Ατομική Διπλωματική Εργασία

Υπολογισμός Ηλεκτρικής Κατανάλωσης Οικιών Χρησιμοποιώντας Αλγορίθμους Μηχανικής Μάθησης

Ανδρέας Λοΐζου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2013

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υπολογισμός Ηλεκτρικής Κατανάλωσης Οικιών Χρησιμοποιώντας Αλγορίθμους Μηχανικής Μάθησης

Ανδρέας Λοΐζου

Επιβλέπων Καθηγητής
Καθ. Ανδρέας Πιτσιλλίδης

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2013

Ευχαριστίες

Θα ήθελα να εκφράσω την απέραντη ευγνωμοσύνη μου και τις ευχαριστίες μου στον επιβλέποντα καθηγητή μου, Καθ. Ανδρέα Πιτσιλλίδη για την πολύτιμη καθοδήγησή του, και την συνεχή υποστήριξη που μου παρείχε κατά την διάρκεια της έρευνάς μου στα πλαίσια της Ατομικής Διπλωματικής Εργασίας. Οι συμβουλές οι οποίες μου παρέχονταν από τον καθ. Πιτσιλλίδη, και η κατανόηση, συμβολή και ενθάρρυνση στις όποιες δυσκολίες προέκυπταν καθ' όλη την έρευνα, ήταν ιδιαίτερα χρήσιμες για να αναλύσω το πρόβλημα το οποίο μελετούσα, αναλύοντάς το πολυδιάστατα και ποικιλοτρόπως, ούτως ώστε να παραχθεί κάτι εμπεριστατωμένο, σφαιρικό και ουσιώδες. Το ενδιαφέρον του, αλλά και ο χαρακτήρας του, διακατέχονταν από επαγγελματισμό αλλά και ήθος, καθώς είναι πρότυπο καθηγητή, και παράδειγμα προς μίμηση. Για αυτό έχω κερδίσει πολλά, τόσο επιστημονικές γνώσεις όσο και αρχές, οι οποίες έχουν επηρεάσει τον τρόπο σκέψης μου.

Επίσης τον ευχαριστώ ιδιαίτερα, για την ευκαιρία την οποία μου έδωσε να ασχοληθώ με ένα θέμα το οποίο είναι εκτός της περιοχής των Δικτύων, αλλά τόσο χρήσιμο για να εμβαθύνω τις γνώσεις μου στο πεδίο της μηχανικής μάθησης το οποίο και θα ακολουθήσω για τις μετέπειτα σπουδές μου. Αυτό μου έχει δώσει μία πρώτη γεύση και εμπειρία του πεδίου αυτού η οποία ήταν μία μοναδική ευκαιρία για να καταλάβω τις δυσκολίες του αλλά και τις διαδικασίες επίλυσης προβλημάτων με κατάλληλους αλγορίθμους.

Επίσης θα ήθελα να ευχαριστήσω τον Δρ. Ανδρέα Καμηλάρη, ο οποίος με επέβλεπε και καθοδηγούσε κάθε μου βήμα σε αυτή την εργασία. Μέσω των εξαιρετικά χρήσιμων συμβουλών και υποστήριξής του είχα κατανοήσει καλύτερα το μεγάλο ποσό της εργασίας που χρειάζεται να καταναλωθεί ούτως ώστε να παραχθεί μία επιστημονική εργασία με τις απαραίτητες βάσεις ώστε να είναι ευρύτατα αποδεκτή, αξιόπιστη καθώς επίσης και ολοκληρωμένη. Ήταν πάντοτε παρόν για να μου παράσχει οποιαδήποτε βοήθεια, τόσο σε πιο τεχνικά ή αλγοριθμικά θέματα, όσο και στον ρόλο συμβούλου στην γενική μου πορεία. Μου έχει επίσης παράσχει ανεκτίμητη βοήθεια στην φάση συλλογής δεδομένων. Η συνεργασία μας είναι ιδιαίτερα ωφέλιμη τόσο σε προσωπικό όσο και σε επαγγελματικό επίπεδο.

Θα ήθελα επίσης να ευχαριστήσω τον Δρ. Ιωάννη Κατάκη για τις συμβουλές του ως ειδικός στην μηχανική μάθηση, και για τον τρόπο που με βοήθησε να προσεγγίσω το πρόβλημα. Ιδιαίτερο ευχαριστώ θα ήθελα να αναφέρω στους διδακτορικούς φοιτητές Βασίλη Βασιλειάδη και Μιχάλη Αγαθοκλέους, οι οποίοι με είχαν εντάξει στις αρχές των νευρωνικών δικτύων, καθώς επίσης στον τρόπο με τον οποίο θα τα χρησιμοποιήσω στην περίπτωση αυτή.

Τέλος ευχαριστώ τους γονείς μου, Λοΐζο και Δέσπω, οι οποίοι είχαν δείξει πλήρη κατανόηση στην προσπάθειά μου αυτή, και με είχαν υποστηρίξει με κάθε τρόπο για να φέρω εις πέρας τα καθήκοντά μου ως προπτυχιακός φοιτητής πληροφορικής.

Περίληψη

Η Διπλωματική αυτή εργασία πραγματεύεται την εκτίμηση της ηλεκτρικής κατανάλωσης μίας οικίας χρησιμοποιώντας αλγορίθμους μηχανικής μάθησης, με βάση τα χαρακτηριστικά της . Αυτά συμπεριλαμβάνουν για παράδειγμα το μέγεθος της οικίας, ή τους τύπους θέρμανσής της.

Στην παρούσα Διπλωματική Εργασία περιγράφεται λεπτομερώς ο τρόπος συλλογής των δεδομένων τα οποία έχουμε μελετήσει, καθώς και για κάθε χαρακτηριστικό των δεδομένων κάποιες στατιστικές του ιδιότητες.

Στη συνέχεια περιγράφονται τα αποτελέσματα τα οποία προκύπτουν από νευρωνικά δίκτυα, τα οποία είχα υλοποιήσει, και είχαν εκπαιδευτεί στα δεδομένα τα οποία έχουμε προαναφέρει. Αναφέρονται τα αποτελέσματά τους, τα οποία δείχνουν μία σαφή επιτυχία των αλγορίθμων να παράξουν μία σχέση μεταξύ των χαρακτηριστικών μίας οικίας και της ηλεκτρικής της κατανάλωσης. Πιο συγκεκριμένα το μέσο απόλυτο σφάλμα το οποίο παράγεται από τους αλγορίθμους είναι στις πλείστες των περιπτώσεων λιγότερο από το μισό της τυπικής απόκλισης.

Το επόμενο κεφάλαιο μελετά την χρησιμοποίηση ακόμη περισσότερων αλγορίθμων για την επίλυση του προβλήματος. Χρησιμοποιούνται Γκαουσιανές διαδικασίες, Ισοτονική Παλινδρόμηση, ελάχιστος μέσος των τετραγώνων, γραμμική παλινδρόμηση, βηματική παλινδρόμηση, Δίκτυα RBF, καθώς επίσης Παλινδρόμηση SMO και ταξινομητής PLS. Και σε αυτό το κεφάλαιο, είχαμε δείξει ότι οι αλγόριθμοι αυτοί μπορούν επίσης να παράξουν ικανοποιητικά αποτελέσματα, με τους αλγορίθμους βηματικής παλινδρόμησης, και ελαχίστου μέσου των τετραγώνων να είναι οι καλύτεροι. Όπως και με τα νευρωνικά δίκτυα, έτσι και εδώ οι αλγόριθμοι έχουν επιτύχει να ρίξουν το μέσο απόλυτο σφάλμα τους τις πλείστες των περιπτώσεων κάτω του μισού της τυπικής απόκλισης

Στο τελευταίο κεφάλαιο, ανακεφαλαιώνουμε τα αποτελέσματα των αλγορίθμων, καθώς επίσης αναφέρεται μελλοντική εργασία η οποία δύναται να γίνει, και να βοηθήσει ακόμη περισσότερο την έρευνα αυτή, να παράξει ακόμη πιο χρήσιμα και ορθά αποτελέσματα

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Μηχανική Μάθηση

Η μηχανική μάθηση, είναι ένας κλάδος της τεχνητής νοημοσύνης, ο οποίος μελετά την κατασκευή και μελέτη συστημάτων τα οποία μπορούν να *μάθουν* από τα δεδομένα. Για παράδειγμα, η μηχανική μάθηση έχει χρησιμοποιηθεί σε φίλτρα ανεπιθύμητων μηνυμάτων (spam filter). Στην περίπτωση αυτή, μελετώντας ένα σύνολο από ανεπιθύμητα μηνύματα, αλλά επίσης και ένα σύνολο από επιθυμητά, μαθαίνει μέσω εκπαίδευσης να τα ξεχωρίζει, και με τον ερχομό ενός νέου άγνωστου μηνύματος, να μπορεί να το ταξινομεί σε δύο από τις δύο αυτές κατηγορίες

Το 1959, ο Arthur Samuel, όρισε τη μηχανική μάθησης ως "Το πεδίο μελέτης το οποίο δίνει στους υπολογιστές την ικανότητα να σκέφτονται, χωρίς να είναι ρητά προγραμματισμένοι" [1].

Υπάρχουν αρκετές εφαρμογές της μηχανικής νοημοσύνης, όπως για παράδειγμα η ρομποτική, ή και η αναγνώριση χαρακτήρων (OCR). Επίσης εφαρμογές της μηχανικής μάθησης βρίσκονται σε κοινωνικά δίκτυα, όπως το Facebook το οποίο μαθαίνει από τις πράξεις του χρήστη, και του συνιστά καινούριους φίλους, ή ομάδες, ή ακόμη του εμφανίζει συχνότερα στην αρχική σελίδα του χρήστη, μηνύματα από τα άτομα με τα οποία έχει

μεγαλύτερες επαφές. Σημαντική μελέτη στο πεδίο της μηχανικής μάθησης έχει γίνει και στο Πανεπιστήμιο Κύπρου, από την αυτοματοποίηση ελέγχου λογισμικού [2], στην πρόβλεψη ποδοσφαιρικών στοιχημάτων [3] αλλά και μέχρι και στον έλεγχο συμφόρησης στο επίπεδο TCP/IP[4].

Η μηχανική μάθηση έχει χρησιμοποιηθεί ευρέως, για αναγνώριση όπως ομιλίας, γραφικών χαρακτήρων, προσώπων, τοπίων, αλλά και σε αυτοκίνητα τα οποία οδηγούνται χωρίς κάποιον να τα καθοδηγεί, για αποτελεσματική αναζήτηση στο διαδίκτυο, αλλά και πρόσφατα με την εφαρμογή προσωπικού βοηθού SIRI στο iPhone. Η μηχανική μάθηση χρησιμοποιείται τόσο εντατικά, ώστε το πιο πιθανό να την χρησιμοποιούμε δεκάδες φορές κάθε ημέρα χωρίς πραγματικά να το γνωρίζουμε.

Αποτελείται από τρεις κυρίως τύπους μάθησης. Αυτοί συμπεριλαμβάνουν την *επιβλεπόμενη μάθηση* στην οποία ανήκουν οι παραμετρικοί και μη παραμετρικοί αλγόριθμοι, μηχανές διανυσμάτων υποστήριξης (SVM), και νευρωνικά δίκτυα. Επίσης υπάρχει η μη επιβλεπόμενη μάθηση στην οποία συμπεριλαμβάνονται κυρίως η ομαδοποίηση (clustering), η μείωση διαστάσεων, τα συστήματα σύστασης (recommender systems). Ο τρίτος τύπος μάθησης είναι η ενισχυτική μάθηση, στην οποία οι αλγόριθμοι ναι μεν παίρνουν κάποια ανάδραση (feedback) σε αντίθεση με τη μη επιβλεπόμενη μάθηση, αλλά όχι την τιμή που αναμένεται να παράξουν σε αντίθεση με την επιβλεπόμενη μάθηση.

1.2 Κοινωνικός Ηλεκτρισμός

Η εφαρμογή Κοινωνικός Ηλεκτρισμός (Social Electricity) είναι μια νέα εφαρμογή η οποία είναι υλοποιημένη στη πλατφόρμα κοινωνικής δικτύωσης Facebook που μας βοηθά να αντιληφθούμε την ηλεκτρική ενέργεια που καταναλώνουμε, μέσω συγκρίσεων με την αντίστοιχη ενέργεια που καταναλώνουν οι φίλοι μας και οι γείτονες μας καθώς και με την συνολική κατανάλωση στην γειτονιά/οδό/χωριό/επαρχία στην οποία διαμένουν. Μόνο μέσω αποτελεσματικών και ρεαλιστικών συγκρίσεων μπορεί ο καταναλωτής να αντιληφθεί την ενεργειακή του συμπεριφορά. Με αυτό τον τρόπο, οι Κύπριοι πολίτες θα συνειδητοποιήσουν εάν καταναλώνουν μεγάλα ποσά ενέργειας, και αυτή η συνειδητοποίηση αναμένεται ότι θα τους ωθήσει να εργαστούν ώστε να μειώσουν την κατανάλωση τους καθώς και τα χρηματικά ποσά που καταβάλλουν κάθε διμηνία.

Πρόκειται για μια συνεργασία του Πανεπιστημίου Κύπρου και της Αρχής Ηλεκτρισμού Κύπρου (ΑΗΚ), με στόχο την μείωση της κατανάλωσης ενέργειας στη Κύπρο και της συνειδητοποίησης των Κύπριων πολιτών για θέματα ενέργειας.

Όπως αναφέρεται στην σελίδα της εφαρμογής στο Facebook, *"Αποστολή* της εφαρμογής είναι η συνειδητοποίηση των Κύπριων πολιτών όσον αφορά την ηλεκτρική ενέργεια που καταναλώνουν, κάτι που αναμένεται να τους ωθήσει να την μειώσουν, προστατεύοντας το περιβάλλον και παράλληλα εξοικονομώντας χρήματα."

Εικόνα 1.1 – Στιγμιότυπο από την εφαρμογή «Κοινωνικός Ηλεκτρισμός»

Η εφαρμογή αυτή είχε σημαντική επιρροή στους Κύπριους χρήστες της, καθώς αποτελεί μία καινοτόμα εφαρμογή. Πιο συγκεκριμένα η εφαρμογή αυτή έχει κερδίσει το πρώτο βραβείο σε διεθνή διαγωνισμό για Πράσινες Εφαρμογές Τεχνολογιών Πληροφορικής και Επικοινωνιών (2nd Green ICT Application Challenge), που διοργανώνει η Διεθνής Ένωση Τηλεπικοινωνιών (International Telecommunications Union, ITU). [5]

Οι συγκρίσεις με γείτονες και φίλους μας προσφέρουν μια ένδειξη σχετικά με τα ποσά ηλεκτρικής ενέργειας που καταναλώνουμε, δεν μπορούν όμως να είναι επακριβείς, επειδή το κάθε άτομο ζει σε διαφορετική οικία με διαφορετικά χαρακτηριστικά.

Στη προσπάθεια μας να κάνουμε την εφαρμογή πιο αποτελεσματική για τους Κύπριους πολίτες, θα θέλαμε να προσθέσουμε την δυνατότητα να καταχωρεί ο κάθε Κύπριος τα στοιχεία της οικίας του και να μπορεί να συγκρίνει αυτόματα την ηλεκτρική του κατανάλωση μόνο με Κύπριους πολίτες που πληρούν παρόμοια χαρακτηριστικά. Έτσι, θα γνωρίζει το

κάθε άτομο αν καταναλώνει χαμηλά/μέτρια/ψηλά ποσά ηλεκτρικής ενέργειας, μέσω συγκρίσεων μεταξύ ανθρώπων που ζουν οπουδήποτε στη Κύπρο, αλλά η οικία τους προσφέρει παρόμοια χαρακτηριστικά με αυτούς! Με αυτό το θέμα πραγματευόμαστε στη Διπλωματική αυτή εργασία

1.3 Ορισμός Προβλήματος Υπολογισμού Ηλεκτρικής Κατανάλωσης με Χρήση Αλγορίθμων Μηχανικής Μάθησης

Η ηλεκτρική κατανάλωση μίας οικίας, αποτελεί ένα από τους μεγαλύτερους παράγοντες εξόδων των κατοίκων της οικίας, αλλά και ένα παράγοντα που επιδρά αρνητικά στην οικολογική αλλοτρίωση. Πολλές φορές στην περίπτωση που γνωρίζουμε εκ των προτέρων κάποιο γεγονός, το οποίο έχει αρνητικό αντίκτυπο, συνειδητοποιούμε καλύτερα την πραγματική κατάσταση, και έτσι προσπαθούμε να το αποφύγουμε ή να μειώσουμε την επίδρασή του.

Έτσι, μας είχε δημιουργηθεί εύλογα το ερώτημα: Μπορούμε να εκτιμήσουμε την ηλεκτρική κατανάλωση ενός σπιτιού ούτως ώστε ένας κάτοικος να μπορεί να γνωρίζει το ποσό της ηλεκτρικής ενέργειας που εκτιμάται να καταναλώσει;

Ο προβληματισμός αυτός μας ώθησε στο να κάνουμε μία παγκόσμια καινοτομία δίχως προηγούμενο. Στο να μελετήσουμε τους παράγοντες από τους οποίους η ηλεκτρική κατανάλωση επηρεάζεται, και μέσω από αυτούς να προσπαθήσουμε να εκτιμήσουμε την κατανάλωση μίας οικίας.

Πιο συγκεκριμένα, ποια είναι τα χαρακτηριστικά ενός σπιτιού, τα οποία συμβάλλουν στην κατανάλωσή ηλεκτρισμού:

Πώς, και ποιοι αλγόριθμοι είναι σε θέση να μας εκπαιδευτούν ούτως ώστε να μας παρέχουν μία εκτίμηση για την ηλεκτρική κατανάλωση δεδομένου τα χαρακτηριστικά σε μία οικία;

1.4 Σκοπός Διπλωματικής Εργασίας

Σκοπός της Ατομικής Διπλωματικής Εργασίας αυτής, είναι τα αποτελέσματα τα οποία θα προκύψουν από την πιο πάνω μελέτη, να ενσωματωθούν, στην εφαρμογή κοινωνικός ηλεκτρισμός. Έτσι θα παράγεται μία εξατομικευμένη εκτίμηση της κατανάλωσης του χρήστη της εφαρμογής, για να είναι σε θέση να κάνει μία πιο ρεαλιστική σύγκριση. Πιο συγκεκριμένα, θα είναι σε θέση να συγκρίνει την πραγματική του κατανάλωση, με εκείνη

την τιμή την οποία αναμένεται να είχε, η οποία υπολογίζεται με τη μελέτη **παρόμοιων σπιτιών** και όχι απλά σε σχέση με τους φίλους του στην εφαρμογή.

Στην διάθεσή μας, δεν υπήρχαν δεδομένα από αισθητήρες τα οποίοι μπορούσαν να κατανοούν και να καταγράφουν την χρήση ηλεκτρικών συσκευών ή λαμπτήρων για να παρακολουθούμε την ζήτηση ηλεκτρικής ενέργειας σε μία οικία.

Εφόσον η δομή του προβλήματός μας είναι τέτοια, εστιαστήκαμε στο να έχουμε δείγμα από πολλές οικίες, ούτως ώστε να μπορούμε να συσχετίσουμε τα χαρακτηριστικά μίας οικίας με την κατανάλωσή της. Οι αισθητήρες θα μας έδιναν πραγματικά δεδομένα την ίδια ώρα. (real time data). Εφόσον όμως εμείς δεν είχαμε στην διάθεσή μας τέτοιου είδους δεδομένα, είχαμε εστιαστεί στα στατικά, σταθερά χαρακτηριστικά μίας οικίας.

Πολλά χαρακτηριστικά ίσως να επηρεάζουν μία οικία όπως την θέρμανση, τον κλιματισμό, την περιοχή της οικίας, το μέγεθός της κλπ, και τα οποία περιγράφονται με σαφήνεια στο Κεφάλαιο 2 – Δεδομένα. Έτσι είχαμε επιλέξει μία λίστα από αυτά, τα οποία προφανώς είχαν την μεγαλύτερη επίδραση στην κατανάλωση, και τα οποία συλλέξαμε από ένα δείγμα 700 ανθρώπων μοιράζοντας έντυπα.

Για αυτές τις 700 οικίες, είχαμε συλλέξει τόσο τα χαρακτηριστικά τους, όσο και την κατανάλωσή τους, για κάθε μία από τις έξι διμηνίες του 2012.
Έχοντας αυτά τα δεδομένα, πως θα μπορούσαμε να εξάγουμε τα κατάλληλα συμπεράσματα, και τις σχέσεις μεταξύ τους ούτως ώστε να είμαστε θέση μέσω από αυτά να πάρουμε τα επιθυμητά αποτελέσματα που είναι να μπορούμε να εκτιμήσουμε με σχετικά μικρό σφάλμα την ηλεκτρική κατανάλωση μίας οικίας δεδομένου ότι έχουμε τα χαρακτηριστικά της;

Τέτοιου είδους προβλήματα λύνονται με τη χρήση αλγορίθμων μηχανικής μάθησης. Οι αλγόριθμοι αυτοί είναι ικανοί να εκπαιδεύονται από τα δεδομένα τα οποία έχουμε συλλέξει, και να βρίσκουν τη σχέση τους με το επιθυμητό αποτέλεσμα. Για τον λόγο αυτό είχαμε επιλέξει να χρησιμοποιήσουμε μία μεγάλη γκάμα αλγορίθμων, και μέσω παρατηρήσεων να φθάσουμε στον αλγόριθμο που μας δίνει το καλύτερο αποτέλεσμα.

Πιο συγκεκριμένα είχαμε χρησιμοποιήσει νευρωνικά δίκτυα που εκπαιδεύονται με τον αλγόριθμο ανάστροφης μετάδοσης με δύο τρόπους. Ο πρώτος είναι για παραγωγή μίας τιμής, ενώ ο δεύτερος αφού είχαμε χωρίσει την επιθυμητή ηλεκτρική κατανάλωση σε πεδία (ranges - ομάδες) να μας αναφέρει σε ποια από αυτές κατατάσσεται η οικία η οποία είχε περαστεί ως

είσοδος. Επίσης είχαμε χρησιμοποιήσει νευρωνικά δίκτυα τύπου RBF τα οποία έχουν την δυνατότητα να διαχωρίσουν μη γραμμικά διαχωρίσιμα δεδομένα, για να μας παράγουν την επιθυμητή κατανάλωση. Επιπλέων είχαμε χρησιμοποιήσει γραμμική παλινδρόμηση (linear regression), αλλά και μηχανές διανυσμάτων υποστήριξης (SVM). Είχαμε επίσης χρησιμοποιήσει τον αλγόριθμο SOM για να δούμε πόσο καλά αποτελέσματα δίνουν αλγόριθμοι ομαδοποίησης, αλλά και τους αλγορίθμους K μέσων, και K κοντινότερων γειτόνων.

Επίσης επεξεργαζόμασταν τα δεδομένα πριν από κάθε εκτέλεση των αλγορίθμων, ούτως ώστε να είμαστε σε θέση να φέρουμε τα δεδομένα σε μία μορφή χρήσιμη, η οποία θα ήταν υποβοηθητική για να μπορεί να αξιοποιηθούν τα δεδομένα όσο το δυνατόν καλύτερα από τους αλγορίθμους.

Το πρόβλημα αυτό είναι πρόβλημα εκτίμησης ηλεκτρικής κατανάλωσης μέσω αλγορίθμων μηχανικής μάθησης, όπως δηλαδή και αναφέρεται στον τίτλο της διπλωματικής μου εργασίας. Ελπίζω τα αποτελέσματα της εργασίας μου να έρθουν σε επαφή σε όσο το δυνατό περισσότερα άτομα, ούτως ώστε να τα βοηθήσουν να συνειδητοποιήσουν την ηλεκτρική κατανάλωσή τους εκ των προτέρων για να μπορούν να την μειώσουν.

Κεφάλαιο 2

Δεδομένα

2.1 Τι δεδομένα χρειάζεται να συλλέξουμε;

Το πρόβλημα της εκτίμησης της ηλεκτρικής κατανάλωσης μέσω των χαρακτηριστικών του σπιτιού, θα πρέπει να λυθεί μέσω ενός αλγορίθμου μηχανικής μάθησης. Χρησιμοποιώντας τον αλγόριθμο αυτό χρειάζεται να επιστρέψουμε μία συνάρτηση
$f(< X >)$ όπου το διάνυσμα <X> είναι το διάνυσμα <x1,x2,x3…xn>. Οι μεταβλητές χ αποτελούν τα χαρακτηριστικά του σπιτιού, οι οποίες θα εξετάζονται από τον αλγόριθμο, και $f(< X >)$ η συνάρτηση που επιστρέφεται από αυτόν. Χρησιμοποιώντας την $f(< X >)$, με είσοδο ένα άγνωστο διάνυσμα <X> ενός νέου σπιτιού, θα πρέπει να επιστρέφεται μία τιμή κατανάλωσης, όσο γίνεται πιο κοντά στην πραγματική.

Τα δεδομένα πρέπει να επιλεχθούν με κατάλληλο τρόπο, ώστε σύμφωνα με την φύση του προβλήματός μας, να είναι σχετικά με την κατανάλωση μίας οικίας, για να μπορούμε σύμφωνα με αυτά να υπολογίσουμε την κατανάλωση της οικίας.

Τι επηρεάζει την κατανάλωση κάθε οικίας; Η κάθε οικία είναι διαφορετική με οποιαδήποτε άλλη. Διαφέρουν στο μέγεθος, στους κατοίκους, στις συσκευές θέρμανσης, στην τοποθεσία κ.α. Όλα τα παραπάνω χαρακτηριστικά, συμβάλλουν όλα μαζί στην ηλεκτρική κατανάλωση μίας οικίας. Κάποια περισσότερο ενώ κάποια λιγότερο. Βέβαια, ίσως ο πιο καθοριστικός παράγοντας όσον αφορά την ηλεκτρική κατανάλωση δεν είναι μετρήσιμος όπως οι παραπάνω αλλά είναι ο ανθρώπινος παράγοντας. Η κατανάλωση μίας οικίας επηρεάζεται τόσο από την συνείδηση των κατοίκων της, όσο και από τα τεχνικά χαρακτηριστικά του σπιτιού, και των συσκευών του. Για παράδειγμα κάποιοι ίσως να αφήνουν τα φώτα ή τις τηλεοράσεις ανοικτές ακόμη και αν δεν βρίσκονται στα δωμάτια αυτά. Ή ακόμη να έχουν τα κλιματιστικά να λειτουργούν αλόγιστα, ή σε ώρες που δεν χρειάζονται. Ο παράγοντας αυτός επηρεάζει τα μάλα την κατανάλωση, και συνεπώς σε μεγάλο βαθμό τα δεδομένα μας. Παρόλα αυτά εμείς έπρεπε να είμαστε σε θέση να βρούμε είναι τα μετρήσιμα στοιχεία τα οποία επηρεάζουν την κατανάλωση μίας οικίας.

Οι σημαντικότεροι παράγοντες που επηρεάζουν την ηλεκτρική κατανάλωση μίας κατοικίας είναι ως επί το πλείστον οι ακόλουθες:

Το μέγεθος του σπιτιού: Το μέγεθος του σπιτιού είναι ένας από τους κυριότερους παράγοντες οι οποίοι επηρεάζουν την ηλεκτρική κατανάλωση. Ένα μικρότερο σπίτι χρειάζεται λιγότερη ενέργεια για θέρμανση, ή για φωτισμό.

Κάτοικοι σπιτιού: Όσο περισσότεροι είναι οι κάτοικοι σε ένα σπίτι τόσο περισσότερο αυξάνεται και η κατανάλωση του σπιτιού καθώς ο καθένας έχει διαφορετικές ανάγκες σε ενέργεια, και όλοι μαζί αυξάνουν την συνολική ζήτηση για ενέργεια. Για παράδειγμα αν στο σπίτι υπάρχει μόνο ένας κάτοικος, ίσως να χρειάζεται μόνο ένα φως αναμμένο στο σπίτι, ενώ όσο αυξάνονται οι κάτοικοι υπάρχει η ανάγκη για χρήση περισσότερων δωματίων στο σπίτι, και ως εκ τούτου ζήτηση ενέργειας για περισσότερες συσκευές.

Συνολικά Δωμάτια Σπιτιού: Τα δωμάτια καθορίζουν ένα σημαντικό ρόλο, καθώς όσα περισσότερα δωμάτια χρησιμοποιούνται από τους κατοίκους τόσο περισσότερη ενέργεια καταναλώνεται για θέρμανση και φωτισμό

Περιοχή σπιτιού: Η περιοχή όπως αντιλαμβανόμαστε καθορίζει την ζήτηση ενέργειας ιδίως για θέρμανση ή και κλιματισμό. Αν για παράδειγμα η κατοικία βρίσκεται σε ορεινές περιοχές, η ζήτηση ενέργειας για θέρμανση τον χειμώνα θα είναι ιδιαίτερα αυξημένη, ενώ μειωμένη θα είναι η ζήτηση για κλιματισμό σε σύγκριση με τις κατοικίες που βρίσκονται σε αστικές περιοχές.

Ιδιόκτητο/Ενοικιαζόμενο σπίτι: Η παράμετρος αυτή αν και ίσως όχι τόσο καθοριστική όσο οι υπόλοιπες έχει σημασία όσον αφορά την ηλεκτρική κατανάλωση.

Θέρμανση: Τι τύποι θέρμανσης χρησιμοποιούνται στο σπίτι; Εάν χρησιμοποιείται για παράδειγμα κλιματιστικό ή θερμάνσεις αλογόνου για θέρμανση, τότε η ζήτηση για ηλεκτρικό ρεύμα θα αυξηθεί.

Κλιματιστικά – Ψύξη: Τι χρησιμοποιείται για να μειωθεί η θερμοκρασία εντός σπιτιού; Χρησιμοποιείται κλιματιστικό ή ανεμιστήρας;

Υπάρχουν πολλαπλοί άλλοι παράγοντες οι οποίοι επηρεάζουν την ηλεκτρική κατανάλωση μίας οικίας, αλλά δεν μπορούσαμε αλλά και δεν χρειαζόταν να τις συλλέξουμε όλες. Ο λόγος είναι ότι με πολλές μεταβλητές οι αλγόριθμοι μηχανικής μάθησης ίσως να δυσκολεύονται να μάθουν, και έτσι να γενικεύουν συνεπώς η μείωση της απόδοσής τους. Επίσης ένα έντυπο για μία έρευνα, επιβάλλεται να είναι μικρό, για να συμπληρωθεί με ακρίβεια, και από μεγαλύτερο δείγμα.

Υπάρχουν πολλοί άλλοι παράγοντες οι οποίοι επηρεάζουν. Αυτοί κάποιες φορές ίσως να είναι λιγότερο μετρήσιμοι. Αυτοί συμπεριλαμβάνουν μεταξύ άλλων τα πιο κάτω:

- Τι λαμπτήρες χρησιμοποιούνται στο σπίτι;
- Πόσες ώρες οι κάτοικοι βρίσκονται σπίτι;
- Πόσο ισχυρά είναι τα κλιματιστικά/ θερμάνσεις;
- Πόσες ώρες ανάβει η κάθε ξεχωριστή συσκευή θέρμανσης ή κλιματισμού;
- Τι συσκευές υπάρχουν στο σπίτι;
- Υπάρχει πιεστικό σύστημα νερού;
- Ευνοεί την κατανάλωση ο προσανατολισμός της οικίας;
- Υπάρχουν ανανεώσιμες πηγές ενέργειας στην οικία;
- Χρησιμοποιείται ηλιακός θερμοσίφωνας;
- Υπάρχει ενεργειακή συνείδηση στο σπίτι;
- Εργάζονται οι κάτοικοι του σπιτιού στο σπίτι;
- Πόσο καινούριες είναι οι συσκευές εντός του σπιτιού;
- Πόσα ψυγεία υπάρχουν στο σπίτι;
- Τι χρησιμοποιείται για θερμομόνωση;
- Πόσο παλιό είναι το σπίτι;
- Από τι έχει κατασκευαστεί το σπίτι;

Καθώς και άλλες πολλές, τις οποίες τις είχαμε υπόψη μας. Εργασία μας στην φάση συλλογής δεδομένων ήταν να επιλέξουμε τις πιο αντιπροσωπευτικές και σημαντικές από αυτούς τους παράγοντες.

