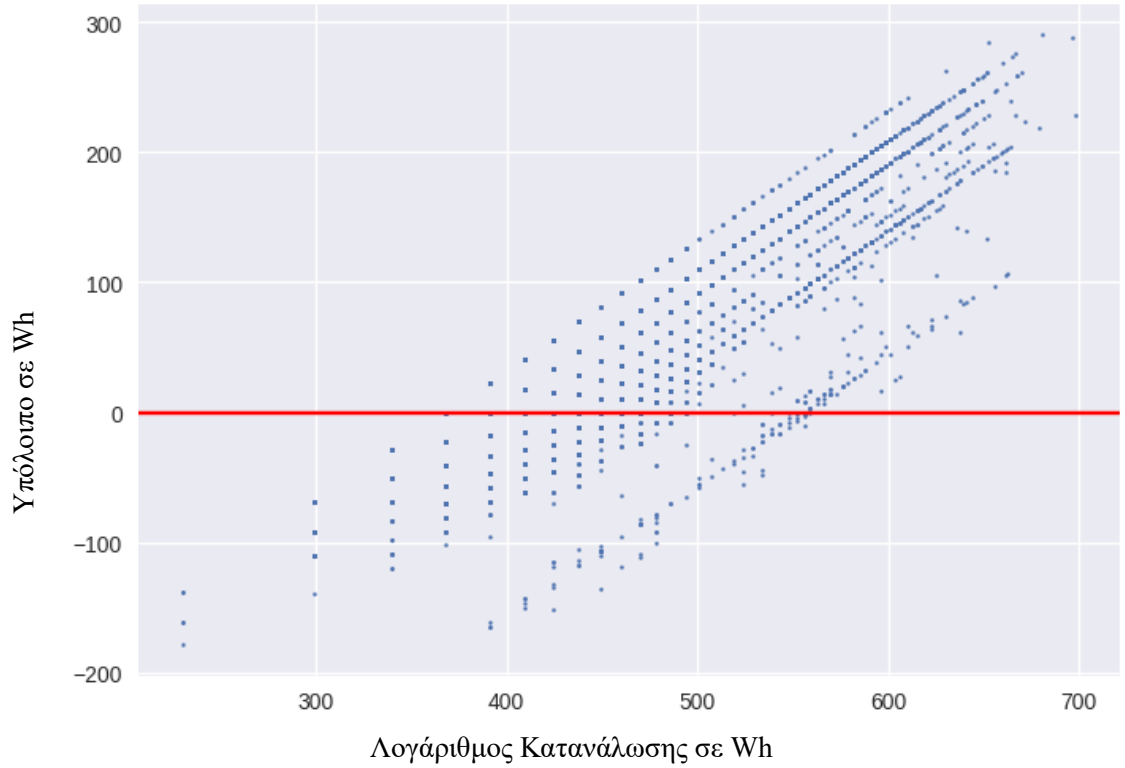
Εφαρμόζοντας το μοντέλο λογιστικής παλινδρόμησης στο σύνολο εκπαίδευσης, αντίστοιχα για λογαριθμική κλίμακα καλατήγουμε στα εξής:



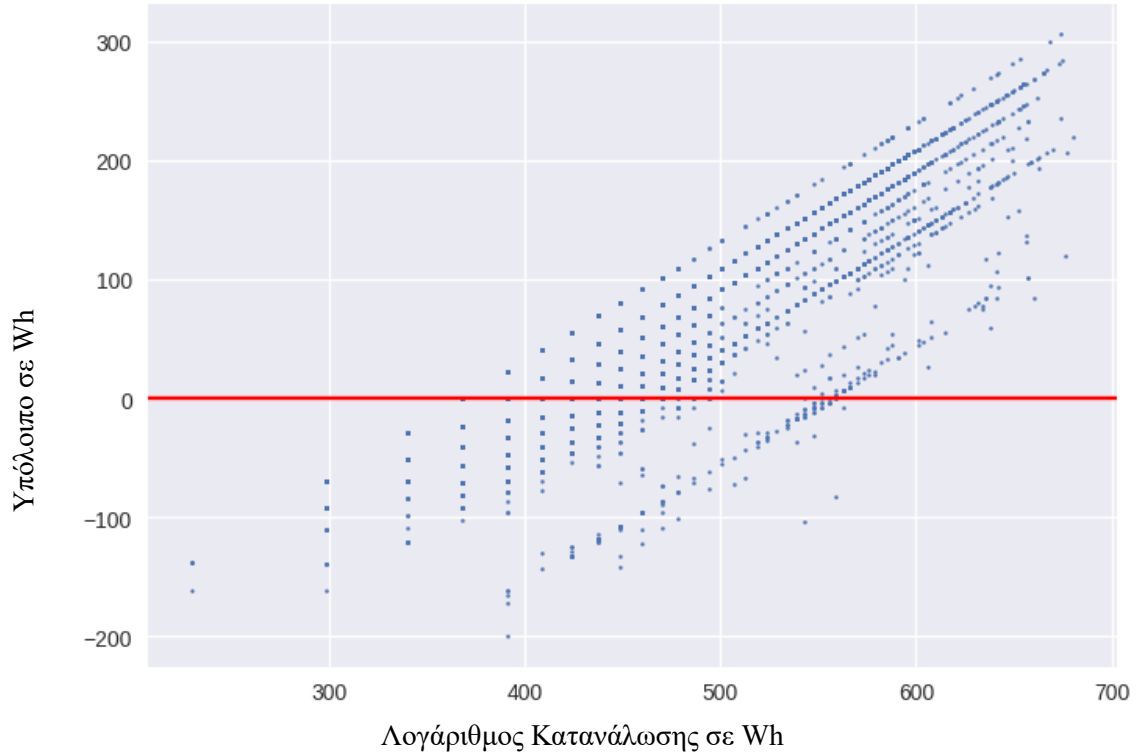
Σχήμα 24: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης



Σχήμα 25: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης



Σχήμα 26: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο εκπαίδευσης

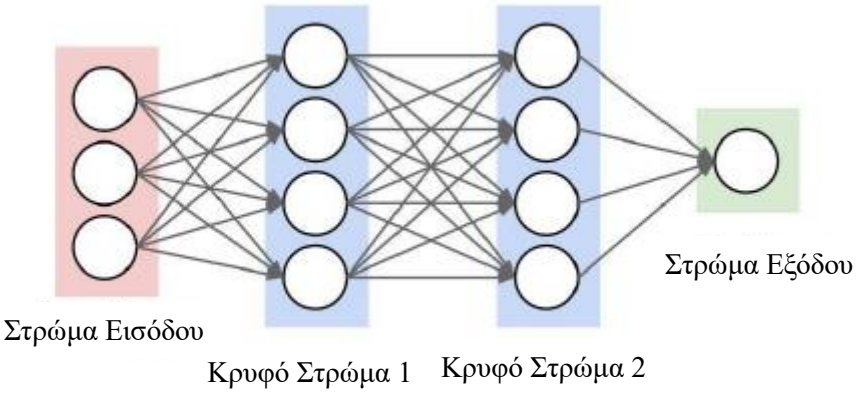


Σχήμα 27: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο αξιολόγησης

Από τα παραπάνω καταλήγουμε ότι το να ασχολούμαστε με λογαριθμική κλίμακα δεν ωφελεί. Πώς μπορούμε, λοιπόν, να πάρουμε καλύτερες προβλέψεις εκμεταλλευόμενοι το δυναμικό εύρος της εξαρτημένης μεταβλητής; Σε αυτό το σημείο θα «χωρίσουμε» την κατανάλωση σε πέντε διαστήματα, βάσει της συνολικής κατανάλωσης ενέργειας των οικιακών συσκευών σε Wh. Πιο συγκεκριμένα, αυτά αντιπροσωπεύουν κατανάλωση από 0 έως 50, από 50 έως 100, από 100 έως 150, από 150 έως 200 και μεγαλύτερη από 200 Wh.

7 Νευρωνικό Δίκτυο

Το feed-forward νευρωνικό δίκτυο είναι ο τύπος νευρωνικού δικτύου, όπου οι συνάψεις γίνονται μόνο προς τα μπροστά. Με βάση το θεώρημα ομοιόμορφης προσέγγισης (Universal Approximation Theorem) μπορούμε να προσεγγίσουμε κάθε συνεχή συνάρτηση με ένα feed-forward δίκτυο με ένα κρυφό στρώμα και πεπερασμένο πλήθος νευρώνων σε ένα συμπαγές σύνολο. Στην παρούσα εργασία θα προσπαθήσουμε να προβλέψουμε τιμές κατανάλωσης ενέργειας.



Σχήμα 28: Feed-Forward Νευρωνικό Δίκτυο

Τα νευρωνικά δίκτυα είναι μηχανές που εκτελούν συγκεκριμένες εργασίες ή λειτουργίες προσομοιώνοντας τη δομή και τη συμπεριφορά των νευρώνων του ανθρώπινου νευρικού συστήματος. Υλοποιούνται σε έναν υπολογιστή, μέσω λογισμικού προσομοίωσης και αποτελούνται από ένα σύνολο διασυνδεδεμένων τεχνητών νευρωνικών κυττάρων που εκπαιδεύονται παράλληλα, μέσω μιας διαδικασίας μάθησης, χρησιμοποιώντας δεδομένα. Η αποκτηθείσα γνώση "αποθηκεύεται" με τη μορφή των συναπτικών βαρών των διασυνδέσεων των νευρώνων και της πόλωσης των νευρώνων των κόμβων του δικτύου.

Τα νευρωνικά δίκτυα αποτελούν πολύ δυνατό ερευνητικό εργαλείο και τα πλεονεκτήματα των εφαρμογών τους είναι πολλά. Μπορούν να αναπαραστήσουν ικανοποιητικά μη γραμμικές συμπεριφορές. Ακόμη, δύνανται να διαχειριστούν ελλιπή δεδομένα ή δεδομένα με υψηλό επίπεδο «θορύβου». Επίσης, ως μη παραμετρική μεθοδολογία, δε βασίζονται σε υποθέσεις που σχετίζονται με τις στατιστικές ιδιότητες των δεδομένων του εκάστοτε προβλήματος. Ωστόσο, τα νευρωνικά δίκτυα έχουν τα μειονεκτήματά τους. Συγκεκριμένα, είναι αδύνατο να εκτιμηθεί η συμβολή κάθε μεταβλητής στα εξαγόμενα αποτελέσματα και δεν παράγουν κάποιους κανόνες που να αποτυπώνουν τη λειτουργία τους. Για το λόγο αυτό, τα νευρωνικά δίκτυα αναφέρονται ως μαύρα κουτιά (black boxes). Επιπροσθέτως, οι ερευνητές τονίζουν ότι δεν υπάρχουν κάποιοι σαφείς κανόνες που σχετίζονται με την επιλογή των παραμέτρων του δικτύου. Το μέγεθος του δικτύου, η επιλογή κατάλληλης συνάρτησης μετασχηματισμού ή συνάρτησης μέτρησης των σφαλμάτων εξαρτάται από το εκάστοτε πρόβλημα.

7.1 Αριθμός Νευρώνων και Στρωμάτων

Η βασική μονάδα ενός νευρωνικού δικτύου είναι ο νευρώνας. Οι νευρώνες επεξεργάζονται τις πληροφορίες που λαμβάνουν με τη μορφή σημάτων, εκτελούν προκαθορισμένες πράξεις σε αυτές και μεταφέρουν τις τροποποιημένες πληροφορίες στον επόμενο νευρώνα ή στην έξοδο του δικτύου.

Για να κατασκευάσουμε ένα νευρωνικό δίκτυο (NN) είναι απαραίτητο να καθορίσουμε την αρχιτεκτονική του. Κάθε τέτοιο δίκτυο αποτελείται από ένα στρώμα εισόδου, κρυφά στρώματα και ένα στρώμα εξόδου.

Κάθε NN έχει ακριβώς ένα στρώμα εισόδου. Το πλήθος των νευρώνων του στρώματος εισόδου είναι ίσο με το πλήθος των μεταβλητών που έχει ένα σύνολο εκπαίδευσης. Στην περίπτωσή μας το πλήθος των μεταβλητών είναι 25.

Κάθε NN έχει ακριβώς ένα στρώμα εξόδου. Το πλήθος των νευρώνων του εξαρτάται από το τι θέλουμε να εκτιμήσουμε τελικά.

Σχετικά με τα κρυφά στρώματα, δεν ισχύουν αντίστοιχοι αυστηροί κανόνες. Μπορούμε να επιλέξουμε το επιθυμητό πλήθος κρυφών στρωμάτων, το οποίο θα βελτιστοποιεί τα αποτελέσματα του δικτύου μας. Αυτό συνήθως γίνεται μέσω διαδικασιών δοκιμής και λάθους (trial and error) ή χρησιμοποιώντας νευρωνικά δίκτυα, τα οποία μπορούν να προσαρμοστούν βάσει του εκάστοτε προβλήματος (self-organising neural networks). Γενικά, ένα γραμμικώς διαχωρίσιμο πρόβλημα δε χρειάζεται καθόλου κρυφά στρώματα. Ο αριθμός τους είναι ανάλογος της πολυπλοκότητας του προβλήματος και επηρεάζεται από το μέγεθος του συνόλου εκπαίδευσης. Τυπικά, ξεκινάμε με απλή αρχιτεκτονική (λόγου χάρη ένα κρυφό στρώμα με λίγους νευρώνες) και σταδιακά αυξάνουμε το πλήθος των νευρώνων ή και των στρωμάτων, ελέγχωντας πάντα την απόδοση σε το σύνολο αξιολόγησης. Εδώ, θα υλοποιούμε τρία νευρωνικά δίκτυα το πρώτο με ένα κρυφό στρώμα με 28 νευρώνες, το δεύτερο με τρία κρυφά στρώματα με 28, 27 και 20 νευρώνες αντίστοιχα και το τρίτο με επτά κρυφά στρώματα με 74, 60 40, 32, 16, 8 και 4.

Αξίζει να σημειωθεί ότι η αύξηση των στρωμάτων επηρεάζει διαφορετικά την πολυπλοκότητα του προβλήματος από την αύξηση των νευρώνων σε ένα στρώμα. Η πρώτη διαδικασία αυξάνει εκθετικά τη μη γραμμικότητα του προβλήματος και την ικανότητα να περιγράψει σύνθετες συναρτήσεις.

7.2 Forward Propagation

Αφού εισάγουμε τα δεδομένα μας στο στρώμα εισόδου, αυτά «προχωρούν» στο πρώτο κρυφό στρώμα. Οι τιμές κάθε νευρώνα πολλαπλασιάζονται με τις σταθερές των βαρών και στο γινόμενο προσθέτουμε τη μεροληψία (bias). Τα βάρη αποτελούν το χαρακτηριστικό της σύνδεσης μεταξύ των νευρώνων των στρωμάτων, ενώ η μεροληψία αποτελεί χαρακτηριστικό των ίδιων των νευρώνων. Ανάμεσα σε δύο διαδοχικά στρώματα θα περιμένουμε να έχουμε α_1 x α_2 βάρη, όπου $\alpha_{1,2}$ οι αριθμοί των

νευρώνων των διαδοχικών στρωμάτων και α_2 μεροληψίες. Σε γλώσσα μαθηματικών ισχύει:

$$z^{(n+1)} = W^{(n)} z^{(n)} + b^{(n)}$$

όπου $z^{(n+1)}$, $z^{(n)}$ έχουν διαστάσεις $(\alpha_1,1)$ και $(\alpha_2,1)$ αντίστοιχα και $W^{(n)}$, $b^{(n)}$ έχουν διαστάσεις (α_1,α_2) και $(\alpha_2,1)$ αντίστοιχα.

Στη συνέχεια, το αποτέλεσμα μετασχηματίζεται μέσω της συνάρτησης ενεργοποίησης σε

$$a^{(n+1)} = f(z^{(n+1)})$$

το οποίο περνά στο επόμενο στρώμα.

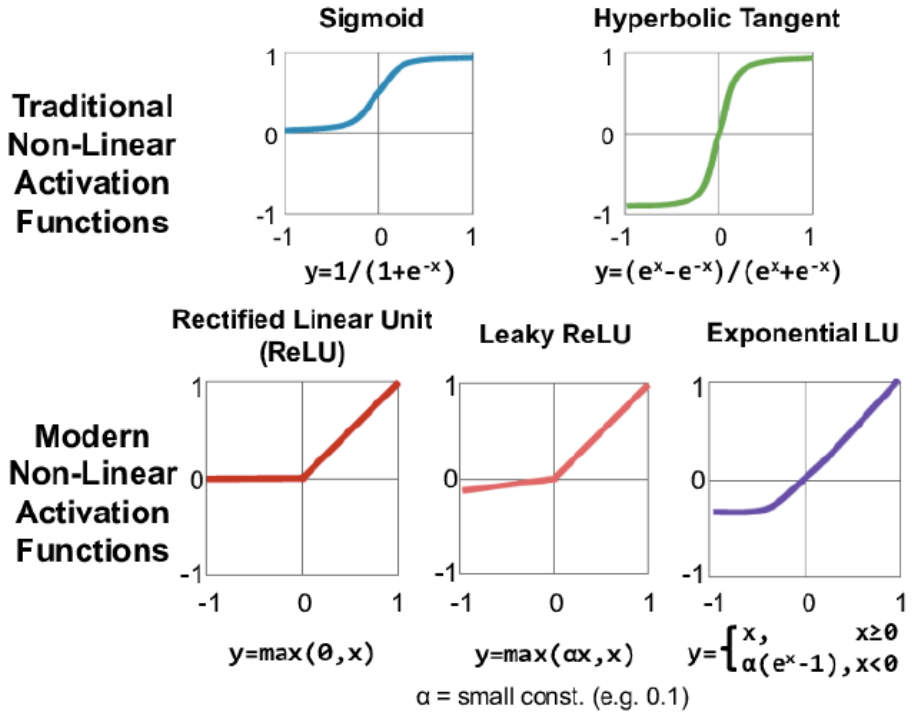
Οι συναρτήσεις ενεργοποίησης είναι ο λόγος που τα NN είναι εργαλείο μη γραμμικής παλινδρόμησης. Αυτές οι συναρτήσεις αφορούν τα κρυφά στρώματα και ικανοποιούν τα εξής χαρακτηριστικά:

- Μη γραμμικότητα: σε αντίθετη περίπτωση θα είχαμε γραμμική παλινδρόμηση. Αν ήταν γραμμική, όπως αναφέρθηκε προηγουμένως, το βάθος του NN δε θα είχε καμία σημασία.

- Ομαλότητα και μονοτονία: ώστε η συνάρτηση σφάλματος να είναι κυρτή και να υπάρχει ελάχιστο.

- Εύρος: πεπερασμένο εύρος συνεπάγεται μεγαλύτερη ευστάθεια, ενώ άπειρο καλύτερα αποτελέσματα. Εδώ υπεισέρχεται η έννοια του ρυθμού εκμάθησης, όπου όσο μεγαλύτερο το εύρος τόσο μικρότερος θα πρέπει να είναι.

Μερικές συναρτήσεις ενεργοποίησης είναι οι παρακάτω:



Σχήμα 29: Βασικές συναρτήσεις ενεργοποίησης

Γενικά, η ReLU έχει παρατηρηθεί να δίνει τα καλύτερα αποτελέσματα στα περισσότερα προβλήματα. Η διορθωμένη γραμμική συνάρτηση ενεργοποίησης ή ReLU για συντομία είναι μια τμηματική γραμμική συνάρτηση που θα εξάγει την είσοδο απευθείας εάν είναι θετική, διαφορετικά, θα εξάγει μηδέν. Αποτελεί την προεπιλεγμένη συνάρτηση ενεργοποίησης για πολλούς τύπους NN, επειδή ένα μοντέλο που τη χρησιμοποιεί είναι ευκολότερο να εκπαιδευτεί και συχνά επιτυγχάνει καλύτερη απόδοση.

Αντίθετα, η σιγμοειδής δε θα ήταν η καλύτερη επιλογή, επειδή η παράγωγος της είναι πολύ μικρή για μεγάλο μέρος τιμών του πεδίου ορισμού της, μην επιτρέποντας έτσι ιδιαίτερες αναπροσαρμογές στα δεδομένα (βλέπε «vanishing gradient problem»).

7.3 Συναρτήσεις Κόστους

Φτάνοντας στο στρώμα εξόδου καλούμαστε να αξιολογήσουμε εάν ο regressor έχει πετύχει τον στόχο του. Για να γίνει αυτό ελαχιστοποιούμε κατάλληλα κάποιες συναρτήσεις κόστους. Μερικές γνωστές από αυτές είναι οι παρακάτω:

- Μέσο Απόλυτο Σφάλμα (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Μέσο Τετραγωνικό Σφάλμα (MSE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Binary Cross Entropy

$$J(x) = -\sum_{i=1}^{N} p(x_i)\log q(x_i)$$

Όπου n το μέγεθος του δείγματος, $\hat{y}_i$ η εκτίμηση και y_i η πραγματική τιμή.

Οι δύο πρώτες ενδείκνυνται για προβλήματα παλινδρόμησης ενώ η τρίτη για ταξινόμηση σε δυο κλάσεις.

7.4 Back propagation

Back Propagation ονομάζεται η διαδικασία αναπροσαρμογής των μεταβλητών ώστε να έχουμε καλύτερη εκτίμηση για το σφάλμα και να ελαχιστοποιείται η συνάρτηση κόστους. Κατά τη διαδικασία αυτή, αρχικά ορίζονται τυχαία τα βάρη των συνδέσεων του δικτύου. Εν συνεχεία, υπολογίζεται το σφάλμα και αν αυτό ξεπερνά μια επιθυμητή τιμή, τότε μέσω της διαδικασίας «διαδίδεται» προς τα πίσω, επαναπροσδιορίζονται τα βάρη σύνδεσης του δικτύου, το δίκτυο επανεκπαιδεύεται και ξαναπραγματοποιείται ο έλεγχος που προαναφέρθηκε, προσαρμόζονται, δηλαδή, τα βάρη του τελευταίου στρώματος και κατόπιν "ενημερώνονται" αυτά των προηγούμενων στρωμάτων.

Κάτι τέτοιο σε κυρτά προβλήματα είναι εύκολο. Δεν είναι όμως όλα τα προβλήματα κυρτά, άρα πρέπει να επιλέξουμε προσεκτικά τον αλγόριθμο οπισθοδρόμησης και τις παραμέτρους του. Γνωστότεροι αλγόριθμοι για αυτή τη διαδικασία είναι οι Gradient Descend, RMSProp και Adam.

Για τον πρώτο αναλυτικά:

Η μέθοδος ανάμεσα στα στρώματα είναι:

$$w(t+1) = w(t) - \rho_t \nabla J$$

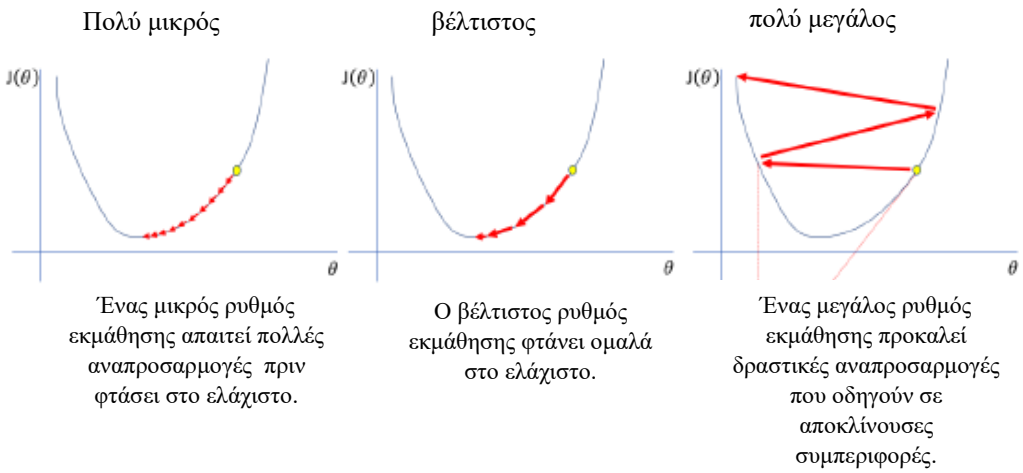
για το τελευταίο στρώμα.

Ενώ για ανάμεσα στα στρώματα ισχύει το εξής:

$$w^{(n)}(t+1) = w^{(n)}(t) - \rho_t \nabla_w f(w^{(n+1)})$$

Η διαδικασία είναι αντίστοιχη και για τις μεροληψίες.

Σαφώς, μας ενδιαφέρει ο ρυθμός εκμάθησης να μην είναι ούτε πολύ μεγάλος, ώστε να μην αποκλίνει το πρόβλημα, αλλά ούτε πολύ μικρός ώστε να μην υπάρχει μεγάλη καθυστέρηση στη σύγκλιση.



Σχήμα 30: Ρυθμοί Εκμάθησης

Για κυρτές συναρτήσεις ο ρυθμός εκμάθησης υπολογίζεται με τον παρακάτω αλγόριθμο:

$$J \approx J(w^*) + \frac{1}{2}\frac{\partial^2 J}{\partial w^2}(w - w^*)^2$$

$$w(t+1) = w(t) - \rho\frac{\partial J}{\partial w}$$

$$= w(t) - \rho\frac{\partial^2 J}{\partial w^2}(w - w^*)$$

$$\rho^* = \left(\frac{\partial^2 J}{\partial w^2}\right)^{-1}$$

Ο αλγόριθμος RMSprop διαφοροποιείται στο πηλίκο με μια ποσότητα που εκτιμούμε με τον παρακάτω τρόπο:

$$\mathbb{E}[g^2]_t = \beta\mathbb{E}[g^2]_{t-1} + (1-\beta)g_t^2$$
$$w_{t+1} = w_t - \frac{\rho}{\sqrt{\mathbb{E}[g^2]_t + \epsilon}}g_t,\ g_t = \nabla_w J(w_{t-1})$$

Αξίζει να σημειωθεί ότι εδώ ο ρυθμός εκμάθησης δεν παραμένει ο ίδιος, αλλά μεταβάλλεται ανάλογα με το πόσο κοντά ή μακριά είμαστε στη βέλτιστη τιμή.

Τέλος, ο αλγόριθμος Adam, ο οποίος είναι και ο πιο αποτελεσματικός, υλοποιείται ως εξής:

Αρχικοποιούμε τις μεταβλητές m0 = u0 = t =0 και όσο ο αλγόριθμος δε συγκλίνει θέτουμε t=t+1 και μετά:

$$g_t = \nabla_\theta f_t(\theta_{t-1})$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$u_t = \beta_2 u_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{u}_t = \frac{m_t}{1-\beta_2^t}$$

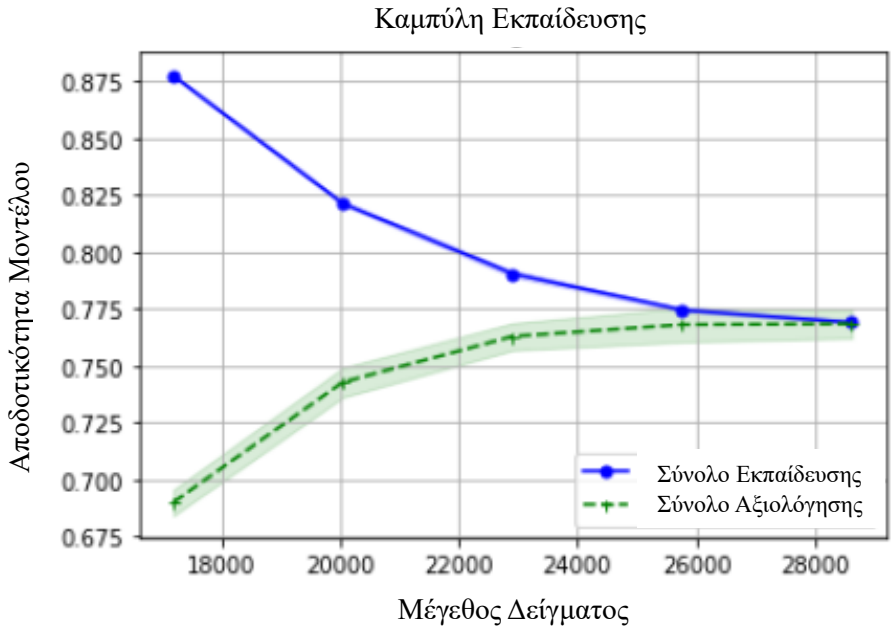
$$\theta_t = \theta_{t-1} - \rho \frac{\hat{m}_t}{sqrt\hat{u}_t + \varepsilon}$$

Μια πλήρης υλοποίηση της παραπάνω διαδικασίας (forward-back propagation) ορίζεται ως μια εποχή.

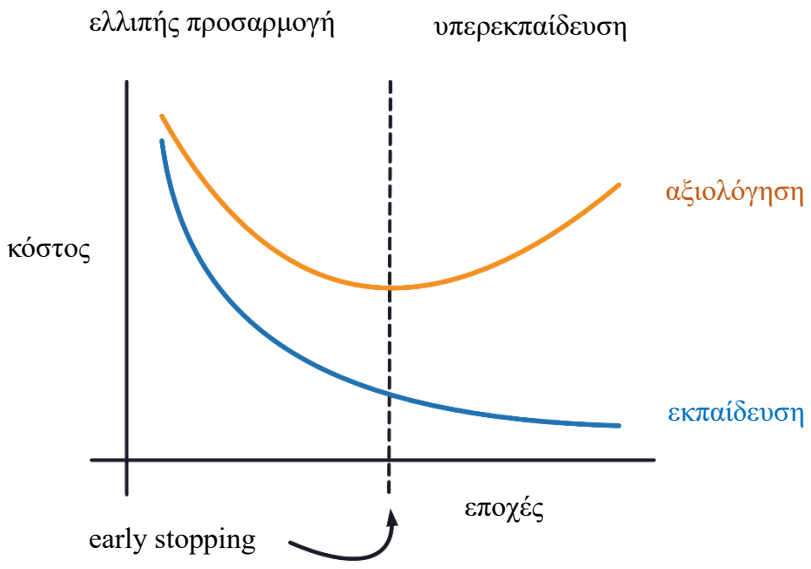
Όσων αφορά στο φαινόμενο της υπερεκπαίδευσης, παρατηρείται ότι μετά από μεγάλο αριθμό επαναλήψεων του «κύκλου» του Νευρωνικού Δικτύου, οδηγούμαστε σε «υπερβολική προσαρμογή» των παραμέτρων. Από ένα σημείο και μετά, ενώ η συνάρτηση κόστους είναι πάντοτε φθίνουσα για το σύνολο εκπαίδευσης, για το σύνολο ελέγχου αρχίζει και αυξάνεται ξανά. Ένα τέτοιο φαινόμενο μας οδηγεί στο συμπέρασμα ότι η επ' άπειρον εκπαίδευση του NN μας οδηγεί μακριά από τα σωστά αποτελέσματα. Εμείς, με χρήση κατάλληλου κριτηρίου τερματισμού, θέλουμε να αποφύγουμε το φαινόμενο αυτό της υπερεκπαίδευσης, έτσι ώστε το αποτέλεσμα του NN να ελαχιστοποιεί τη συνάρτηση κόστους και για το σύνολο ελέγχου.

Το κριτήριο τερματισμού που επιλέγουμε, σταματάει τη συνάρτηση εάν παρατηρηθεί αύξηση της συνάρτησης κόστους μεγαλύτερη από μια σταθερά ανοχής παραπάνω από 20, λόγου χάρη, κύκλους επανάληψης. Πράγματι, όπως θα δούμε και παρακάτω, ένα τέτοιο κριτήριο δεν επιτρέπει στη συνάρτηση κόστους να γίνει και πάλι αύξουσα για το σύνολο ελέγχου και έτσι το NN μας αποφεύγει την υπερεκπαίδευση.