2.2 Τι δεδομένα συλλέξαμε;

Στα πλαίσια της παρακολούθησης των παραμέτρων που επηρεάζουν την κατανάλωση, είχαμε μοιράσει εκατοντάδες έντυπα, για να πάρουμε δεδομένα.

Τα δεδομένα τα οποία έχουμε υποστηρίξει ότι με την επεξεργασία τους είναι δυνατόν να μας δώσουν μία εκτίμηση της ηλεκτρικής κατανάλωσης μίας οικίας είναι τα ακόλουθα:

Το όνομα στην βάση δεδομένων αναφέρεται, λόγω του ότι θα ακολουθήσουν στατιστικά του κάθε χαρακτηριστικού στο κεφάλαιο αυτό, καθώς και αποτελέσματα των αλγορίθμων σε επόμενα κεφάλαια, οι οποίοι έχουν ως συστατικά τους τα ονόματα αυτά.

Αριθμός Χαρακτηριστικού	Νόημα	Όνομα στην Βάση
1	Το σπίτι βρίσκεται σε πόλη?	URBAN
3	Το σπίτι βρίσκεται σε χωριό;	RURAL
3	Το σπίτι βρίσκεται σε ορεινή περιοχή?	MOUNTAINOUS
4	Το σπίτι είναι ιδιόκτητο ή ενοικιαζόμενο?	PROPRIETARY
5	Το σπίτι είναι κατοικία (αληθές) ή διαμέρισμα (ψευδές)?	HOUSE
6	Μένετε σε μεζονέτα?	MAISONETTE
7	Το σπίτι έχει πισίνα?	POOL
8	Το σπίτι έχει Jacuzzi ή SPA?	JACUZZI_SPA
9	Το σπίτι είναι θερμομονωτικό?	INSULATION

Πίνακας 2.1 Γενικές πληροφορίες οικίας. Πεδίο ορισμού μεταβλητών Αληθές (0) ή Ψευδές (1)

Επίσης λόγω του ότι η θέρμανση και ο κλιματισμός, παίζουν καθοριστικό παράγοντα στην ηλεκτρική κατανάλωση, θεωρήσαμε σωστό να ληφθούν υπόψη. Έτσι είχαμε ζητήσει από το δείγμα μας και λάβει τους παρακάτω τύπους θέρμανσης και κλιματισμού.

Αριθμός Χαρακτηριστικού	Τύπος θέρμανσης	Όνομα στην βάση
10	Κεντρική θέρμανση πετρελαίου	CENTRAL_HEATING_PETROLEUM
11	Καλοριφέρ	RADIATOR
12	Θερμάνσεις Αλογόνου	HALOGEN_HEATERS
13	Θερμαινόμενο Πάτωμα	FLOOR_HEATING
14	Σόμπες Γκαζιού	GAS_HEATERS
15	Τζάκι	FIREPLACE
16	Θερμοσυσσωρευτές ΑΗΚ	STORAGE_HEATERS
17	Σύστημα κλιματισμού με ζεστό αέρα	HOT_AIR_CONDITIONER
18	Κανένας	NONE_HEATING

Πίνακας 2.2 Τύποι Θέρμανσης οι οποίοι ζητήθηκαν από το δείγμα και λήφθηκαν υπόψη από τους αλγορίθμους. Πεδίο ορισμού μεταβλητών Αληθές (0) ή Ψευδές (1)

Τύποι κλιματισμού (Πεδίο ορισμού: Αληθείς ψευδής)

Αριθμός Χαρακτηριστικού	Τύπος κλιματισμού	Όνομα στην Βάση
19	Σύστημα κλιματισμού	COLD_AIR_CONDITIONER
20	Αεριστήρας	FAN
21	Κανένας	NONE_COOLING

Πίνακας 2.3 Τύποι κλιματισμού οι οποίοι ζητήθηκαν από το δείγμα και λήφθηκαν υπόψη από τους αλγορίθμους. Πεδίο ορισμού μεταβλητών Αληθές (0) ή Ψευδές (1)

Επιπλέων χαρακτηριστικά οικίας (Πεδίο ορισμού: Φυσικοί αριθμοί)

Αριθμός χαρακτηριστικού	Χαρακτηριστικό	Όνομα στην βάση δεδομένων
22	Τετραγωνικά μέτρα σπιτιού	SQUARE_METERS
23	Συνολικός αριθμός	TOTAL_NUMBERS
24	Ενήλικες κάτοικοι	ADULTS
25	Ανήλικοι κάτοικοι	UNDERAGE

Πίνακας 2.4 Αριθμητικά χαρακτηριστικά οικιών τα οποία ζητήθηκαν από το δείγμα και λήφθηκαν υπόψη από τους αλγορίθμους. Πεδίο ορισμού μεταβλητών οι Φυσικοί Αριθμοί.

2.3 Διαδικασία Συλλογής Δεδομένων

Λόγω προστασίας προσωπικών δεδομένων των εθελοντών από τους οποίους ζητούσαμε να συμπληρώσουν τα χαρακτηριστικά της οικίας τους, τα δεδομένα έπρεπε να τα ζητήσουμε σε έντυπη μορφή και όχι ηλεκτρονικά. Ο λόγος είναι ότι στην έντυπη μορφή θα μπορούσαν να υπογράψουν για να μας δοθεί εξουσιοδότηση ούτως ώστε να πάρουμε την ηλεκτρική τους κατανάλωση από την ΑΗΚ.

Το γεγονός όμως ότι οι φόρμες έπρεπε να ήταν σε έντυπη μορφή, είχε δυσκολέψει πολύ το έργο μας, καθώς έπρεπε να δίνουμε το έντυπο ένα – ένα στον κάθε ενδιαφερόμενο, να το συμπληρώνει, να το επιστρέφει πίσω, καθώς επίσης να καταχωρούμε ξανά όλες τις απαντήσεις του ηλεκτρονικά σε μία βάση δεδομένων. Λόγω αυτού όμως, η συλλογή των δεδομένων δεν μπορούσε να γίνει γρήγορα και εύκολα χρησιμοποιώντας πολύ απλά την τεχνολογία. Έτσι ένα σχετικά τεράστιο μέρος της διπλωματικής εργασίας αυτής χρησιμοποιήθηκε για συλλογή δεδομένων. Επίσης χωρίς ένα λογικό αριθμό από δεδομένα δεν θα μπορούσαν να γίνονται παράλληλα πειράματα, και έτσι είχε αποδοθεί μία πολύ μεγάλη προσπάθεια στην συλλογή δεδομένων.

Τα δεδομένα είχαν παραληφθεί από ένα αντιπροσωπευτικό δείγμα ανθρώπων διαφόρων ηλικιών αλλά και από διαφορετικά μέρη διαμονής από όλη την Κύπρο.

Έντυπα είχαν μαζευτεί από γνωστούς, ή αγνώστους μοιράζοντας έντυπα σε κτίρια όπως στο MALL of Cyprus. Εκεί η συμμετοχή ήταν μεγάλη εφόσον οι εθελοντές συμπλήρωσαν αρκετά ερωτηματολόγια. Εκτός αυτού, έντυπα είχαν επίσης μοιραστεί και σε δημοτικά σχολεία. Τα έντυπα τα οποία είχαν μοιραστεί στα δημοτικά σχολεία ήταν ένας αρκετά μεγάλος αριθμός, και ένα μεγάλο ποσοστό είχε συμπληρωθεί από τους γονείς των παιδιών, και επιστράφηκε πίσω σε εμάς.

Ο λόγος που προωθήσαμε έντυπα στα δημοτικά σχολεία ήταν για να συμπληρωθούν κατά συρροή σε μικρό σχετικά χρονικό διάστημα, με ακρίβεια, και επίσης για να συμπληρωθεί στο σπίτι, όπου οι γονείς των παιδιών είχαν στην διάθεσή τους και τους αριθμούς των λογαριασμών τους στην ΑΗΚ. Έτσι θα μπορούσαμε και εμείς να τους βρούμε ευκολότερα μέσα από την βάση δεδομένων της ΑΗΚ. Ευχαριστώ ιδιαίτερα για αυτό την Διεύθυνση

Δημοτικής Εκπαίδευσης η οποία μου παρείχε την άδεια να μοιράσω τα έντυπα στα δημοτικά σχολεία.

Πήραμε επίσης έντυπα επίσης μέσω ηλεκτρονικού ταχυδρομείου μετά από προώθηση του ερευνητικού μας έργου από την MTN. Είχα προσωπικά επικοινωνήσει μαζί τους καθώς επίσης και με άλλες εταιρείες μέσω της σελίδας κοινωνικής δικτύωσης Facebook. Πιο συγκεκριμένα, η MTN η οποία έχει πέραν των 60 χιλιάδων φίλων στο κοινωνικό δίκτυο είχε ανακοινώσει στον σύνδεσμο http://goo.gl/XK03G το εξής:

«Θέλετε να συμμετάσχετε σε μια καινοτομική έρευνα για την κατανάλωση ηλεκτρικής ενέργειας, που γίνεται από Κύπριους φοιτητές για πρώτη φορά στον κόσμο και μας αφορά όλους; Κάντε κλικ στο πιο κάτω link και ακολουθήστε τις οδηγίες!

http://www.socialelectricity.net/

Do you want to participate in an innovative research about electricity consumption that is conducted by Cypriot students for the first time in the world and concerns all of us? Click the link below and follow the instructions!

http://www.socialelectricity.net/indexEn.html»

Η συγκεκριμένη ανακοίνωση, μία από τις ελάχιστες ανακοινώσεις τις οποίες κάνει η MTN πέρα από την προώθηση των υπηρεσιών τους, είχε προωθήσει εκτός από το έντυπό μας, και το ερευνητικό μας έργο Social Electricity, και τους ευχαριστώ ιδιαίτερα για αυτό.

Συνολικά, από την διαδικασία συλλογής δεδομένων είχαμε μαζέψει 703 έντυπα, όπου γύρω στα 500 ήταν έγκυρα, και τα οποία χρησιμοποιήσαμε για εκπαίδευση των αλγορίθμων μας. Τα υπόλοιπα περίπου 200 είχαν ελλιπείς τιμές, ή είχαν άλλα μη έγκυρα στοιχεία.

Το έντυπο το οποίο έχει μοιρασθεί υπάρχει στο παράρτημα [A1], η αγγλική του έκδοση στο [A2] ενώ μία ανανεωμένη ελληνική έκδοση η οποία περιέχει πεδία ταυτότητας, μοναδικού αριθμού οικίας στην ΑΗΚ, καθώς και δύο επιπλέων τύπους θερμάνσεων υπάρχει στο [A3] .

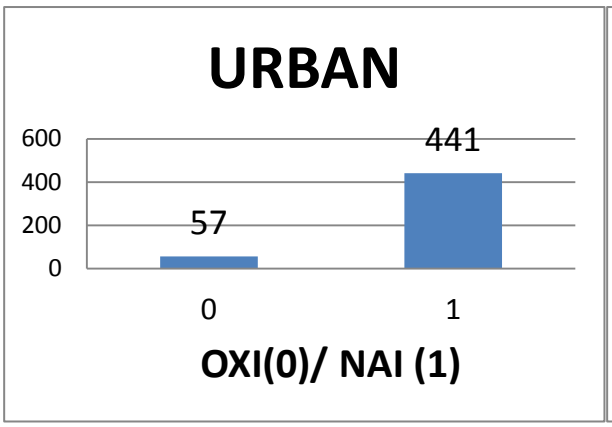
2.4 Ανάλυση και γραφική αναπαράσταση δεδομένων

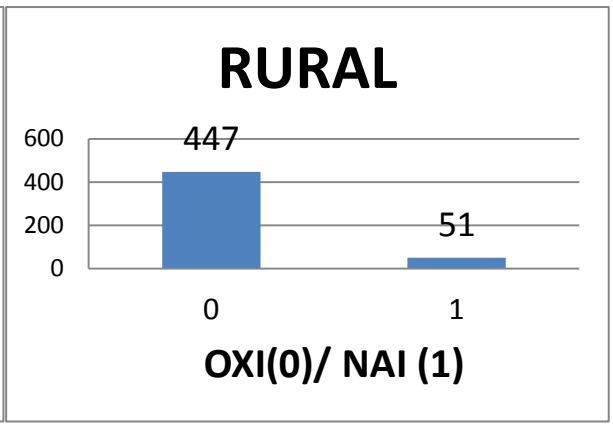
2.4.1 Χαρακτηριστικά οικιών

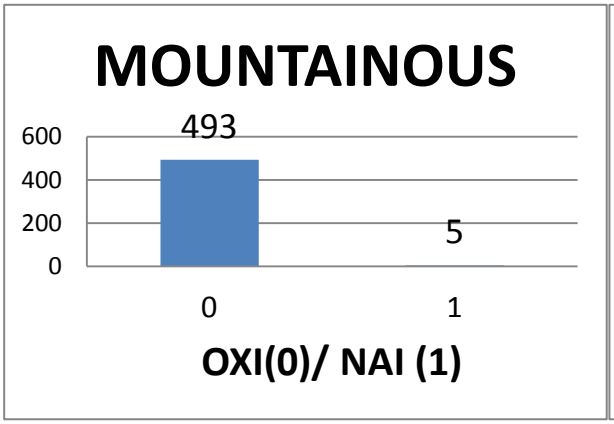
Τα χαρακτηριστικά των οικιών που έχουμε μελετήσει, μπορούμε να τα κατανοήσουμε καλύτερα, εάν δούμε πως αυτά κατανέμονται στις τιμές που παίρνουν. Έτσι πιο κάτω φαίνονται οι κατανομές αυτές των χαρακτηριστικών των οικιών που έχουμε μελετήσει για το 2012. Εδώ να σημειώσουμε ότι τα στοιχεία αυτά ήταν σταθερά και στις έξι διμηνίες που είχαμε μελετήσει. Οι πιο κάτω γραφικές παραστάσεις δίνονται για να έχουμε μία πιο ολοκληρωμένη εικόνα σχετικά με τα χαρακτηριστικά των 500 σπιτιών τα οποία χρησιμοποιήσαμε και στις 6 διμηνίες.

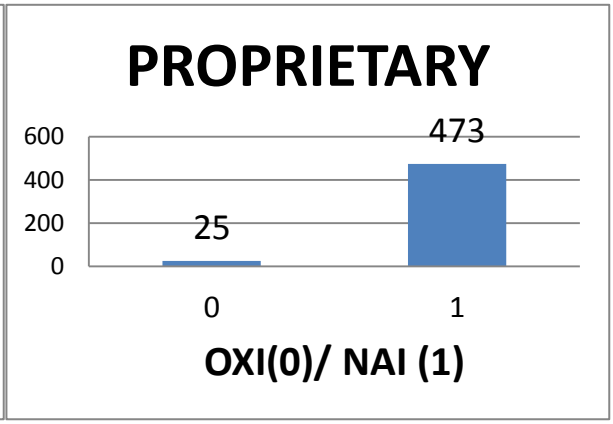
Πιο κάτω ακολουθούν οι γραφικές παραστάσεις των υπολοίπων χαρακτηριστικών στο δείγμα. Τα ονόματα τα οποία αναφέρονται στους τίτλους των γραφικών παραστάσεων επεξηγούνται αναλυτικά στο υποκεφάλαιο 2.2 της αναφοράς αυτής.

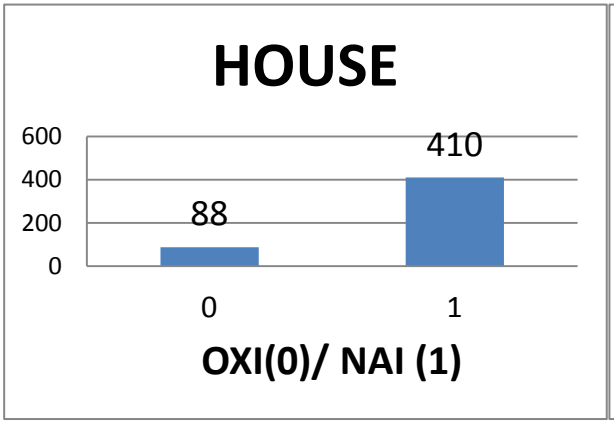
Παρατηρούμε επίσης ότι πολλά από αυτά βρίσκονται σε αρκετή ανομοιομορφία μεταξύ τους. Για παράδειγμα πολύ λίγα άτομα μένουν σε ορεινές περιοχές ή έχουν jacuzzi ή SPA. Την ανομοιομορφία αυτή καλούνται να την αντιμετωπίσουν οι αλγόριθμοι μηχανικής μάθησης. Γενικά, εάν υπάρχει ανομοιομωρφία στα δεδομένα, αυτό που προτείνεται είναι να υπάρχουν πολλά δεδομένα, ούτως ώστε οι αλγορίθμοι να μπορούν να βγάζουν λογικά και σαφή συμπεράσματα. Το πλήθος των 500 εγγραφών που είχαμε ήταν αρκετό για τα δεδομένα της Κύπρου, και δεδομένου της μη χρήσης της τεχνολογίας για την συλλογή τους, αλλά σχετικά αρκετά μικρό για την εκπαίδευση των αλγορίθμων μηχανικής μάθησης.

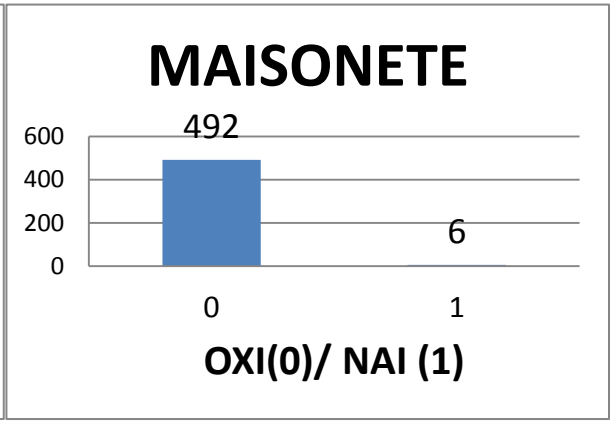

URBAN
600
400
200
0
57
441
0
1
OXI(0)/ NAI (1)

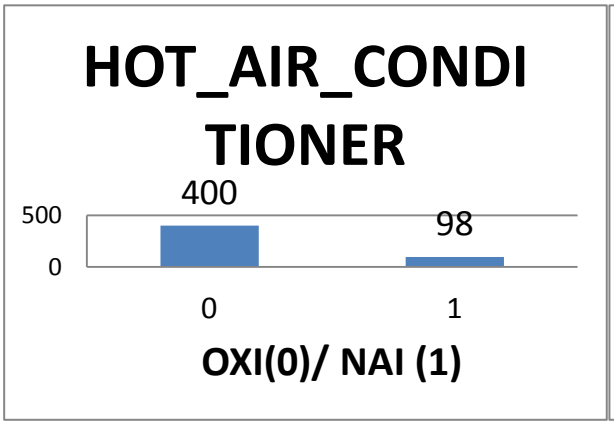

RURAL
600
400
200
0
447
51
0
1
OXI(0)/ NAI (1)

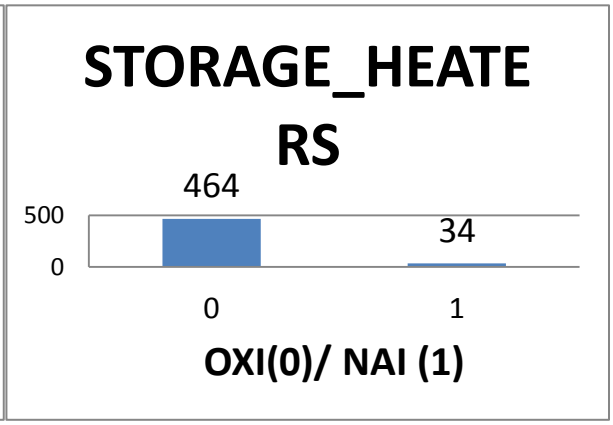

MOUNTAINOUS
600
400
200
0
493
5
0
1
OXI(0)/ NAI (1)

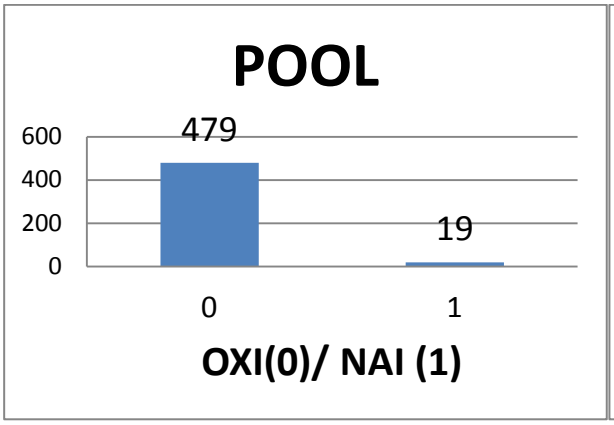

PROPRIETARY
600
400
200
0
25
473
0
1
OXI(0)/ NAI (1)

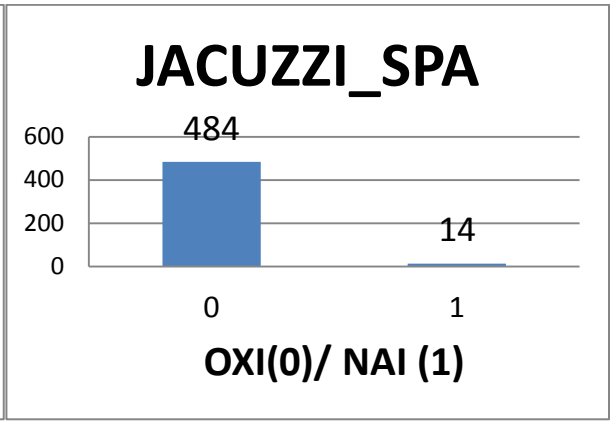

HOUSE
600
400
200
0
88
410
0
1
OXI(0)/ NAI (1)

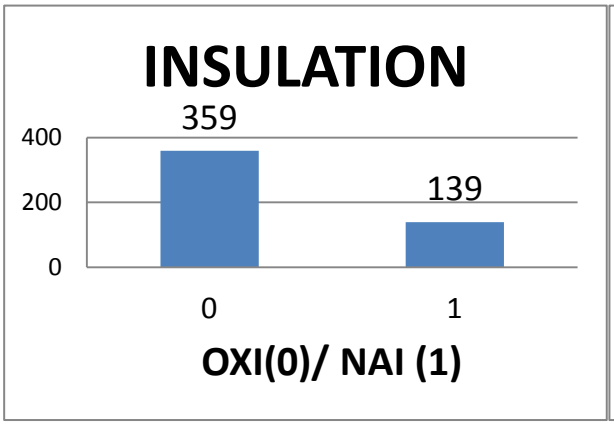

MAISONETE
600
400
200
0
492
6
0
1
OXI(0)/ NAI (1)

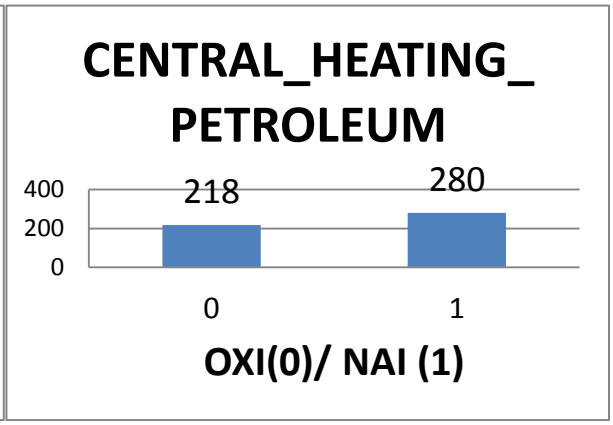

HOT_AIR_CONDI TIONER
500
0
400
98
0
1
OXI(0)/ NAI (1)

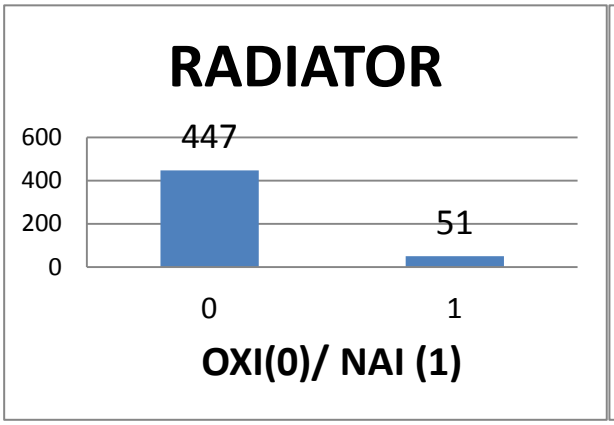

STORAGE_HEATE RS
500
0
464
34
0
1
OXI(0)/ NAI (1)

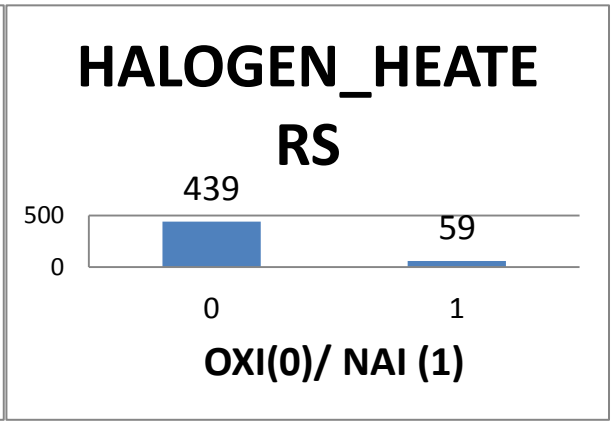

POOL
600
400
200
0
479
19
0
1
OXI(0)/ NAI (1)

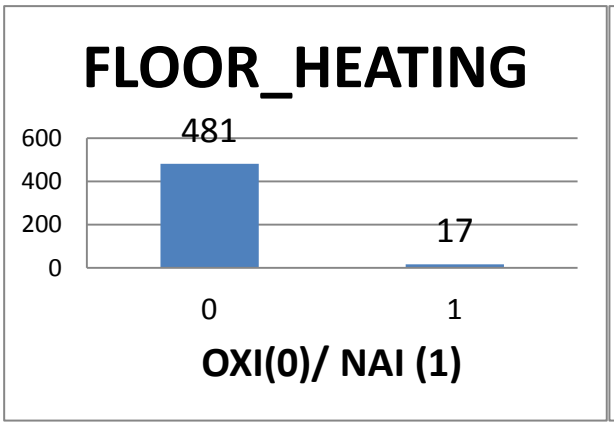

JACUZZI_SPA
600
400
200
0
484
14
0
1
OXI(0)/ NAI (1)

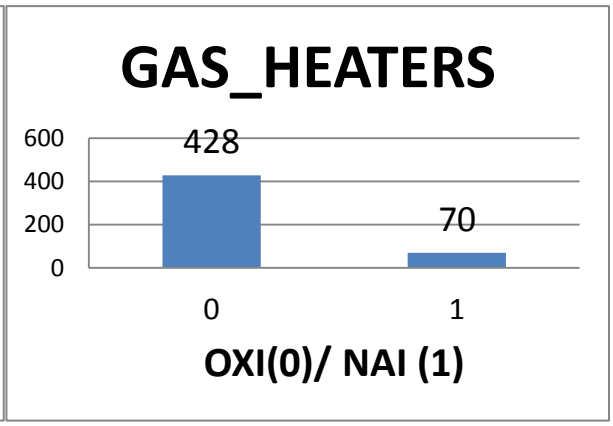

INSULATION
400
200
0
359
139
0
1
OXI(0)/ NAI (1)

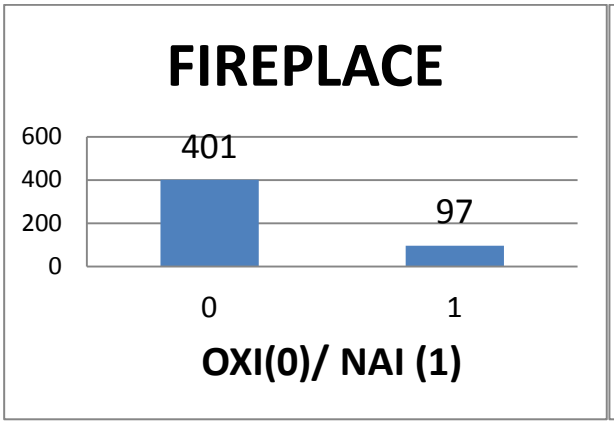

CENTRAL_HEATING_
PETROLEUM
400
200
0
218
280
0
1
OXI(0)/ NAI (1)

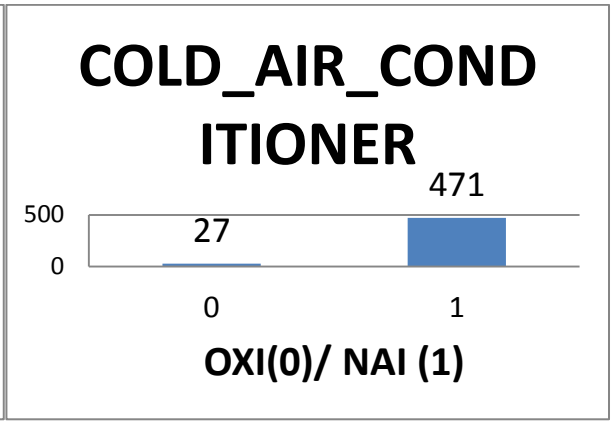

RADIATOR
600
400
200
0
447
51
0
1
OXI(0)/ NAI (1)

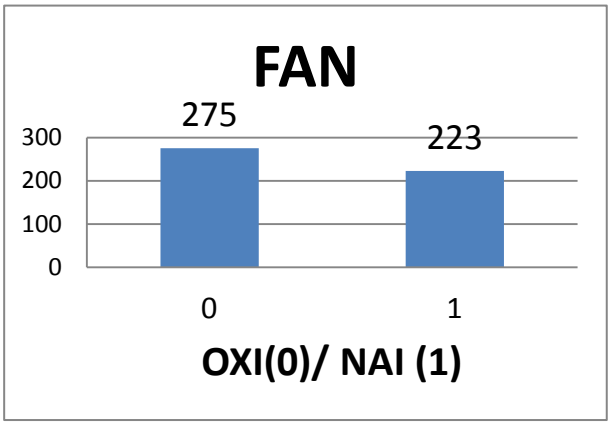

HALOGEN_HEATE
RS
500
0
439
59
0
1
OXI(0)/ NAI (1)


FLOOR_HEATING
600
400
200
0
481
17
0
1
OXI(0)/ NAI (1)

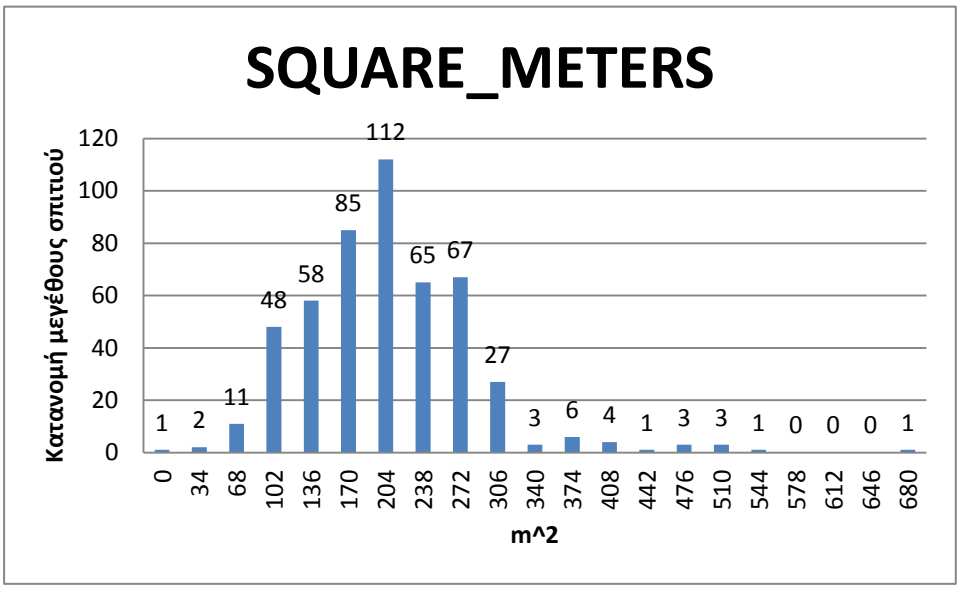

GAS_HEATERS
600
400
200
0
428
70
0
1
OXI(0)/ NAI (1)



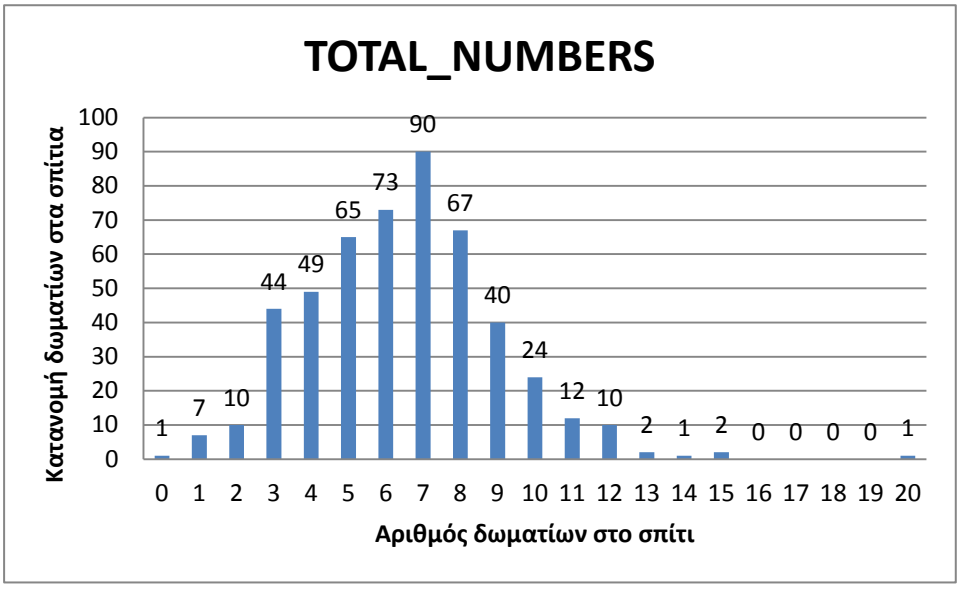




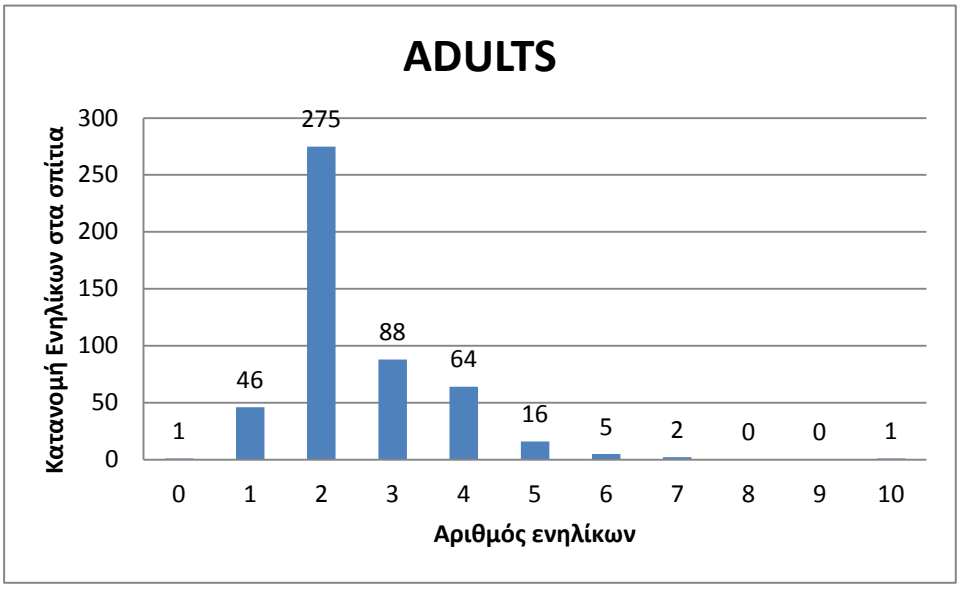
Τα πιο πάνω χαρακτηριστικά παίρνουν τιμές αληθής ή ψευδής, ενώ τα πιο κάτω παίρνουν θετικούς ακαίρεους.