Για να ελέγξουμε αν το κριτήριο τερματισμού της εκπαίδευσης που θέσαμε παραπάνω είναι επαρκές, για να μην έχουμε υπερπροσαρμογή στους εκτιμητές θα μεταποιήσουμε τη μέθοδο Early Stopping.



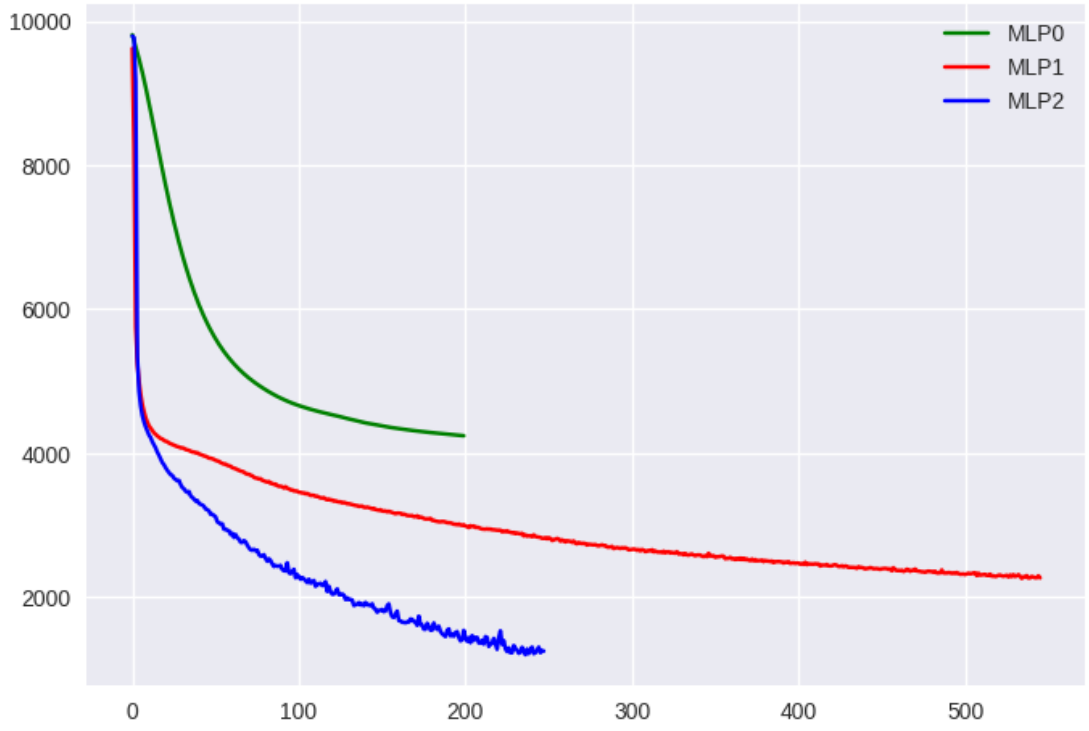
Σχήμα 31: Καμπύλη εκπαίδευσης



Σχήμα 32: Καμπύλες βέλτιστης εκπαίδευσης

Το σημείο ελαχίστου (Early Stopping στο Σχήμα 32) σηματοδοτεί το σημείο, στο οποίο εμφανίζεται υπερεκπαίδευση, δηλαδή τα βάρη προσαρμόζονται υπερβολικά στο σύνολο εκπαίδευσης και χάνεται η ιδιότητα γενίκευσης της εκπαίδευσης.

Όπως αναφέρθηκε και προηγουμένως σε αυτή την εργασία υλοποιήθηκαν 3 Νευρωνικά Δίκτυα, τυπικά αρχίζοντας με την πιο απλή αρχιτεκτονική και σταδιακά προστέθηκαν νευρώνες και κρυφά στρώματα. Πιο συγκεκριμένα, το πρώτο με ένα κρυφό στρώμα με 28 νευρώνες, το δεύτερο με τρία κρυφά στρώματα με 28, 27, 20 νευρώνες αντίστοιχα και το τρίτο με επτά κρυφά στρώματα με 74, 60, 40, 32, 16, 8, 4. Οι συναρτήσεις ενεργοποίησης που χρησιμοποιήθηκαν είναι η logistic, η relu και η relu αντίστοιχα. Καθώς ως βελτιστοποιητής χρησιμοποιήθηκε ο Adam. Στην παρακάτω εικόνα παρουσιάζονται οι συναρτήσεις σφάλματος και για τα τρία NN.

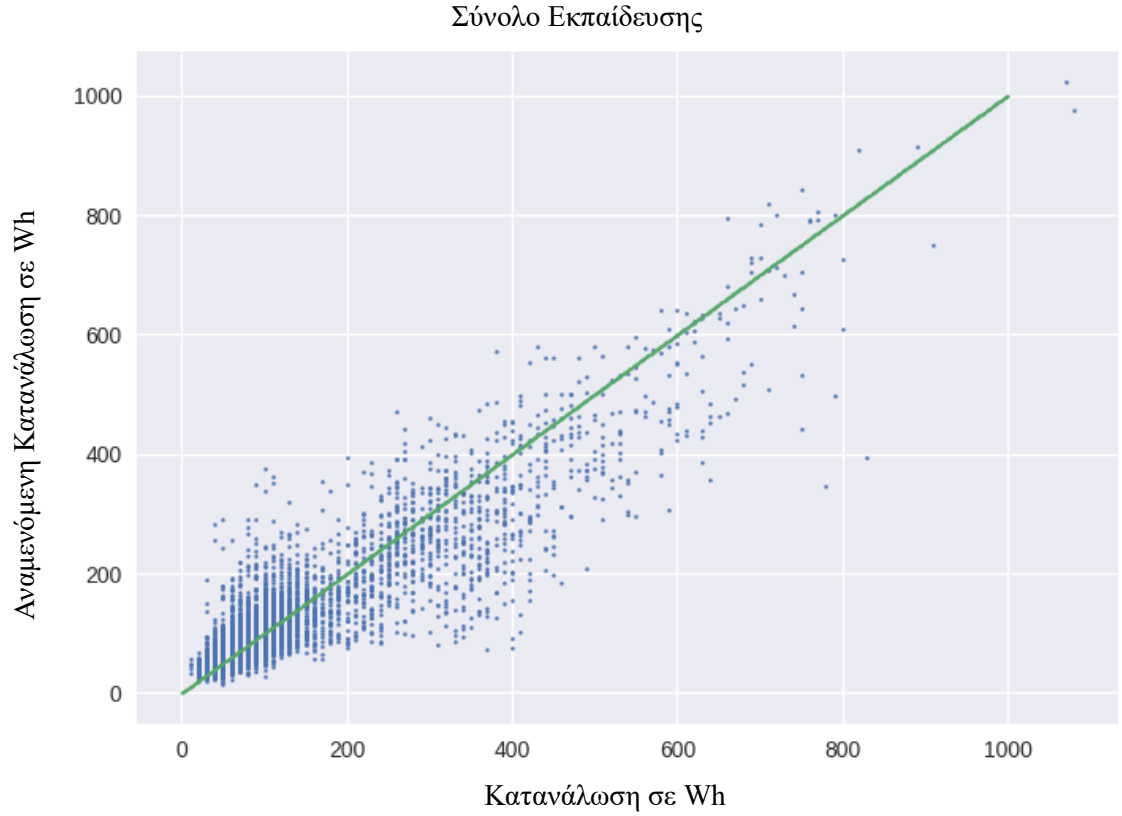


Σχήμα 33: Συναρτήσεις Σφάλματος

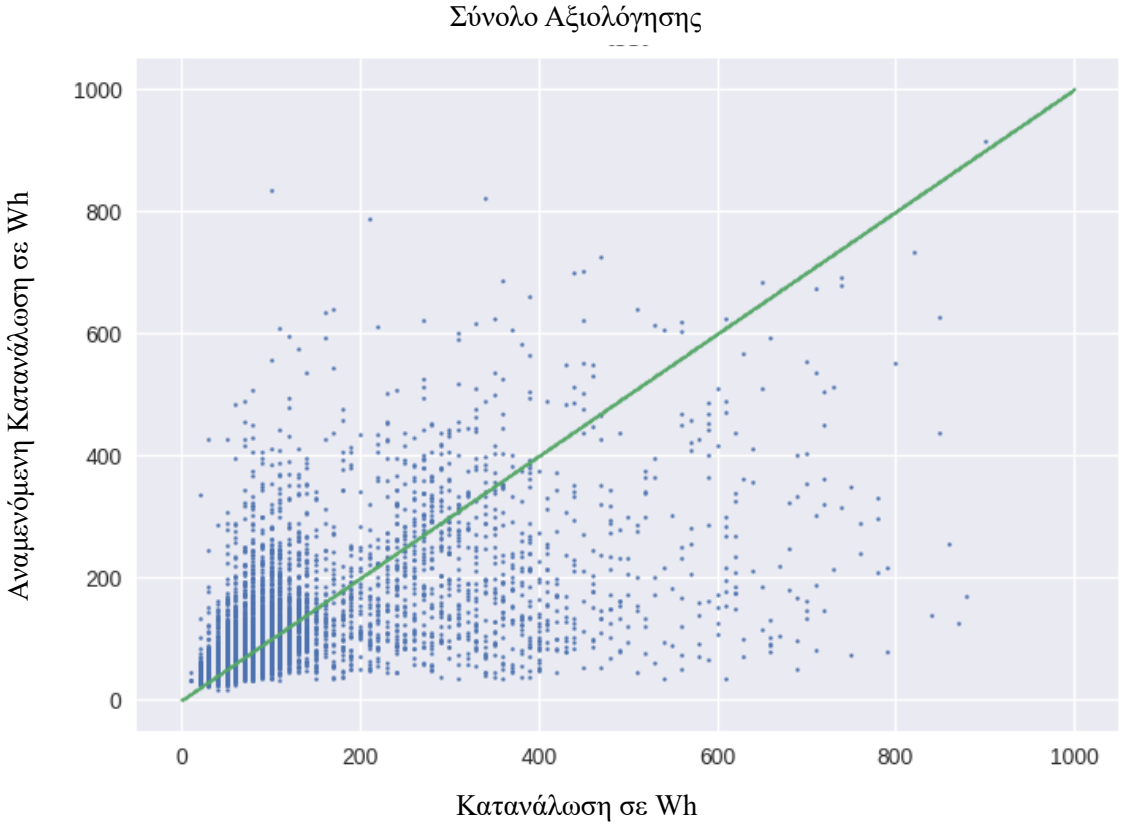
7.5 Αποτελέσματα Νευρωνικού Δικτύου

Σε αυτή την υποενότητα παρουσιάζονται τα αποτελέσματα που λάβαμε μετά την προσαρμογή και υλοποίηση του νευρωνικού δικτύου με τις καλύτερες προβλέψεις από τα τρία , δηλαδή αυτό με τα επτά κρυφά στρώματα με 74, 60, 40, 32, 16, 8, 4 νευρώνες αντίστοιχα με συνάρτηση ενεργοποίησης ReLU και βελτιστοποιητή Adam.

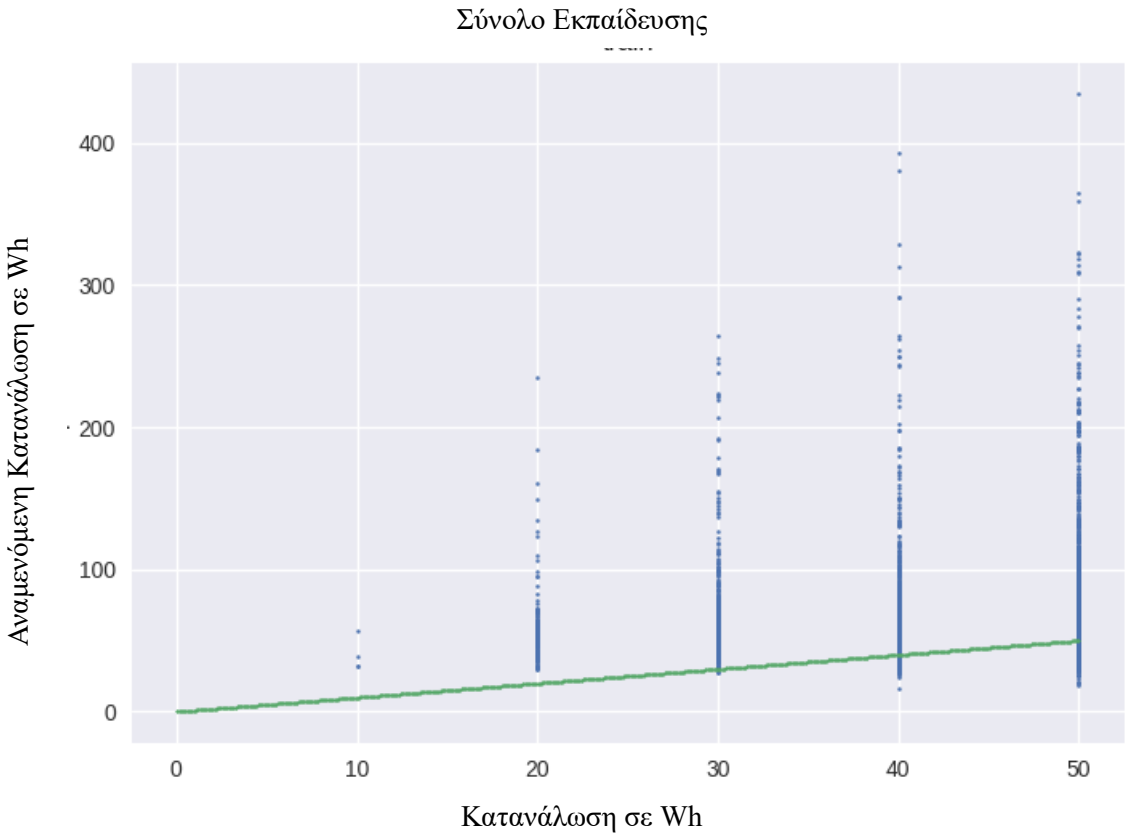
Στα παρακάτω σχήματα η πράσινη ευθεία (όπου η πραγματική τιμή ισούται με την αναμενόμενη τιμή) και η μπλε (όπου τα υπόλοιπα πραγματικής μείον αναμενόμενης τιμής είναι μηδενικά) είναι βοηθητικές οπτικά.



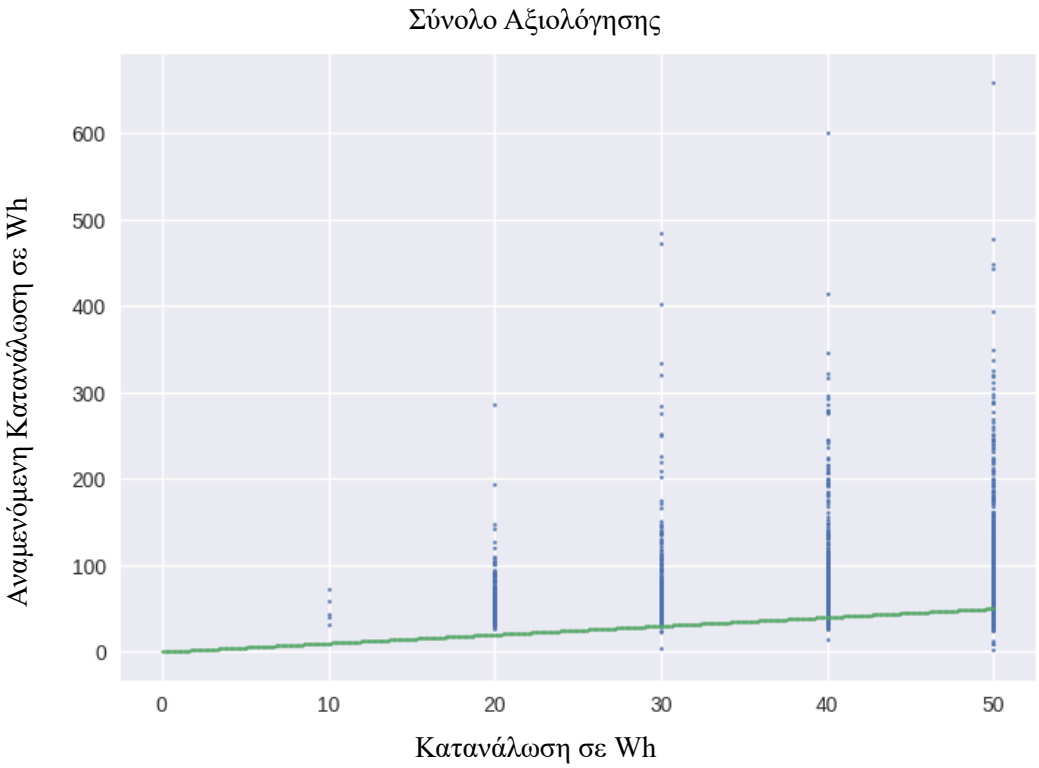
Σχήμα 34: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης



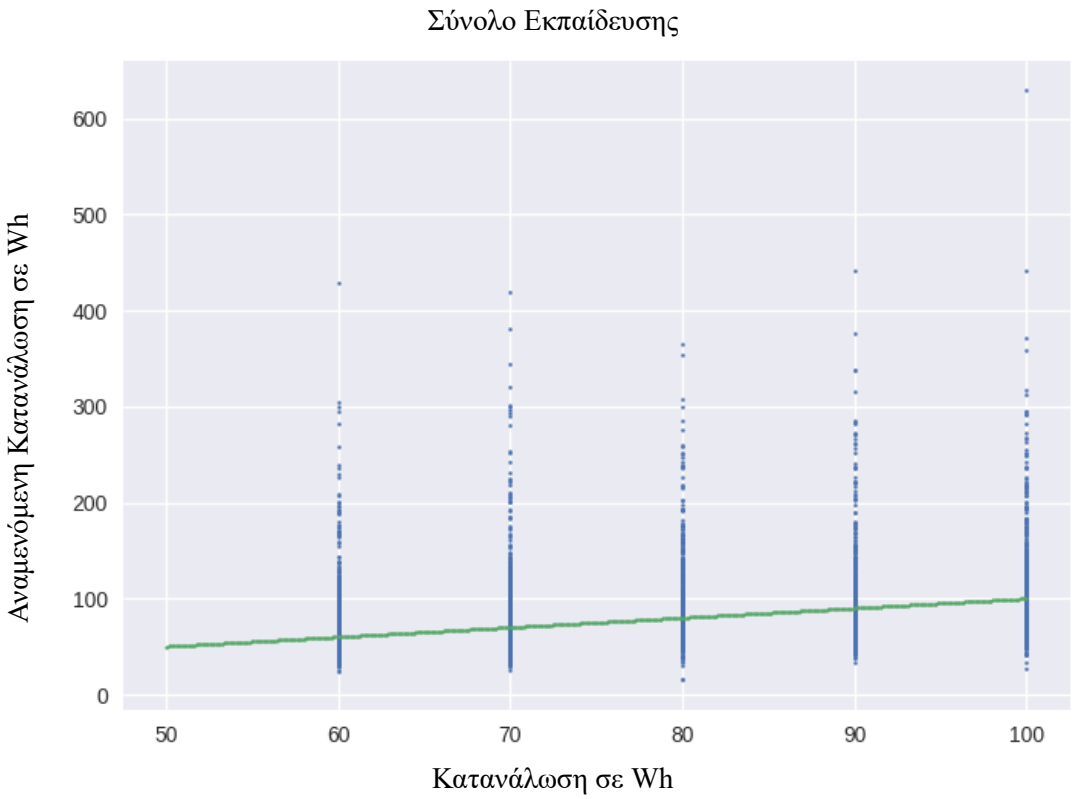
Σχήμα 35: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης



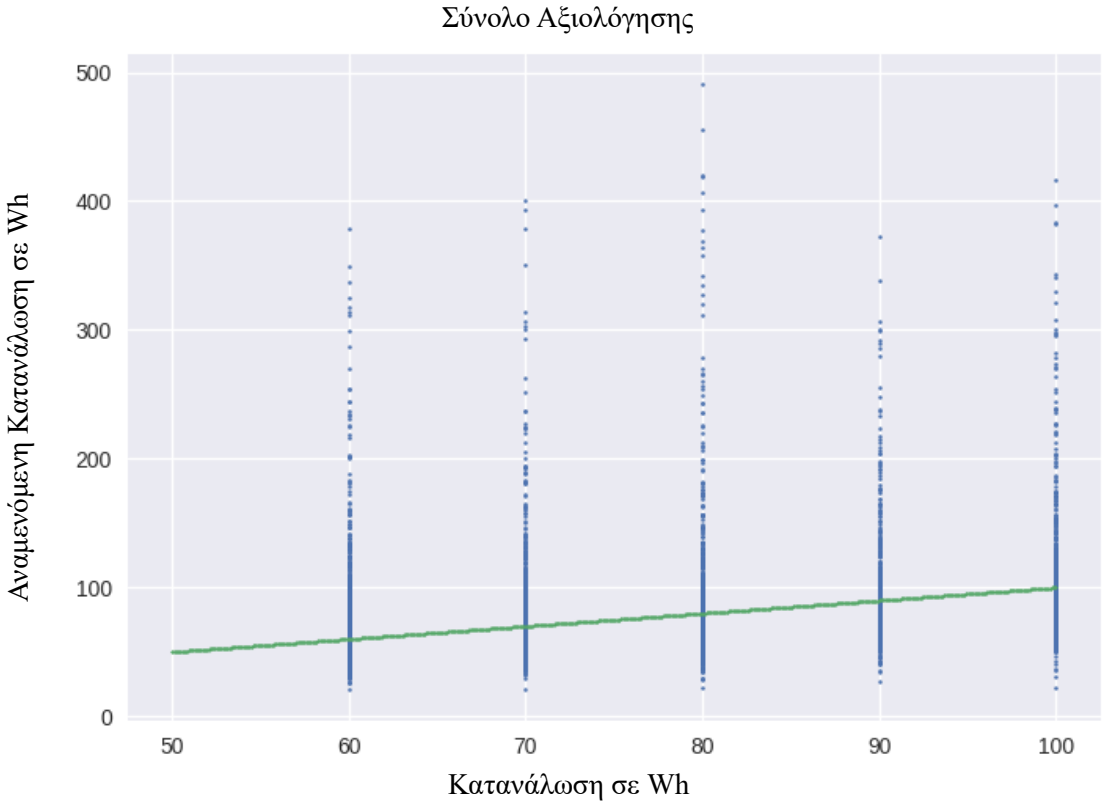
Σχήμα 36: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές από 0 έως 50 Wh



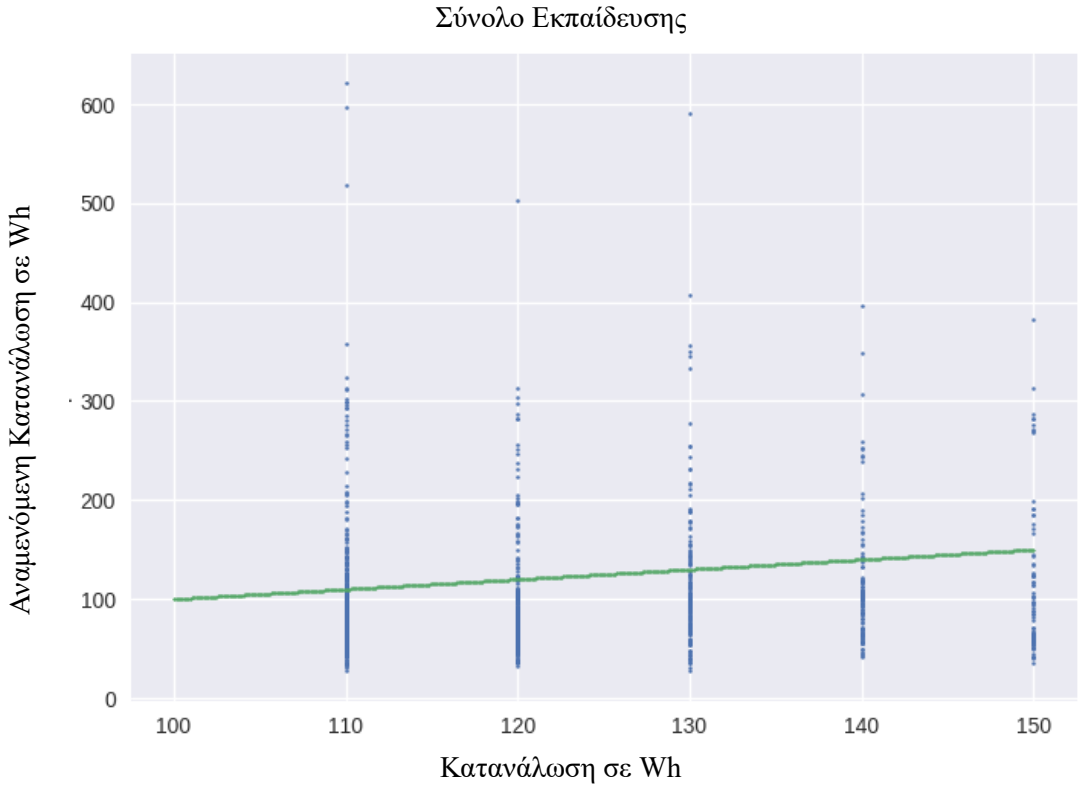
Σχήμα 37: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές από 0 έως 50 Wh



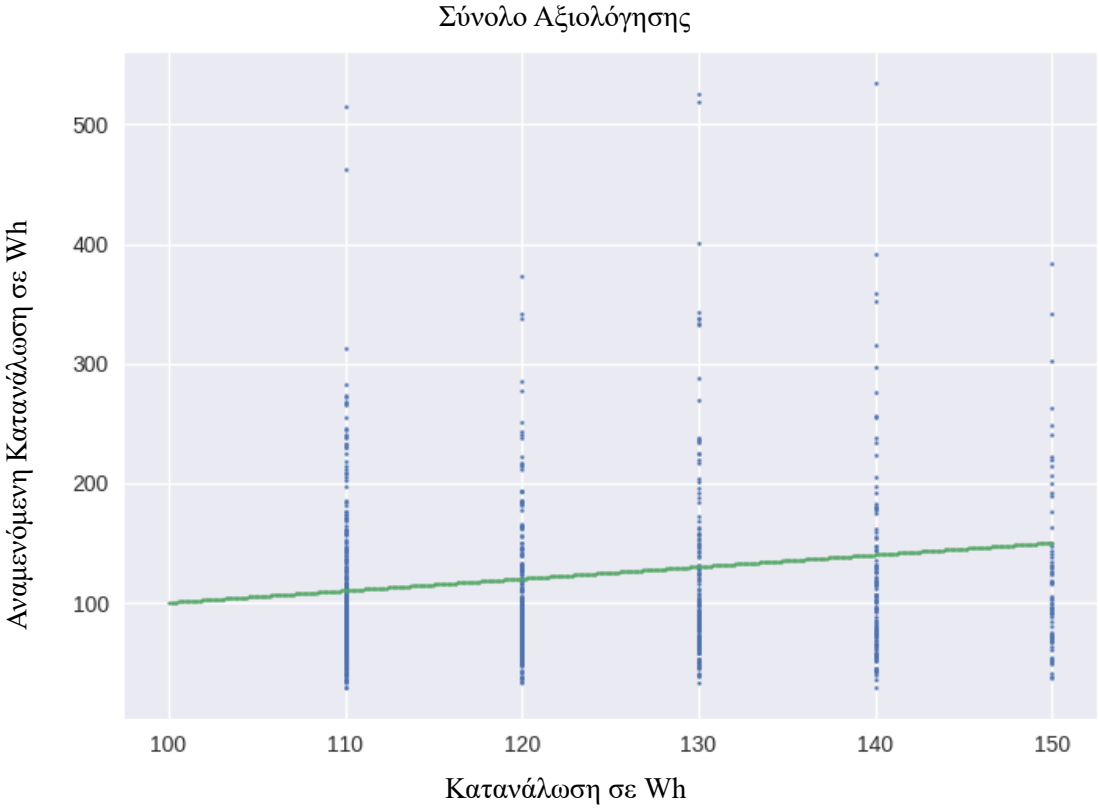
Σχήμα 38: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές από 50 έως 100 Wh



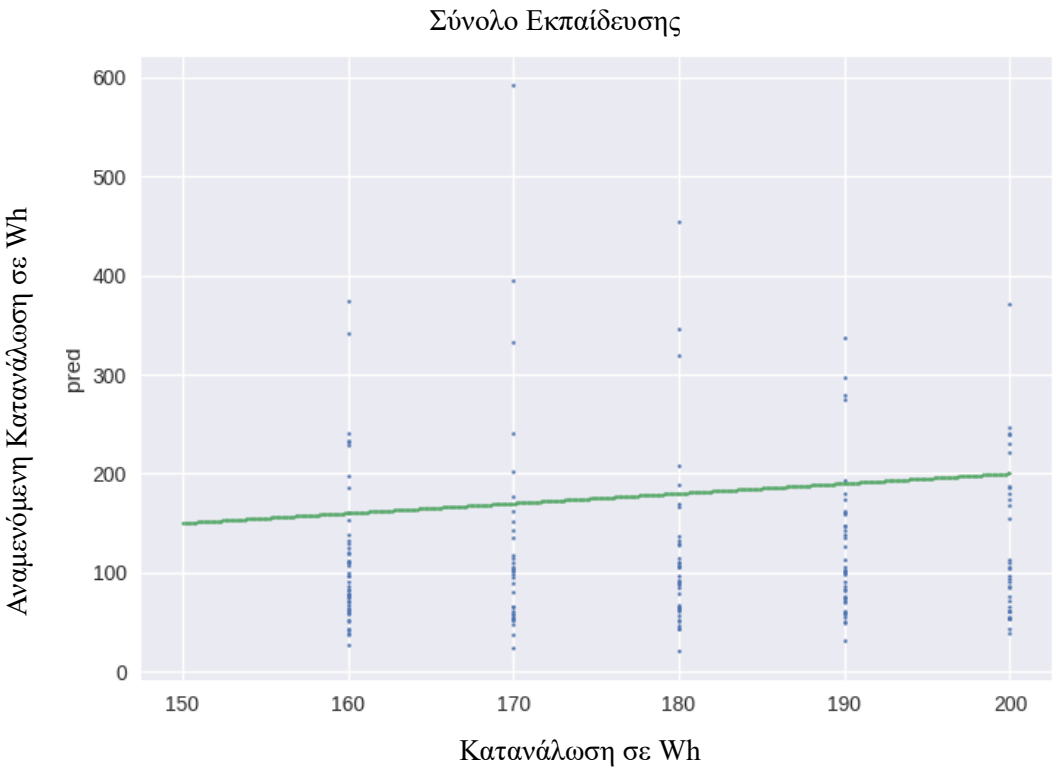
Σχήμα 39: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές από 50 έως 100 Wh



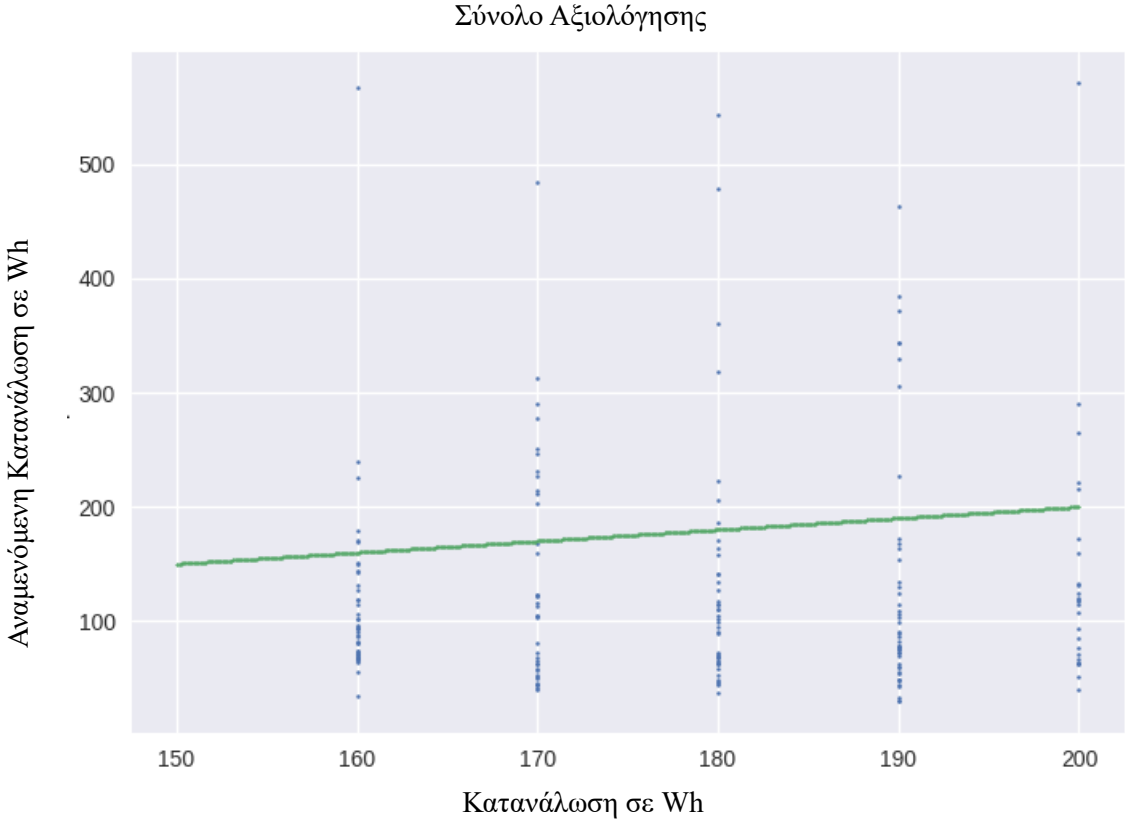
Σχήμα 40 : Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές από 100 έως 150 Wh



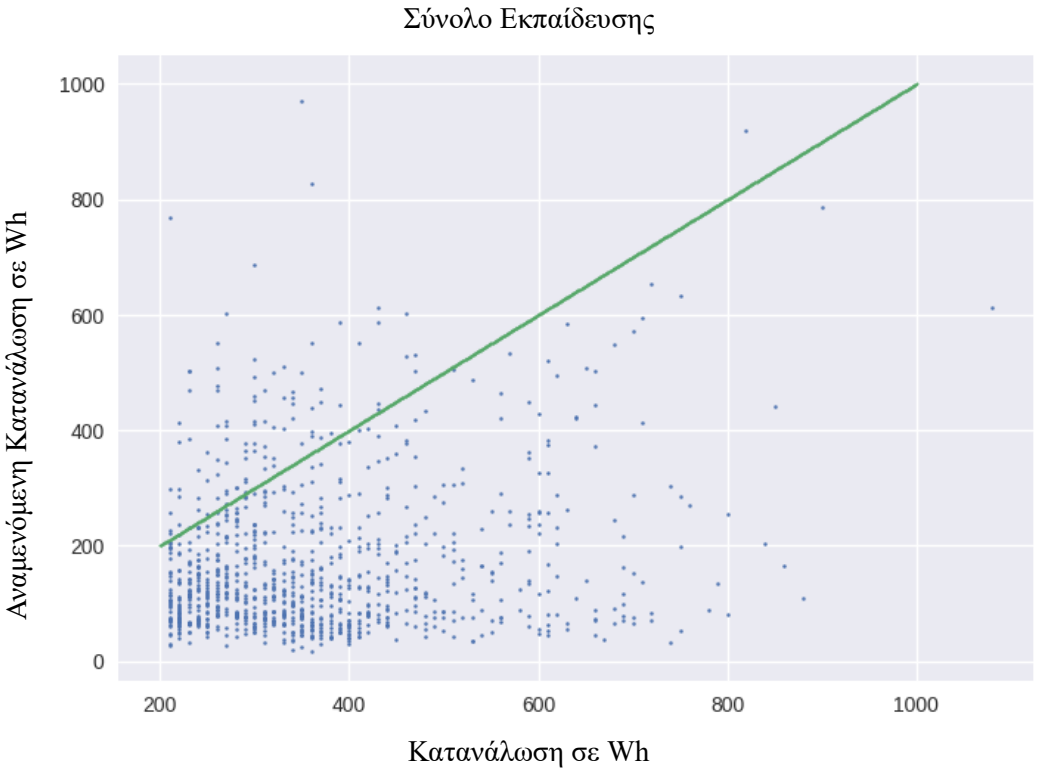
Σχήμα 41: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές από 100 έως 150 Wh



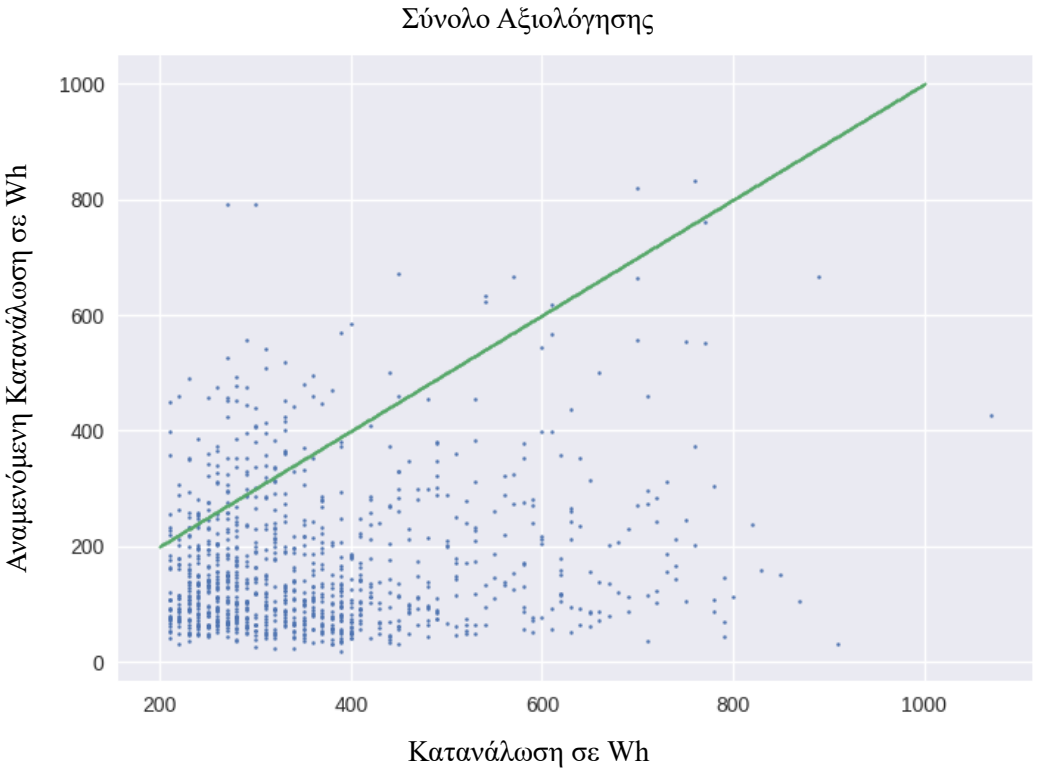
Σχήμα 42: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές από 150 έως 200 Wh



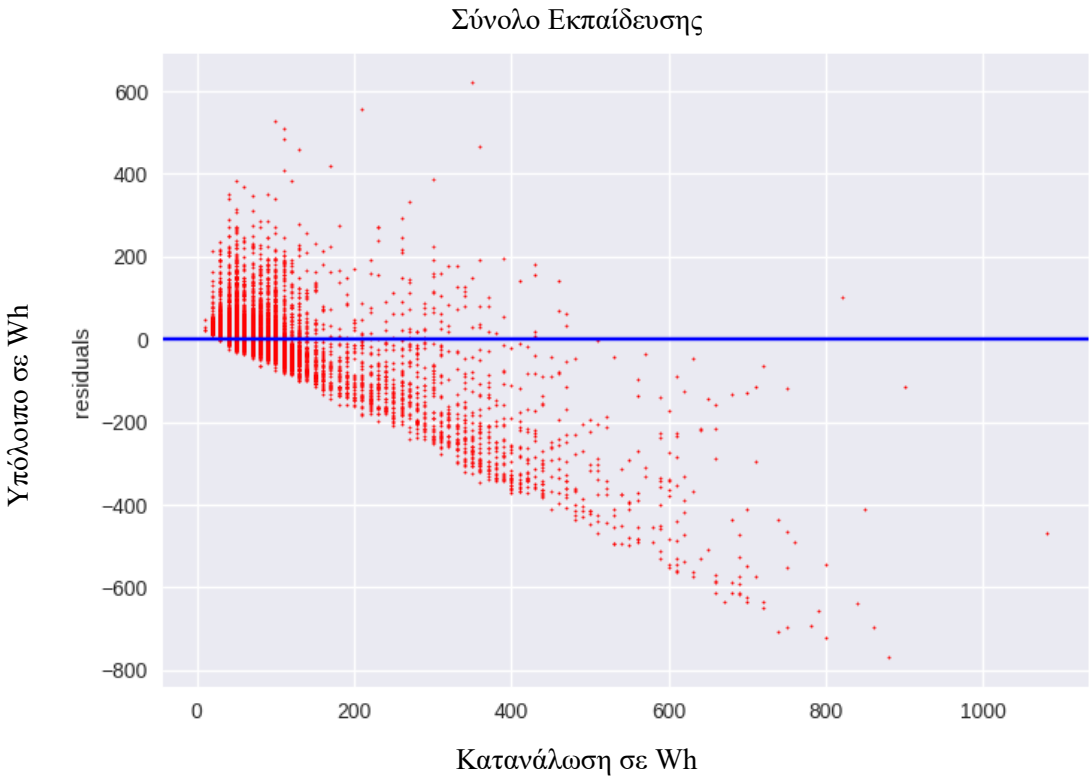
Σχήμα 43: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές από 150 έως 200 Wh



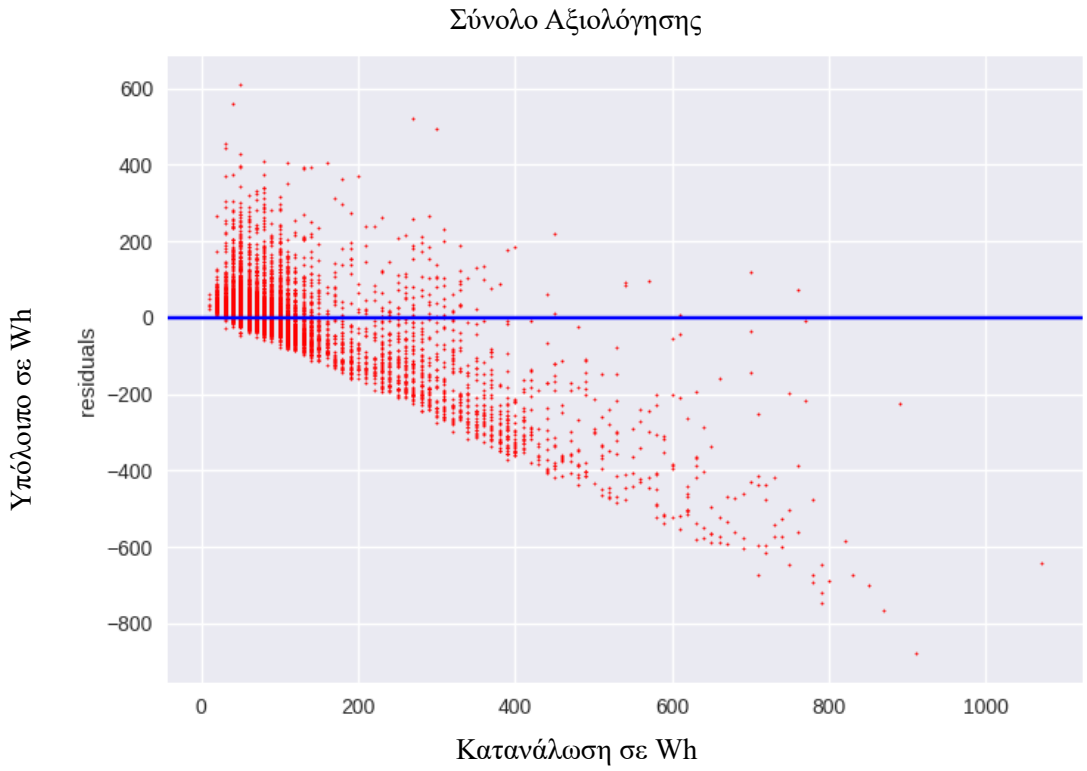
Σχήμα 44: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές μεγαλύτερες από 200 Wh



Σχήμα 45: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές μεγαλύτερες από 200 Wh



Σχήμα 46: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο εκπαίδευσης

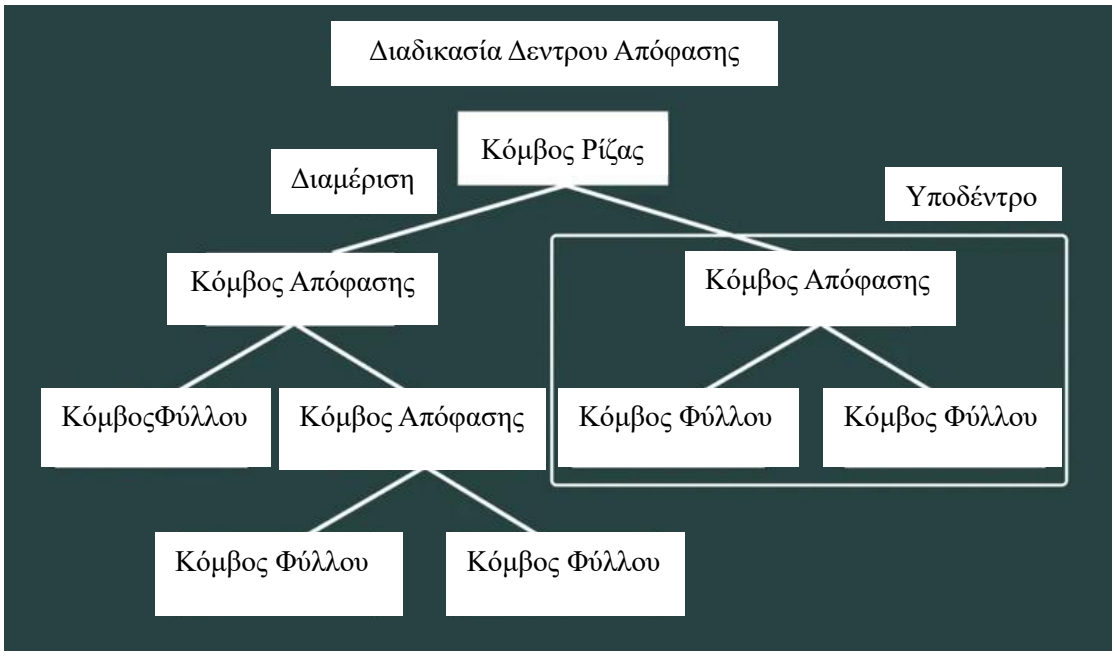


Σχήμα 47: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο αξιολόγησης

8 Ενδυναμωμένο Δέντρο Απόφασης

8.1 Ένα Δέντρο Απόφασης

Το δέντρο αποφάσεων δημιουργεί μοντέλα παλινδρόμησης ή ταξινόμησης με δενδρική δομή. Χωρίζει το σύνολο δεδομένων σε όλο και μικρότερα υποσύνολα, ενώ ταυτόχρονα αναπτύσσεται σταδιακά ένα σχετικό δέντρο αποφάσεων. Το τελικό αποτέλεσμα είναι ένα δέντρο με κόμβους απόφασης και κόμβους φύλλων. Ο κόμβος ρίζας αντιπροσωπεύει ολόκληρο το σύνολο δεδομένων. Ένας κόμβος απόφασης έχει δύο ή περισσότερες διακλαδώσεις, καθεμία από τις οποίες αντιπροσωπεύει τιμές για το εξεταζόμενο χαρακτηριστικό. Ένας κόμβος, ο οποίος διαιρείται σε υποκόμβους, ονομάζεται γονέας κόμβος υποκόμβων, ενώ οι υποκόμβοι είναι το παιδί του γονέα κόμβου. Στο παρακάτω σχήμα, ο κόμβος απόφασης είναι ο γονέας των τερματικών κόμβων (παιδί). Ο κόμβος φύλλου (στην περίπτωση μας η συνολική κατανάλωση ενέργειας από οικιακές συσκευές) αντιπροσωπεύει μια απόφαση σχετικά με την τιμή, την οποία θέλουμε να προβλέψουμε. Χρήσιμος όρος είναι εξίσου το κλάδεμα, δηλαδή η αφαίρεση υποκόμβων ενός κόμβου απόφασης. Το κλάδεμα γίνεται συχνά στα δέντρα αποφάσεων για να αποφευχθεί η υπερβολική προσαρμογή. Τα δέντρα αποφάσεων μπορούν να χειριστούν τόσο κατηγορικά όσο και αριθμητικά δεδομένα.

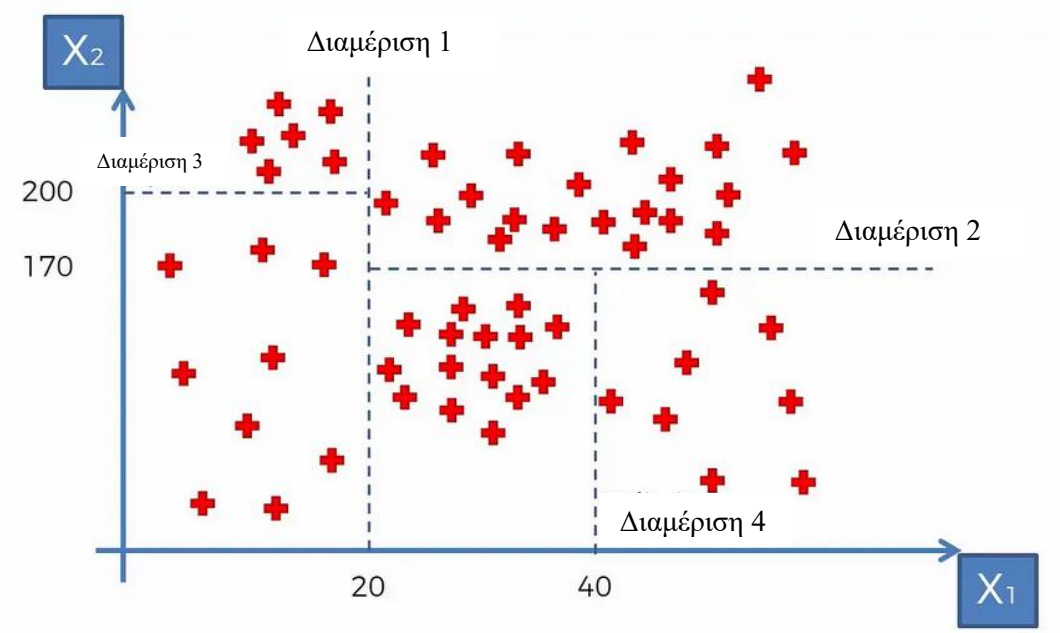


Σχήμα 48: Δέντρο Απόφασης

Η κατασκευή των δέντρων απόφασης λειτουργεί συνήθως από πάνω προς τα κάτω, επιλέγοντας σε κάθε βήμα μια μεταβλητή που χωρίζει καλύτερα το σύνολο των στοιχείων. Κάθε υποδέντρο του μοντέλου μπορεί να αναπαρασταθεί ως δυαδικό δέντρο, όπου ένας κόμβος απόφασης χωρίζεται σε δύο κόμβους με βάση τις συνθήκες.

Ένα δέντρο απόφασης κατασκευάζεται με αναδρομική κατάτμηση - ξεκινώντας από τον κόμβο - ρίζα (γνωστό ως πρώτος γονέας), κάθε κόμβος μπορεί να χωριστεί σε αριστερούς και δεξιούς κόμβους - παιδιά. Αυτοί οι κόμβοι μπορούν στη συνέχεια να χωριστούν περαιτέρω και να γίνουν οι ίδιοι κόμβοι - γονείς των κόμβων-παιδιών που προκύπτουν.

Όπως μπορούμε να δούμε παρακάτω είναι ένα διάγραμμα διασποράς του συνόλου δεδομένων, στο οποίο όλα τα σημεία δεδομένων είναι σημειωμένα στο επίπεδο, όπου x1 και x2 είναι ανεξάρτητες μεταβλητές και Y είναι η εξαρτημένη μεταβλητή που θέλουμε να προβλέψουμε (καθώς αυτό είναι ένα δισδιάστατο διάγραμμα ο άξονας y δεν είναι ορατός).



Σχήμα 49: Διαμέριση του Χώρου σε Παραλληλόγραμμα

Κριτήριο μας αποτελεί το να αυξάνουμε την ομοιογένεια ή καθαρότητα, δηλαδή τα ταξινομούμενα στοιχεία του δείγματός μας να δείχνουν «προτίμηση» σε ένα συγκεκριμένο κόμβο. Είναι απαραίτητο να ορίσουμε ένα μέτρο μη καθαρότητας, ώστε να μπορέσουμε να ποσοτικοποιήσουμε αυτή την έννοια. Έστω P(wi|t) συμβολίζουμε την πιθανότητα ενός διανύσματος Xt που σχετίζεται με τον κόμβο T να είναι σωστά ταξινομημένο.

Ορίζουμε τη μη καθαρότητα του κόμβου t:

$$I(t) = -\sum_{i=1}^{N} P(\omega_i|t) log P(\omega_i|t)$$

Οι πιθανότητες εκτιμώνται με τα αντίστοιχα ποσοστά εμφάνισης στο σύνολο εκπαίδευσης. Ορίζουμε NtY και NtN ότι τα σημεία στέλνονται στον κόμβο “yes” ή “no” αντίστοιχα. Τότε η μείωση στη μη καθαρότητα είναι:

$$\Delta I(t) = I(t) - \frac{N_{tY}}{N_t} I(t_Y) - \frac{N_{tN}}{N_t} I(t_N)$$

Όπου Nt είναι το πλήθος όλων των σημείων και I(tY), I(tN) είναι οι μη καθαρότητες των νέων κόμβων. Ο στόχος είναι να επιλέξουμε τις κατάλληλες ερωτήσεις, οι οποίες οδηγούν στην υψηλότερη μείωση της μη καθαρότητας. Τέλος, επιλέγουμε ένα κριτήριο τερματισμού, μέχρι ένας κόμβος να χαρακτηριστεί ως φύλλο (δηλαδή ως τελικός κόμβος).

Πλεονέκτημα της μεθόδου αποτελεί η διαφάνεια της διαδικασίας απόφασης, αλλά και το γεγονός ότι αδιαφορεί προς χαρακτηριστικές μεταβλητές με μικρό επίπεδο σημαντικότητας. Ωστόσο, είναι ευαίσθητη σε στατιστικές διακυμάνσεις του δείγματος εκπαίδευσης.

8.2 Ενδυναμωμένα Δέντρα Απόφασης (Δάσος)

Με τη μέθοδο αυτή (BDT) χρησιμοποιούμε ουσιαστικά περισσότερα του ενός δέντρα, για να κάνουμε την πρόβλεψη. Για να πραγματοποιηθεί κάτι τέτοιο, χρησιμοποιούμε πολλά δέντρα απόφασης μικρού βάθους με τα ίδια δεδομένα εκπαίδευσης. Αν J(x;u) είναι το αποτέλεσμα ενός δέντρου, όπου x ένα διάνυσμα χαρακτηριστικών και u ένα διάνυσμα παραμέτρων, τότε για πολλά δέντρα απόφασης θα ισχύει:

$$F(x) = \sum_{} a_k J(x; u_k)$$

όπου α_κ τα αντίστοιχα βάρη και f(x)=sign(F(x)). Δηλαδή με ίδια δεδομένα εκπαιδεύουμε πολλούς εκτιμητές και παίρνουμε ένα άθροισα αυτών με βάρη α_κ.

Σε αυτή τη διαδικασία, για να προσδιορίσουμε τα βάρη πρέπει να ελαχιστοποιήσουμε τη συνάρτηση κόστους:

$$T(a_k; u_k) = \sum_{i=1}^{N} \exp(-y_i F(x_i))$$

Με αυτή τη συνάρτηση κόστους τα λάθος ταξινομημένα στοιχεία βαρύνονται πολύ περισσότερο (penalty) σε σχέση με αυτά με σωστή ταξινόμηση.

Έτσι, ορίζοντας το μερικό άθροισμα

$$F_m(x) = \sum_{k=1}^{m} a_k J(x; u_k)$$

,

όπου

$$F_m(x) = F_{m-1}(x) + a_m J(x; u_m)$$

και έτσι μπορούμε να γράψουμε τη συνάρτηση κόστους ως:

$$T(a_m; u_m) = \sum_{i=1}^{N} w_i^m \exp(-y_i a_m J(x_i; u_m))$$

οπότε το κάθε διάνυσμα x_i αποκτά βάρος w_i, το οποίο σχετίζεται με το προηγούμενο βήμα κάθε φορά.

Για την εκπαίδευση στο βήμα m βρίσκουμε τα u_m και ελαχιστοποιούμε το σφάλμα P_m:

$$P_m = \sum_{y_i J(x_i; u_m) < 0} w_i^m .$$

Μπορούμε τώρα να γράψουμε τη συνάρτηση κόστους σαν συνάρτηση του σφάλματος ταξινόμησης:

$$T(a_m, u_m) = e^{-a_m}(1 - P_m) + e^{a_m} P_m$$

με σημείο ελαχίστου

$$a_m = \frac{1}{2} \ln(\frac{1 - P_m}{P_m}) .$$

Τα βάρη w^{m+1} υπολογίζονται βελτιστοποιώντας τις τιμές των P_m και α_m:

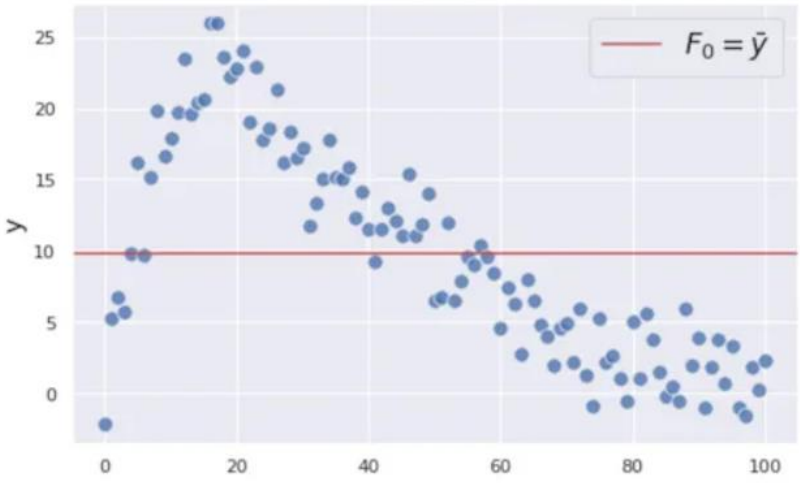
$$w_i^{m+1} = \frac{\exp(-y_i F_m(x_i))}{Z_m} = \frac{w_i^m \exp(-y_i a_m J(x_i; u_m))}{Z_m}$$

όπου Z_m είναι παράγοντας κανονικοποίησης για το άθροισμα των βαρών στο m+1 βήμα να ισούται με 1.

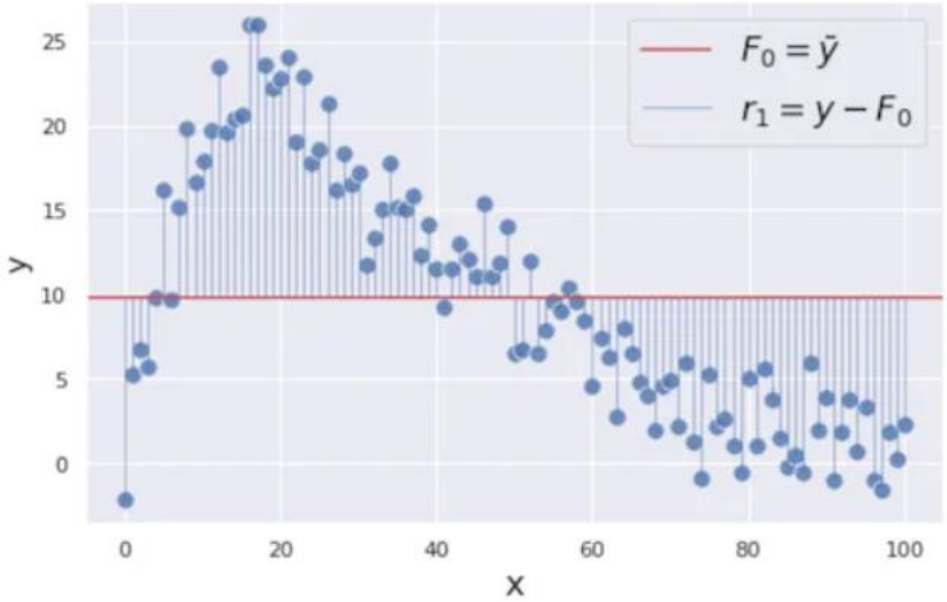
8.3 Gradient Boosting Decision Trees

Όταν ένα δέντρο απόφασης είναι ο αδύναμος εκτιμητής, ο αλγόριθμος που προκύπτει ονομάζεται gradient- boosted trees. Το μοντέλο αυτό χτίζεται σταδιακά, όπως και άλλες μέθοδοι ενίσχυσης, αλλά γενικεύει και συνδυάζει άλλες μεθόδους με σκοπό τη βέλτιστη προσαρμογή και τα καλύτερα αποτελέσματα.

Έχουμε y_i την παρατηρούμενη τιμή y_i, $\hat{y}_i$ την αναμενόμενη τιμή του y_i, και n το πλήθος του δείγματος. Τώρα, ας θεωρήσουμε έναν αλγόριθμο ενίσχυσης κλίσης με M στάδια. Σε κάθε στάδιο m ($1 \leq m \leq M$) του gradient boosting, υποθέτουμε κάποιο ατελές μοντέλο F_m (για μικρό m, αυτό το μοντέλο μπορεί απλά να επιστρέφει $\hat{y}_i = \bar{y}$).

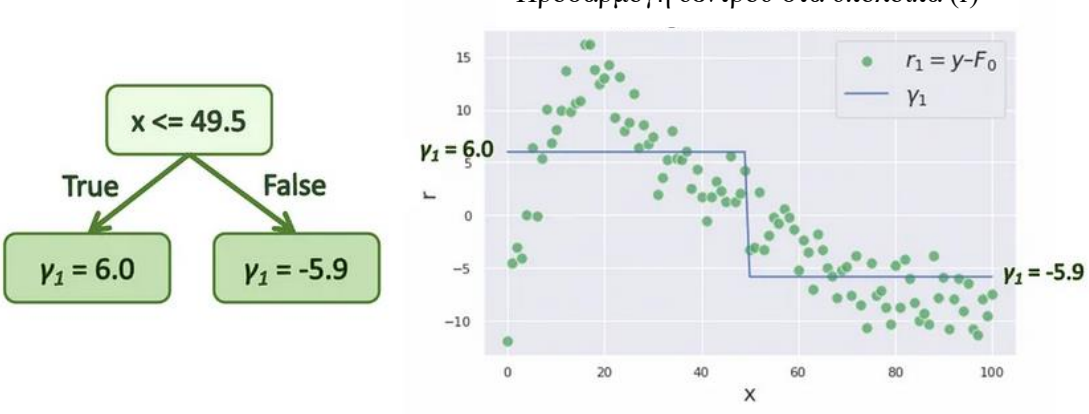


Σχήμα 50: $F_o=\bar{y}$



Σχήμα 51: Προσαρμογή r_1

Προσαρμογή δέντρου στα υπόλοιπα (r)



Σχήμα 52: Προσαρμογή υπολοίπων

Η πρόβλεψη γ_1 εντάσσεται στο μοντέλο, ώστε να μειωθεί το υπόλοιπο σφάλματος.