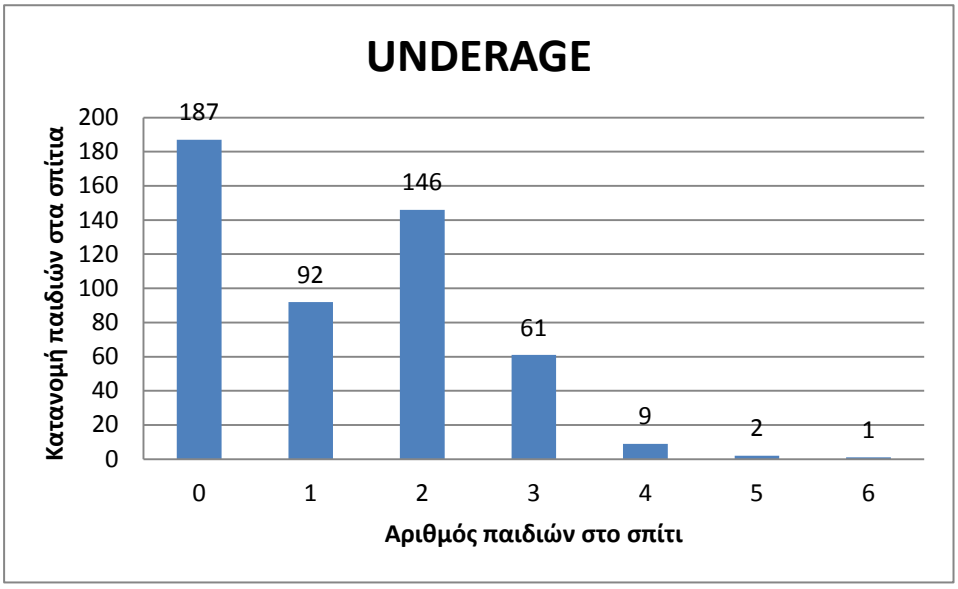
Σχήμα 2.1 : Γραφική παράσταση κατανομής τετραγωνικών μέτρων σπιτιού



Σχήμα 2.2 : Γραφική παράσταση κατανομής των συνολικών δωματίων στις οικίες



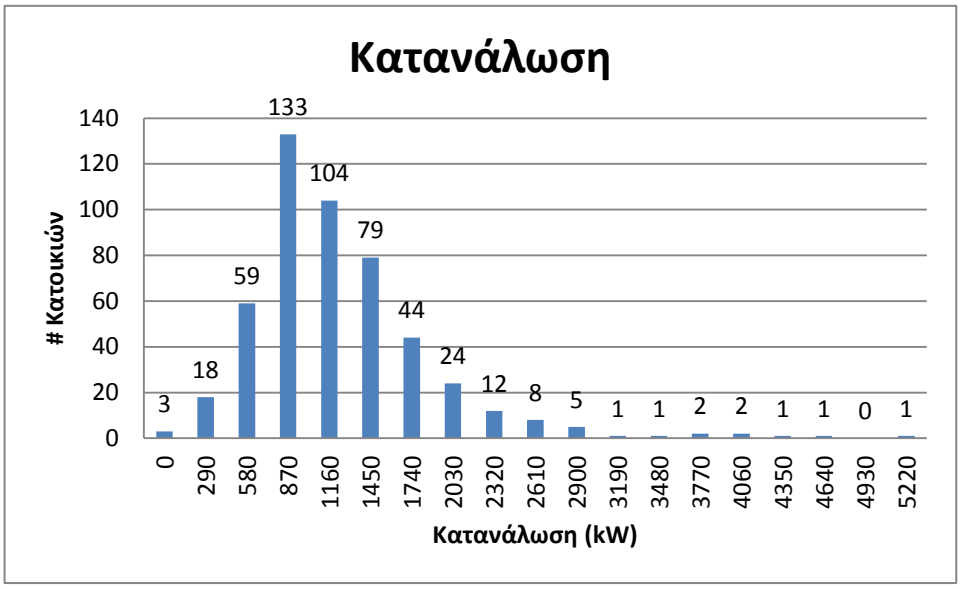
Σχήμα 2.3 : Γραφική παράσταση κατανομής ενηλίκων κατοίκων των οικιών



Σχήμα 2.4 : Γραφική παράσταση κατανομής ανήλικων κατοίκων στα σπίτια

2.4.2 Διμηνία Ιανουαρίου – Φεβρουαρίου

Στην παράγραφο αυτή φαίνεται το ιστόγραμμα της ηλεκτρικής κατανάλωσης των οικιών κατά την περίοδο Ιανουαρίου και Φεβρουαρίου 2012 στο δείγμα μας στο σχήμα 2.5. Επίσης ο Πίνακας 2.5 παρουσιάζει αναλυτικά τα δεδομένα των χαρακτηριστικών των δεδομένων μας, τα οποία καλούνται οι αλγόριθμοι μηχανικής μάθησης να λάβουν υπόψη. Πιο συγκεκριμένα ο Πίνακας αυτός παρουσιάζει για κάθε χαρακτηριστικό των οικιών και της κατανάλωσης τους τον μικρότερο, μεγαλύτερο, μέσο (mean) και τυπική απόκλιση (standard deviation), ούτως ώστε να μπορούμε να κατανοήσουμε τα αποτελέσματα των αλγορίθμων μας και να εξάγουμε ορθά συμπεράσματα.



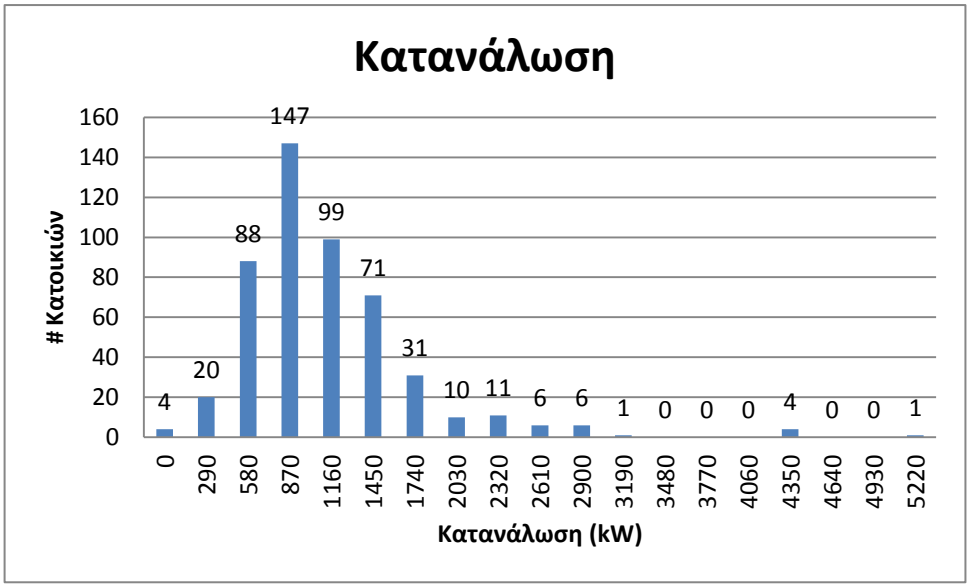
Σχήμα 2.5 : Ιστόγραμμα ηλεκτρικής κατανάλωσης των οικιών κατά τη διμηνία Ιανουαρίου – Φεβρουαρίου 2012 του δείγματός μας.

Όνομα	Ελάχιστο (Minimum)	Μέγιστο (Maximum)	Μέσος (Mean)	Τυπική απόκλιση (Standard Deviation)
TOTAL_CONSUMPTION	0	4946	1092,247	645,227
URBAN	0	1	0,886	0,319
RURAL	0	1	0,102	0,303
MOUNTAINOUS	0	1	0,010	0,100
PROPRIETARY	0	1	0,950	0,219
HOUSE	0	1	0,823	0,382
MAISONETTE	0	1	0,010	0,100
POOL	0	1	0,038	0,192
JACUZZI_SPA	0	1	0,028	0,165
INSULATION	0	1	0,279	0,449
HOT_AIR_CONDITIONER	0	1	0,197	0,398
STORAGE_HEATERS	0	1	0,068	0,252
CENTRAL_HEATING_PETROLEUM	0	1	0,562	0,497
RADIATOR	0	1	0,102	0,303
HALOGEN_HEATERS	0	1	0,118	0,323
FLOOR_HEATING	0	1	0,034	0,182
FIREPLACE	0	1	0,141	0,348
GAS_HEATERS	0	1	0,195	0,396
NONE_HEATING	0	1	0,024	0,154
COLD_AIR_CONDITIONER	0	1	0,946	0,227
FAN	0	1	0,448	0,498
NONE_COOLING	0	1	0,006	0,077
SQUARE_METERS	0	651	189,369	78,507
TOTAL_NUMBERS	0	20	6,478	2,536
ADULTS	0	10	2,510	1,101
UNDERAGE	0	6	1,243	1,175

Πίνακας 2.5 : Μικρότερη(minimum) τιμή, μεγαλύτερη (maximum) τιμή, μέση τιμή (mean) και τυπική απόκλιση(standard deviation) για το κάθε χαρακτηριστικό στο δείγμα μας για την περίοδο Ιανουαρίου – Φεβρουαρίου 2012.

2.4.3 Διμηνία Μαρτίου – Απριλίου

Στην παράγραφο αυτή φαίνεται το ιστόγραμμα της ηλεκτρικής κατανάλωσης των οικιών κατά την περίοδο Μαρτίου και Απριλίου 2012 στο δείγμα μας στο σχήμα 2.6. Επίσης ο Πίνακας 2.6 παρουσιάζει αναλυτικά τα δεδομένα των χαρακτηριστικών των δεδομένων μας, τα οποία καλούνται οι αλγόριθμοι μηχανικής μάθησης να λάβουν υπόψη. Πιο συγκεκριμένα ο Πίνακας αυτός παρουσιάζει για κάθε χαρακτηριστικό των οικιών και της κατανάλωσης τους τον μικρότερο, μεγαλύτερο, μέσο (mean) και τυπική απόκλιση (standard deviation), ούτως ώστε να μπορούμε να κατανοήσουμε τα αποτελέσματα των αλγορίθμων μας και να εξάγουμε ορθά συμπεράσματα.



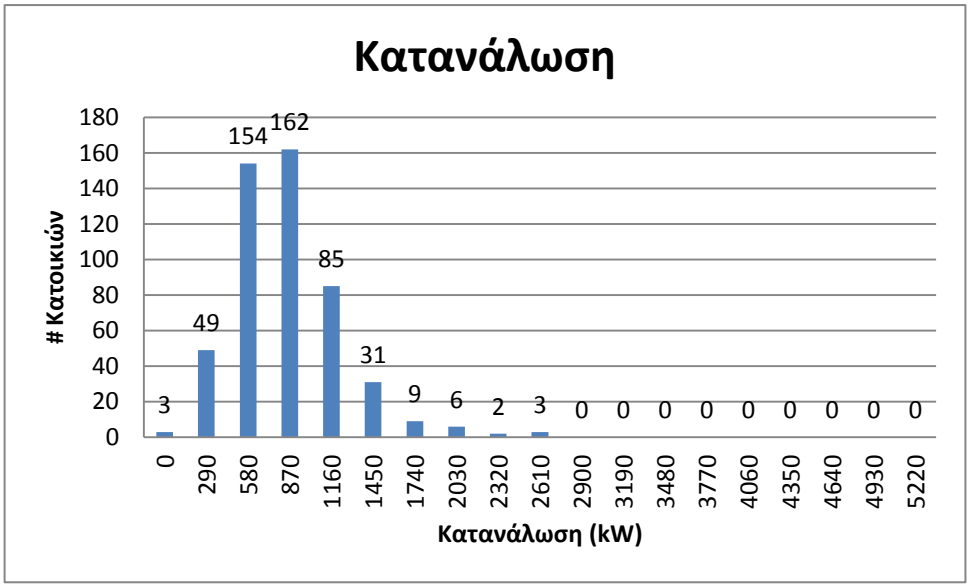
Σχήμα 2.6 : Ιστόγραμμα ηλεκτρικής κατανάλωσης των οικιών κατά τη διμηνία Μαρτίου - Απριλίου 2012 του δείγματός μας.

Όνομα	Ελάχιστο (Minimum)	Μέγιστο (Maximum)	Μέσος (Mean)	Τυπική απόκλιση (Standard Deviation)
TOTAL_CONSUMPTION	0	5456	986,564	644,466
URBAN	0	1	0,884	0,321
RURAL	0	1	0,104	0,306
MOUNTAINOUS	0	1	0,010	0,100
PROPRIETARY	0	1	0,948	0,223
HOUSE	0	1	0,823	0,382
MAISONETTE	0	1	0,010	0,100
POOL	0	1	0,036	0,187
JACUZZI_SPA	0	1	0,028	0,165
INSULATION	0	1	0,281	0,450
HOT_AIR_CONDITIONER	0	1	0,197	0,398
STORAGE_HEATERS	0	1	0,068	0,252
CENTRAL_HEATING_PETROLEUM	0	1	0,560	0,497
RADIATOR	0	1	0,102	0,303
HALOGEN_HEATERS	0	1	0,118	0,323
FLOOR_HEATING	0	1	0,034	0,182
FIREPLACE	0	1	0,141	0,348
GAS_HEATERS	0	1	0,193	0,395
NONE_HEATING	0	1	0,024	0,154
COLD_AIR_CONDITIONER	0	1	0,944	0,231
FAN	0	1	0,448	0,498
NONE_COOLING	0	1	0,006	0,077
SQUARE_METERS	0	651	188,908	78,506
TOTAL_NUMBERS	0	20	6,464	2,541
ADULTS	0	10	2,508	1,101
UNDERAGE	0	6	1,233	1,167

Πίνακας 2.6 : Μικρότερη(minimum) τιμή, μεγαλύτερη (maximum) τιμή, μέση τιμή (mean) και τυπική απόκλιση(standard deviation) για το κάθε χαρακτηριστικό στο δείγμα μας για την περίοδο Μαρτίου - Απριλίου 2012.

2.4.4 Διμηνία Μαΐου – Ιουνίου

Στην παράγραφο αυτή φαίνεται το ιστόγραμμα της ηλεκτρικής κατανάλωσης των οικιών κατά την περίοδο Μαΐου και Ιουνίου 2012 στο δείγμα μας στο σχήμα 2.7. Επίσης ο Πίνακας 2.7 παρουσιάζει αναλυτικά τα δεδομένα των χαρακτηριστικών των δεδομένων μας, τα οποία καλούνται οι αλγόριθμοι μηχανικής μάθησης να λάβουν υπόψη. Πιο συγκεκριμένα ο Πίνακας αυτός παρουσιάζει για κάθε χαρακτηριστικό των οικιών και της κατανάλωσης τους τον μικρότερο, μεγαλύτερο, μέσο (mean) και τυπική απόκλιση (standard deviation), ούτως ώστε να μπορούμε να κατανοήσουμε τα αποτελέσματα των αλγορίθμων μας και να εξάγουμε ορθά συμπεράσματα.



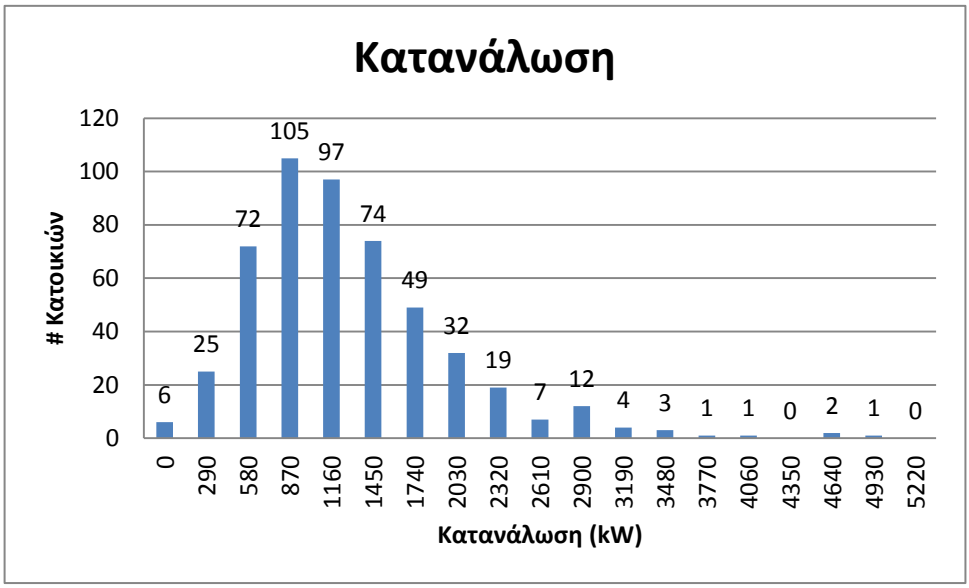
Σχήμα 2.7 : Ιστόγραμμα ηλεκτρικής κατανάλωσης των οικιών κατά τη διμηνία Μαΐου - Ιουνίου 2012 του δείγματός μας.

Όνομα	Ελάχιστο (Minimum)	Μέγιστο (Maximum)	Μέσος (Mean)	Τυπική απόκλιση (Standard Deviation)
TOTAL_CONSUMPTION	0	2409	714,119	381,491
URBAN	0	1	0,883	0,322
RURAL	0	1	0,105	0,307
MOUNTAINOUS	0	1	0,010	0,099
PROPRIETARY	0	1	0,943	0,233
HOUSE	0	1	0,820	0,385
MAISONETTE	0	1	0,010	0,099
POOL	0	1	0,038	0,190
JACUZZI_SPA	0	1	0,028	0,164
INSULATION	0	1	0,283	0,451
HOT_AIR_CONDITIONER	0	1	0,200	0,400
STORAGE_HEATERS	0	1	0,067	0,251
CENTRAL_HEATING_PETROLEUM	0	1	0,558	0,497
RADIATOR	0	1	0,101	0,302
HALOGEN_HEATERS	0	1	0,123	0,329
FLOOR_HEATING	0	1	0,034	0,181
FIREPLACE	0	1	0,139	0,346
GAS_HEATERS	0	1	0,192	0,394
NONE_HEATING	0	1	0,026	0,159
COLD_AIR_CONDITIONER	0	1	0,945	0,229
FAN	0	1	0,448	0,498
NONE_COOLING	0	1	0,006	0,077
SQUARE_METERS	0	651	188,457	78,731
TOTAL_NUMBERS	0	20	6,436	2,544
ADULTS	0	10	2,499	1,099
UNDERAGE	0	6	1,234	1,174

Πίνακας 2.7 : Μικρότερη(minimum) τιμή, μεγαλύτερη (maximum) τιμή, μέση τιμή (mean) και τυπική απόκλιση(standard deviation) για το κάθε χαρακτηριστικό στο δείγμα μας για την περίοδο Μαΐου - Ιουνίου 2012.

2.4.5 Διμηνία Ιουλίου – Αυγούστου

Στην παράγραφο αυτή φαίνεται το ιστόγραμμα της ηλεκτρικής κατανάλωσης των οικιών κατά την περίοδο Ιουλίου και Αυγούστου 2012 στο δείγμα μας στο σχήμα 2.8. Επίσης ο Πίνακας 2.8 παρουσιάζει αναλυτικά τα δεδομένα των χαρακτηριστικών των δεδομένων μας, τα οποία καλούνται οι αλγόριθμοι μηχανικής μάθησης να λάβουν υπόψη. Πιο συγκεκριμένα ο Πίνακας αυτός παρουσιάζει για κάθε χαρακτηριστικό των οικιών και της κατανάλωσης τους τον μικρότερο, μεγαλύτερο, μέσο (mean) και τυπική απόκλιση (standard deviation), ούτως ώστε να μπορούμε να κατανοήσουμε τα αποτελέσματα των αλγορίθμων μας και να εξάγουμε ορθά συμπεράσματα.



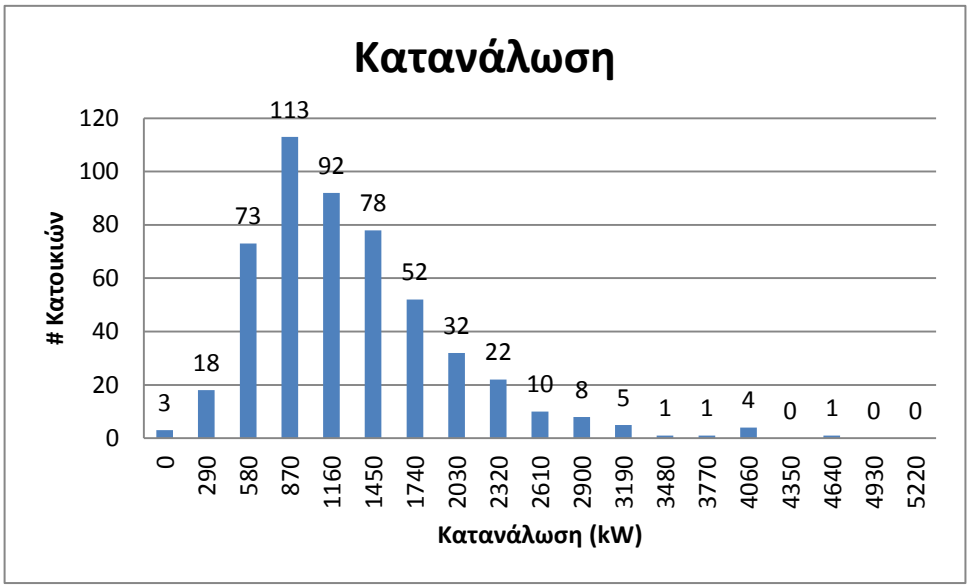
Σχήμα 2.8 : Ιστόγραμμα ηλεκτρικής κατανάλωσης των οικιών κατά τη διμηνία Ιουλίου - Αυγούστου 2012 του δείγματός μας.

Όνομα	Ελάχιστο (Minimum)	Μέγιστο (Maximum)	Μέσος (Mean)	Τυπική απόκλιση (Standard Deviation)
TOTAL_CONSUMPTION	0	5899	1149,775	741,270
URBAN	0	1	0,885	0,320
RURAL	0	1	0,104	0,305
MOUNTAINOUS	0	1	0,010	0,099
PROPRIETARY	0	1	0,941	0,235
HOUSE	0	1	0,820	0,385
MAISONETTE	0	1	0,010	0,099
POOL	0	1	0,039	0,194
JACUZZI_SPA	0	1	0,027	0,163
INSULATION	0	1	0,282	0,450
HOT_AIR_CONDITIONER	0	1	0,198	0,399
STORAGE_HEATERS	0	1	0,067	0,249
CENTRAL_HEATING_PETROLEUM	0	1	0,558	0,497
RADIATOR	0	1	0,100	0,300
HALOGEN_HEATERS	0	1	0,121	0,327
FLOOR_HEATING	0	1	0,035	0,185
FIREPLACE	0	1	0,137	0,344
GAS_HEATERS	0	1	0,190	0,393
NONE_HEATING	0	1	0,027	0,163
COLD_AIR_CONDITIONER	0	1	0,945	0,228
FAN	0	1	0,446	0,498
NONE_COOLING	0	1	0,008	0,088
SQUARE_METERS	0	651	188,910	78,965
TOTAL_NUMBERS	0	20	6,425	2,546
ADULTS	0	10	2,493	1,095
UNDERAGE	0	6	1,239	1,176

Πίνακας 2.8 : Μικρότερη(minimum) τιμή, μεγαλύτερη (maximum) τιμή, μέση τιμή (mean) και τυπική απόκλιση(standard deviation) για το κάθε χαρακτηριστικό στο δείγμα μας για την περίοδο Ιουλίου - Αυγούστου 2012.

2.4.6 Διμηνία Σεπτέμβρη – Οκτώβρη

Στην παράγραφο αυτή φαίνεται το ιστόγραμμα της ηλεκτρικής κατανάλωσης των οικιών κατά την περίοδο Σεπτεμβρίου και Οκτωβρίου 2012 στο δείγμα μας στο σχήμα 2.9. Επίσης ο Πίνακας 2.9 παρουσιάζει αναλυτικά τα δεδομένα των χαρακτηριστικών των δεδομένων μας, τα οποία καλούνται οι αλγόριθμοι μηχανικής μάθησης να λάβουν υπόψη. Πιο συγκεκριμένα ο Πίνακας αυτός παρουσιάζει για κάθε χαρακτηριστικό των οικιών και της κατανάλωσης τους τον μικρότερο, μεγαλύτερο, μέσο (mean) και τυπική απόκλιση (standard deviation), ούτως ώστε να μπορούμε να κατανοήσουμε τα αποτελέσματα των αλγορίθμων μας και να εξάγουμε ορθά συμπεράσματα.



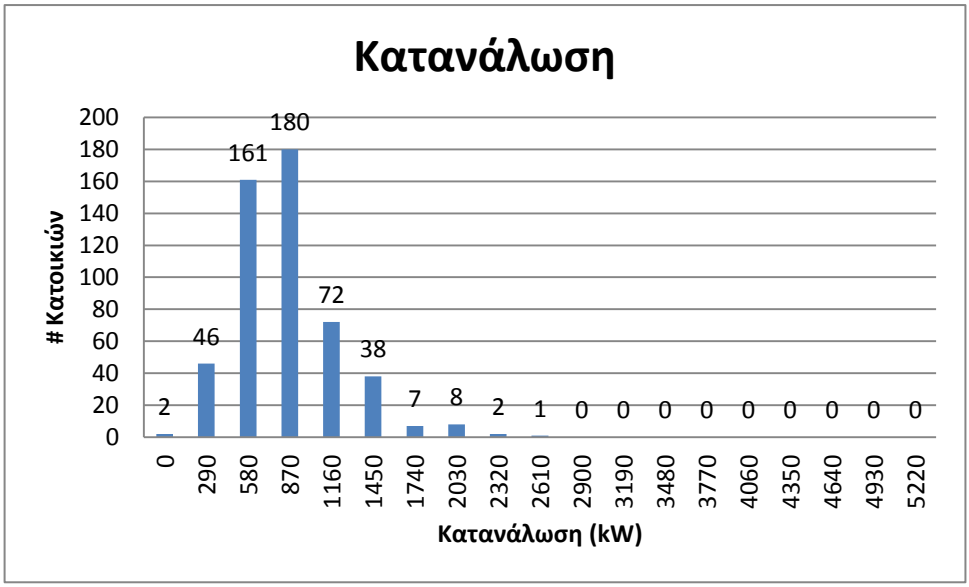
Σχήμα 2.9 : Ιστόγραμμα ηλεκτρικής κατανάλωσης των οικιών κατά τη διμηνία Σεπτεμβρίου – Οκτωμβρίου 2012 του δείγματός μας.

Όνομα	Ελάχιστο (Minimum)	Μέγιστο (Maximum)	Μέσος (Mean)	Τυπική απόκλιση (Standard Deviation)
TOTAL_CONSUMPTION	0	6329	1156,490	725,863
URBAN	0	1	0,885	0,319
RURAL	0	1	0,103	0,304
MOUNTAINOUS	0	1	0,010	0,098
PROPRIETARY	0	1	0,938	0,242
HOUSE	0	1	0,815	0,389
MAISONETTE	0	1	0,010	0,098
POOL	0	1	0,039	0,194
JACUZZI_SPA	0	1	0,027	0,163
INSULATION	0	1	0,282	0,450
HOT_AIR_CONDITIONER	0	1	0,198	0,399
STORAGE_HEATERS	0	1	0,066	0,249
CENTRAL_HEATING_PETROLEUM	0	1	0,554	0,498
RADIATOR	0	1	0,099	0,299
HALOGEN_HEATERS	0	1	0,121	0,326
FLOOR_HEATING	0	1	0,037	0,189
FIREPLACE	0	1	0,138	0,345
GAS_HEATERS	0	1	0,189	0,392
NONE_HEATING	0	1	0,027	0,163
COLD_AIR_CONDITIONER	0	1	0,946	0,227
FAN	0	1	0,444	0,497
NONE_COOLING	0	1	0,008	0,088
SQUARE_METERS	0	651	188,411	79,026
TOTAL_NUMBERS	0	20	6,409	2,551
ADULTS	0	10	2,488	1,094
UNDERAGE	0	6	1,235	1,174

Πίνακας 2.9 : Μικρότερη(minimum) τιμή, μεγαλύτερη (maximum) τιμή, μέση τιμή (mean) και τυπική απόκλιση(standard deviation) για το κάθε χαρακτηριστικό στο δείγμα μας για την περίοδο Σεπτεμβρίου – Οκτωβρίου 2012.