Ορίζουμε ως ν το ρυθμό εκμάθησης του μοντέλου, ο οποίος παίρνει τιμές από 0 μέχρι 1.

$$F_1 = F_0 + \nu \cdot \gamma_1$$

Αυτό συνεχίζεται και σε γλώσσα μαθηματικών μεταφράζεται ως εξής:

$$F_0(x) = \underset{\gamma}{argmin} \sum_{i=1}^{n} L(y_i, \gamma)$$

Εδώ F_o είναι η αρχική σταθερά που ορίστηκε και L η συνάρτηση σφάλματος

$$L = (y_i - \gamma)^2$$

για argmin ψάχνουμε το γ που να ελαχιστοποιεί τη συνάρτηση σφάλματος

$$\begin{aligned}
\frac{\partial}{\partial \gamma} \sum_{i=1}^{n} L &= \frac{\partial}{\partial \gamma} \sum_{i=1}^{n} (y_i - \gamma)^2 \\
&= -2 \sum_{i=1}^{n} (y_i - \gamma) \\
&= -2 \sum_{i=1}^{n} y_i + 2n\gamma
\end{aligned}$$

$$\begin{aligned}
-2 \sum_{i=1}^{n} y_i + 2n\gamma &= 0 \\
n\gamma &= \sum_{i=1}^{n} y_i \\
\gamma &= \frac{1}{n} \sum_{i=1}^{n} y_i = \bar{y}
\end{aligned}$$

Για m από 1 μέχρι M:

Υπολογίζουμε τα υπόλοιπα για i= 1, …, n

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)}$$

$$\begin{aligned}
&= -\frac{\partial (y_i - F_{m-1})^2}{\partial F_{m-1}} \\
&= 2(y_i - F_{m-1})
\end{aligned}$$

Εκπαιδεύουμε το δέντρο παλινδρόμησης με χαρακτηριστικά x ενάντια στα r και φτιάχνουμε τον τερματικό κόμβο σύμφωνα με R_{jm} για j=1, …, J_m, όπου j ο τερματικό κόμβος, m το περιεχόμενο (index) και J το συνολικό πλήθος των φύλλων.

Υπολογίζουμε για για j=1, …, J_m

$$\gamma_{jm} = \underset{\gamma}{argmin} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$$

$$= \underset{\gamma}{argmin} \sum_{x_i \in R_{jm}} (y_i - F_{m-1}(x_i) - \gamma)^2$$

Βρίσκουμε το γ_{jm} που μηδενίζει την παράγωγο των αθροισμάτων ως προς γ

$$\frac{\partial}{\partial \gamma} \sum_{x_i \in R_{jm}} (y_i - F_{m-1}(x_i) - \gamma)^2 = 0$$

$$-2 \sum_{x_i \in R_{jm}} (y_i - F_{m-1}(x_i) - \gamma) = 0$$

$$n_j \gamma = \sum_{x_i \in R_{jm}} (y_i - F_{m-1}(x_i))$$

$$\gamma = \frac{1}{n_j} \sum_{x_i \in R_{jm}} r_{im}$$

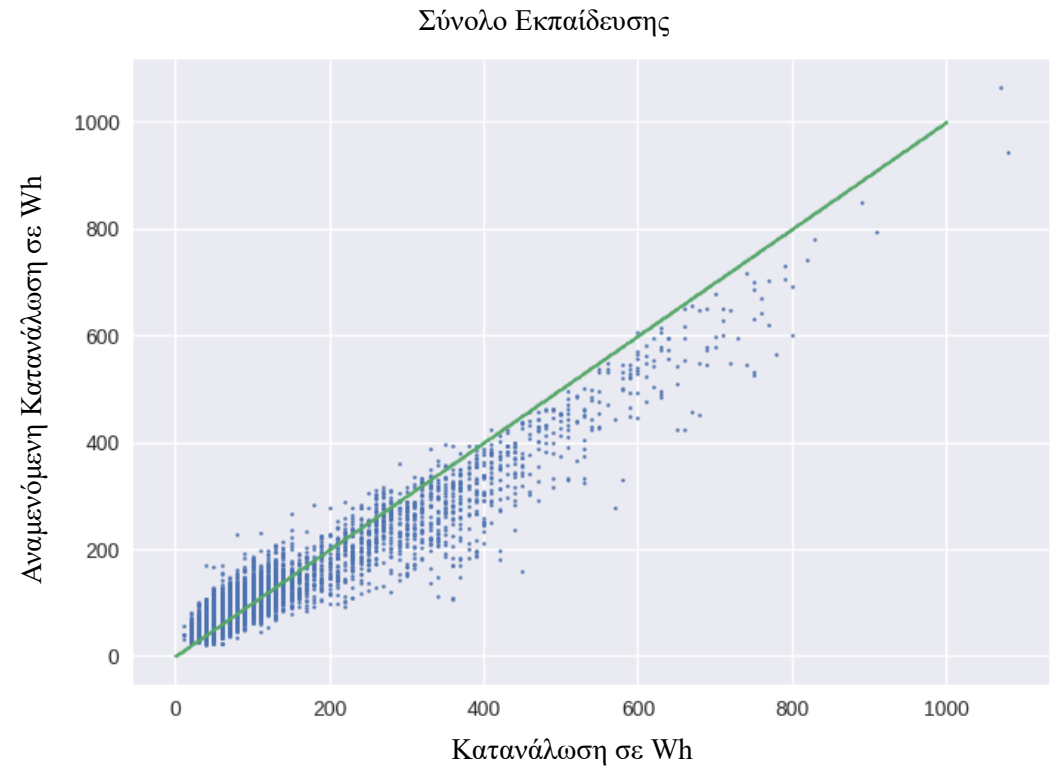
και, τέλος, επαναπροσδιορίζουμε το μοντέλο

$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} 1(x \in R_{jm})$$

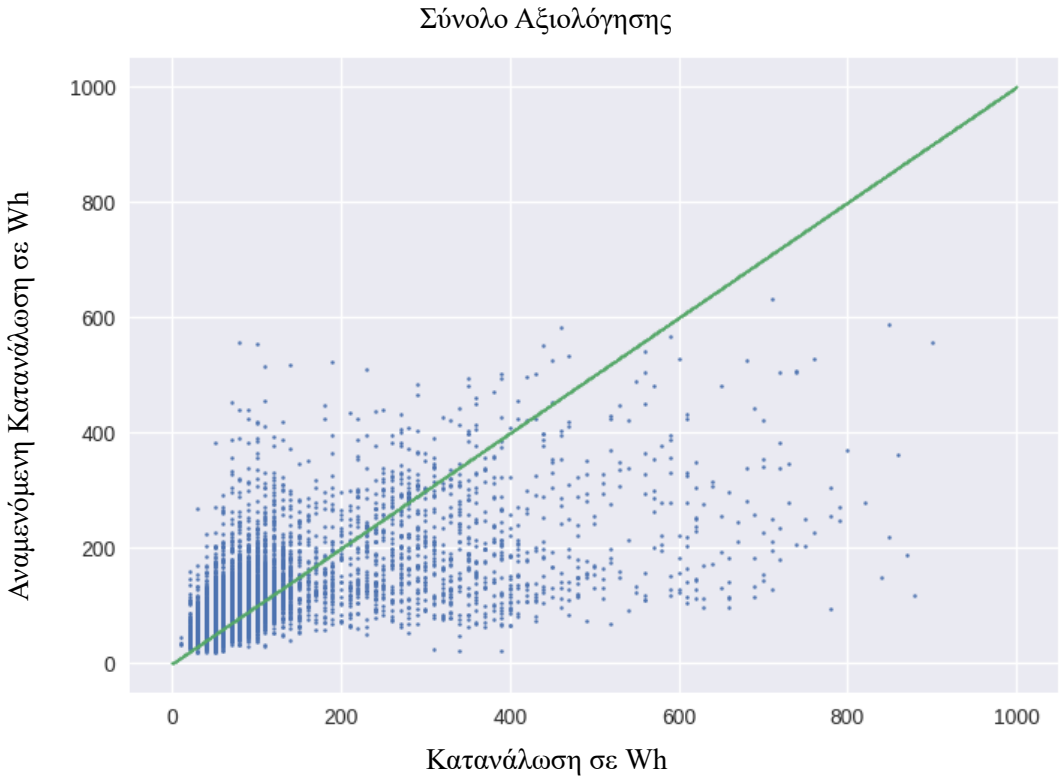
.

Σε αυτή την υποενότητα παρουσιάζονται τα αποτελέσματα που λάβαμε μετά την προσαρμογή και υλοποίηση του Gradient Boosting Regressor με τις καλύτερες προβλέψεις, δηλαδή αυτόν με πλήθος σταδίων boosting 300, ρυθμό εκμάθησης 0.15 και μέγιστο βάθος 5.

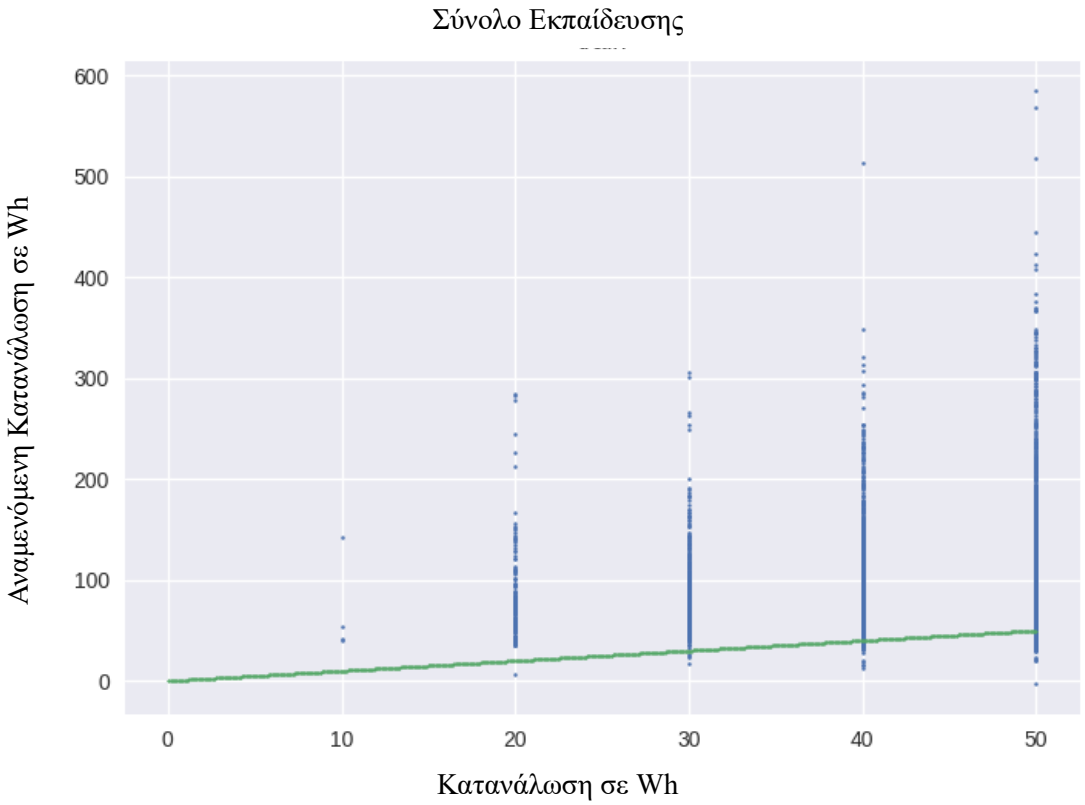
Όπως και προηγουμένως, στα παρακάτω σχήματα η πράσινη ευθεία (όπου η πραγματική τιμή ισούται με την αναμενόμενη τιμή) και η μπλε (όπου τα υπόλοιπα πραγματικής μείον αναμενόμενης τιμής είναι μηδενικά) είναι βοηθητικές οπτικά.



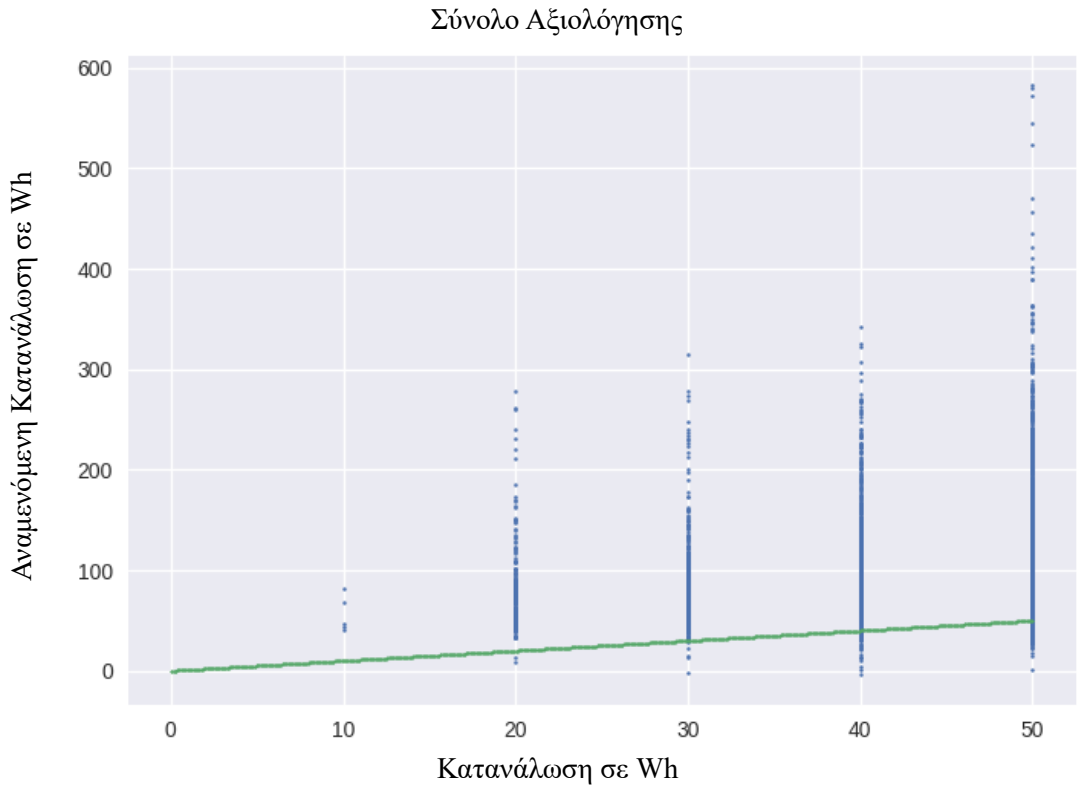
Σχήμα 53: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης



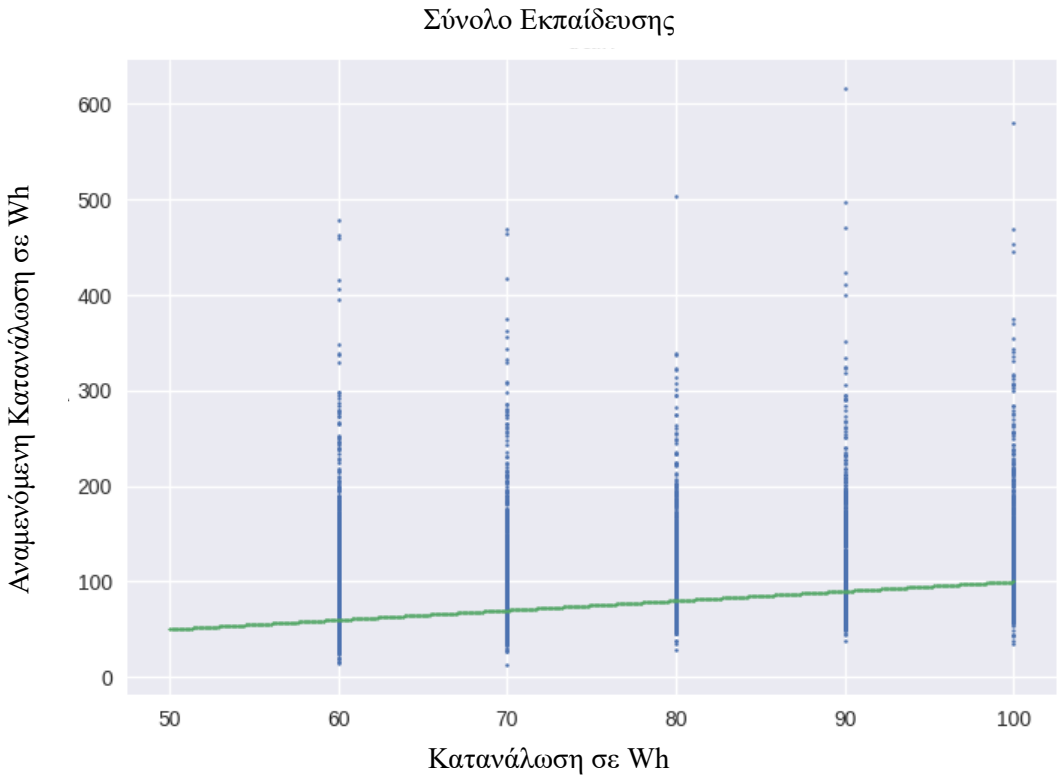
Σχήμα 54: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης



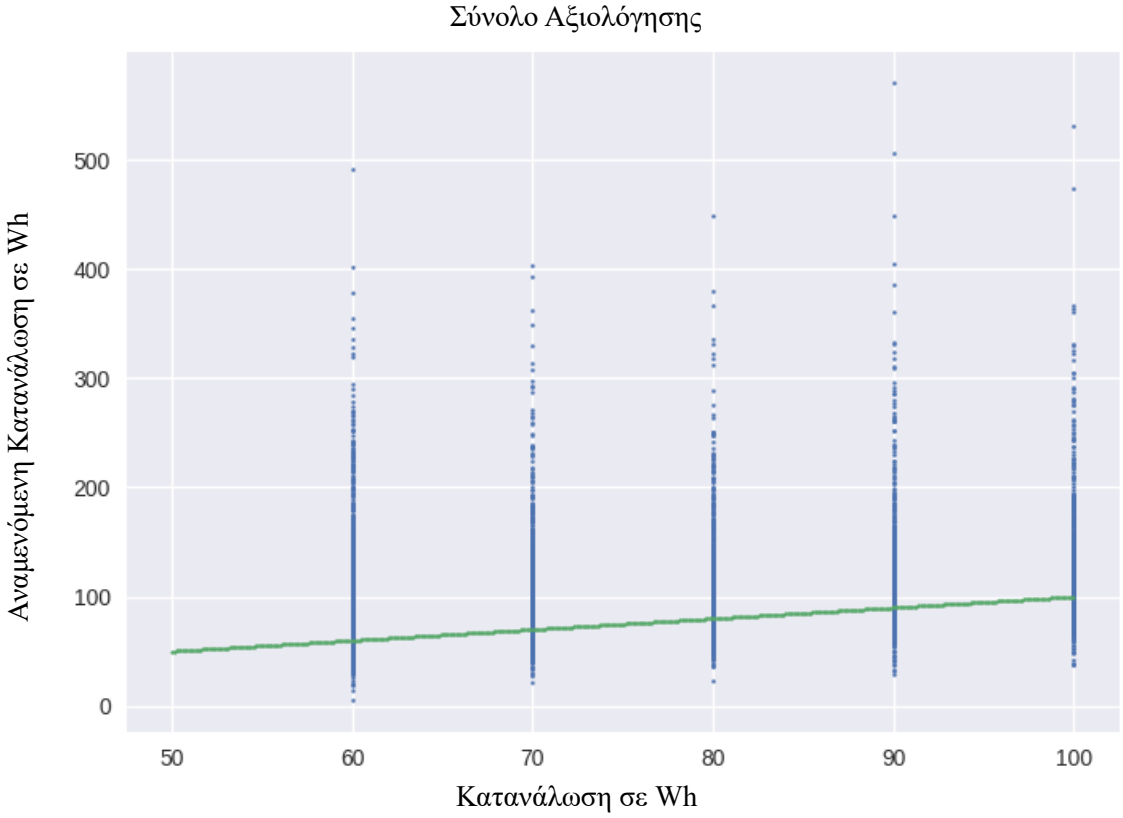
Σχήμα 55: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές από 0 έως 50 Wh



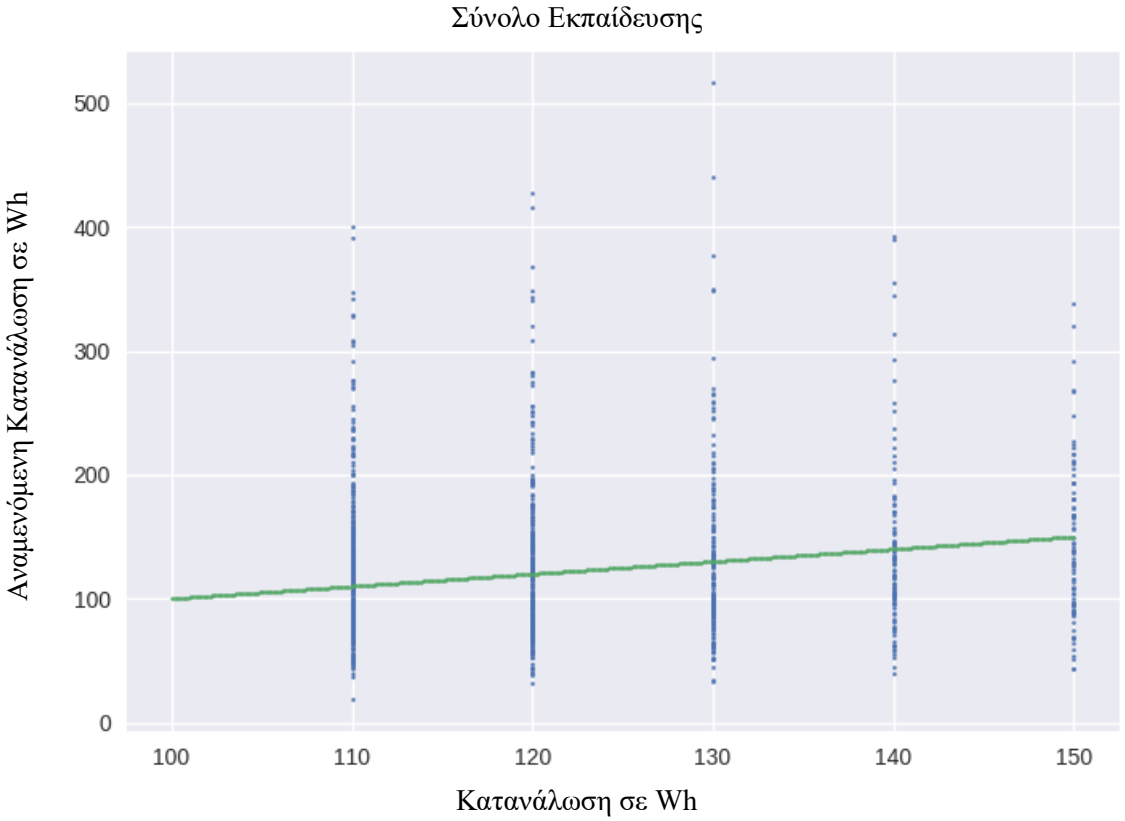
Σχήμα 56: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές από 0 έως 50 Wh



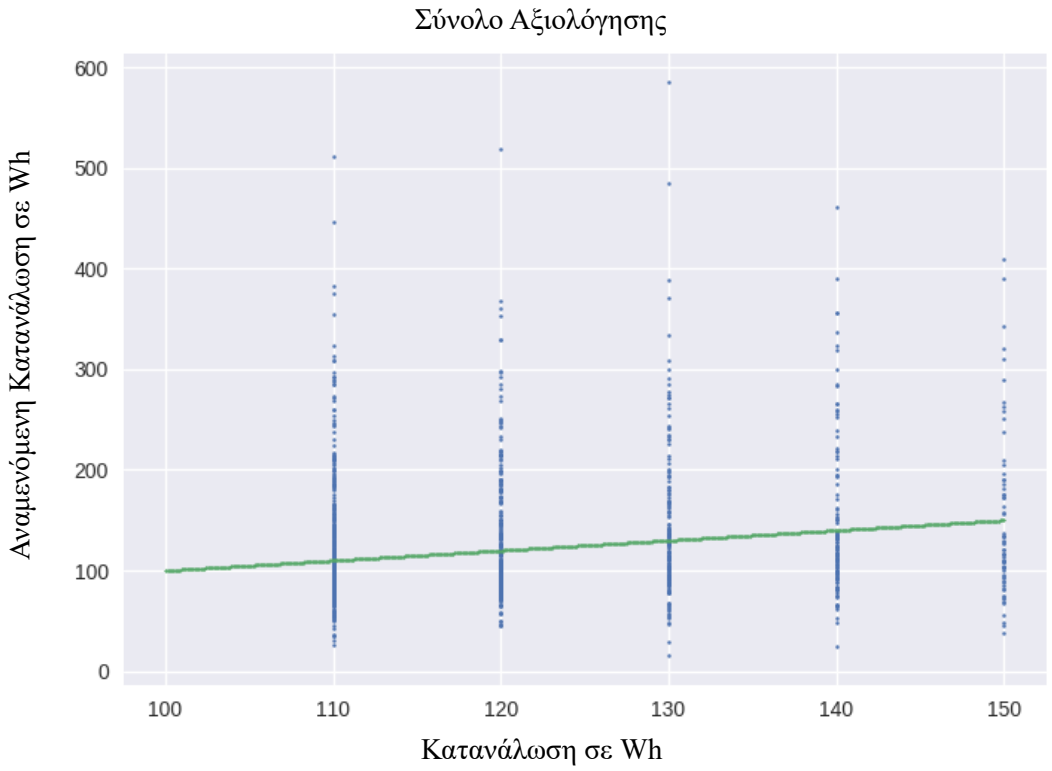
Σχήμα 57: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές από 50 έως 100 Wh



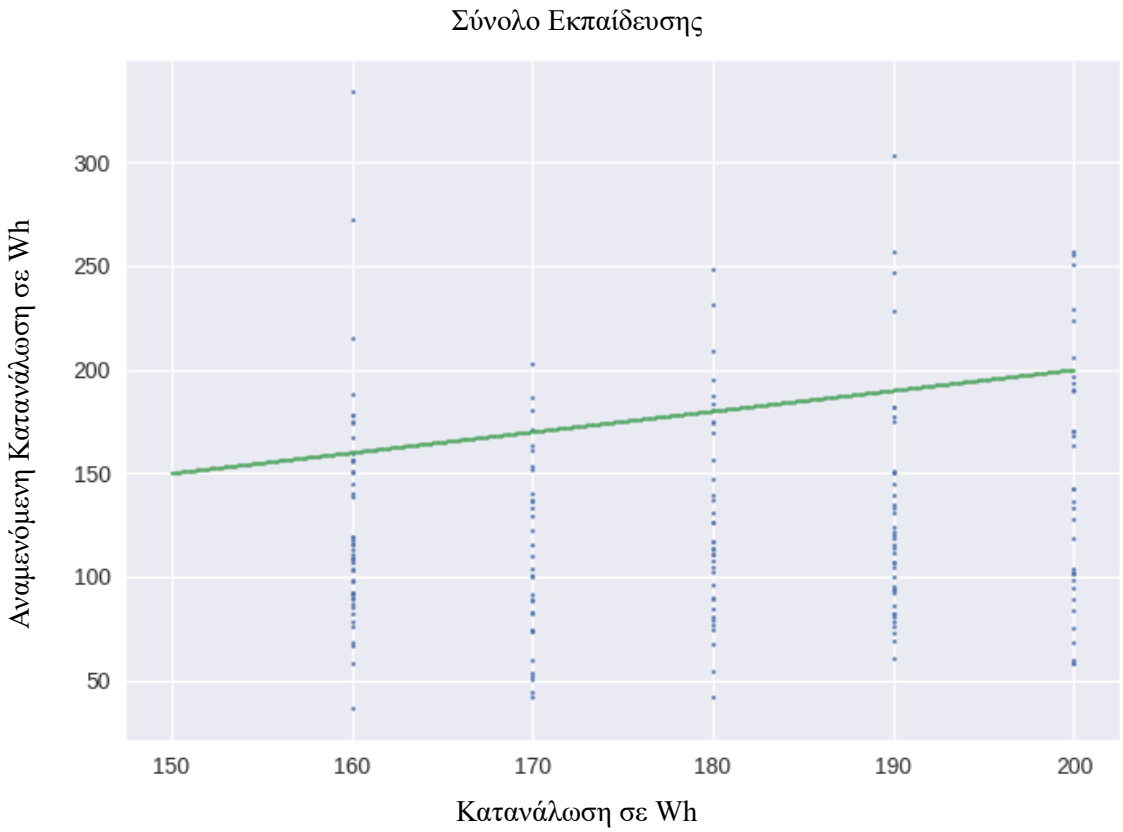
Σχήμα 58: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές από 50 έως 100 Wh



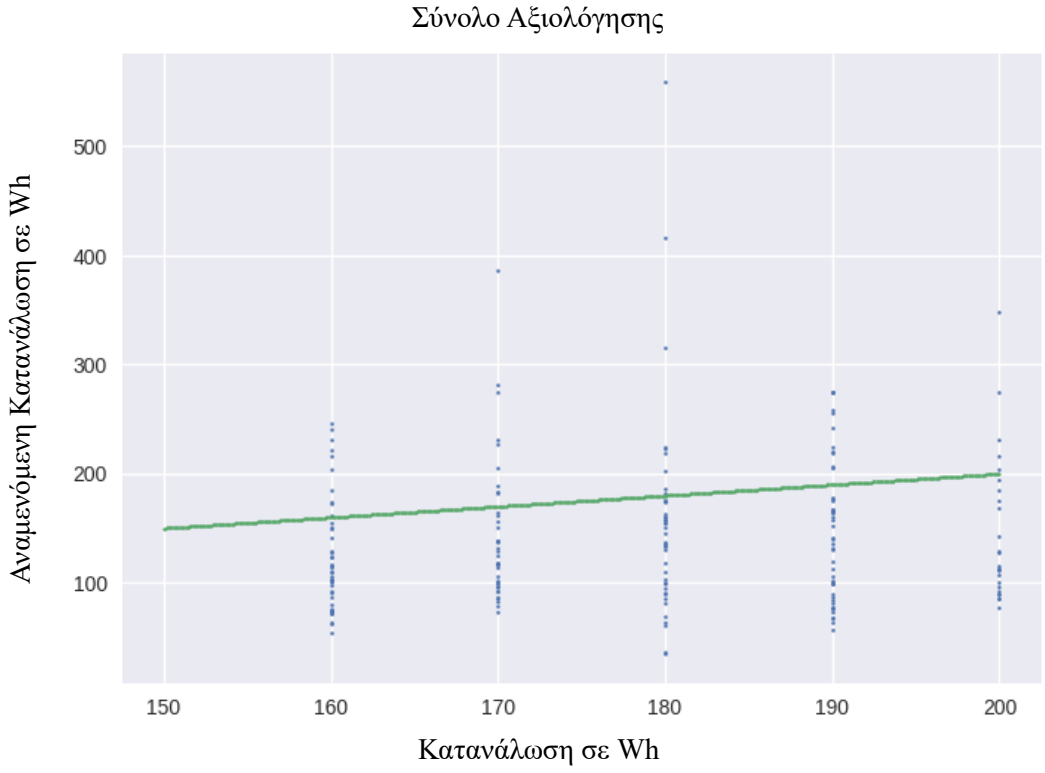
Σχήμα 59: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές από 100 έως 150 Wh