2.4.7 Διμηνία Νοέμβρη – Δεκέμβρη

Στην παράγραφο αυτή φαίνεται το ιστόγραμμα της ηλεκτρικής κατανάλωσης των οικιών κατά την περίοδο Νοεμβρίου και Δεκεμβρίου 2012 στο δείγμα μας στο σχήμα 2.10. Επίσης ο Πίνακας 2.10 παρουσιάζει αναλυτικά τα δεδομένα των χαρακτηριστικών των δεδομένων μας, τα οποία καλούνται οι αλγόριθμοι μηχανικής μάθησης να λάβουν υπόψη. Πιο συγκεκριμένα ο Πίνακας αυτός παρουσιάζει για κάθε χαρακτηριστικό των οικιών και της κατανάλωσης τους τον μικρότερο, μεγαλύτερο, μέσο (mean) και τυπική απόκλιση (standard deviation), ούτως ώστε να μπορούμε να κατανοήσουμε τα αποτελέσματα των αλγορίθμων μας και να εξάγουμε ορθά συμπεράσματα.



Σχήμα 2.10 : Ιστόγραμμα ηλεκτρικής κατανάλωσης των οικιών κατά τη διμηνία Νοεμβρίου - Δεκεμβρίου 2012 του δείγματός μας.

Όνομα	Ελάχιστο (Minimum)	Μέγιστο (Maximum)	Μέσος (Mean)	Τυπική απόκλιση (Standard Deviation)
TOTAL_CONSUMPTION	0	2381	704,079	360,200
URBAN	0	1	0,886	0,318
RURAL	0	1	0,103	0,304
MOUNTAINOUS	0	1	0,010	0,098
PROPRIETARY	0	1	0,938	0,241
HOUSE	0	1	0,814	0,389
MAISONETTE	0	1	0,010	0,098
POOL	0	1	0,039	0,193
JACUZZI_SPA	0	1	0,027	0,162
INSULATION	0	1	0,280	0,450
HOT_AIR_CONDITIONER	0	1	0,199	0,400
STORAGE_HEATERS	0	1	0,068	0,251
CENTRAL_HEATING_ PETROLEUM	0	1	0,551	0,498
RADIATOR	0	1	0,099	0,298
HALOGEN_HEATERS	0	1	0,120	0,325
FLOOR_HEATING	0	1	0,039	0,193
FIREPLACE	0	1	0,139	0,347
GAS_HEATERS	0	1	0,188	0,391
NONE_HEATING	0	1	0,027	0,162
COLD_AIR_CONDITIONER	0	1	0,946	0,227
FAN	0	1	0,445	0,497
NONE_COOLING	0	1	0,008	0,088
SQUARE_METERS	0	651	188,302	78,976
TOTAL_NUMBERS	0	20	6,400	2,556
ADULTS	0	10	2,487	1,092
UNDERAGE	0	6	1,236	1,172

Πίνακας 2.10 : Μικρότερη(minimum) τιμή, μεγαλύτερη (maximum) τιμή, μέση τιμή (mean) και τυπική απόκλιση(standard deviation) για το κάθε χαρακτηριστικό στο δείγμα μας για την περίοδο Νοεμβρίου – Δεκεμβρίου 2012.

2.5 Προεπεξεργασία και Κανονικοποίηση Δεδομένων

Τα πιο πάνω δεδομένα προτού εισαχθούν σαν είσοδο στους αλγορίθμους μηχανικής μάθησης, έπρεπε να υπόκειντο σε επεξεργασία. Ανάλογα με τους διαφόρους αλγορίθμους, τους οποίους θα αναφέρουμε στην συνέχεια, χρησιμοποιείτο διαφορετική προεπεξεργασία των δεδομένων. Μερικές από τις κυριότερες μορφές προεπεξεργασίας οι οποίες έλαβαν χώρο ήταν οι ακόλουθες

1. Καμία προεπεξεργασία. Τα δεδομένα εισάγονταν στους αλγορίθμους ακριβώς ως έχουν. Αυτή η περίπτωση γινόταν για να υπάρχει ένα μέτρο σύγκρισης σε σχέση με τις υπόλοιπες μορφές προεπεξεργασίας
2. Κανονικοποίηση όλων των τιμών που ορίζονται στο σύνολο των πραγματικών αριθμών σε από 0 έως 1 ή -1 έως 1. Αυτό είναι εξαιρετικά χρήσιμο σε πληθώρα αλγορίθμων όπως για παράδειγμα τα νευρωνικά δίκτυα. Είναι εξαιρετικά βοηθητική μέθοδος για να μπορούν οι αλγόριθμοι μας να έχουν μία γρηγορότερη και ορθότερη απάντηση. Η μέθοδος αυτή όμως πλήττεται από ένα σημαντικότατο μειονέκτημα. Το γεγονός ότι η μέγιστη τιμή στα δεδομένα μας παίρνει την τιμή 1, αποτελεί πλήγμα, καθώς εάν στα άγνωστα δεδομένα, τα οποία θα θέσει ο χρήστης στο σύστημα η τιμή σε μία μεταβλητή είναι μεγαλύτερη από αυτήν που είχαμε εμείς στα δεδομένα εκπαίδευσης ο αλγόριθμος δεν θα μπορεί να βγάλει μία ορθή τιμή. Για αυτό υπάρχουν δύο πιθανές λύσεις. Είτε θα κανονικοποιούμε τα δεδομένα λαμβάνοντας υπόψη την zero mean normalization η οποία λαμβάνει υπόψη τον μέσο (mean) των τιμών. Η δεύτερη περίπτωση είναι να το αφήσουμε ως έχει, και να μην αφήνουμε τον χρήστη να δώσει τιμή μεγαλύτερη από αυτή που είχαμε δει στα δεδομένα εκπαίδευσης μας.
3. Επιλογή χαρακτηριστικών. «Attribute Selection». Κάποια χαρακτηριστικά στο δείγμα μας, μπορεί να μην επηρεάζουν πολύ την ηλεκτρική κατανάλωση, ή να παίρνουν ανομοιόμορφες τιμές (πχ μόνο 3 εγγραφές να έχουν τη μεταβλητή maisonette να αληθής), και έτσι όχι μόνο δεν βοηθούν τον αλγόριθμο να κατηγοριοποιήσει σωστά, αλλά προσθέτουν επιπλέων φόρτο εργασίας στα δεδομένα μας, που μπορεί εκτός από υπολογιστική ισχύ να έχει ως αποτέλεσμα και λανθασμένο αποτέλεσμα. Έτσι επιλέγονται οι μεταβλητές που φαίνονται ότι επηρεάζουν περισσότερο την ηλεκτρική

κατανάλωση, και που πράγματι είναι βοηθητικές στον αλγόριθμο ούτως ώστε να κάνει ορθότερες εκτιμήσεις.

4. Διαγραφή θορύβου. Το δείγμα μας κάποιες φορές είχε 3-4 εγγραφές με κατανάλωση τριπλάσια ή τετραπλάσια από ένα κανονικό σπίτι ή κατανάλωση που πλησιάζει το μηδέν. Αυτές οι εγγραφές είχαν θεωρηθεί ως προβληματικές («Θόρυβος»), αφού οι οικίες αυτές μπορεί να ήταν εκτός από οικίες, και ο χώρος εργασίας κάποιων, και δεν θα μπορούσαν σε καμία περίπτωση να βοηθήσουν τους αλγορίθμους μας να εκτιμούν σωστά την ηλεκτρική κατανάλωση μίας πραγματικής οικίας. Με την ίδια λογική, κάποιες εγγραφές είχαν κατανάλωση <200 κιλοβατώρες, και οι οικίες αυτές θα μπορούσαν να θεωρηθούν είτε ως εξοχικές κατοικίες, ή ως κατοικίες που χρησιμοποιούνταν σποραδικά αφού η κατανάλωση εδώ ήταν εξαιρετικά μικρή. Έτσι και αυτές οι περιπτώσεις είχαν επίσης διαγραφεί.
5. Συνδυασμός μεταβλητών. Κάποιες μεταβλητές ίσως να έχουν παρόμοιο αντίκτυπο με κάποιες άλλες, και έτσι ίσως να είναι εφικτό οι δύο μεταβλητές να ομαδοποιηθούν. Δηλαδή η μεταβλητή JACCUZZI _SPA ίσως να έχει παρόμοιο αντίκτυπο με την μεταβλητή POOL και ίσως οι δύο να πρέπει να ομαδοποιηθούν. Το ίδιο ίσως να ισχύει και με τις συσκευές θέρμανσης. Έχουν δημιουργηθεί αρκετές συγχωνεύσεις μεταβλητών, που αναφέρονται στα επόμενα κεφάλαια.
6. Διασπάσεις μεταβλητών, ετικετοποίηση μεταβλητών. Έχει ακόμη δοκιμαστεί και η περίπτωση όπου τα δεδομένα αντί να εισάγονται στους αλγορίθμους ως αριθμοί, να περνούν ως κατηγορίες με ετικέτες. Για παράδειγμα υπάρχουν δύο μεταβλητές που δηλώνουν τον τύπο του σπιτιού (αν είναι κατοικία, ή όχι –το οποίο δηλώνει διαμέρισμα- ή αν είναι μεζονέτα). Αυτός ο τύπος θα μπορούσε να γραφεί ως μια μεταβλητή η οποία έχει το πεδίο ορισμού {HOUSE,FLAT,MESSONETTE}. Έχει χρησιμοποιηθεί και αυτός ο τύπος κατά την διάρκεια των πειραμάτων μας για να μπορούμε επίσης να συγκρίνουμε. Δεν έχει προκαλέσει παρόλα αυτά κάποια καλύτερα αποτελέσματα
7. Διακριτοποίηση των δεδομένων του διανύσματος χαρακτηριστικών του σπιτιού <X>. Δηλαδή όπου αυτό είναι εφικτό, αντί να εισάγονται στους αλγορίθμους πραγματικοί αριθμοί, εισάγονται κατηγορίες αριθμών. Για παράδειγμα αντί να εισαχθεί ο αριθμός 211 σαν είσοδο στον αλγόριθμο για την μεταβλητή τετραγωνικά μέτρα σπιτιού, να εισαχθεί κατηγορία «190-220». Έτσι αντί να έχουμε φυσικούς αριθμούς έχουμε πολύ λιγότερες κατηγορίες αριθμών (~10-15). Αυτό δύναται να βοηθήσει τον αλγόριθμο να κάνει μία γρηγορότερη εκτίμηση όσο αφορά την ηλεκτρική κατανάλωση των σπιτιών. Η μέθοδος αυτή είχε δοκιμαστεί τόσο στα νευρωνικά δίκτυα, όσο και σε άλλους αλγόριθμους στο πρόβλημά μας

8. Διακριτοποίηση των δεδομένων που αφορούν την ηλεκτρική κατανάλωση μίας οικίας. Η μέθοδος αυτή είναι παρόμοια με την προηγούμενη, αλλά αυτή την φορά εφαρμόζεται σε αυτό που θα θέλαμε να μας παράγει ο αλγόριθμος μας. Έτσι τώρα θα μας επιστρέφει μία κατηγορία τιμών αντί μία πραγματική τιμή. Δηλαδή αντί να επιστρέφει τώρα «η εκτίμηση είναι 1041 κιλοβατώρες για το σπίτι με αυτά τα δεδομένα», θα επιστρέφει η εκτίμηση είναι ότι ενέργεια που θα καταναλώσει το σπίτι αυτό αναμένεται να είναι στην κατηγορία «1000-1050» κιλοβατώρες/
9. Συνδυασμός των πιο πάνω διαφορετικών μεθόδων κανονικοποίησης και προεπεξεργασίας

Κεφάλαιο 3

Πιθανοί Αλγόριθμοι και εργαλεία Επίλυσης του προβλήματος

3.1 Κατηγορίες Αλγορίθμων

Έχουν υλοποιηθεί πληθώρα αλγορίθμων με χρήση της γλώσσας προγραμματισμού Java, αλλά επίσης και έχουν ακόμη χρησιμοποιηθεί πολλοί άλλοι αλγόριθμοι χρησιμοποιώντας τα εργαλεία μηχανικής μάθησης Weka και Rapidminer.

Οι αλγόριθμοι οι οποίοι έχουν χρησιμοποιηθεί μπορούν να διαχωριστούν με βάση τον τύπο μάθησής τους σε δύο κατηγορίες

3.1.1 Επιβλεπόμενη μάθηση

Στην *επιβλεπόμενη μάθηση* ο πράκτορας παρατηρεί κάποια ζευγάρια εισόδου-εξόδου και μαθαίνει μία συνάρτηση η οποία αντιστοιχίζει από την είσοδο στην έξοδο.
Η μάθηση στους αλγορίθμους αυτούς απαιτεί ένα «δάσκαλο» ο οποίος κατά την διάρκεια της εκπαίδευσης δίνει τα σωστά αποτελέσματα στον αλγόριθμο. Δηλαδή στην δική μας περίπτωση, κατά την διάρκεια της εκπαίδευσης, όταν ο αλγόριθμος παράγει ένα αποτέλεσμα και ο «δάσκαλος» αυτός του δίνει την πραγματική κατανάλωση, ούτως ώστε να μπορεί να αλλάξει τις μεταβλητές του και τα βάρη του με τέτοιο τρόπο, ώστε να προσαρμοστεί και να επιστρέφει καλύτερες απαντήσεις στο μέλλον.
Στις περιπτώσεις που έχουμε τόσο τα δεδομένα εισόδου, όσο και τα επιθυμητά αποτελέσματα για ένα σύνολο εγγραφών στο οποίο θα εκπαιδευτούν οι αλγόριθμοι, τα οποία έχουμε στην δική μας περίπτωση, η μέθοδος αυτή προτιμείται, καθώς οι αλγόριθμοι επιστρέφουν πιο ακριβή αποτελέσματα και συνήθως εκπαιδεύονται γρηγορότερα.
Κάποιοι αλγόριθμοι και τεχνικές οι οποίοι χρησιμοποιούν επιβλεπόμενη μάθηση είναι

1. Τεχνητά νευρωνικά δίκτυα
2. Back Propagation – (αλγόριθμος οπισθόδρομης μετάδοσης σφάλματος)
3. Αλγόριθμος κοντινότερων γειτόνων
4. Παλινδρόμηση με Γκαουσιανές Διαδικασίες
5. Δέντρα αποφάσεων

3.1.2 Μη Επιβλεπόμενη Μάθηση

Στη μηχανική μάθηση, η *μη επιβλεπόμενη μάθηση* αναφέρεται στο πρόβλημα της προσπάθειας να βρεθεί μία δομή σε δεδομένα χωρίς ετικέτα (unlabeled data). Εφόσον τα παραδείγματα τα οποία δίνονται στον αλγόριθμο είναι χωρίς ετικέτα, δεν υπάρχει κάποιο σήμα το οποίο δηλώνει κάποιο σφάλμα ή αμοιβή, για να μπορούμε να αξιολογήσουμε την πιθανή λύση.

Ένα παράδειγμα για να μπορούμε να κατανοήσουμε τι είναι δηλαδή η μη επιβλεπόμενη μάθηση είναι το παρακάτω. Έστω ότι έχουμε ένα σύνολο χειρόγραφων γραμμάτων. Ένας αλγόριθμος μη επιβλεπόμενης μάθησης θα πάρει το σύνολο των γραμμάτων αυτών (χωρίς την κλάση τους δηλ. το γράμμα του αλφαβήτου στο οποίο αντιστοιχούν). Θα προσπαθήσει στη συνέχεια να ομαδοποιήσει τα γράμματα σύμφωνα με κοινά τους χαρακτηριστικά (οι δομή η οποία είχαμε αναφέρει προηγουμένως), όπως εάν για παράδειγμα θα δει ποια από

αυτά τα γράμματα αποτελούνται μόνο από ένα κύκλο, για να σχηματίσει το γράμμα ο. Ο αλγόριθμος αυτός λειτουργεί μόνος του, παρατηρώντας σχέσεις στα δεδομένα χωρίς απολύτως καμία ανάδραση (feedback).

Κάποια παραδείγματα μη επιβλεπόμενης μάθησης είναι

1. Ομαδοποίηση (π.χ. K-μέσοι)
2. Τυφλός διαχωρισμός σήματος με χρήση τεχνικών εξαγωγής χαρακτηριστικών για μείωση διαστάσεων (π.χ. παραγοντοποίηση μη αρνητικού πίνακα)
3. Χάρτης αυτοοργάνωσης -Self-Organizing Maps , SOM- (π.χ. Χάρτης αυτοοργάνωσης Kohonen)

3.1.3 Ενισχυτική μάθηση

Στην ενισχυτική μάθηση σε αντίθεση με την επιβλεπόμενη μάθηση, δεν έχουμε κάποιο δάσκαλο όπως στην ενισχυτική. Στην περίπτωση αυτή, έχουμε κάποιο κριτή. Για παράδειγμα για ένα μεταφραστή κειμένου όπως το Google Translate, εάν υπάρχουν αστέρια για να κρίνουμε μία μετάφραση, και κρίνουμε την μετάφραση ως όχι ικανοποιητική, τότε ο αλγόριθμος παίρνει την αμοιβή ή τιμωρία την οποία του δίνουμε με το εργαλείο κριτικής των αστεριών, και προσπαθεί να μάθει να παράγει καλύτερες μεταφράσεις μέσω των κριτικών αυτών. Προσέξτε ότι δεν του έχουμε αναφέρει ποια είναι η σωστή μετάφραση, αλλά μονάχα μία αξιολόγησή της.
Πιο συγκεκριμένα στην ενισχυτική μάθηση ο πράκτορας/αλγόριθμος μαθαίνει από μία ακολουθία από ενισχύσεις, οι οποίες μπορεί να είναι αμοιβές ή τιμωρίες. Η πρακτική αυτή είναι εξαιρετικά ρεαλιστική, καθώς χρησιμοποιείται ευρέως για να μάθει κάποιος άνθρωπος. Για παράδειγμα, εάν ένα μικρό παιδί, κάνει τα μαθήματά του, τότε θα πάμε μαζί του στο πάρκο για να παίξει. Αυτό είναι μία μορφή αμοιβής, και θα συνεχίζει να κάνει τη σωστή πράξη, το να διαβάζει τα μαθήματά του, για να παίρνει την αμοιβή αυτή. Στην αντίθετη περίπτωση, πιθανώς να τιμωρηθεί, παραμένοντας στο δωμάτιό του για το υπόλοιπο της ημέρας. Έτσι θα κατανοήσει ότι έχει κάνει κάτι κακό, και θα μάθει μέσω αυτού, για να μπορεί να παίρνει καλύτερες αμοιβές.

3.2 Εργαλεία τα οποία χρησιμοποιήθηκαν

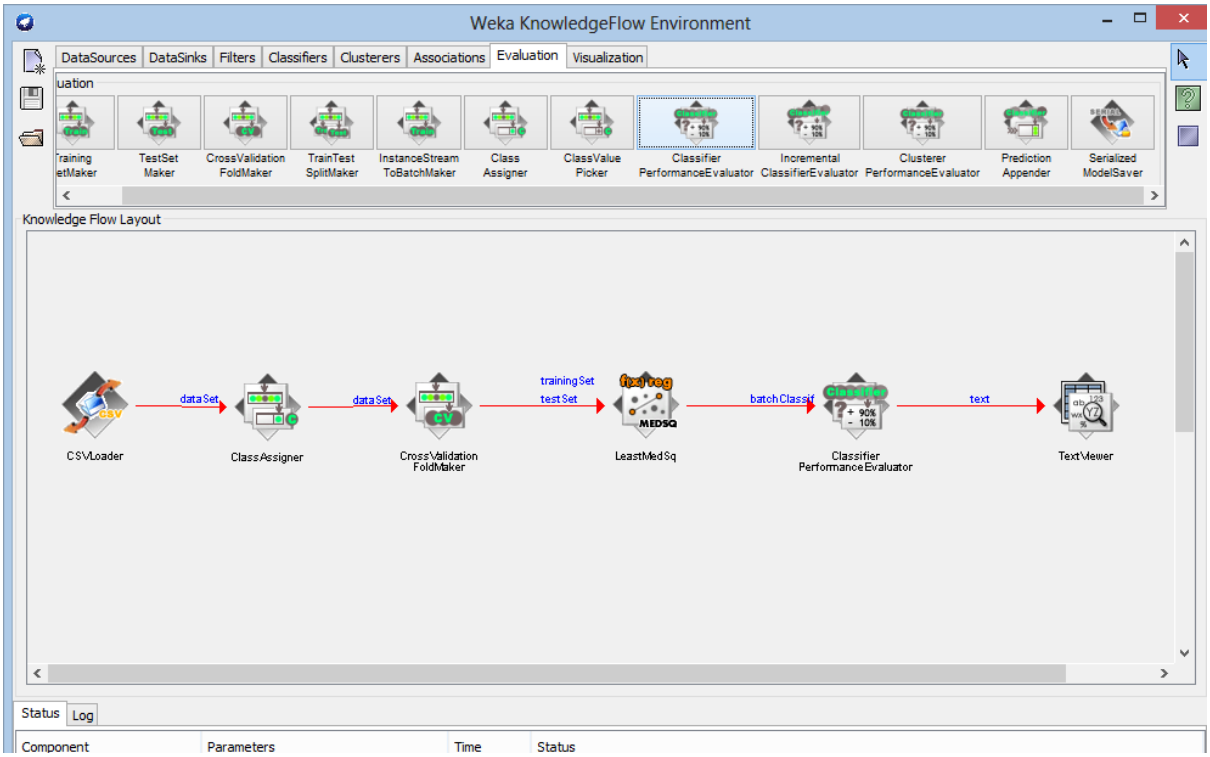
Μέσω αυτής της διπλωματικής εργασίας, έχω αποκομίσει πολλά, γιατί έχω αναπτύξει τόσο τις δικές μου υλοποιήσεις των αλγορίθμων στην γλώσσα προγραμματισμού Java, όσο επίσης

έχω μάθει να χρησιμοποιώ πολύτιμα εργαλεία, τα οποία χρησιμοποιούνται από επαγγελματίες επιστήμονες μηχανικής μάθησης σε όλο τον κόσμο.

Πιο συγκεκριμένα, έχω υλοποιήσει σε Java αλγορίθμους νευρωνικών δικτύων. Οι αλγόριθμοι οι οποίοι είχα δημιουργήσει, ήταν νευρωνικά δίκτυα τύπου MLP τόσο για παραγωγή μίας τιμής (συνάρτηση), όσο και για παραγωγή μίας κατηγορίας (κατηγοριοποίηση). Επίσης έχω δημιουργήσει νευρωνικά δίκτυα τύπου RBF, καθώς επίσης και τον χάρτη αυτό-οργάνωσης Κοχόνεν (Kohonen SOM).

Επίσης είχε χρησιμοποιηθεί το εργαλείο μηχανικής μάθησης WEKA[31]. Το εργαλείο αυτό, προσφέρει πολύτιμες λειτουργίες, και υλοποιήσεις αλγορίθμων, βελτιστοποιημένων, για αποφυγή της σπατάλης χρόνου για υλοποίηση των ήδη υπαρχουσών αλγορίθμων. Το εργαλείο αυτό είναι αυτοματοποιημένο, και περιέχει ένα μεγάλο αριθμό από αλγορίθμους οι οποίοι χρησιμοποιούνται ευρέως. Είναι ένα από τα δημοφιλέστερα εργαλεία, καθώς χρησιμοποιείται σε πολλά ακαδημαϊκά ιδρύματα και οργανισμούς. Το γεγονός ότι το έργο αυτό είναι ανοικτού κώδικα, το καθιστά ένα από τα πιο αξιόπιστα και ολοκληρωμένα, καθώς οποιοσδήποτε ο οποίος έχει κάποια γνώση, μπορεί να προσθέσει, ή να αλλάξει προς το καλύτερο τον κώδικα. Παρέχει γραφική αναπαράσταση των δεδομένων, καθώς επίσης παρέχει δυνατότητες για γρήγορη και εύκολη προεπεξεργασία των δεδομένων.

Έχει ακόμη χρησιμοποιηθεί το εργαλείο Rapidminer [32], το οποίο είναι ακόμη ένα βοηθητικό εργαλείο, καθώς περιέχει μία μεγάλη γκάμα αλγορίθμων, αλλά επίσης διαθέτει γραφική διεπαφή η οποία δείχνει την ροή με την οποία οι διάφορες φάσεις στους αλγορίθμους πρέπει να συνδέονται. Έτσι, κάποιος μπορεί να το χρησιμοποιήσει πολύ εύκολα όπως φαίνεται στην εικόνα πιο κάτω.



Εικόνα 3.1 – Παράδειγμα εκτέλεσης προγράμματος Rapidminer.

Επίσης είχα χρησιμοποιήσει το εργαλείο Octave [33], στο οποίο είχα υλοποιήσει τον αλγόριθμο γραμμικής παλινδρόμησης, καθώς επίσης είχα μία μικρή επαφή με το εργαλείο στατιστικής R [34] από το οποίο όμως δεν είχα κάποια καλύτερα αποτελέσματα.

3.3 Επεξήγηση όρων μηχανικής μάθησης

Εκπαίδευση – Training: Είναι η φάση κατά την οποία ο αλγόριθμος μας διαβάζει *ένα υποσύνολο* των δεδομένων που έχουμε, το οποίο ονομάζεται *σύνολο εκπαίδευσης (training set)* και προσπαθεί να μάθει από αυτό αναπροσαρμόζοντας τις διάφορες μεταβλητές του σύμφωνα με τις μεθόδους που καθορίζονται στον αλγόριθμο. Κατά την εκμάθηση ενός αλγορίθμου εκτελείται πολλές φορές η φάση της εκπαίδευσης.

Έλεγχος – Testing: Είναι η φάση κατά την οποία ο αλγόριθμος μας διαβάζει *ένα υποσύνολο* των δεδομένων που έχουμε, το οποίο ονομάζεται *σύνολο ελέγχου (testing set).* Στην φάση αυτή, ο αλγόριθμος τρέχει έχοντας σαν είσοδο άγνωστα δεδομένα (το σύνολο ελέγχου). Δεν προσπαθεί να μάθει από αυτά, απλά παράγει ένα αποτέλεσμα για κάθε μία από τις εγγραφές του συνόλου. Αυτό για να είμαστε σε θέση να αξιολογούμε την πρόοδο του αλγορίθμου μας σε άγνωστα δεδομένα, και την ικανότητά του να γενικεύει. Το σφάλμα το οποίο εμείς χρειάζεται να παίρνουμε κατά την εκτέλεση των πειραμάτων μας είναι το σφάλμα της φάσης ελέγχου, και όχι της φάσης εκπαίδευσης, καθώς μας ενδιαφέρει η επιτυχία του αλγορίθμου

στα άγνωστα δεδομένα, και όχι στα δεδομένα τα οποία έχει ξαναδεί, και προσπαθήσει να τα μάθει. Κατά την εκμάθηση ενός αλγορίθμου εκτελείται πολλές φορές η φάση του ελέγχου.

Εποχή ή Επανάληψη: Είναι μία διαδοχική εκτέλεση της φάσης εκπαίδευσης η οποία ακολουθείται από μία εκτέλεση της φάσης ελέγχου. Ένας αλγόριθμος για να μπορεί να μάθει συνήθως απαιτούνται περισσότερες από μία εποχές. Αυτό σημαίνει ότι θα τρέχει για τα δεδομένα εκπαίδευσης, να αναπροσαρμόζεται, για να μάθει από αυτά, μία φάση ελέγχου στην συνέχεια, μετά ξανά τα δεδομένα εκπαίδευσης για να μάθει καλύτερα, και ούτω καθεξής.

Γενίκευση: Αυτό που συνήθως απαιτείται από τους αλγορίθμους μηχανικής μάθησης είναι η ικανότητά τους να γενικεύουν. Αυτό σημαίνει ότι δεν μαθαίνουν απέξω τα δεδομένα του συνόλου εκπαίδευσης (παπαγαλία) αλλά μαθαίνουν απλά τις σχέσεις τους με την επιθυμητή έξοδό τους, ή τις σχέσεις μεταξύ τους. Αυτό που συνήθως θέλουμε δεν είναι να μπορεί ο αλγόριθμος να παράγει ξανά τις τιμές τις οποίες έχει δει στο σύνολο εκπαίδευσης, αλλά να μπορεί να εφαρμόσει τις σχέσεις τις οποίες χρειαζόμαστε σε άγνωστα δεδομένα.

Εξειδίκευση: Είναι η εκμάθηση των δεδομένων τα οποία υπάρχουν στο σύνολο εκπαίδευσης ως έχουν, Χάνεται δηλαδή η ικανότητα να μαθαίνει τις σχέσεις μεταξύ των δεδομένων, τις οποίες χρειάζεται να μάθει (γενικότητα).

Κ-πτυχη διασταυρωμένη επικύρωση - K-folds Cross Validation: Η διασταυρωμένη επικύρωση, είναι μέθοδος, η οποία χρησιμοποιείται για να μπορούμε να αξιολογούμε τον αλγόριθμο μας και τα αποτελέσματά του στα δεδομένα μας. Πιο συγκεκριμένα αποτελεί μία τεχνική επικύρωσης μοντέλου, για να αξιολογήσει πως τα αποτελέσματα μίας στατιστικής ανάλυσης θα γενικεύσουν σε ένα ανεξάρτητο σύνολο δεδομένων. Χρησιμοποιείται σε περιπτώσεις όπου ο στόχος είναι εκτίμηση, όπως και στο δικό μας πρόβλημα, και κάποιος χρειάζεται να εκτιμήσει την ακρίβεια ενός μοντέλου εκτίμησης στην πρακτική του εφαρμογή. Μία φάση της διασταυρωμένης επικύρωσης συμπεριλαμβάνει τεμαχισμό του δείγματος δεδομένων, ενός συνόλου δεδομένων δηλαδή (dataset) σε ανεξάρτητα – συμπληρωματικά υποσύνολα. Στη συνέχεια γίνεται η ανάλυση στο ένα υποσύνολο, το οποίο το ονομάζουμε υποσύνολο εκπαίδευσης, και αξιολογούμε την ανάλυση στα υπόλοιπα υποσύνολα, τα οποία τα ονομάζουμε υποσύνολα ελέγχου. Για αύξηση της μεταβλητότητας, γίνονται πολλαπλές εκτελέσεις διασταυρωμένων επικυρώσεων, με τα διαφορετικά υποσύνολα δεδομένων, και τα αποτελέσματα αξιολόγησης είναι ο μέσος όρος της αξιολόγησης των διαφόρων φάσεων.

Στην Κ-πτυχη διασταυρωμένη-επικύρωση, το αρχικό σύνολο δεδομένων, τυχαία διαχωρίζεται σε Κ ίσου μεγέθους υποσύνολα δεδομένων. Γίνονται συνολικά Κ επαναλήψεις,(οι πτυχές), και σε κάθε μια επιλέγεται ένα εκ των Κ υποσυνόλων για έλεγχο, ενώ τα υπόλοιπα Κ-1 για εκπαίδευση. Κάθε ένα εκ των Κ υποσυνόλων χρησιμοποιείται ακριβώς μία φορά ως σύνολο ελέγχου. Πλεονέκτημα της μεθόδου αυτής, έναντι της επαναλαμβανόμενης κατασκευής υποσυνόλων, είναι ότι όλες οι εγγραφές χρησιμοποιούνται τόσο για έλεγχο όσο και για εκπαίδευση, και για κάθε φάση ελέγχου χρησιμοποιούνται ακριβώς μία φορά. Στην δική μας περίπτωση, χρησιμοποιήσαμε πολλές φορές την μέθοδο των Κ-πτυχων διασταυρωμένων επικυρώσεων, με τιμή $K=10$.

Γραμμικά διαχωρίσιμα (linearly separable): Στη μηχανική μάθηση, κάθε εγγραφή έχει μέγεθος n και έτσι μπορεί να αναπαρασταθεί στον n-διάστατο χώρο ως ένα σημείο του. Δύο σημαία είναι γραμμικά διαχωρίσιμα σε ένα n-διάστατο χώρο, εάν μπορούν να διαχωριστούν από μία υπερεπιφάνεια. Υπάρχουν δεδομένα τα οποία δεν είναι γραμμικά διαχωρίσιμα, και έτσι σε ένα πρόβλημα κατηγοριοποίησης, δεν θα μπορούμε να διαχωρίσουμε τα δεδομένα μας στις διάφορες κατηγορίες.

Αναγωγή υψηλότερων διαστάσεων: Μπορούμε να χρησιμοποιήσουμε μη γραμμικές μεθόδους π.χ. Gaussian ούτως ώστε να ανυψώσουμε το σημείο των n-διαστάσεων σε υψηλότερες, για να μπορεί να διαχωριστεί από τους αλγορίθμους μας.

Παλινδρόμηση: Μία τεχνική στατιστικής, η οποία χρησιμοποιείται για να εκτιμάται η σχέση μεταξύ διαφόρων μεταβλητών. Δίνεται έμφαση στην σχέση μεταξύ μίας εξαρτημένης μεταβλητής, και μίας ή περισσότερων ανεξάρτητων μεταβλητών. Έτσι προσπαθούμε δοθεισών των ανεξάρτητων μεταβλητών να βρούμε μία εκτίμηση για την εξαρτημένη μεταβλητή. Ο στόχος τον οποίο θέλουμε να εκτιμήσουμε είναι μία συνάρτηση των ανεξάρτητων μεταβλητών, η οποία λέγεται συνάρτηση παλινδρόμησης.

Συνάρτηση(function): Δεδομένου ενός συνόλου μεταβλητών εισόδου, παράγει μία μεταβλητή εξόδου (πχ την ηλεκτρική κατανάλωση).

Κατηγοριοποίηση(classification): Δεδομένου ενός συνόλου εγγραφών, και ενός συνόλου κατηγοριών (labels), είναι η διαδικασία ανάθεσης μίας κατηγορίας σε μία εγγραφή. Για παράδειγμα εάν έχουμε ένα σύνολο από χειρόγραφους χαρακτήρες, θα επιθυμούσαμε να θέσουμε την ετικέτα «α» σε όλους τους χαρακτήρες οι οποίοι είναι «α» στο σύνολο μας.

Ομαδοποίηση(clustering): Είναι η διαδικασία της τοποθέτησης ενός συνόλου δεδομένων σε ομάδες, στις οποίες τα δεδομένα που τις αποτελούν έχουν κάποια κοινά χαρακτηριστικά.

3.4 Περιγραφές Αλγορίθμων που χρησιμοποιήθηκαν

Στην διπλωματική αυτή εργασία, έχουν χρησιμοποιηθεί αρκετοί αλγόριθμοι μηχανικής μάθησης. Αυτό γιατί θέλαμε η ερευνά μας, να έχει την απαραίτητη τεκμηρίωση, και να έχουμε όσο πιο ορθά αποτελέσματα είναι δυνατόν. Τον κάθε αλγόριθμο τον εκπαιδεύαμε με διαφορετικές παραμέτρους κάθε φορά, αλλά και τα δεδομένα τύγχαναν διαφορετικής προεπεξεργασίας, ούτως ώστε να μελετήσουμε κάθε ενδεχόμενο. Πιο συγκεκριμένα έχουν χρησιμοποιηθεί οι αλγόριθμοι: Νευρωνικά Δίκτυα και πιο συγκεκριμένα η πιο συχνή μορφή τους, που είναι το Πολυεπίπεδο perceptron με χρήση του αλγορίθμου μάθησης οπισθοδρομικής μετάδοσης σφάλματος (back propagation), γκαουσιανές διαδικασίες, ισοτονική παλινδρόμηση, γραμμική παλινδρόμηση, βηματική παλινδρόμηση, δίκτυο RBF, απλή γραμμική παλινδρόμηση, SMO παλινδρόμηση και ταξινομητής PLS. Στο υποκεφάλαιο αυτό, θα δούμε επιγραμματικά τους αλγορίθμους αυτούς.

3.4.1 Νευρωνικά Δίκτυα – Πολυεπίπεδο Perceptron

Αγγλικός όρος: Artificial Neural Network – Multilayer Perceptron

Νευρωνικά Δίκτυα (Artificial Neural Networks)

Ένα τεχνητό νευρωνικό δίκτυο, είναι ένα μαθηματικό μοντέλο, το οποίο έχει επηρεαστεί από τα βιολογικά νευρωνικά δίκτυα, τα οποία υπάρχουν στον εγκέφαλό μας. Ένα νευρωνικό δίκτυο, αποτελείται από συνδεδεμένους κόμβους τεχνητών νευρώνων, οι οποίοι αποτελούν συναρτήσεις. Δέχονται εισόδους από τους κόμβους που είναι συνδεδεμένοι με αυτούς, συναθροίζουν τις εισόδους επί το βάρος των ακμών που δέχονται και έτσι παράγουν μία τιμή, την οποία την στέλλουν και οι ίδιοι με την σειρά τους στους κόμβους που είναι συνδεδεμένοι. Οι συνδέσεις είναι κατευθυνόμενες. Δεν θα αναφερθούν περεταίρω λεπτομέρειες για αυτά, μιας και αναλύεται το Πολυεπίπεδο Perceptron πιο κάτω, και το οποίο έχει χρησιμοποιηθεί.

Multilayer Perceptrons

Ένα Πολυεπίπεδο Perceptron (MLP) είναι ένα εμπρόσθιο μοντέλο νευρωνικού δικτύου, το οποίο συνδυάζει ένα σύνολο δεδομένων εισόδου, σε ένα σύνολο κατάλληλων εξόδων.

Ένα MLP είναι μία παραλλαγή του κανονικού γραμμικού (linear) perceptron και μπορεί να διαχωρίσει δεδομένα τα οποία δεν είναι γραμμικά διαχωρίσιμα, ανάγοντας τα σε υψηλότερες διαστάσεις.

Ένα MLP αποτελείται από πολλαπλά επίπεδα κόμβων σε ένα κατευθυνόμενο γράφο, και κάθε επίπεδο είναι πλήρως συνδεδεμένο με το επόμενό του. Εκτός από τους κόμβους εισόδου, κάθε κόμβος είναι ένας νευρώνας με μία μη γραμμική συνάρτηση ενεργοποίησης. Ένα MLP χρησιμοποιεί τον αλγόριθμο ανάστροφης μετάδοσης (backpropagation) ο οποίος είναι ένας αλγόριθμος επιβλεπόμενης μάθησης για να εκπαιδευτεί το δίκτυο.

Για να βρούμε την έξοδο **ενός νευρώνα** πρέπει να υπολογίσουμε πρώτα το weighted sum των ακμών που εισέρχονται στον νευρώνα.

Αυτό μας δίνεται από τον τύπο:

$$Weighted\ Sum = \sum_{\iota=0}^{n-1} W_i * X_i$$

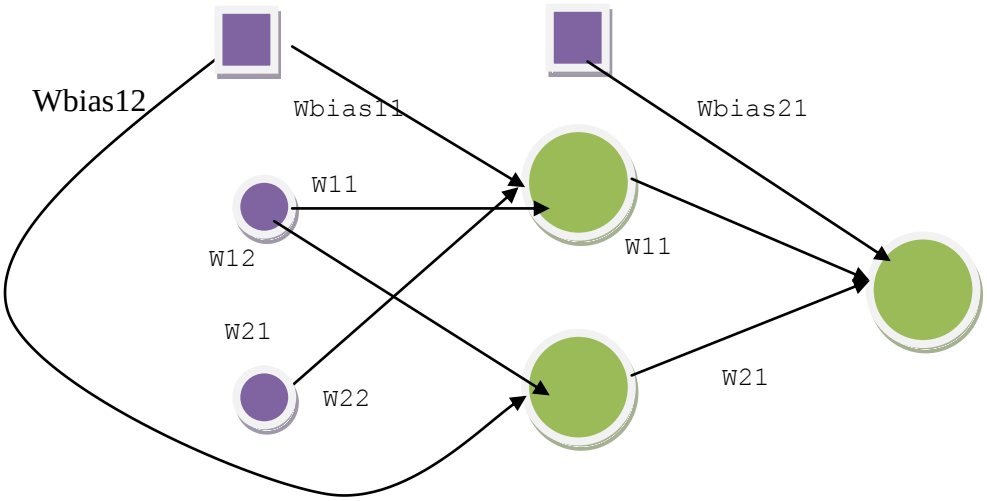
Όπου i η ακμή που τροφοδοτεί τον νευρώνα, wi το βάρος της ακμής αυτής σε σχέση με τον νευρώνα και xi η τιμή εξόδου του νευρώνα που συνδέεται με την ακμή αυτή με τον νευρώνα που εξετάζουμε ή η είσοδος, που συνδέεται με την ακμή αυτή.
Και η έξοδος του νευρώνα αυτού δίνεται από τον τύπο της σιγμοειδούς (σε διαφορετικές περιπτώσεις η έξοδος του νευρώνα δίνεται από την συνάρτηση σκαλί, η οποία όμως έχει λιγότερα πλεονεκτήματα)

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

Ως x θα παίρνει η συνάρτηση το weighted sum που προσπίπτει στον νευρώνα και έτσι έχουμε το Y το οποίο είναι η έξοδος του νευρώνα

$$\gamma = \frac{1}{1 + e^{-a \sum_{i=0}^{n-1} W_{i*} X_i}}$$

Σε κάθε επανάληψη, το αποτέλεσμα του αλγορίθμου, συγκρίνεται με το επιθυμητό αποτέλεσμα (επειδή έχουμε επιβλεπόμενη μάθηση) και γίνονται μεταβολές στα βάρη των ακμών μεταξύ των νευρώνων από το τέλος προς την αρχή, και για αυτό ο αλγόριθμος λέγεται και back-propagation.



Σχήμα 1.1: Παράδειγμα MLP

Περιληπτικά, είναι ένας ταξινομητής, ο οποίος χρησιμοποιεί τον αλγόριθμο ανάστροφης μετάδοσης σφάλματος για να κατηγοριοποιήσει τα δεδομένα. Εμείς έχουμε χρησιμοποιήσει τις σιγμοειδής συναρτήσεις για συναρτήσεις των κόμβων. Μπορούμε να θέσουμε όρους της ορμής (momentum) ο οποίος είναι βοηθητικός για να αποφεύγει τοπικά ελάχιστα, ή decay, δηλαδή να μειώνεται ο ρυθμός μάθησης με την πάροδο των εποχών. Μπορούμε ακόμη να καθορίσουμε τον αριθμό των κρυφών επιπέδων (hidden layers) του δικτύου, καθώς επίσης τον ρυθμό μάθησης (learning rate). Επίσης τα δεδομένα κοινωνικοποιούνται στις τιμές από -1 και 1. Αυτό έχει ως αποτέλεσμα καλύτερη απόδοση στο δίκτυο.

3.4.2 Γκαουσιανές Διαδικασίες

Αγγλικός όρος: Gaussian Processes

Μια γκαουσιανή διαδικασία στη στατιστική είναι μια στοχαστική διαδικασία, στην οποία τα επιτεύγματα αποτελούνται από τυχαίες τιμές, συσχετιζόμενες με κάθε σημείο σε ένα διάστημα του χρόνου ή του χώρου έτσι ώστε κάθε τυχαία μεταβλητή ακολουθεί τη κανονική κατανομή. Επίσης κάθε πεπερασμένη συλλογή αυτών των τυχαίων μεταβλητών έχει πολυδιάστατη κανονική κατανομή. Κάποιος μπορεί να φανταστεί την Γκαουσιανή διαδικασία, ως μία ∞-διάστατη γενίκευση της πολυδιάστατης κανονικής κατανομής.
Έχει πάρει το όνομά της από τον Gauss, επειδή είναι βασισμένη στην έννοια της κανονικής κατανομής, η οποία λέγεται και Γκαουσιανή κατανομή.

Λόγω του γεγονότος ότι κληρονομούνται ιδιότητες από τη κανονική κατανομή, αυτό κάνει τις Γκαουσιανές διαδικασίες σημαντικές. Για παράδειγμα, εάν μία τυχαία διαδικασία μοντελλοποιηθεί ως Γκαουσιανή διαδικασία, η κατανομή διαφόρων ποσοτήτων, όπως τον μέσο όρο της τιμής της διαδικασίας σε ένα εύρος χρόνου, ή το σφάλμα στον υπολογισμό του μέσου όρου, χρησιμοποιώντας τυχαίες τιμές σε ένα μικρό σύνολο φορών, που προκύπτουν μπορεί να εξηγηθεί άμεσα. [10]

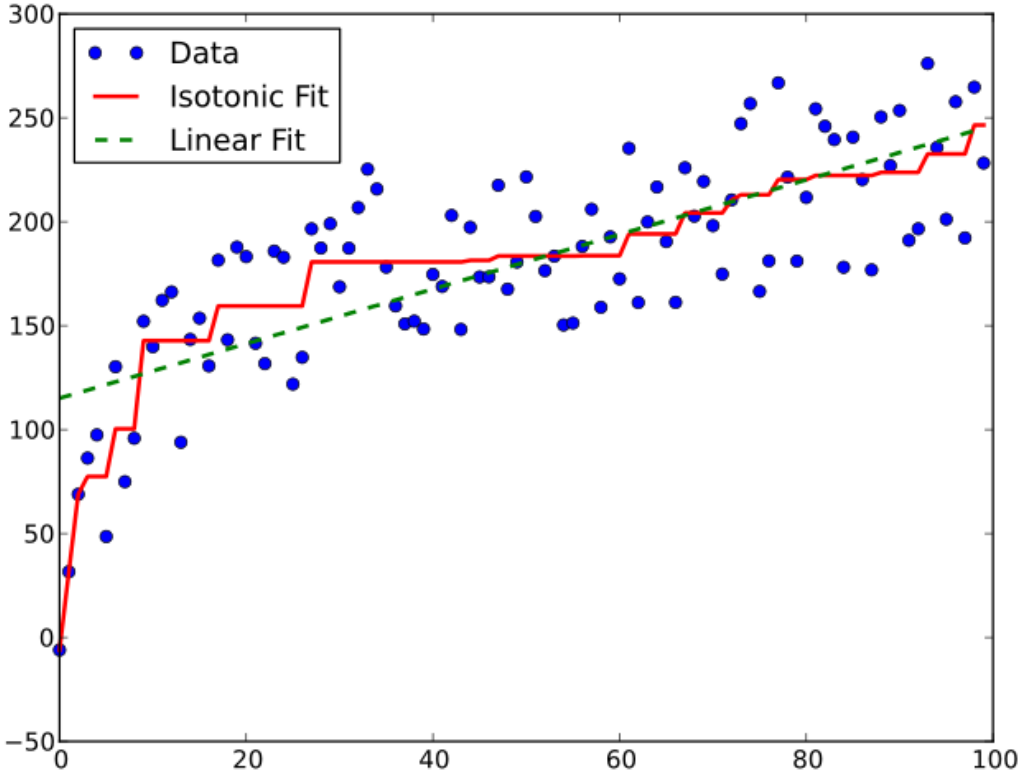
Η Γκαουσιανή διαδικασία είναι μία ισχυρή μη-παραμετρική τεχνική μηχανικής μάθησης για κατασκευασμό περιεκτικών πιθανοτικών μοντέλων πραγματικών προβλημάτων. [11]
Η Γκαουσιανή διαδικασία είναι μία στοχαστική διαδικασία $X_t, t \in T$, έτσι ώστε κάθε πεπερασμένος γραμμικός συνδυασμός δειγμάτων έχει μία κοινή Γκαουσιανή κατανομή, το οποίο σημαίνει ότι, κάθε γραμμική συνάρτηση που εφαρμόζεται στη δειγματική συνάρτηση X_t θα δώσει ένα αποτέλεσμα κατανεμημένο κανονικά.

Συμβολικά, γράφουμε ότι $X \sim GP(m, K)$ το οποίο ερμηνεύεται ως η τυχαία συνάρτηση X κατανέμεται ως μία Γκαουσιανή συνάρτηση, με μέση συνάρτηση m και συνάρτηση συνδιακύμανσης K. [12]

Είχα χρησιμοποιήσει τον αλγόριθμο των Γκαουσιανών Διαδικασιών (χωρίς ρύθμιση της υπερπαραμέτρου [25]). Είχαμε την δυνατότητα να καθορίσουμε πως τα δεδομένα θα εισάγονταν στο σύστημα, όπως τις επιλογές της κανονικοποίησης των δεδομένων, της διακριτοποίησης τους, ή κανένα από τα πιο πάνω. Επίσης σαν πυρήνα του αλγορίθμου, είχαμε επιλέξει τον πυρήνα RBF.

3.4.3 Ισοτονική Παλινδρόμηση

Αγγλικός όρος: Isotonic Regression



Σχήμα 3.1 – Γραφική παράσταση ισοτονικής και γραμμικής συνάρτησης [12]

Η ισοτονική παλινδρόμηση, πραγματεύεται με την εύρεση ενός σταθμισμένου (weighted) τεριάσματος ελαχίστων τετραγώνων (least squares) $x \in \mathbb{R}^n$ σε ένα διάνυσμα $\alpha \in \mathbb{R}^n$ με διάνυσμα βαρών $w \in \mathbb{R}^n$ που υπόκεινται σε ένα σύνολο μη αντιφατικών περιορισμών του είδους $x_i \geq x_j$.

Τέτοιου είδους περιορισμοί καθορίζουν την μερική ή ολική σειρά, και μπορούν να αναπαρασταθούν ως ένας κατευθυνόμενος γράφος $G = (N, E)$ όπου το N είναι το σύνολο των μεταβλητών που εμπλέκονται, και E είναι το σύνολο των ζευγαριών (I,j) για κάθε περιορισμό $x_i \geq x_j$ .. Έτσι το πρόβλημα της ισοτονικής παλινδρόμησης αντιστοιχεί στο παρακάτω τετραγωνικό πρόγραμμα [14]

$$min \sum_{i=1}^{n} w_i (x_i - a_i)^2 \quad \text{όπου } x_i \geq x_j \; \forall (i,j) \in E$$

Ο αλγόριθμός μας, διαλέγει το χαρακτηριστικό με το οποίο τα αποτελέσματα έχουν το χαμηλότερο τετραγωνικό σφάλμα.

3.4.4 Γραμμική Παλινδρόμηση

Αγγλικός όρος: Linear Regression

Η γραμμική παλινδρόμηση είναι μια πρακτική, για να μοντελοποιήσουμε την σχέση μεταξύ μίας μονοδιάστατης εξαρτημένης μεταβλητής y και μίας ή περισσότερων ανεξάρτητων μεταβλητών οι οποίες συμβολίζονται με X, Στην περίπτωση μίας ανεξάρτητης μεταβλητής, ονομάζεται απλή γραμμική παλινδρόμηση. Για περισσότερες των μία μεταβλητών, ονομάζεται πολλαπλή γραμμική παλινδρόμηση.

Δεδομένου ενός συνόλου $\{y_i, x_{i1}, x_{i2}, \dots, x_{ip}\}$ $\text{όπου } i = 1 \rightarrow n$ για n στατιστικές μονάδες, ένα μοντέλο γραμμικής παλινδρόμησης, υποθέτει ότι η σχέση μεταξύ της εξαρτημένης μεταβλητής y_iκαι του δυανίσματος-p των παλινδρομικών x_i είναι γραμμική.
[15]

Η έκδοση της γραμμικής παλινδρόμησης την οποία εμείς τρέξαμε, χρησιμοποιεί το κριτήριο Akaike για επιλογή μοντέλου, και είναι δυνατό να χειριστεί δεδομένα με βάρη. Επίσης έχουμε την επιλογή να διαλέξουμε ένα υποσύνολο χαρακτηριστικών και να μην τα χρησιμοποιήσουμε όλα. Οι διαθέσιμες επιλογές που έχουμε είναι : Να μην υπάρχει επιλογή χαρακτηριστικών (No attribute selection), να έχουμε επιλογή χαρακτηριστικών χρησιμοποιώντας την μέθοδο M5 (περνούμε από όλα τα χαρακτηριστικά και αφαιρούμε αυτό με το μικρότερο τυποποιημένο συντελεστή –standardized coefficient- μέχρι να μην παρακολουθηθεί καμία βελτίωση στην εκτίμηση του σφάλματος που δίνεται από το κριτήριο πληροφοριών Akaike), και άπληστη επιλογή χρησιμοποιώντας την μετρική πληροφορίας Akaike.

Επίσης μπορούμε να επιλέξουμε εάν θα διαγράψουμε τα συνευθειακά χαρακτηριστικά, καθώς επίσης να διαλέξουμε εμείς την τιμή της κορυφογραμμής.

3.4.5 Βηματική Παλινδρόμηση

Αγγλικός όρος: Pace Regression

Η βηματική παλινδρόμηση [16], κάτω από συνθήκες κανονικότητας, είναι αποδεδειγμένη να είναι βέλτιστη όταν ο αριθμός των συντελεστών (coefficients) τείνει στο άπειρο. Αποτελείται από ένα σύνολο εκτιμητών (estimators) οι οποίοι είναι είτε γενικά βέλτιστοι, είτε βέλτιστοι κάτω από συγκεκριμένες προϋποθέσεις. [17]. Περισσότερες πληροφορίες υπάρχουν στα [23] [24].

Επίσης έχουμε την επιλογή να διαλέξουμε εκτιμητή. Θα μπορούσαμε κάλλιστα για εκτιμητή, να επιλέξουμε οποιαδήποτε από τις πιο κάτω επιλογές.

1. eb -- Empirical Bayes estimator for noraml mixture (default)
2. nested -- Optimal nested model selector for normal mixture
3. subset -- Optimal subset selector for normal mixture
4. pace2 -- PACE2 for Chi-square mixture
5. pace4 -- PACE4 for Chi-square mixture
6. pace6 -- PACE6 for Chi-square mixture
7. ols -- Ordinary least squares estimator
8. aic -- AIC estimator
9. bic -- BIC estimator
10. ric -- RIC estimator
11. olsc -- Ordinary least squares subset selector with a threshold

3.4.6 Δίκτυο RBF

Αγγλικός όρος: Radial Basis Function (RBF) Network

Το δίκτυο Radial Basis Function προέρχονται από τη θεωρία της προσεγγιστικής συνάρτησης (function approximation). Οι βασικότερες αρχές από τις οποίες διέπεται είναι οι ακόλουθες:

1. Είναι εμπρόσθια τροφοδοτούμενα δίκτυα (feed-forward networks) με δύο επίπεδα.
2. Οι κρυμμένοι κόμβοι υλοποιούν ένα σύνολο από συναρτήσεις ακτινικών βάσεων όπως τις Gaussian συναρτήσεις.
3. Η εκπαίδευση του δικτύου χωρίζεται σε δύο στάδια: πρώτα καθορίζονται τα βάρη από την είσοδο στο κρυμμένο στρώμα, και στη συνέχεια τα βάρη από το κρυμμένο επίπεδο στο επίπεδο εξόδου.

4. Η εκπαίδευση και μάθηση είναι πολύ γρήγορη.
5. Τα δίκτυα είναι πολύ καλά στην παρεμβολή.

[9][18][19][20]

Ο αλγόριθμος αυτός διέπεται από διαφορετικές παραμέτρους. Μία από αυτές είναι η ελάχιστη τυπική απόκλιση (minimum standard deviation). Επίσης υλοποίει ένα κανονικοποιημένο RBF δίκτυο. Χρησιμοποιεί τον αλγόριθμο ομαδοποίησης K-μέσων και μαθαίνει μέσω της γραμμικής παλινδρόμησης. Συμμετρικές πολυμεταβλητές Γκαουσιανές ταιριάζουν (fit) στα δεδομένα από κάθε ομάδα (cluster). Μπορούμε να καθορίσουμε τον τυχαίο σπόρο (random seed) ο οποίος θα περαστεί στον αλγόριθμο των K-Μέσων, τον αριθμό τον ομάδων που θα θέλαμε να παραχθούν από τον αλγόριθμο των K-Μέσων, καθώς και την τιμή της κορυφογραμμής για τον λογιστική ή γραμμική παλινδρόμηση.

3.4.7 Απλή Γραμμική Παλινδρόμηση

Αγγλικός όρος: Simple Linear Regression

Η απλή γραμμική παλινδρόμηση, είναι ο εκτιμητής των ελαχίστων τετραγώνων ενός μοντέλου γραμμικής παλινδρόμησης, με μόνο μία ανεξάρτητη μεταβλητή. Η απλή γραμμική παλινδρόμηση, ταιριάζει μία ευθεία γραμμή διαμέσου του συνόλου των n σημείων, με τέτοιο τρόπο, που κάνει το άθροισμα των τετραγώνων των υπολοίπων του μοντέλου, όσο πιο μικρό γίνεται. Η κλίση της ευθείας είναι ίση με τη σχέση μεταξύ των χαρακτηριστικών y και x διορθωμένο από τον λόγο των κανονικών κατανομών των μεταβλητών αυτών. [21]

Ο αλγόριθμός μας, διαλέγει το χαρακτηριστικό το οποίο έχει ως αποτέλεσμα το μικρότερο τετραγωνικό σφάλμα.

3.4.8 SMO Παλινδρόμηση

Αγγλικός όρος: Sequential Minimal Optimization Regression

Ο SMO είναι ένας αλγόριθμος ο οποίος λύνει το πρόβλημα βελτιστοποίησης κατά τη διάρκεια της εκπαίδευσης των μηχανών διανυσμάτων υποστήριξης (SVM). Ο αλγόριθμος αυτός είναι πολύ διαδεδομένος για εκπαίδευση των SVM καθώς οι υπόλοιποι αλγόριθμοι των SVM ήταν πολύ πιο αργοί και πολύπλοκοι.[22]

Ο αλγόριθμός μας υλοποιεί τις μηχανές διανυσμάτων υποστήριξης για παλινδρόμηση. Οι παράμετροι μπορεί να μαθευτούν με διάφορους αλγορίθμους. Χρησιμοποιούμε τον

αλγόριθμο RegSMOImproved ο οποίος έχει προκύψει από τον Shevade, Keerthi et al, και είναι ο προκαθορισμένος βελτιστοποιητής (Optimizer). Μπορούμε να μάθουμε περισσότερα κοιτάζοντας τα: [26] [27].

Μπορούμε να επιλέξουμε την παράμετρο πολυπλοκότητας C, τον τρόπο μετατροπής των δεδομένων (καμία, κανονικοποίηση, διακριτοποίηση), τον πυρήνα για να χρησιμοποιήσουμε, και τον αλγόριθμο μάθησης (regOptimizer).

3.4.9 Ταξινομητής PLS

Αγγλικός όρος: Partial Least Squares Classifier
Ο αλγόριθμος αυτός, αντί να βρίσκει υπερεπιφάνειες της ελάχιστης διακύμανσης μεταξύ της εξαρτημένης και ανεξάρτητης μεταβλητής, βρίσκει ένα γραμμικό μοντέλο παλινδρόμησης, κάνοντας την προβολή των μεταβλητών που προβλέψαμε και των παρατηρητέων(observable) μεταβλητών σε ένα νέο χώρο (space). Χρησιμοποιείται για να κάνουμε προβλέψεις.

3.4.10 Παλινδρόμηση Ελαχίστου Μέσου των Τετραγώνων

Αγγλικός όρος: Least Median of Squares Regression

Ο συγκεκριμένος αλγόριθμος, προσπαθεί, όπως και άλλοι, να μειώσει το άθροισμα των τετραγώνων των υπολοίπων. Η ελάχιστη τετραγωνική παλινδρόμηση, με το χαμηλότερο μέσο τετραγωνικό σφάλμα, επιλέγεται ως το τελικό μοντέλο. Μπορούμε να δούμε περισσότερες πληροφορίες, για αυτό τον καθόλου πολύπλοκο αλγόριθμο στα [28][30]. Μπορούμε ακόμη στην δική μας υλοποίησης, να θέσουμε το μέγεθος των τυχαίων δειγμάτων, τα οποία χρησιμοποιούνται για να δημιουργήσουμε τις συναρτήσεις της ελάχιστης τετραγωνικής παλινδρόμησης

3.4.11 Λογιστική Παλινδρόμηση

Αγγλικός όρος: Logistic Regression
Η πιο κάτω περιγραφή είναι η περιγραφή η οποία δίνεται από το εργαλείο WEKA
Συνοπτικά: Είναι κλάση για δημιουργία και χρήση ενός μοντέλου λογιστικής παλινδρόμησης με ένα εκτιμητή κορυφογραμμής (ridge estimator). Υπάρχουν κάποιες μικρές αλλαγές σε σχέση με τη δημοσίευση των leCessie και van Houwelingen(1992) [29]:

Εάν υπάρχουν K κλάσεις για n περιπτώσεις (instances) με m χαρακτηριστικά (attributes), η παράμετρος Πίνακα B για να υπολογιστεί θα είναι ένας πίνακας m*(k-1).

Η πιθανότητα της κλάσης j με εξαίρεση την τελευταία κλάση είναι

$$P_j(X_i) = \frac{e^{x_i B_j}}{\sum_{j=1}^{k-1} e^{x_i B_j} + 1}$$

Η τελευταία κλάση έχει πιθανότητα

$$P_{last} = 1 - \sum_{j=1}^{k-1} P_j(X_i) = \frac{1}{\sum_{j=1}^{k-1} e^{x_i B_j} + 1}$$

Η (αρνητική) πολυωνυμική λογαριθμικό-πιθανότητα είναι:

$$L = -\sum_{i=1}^{n}\left[\sum_{j=1}^{k-1} Y_{ij} * \ln P_j(X_i) + \left(1 - \sum_{j=1}^{k-1}\left(Y_{ij}\right)\right) * \ln\left(1 - \sum_{j=1}^{k-1}\left(P_j(X_i)\right)\right)\right] + ridge * (B^2)$$

Για να βρούμε τον πίνακα B για τον οποίο το L ελαχιστοποιείται, χρησιμοποιείται η μέθοδος Quasi-Newton για να ψάξουμε τις βελτιστοποιημένες τιμές για τις m*(k-1) μεταβλητές. Πριν χρησιμοποιήσουμε την διαδικασία βελτιστοποίησης, ‘στριμώχνουμε’ τα δεδομένα σε ένα διάνυσμα m*(k-1). Για περισσότερες πληροφορίες στις διαδικασίες βελτιστοποίησης, κάποιος μπορεί να ελέγξει την κλάση weka.core.Optimization.

Παρόλο που η αυθεντική Λογιστική Παλινδρόμηση, δεν χειρίζεται βάρη στα δεδομένα, η παραλλαγή του αλγορίθμου που χρησιμοποιώ, έχει αλλάξει ελαφρώς, για να μπορεί να τα χειρίζεται.

Για περισσότερες πληροφορίες μπορούμε να μελετήσουμε το [29]

Στην Weka, μπορούμε να καθορίσουμε τις μέγιστες επαναλήψεις (maxIts) καθώς και την κορυφογραμμή (ridge). Ως επι το πλείστον, δεν καθόριζα τις μέγιστες επαναλήψεις, αλλά άφηνα τον αλγόριθμο να τρέχει μέχρις ότου να συγκλίνει.

Κεφάλαιο 4

Πειράματα Χρησιμοποιώντας Νευρωνικά Δίκτυα

4.1 Τύποι Νευρωνικών Δικτύων και μορφές των δεδομένων εισόδου

Τα νευρωνικά δίκτυα, είναι αναμφισβήτητα μία μεγάλη κατηγορία αλγορίθμων οι οποίοι χρησιμοποιούνται στην μηχανική μάθηση. Μιμούνται τον ανθρώπινο εγκέφαλο, και τον τρόπο που επικοινωνούν οι νευρώνες μεταξύ τους. Δηλαδή μιμούνται το γεγονός ότι οι νευρώνες ενώνονται με άλλους νευρώνες, στους οποίους στέλνουν πληροφορίες. Οι συνδέσεις μεταξύ των νευρώνων αυτών είναι κατευθυνόμενες.

Στην διπλωματική αυτή εργασία, είχαν χρησιμοποιηθεί αρκετοί διαφορετικοί τύποι νευρωνικών δικτύων. Πιο συγκεκριμένα είχαν χρησιμοποιηθεί τα πολυεπίπεδα Perceptron (Multilayer Perceptron ή MLP) με τον αλγόριθμο εκπαίδευσης ανάστροφης μετάδοσης

σφάλματος (Backpropagation algorithm) τα οποία και περιγράφονται λεπτομερώς στο κεφάλαιο 3.

Έχουν υλοποιηθεί δύο διαφορετικοί τύποι νευρωνικών MLP δικτύων. Ο πρώτος με δεδομένο μία είσοδο, παρέχει μία τιμή ως έξοδος, ενώ ο δεύτερος με δεδομένο τα χαρακτηριστικά μίας οικίας, παρέχει μία κατηγορία για την οικία ως έξοδο. Τα δύο νευρωνικά και τα αποτελέσματα που έχουν προκύψει από την μελέτη τους αναλύονται εκτενώς στο κεφάλαιο αυτό.