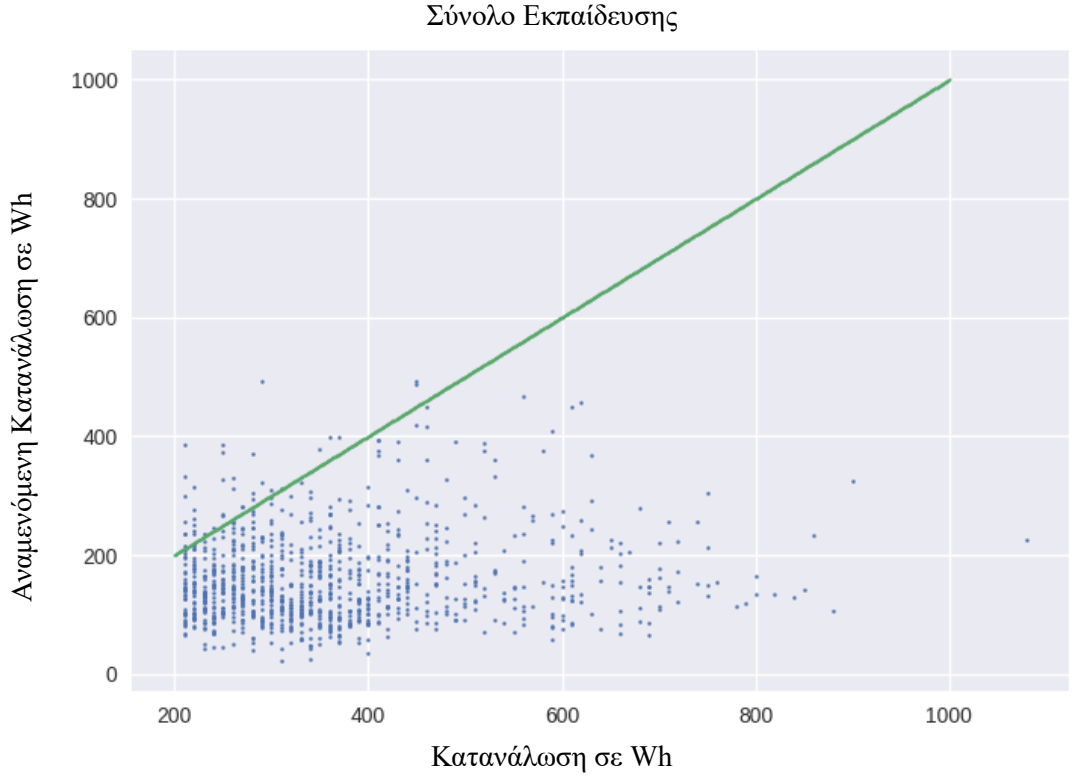
Σχήμα 60: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές από 100 έως 150 Wh



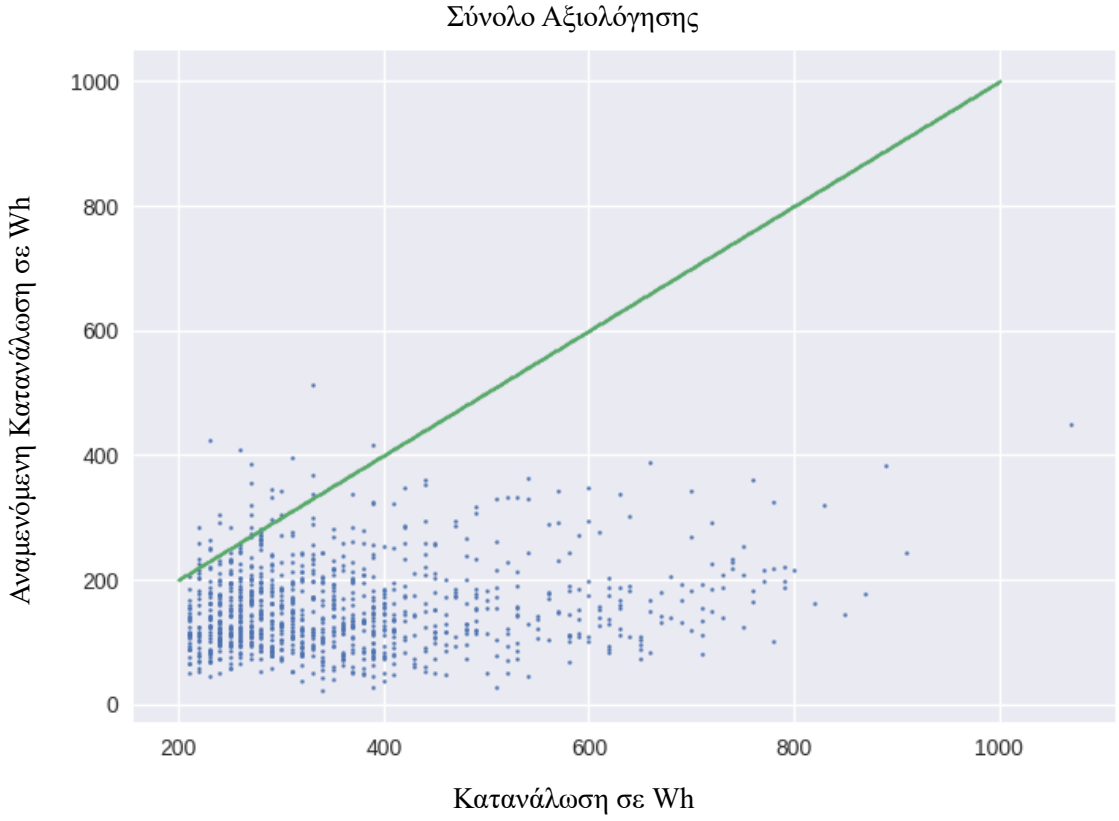
Σχήμα 61: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές από 150 έως 200 Wh



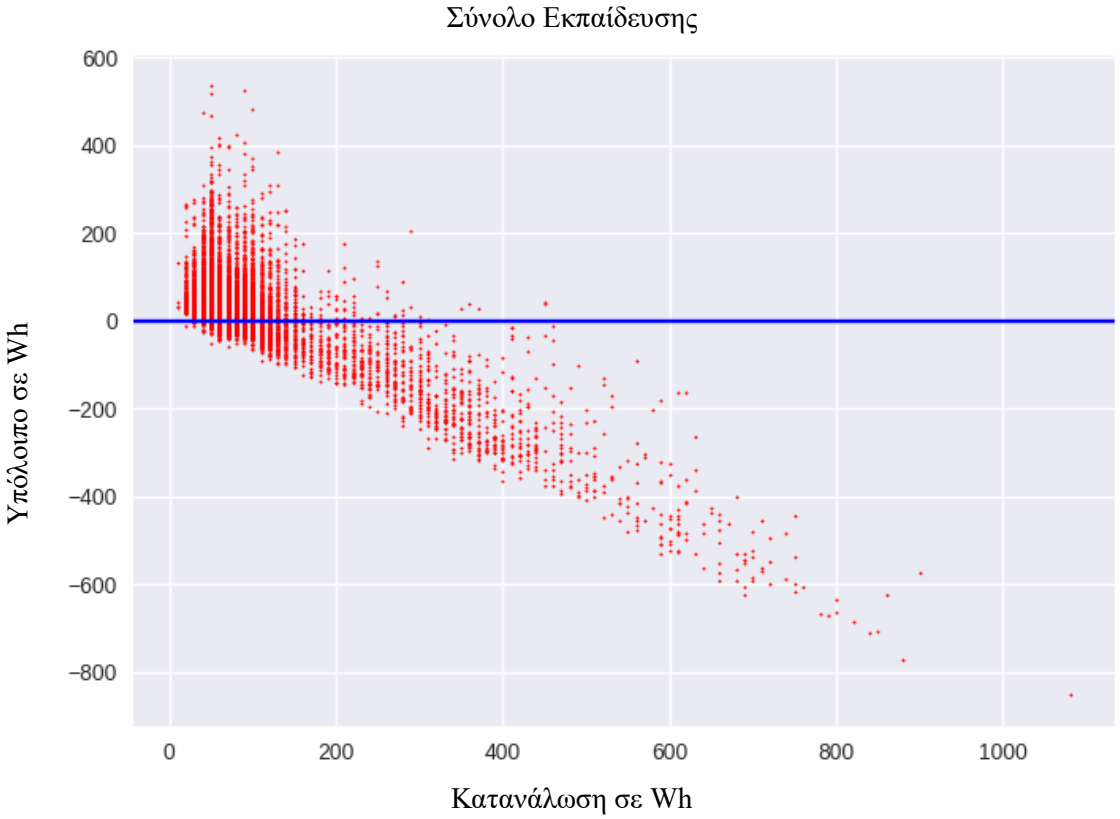
Σχήμα 62: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές από 150 έως 200 Wh



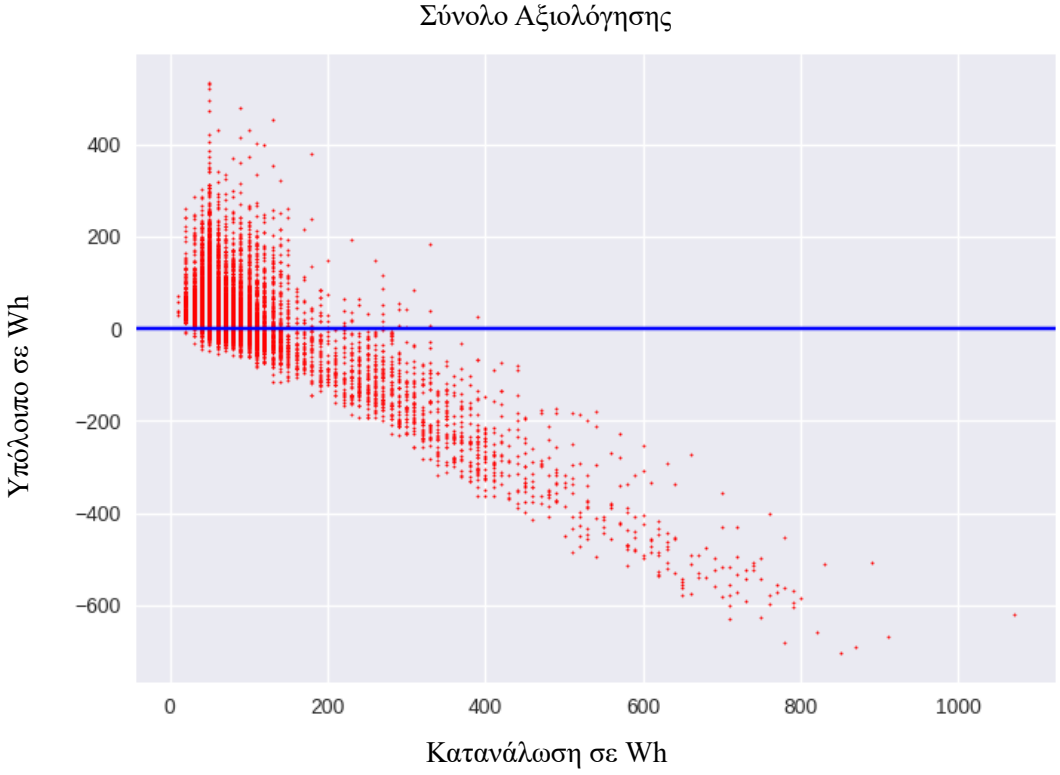
Σχήμα 63: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές πάνω από 200 Wh



Σχήμα 64: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές πάνω από 200 Wh



Σχήμα 65: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο εκπαίδευσης



Σχήμα 66: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο αξιολόγησης

8.4 Random Forest Regressor

Τα τυχαία δάση ή τυχαία δάση απόφασης είναι μια μέθοδος μάθησης του συνόλου για ταξινόμηση, παλινδρόμηση και άλλες εργασίες που λειτουργεί με την κατασκευή ενός πλήθους δέντρων απόφασης κατά τη διάρκεια της εκπαίδευσης. Για εργασίες ταξινόμησης, η έξοδος του τυχαίου δάσους είναι η κλάση που επιλέγεται από τα περισσότερα δέντρα. Για τις εργασίες παλινδρόμησης, επιστρέφεται ο μέσος όρος ή η μέση πρόβλεψη των μεμονωμένων δέντρων. Τα τυχαία δάση απόφασης διορθώνουν τη συνήθεια των δέντρων απόφασης να προσαρμόζονται υπερβολικά στο σύνολο εκπαίδευσής τους. Τα τυχαία δάση γενικά είναι λιγότερο αποδοτικά από τα δέντρα με ενίσχυση κλίσης. Ωστόσο, τα χαρακτηριστικά των δεδομένων μπορούν να επηρεάσουν την απόδοσή τους.

Ειδικότερα, τα δέντρα που αναπτύσσονται σε μεγάλο βάθος τείνουν να μαθαίνουν εξαιρετικά ακανόνιστα μοτίβα: προσαρμόζουν υπερβολικά τα σύνολα εκπαίδευσής τους, δηλαδή έχουν χαμηλή μεροληψία, αλλά πολύ υψηλή διακύμανση. Τα τυχαία δάση είναι ένας τρόπος υπολογισμού του μέσου όρου πολλαπλών βαθιών δέντρων απόφασης, που εκπαιδεύονται σε διαφορετικά τμήματα του ίδιου συνόλου εκπαίδευσης, με στόχο τη μείωση της διακύμανσης. Αυτό γίνεται εις βάρος μιας μικρής αύξησης της μεροληψίας και μιας απώλειας ερμηνευσιμότητας, αλλά γενικά ενισχύει σημαντικά την απόδοση του τελικού μοντέλου.

Τα δάση είναι κάτι σαν το σύνολο των προσπαθειών του αλγορίθμου δέντρων αποφάσεων. Λαμβάνοντας υπόψη την ομαδική εργασία πολλών δέντρων, βελτιώνεται έτσι η απόδοση ενός μεμονωμένου τυχαίου δέντρου. Αν και δεν είναι εντελώς παρόμοια, τα δάση δίνουν τα αποτελέσματα μιας διασταυρούμενης επικύρωσης k-πτυχών (k-fold cross validation).

Ο αλγόριθμος εκπαίδευσης για τα τυχαία δάση εφαρμόζει τη γενική τεχνική του bootstrap aggregating, ή bagging. Δεδομένου ενός συνόλου εκπαίδευσης $X = x_1, ..., x_n$ με αποκρίσεις $Y = y_1, ..., y_n$, η μέθοδος bagging επιλέγει επανειλημμένα (B φορές) ένα τυχαίο δείγμα με αντικατάσταση από το σύνολο εκπαίδευσης και προσαρμόζει δέντρα σε αυτά τα δείγματα:

Για b = 1, ..., B:

- λαμβάνουμε δείγμα, με αντικατάσταση, n παραδειγμάτων εκπαίδευσης από τα X, Y- ας τα ονομάσουμε X_b, Y_b,
- εκπαιδεύουμε ένα δέντρο παλινδρόμησης f_b στα X_b, Y_b.

Μετά την εκπαίδευση, οι προβλέψεις για τα αόρατα δείγματα x' μπορούν να γίνουν με τη μέση τιμή των προβλέψεων από όλα τα μεμονωμένα δέντρα παλινδρόμησης στο x':

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x')$$

ή με τη λήψη της πλειοψηφίας στην περίπτωση των δέντρων ταξινόμησης.

Αυτή η διαδικασία bootstrapping οδηγεί σε καλύτερη απόδοση του μοντέλου, επειδή μειώνει τη διακύμανση του μοντέλου, χωρίς να αυξάνει τη μεροληψία. Αυτό σημαίνει ότι ενώ οι προβλέψεις ενός μεμονωμένου δέντρου είναι ιδιαίτερα ευαίσθητες στο θόρυβο του συνόλου εκπαίδευσής του, ο μέσος όρος πολλών δέντρων δεν είναι, εφόσον τα δέντρα δεν συσχετίζονται. Η απλή εκπαίδευση πολλών δέντρων σε ένα μόνο σύνολο εκπαίδευσης θα έδινε δέντρα με έντονη συσχέτιση (ή ακόμη και το ίδιο δέντρο πολλές φορές, αν ο αλγόριθμος εκπαίδευσης είναι ντετερμινιστικός). Η δειγματοληψία bootstrap είναι ένας τρόπος απο-συσχέτισης των δέντρων, παρουσιάζοντάς τους διαφορετικά σύνολα εκπαίδευσης.

Επιπλέον, μια εκτίμηση της αβεβαιότητας της πρόβλεψης μπορεί να γίνει ως η τυπική απόκλιση των προβλέψεων από όλα τα μεμονωμένα δέντρα παλινδρόμησης στο x':

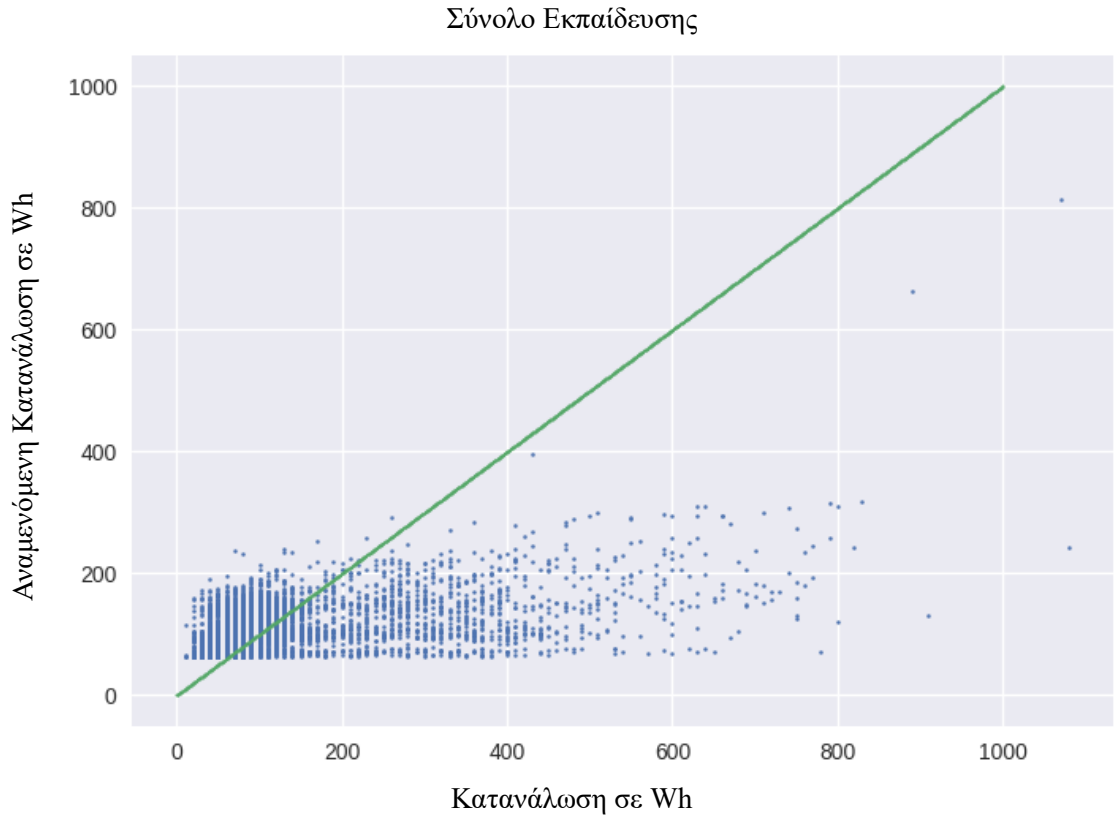
$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}.$$

Ο αριθμός των δειγμάτων/δέντρων, B, είναι μια ελεύθερη παράμετρος. Συνήθως, χρησιμοποιούνται μερικές εκατοντάδες έως αρκετές χιλιάδες δέντρα, ανάλογα με το μέγεθος και τη φύση του συνόλου εκπαίδευσης. Ένας βέλτιστος αριθμός δέντρων B μπορεί να βρεθεί με τη χρήση διασταυρούμενης επικύρωσης ή παρατηρώντας το out-of-bag error: το μέσο σφάλμα πρόβλεψης σε κάθε δείγμα εκπαίδευσης x_i, χρησιμοποιώντας μόνο τα δέντρα που δεν είχαν το x_i στο bootstrap δείγμα τους. Τα σφάλματα εκπαίδευσης και αξιολόγησης τείνουν να εξισορροπούνται μετά την προσαρμογή κάποιου αριθμού δέντρων.

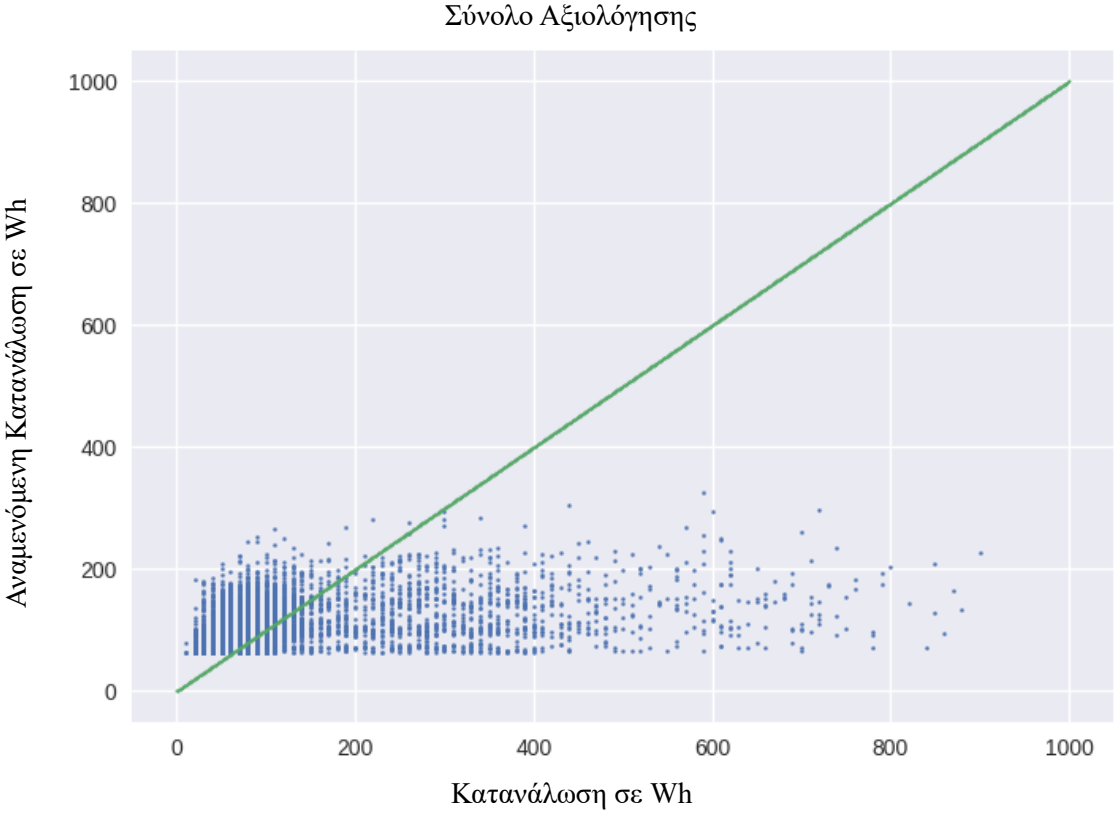
Σε αντίθεση με τα GBDT το τυχαίο δάσος επιλέγει τυχαία τις παρατηρήσεις. Δημιουργεί ένα δέντρο αποφάσεων και λαμβάνεται ο μέσος όρος του αποτελέσματος. Δεν χρησιμοποιεί κανένα σύνολο τύπων.

Σε αυτή την υποενότητα παρουσιάζονται τα αποτελέσματα που λάβαμε μετά την προσαρμογή και υλοποίηση του Random Forest Regressor με τις καλύτερες προβλέψεις, δηλαδή αυτόν με πλήθος σταδίων boosting 300 και μέγιστο βάθος 5.

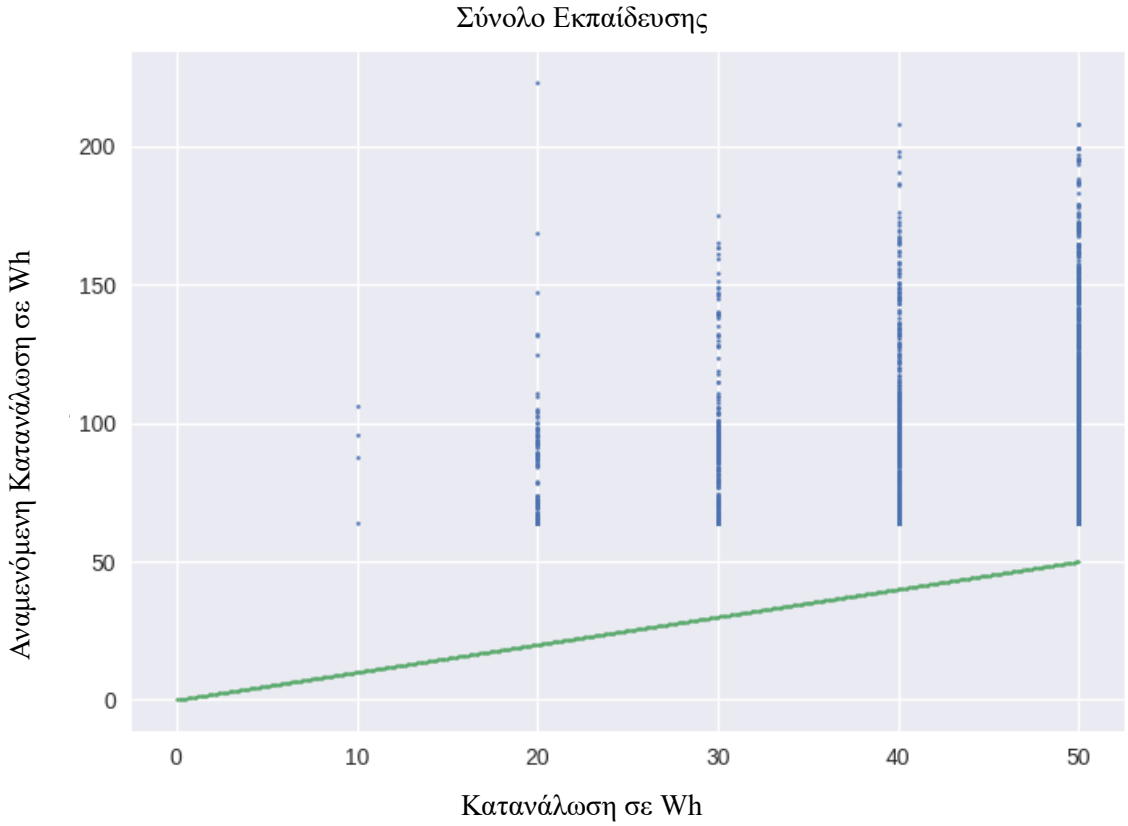
Όπως και προηγουμένως, στα παρακάτω σχήματα η πράσινη ευθεία (όπου η πραγματική τιμή ισούται με την αναμενόμενη τιμή) και η μπλε (όπου τα υπόλοιπα πραγματικής μείον αναμενόμενης τιμής είναι μηδενικά) είναι βοηθητικές οπτικά.