4.2 Νευρωνικά Δίκτυα (BackProp) συνάρτησης

Για να είμαστε σε θέση να αναλύσουμε τα δεδομένα μας με χρήση νευρωνικού δικτύου, έχω υλοποιήσει στη γλώσσα προγραμματισμού Java, ένα νευρωνικό δίκτυο MLP εκπαιδευόμενο με Backpropagation.

Τα νευρωνικά δίκτυα, απαιτούσαν τη δημιουργία δύο αρχείων με εγγραφές, το καθένα με ένα διαφορετικό σύνολο εγγραφών. Ένα για εκπαίδευση του αλγορίθμου και ένα για έλεγχό του. Οι εγγραφές αυτές έπρεπε να ήταν τυχαία από τη βάση δεδομένων. Για να είναι κατανοητός ο τρόπος με τον οποίο πήρα τα δεδομένα, έχω παραθέσει τα ερωτήματα SQL τα οποίο είχα συγγράψει, στο B1,B2,B3 και B4, και τα οποία δημιουργούν τα δύο σύνολα τα οποία χρειάζονται οι αλγόριθμοι.

Όπως φαίνεται και από τις επερωτήσεις SQL, είχα διαμοιράσει *τυχαία* 375 εγγραφές για την εκπαίδευση ενώ τις υπόλοιπες περίπου 125 για τον έλεγχο του αλγορίθμου.

Ο αλγόριθμος νευρωνικών δικτύων τον οποίο έχω υλοποιήσει, παίρνει σαν παραμέτρους τους κόμβους εισόδου, εξόδου, και τους κόμβους για κάθε ένα από δύο κρυφά επίπεδα νευρώνων. Επίσης παίρνει σαν παράμετρο τον ρυθμό μάθησης, και την ορμή του νευρωνικού δικτύου, καθώς επίσης τα αρχεία τα οποία θα δεχθεί για εκπαίδευση καθώς και για έλεγχο. Το αρχείο των παραμέτρων βρίσκεται στο A4, και ο κώδικας του νευρωνικού δικτύου βρίσκονται στο παράρτημα A1 και A2.

Είχαν γίνει αρκετά πειράματα αντιπροσωπευτικό *δείγμα* των οποίων περιγράφω ως πιο κάτω.

Πείραμα 1: Τα δεδομένα έχουν εισαχθεί στον αλγόριθμο ως είναι, χωρίς κάποια αλλαγή στην μορφή τους, και καμία διαγραφή, ούτε χαρακτηριστικών αλλά ούτε και εγγραφών οι οποίες αποκλίνουν αδικαιολόγητα. Ο αλγόριθμος στη συνέχεια τα κανονικοποιεί στην τιμή -1 μέχρι 1. Για αρχή έχουμε ξεκινήσει με ένα σχετικά μικρό ρυθμό μάθησης που είναι 0.2 καθώς και με ρυθμό ορμής 0.2. Η αρχιτεκτονική του δικτύου έχει για αρχή 8 νευρώνες στο πρώτο επίπεδο και 6 στο δεύτερο. Επίσης πίνακα στην στήλη επαναλήψεων στους πιο κάτω πίνακες παρουσιάζεται η επανάληψη στην οποία επέρχεται η βέλτιστη τιμή στην περίπτωση αυτή. Τα χαρακτηριστικά τα οποία εισάγονται στον αλγόριθμο είναι τα 25 χαρακτηριστικά ως έχουν, ενώ απαιτείται από τον αλγόριθμο να παράγει μία τιμή, την εκτίμηση της κατανάλωσης.

Πείραμα 2: Τα δεδομένα έχουν ακριβώς όπως και πιο πάνω. Μειώσαμε όμως τον ρυθμό μάθησης από 0,2 σε 0,1.

Πείραμα 3: Με ρυθμό μάθησης 0,1 τώρα έχουμε μειώσει και την ορμή σε 0,1 για να δούμε πόσο η ορμή επηρεάζει το δίκτυο

Πείραμα 4: Τώρα έχουμε χαμηλώσει ακόμη περισσότερο τον ρυθμό μάθησης από το 0,1 στα 0,01.

Πείραμα 5: Έχοντας τον πιο πάνω ρυθμό μάθησης αυξήσαμε κατά ένα νευρώνα το πρώτο και το δεύτερο επίπεδο για να μελετήσουμε πως η αλλαγή στους νευρώνες βοηθά το δίκτυό μας να εκπαιδευτεί καλύτερα.

Σημείωση: από το πείραμα αυτό (6) συμπεριλαμβανομένου και κάτω, στα δεδομένα δημιουργήσαμε ακόμη μία τιμή. Δηλαδή αντί να έχουμε δύο χαρακτηριστικά για τον τύπο σπιτιού (HOUSE,MAISONETTE), έχουμε δημιουργήσει τρεις (HOUSE, FLAT, MAISONETTE) μέσω των πιο πάνω δύο, για να μπορούν τα νευρωνικά να είναι σε θέση να κρίνουν ευκολότερα. Το ερώτημα της βάσης (DB query) το οποίο μας δίνει τα δεδομένα τα οποία χρειαζόμαστε βρίσκεται στο Β.2

Πείραμα 6: Με ρυθμό μάθησης 0,1 τώρα έχουμε μειώσει και την ορμή σε 0,1 για να δούμε πόσο η ορμή επηρεάζει το δίκτυο, εφαρμόζοντας αυτή τη φορά δεδομένα στα οποία έχουμε απομακρύνει τις εγγραφές οι οποίες είχαν εξαιρετικά μικρή ή εξαιρετικά μεγάλη κατανάλωση αδικαιολόγητα.

Πείραμα 7: Τώρα έχουμε χαμηλώσει ακόμη περισσότερο τον ρυθμό μάθησης από το 0,1 στα 0,01, εφαρμόζοντας αυτή τη φορά δεδομένα στα οποία έχουμε απομακρύνει τις εγγραφές οι οποίες είχαν εξαιρετικά μικρή ή εξαιρετικά μεγάλη κατανάλωση αδικαιολόγητα.

Πείραμα 8: Έχοντας τον πιο πάνω ρυθμό μάθησης αυξήσαμε κατά ένα νευρώνα το πρώτο και το δεύτερο επίπεδο, εφαρμόζοντας αυτή τη φορά δεδομένα στα οποία έχουμε απομακρύνει τις εγγραφές οι οποίες είχαν εξαιρετικά μικρή ή εξαιρετικά μεγάλη κατανάλωση αδικαιολόγητα.

Πείραμα 9: Με ρυθμό μάθησης 0,1 τώρα έχουμε μειώσει και την ορμή σε 0,1 για να δούμε πόσο η ορμή επηρεάζει το δίκτυο, εφαρμόζοντας αυτή τη φορά δεδομένα στα οποία έχουμε απομακρύνει τις εγγραφές οι οποίες είχαν εξαιρετικά μικρή ή εξαιρετικά μεγάλη κατανάλωση αδικαιολόγητα. Αυτή τη φορά εισάγαμε τρεις ομάδες για κάθε ένα από τα αριθμητικά χαρακτηριστικά. Δηλαδή τώρα να παίρνει σαν είσοδο μία τιμή για κάποιο αριθμητικό χαρακτηριστικό (πχ τετραγωνικά μέτρα) παίρνει την ομάδα του (πχ μικρό , μέτριο, μεγάλο)

Πείραμα 10: Τώρα έχουμε χαμηλώσει ακόμη περισσότερο τον ρυθμό μάθησης από το 0,1 στα 0,01, εφαρμόζοντας αυτή τη φορά δεδομένα στα οποία έχουμε απομακρύνει τις εγγραφές οι οποίες είχαν εξαιρετικά μικρή ή εξαιρετικά μεγάλη κατανάλωση αδικαιολόγητα. Αυτή τη φορά εισάγαμε τρεις ομάδες για κάθε ένα από τα αριθμητικά χαρακτηριστικά. Δηλαδή τώρα να παίρνει σαν είσοδο μία τιμή για κάποιο αριθμητικό χαρακτηριστικό (πχ τετραγωνικά μέτρα) παίρνει την ομάδα του (πχ μικρό , μέτριο, μεγάλο)

Πείραμα 11: Έχοντας τον πιο πάνω ρυθμό μάθησης αυξήσαμε κατά ένα νευρώνα το πρώτο και το δεύτερο επίπεδο, εφαρμόζοντας αυτή τη φορά δεδομένα στα οποία έχουμε απομακρύνει τις εγγραφές οι οποίες είχαν εξαιρετικά μικρή ή εξαιρετικά μεγάλη κατανάλωση αδικαιολόγητα Αυτή τη φορά εισάγαμε τρεις ομάδες για κάθε ένα από τα αριθμητικά χαρακτηριστικά. Δηλαδή τώρα να παίρνει σαν είσοδο μία τιμή για κάποιο αριθμητικό χαρακτηριστικό (πχ τετραγωνικά μέτρα) παίρνει την ομάδα του (πχ μικρό , μέτριο, μεγάλο). Τώρα ο ρυθμός μάθησης είναι 0,15

Ως σφάλμα μετρούμε το μέσο απόλυτο σφάλμα στο σύνολο ελέγχου (στα άγνωστα δεδομένα).

Ο τίτλος των πιο κάτω πειραμάτων έχει ως εξής:

<Κεφάλαιο>.<Υποκεφάλαιο><Διμηνία (1-6)><Αύξων αριθμός πειράματος>

Επίσης δίνω τα ακρώνυμα τα οποία υπάρχουν στον πιο κάτω πίνακα

Ο.Π.: Όνομα πειράματος

ΔΙΜ.: Διμηνία

Δ.Α.Τ. : Διαγραφή Αποκλίνουσων τιμών

Ν.Π.Ε: Νευρώνες πρώτου επιπέδου

Ν.Δ.Ε: Νευρώνες δεύτερου επίπεδου

Ν.ΕΙ: Νευρώνες Εισόδου

Ν.ΕΞ.: Νευρώνες Εξόδου

Ρ.Μ.: Ρυθμός Μάθησης (Learning Rate)

ΟΡΜ.: Ορμή (momentum)

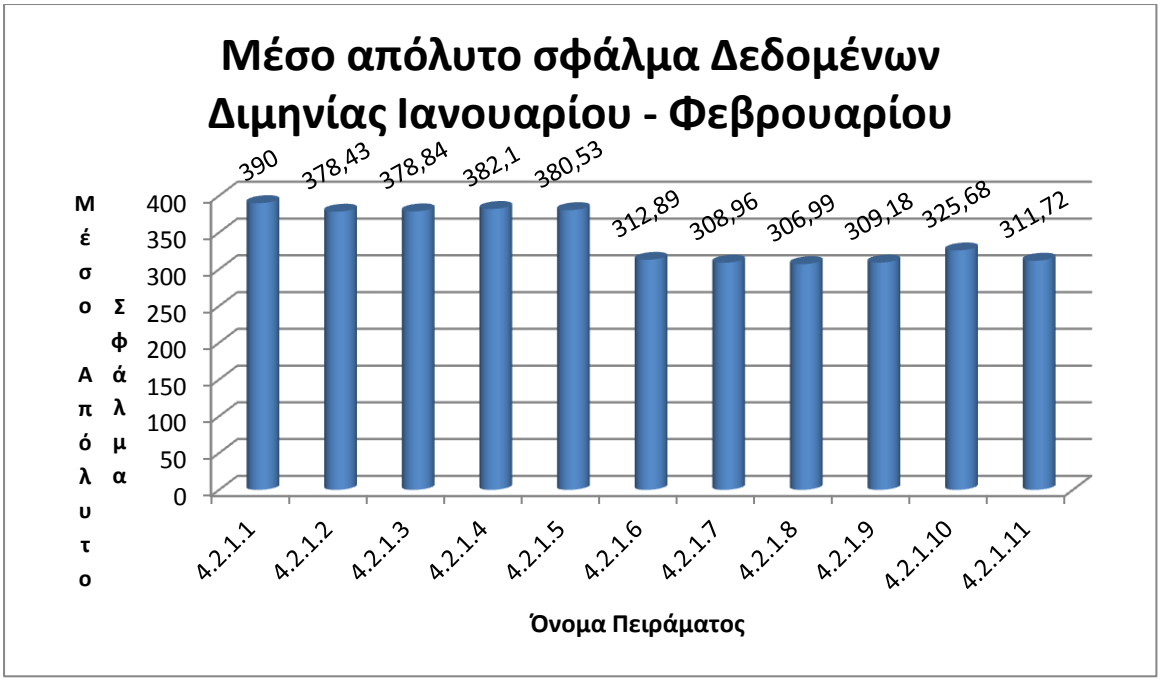
ΕΠΑ.: Επαναλήψεις

Μ.Α. Σφάλμα: Μέσο απόλυτο σφάλμα

4.2.1 Διμηνία Ιανουαρίου - Φεβρουαρίου

Ο.Π.	Δ.Α.Τ.	Ν.Π.Ε	Ν.Δ.Ε	Ν.ΕΙ.	Ν.ΕΞ.	Ρ.Μ.	ΟΡΜ.	ΕΠΑ.	Μ. Α. Σφάλμα
4.2.1.1	Όχι	8	6	25	1	0.2	0.2	224	390
4.2.1.2	Όχι	8	6	25	1	0.1	0.2	358	378,43
4.2.1.3	Όχι	8	6	25	1	0.1	0.1	358	378,84
4.2.1.4	Όχι	8	6	25	1	0.01	0.1	4171	382,1
4.2.1.5	Όχι	9	7	25	1	0.01	0.1	3816	380,53
4.2.1.6	Ναι	8	6	26	1	0.1	0.1	456	312,89
4.2.1.7	Ναι	8	6	26	1	0.01	0.1	4172	308.96
4.2.1.8	Ναι	9	7	26	1	0.01	0.1	7351	306,99
4.2.1.9	Ναι	8	6	34	1	0.1	0.1	279	309.18
4.2.1.10	Ναι	8	6	34	1	0.01	0.1	3390	325.68
4.2.1.11	Ναι	9	7	34	1	0.15	0.1	3266	311,72

Πίνακας 4.1 : Πειράματα (παραμέτροι και τελικό μέσο απόλυτο σφάλμα) 1 μέχρι 11 με χρήση νευρωνικού δικτύου συνάρτησης στα δεδομένα διμηνίας Ιανουαρίου - Φεβρουαρίου



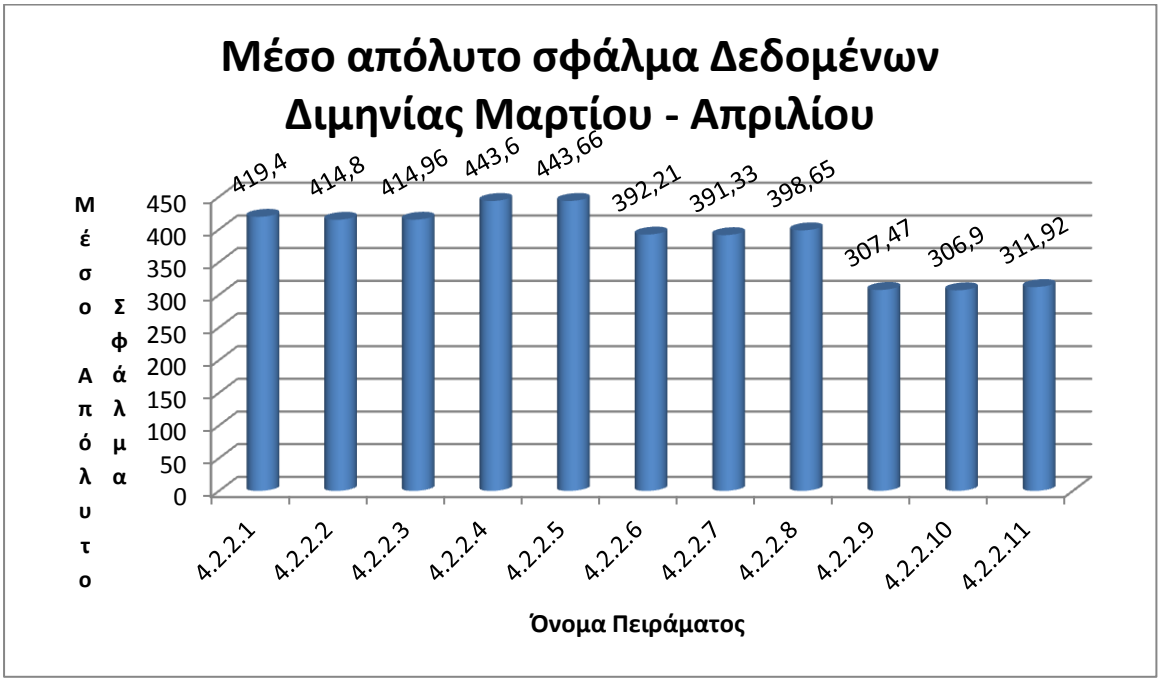
Γράφημα 4.1 : Μέσο απόλυτο σφάλμα για κάθε ένα από τα πειράματα 1 μέχρι 11 με χρήση νευρωνικού δικτύου συνάρτησης στα δεδομένα διμηνίας Ιανουαρίου - Φεβρουαρίου

Αποτελέσματα: Στην διμηνία αυτή, ο μέση τιμή της συνολικής κατανάλωσης είναι 1092 και η τυπική απόκλιση είναι 645. Μετά από την εκπαίδευση του αλγορίθμου, και από την εφαρμογή του σε *άγνωστα δεδομένα* είχαμε αρκετά σημαντική επιτυχία αφού είδαμε ότι ο αλγόριθμος μπορεί να μάθει από τα δεδομένα, και συνεπώς εντοπίζει μία σχέση μεταξύ τους. Πιο συγκεκριμένα, με είσοδο το σύνολο ελέγχου χωρίς προηγούμενη προεπεξεργασία (μονάχα κανονικοποίηση των δεδομένων), το μέσο απόλυτο σφάλμα είχε φθάσει τα 378,43. Με την προσθήκη της μεταβλητής FLAT, και με διαγραφή των αδικαιολόγητα εξαιρετικά υψηλής και χαμηλής κατανάλωσης εγγραφών, το μέσο απόλυτο σφάλμα στα δεδομένα είχε φθάσει το 307 ενώ με την κατηγοριοποίηση των αριθμητικών μεταβλητών σε τρεις ομάδες την κάθε μία, είχαμε μέσο απόλυτο σφάλμα το οποίο ήταν 309. Έτσι μπορούμε να δούμε ότι το μέσο απόλυτο σφάλμα στο δεύτερο και τρίτο σύνολο δεδομένων ήταν αρκετά χαμηλότερο από το πρώτο. Επίσης στη δεύτερη περίπτωση στην διμηνία αυτή, υπήρχαν ελαφρώς καλύτερα αποτελέσματα από ότι στην τρίτη περίπτωση

4.2.2 Διμηνία Μαρτίου - Απριλίου

Ο.Π.	Δ.Α.Τ.	Ν.Π.Ε	Ν.Δ.Ε	Ν.ΕΙ.	Ν.ΕΞ.	Ρ.Μ.	ΟΡΜ.	ΕΠΑ.	Μ. Α. Σφάλμα
4.2.2.1	Όχι	8	6	25	1	0.2	0.2	361	419,4
4.2.2.2	Όχι	8	6	25	1	0.1	0.2	825	414,8
4.2.2.3	Όχι	8	6	25	1	0.1	0.1	793	414.96
4.2.2.4	Όχι	8	6	25	1	0.01	0.1	4994	443.66
4.2.2.5	Όχι	9	7	25	1	0.01	0.1	4989	443.66
4.2.2.6	Ναι	8	6	26	1	0.1	0.1	643	392.21
4.2.2.7	Ναι	8	6	26	1	0.01	0.1	10850	391.33
4.2.2.8	Ναι	9	7	26	1	0.01	0.1	5826	398,65
4.2.2.9	Ναι	8	6	34	1	0.1	0.1	197	307,47
4.2.2.10	Ναι	8	6	34	1	0.01	0.1	1704	306,9
4.2.2.11	Ναι	9	7	34	1	0.15	0.1	350	311,92

Πίνακας 4.2 : Πειράματα (παραμέτροι και τελικό μέσο απόλυτο σφάλμα) 1 μέχρι 11 με χρήση νευρωνικού δικτύου συνάρτησης στα δεδομένα διμηνίας Μαρτίου - Απριλίου



Γράφημα 4.2 : Μέσο απόλυτο σφάλμα για κάθε ένα από τα πειράματα 1 μέχρι 11 με χρήση νευρωνικού δικτύου συνάρτησης στα δεδομένα διμηνίας Μαρτίου - Απριλίου

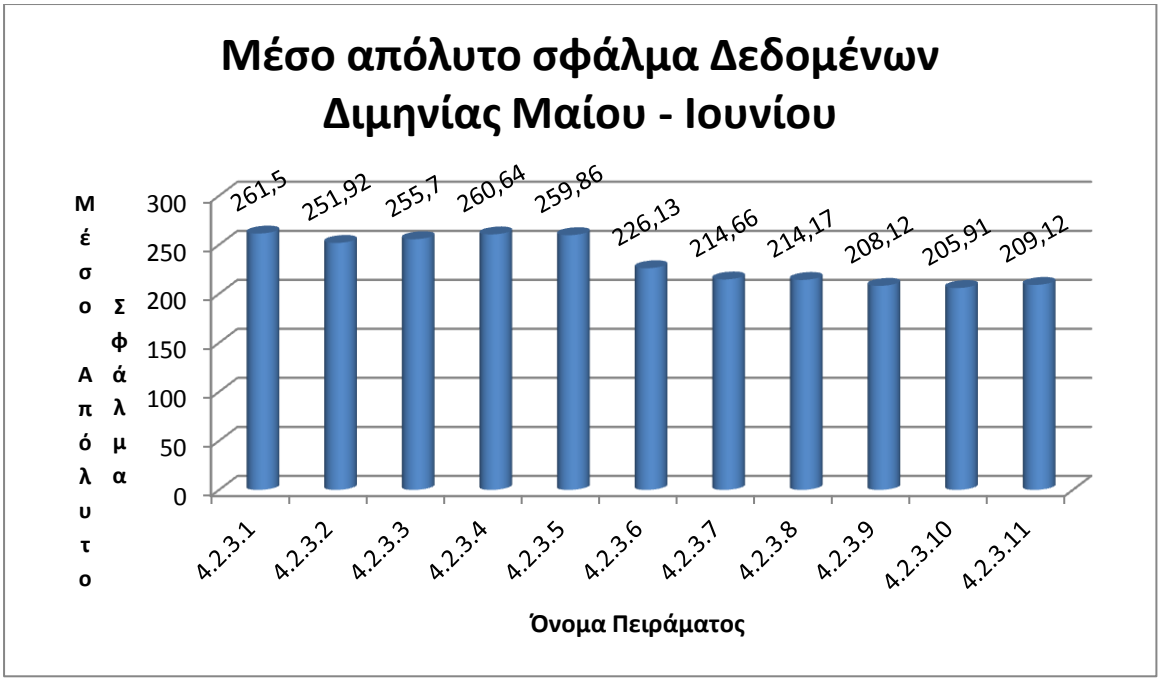
Αποτελέσματα:

Στην διμηνία αυτή, ο μέση τιμή της συνολικής κατανάλωσης είναι 986 και η τυπική απόκλιση είναι 644. Μετά από την εκπαίδευση του αλγορίθμου, και από την εφαρμογή του σε *άγνωστα δεδομένα* είχαμε αρκετά σημαντική επιτυχία αφού είδαμε ότι ο αλγόριθμος μπορεί να μάθει από τα δεδομένα, και συνεπώς εντοπίζει μία σχέση μεταξύ τους. Πιο συγκεκριμένα, με είσοδο το σύνολο ελέγχου χωρίς προηγούμενη προεπεξεργασία (μονάχα κανονικοποίηση των δεδομένων), το μέσο απόλυτο σφάλμα είχε φθάσει τα 414,8. Με την προσθήκη της μεταβλητής FLAT, και με διαγραφή των αδικαιολόγητα εξαιρετικά υψηλής και χαμηλής κατανάλωσης εγγραφών, το μέσο απόλυτο σφάλμα στα δεδομένα είχε φθάσει το 391,33 ενώ με την κατηγοριοποίηση των αριθμητικών μεταβλητών σε τρεις ομάδες την κάθε μία, είχαμε μέσο απόλυτο σφάλμα το οποίο ήταν 306. Έτσι μπορούμε να δούμε ότι το μέσο απόλυτο σφάλμα στο δεύτερο και τρίτο σύνολο δεδομένων ήταν αρκετά χαμηλότερο από το πρώτο. Επίσης στη τρίτη περίπτωση στην διμηνία αυτή, υπήρχαν αρκετά καλύτερα αποτελέσματα από ότι στην δεύτερη περίπτωση

4.2.3 Διμηνία Μαΐου - Ιουνίου

Ο.Π.	Δ.Α.Τ.	Ν.Π.Ε	Ν.Δ.Ε	Ν.ΕΙ.	Ν.ΕΞ.	Ρ.Μ.	ΟΡΜ.	ΕΠΑ.	Μ. Α. Σφάλμα
4.2.3.1	Όχι	8	6	25	1	0.2	0.2	256	261.5
4.2.3.2	Όχι	8	6	25	1	0.1	0.2	520	251,92
4.2.3.3	Όχι	8	6	25	1	0.1	0.1	532	255.70
4.2.3.4	Όχι	8	6	25	1	0.01	0.1	6699	260.64
4.2.3.5	Όχι	9	7	25	1	0.01	0.1	7340	259.86
4.2.3.6	Ναι	8	6	26	1	0.1	0.1	274	226.13
4.2.3.7	Ναι	8	6	26	1	0.01	0.1	4618	214.66
4.2.3.8	Ναι	9	7	26	1	0.01	0.1	3966	214.17
4.2.3.9	Ναι	8	6	34	1	0.1	0.1	416	208,12
4.2.3.10	Ναι	10	10	34	1	0.01	0.1	9997	205.91
4.2.3.11	Ναι	9	7	34	1	0.15	0.1	305	209,12

Πίνακας 4.3 : Πειράματα (παραμέτροι και τελικό μέσο απόλυτο σφάλμα) 1 μέχρι 11 με χρήση νευρωνικού δικτύου συνάρτησης στα δεδομένα διμηνίας Μαΐου - Ιουνίου



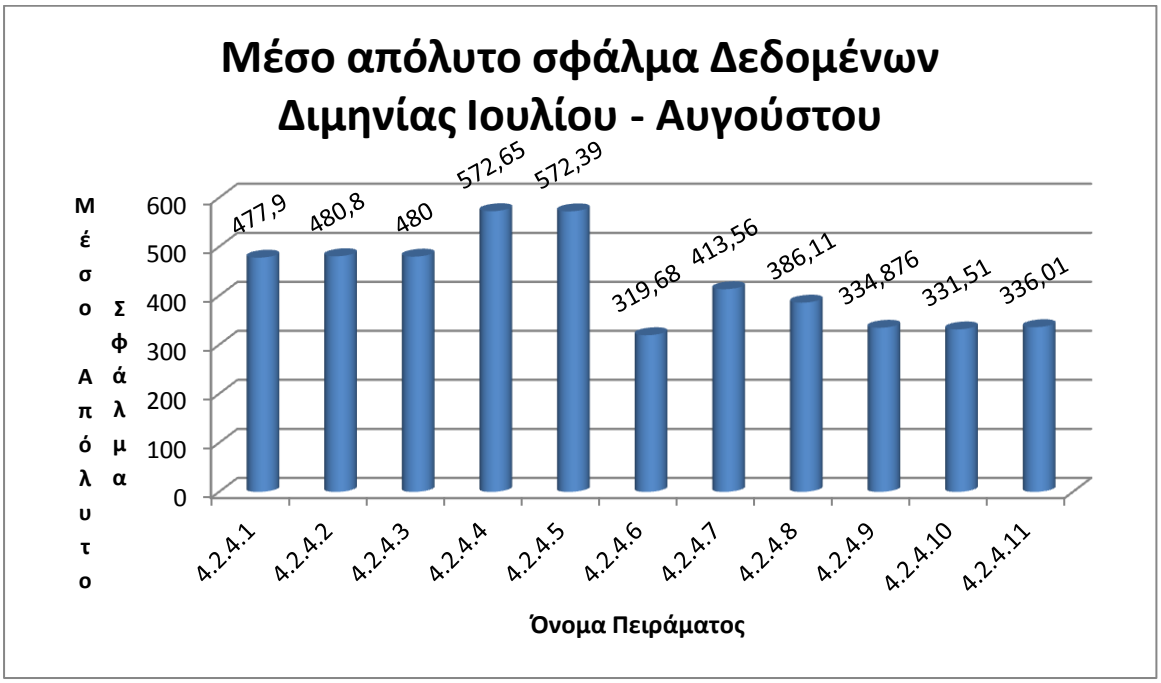
Γράφημα 4.3 : Μέσο απόλυτο σφάλμα για κάθε ένα από τα πειράματα 1 μέχρι 11 με χρήση νευρωνικού δικτύου συνάρτησης στα δεδομένα διμηνίας Μαίου - Ιουνίου

Αποτελέσματα: Στην διμηνία αυτή, ο μέση τιμή της συνολικής κατανάλωσης είναι 714 και η τυπική απόκλιση είναι 381. Μετά από την εκπαίδευση του αλγορίθμου, και από την εφαρμογή του σε *άγνωστα δεδομένα* είχαμε αρκετά σημαντική επιτυχία αφού είδαμε ότι ο αλγόριθμος μπορεί να μάθει από τα δεδομένα, και συνεπώς εντοπίζει μία σχέση μεταξύ τους. Πιο συγκεκριμένα, με είσοδο το σύνολο ελέγχου χωρίς προηγούμενη προεπεξεργασία (μονάχα κανονικοποίηση των δεδομένων), το μέσο απόλυτο σφάλμα είχε φθάσει τα 255,7. Με την προσθήκη της μεταβλητής FLAT, και με διαγραφή των αδικαιολόγητα εξαιρετικά υψηλής και χαμηλής κατανάλωσης εγγραφών, το μέσο απόλυτο σφάλμα στα δεδομένα είχε φθάσει το 214,17 ενώ με την κατηγοριοποίηση των αριθμητικών μεταβλητών σε τρεις ομάδες την κάθε μία, είχαμε μέσο απόλυτο σφάλμα το οποίο ήταν 205,91. Έτσι μπορούμε να δούμε ότι το μέσο απόλυτο σφάλμα στο δεύτερο και τρίτο σύνολο δεδομένων ήταν αρκετά χαμηλότερο από το πρώτο. Επίσης στη τρίτη περίπτωση στην διμηνία αυτή, υπήρχαν ελαφρώς καλύτερα αποτελέσματα από ότι στην δεύτερη περίπτωση.