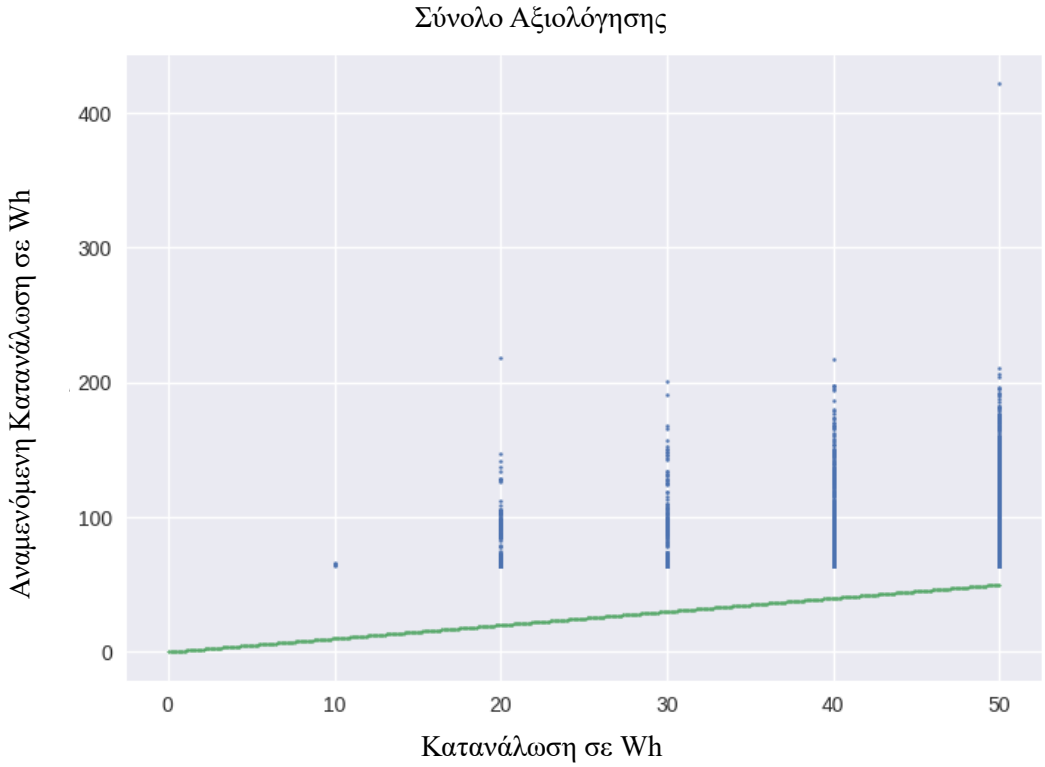
Σχήμα 67: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης



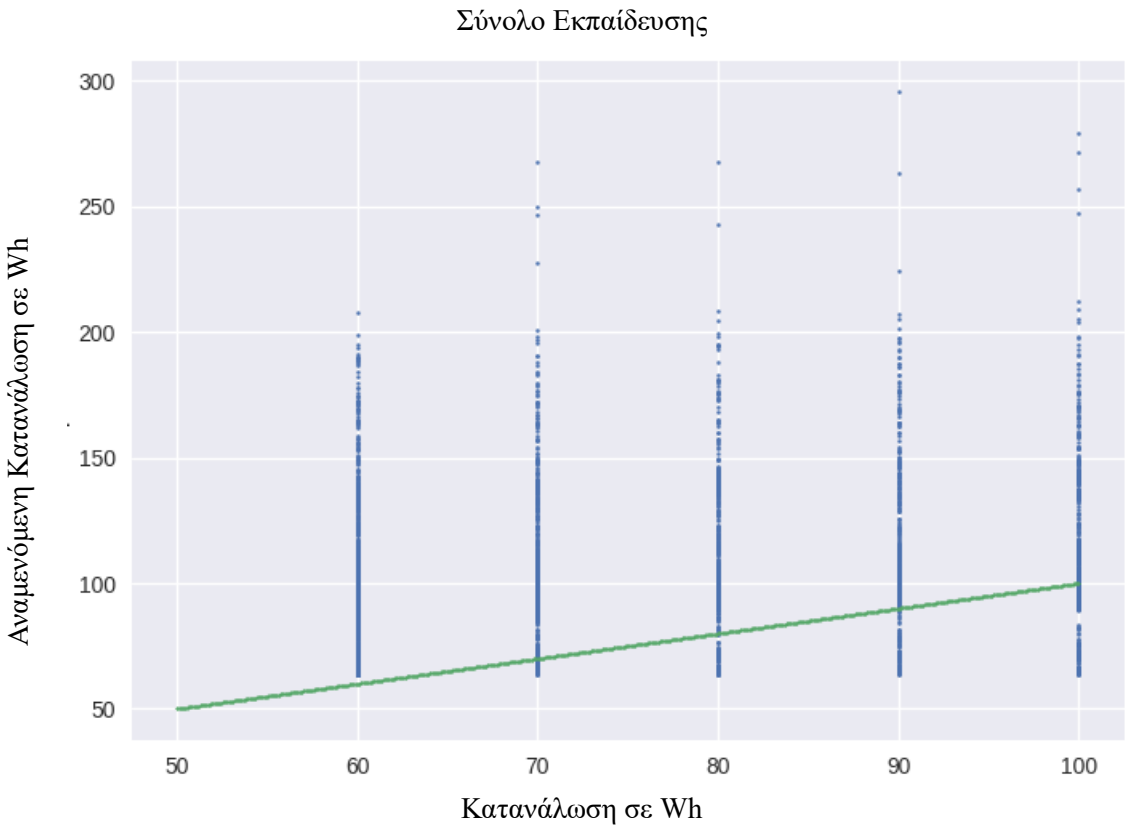
Σχήμα 68: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης



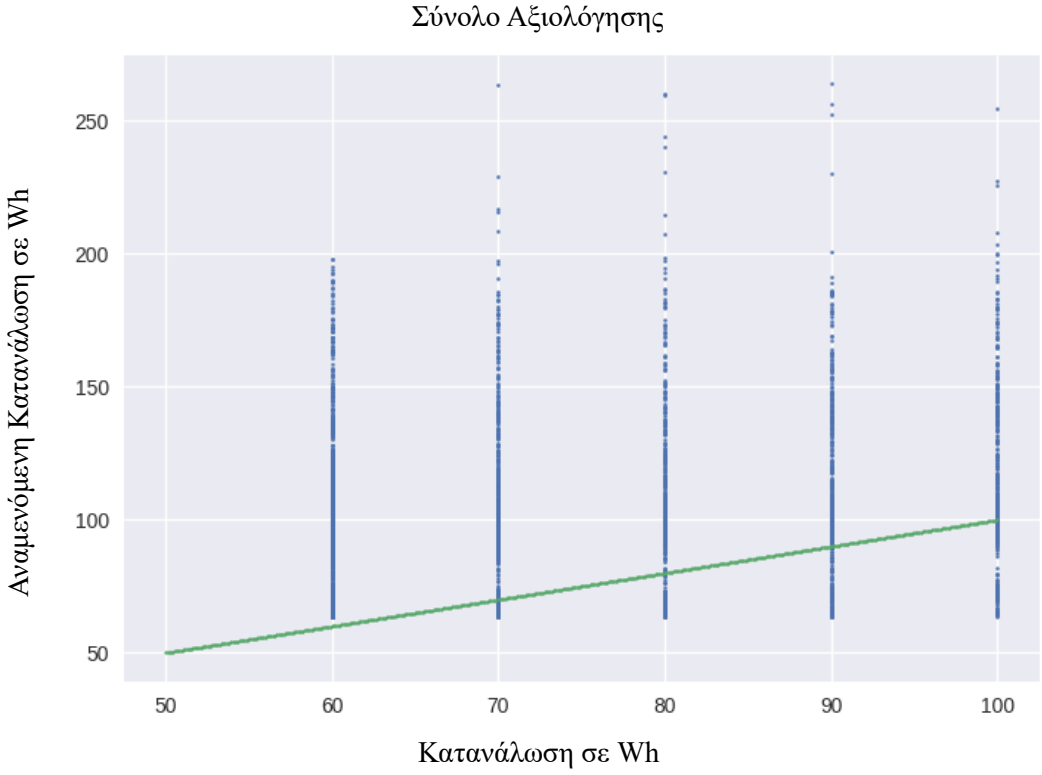
Σχήμα 69: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές από 0 έως 50 Wh



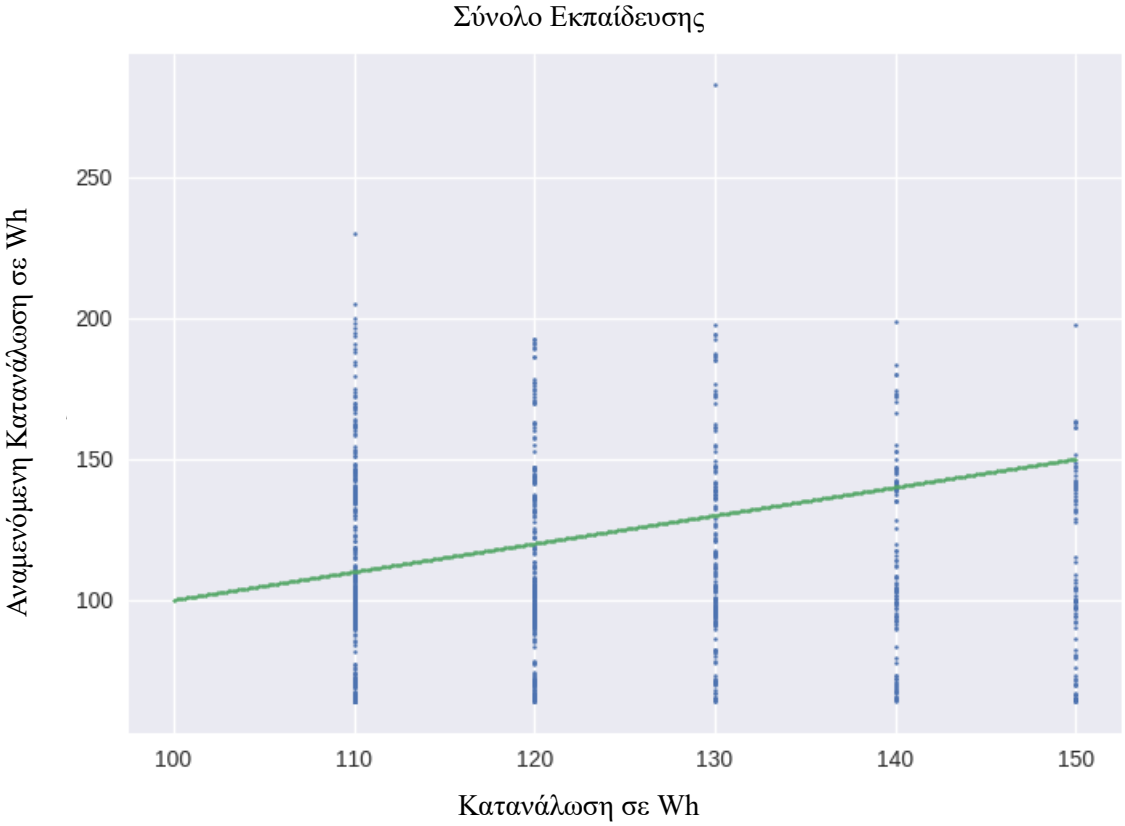
Σχήμα 70: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές από 0 έως 50 Wh



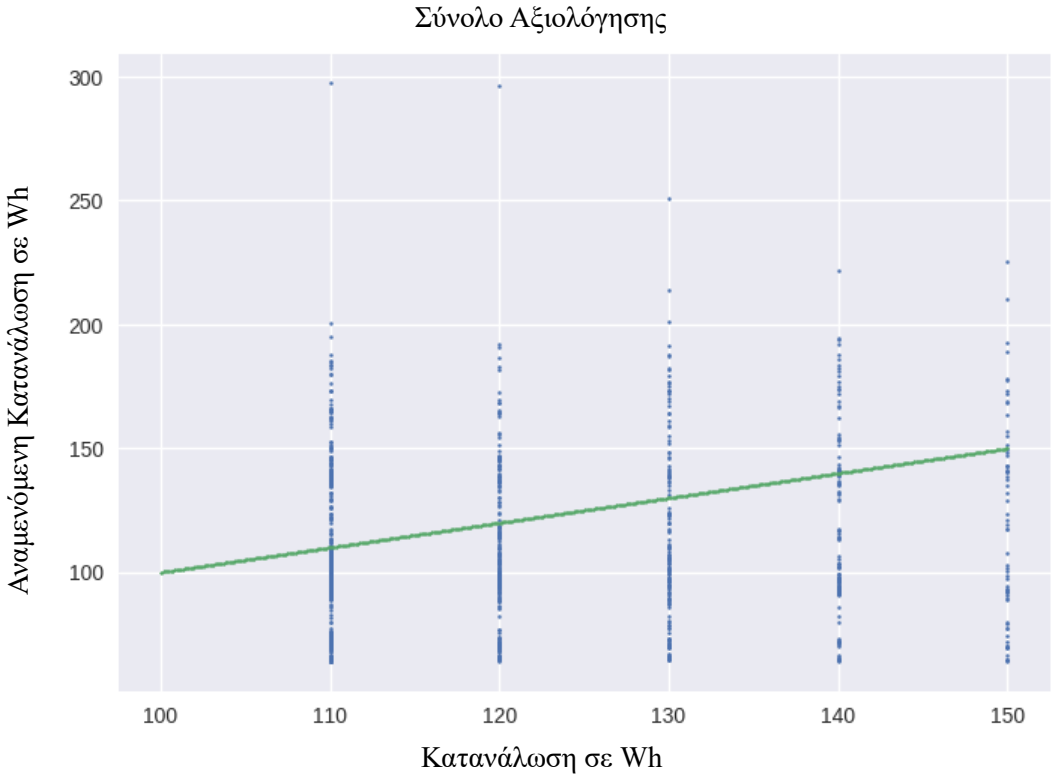
Σχήμα 71: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές από 50 έως 100 Wh



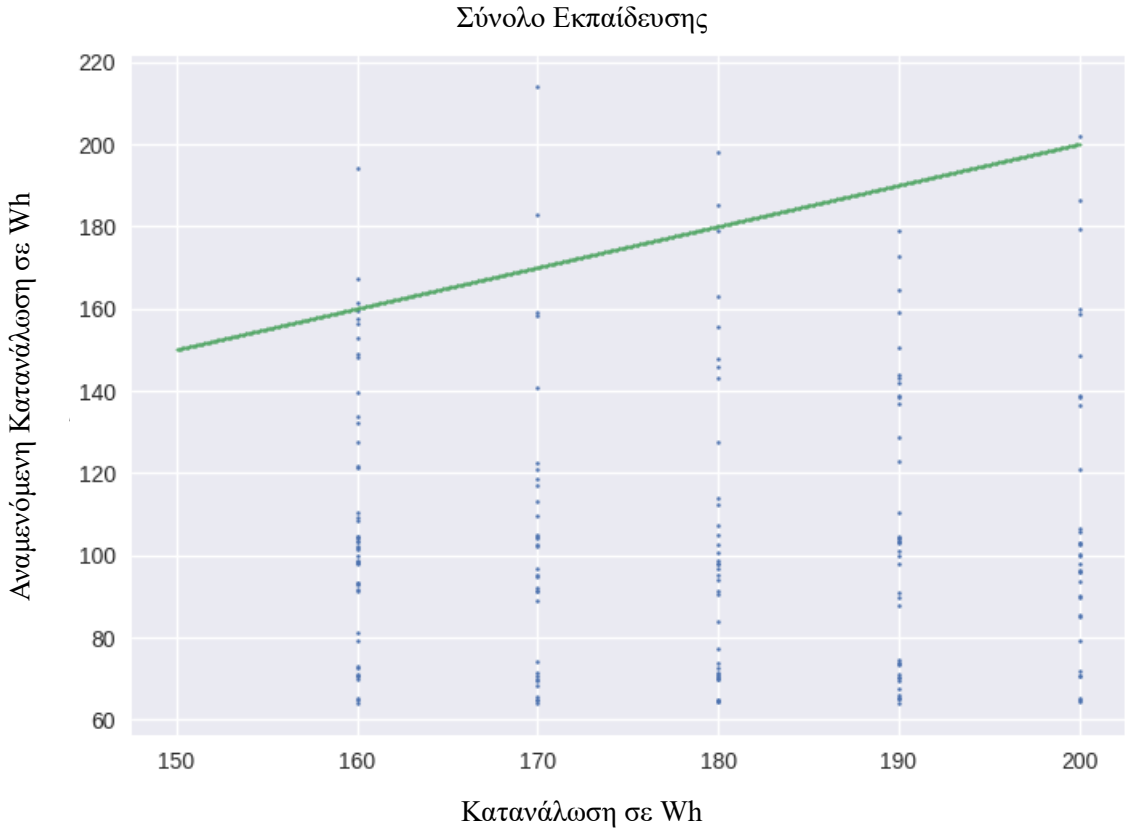
Σχήμα 72: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές από 50 έως 100 Wh



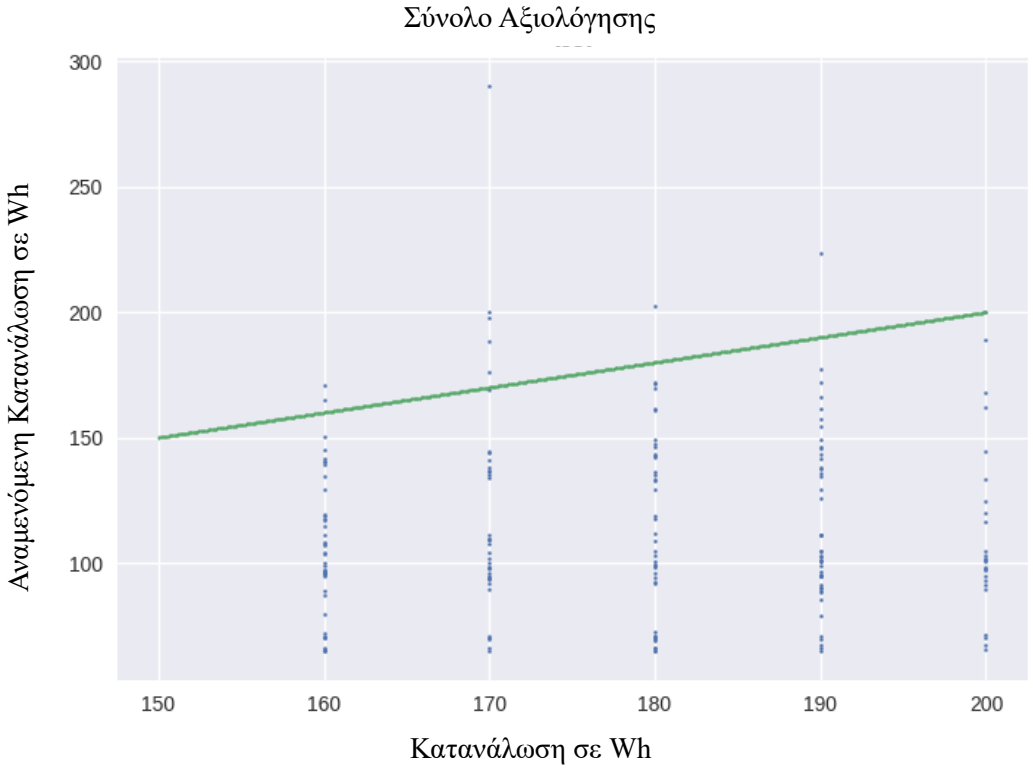
Σχήμα 73: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές από 100 έως 150 Wh



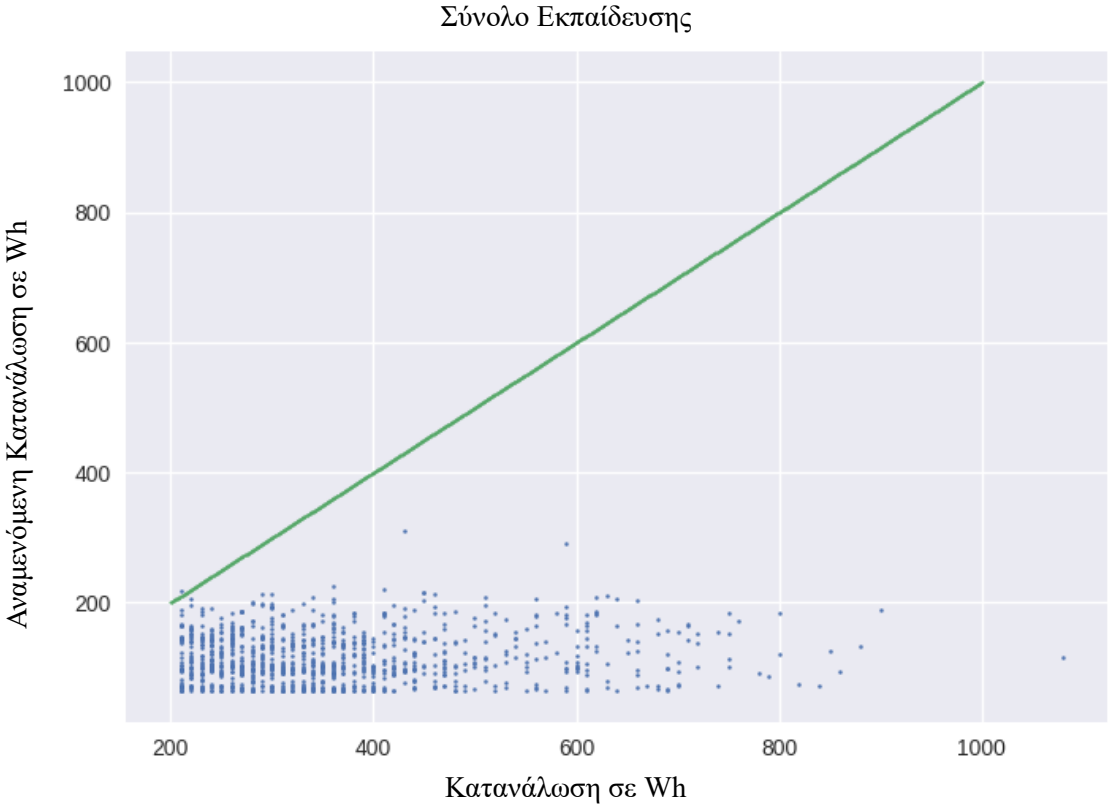
Σχήμα 74: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές από 100 έως 150 Wh



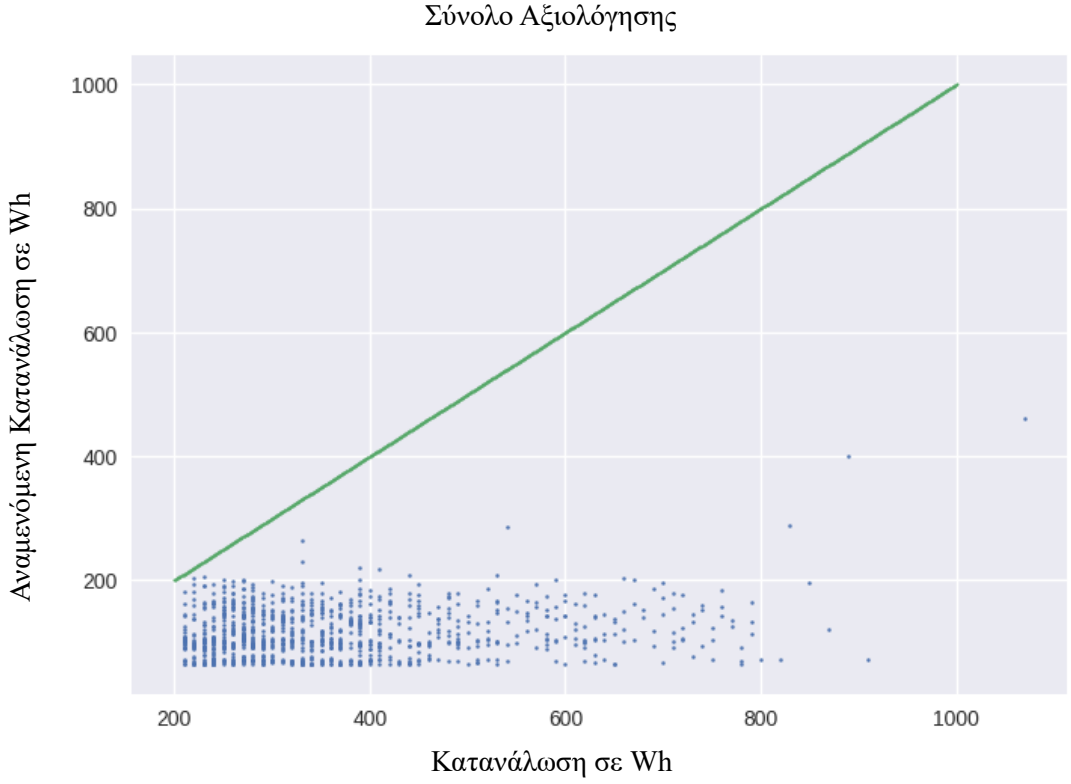
Σχήμα 75: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές από 150 έως 200 Wh



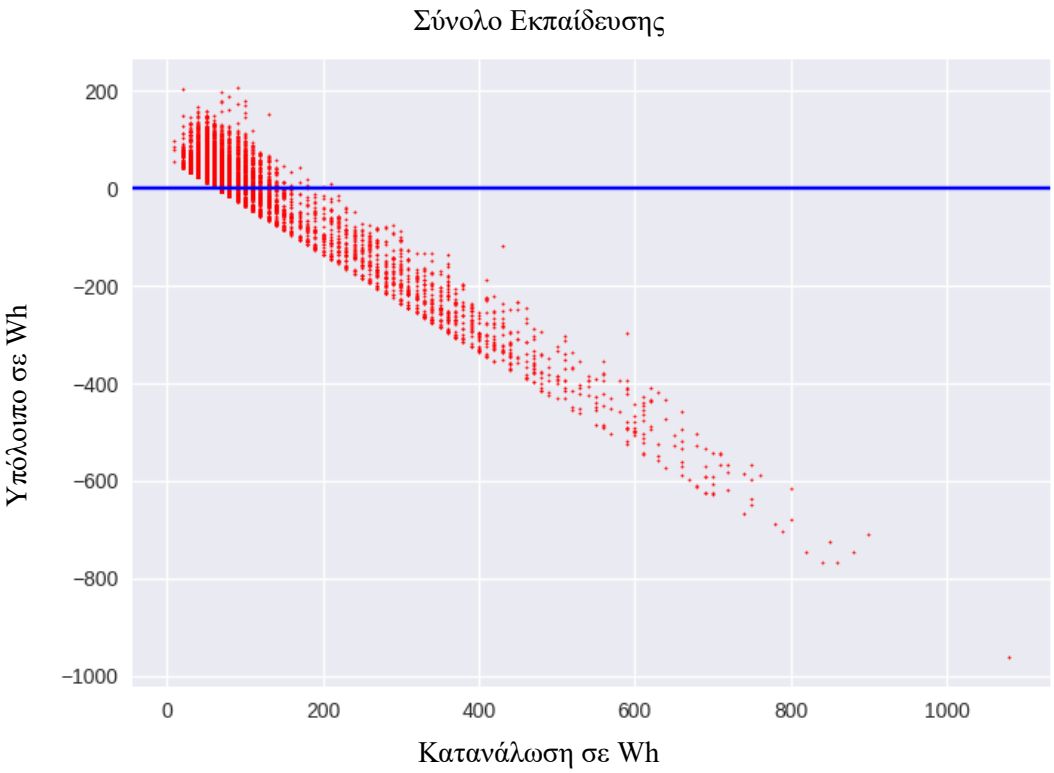
Σχήμα 76: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές από 150 έως 200 Wh



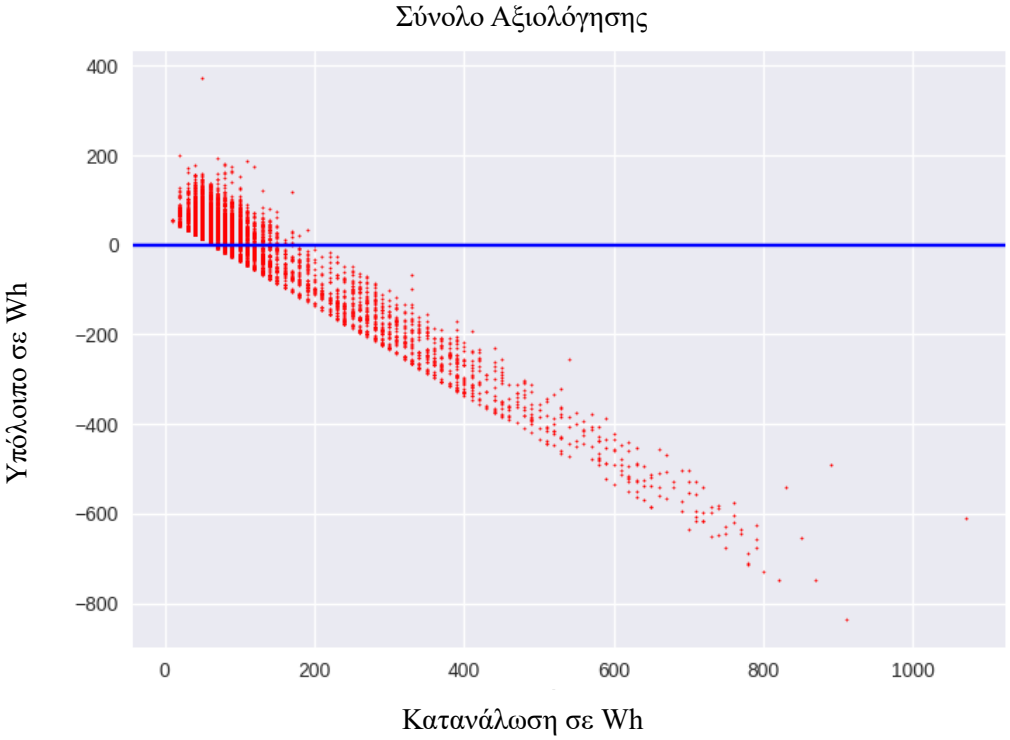
Σχήμα 77: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο εκπαίδευσης για τιμές πάνω από 200 Wh



Σχήμα 78: Διάγραμμα πραγματικής- αναμενόμενης τιμής στο σύνολο αξιολόγησης για τιμές πάνω από 200 Wh



Σχήμα 79: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο εκπαίδευσης

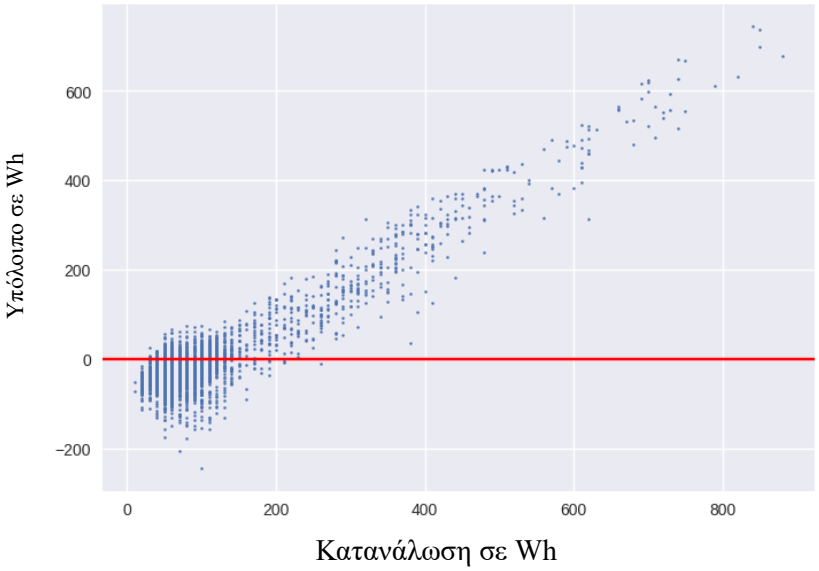


Σχήμα 80: Διάγραμμα υπολοίπου συναρτήσει πραγματικής τιμής στο σύνολο αξιολόγησης

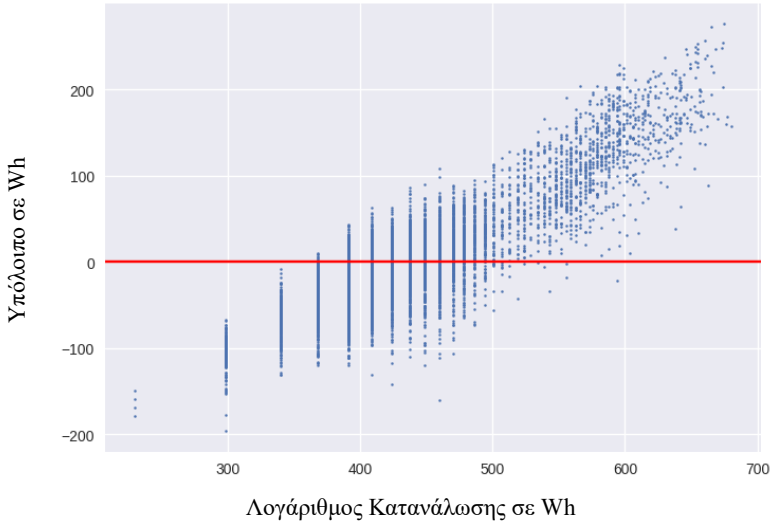
9 Συζήτηση Αποτελεσμάτων

Στην υποενότητα αυτή καλούμαστε να αξιολογήσουμε την αποδοτικότητα των μοντέλων. Το εργαλείο που μας βοηθά σε αυτή τη διαδικασία αποφασίζουμε να είναι το διάγραμμα των υπολοίπων σε σχέση με την πραγματική τιμή της κατανάλωσης ενέργειας σε Wh του συνόλου αξιολόγησης.

Όπως και προηγουμένως, στα παρακάτω σχήματα η οριζόντια ευθεία (όπου τα υπόλοιπα πραγματικής μείον αναμενόμενης τιμής είναι μηδενικά) είναι βοηθητική οπτικά.