4.2.4 Διμηνία Ιουλίου - Αυγούστου

Ο.Π.	Δ.Α.Τ.	Ν.Π.Ε	Ν.Δ.Ε	Ν.ΕΙ.	Ν.ΕΞ.	Ρ.Μ.	ΟΡΜ.	ΕΠΑ.	Μ. Α. Σφάλμα
4.2.4.1	Όχι	8	6	25	1	0.2	0.2	285	477.9
4.2.4.2	Όχι	8	6	25	1	0.1	0.2	530	480.8
4.2.4.3	Όχι	8	6	25	1	0.1	0.1	694	480.0
4.2.4.4	Όχι	8	6	25	1	0.01	0.1	16586	572,65
4.2.4.5	Όχι	9	7	25	1	0.01	0.1	20648	572.39
4.2.4.6	Ναι	8	6	26	1	0.1	0.1	440	319.68
4.2.4.7	Ναι	8	6	26	1	0.01	0.1	1393	413,56
4.2.4.8	Ναι	9	7	26	1	0.01	0.1	8043	386.11
4.2.4.9	Ναι	8	6	34	1	0.1	0.1	116	334.876
4.2.4.10	Ναι	8	8	34	1	0.01	0.1	3275	331.51
4.2.4.11	Ναι	9	7	34	1	0.15	0.1	61	336.01

Πίνακας 4.4 : Πειράματα (παραμέτροι και τελικό μέσο απόλυτο σφάλμα) 1 μέχρι 11 με χρήση νευρωνικού δικτύου συνάρτησης στα δεδομένα διμηνίας Ιουλίου-Αυγούστου



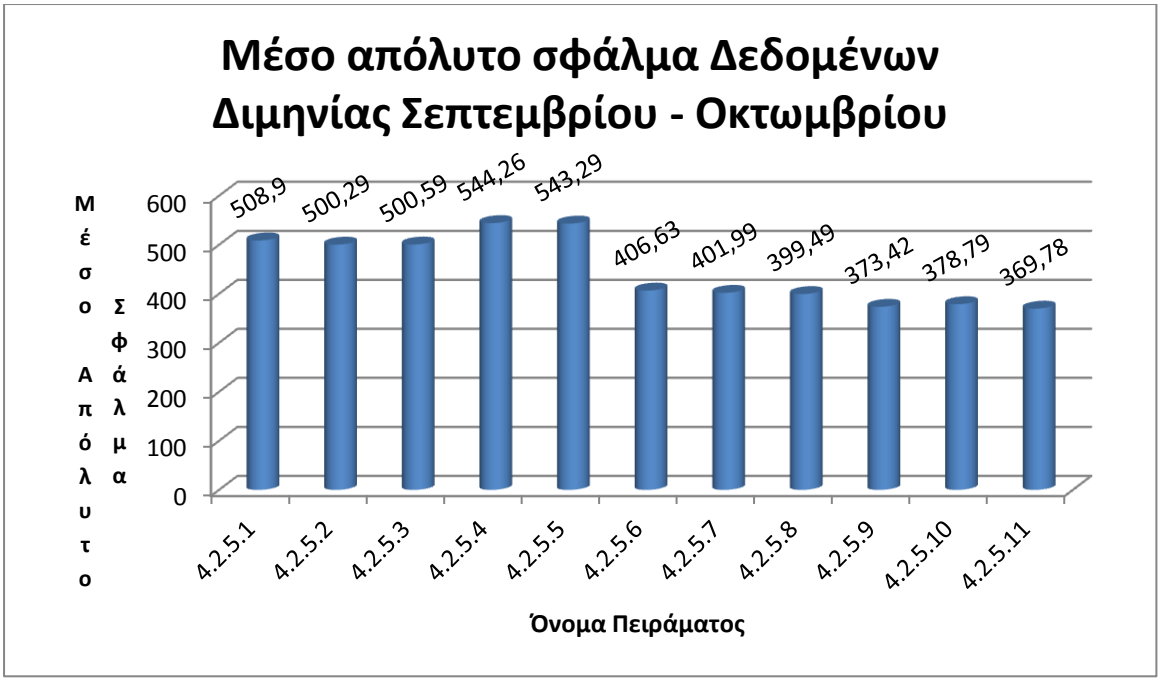
Γράφημα 4.4 : Μέσο απόλυτο σφάλμα για κάθε ένα από τα πειράματα 1 μέχρι 11 με χρήση νευρωνικού δικτύου συνάρτησης στα δεδομένα διμηνίας Ιουλίου - Αυγούστου

Αποτελέσματα: Στην διμηνία αυτή, ο μέση τιμή της συνολικής κατανάλωσης είναι 1149 και η τυπική απόκλιση είναι 741. Μετά από την εκπαίδευση του αλγορίθμου, και από την εφαρμογή του σε *άγνωστα δεδομένα* είχαμε αρκετά σημαντική επιτυχία αφού είδαμε ότι ο αλγόριθμος μπορεί να μάθει από τα δεδομένα, και συνεπώς εντοπίζει μία σχέση μεταξύ τους. Πιο συγκεκριμένα, με είσοδο το σύνολο ελέγχου χωρίς προηγούμενη προεπεξεργασία (μονάχα κανονικοποίηση των δεδομένων), το μέσο απόλυτο σφάλμα είχε φθάσει τα 477,9. Με την προσθήκη της μεταβλητής FLAT, και με διαγραφή των αδικαιολόγητα εξαιρετικά υψηλής και χαμηλής κατανάλωσης εγγραφών, το μέσο απόλυτο σφάλμα στα δεδομένα είχε φθάσει το 319,68 ενώ με την κατηγοριοποίηση των αριθμητικών μεταβλητών σε τρεις ομάδες την κάθε μία, είχαμε μέσο απόλυτο σφάλμα το οποίο ήταν 331,51. Έτσι μπορούμε να δούμε ότι το μέσο απόλυτο σφάλμα στο δεύτερο και τρίτο σύνολο δεδομένων ήταν αρκετά χαμηλότερο από το πρώτο. Επίσης στη δεύτερη περίπτωση στην διμηνία αυτή, υπήρχαν ελαφρώς καλύτερα αποτελέσματα από ότι στην τρίτη περίπτωση.

4.2.5 Διμηνία Σεπτεμβρίου – Οκτωβρίου

Ο.Π.	Δ.Α.Τ.	Ν.Π.Ε	Ν.Δ.Ε	Ν.ΕΙ.	Ν.ΕΞ.	Ρ.Μ.	ΟΡΜ.	ΕΠΑ.	Μ. Α. Σφάλμα
4.2.5.1	Όχι	8	6	25	1	0.2	0.2	302	508.9
4.2.5.2	Όχι	8	6	25	1	0.1	0.2	645	500.29
4.2.5.3	Όχι	8	6	25	1	0.1	0.1	627	500.59
4.2.5.4	Όχι	8	6	25	1	0.01	0.1	6312	544.26
4.2.5.5	Όχι	9	7	25	1	0.01	0.1	6102	543.29
4.2.5.6	Ναι	8	6	26	1	0.1	0.1	384	406.63
4.2.5.7	Ναι	8	6	26	1	0.01	0.1	11179	401.99
4.2.5.8	Ναι	9	7	26	1	0.01	0.1	8110	399.49
4.2.5.9	Ναι	8	6	34	1	0.1	0.1	157	373.42
4.2.5.10	Ναι	8	8	34	1	0.01	0.1	3254	378.79
4.2.5.11	Ναι	9	7	34	1	0.15	0.1	110	369.78

Πίνακας 4.5 : Πειράματα (παραμέτροι και τελικό μέσο απόλυτο σφάλμα) 1 μέχρι 11 με χρήση νευρωνικού δικτύου συνάρτησης στα δεδομένα διμηνίας Σεπτεμβρίου - Οκτωμβρίου



Γράφημα 4.5 : Μέσο απόλυτο σφάλμα για κάθε ένα από τα πειράματα 1 μέχρι 11 με χρήση νευρωνικού δικτύου συνάρτησης στα δεδομένα διμηνίας Σεπτεμβρίου - Οκτωμβρίου

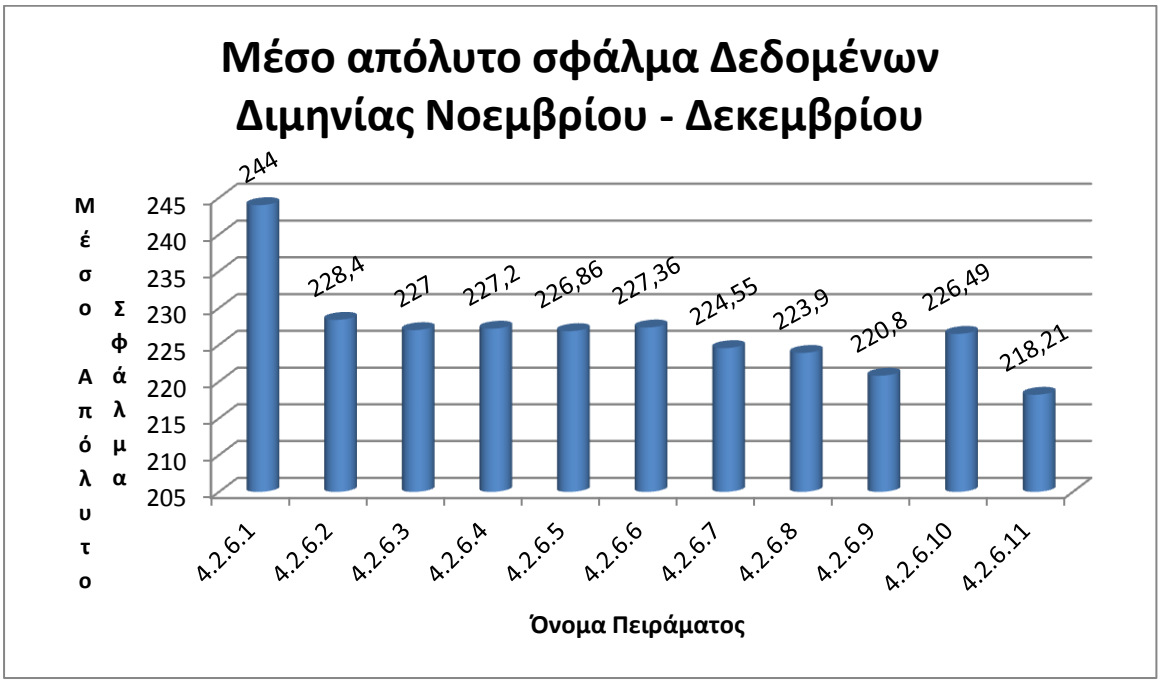
Αποτελέσματα:

Στην διμηνία αυτή, ο μέση τιμή της συνολικής κατανάλωσης είναι 1156 και η τυπική απόκλιση είναι 725. Μετά από την εκπαίδευση του αλγορίθμου, και από την εφαρμογή του σε *άγνωστα δεδομένα* είχαμε αρκετά σημαντική επιτυχία αφού είδαμε ότι ο αλγόριθμος μπορεί να μάθει από τα δεδομένα, και συνεπώς εντοπίζει μία σχέση μεταξύ τους. Πιο συγκεκριμένα, με είσοδο το σύνολο ελέγχου χωρίς προηγούμενη προεπεξεργασία (μονάχα κανονικοποίηση των δεδομένων), το μέσο απόλυτο σφάλμα είχε φθάσει τα 500,29. Με την προσθήκη της μεταβλητής FLAT, και με διαγραφή των αδικαιολόγητα εξαιρετικά υψηλής και χαμηλής κατανάλωσης εγγραφών, το μέσο απόλυτο σφάλμα στα δεδομένα είχε φθάσει το 399,49 ενώ με την κατηγοριοποίηση των αριθμητικών μεταβλητών σε τρεις ομάδες την κάθε μία, είχαμε μέσο απόλυτο σφάλμα το οποίο ήταν 369,78. Έτσι μπορούμε να δούμε ότι το μέσο απόλυτο σφάλμα στο δεύτερο και τρίτο σύνολο δεδομένων ήταν αρκετά χαμηλότερο από το πρώτο. Επίσης στη τρίτη περίπτωση στην διμηνία αυτή, υπήρχαν ελαφρώς καλύτερα αποτελέσματα από ότι στην δεύτερη περίπτωση.

4.2.6 Διμηνία Νοεμβρίου - Δεκεμβρίου

Ο.Π.	Δ.Α.Τ.	Ν.Π.Ε	Ν.Δ.Ε	Ν.ΕΙ.	Ν.ΕΞ.	Ρ.Μ.	ΟΡΜ.	ΕΠΑ.	Μ. Α. Σφάλμα
4.2.6.1	Όχι	8	6	25	1	0.2	0.2	327	244
4.2.6.2	Όχι	8	6	25	1	0.1	0.2	99	228.4
4.2.6.3	Όχι	8	6	25	1	0.1	0.1	135	227
4.2.6.4	Όχι	8	6	25	1	0.01	0.1	2637	227.20
4.2.6.5	Όχι	9	7	25	1	0.01	0.1	1773	226,86
4.2.6.6	Ναι	8	6	26	1	0.1	0.1	555	227.36
4.2.6.7	Ναι	8	6	26	1	0.01	0.1	5124	224.55
4.2.6.8	Ναι	9	7	26	1	0.01	0.1	4151	223.9
4.2.6.9	Ναι	8	6	34	1	0.1	0.1	156	220.80
4.2.6.10	Ναι	8	6	34	1	0.01	0.1	1306	226.49
4.2.6.11	Ναι	9	7	34	1	0.15	0.1	85	218.21

Πίνακας 4.6 : Πειράματα (παραμέτροι και τελικό μέσο απόλυτο σφάλμα) 1μέχρι 11 με χρήση νευρωνικού δικτύου συνάρτησης στα δεδομένα διμηνίας Νοεμβρίου - Δεκεμβρίου



Γράφημα 4.6 : Μέσο απόλυτο σφάλμα για κάθε ένα από τα πειράματα 1 μέχρι 11 με χρήση νευρωνικού δικτύου συνάρτησης στα δεδομένα διμηνίας Νοεμβρίου – Δεκεμβρίου

Αποτελέσματα: Στην διμηνία αυτή, ο μέση τιμή της συνολικής κατανάλωσης είναι 704 και η τυπική απόκλιση είναι 360. Μετά από την εκπαίδευση του αλγορίθμου, και από την εφαρμογή του σε *άγνωστα δεδομένα* είχαμε αρκετά σημαντική επιτυχία αφού είδαμε ότι ο αλγόριθμος μπορεί να μάθει από τα δεδομένα, και συνεπώς εντοπίζει μία σχέση μεταξύ τους. Πιο συγκεκριμένα, με είσοδο το σύνολο ελέγχου χωρίς προηγούμενη προεπεξεργασία (μονάχα κανονικοποίηση των δεδομένων), το μέσο απόλυτο σφάλμα είχε φθάσει τα 226,86. Με την προσθήκη της μεταβλητής FLAT, και με διαγραφή των αδικαιολόγητα εξαιρετικά υψηλής και χαμηλής κατανάλωσης εγγραφών, το μέσο απόλυτο σφάλμα στα δεδομένα είχε φθάσει το 223,9 ενώ με την κατηγοριοποίηση των αριθμητικών μεταβλητών σε τρεις ομάδες την κάθε μία, είχαμε μέσο απόλυτο σφάλμα το οποίο ήταν 218,21. Έτσι μπορούμε να δούμε ότι το μέσο απόλυτο σφάλμα στο δεύτερο και τρίτο σύνολο δεδομένων ήταν αρκετά χαμηλότερο από το πρώτο. Επίσης στη τρίτη περίπτωση στην διμηνία αυτή, υπήρχαν ελαφρώς καλύτερα αποτελέσματα από ότι στην δεύτερη περίπτωση.

Συμπεράσματα Πειράμάτων MLP Function

Πριν αναφέρω τα συμπεράσματα μας, θα ήθελα να αναφέρω ότι το πιο πάνω μέσο απόλυτο σφάλμα το οποίο υπάρχει στους παραπάνω πίνακες, είναι ο μέσος όρος 5 εκτελέσεων του πειράματος στα ίδια δεδομένα. Ο λόγος που τρέχαμε κάθε πείραμα 5 φορές, είναι για να παρατηρήσουμε ορθότερα τα αποτελέσματα των αλγορίθμων μας καθώς με μόνο μία εκτέλεση, θα μπορούσαμε να καταλήξουμε σε μη αντικατοπτριστικά δεδομένα ως προς τις παραμέτρους του αλγορίθμου και το δείγμα μας.

Εδώ μπορούμε να παράξουμε ως σευμπέρασμα το γεγονός ότι οι αλγόριθμοι έχουν ένα μεγάλο ποσοστό επιτυχίας, καθώς ενώ υπάρχει μία μεγάλη μία μεγάλη διακύμανση στην κατανάλωση των δεδομένων στις διάφορες οικίες, ο αλγόριθμος βρίσκει λύσεις οι οποίες δεν είναι εξαιρετικές αλλά δίνουν μία τιμή η οποία είναι βελτιωμένη κατά πολύ σε σχέση με τη διακύμανση. Επίσης μπορούμε να παρατηρήσουμε ότι στα πρώτα 5 πειράματα για κάθε μία από τις διμηνίες, στα οποία τα δεδομένα εισάγονταν ως έχουν, τα αποτελέσματα είναι ιδιαίτερα βοηθητικά. Εντούτοις στα επόμενα τρία πειράματα, εφόσον είχαμε δημιουργήσει και την μεταβλητή FLAT και είχαμε αφαιρέσει ένα μικρό ποσοστό δεδομένων τα οποία είχαν αδικαιολόγητα μικρή ή μεγάλη κατανάλωση (επειδή μπορεί να ήταν απλά εξοχικές κατοικίες

για παράδειγμα, ή η οικία χρησιμοποιώταν και ως εργασιακός χώρος) τα αποτελέσματα των αλγορίθμων είχαν βελτιωθεί κατά πολύ.
Είχαμε παρατηρήσει όμως ότι δεν «ενδιαφέρει» ένα αλγόριθμο εάν το σπίτι είναι 190 ή 200 τετραγωνικά μέτρα, το οποίο χαρακτηριστικό είναι αριθμητικό και συνεπώς ίσως δυσκολεύει ένα αλγόριθμο να μάθει. Έτσι είχαμε τρέξει και τους αλγορίθμους με κλάσεις στις αριθμητικές παραμέτρους τους. Στα πειράματα αυτά (Πείραμα 8 μέχρι Πείραμα 11). Έχουμε παρατηρήσει ότι στην περίπτωση αυτή έχουμε ακόμη καλύτερα αποτελέσματα.

4.3 Νευρωνικά Δίκτυα (BackProp) Κατηγοριοποίησης

Για να είμαστε σε θέση να αναλύσουμε τα δεδομένα μας με χρήση νευρωνικού δικτύου, έχω υλοποιήσει στη γλώσσα προγραμματισμού Java, ένα νευρωνικό δίκτυο MLP εκπαιδευόμενο με Back-propagation το οποίο χρησιμοποιείται για κατηγοριοποίηση.

Τα νευρωνικά δίκτυα, απαιτούσαν τη δημιουργία δύο αρχείων με εγγραφές, το καθένα με ένα διαφορετικό σύνολο εγγραφών. Ένα για εκπαίδευση του αλγορίθμου και ένα για έλεγχό του. Οι εγγραφές αυτές έπρεπε να ήταν τυχαία από τη βάση δεδομένων. Για να είναι κατανοητός ο τρόπος με τον οποίο πήρα τα δεδομένα, έχω παραθέσει τα ερωτήματα SQL τα οποίο είχα συγγράψει, στο Β4 και Β5, και τα οποία δημιουργούν τα δύο σύνολα τα οποία είχαμε περάσει στον συγκεκριμένο αλγόριθμο.

Όπως φαίνεται και από τις επερωτήσεις SQL, είχα διαμοιράσει *τυχαία* 375 εγγραφές για την εκπαίδευση ενώ τις υπόλοιπες περίπου 125 για τον έλεγχο του αλγορίθμου.

Ο αλγόριθμος νευρωνικών δικτύων τον οποίο έχω υλοποιήσει, παίρνει σαν παραμέτρους τους κόμβους εισόδου, εξόδου, και τους κόμβους για κάθε ένα από δύο κρυφά επίπεδα νευρώνων. Επίσης παίρνει σαν παράμετρο τον ρυθμό μάθησης, και την ορμή του νευρωνικού δικτύου, καθώς επίσης τα αρχεία τα οποία θα δεχθεί για εκπαίδευση καθώς και για έλεγχο. Ενδεικτικό αρχείο παραμέτρων βρίσκεται στο Α4, και ο κώδικας του νευρωνικού δικτύου βρίσκονται στα παραρτήματα Α2 (neuron.java) και Α3(Το δίκτυο ως έχει).

Σκοπός του νευρωνικού αυτού δικτύου δεν είναι να παράξει μία τιμή όπως στο νευρωνικό δίκτυο το οποίο παρουσιάζεται στο υποκεφάλαιο 4.1, αλλά να κατηγοριοποιήσει ένα σπίτι σε μία κατηγορία κατανάλωσης (πχ ανήκει στην κατηγορία 1000-1150 KW). Αυτό είναι

επιθυμητό αρκετές φορές αφού ίσως να μην μπορεί ή να μην χρειάζεται το νευρωνικό δίκτυο να παράξει μία τιμή πολύ κοντά στην πραγματική, αλλά να είναι πιο χρήσιμο να κατηγοριοποιήσει την εγγραφή σε μια κατηγορία.

Σε αντίθεση με το νευρωνικό το οποίο παρουσιάζεται στο υποκεφάλαιο 4.2, στο οποίο παρουσιάζουμε το μέσο απόλυτο σφάλμα, ούτως ώστε να μπορούμε να δούμε πόσο καλά είναι τα αποτελέσματα του αλγορίθμου μας, στο υποκεφάλαιο αυτό, παρουσιάζουμε τον ρυθμό επιτυχίας. Αυτό είναι λόγω του ότι θέλουμε να δούμε το ποσοστό των επιτυχημένων κατηγοριοποιήσεων των εγγραφών.

Είχαν γίνει αρκετά πειράματα αντιπροσωπευτικό *δείγμα* των οποίων περιγράφω ως πιο κάτω.

Πειράματα 1,2 και 3: Τα δεδομένα έχουν εισαχθεί στον αλγόριθμο με την δημιουργία του χαρακτηριστικού FLAT από τα χαρακτηριστικά House και Maisonette, καθώς επίσης είχαν διαγραφεί οι εγγραφές οι οποίες είχαν κατανάλωση πάρα πολύ χαμηλή ή πάρα πολύ ψηλή, χωρίς να δικαιολογούνται από τα χαρακτηριστικά της οικίας. Ο αλγόριθμος στη συνέχεια τα κανονικοποιεί στην τιμή -1 μέχρι 1. Για αρχή έχουμε ξεκινήσει με ένα σχετικά μικρό ρυθμό μάθησης και ρυθμό ορμής. Στα τρία αυτά πειράματα μεταβάλλουμε ελαφρώς την αρχιτεκτονική του δικτύου (τον αριθμό των νευρώνων σε κάθε κρυφό επίπεδο), τον ρυθμό μάθησης αλλά και τον ρυθμό ορμής. Επίσης πίνακα στην στήλη επαναλήψεων στους πιο κάτω πίνακες παρουσιάζεται η επανάληψη στην οποία έχουμε το μεγαλύτερο success rate. Τα χαρακτηριστικά τα οποία εισάγονται στον αλγόριθμο είναι τα 26 χαρακτηριστικά, ενώ απαιτείται από τον αλγόριθμο να παράγει μία τιμή, η οποία είναι μία εκ των τεσσάρων κατηγοριών κατανάλωσης της οικίας τις οποίες έχουμε δημιουργήσει.

Πειράματα 4,5 και 6: Αυτή τη φορά εισάγαμε τρεις ομάδες για κάθε ένα από τα αριθμητικά χαρακτηριστικά. Δηλαδή τώρα να παίρνει σαν είσοδο μία τιμή για κάποιο αριθμητικό χαρακτηριστικό (πχ τετραγωνικά μέτρα) παίρνει την ομάδα του (πχ μικρό , μέτριο, μεγάλο). Επίσης στα τρία αυτά πειράματα μεταβάλλουμε ελαφρώς την αρχιτεκτονική του δικτύου (τον αριθμό των νευρώνων σε κάθε κρυφό επίπεδο), τον ρυθμό μάθησης αλλά και τον ρυθμό ορμής.

Ως ρυθμό επιτυχίας μετρούμε τον αριθμό των εγγραφών οι οποίες έχουν κατηγοριοποιηθεί σωστά ως προς το όλον στο σύνολο ελέγχου (στα άγνωστα δεδομένα).

Ο τίτλος των πιο κάτω πειραμάτων έχει ως εξής:

<Κεφάλαιο>.<Υποκεφάλαιο><Διμηνία (1-6)><Αύξων αριθμός πειράματος>

Είχα χωρίσει την κατανάλωση σε τέσσερις κατηγορίες. Οι κατηγορίες ήταν διαφορετικές ανάλογα με τη διμηνία, λόγω του ότι υπάρχει διαφορετική μέση κατανάλωση ανά διμηνία, και ήθελα η ανάλυσή μου να είναι όσο πιο ορθή γίνεται. Έτσι παραθέτω τις κατηγορίες για κάθε μία από τις 6 διμηνίες.

Τα όρια των τεσσάρων πιο κάτω κατηγοριών επιλέχθηκαν μετά από προσεκτική μελέτη, ούτως ώστε να περιέχουν τον ισόποσο αριθμό εγγραφών το κάθε ένα από αυτά, σε κάθε μία από τις διμηνίες.

Διμηνία	**Κατηγορία Α** **Μικρότερο από:**	**Κατηγορία Β** **Μεταξύ**	**Κατηγορία Γ** **Μεταξύ**	**Κατηγορία Δ** **Μεγαλύτερο από**
Ιανουάριος – Φεβρουάριος	700	700 και 970	971 και 1350	1350
Μάρτιος – Απρίλιος	600	600 και 850	851 και 1200	1200
Μάιος – Ιούνιος	480	481 και 650	651 και 900	900
Ιούλιος – Αύγουστος	650	651 και 1000	1001 και 1400	1400
Σεπτέμβρης – Οκτώβριος	650	651 και 950	951 και 1400	1400
Νοέμβρης – Δεκέμβρης	470	471 και 625	626 και 850	850

Επίσης δίνω τα ακρώνυμα τα οποία υπάρχουν στον πιο κάτω πίνακα

Ο.Π.: Όνομα πειράματος

ΔΙΜ.: Διμηνία

Δ.Α.Τ. : Διαγραφή Αποκλίνουσων τιμών

Ν.Π.Ε: Νευρώνες πρώτου επιπέδου

Ν.Δ.Ε: Νευρώνες δεύτερου επίπεδου

Ν.ΕΙ: Νευρώνες Εισόδου

Ν.ΕΞ.: Νευρώνες Εξόδου

Ρ.Μ.: Ρυθμός Μάθησης (Learning Rate)

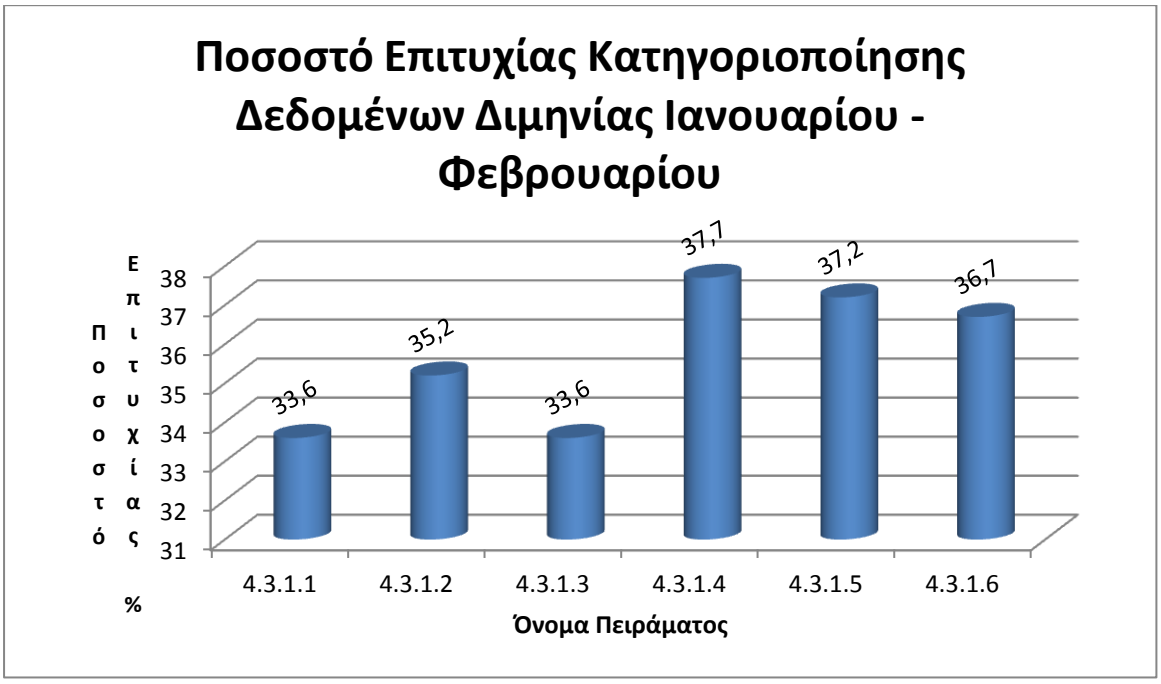
ΟΡΜ.: Ορμή (momentum)

ΕΠΑ.: Επαναλήψεις

4.3.1 Διμηνία Ιανουαρίου - Φεβρουαρίου

Ο.Π.	Δ.Α.Τ.	Ν.Π.Ε	Ν.Δ.Ε	Ν.ΕΙ.	Ν.ΕΞ.	Ρ.Μ.	ΟΡΜ.	ΕΠΑ.	Ποσοστό Επιτυχίας %
4.3.1.1	Ναι	8	6	26	4	0.1	0.1	264	33,6
4.3.1.2	Ναι	9	8	26	4	0.15	0.1	283	35,2
4.3.1.3	Ναι	9	7	26	4	0.01	0.1	1434	33,6
4.3.1.4	Ναι	9	9	34	4	0.3	0.05	83	37,7
4.3.1.5	Ναι	9	9	34	4	0.15	0.05	419	37,2
4.3.1.6	Ναι	12	12	34	4	0.09	0.05	489	36,7

Πίνακας 4.7 : Πειράματα (παραμέτροι και τελικό μέσο απόλυτο σφάλμα) 1 μέχρι 6 με χρήση νευρωνικού δικτύου κατηγοριοποίησης στα δεδομένα διμηνίας Ιανουαρίου - Φεβρουαρίου



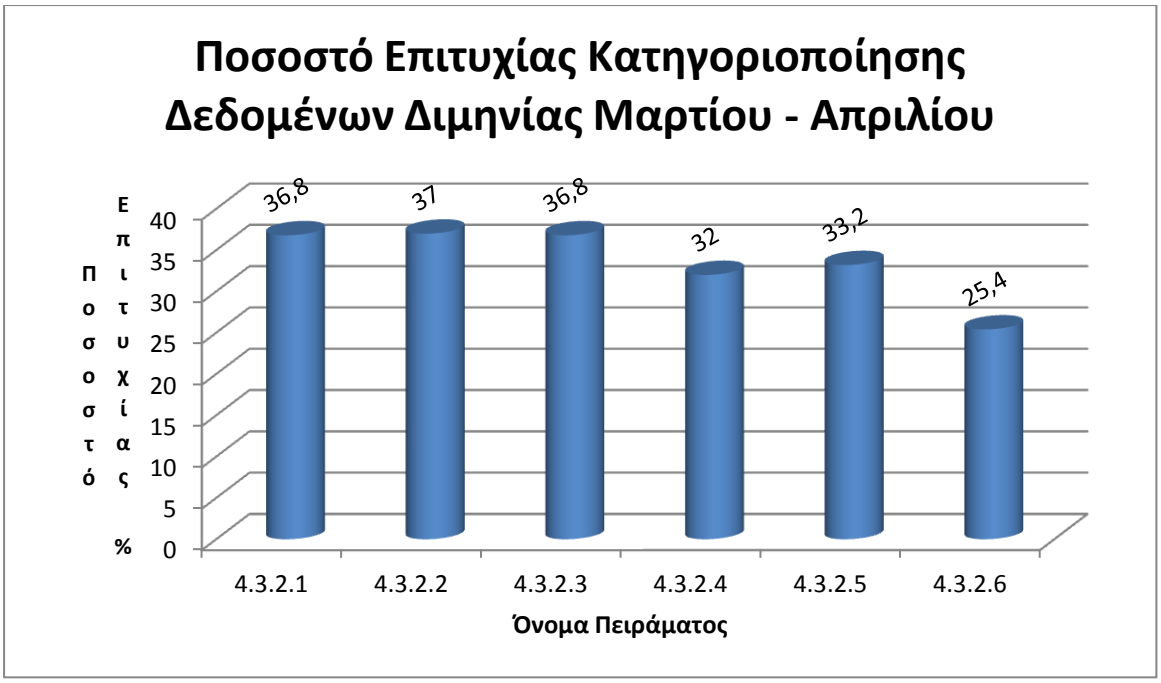
Γράφημα 4.7 : Ποσοστό επιτυχίας κατηγοριοποίησης δεδομένων σε κλάση κατανάλωσης για κάθε ένα από τα πειράματα 1 μέχρι 6 με χρήση νευρωνικού δικτύου κατηγοριοποίησης στα δεδομένα διμηνίας Ιανουαρίου - Φεβρουαρίου

Στην διμηνία αυτή, ο μέση τιμή της συνολικής κατανάλωσης είναι 1092 και η τυπική απόκλιση είναι 645. Μετά από την εκπαίδευση του αλγορίθμου, και από την εφαρμογή του σε *άγνωστα δεδομένα* το ποσοστό επιτυχίας με είσοδο το σύνολο ελέγχου το οποίο περιέχει την μεταβλητή FLAT, και τη διαγραφή των αδικαιολόγητα εξαιρετικά υψηλής και χαμηλής κατανάλωσης εγγραφών, είχε φθάσει το 35.2%. Σε αντίθεση, με την κατηγοριοποίηση των αριθμητικών μεταβλητών σε τρεις ομάδες την κάθε μία, είχαμε ποσοστό επιτυχίας 37,7. Έτσι μπορούμε να δούμε ότι το ποσοστό επυτιχίας στο δεύτερο σύνολο δεδομένων είναι γενικά υψυλότερο από το πρώτο.