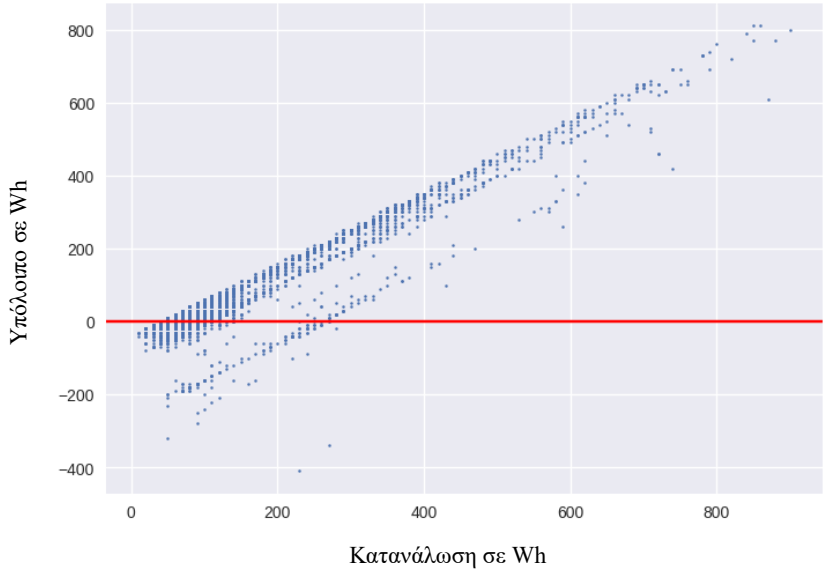


Σχήμα 81: Πολλαπλή Γραμμική Παλινδρόμηση της Συνολικής Κατανάλωσης

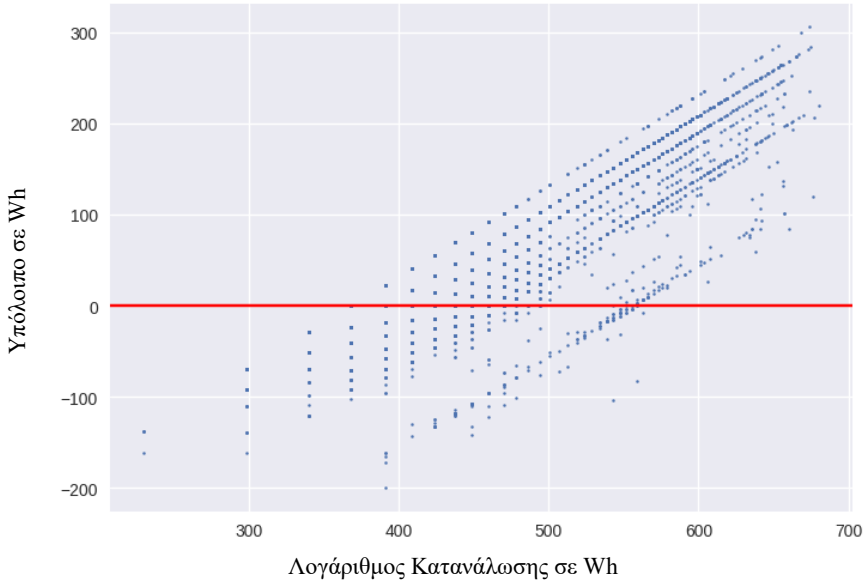


Σχήμα 82: Πολλαπλή Γραμμική Παλινδρόμηση της Συνολικής Κατανάλωσης σε Λογαριθμιμένη Κλίμακα

Όπως αναφέρθηκε και σε προηγούμενη ενότητα, για την Πολλαπλή Γραμμική Παλινδρόμηση σημειώνουμε ότι για τις μεγαλύτερες παρατηρήσεις καλύτερα λειτουργεί το μοντέλο με τη λογαριθμιμένη κλίμακα (Σχήμα 82), ωστόσο στις πιο μικρές λειτουργεί καλύτερα το πρώτο (Σχήμα 81). Ξέρουμε ότι οι περισσότερες παρατηρήσεις είναι μικρές, οπότε το πρώτο μοντέλο που εφαρμόσαμε μας εξυπηρετεί καλύτερα. Ακριβώς το ίδιο συμβαίνει και για τη Λογιστική Παλινδρόμηση (Σχήμα 83 και 84 αντίστοιχα).



Σχήμα 83: Λογιστική Παλινδρόμηση της Συνολικής Κατανάλωσης



Σχήμα 84: Λογιστική Παλινδρόμηση της Συνολικής Κατανάλωσης σε Λογαριθμιμένη Κλίμακα

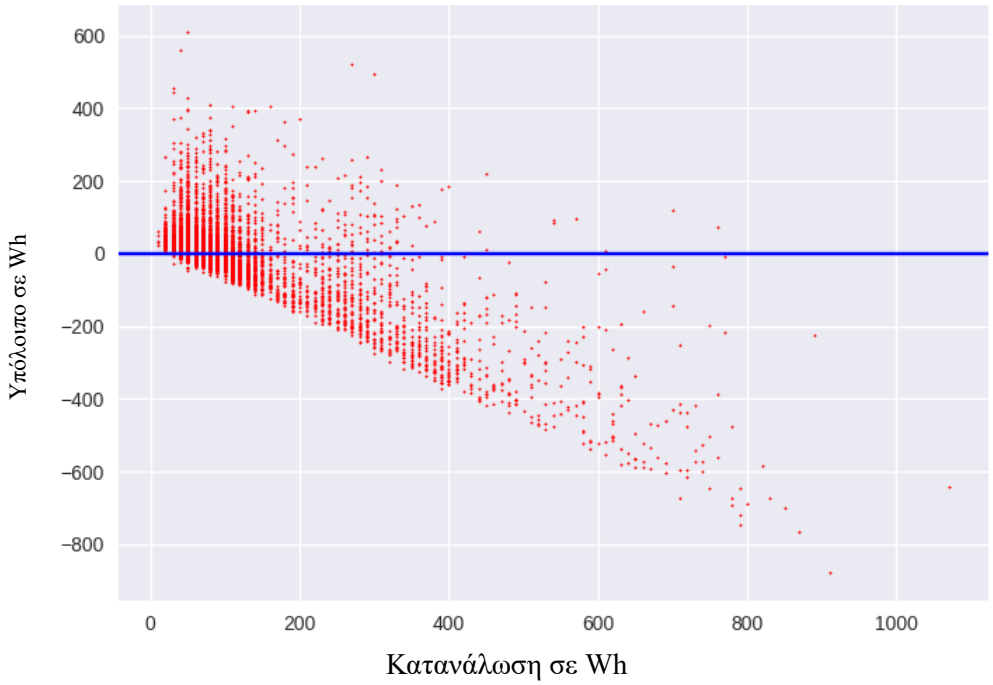
Προχωρώντας τώρα στους πιο αποτελεσματικούς εκτιμητές, είναι απαραίτητο να αναφέρουμε στοιχεία σχετικά με το μέγεθος του δείγματος, ώστε να έχουμε σωστή εικόνα σχετικά με το ποια είναι τα όρια του bulk και ποιες παρατηρήσεις είναι στατιστικά σημαντικότερες. Πιο συγκεκριμένα:

	Πλήθος παρατηρήσεων	Ποσοστό επί του συνόλου
Παρατηρήσεις από 0 έως 50 Wh	7462	37.81%
Παρατηρήσεις από 50 έως 100 Wh	8040	40.74%
Παρατηρήσεις από 100 έως 150 Wh	1935	9.81%
Παρατηρήσεις από 150 έως 200 Wh	382	1.94%
Παρατηρήσεις μεγαλύτερες από 200 Wh	1916	9.70%
Σύνολο	19735	100%

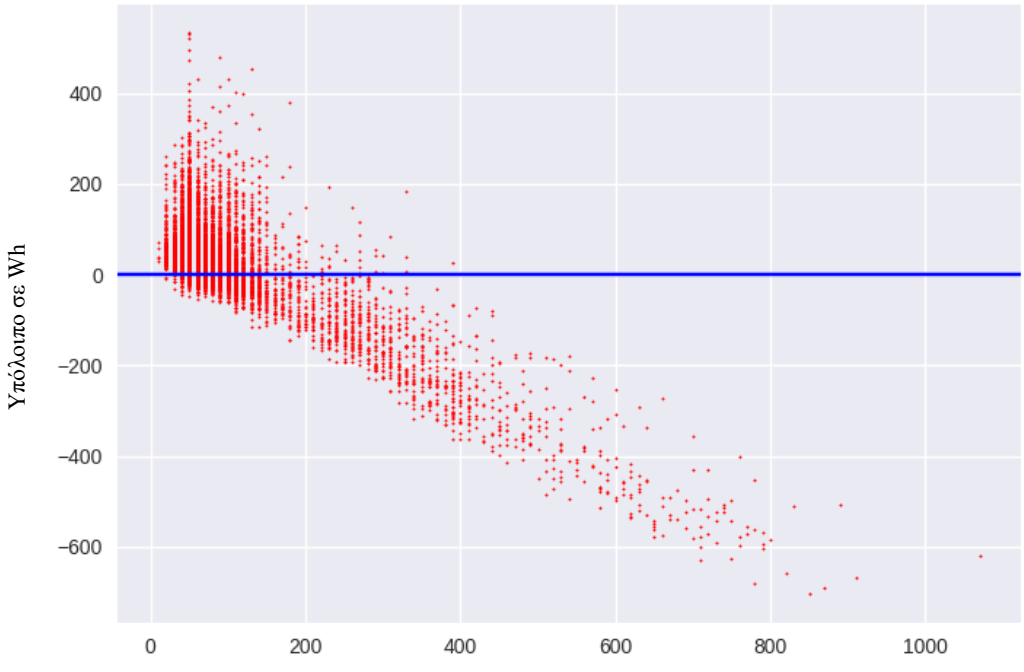
Πίνακας 2: Διαμέριση Παρατηρήσεων

Λαμβάνοντας υπόψη τα πλήθη που αναφέρθηκαν παραπάνω και τα παρακάτω διαγράμματα συμπεραίνουμε ότι ο καλύτερος εκτιμητής είναι το νευρωνικό δίκτυο. Αρχικά, ένα καλό δείγμα είναι ότι οι τιμές από 400 Wh και μετά αραιώνουν (κατακόρυφα) στο NN, σε αντίθεση με το GBDT και το random forest. Αυτό σημαίνει ότι η προσαρμογή είναι καλή, αλλά δεν έχουμε υπερπροσαρμογή. Ας δούμε χαρακτηριστικά το rf, όπου η σχέση μοιάζει με γραμμική. Συγκρίνοντας τώρα το NN και το GBDT, παρατηρούμε ότι για τιμές μέχρι 200 Wh το bulk στο NN έχει καλύτερη εικόνα. Αλλά τι σημαίνει αυτό;

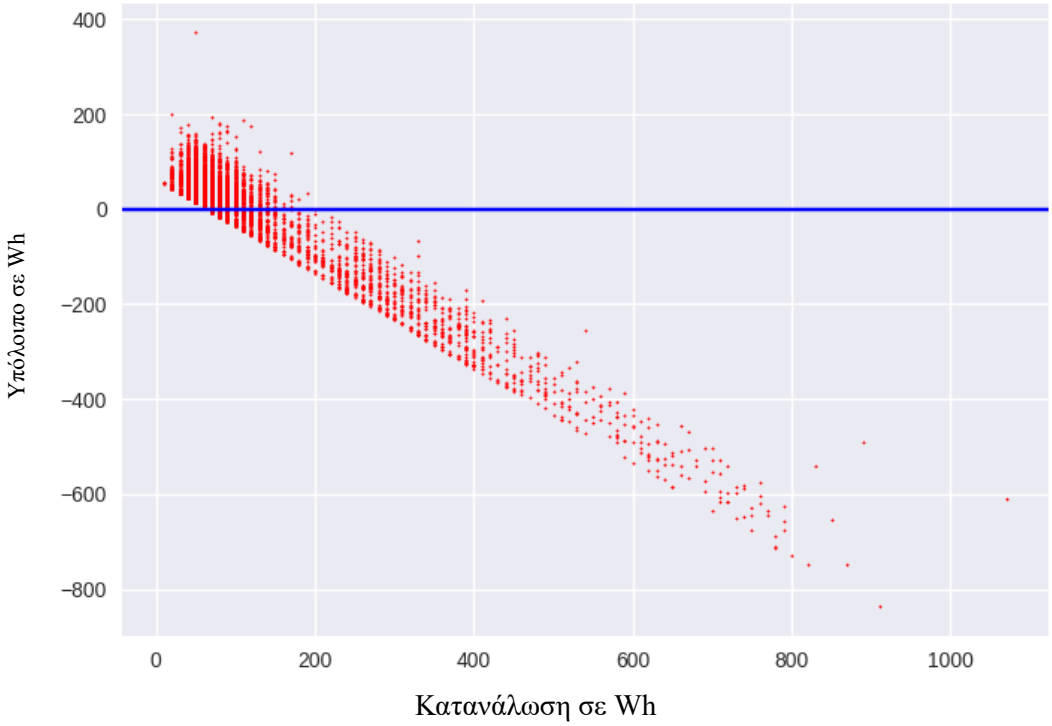
Αυτό μεταφράζεται κοιτώντας πάνω από την οριζόντια γραμμή (μηδενικών υπολοίπων). Βλέπουμε μια πιο στρογγυλεμένη κορυφή στο NN με κορυφή λίγο κάτω από 200 Wh, ενώ στο GBDT βλέπουμε αυτή την κορυφή να κάνει «μύτη» και σε τιμή, μάλιστα, μεγαλύτερη του 200 Wh (περίπου 300 Wh).



Σχήμα 85: Νευρωνικό Δίκτυο με 7 Κρυφά Στρώματα



Σχήμα 86: GBDT



Σχήμα 87: Random Forest Regressor

Πιο συγκεκριμένα, υπολογίζοντας το R^2 για κάθε μοντέλο (Πίνακας 3) επιβεβαιώνουμε τα παραπάνω. Δηλαδή ότι οι δύο πρώτοι μέθοδοι δεν έχουν καλή ικανότητα πρόβλεψης. Ακόμη, συγκρίνοντας τα τρία Νευρωνικά Δίκτυα μεταξύ τους, αυτό με τα επτά κρυφά στρώματα είναι το καλύτερο. Το Random Forest δεν δίνει καλές προβλέψεις. Τελικά, τα δύο μοντέλα με μεγάλο ποσοστό αποδοτικότητας είναι το Νευρωνικό Δίκτυο με επτά κρυφά στρώματα κα το GBDT. Ωστόσο, το ποσοστό του δεύτερου δεν πρέπει να μας ξεγελάσει, αφού είναι προϊόν πιθανής υπερεκπαίδευσης. Τα αποτελέσματα έχουν ως εξής:

Μοντέλο	Αποδοτικότητα στο σύνολο εκπαίδευσης	Αποδοτικότητα στο σύνολο αξιολόγησης
Γραμμική Παλινδρόμηση	17.03%	15.86%
Λογιστική Παλινδρόμηση	26.76%	25.60%
Νευρωνικό Δίκτυο με ένα κρυφό στρώμα	17.15%	15.89%
Νευρωνικό Δίκτυο με τρία κρυφά στρώματα	54.55%	24.93%
Νευρωνικό Δίκτυο με επτά κρυφά στρώματα	80.76%	40.73%
Gradient Boosted Decision Trees	88.30%	39.60%
Random Forest	26.41%	17.62%

Πίνακας 3: Αποδοτικότητες Μοντέλων

Σημείωση:

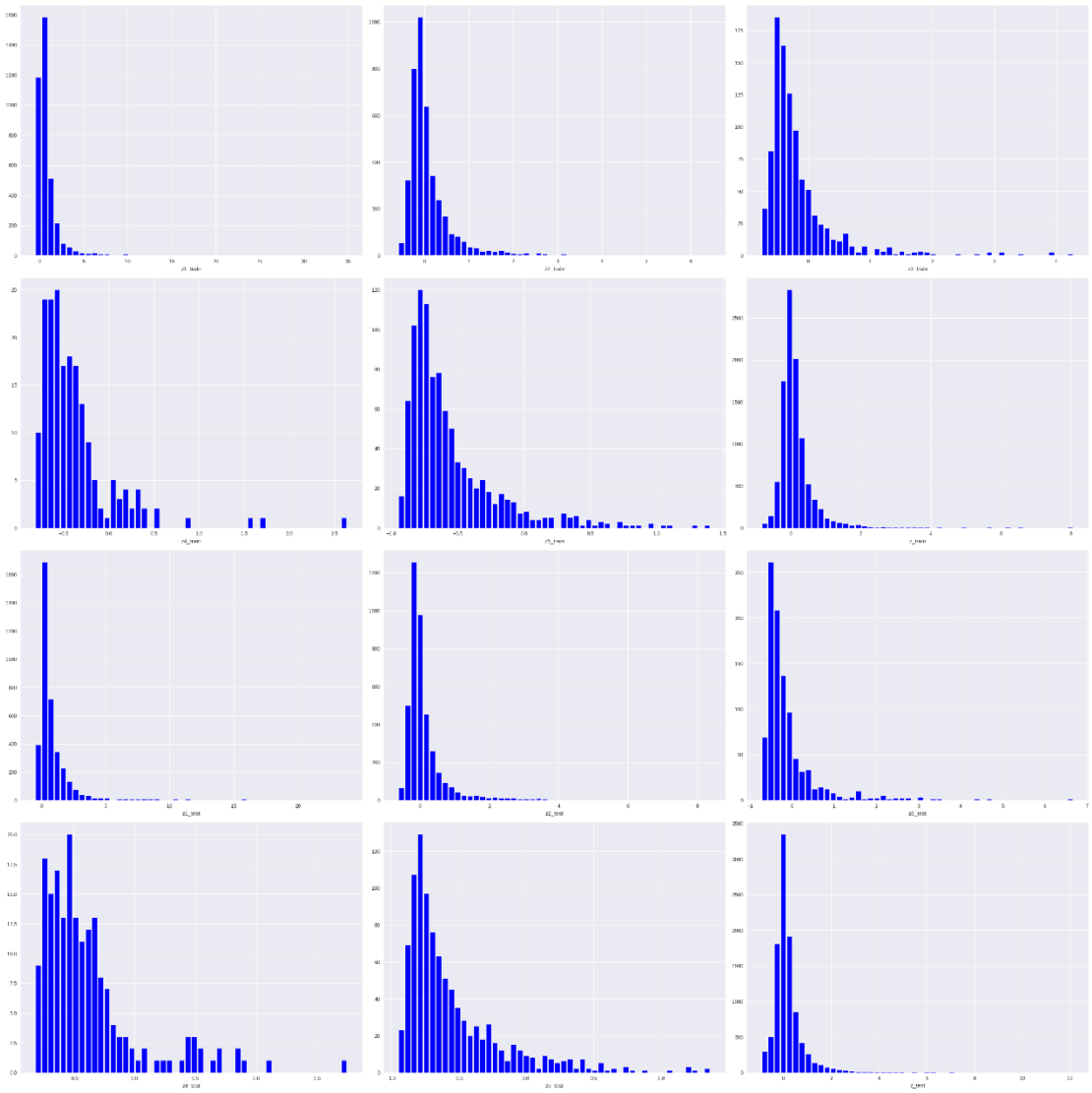
$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \hat{y})^2}$$

όπου $\hat{y}$ είναι η αναμενόμενη τιμή του y και $\bar{y}$ είναι η μέση τιμή του y.

Ας οπτικοποιήσουμε λοιπόν το πόσο καλός εκτιμητής είναι το NN που χρησιμοποιήθηκε. Έχοντας ήδη χωρίσει το δείγμα στα δύο σε σύνολο εκπαίδευσης και αξιολόγησης και έπειτα, αυτά στα 5 ανάλογα με την συνολική κατανάλωση, τότε υπολογίζουμε τις μεταβλητές z (λόγους υπολοίπων προς πραγματική κατανάλωση):

- z1_train = residuals1_train / y1_true_train
- z2_train = residuals2_train / y2_true_train
- z3_train = residuals3_train / y3_true_train
- z4_train = residuals4_train / y4_true_train
- z5_train = residuals5_train / y5_true_train
- z_train = residuals_train / y_true_train
- z1_test = residuals1_test / y1_true_test
- z2_test = residuals2_test / y2_true_test
- z3_test = residuals3_test / y3_true_test
- z4_test = residuals4_test / y4_true_test
- z5_test = residuals5_test / y5_true_test
- z_test = residuals_test / y_true_test

και τα ιστογράμματα αυτών με την αναφερθείσα σειρά (Σχήμα 88)



Σχήμα 88: Ιστογράμματα z

Παρατηρώντας τα ιστογράμματα αυτά βλέπουμε ότι έχουν είναι πολύ ξεκάθαρο το κελί (bin) που περιέχει τις πιο συχνές τιμές (mode – Most Probable Value).

Το mode είναι η τιμή που εμφανίζεται συχνότερα σε ένα σύνολο δεδομένων. Εάν η X είναι μια διακριτή τυχαία μεταβλητή, το mode είναι η τιμή x, στην οποία η συνάρτηση μάζας πιθανότητας παίρνει τη μέγιστη τιμή της (δηλαδή, $x=\text{argmax}_{x_i} P(X = x_i)$). Με άλλα λόγια, είναι η τιμή που είναι πιο πιθανό να πάρει κάποιος που κάνει τη δειγματοληψία.

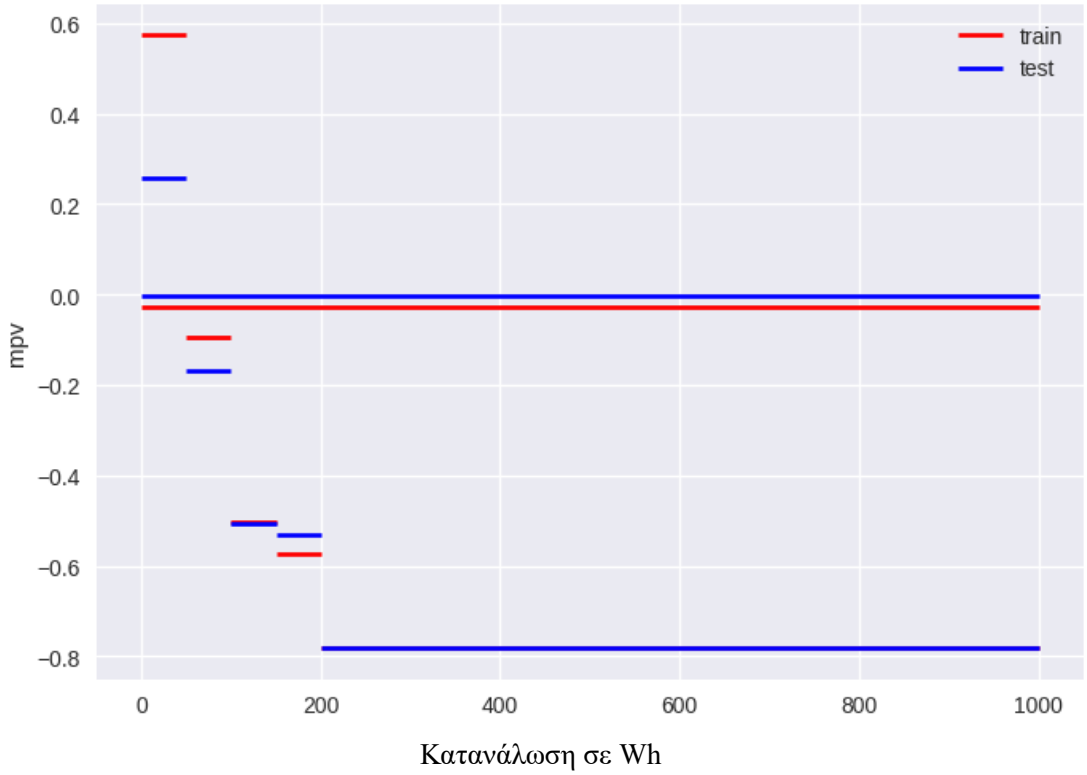
Όπως ο στατιστικός μέσος όρος (mean) και η διάμεσος (median), το mode είναι ένας τρόπος έκφρασης, σε έναν (συνήθως) αριθμό, σημαντικών πληροφοριών για μια τυχαία μεταβλητή ή έναν πληθυσμό. Η αριθμητική τιμή του mode είναι η ίδια με εκείνη του μέσου όρου και της διαμέσου σε μια κανονική κατανομή, ενώ μπορεί να είναι πολύ διαφορετική σε πολύ λοξές κατανομές.

Το mode ενός δείγματος είναι το στοιχείο που εμφανίζεται συχνότερα στη δειγματοληψία. Για παράδειγμα, το mode του δείγματος [1, 3, 6, 6, 6, 6, 6, 7, 7, 12, 12, 17] είναι 6. Για ένα δείγμα από μια συνεχή κατανομή, η έννοια είναι άχρηστη στην ακατέργαστη μορφή της, καθώς καμία τιμή δεν θα είναι ακριβώς ίδια, οπότε κάθε τιμή θα εμφανίζεται ακριβώς μία φορά. Προκειμένου να εκτιμηθεί το mode της υποκείμενης κατανομής, η συνήθης πρακτική είναι η διακριτοποίηση των δεδομένων με την ανεύρεση της συχνότητας σε διαστήματα ίσης απόστασης, όπως για τη δημιουργία ενός ιστογράμματος, αντικαθιστώντας ουσιαστικά τις τιμές με τα μέσα των διαστημάτων στα οποία ανατίθενται. Το mode είναι τότε η τιμή, όπου το ιστόγραμμα φτάνει στην κορυφή του. Είναι σημαντικό να επιλέξουμε ένα αντιπροσωπευτικό πλάτος των κελιών που να είναι συμβατό με το μέγεθος του δείγματος και τις τιμές που λαμβάνει.

Τα αποτελέσματα είναι:

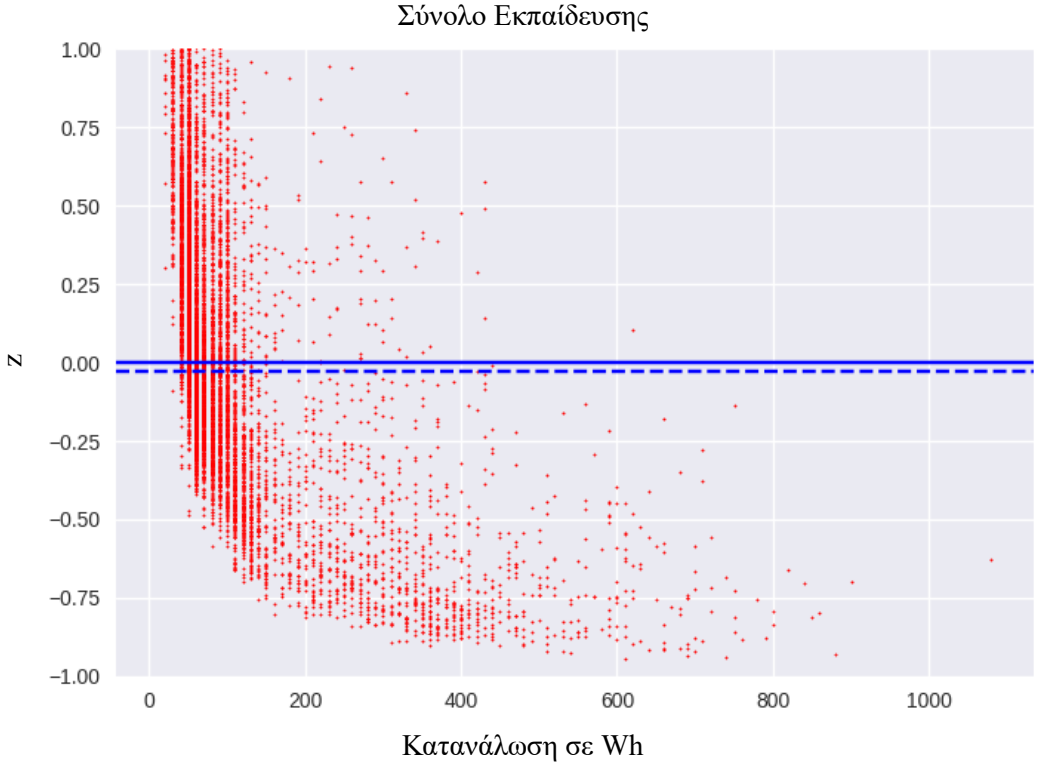
- z1_train: το κελί με τις περισσότερες τιμές είναι το (0.2192, 0.9271)
  mode = 0.5731
- z2_train: το κελί με τις περισσότερες τιμές είναι το (-0.1661, -0.0255)
  mode = -0.0958
- z3_train: το κελί με τις περισσότερες τιμές είναι το (-0.5525, -0.4518)
  mode = -0.5021
- z4_train: το κελί με τις περισσότερες τιμές είναι το (-0.6068, -0.5376)
  mode = -0.5722
- z5_train: το κελί με τις περισσότερες τιμές είναι το (-0.8051, -0.7581)
  mode = -0.7816
- z_train: το κελί με τις περισσότερες τιμές είναι το (-0.1157, 0.0623)
  mode = -0.0266
- z1_test: το κελί με τις περισσότερες τιμές είναι το (0.0162, 0.5024)
  mode = 0.2593
- z2_test: το κελί με τις περισσότερες τιμές είναι το (-0.2586, -0.0788)
  mode = -0.1687
- z3_test: το κελί με τις περισσότερες τιμές είναι το (-0.5794, -0.4312)
  mode = -0.5053
- z4_test: το κελί με τις περισσότερες τιμές είναι το (-0.5556, -0.5044)
  mode = -0.5300
- z5_test: το κελί με τις περισσότερες τιμές είναι το (-0.8055, -0.7593)
  mode = -0.7824
- z_test: το κελί με τις περισσότερες τιμές είναι το (-0.1318, 0.1293)
  mode = -0.0012

Στο Σχήμα 89 αναπαρίστανται οι πιο συχνές τιμές (mode – Most Probable Values) του μοντέλου. Με κόκκινο χρώμα για το σύνολο εκπαίδευση και με μπλε για το σύνολο αξιολόγησης. Μας χαροποιεί ιδιαίτερα το γεγονός ότι η πιο συχνή τιμή του z_test είναι πιο κοντά στο μηδέν σε σχέση με την πιο συχνή τιμή του z_train. Αυτό σημαίνει ότι η προσαρμογή είναι πολύ καλή και το μοντέλο έχει καλή ικανότητα γενίκευσης.

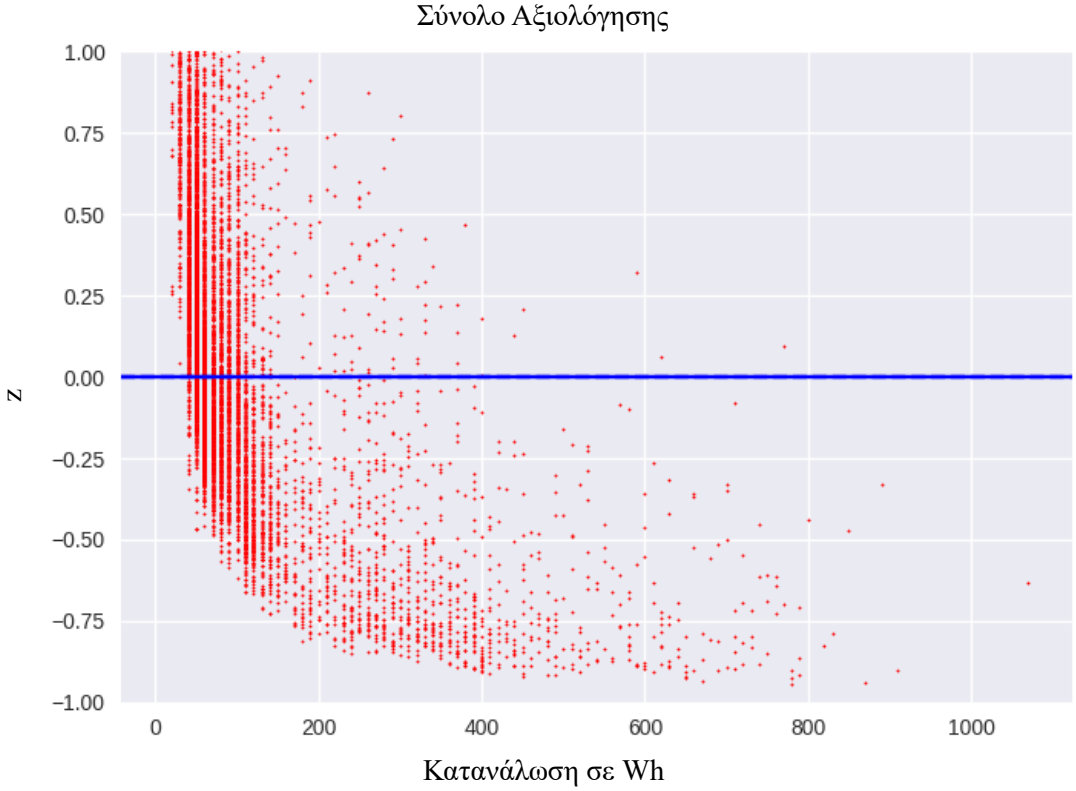


Σχήμα 89: Διαγράμματα πιο συχνών τιμών (mode- mpv)

Στα παρακάτω διαγράμματα οι συνεχόμενες γραμμές είναι βοηθητικές οπτικά και οι διακεκομμένες μας δείχνουν τις πιο συχνές τιμές του z_train (Σχήμα 90) και του z_test (Σχήμα 91) αντίστοιχα. Συμπερασματικά, το νευρωνικό δίκτυο είναι πράγματι ένα καλός εκτιμητής.



Σχήμα 90: Διαγράμματα πραγματικής τιμής- z στο σύνολο εκπαίδευσης



Σχήμα 91: Διαγράμματα πραγματικής τιμής- z στο σύνολο αξιολόγησης

10 Σύνοψη – Συμπεράσματα – Επεκτάσεις

Σκοπός αυτής της εργασίας είναι η αξιοποίηση μεθόδων μηχανικής μάθησης για τη δημιουργία μοντέλων πρόβλεψης κατανάλωσης ενέργειας. Τα δεδομένα που χρησιμοποιούνται περιλαμβάνουν μετρήσεις θερμοκρασίας και υγρασίας 9 δωματίων ενός σπιτιού, που παρακολουθούνται με ένα ασύρματο δίκτυο αισθητήρων ZigBee. Περιλαμβάνει επίσης μετεωρολογικά και κλιματικά δεδομένα, όπως θερμοκρασία, πίεση, υγρασία, ταχύτητα ανέμου, ορατότητα και σημείο υγροποίησης που μετρήθηκαν από το αεροδρόμιο Chievres. Το σύνολο δεδομένων συλλεγόταν 4,5 μήνες.

Το σύνολο δεδομένων προέρχεται από την ιστοσελίδα kaggle.com. Έχουμε στη διάθεση μας 19735 γεγονότα. Συνεπώς το αρχείο περιέχει 19735 γραμμές και 29 στήλες εκ των οποίων οι 28 είναι οι επεξηγηματικές μεταβλητές που χαρακτηρίζουν το κάθε γεγονός και η μια εξαρτώμενη μεταβλητή.

Πρώτα εφαρμόστηκε ένας γραμμικός αλγόριθμος πολλαπλής παλινδρόμησης. Έπειτα, υλοποιήθηκαν λογιστική παλινδρόμηση, Νευρωνικά Δίκτυα και Ενδυναμωμένα Δάση (Gradient Boosted Decision Tree και Random Forest Decision Tree). Τις αποδοτικότερες προβλέψεις έδωσε το Νευρωνικό Δίκτυο με τα περισσότερα κρυφά στρώματα (7).

Για την περεταίρω μελέτη του αντικειμένου της παρούσας Διπλωματικής Εργασίας:

- Θα μπορούσαν να υλοποιηθούν και μοντέλα, τα οποία δεν βασίζονται στην επιβλεπόμενη μάθηση και να εξεταστούν μέθοδοι μη επιβλεπόμενης ή ενισχυμένης μάθησης.
- Θα μπορούσε, ακόμη, να εξεταστεί κάποια ενδεχόμενη μείωση διαστάσεων με αξιολόγηση συνεισφοράς κάθε μεταβλητής.
- Τέλος, θα ήταν ενδιαφέρον να ασχοληθεί κανείς με τα δεδομένα που έχουν να κάνουν με την ημερομηνία και την ώρα, τα οποία στην παρούσα εργασία αγνοήθηκαν. Τα χαρακτηριστικά αυτά θα μπορούσαν να βοηθήσουν τους αλγόριθμους να αξιοποιήσουν τυχούσα περιοδικότητα στις χρονοσειρές. Η περιοδικότητα θα μπορούσε να είναι ημερήσια ή εβδομαδιαία. Μια ακόμα πληροφορία που θα μπορούσε να βελτιώσει τα αποτελέσματα θα ήταν η αξιοποίηση των αργιών.

Παράρτημα

Υλοποίηση στην Python:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
from matplotlib import pyplot as plt
import warnings
warnings.simplefilter('ignore')
%matplotlib Inline
import os
import io
plt.style.use('seaborn')
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
%matplotlib inline
from sklearn.tree import DecisionTreeClassifier
import math
from sklearn.preprocessing import MinMaxScaler # Normalize
from sklearn.metrics import mean_squared_error # Loss Function
from sklearn.model_selection import train_test_split
pd.set_option('display.max_columns', 500)
import warnings
warnings.filterwarnings('ignore')
from scipy import stats
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor,
ExtraTreesRegressor, AdaBoostRegressor
from sklearn.svm import SVR
from math import sqrt
```

```
from sklearn import metrics
from sklearn.decomposition import PCA
from scipy import stats
from google.colab import files

data_to_load = files.upload()
import io

df = pd.read_csv(io.BytesIO(data_to_load['KAG_energydata_complete.csv']))
df = df[['lights', 'T1', 'RH_1', 'T2', 'RH_2', 'T3', 'RH_3', 'T4', 'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed', 'Visibility', 'Tdewpoint','Appliances']]
#df = df[['lights', 'T1', 'RH_1', 'T2', 'RH_2', 'T3', 'RH_3', 'T4', 'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed', 'Visibility', 'Tdewpoint', 'rv1', 'rv2', 'Appliances']]

#missing data
df.info()

X = df[['lights', 'T1', 'RH_1', 'T2', 'RH_2', 'T3', 'RH_3', 'T4', 'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed', 'Visibility', 'Tdewpoint']]
yf = df.iloc[:, -1].values
names = df.columns
vars = names[:-1]

#histograms
fig, axs = plt.subplots(10,3,figsize=(30,60))
fig.tight_layout()
for i in range(3):
  axs[0,i].hist(df[[vars[i]]],bins=50,color='blue',rwidth=0.78,stacked=True)
  axs[0,i].set(xlabel = 'Variable_' +names[i+0])
```

```
for i in range(3):
  x = df[[vars[i+3]]]
  axs[1,i].hist(df[[vars[i+3]]],bins=50,color='blue',rwidth=0.78,stacked=True)
  axs[1,i].set(xlabel = 'Variable_' +names[i+3])

for i in range(3):
  axs[2,i].hist(df[[vars[i+6]]],bins=50,color='blue',rwidth=0.78,stacked=True)
  axs[2,i].set(xlabel = 'Variable_' +names[i+6])

for i in range(3):
  axs[3,i].hist(df[[vars[i+9]]],bins=50,color='blue',rwidth=0.78,stacked=True)
  axs[3,i].set(xlabel = 'Variable_' +names[i+9])

for i in range(3):
  axs[4,i].hist(df[[vars[i+12]]],bins=50,color='blue',rwidth=0.78,stacked=True)
  axs[4,i].set(xlabel = 'Variable_' +names[i+12])

for i in range(3):
  axs[5,i].hist(df[[vars[i+15]]],bins=50,color='blue',rwidth=0.78,stacked=True)
  axs[5,i].set(xlabel = 'Variable_' +names[i+15])

for i in range(3):
  axs[6,i].hist(df[[vars[i+18]]],bins=50,color='blue',rwidth=0.78,stacked=True)
  axs[6,i].set(xlabel = 'Variable_' +names[i+18])

for i in range(3):
  axs[7,i].hist(df[[vars[i+21]]],bins=50,color='blue',rwidth=0.78,stacked=True)
  axs[7,i].set(xlabel = 'Variable_' +names[i+21])
```

```
for i in range(3):
  axs[8,i].hist(df[[vars[i+24]]],bins=50,color='blue',rwidth=0.78,stacked=True)
  axs[8,i].set(xlabel = 'Variable_' +names[i+24])


for i in range(1):
  axs[9,i].hist(df[['Appliances']],bins=50,color='blue',rwidth=0.78,stacked=True)
  axs[9,i].set(xlabel = 'Variable_' +names[i+27])
#scatter matrix
sns.pairplot(df, kind='scatter')
plt.show()
#correlation matrix
plt.figure(figsize = (20,10))
sns.heatmap(data= df.corr(), cmap="YlGnBu", annot= True)
plt.title("Pairwise correlation of all the columns in the dataframe ")
plt.show()
#box plots
plt.figure(figsize = (25,9))
plt.ylim(-10, 200)
boxplot = df.boxplot(column=['lights', 'T1', 'RH_1', 'T2', 'RH_2', 'T3', 'RH_3', 'T4',
'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'T9', 'RH_9', 'T_out',
'Press_mm_hg', 'RH_out', 'Windspeed', 'Visibility',
'Tdewpoint','rv1','rv2','Appliances'])
plt.show()
#spliting to train and test set
X_train, X_test, yf_train, yf_test = train_test_split(X, yf, test_size = 0.5, random_state
= 0)
#Scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# PCA
pca = PCA()
```

```
m = pca.fit(X_train)
X_train=m.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
print('Variance explained by all 25 principal components=',
sum(m.explained_variance_ratio_*100))
x=np.arange(1,26,1)
a= np.cumsum((pca.explained_variance_ratio_ *100))
plt.plot(a)
plt.xticks(x, x)
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.grid(True)
plt.show()

print('Variance Explained by the first 4 Principal Components is= ', a[3])

x=np.arange(1,26,1)
plt.bar(x,pca.singular_values_,width=0.78,color='blue')
plt.xlabel('Eigenvalue index')
plt.ylabel('Eigenvalue')
plt.xticks(x, x)
plt.title('PCA')
plt.grid(True)
plt.show()
#svd
u, e, v = np.linalg.svd(X_train)



x=np.arange(0,25,1)
#labels = ['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17']
```

```
plt.bar(x,e,width=1,edgecolor='black')
plt.xlabel('Eigenvalue index')
plt.ylabel('Eigenvalue')
plt.title('svd')
plt.grid(True)
plt.xticks(x, x)
plt.show()
#linear regression
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, yf_train)
X = X_train
Y = yf_train

#Linear Regression
lr = LinearRegression()
lr.fit(X,Y)
w_0 = lr.coef_[0]
w_vector1 = lr.coef_
print("LR coefficients(SKLEARN - LR) for 25 values are")
print(w_vector1)
print("and the w0 (stable coefficient) is")
print(w_0)
#feature importance
importances = pd.DataFrame(data={
   'Attribute': vars,
   'Importance': lr.coef_
})
importances = importances.sort_values(by='Importance', ascending=False)
plt.bar(x=importances['Attribute'], height=importances['Importance'],
color='#087E8B')
```

```
plt.title('Feature importances obtained from coefficients', size=20)
plt.xticks(rotation='vertical')
plt.show()
#plots
plt.scatter(yf_train, yf_trainpred,s=2)
xx = np.linspace(0,800,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.show()
plt.scatter(yf_test, yf_pred,s=2)
xx = np.linspace(0,800,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.show()
plt.scatter(yf_train, yf_train-yf_trainpred ,s=2)
plt.axhline(y=0, color='r', linestyle='-')
plt.xlabel('true')
plt.ylabel('residual')
plt.show()
plt.scatter(yf_test, yf_test-yf_pred ,s=2)
plt.axhline(y=0, color='r', linestyle='-')
plt.xlabel('true')
plt.ylabel('residual')
plt.show()
#logistic regression
from sklearn.linear_model import LogisticRegression
regressorl = LogisticRegression()
regressorl.fit(X_train, yf_train)
```

```
yf_predl = regressorl.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((yf_predl.reshape(len(yf_predl),1),
yf_test.reshape(len(yf_test),1)),1))
#ta plots einai akribws analoga
#logy
y2 = data[['Appliances']]
y1=np.floor(np.log(y2)*100)
data.Appliances = y1
#gia gbdt
class_index=[]
for i in yf:
    if i<=50:
        class_index.append(1)
    elif i>50 and i<=100:
        class_index.append(2)
    elif i>100 and i<=150:
        class_index.append(3)
    elif i>150 and i<=200:
        class_index.append(4)
    elif i>200:
        class_index.append(5)
df['class']=np.array(class_index)
df1 = df.groupby('class')
X1 = df1.get_group(1)
X2 = df1.get_group(2)
X3 = df1.get_group(3)
X4 = df1.get_group(4)
X5 = df1.get_group(5)
yf1 = df1.get_group(1)
yf1 = yf1[['Appliances']]
```

```
yf2 = df1.get_group(2)

yf2 = yf2[['Appliances']]

yf3 = df1.get_group(3)

yf3 = yf3[['Appliances']]

yf4 = df1.get_group(4)

yf4 = yf4[['Appliances']]

yf5 = df1.get_group(5)

yf5 = yf5[['Appliances']]

X1_train, X1_test, yf1_train, yf1_test = train_test_split(X1, yf1, test_size = 0.5,
random_state = 0)

X2_train, X2_test, yf2_train, yf2_test = train_test_split(X2, yf2, test_size = 0.5,
random_state = 0)

X3_train, X3_test, yf3_train, yf3_test = train_test_split(X3, yf3, test_size = 0.5,
random_state = 0)

X4_train, X4_test, yf4_train, yf4_test = train_test_split(X4, yf4, test_size = 0.5,
random_state = 0)

X5_train, X5_test, yf5_train, yf5_test = train_test_split(X5, yf5, test_size = 0.5,
random_state = 0)

X_train, X_test, yf_train, yf_test = train_test_split(X, yf, test_size = 0.5, random_state
= 0)

X1_train=X1_train.iloc[:,:-2]

X2_train=X2_train.iloc[:,:-2]

X3_train=X3_train.iloc[:,:-2]

X4_train=X4_train.iloc[:,:-2]

X5_train=X5_train.iloc[:,:-2]

X1_test=X1_test.iloc[:,:-2]

X2_test=X2_test.iloc[:,:-2]

X3_test=X3_test.iloc[:,:-2]

X4_test=X4_test.iloc[:,:-2]

X5_test=X5_test.iloc[:,:-2]

#Scaling

sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X1_train = sc.fit_transform(X1_train)
X2_train = sc.fit_transform(X2_train)
X3_train = sc.fit_transform(X3_train)
X4_train = sc.fit_transform(X4_train)
X5_train = sc.fit_transform(X5_train)
X1_test = sc.fit_transform(X1_test)
X2_test = sc.fit_transform(X2_test)
X3_test = sc.fit_transform(X3_test)
X4_test = sc.fit_transform(X4_test)
X5_test = sc.fit_transform(X5_test)
#Gradient Boosting Decision Trees
#lr_list = [0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1]
lr_list = [0.15]
for learning_rate in lr_list:
    gb_clf = GradientBoostingRegressor(n_estimators=300,
learning_rate=learning_rate,  max_depth=5, random_state=0)
    gb_clf.fit(X_train,yf_train)


    print("Learning rate: ", learning_rate)
    print("Accuracy score (training): {0:.3f}".format(gb_clf.score(X_train, yf_train)))
    print("Accuracy score (validation): {0:.3f}".format(gb_clf.score(X_test, yf_test)))
    predictions3te = gb_clf.predict(X_test)
plt.scatter(yf_test, predictions3te,s=2)
xx = np.linspace(0,1000,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.title('test')
plt.show()
```

```
predictions3tr= gb_clf.predict(X_train)
plt.scatter(yf_train, predictions3tr,s=2)
xx = np.linspace(0,1000,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.title('train')
plt.show()
#gia 0-50
print("Learning rate: ", learning_rate)
print("Accuracy score (training): {0:.3f}".format(gb_clf.score(X1_train, yf1_train)))
print("Accuracy score (validation): {0:.3f}".format(gb_clf.score(X1_test, yf1_test)))
predictions331 = gb_clf.predict(X1_test)
plt.scatter(yf1_test, predictions331,s=2)
xx = np.linspace(0,50,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.title('test')
plt.show()
predictions3331 = gb_clf.predict(X1_train)
plt.scatter(yf1_train, predictions3331,s=2)
xx = np.linspace(0,50,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.title('train')
plt.show()
#akribws idia gia 50-100, 100-150, 150-200, 200+
# Convert columns to numpy array
```

```
yf1_test = yf1_test.to_numpy()
yf2_test = yf2_test.to_numpy()
yf3_test = yf3_test.to_numpy()
yf4_test = yf4_test.to_numpy()
yf5_test = yf5_test.to_numpy()
yf1_train = yf1_train.to_numpy()
yf2_train = yf2_train.to_numpy()
yf3_train = yf3_train.to_numpy()
yf4_train = yf4_train.to_numpy()
yf5_train = yf5_train.to_numpy()
res1train=predictions3331-yf1_train.ravel()
res2train=predictions3332-yf2_train.ravel()
res3train=predictions3333-yf3_train.ravel()
res4train=predictions3334-yf4_train.ravel()
res5train=predictions3335-yf5_train.ravel()
restrain=predictions3tr-yf_train.ravel()

res1test=predictions331-yf1_test.ravel()
res2test=predictions332-yf2_test.ravel()
res3test=predictions333-yf3_test.ravel()
res4test=predictions334-yf4_test.ravel()
res5test=predictions335-yf5_test.ravel()
restest=predictions3te-yf_test.ravel()
plt.scatter(yf1_train, res1train, color='r',s=1)
plt.scatter(yf2_train, res2train, color='r',s=1)
plt.scatter(yf3_train, res3train, color='r',s=1)
plt.scatter(yf4_train, res4train, color='r',s=1)
plt.scatter(yf5_train, res5train, color='r',s=1)
plt.axhline(y=0, color='b', linestyle='-')
plt.xlabel("true")
```

```
plt.ylabel("residuals")
plt.title("train")
plt.show()
plt.scatter(yf1_test, res1test, color='r',s=1)
plt.scatter(yf2_test, res2test, color='r',s=1)
plt.scatter(yf3_test, res3test, color='r',s=1)
plt.scatter(yf4_test, res4test, color='r',s=1)
plt.scatter(yf5_test, res5test, color='r',s=1)
plt.axhline(y=0, color='b', linestyle='-')
plt.xlabel("true")
plt.ylabel("residuals")
plt.title("test")
plt.show()
#random forest
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
rf_clf = RandomForestRegressor(n_estimators=300,max_depth=5, random_state=0)
rf_clf.fit(X_train, yf_train)

print("Accuracy score (training): {0:.3f}".format(rf_clf.score(X_train, yf_train)))
print("Accuracy score (validation): {0:.3f}".format(rf_clf.score(X_test, yf_test)))
predictionsrfte = rf_clf.predict(X_test)
plt.scatter(yf_test, predictionsrfte,s=2)
xx = np.linspace(0,1000,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.title('test')
plt.show()
predictionsrftr = rf_clf.predict(X_train)
```

```
plt.scatter(yf_train, predictionsrftr,s=2)
xx = np.linspace(0,1000,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.title('train')
plt.show()
#gia 0-50
predictionsrfte1 = rf_clf.predict(X1_test)
plt.scatter(yf1_test, predictionsrfte1,s=2)
xx = np.linspace(0,50,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.title('test')
plt.show()
predictionsrftr1 = rf_clf.predict(X1_train)
plt.scatter(yf1_train, predictionsrftr1,s=2)
xx = np.linspace(0,50,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.title('train')
plt.show()
#akribws idia gia 50-100, 100-150, 150-200, 200+
res1train=predictionsrftr1-yf1_train.ravel()
res2train=predictionsrftr2-yf2_train.ravel()
res3train=predictionsrftr3-yf3_train.ravel()
res4train=predictionsrftr4-yf4_train.ravel()
res5train=predictionsrftr5-yf5_train.ravel()
```

```
restrain=predictionsrftr-yf_train.ravel()

res1test=predictionsrfte1-yf1_test.ravel()
res2test=predictionsrfte2-yf2_test.ravel()
res3test=predictionsrfte3-yf3_test.ravel()
res4test=predictionsrfte4-yf4_test.ravel()
res5test=predictionsrfte5-yf5_test.ravel()
restest=predictionsrfte-yf_test.ravel()
plt.scatter(yf1_train, res1train, color='r',s=1)
plt.scatter(yf2_train, res2train, color='r',s=1)
plt.scatter(yf3_train, res3train, color='r',s=1)
plt.scatter(yf4_train, res4train, color='r',s=1)
plt.scatter(yf5_train, res5train, color='r',s=1)
plt.axhline(y=0, color='b', linestyle='-')
plt.xlabel("true")
plt.ylabel("residuals")
plt.title("train")
plt.show()

plt.scatter(yf1_test, res1test, color='r',s=1)
plt.scatter(yf2_test, res2test, color='r',s=1)
plt.scatter(yf3_test, res3test, color='r',s=1)
plt.scatter(yf4_test, res4test, color='r',s=1)
plt.scatter(yf5_test, res5test, color='r',s=1)
plt.axhline(y=0, color='b', linestyle='-')
plt.xlabel("true")
plt.ylabel("residuals")
plt.title("test")
plt.show()
#NN
```

```
from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.neural_network import MLPRegressor
clf0 = MLPRegressor(
    hidden_layer_sizes=(28),
    activation="logistic",
    verbose=False,
    max_iter=200
)
clf1 = MLPRegressor(    hidden_layer_sizes=(28,27,20),
    activation="relu",
    verbose=False,
    max_iter=800
)

clf2 = MLPRegressor(    hidden_layer_sizes=(74,60,40,32,16,8,4),
    activation="relu",
    verbose=False,
    max_iter=800
)

clf0.fit(X_train, yf_train)
clf1.fit(X_train, yf_train)
clf2.fit(X_train, yf_train)

print(clf0.score(X_train,yf_train))
print(clf0.score(X_test,yf_test))
print(clf1.score(X_train,yf_train))
print(clf1.score(X_test,yf_test))
print(clf2.score(X_train,yf_train))
print(clf2.score(X_test,yf_test))
```

```
print(mlp0.score(X_train,yf_train))
print(mlp0.score(X_test,yf_test))
print(mlp1.score(X_train,yf_train))
print(mlp1.score(X_test,yf_test))


plt.plot(clf0.loss_curve_,color="green",label="MLP0")
plt.plot(clf1.loss_curve_,color="red",label="MLP1")
plt.plot(clf2.loss_curve_,color="blue",label="MLP2")
plt.legend(loc="upper right")
plt.show()
#learning curve
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import learning_curve
from sklearn import datasets
import matplotlib.pyplot as plt
# Create a pipeline; This will be passed as an estimator to learning curve method
#
pipeline = make_pipeline(StandardScaler(),
                         LogisticRegression(penalty='l2', solver='lbfgs', random_state=1,
max_iter=10000))
#pipeline = make_pipeline(StandardScaler(),
#                         MLPClassifier(hidden_layer_sizes=(74,60,40,32,16,8,4),
max_iter=800))
# Use learning curve to get training and test scores along with train sizes
#
train_sizes, train_scores, test_scores = learning_curve(estimator=pipeline, X=X_train,
y=yf_train,
                                                        cv=10, train_sizes=np.linspace(0.1, 1.0, 10),
```

```
                                                    n_jobs=1)
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)
#
# Plot the learning curve
#
plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5,
label='Training Accuracy')
plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std,
alpha=0.15, color='blue')
plt.plot(train_sizes, test_mean, color='green', marker='+', markersize=5, linestyle='--',
label='Validation Accuracy')
plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std, alpha=0.15,
color='green')
plt.title('Learning Curve')
plt.xlabel('Training Data Size')
plt.ylabel('Model accuracy')
plt.legend(loc='lower right')
plt.show()
#plots
predictionsclftest = clf2.predict(X_test)
plt.scatter(yf_test, predictionsclftest,s=2)
xx = np.linspace(0,1000,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.title('test')
plt.show()
predictionsclftrain = clf2.predict(X_train)
plt.scatter(yf_train, predictionsclftrain,s=2)
```

```
xx = np.linspace(0,1000,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.title('train')
plt.show()
#gia 0-50
predictionsclf21 = clf2.predict(X1_test)
plt.scatter(yf1_test, predictionsclf21,s=2)
xx = np.linspace(0,50,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.title('test')
plt.show()
predictionsclf221 = clf2.predict(X1_train)
plt.scatter(yf1_train, predictionsclf221,s=2)
xx = np.linspace(0,50,1000)
plt.scatter(xx,xx,s=1)
plt.xlabel('true')
plt.ylabel('pred')
plt.title('train')
plt.show()
#akribws idia gia 50-100, 100-150, 150-200, 200+
#residual plots
# Convert columns to numpy array
yf1_test = yf1_test.to_numpy()
yf2_test = yf2_test.to_numpy()
yf3_test = yf3_test.to_numpy()
yf4_test = yf4_test.to_numpy()
```

```
yf5_test = yf5_test.to_numpy()
yf1_train = yf1_train.to_numpy()
yf2_train = yf2_train.to_numpy()
yf3_train = yf3_train.to_numpy()
yf4_train = yf4_train.to_numpy()
yf5_train = yf5_train.to_numpy()
res1train=predictionsclf221-yf1_train.ravel()
res2train=predictionsclf222-yf2_train.ravel()
res3train=predictionsclf223-yf3_train.ravel()
res4train=predictionsclf224-yf4_train.ravel()
res5train=predictionsclf225-yf5_train.ravel()
restrain=predictionsclftrain-yf_train.ravel()

res1test=predictionsclf21-yf1_test.ravel()
res2test=predictionsclf22-yf2_test.ravel()
res3test=predictionsclf23-yf3_test.ravel()
res4test=predictionsclf24-yf4_test.ravel()
res5test=predictionsclf25-yf5_test.ravel()
restest=predictionsclftest-yf_test.ravel()
plt.scatter(yf1_train, res1train, color='r',s=1)
plt.scatter(yf2_train, res2train, color='r',s=1)
plt.scatter(yf3_train, res3train, color='r',s=1)
plt.scatter(yf4_train, res4train, color='r',s=1)
plt.scatter(yf5_train, res5train, color='r',s=1)
plt.axhline(y=0, color='b', linestyle='-')
plt.xlabel("true")
plt.ylabel("residuals")
plt.title("train")
plt.show()
plt.scatter(yf1_test, res1test, color='r',s=1)
```

```
plt.scatter(yf2_test, res2test, color='r',s=1)
plt.scatter(yf3_test, res3test, color='r',s=1)
plt.scatter(yf4_test, res4test, color='r',s=1)
plt.scatter(yf5_test, res5test, color='r',s=1)
plt.axhline(y=0, color='b', linestyle='-')
plt.xlabel("true")
plt.ylabel("residuals")
plt.title("test")
plt.show()
#z
z1_train=res1train/yf1_train.ravel()
z2_train=res2train/yf2_train.ravel()
z3_train=res3train/yf3_train.ravel()
z4_train=res4train/yf4_train.ravel()
z5_train=res5train/yf5_train.ravel()
z_train=restrain/yf_train.ravel()

z1_test=res1test/yf1_test.ravel()
z2_test=res2test/yf2_test.ravel()
z3_test=res3test/yf3_test.ravel()
z4_test=res4test/yf4_test.ravel()
z5_test=res5test/yf5_test.ravel()
z_test=restest/yf_test.ravel()
z1_train = pd.Series(z1_train)
z2_train = pd.Series(z2_train)
z3_train = pd.Series(z3_train)
z4_train = pd.Series(z4_train)
z5_train = pd.Series(z5_train)
z_train = pd.Series(z_train)
z1_test = pd.Series(z1_test)
```

```
z2_test = pd.Series(z2_test)
z3_test = pd.Series(z3_test)
z4_test = pd.Series(z4_test)
z5_test = pd.Series(z5_test)
z_test = pd.Series(z_test)
#hists of z
titles =
['z1_train','z2_train','z3_train','z4_train','z5_train','z_train','z1_test','z2_test','z3_test','z4
_test','z5_test','z_test']


fig, axs = plt.subplots(4, 3,figsize=(30,30), tight_layout=True)
axs[0,0].hist(z1_train, bins=50, color='blue',rwidth=0.78,stacked=True)
axs[0,1].hist(z2_train, bins=50, color='blue',rwidth=0.78,stacked=True)
axs[0,2].hist(z3_train, bins=50, color='blue',rwidth=0.78,stacked=True)
axs[1,0].hist(z4_train, bins=50, color='blue',rwidth=0.78,stacked=True)
axs[1,1].hist(z5_train, bins=50, color='blue',rwidth=0.78,stacked=True)
axs[1,2].hist(z_train, bins=50, color='blue',rwidth=0.78,stacked=True)
axs[2,0].hist(z1_test, bins=50, color='blue',rwidth=0.78,stacked=True)
axs[2,1].hist(z2_test, bins=50, color='blue',rwidth=0.78,stacked=True)
axs[2,2].hist(z3_test, bins=50, color='blue',rwidth=0.78,stacked=True)
axs[3,0].hist(z4_test, bins=50, color='blue',rwidth=0.78,stacked=True)
axs[3,1].hist(z5_test, bins=50, color='blue',rwidth=0.78,stacked=True)
axs[3,2].hist(z_test, bins=50, color='blue',rwidth=0.78,stacked=True)
for i in range(3):
  axs[0,i].set(xlabel = titles[i+0])
  axs[1,i].set(xlabel = titles[i+3])
  axs[2,i].set(xlabel = titles[i+6])
  axs[3,i].set(xlabel = titles[i+9])
#mpvs
n, bins, patches = plt.hist(z_train, bins=50)
plt.show()
```

```
mode_index = n.argmax()

# the most frequent bin

print('the most frequent bin:(' + str(bins[mode_index]) + ',' + str(bins[mode_index+1])
+ ')')

# the mode

print('the mode:'+ str((bins[mode_index] + bins[mode_index+1])/2))

mpvtr=(bins[mode_index] + bins[mode_index+1])/2

#akribws idia gia z1_train, z2_train, z3_train, z4_train, z5_train , z_test, z1_test,
z2_test, z3_test, z4_test, z5_test

#plots ytrue vs z

plt.scatter(yf1_train, z1_train, color='r',s=1)

plt.scatter(yf2_train, z2_train, color='r',s=1)

plt.scatter(yf3_train, z3_train, color='r',s=1)

plt.scatter(yf4_train, z4_train, color='r',s=1)

plt.scatter(yf5_train, z5_train, color='r',s=1)

plt.axhline(y=0, color='b', linestyle='-')

plt.axhline(y=mpvtr, color='b', linestyle='dashed')

plt.xlabel("true")

plt.ylabel("residuals/true")

plt.title("train")

plt.ylim([-1, 1])

plt.show()

plt.scatter(yf1_test, z1_test, color='r',s=1)

plt.scatter(yf2_test, z2_test, color='r',s=1)

plt.scatter(yf3_test, z3_test, color='r',s=1)

plt.scatter(yf4_test, z4_test, color='r',s=1)

plt.scatter(yf5_test, z5_test, color='r',s=1)

plt.axhline(y=0, color='b', linestyle='-')

plt.axhline(y=mpvte, color='b', linestyle='dashed')

plt.xlabel("true")

plt.ylabel("residuals/true")
```

```
plt.title("test")
plt.ylim([-1, 1])
plt.show()
#hline plots
fig, ax = plt.subplots()
ax.hlines(y=mpv1tr, xmin=0, xmax=50, linewidth=2, color='r',label='train')
ax.hlines(y=mpv2tr, xmin=50, xmax=100, linewidth=2, color='r')
ax.hlines(y=mpv3tr, xmin=100, xmax=150, linewidth=2, color='r')
ax.hlines(y=mpv4tr, xmin=150, xmax=200, linewidth=2, color='r')
ax.hlines(y=mpv5tr, xmin=200, xmax=1000, linewidth=2, color='r')
ax.hlines(y=mpvtr, xmin=0, xmax=1000, linewidth=2, color='r')
ax.hlines(y=mpv1te, xmin=0, xmax=50, linewidth=2, color='b',label='test')
ax.hlines(y=mpv2te, xmin=50, xmax=100, linewidth=2, color='b')
ax.hlines(y=mpv3te, xmin=100, xmax=150, linewidth=2, color='b')
ax.hlines(y=mpv4te, xmin=150, xmax=200, linewidth=2, color='b')
ax.hlines(y=mpv5te, xmin=200, xmax=1000, linewidth=2, color='b')
ax.hlines(y=mpvte, xmin=0, xmax=1000, linewidth=2, color='b')
plt.xlabel('y_true')
plt.ylabel('mpv')
plt.legend()
plt.show()
```

Βιβλιογραφία

[1] Caroni C.– Oikonomou P., Statistical Regression Models, 2010

[2] Friedman J., Greedy Function Approximation: A Gradient Boosting Machine, 2001

[3] Kousouris K., Lecture Notes, Pattern Recognition and Neural Networks, 2023

[4] Koutroumbas K. – Theodoridis S., Pattern Recognition, 2008

[5] Liaw A. & Wiener M., "Classification and Regression by random Forest", R News (2002) Vol. 2/3p. 18