4.3.2 Διμηνία Μαρτίου – Απριλίου

Ο.Π.	Δ.Α.Τ.	Ν.Π.Ε	Ν.Δ.Ε	Ν.ΕΙ.	Ν.ΕΞ.	Ρ.Μ.	ΟΡΜ.	ΕΠΑ.	Ποσοστό Επιτυχίας %
4.3.2.1	Ναι	8	6	26	4	0.1	0.2	362	36,8
4.3.2.2	Ναι	8	6	26	4	0.2	0.1	47	37
4.3.2.3	Ναι	9	7	26	4	0.01	0.1	53	36,8
4.3.2.4	Ναι	9	9	34	4	0.3	0.05	237	32
4.3.2.5	Ναι	9	9	34	4	0.15	0.05	105	33,2
4.3.2.6	Ναι	12	12	34	4	0.09	0.05	19	25,4

Πίνακας 4.8 : Πειράματα (παραμέτροι και τελικό μέσο απόλυτο σφάλμα) 1 μέχρι 6 με χρήση νευρωνικού δικτύου κατηγοριοποίησης στα δεδομένα διμηνίας Μαρτίου – Απριλίου



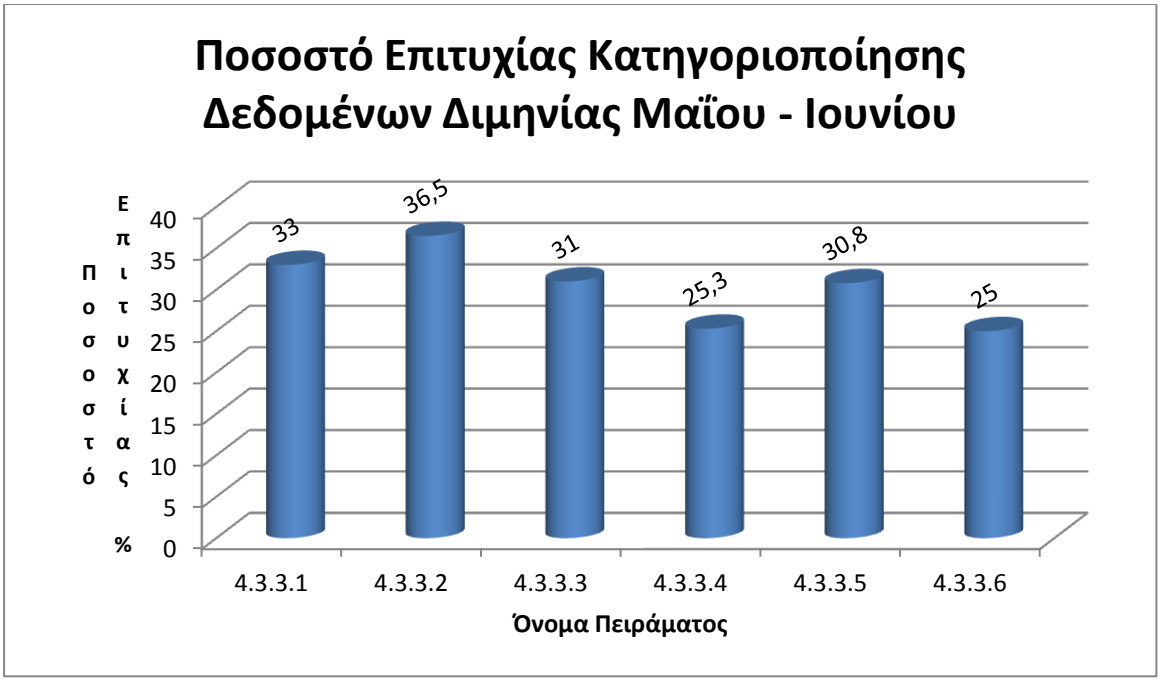
Γράφημα 4.8 : Ποσοστό επιτυχίας κατηγοριοποίησης δεδομένων σε κλάση κατανάλωσης για κάθε ένα από τα πειράματα 1 μέχρι 6 με χρήση νευρωνικού δικτύου κατηγοριοποίησης στα δεδομένα διμηνίας Μαρτίου - Απριλίου

Στην διμηνία αυτή, ο μέση τιμή της συνολικής κατανάλωσης είναι 986 και η τυπική απόκλιση είναι 644. Μετά από την εκπαίδευση του αλγορίθμου, και από την εφαρμογή του σε *άγνωστα δεδομένα* το ποσοστό επιτυχίας με είσοδο το σύνολο ελέγχου το οποίο περιέχει την μεταβλητή FLAT, και τη διαγραφή των αδικαιολόγητα εξαιρετικά υψηλής και χαμηλής κατανάλωσης εγγραφών, είχε φθάσει το 37%. Σε αντίθεση, με την κατηγοριοποίηση των αριθμητικών μεταβλητών σε τρεις ομάδες την κάθε μία, είχαμε ποσοστό επιτυχίας 33,2. Έτσι μπορούμε να δούμε ότι το ποσοστό επυτιχίας στο δεύτερο σύνολο δεδομένων είναι γενικά χαμηλότερο από το πρώτο.

4.3.3 Διμηνία Μαΐου - Ιουνίου

Ο.Π.	Δ.Α.Τ.	Ν.Π.Ε	Ν.Δ.Ε	Ν.ΕΙ.	Ν.ΕΞ.	Ρ.Μ.	ΟΡΜ.	ΕΠΑ.	Ποσοστό Επιτυχίας %
4.3.3.1	Ναι	8	6	26	4	0.1	0.2	100	33
4.3.3.2	Ναι	8	6	26	4	0.2	0.1	882	36,5
4.3.3.3	Ναι	10	10	26	4	0.15	0.1	63	31
4.3.3.4	Ναι	9	9	34	4	0.3	0.05	496	25,3
4.3.3.5	Ναι	9	9	34	4	0.15	0.05	398	30,8
4.3.3.6	Ναι	12	12	34	4	0.09	0.05	486	25

Πίνακας 4.9 : Πειράματα (παραμέτροι και τελικό μέσο απόλυτο σφάλμα) 1 μέχρι 6 με χρήση νευρωνικού δικτύου κατηγοριοποίησης στα δεδομένα διμηνίας Μαΐου - Ιουνίου



Γράφημα 4.9 : Ποσοστό επιτυχίας κατηγοριοποίησης δεδομένων σε κλάση κατανάλωσης για κάθε ένα από τα πειράματα 1 μέχρι 6 με χρήση νευρωνικού δικτύου κατηγοριοποίησης στα δεδομένα διμηνίας Μαίου - Ιουνίου

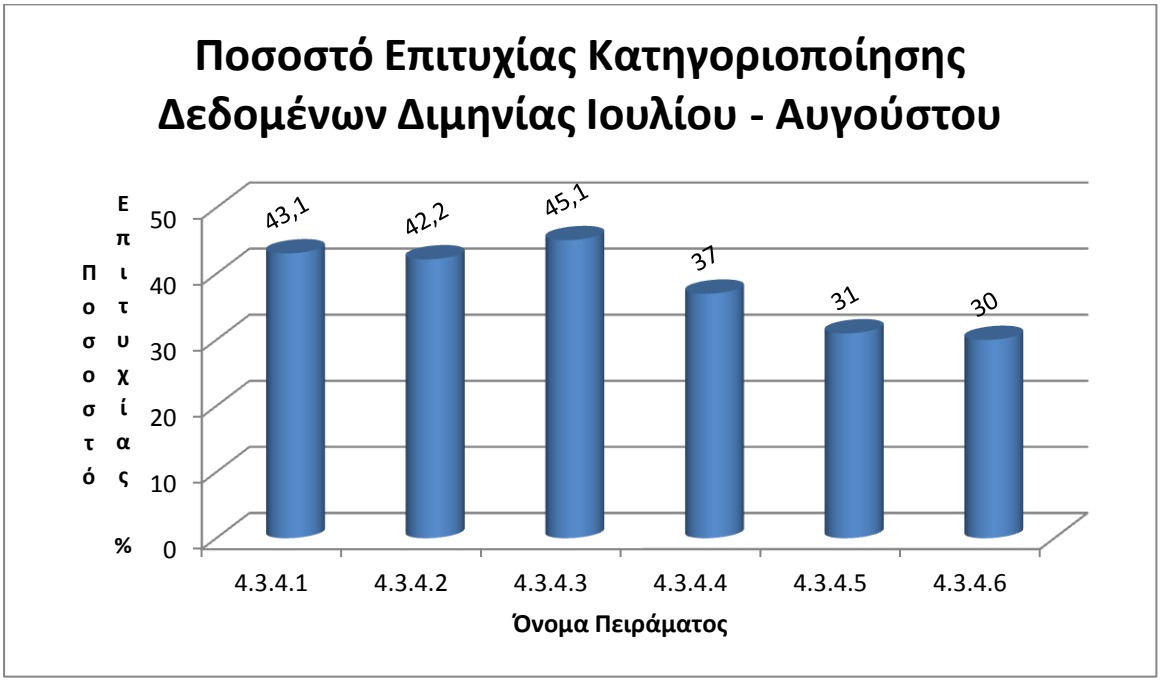
Στην διμηνία αυτή, ο μέση τιμή της συνολικής κατανάλωσης είναι 714 και η τυπική απόκλιση είναι 381. Μετά από την εκπαίδευση του αλγορίθμου, και από την εφαρμογή του σε *άγνωστα δεδομένα* το ποσοστό επιτυχίας με είσοδο το σύνολο ελέγχου το οποίο περιέχει την μεταβλητή FLAT, και τη διαγραφή των αδικαιολόγητα εξαιρετικά υψηλής και χαμηλής κατανάλωσης εγγραφών, είχε φθάσει το 36.5%. Σε αντίθεση, με την κατηγοριοποίηση των αριθμητικών μεταβλητών σε τρεις ομάδες την κάθε μία, είχαμε ποσοστό επιτυχίας 30.8. Έτσι μπορούμε να δούμε ότι το ποσοστό επυτιχίας στο δεύτερο σύνολο δεδομένων είναι γενικά χαμηλότερο από το πρώτο.

4.3.4 Διμηνία Ιουλίου - Αυγούστου

Ο.Π.	Δ.Α.Τ.	Ν.Π.Ε	Ν.Δ.Ε	Ν.ΕΙ.	Ν.ΕΞ.	Ρ.Μ.	ΟΡΜ.	ΕΠΑ.	Ποσοστό Επιτυχίας %
4.3.4.1	Ναι	8	6	26	4	0.1	0.5	173	43,1
4.3.4.2	Ναι	9	9	26	4	0.15	0.05	126	42,2
4.3.4.3	Ναι	10	10	26	4	0.4	0.5	45	45,1
4.3.4.4	Ναι	9	9	34	4	0.3	0.1	470	37

4.3.4.5	Ναι	9	9	34	4	0.15	0.1	496	31
4.3.4.6	Ναι	12	12	34	4	0.09	0.1	852	30

Πίνακας 4.10 : Πειράματα (παραμέτροι και τελικό μέσο απόλυτο σφάλμα) 1 μέχρι 6 με χρήση νευρωνικού δικτύου κατηγοριοποίησης στα δεδομένα διμηνίας Ιουλίου-Αυγούστου



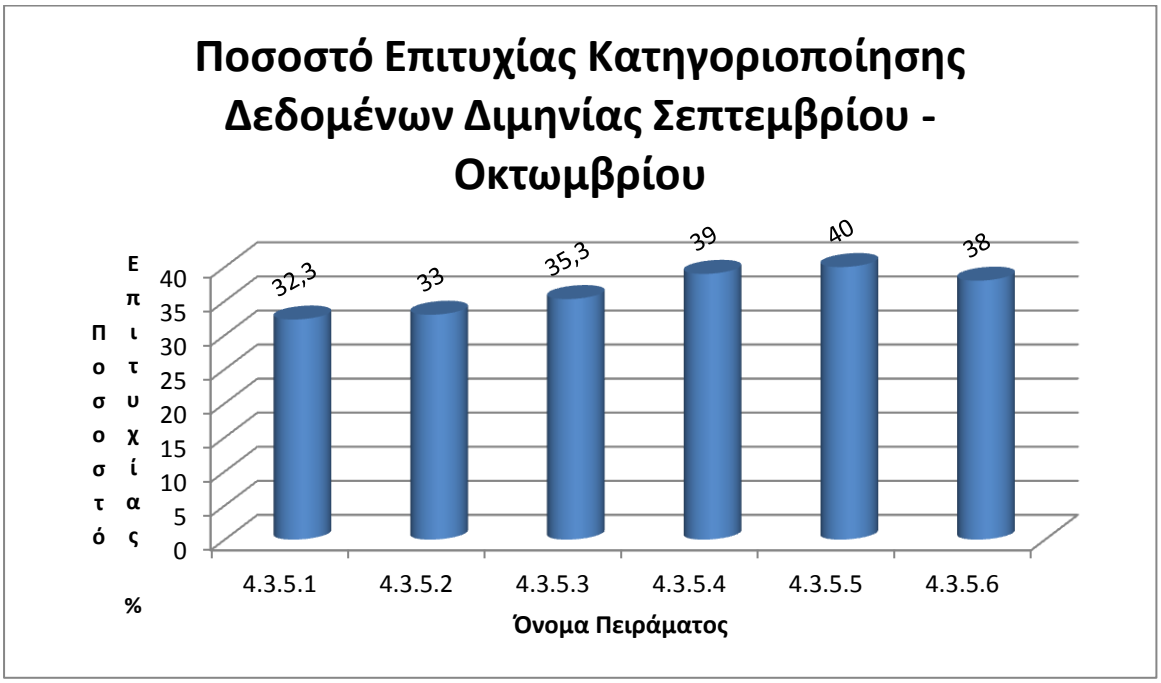
Γράφημα 4.10 : Ποσοστό επιτυχίας κατηγοριοποίησης δεδομένων σε κλάση κατανάλωσης για κάθε ένα από τα πειράματα 1 μέχρι 6 με χρήση νευρωνικού δικτύου κατηγοριοποίησης στα δεδομένα διμηνίας Ιουλίου - Αυγούστου

Στην διμηνία αυτή, ο μέση τιμή της συνολικής κατανάλωσης είναι 1149 και η τυπική απόκλιση είναι 741. Μετά από την εκπαίδευση του αλγορίθμου, και από την εφαρμογή του σε *άγνωστα δεδομένα* το ποσοστό επιτυχίας με είσοδο το σύνολο ελέγχου το οποίο περιέχει την μεταβλητή FLAT, και τη διαγραφή των αδικαιολόγητα εξαιρετικά υψηλής και χαμηλής κατανάλωσης εγγραφών, είχε φθάσει το 45.1%. Σε αντίθεση, με την κατηγοριοποίηση των αριθμητικών μεταβλητών σε τρεις ομάδες την κάθε μία, είχαμε ποσοστό επιτυχίας 37%. Έτσι μπορούμε να δούμε ότι το ποσοστό επυτιχίας στο δεύτερο σύνολο δεδομένων είναι γενικά χαμηλότερο από το πρώτο.

4.3.5 Διμηνία Σεπτεμβρίου - Οκτωβρίου

Ο.Π.	Δ.Α.Τ.	Ν.Π.Ε	Ν.Δ.Ε	Ν.ΕΙ.	Ν.ΕΞ.	Ρ.Μ.	ΟΡΜ.	ΕΠΑ.	Ποσοστό Επιτυχίας %
4.3.5.1	Ναι	8	6	26	4	0.3	0.5	39	32,3
4.3.5.2	Ναι	9	9	26	4	0.15	0.05	157	33
4.3.5.3	Ναι	10	10	26	4	0.4	0.05	71	35,3
4.3.5.4	Ναι	9	9	34	4	0.3	0.1	20	39
4.3.5.5	Ναι	9	9	34	4	0.15	0.1	47	40
4.3.5.6	Ναι	12	12	34	4	0.09	0.1	84	38

Πίνακας 4.11 : Πειράματα (παραμέτροι και τελικό μέσο απόλυτο σφάλμα) 1 μέχρι 6 με χρήση νευρωνικού δικτύου κατηγοριοποίησης στα δεδομένα διμηνίας Σεπτεμβρίου - Οκτωμβρίου



Γράφημα 4.11 : Ποσοστό επιτυχίας κατηγοριοποίησης δεδομένων σε κλάση κατανάλωσης για κάθε ένα από τα πειράματα 1 μέχρι 6 με χρήση νευρωνικού δικτύου κατηγοριοποίησης στα δεδομένα διμηνίας Σεπτεμβρίου - Οκτωμβρίου

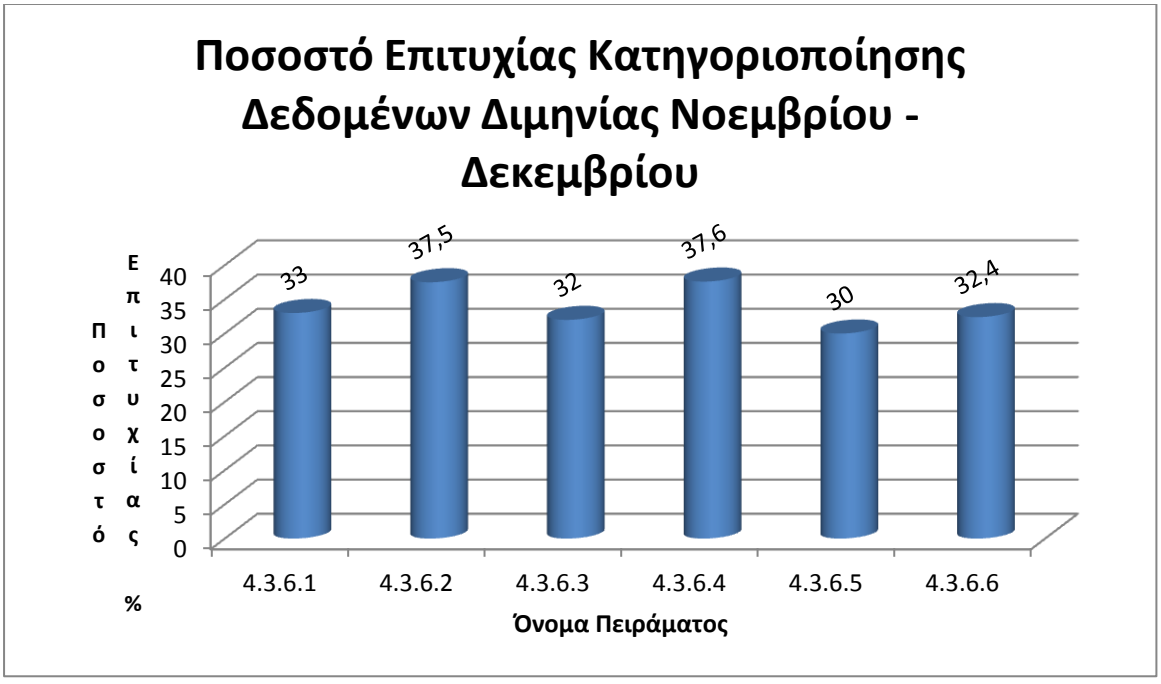
Στην διμηνία αυτή, ο μέση τιμή της συνολικής κατανάλωσης είναι 1156 και η τυπική απόκλιση είναι 725. Μετά από την εκπαίδευση του αλγορίθμου, και από την εφαρμογή του σε *άγνωστα δεδομένα* το ποσοστό επιτυχίας με είσοδο το σύνολο ελέγχου το οποίο περιέχει την μεταβλητή FLAT, και τη διαγραφή των αδικαιολόγητα εξαιρετικά υψηλής και χαμηλής κατανάλωσης εγγραφών, είχε φθάσει το 35.3%. Σε αντίθεση, με την κατηγοριοποίηση των

αριθμητικών μεταβλητών σε τρεις ομάδες την κάθε μία, είχαμε ποσοστό επιτυχίας 40%. Έτσι μπορούμε να δούμε ότι το ποσοστό επυτιχίας στο δεύτερο σύνολο δεδομένων είναι γενικά υψυλότερο από το πρώτο.

4.3.6 Διμηνία Νοεμβρίου – Δεκεμβρίου

Ο.Π.	Δ.Α.Τ.	Ν.Π.Ε	Ν.Δ.Ε	Ν.ΕΙ.	Ν.ΕΞ.	Ρ.Μ.	ΟΡΜ.	ΕΠΑ.	Ποσοστό Επιτυχίας %
4.3.6.1	Ναι	8	6	26	4	0.3	0.05	238	33
4.3.6.2	Ναι	9	9	26	4	0.15	0.05	46	37,5
4.3.6.3	Ναι	15	15	26	4	0.09	0.05	323	32
4.3.6.4	Ναι	9	9	34	4	0.3	0.1	10	37,6
4.3.6.5	Ναι	7	7	34	4	0.15	0.1	589	30
4.3.6.6	Ναι	12	12	34	4	0.09	0.15	1787	32,4

Πίνακας 4.12 : Πειράματα (παραμέτροι και τελικό μέσο απόλυτο σφάλμα) 1μέχρι 6 με χρήση νευρωνικού δικτύου κατηγοριοποίησης στα δεδομένα διμηνίας Νοεμβρίου - Δεκεμβρίου



Γράφημα 4.6 : Ποσοστό επιτυχίας κατηγοριοποίησης δεδομένων σε κλάση κατανάλωσης για κάθε ένα από τα πειράματα 1 μέχρι 6 με χρήση νευρωνικού δικτύου κατηγοριοποίησης στα δεδομένα διμηνίας Νοεμβρίου – Δεκεμβρίου

Στην διμηνία αυτή, ο μέση τιμή της συνολικής κατανάλωσης είναι 704 και η τυπική απόκλιση είναι 360. Μετά από την εκπαίδευση του αλγορίθμου, και από την εφαρμογή του σε *άγνωστα δεδομένα* το ποσοστό επιτυχίας με είσοδο το σύνολο ελέγχου το οποίο περιέχει την μεταβλητή FLAT, και τη διαγραφή των αδικαιολόγητα εξαιρετικά υψηλής και χαμηλής κατανάλωσης εγγραφών, είχε φθάσει το 37.5%. Σε αντίθεση, με την κατηγοριοποίηση των αριθμητικών μεταβλητών σε τρεις ομάδες την κάθε μία, είχαμε ποσοστό επιτυχίας 37,6. Έτσι μπορούμε να δούμε ότι το ποσοστό επυτιχίας στο δεύτερο σύνολο δεδομένων δεν έχει ιδιαίτερες διαφορές από το πρώτο.

Συμπεράσματα Πειράμάτων MLP Classification

Πριν αναφέρω τα συμπεράσματα μας, θα ήθελα να αναφέρω ότι το πιο πάνω ποσοστό επιτυχίας το οποίο υπάρχει στους παραπάνω πίνακες, είναι ο μέσος όρος 5 εκτελέσεων του πειράματος στα ίδια δεδομένα. Ο λόγος που τρέχαμε κάθε πείραμα 5 φορές, είναι για να παρατηρήσουμε ορθότερα τα αποτελέσματα των αλγορίθμων μας καθώς με μόνο μία εκτέλεση, θα μπορούσαμε να καταλήξουμε σε μη αντικατοπτριστικά δεδομένα ως προς τις παραμέτρους του αλγορίθμου και το δείγμα μας.

Εδώ μπορούμε να παράξουμε ως συμπέρασμα το γεγονός ότι οι αλγόριθμοι έχουν δεν έχουν αναπτύξει την δινατότητα να κατηγοριοποιούν. Οι τέσσερις κατηγορίές έχουν ισόποσες εγγραφές, και έτσι σε περίπτωση που απαντούσαμε τυχαία για κάποια άγνωστη εγγραφή, τότε θα είχαμε ποσοστό επιτυχίας γύρω στα 25%. Στα συγκεκριμένα πειράματα το ποσοστό επιτυχίας κυμαίνεται γύρω στα 30% με 40 %. Από αυτού διαφαίνεται ότι ο αλγόριθμος δεν κρίνει τυχαία, αλλά μαθαίνει λίγο, αλλά τα αποτελέσματα αυτά δεν θα μπορούσαμε να πούμε ότι μας ικανοποιούν, καθώς είναι σχετικά χαμηλά Επίσης μπορούμε να παρατηρήσουμε ότι στα πρώτα 3 πειράματα για κάθε μία από τις διμηνίες, στα οποία τα δεδομένα εισάγονταν ως μαζί με τις αριθμητικές τους τιμές είχαν διαφορετικά αποτελέσματα ως σύνολο από τα τρία πειράματα όπου οι αριθμητικές μεταβλητές είχαν μετατραπεί σε κατηγορίες. Σε γενικές γραμμές η κατηγοριοποίηση των αριθμητικών δεδομένων δεν είχε βοηθήσει ιδιαίτερα αλλά μάλλον το αντίθετο αφού τα αποτελέσματα στα πρώτα τρία πειράματα είναι ελαφρώς καλύτερα.

Κεφάλαιο 5

Πειράματα με Περισσότερους Αλγορίθμους Μηχανικής Μάθησης

5.1 Εισαγωγή

Για να είναι τελείως τεκμηριωμένα τα αποτελέσματά που θα προέκυπταν από την διπλωματική αυτή εργασία, θα έπρεπε να γίνουν αρκετά πειράματα, με διάφορους αλγορίθμους, και διαφορετικές παραμέτρους σε κάθε εκτέλεση, οι οποίες θα μας δήλωναν τις ικανότητες των αλγορίθμων για να παράξουν μία όσο πιο βέλτιστη λύση γίνεται στο πρόβλημά μας. Έγιναν πειράματα με *10 διαφορετικούς αλγορίθμους*, και στις 6 διμηνίες, έχοντας διαφορετικές παραμέτρους κάθε φορά. Κάθε αλγόριθμος έτρεχε 10 φορές σε κάθε διμηνία για κάθε σύνολο παραμέτρων του. Επίσης εφαρμόζαμε κάθε φορά διαφορετική

προεπεξεργασία στα δεδομένα μας, για να μελετήσουμε την μορφή την οποία έπρεπε να έχουν τα δεδομένα μας, ούτως ώστε να έχουμε τα καλύτερα δυνατά αποτελέσματα. Ο έλεγχος των δεδομένων γινόταν με την διαδικασία 10-folds cross-validation η οποία περιγράφεται στο υποκεφάλαιο 3.3. Τα πιο πάνω είχαν αποτέλεσμα να γίνουν συνολικά **περισσότερα των 100.000 πειραμάτων** μόνο με τους παραπάνω αλγορίθμους**.** Ο αριθμός αυτός είναι ίσως ικανός να δείξει το μέγεθος της μελέτης, την οποία είχαμε κάνει στο σύνολο των δεδομένων μας. Επίσης παρουσιάζουμε στο 5.4 αποτελέσματα πειραμάτων τα οποία είχαμε εκτελέσει με σύνολα δεδομένων διαφορετικού μεγέθους, για να μπορούμε να μελετήσουμε πως οι αλγόριθμοι μας βελτιώνονται ανάλογα με την αύξηση των δεδομένων.

5.2 Περιγραφή πειραμάτων

Πείραμα 1: Χρησιμοποιήσαμε Γκαουσιανές διαδικασίες. Είχαμε θέσει την παράμετρο του θορύβου στο 1, και είχαμε κανονικοποιήσει τα δεδομένα. Επίσης ως συνάρτηση πυρήνα, είχαμε χρησιμοποιήσει τον πυρήνα τύπου RBF με παράμετρο gamma ίση με 1.

Πείραμα 2: Χρησιμοποιήσαμε Γκαουσιανές διαδικασίες. Είχαμε θέσει την παράμετρο του θορύβου στο 1,1, και είχαμε κανονικοποιήσει τα δεδομένα. Επίσης ως συνάρτηση πυρήνα, είχαμε χρησιμοποιήσει τον πυρήνα τύπου RBF με παράμετρο gamma ίση με 1.

Πείραμα 3: Χρησιμοποιήσαμε Γκαουσιανές διαδικασίες. Είχαμε θέσει την παράμετρο του θορύβου στο 0,9, και είχαμε κανονικοποιήσει τα δεδομένα. Επίσης ως συνάρτηση πυρήνα, είχαμε χρησιμοποιήσει τον πυρήνα τύπου RBF με παράμετρο gamma ίση με 1.

Πείραμα 4: Χρησιμοποιήσαμε Γκαουσιανές διαδικασίες. Είχαμε θέσει την παράμετρο του θορύβου στο 0,9, και είχαμε κανονικοποιήσει τα δεδομένα. Επίσης ως συνάρτηση πυρήνα, είχαμε χρησιμοποιήσει τον πυρήνα τύπου RBF με παράμετρο gamma ίση με 0,9.

Πείραμα 5: Χρησιμοποιήσαμε Γκαουσιανές διαδικασίες. Είχαμε θέσει την παράμετρο του θορύβου στο 1, και είχαμε διακριτοποιείσει τα δεδομένα. Επίσης ως συνάρτηση πυρήνα, είχαμε χρησιμοποιήσει τον πυρήνα τύπου RBF με παράμετρο gamma ίση με 1.

Πείραμα 6: Χρησιμοποιήσαμε Ισοτονική Παλινδρόμηση.

Πείραμα 7: Χρησιμοποιήσαμε Ελάχιστο Μέσο Τετραγώνων. Είχαμε θέσει την παράμετρο των τυχαίων δειγμάτων που χρειάζονται για να δημιουργηθούν οι συναρτήσεις ελαχίστου μέσου τετραγώνου στο 4.

Πείραμα 8: Χρησιμοποιήσαμε Ελάχιστο Μέσο Τετραγώνων. Είχαμε θέσει την παράμετρο των τυχαίων δειγμάτων που χρειάζονται για να δημιουργηθούν οι συναρτήσεις ελαχίστου μέσου τετραγώνου στο 5.

Πείραμα 9: Χρησιμοποιήσαμε Ελάχιστο Μέσο Τετραγώνων. Είχαμε θέσει την παράμετρο των τυχαίων δειγμάτων που χρειάζονται για να δημιουργηθούν οι συναρτήσεις ελαχίστου μέσου τετραγώνου στο 6.

Πείραμα 10: Χρησιμοποιήσαμε Ελάχιστο Μέσο Τετραγώνων. Είχαμε θέσει την παράμετρο των τυχαίων δειγμάτων που χρειάζονται για να δημιουργηθούν οι συναρτήσεις ελαχίστου μέσου τετραγώνου στο 7.

Πείραμα 11: Χρησιμοποιήσαμε Γραμμική Παλινδρόμηση. Είχαμε ακόμη χρησιμοοιήσει επιλογή χαρακτηριστικών (attribute selection) με την μέθοδο M5. Είχαμε θέσει την παράμετρο της κορυφογραμμής στο 1.0E-8.

Πείραμα 12: Χρησιμοποιήσαμε Γραμμική Παλινδρόμηση. Δεν είχαμε χρησιμοοιήσει επιλογή χαρακτηριστικών και έτσι όλα τα χαρακτηριστικά εισήχθησαν στο μοντέλλο. Είχαμε θέσει την παράμετρο της κορυφογραμμής στο 1.0E-8.

Πείραμα 13: Χρησιμοποιήσαμε Γραμμική Παλινδρόμηση. Είχαμε ακόμη χρησιμοοιήσει επιλογή χαρακτηριστικών (attribute selection) με την άπλυστη μέθοδο. Είχαμε θέσει την παράμετρο της κορυφογραμμής στο 1.0E-8.

Πείραμα 14: Χρησιμοποιήσαμε Γραμμική Παλινδρόμηση. Είχαμε ακόμη χρησιμοοιήσει επιλογή χαρακτηριστικών (attribute selection) με την μέθοδο M5. Είχαμε θέσει την παράμετρο της κορυφογραμμής στο 2.0E-8.

Πείραμα 15: Χρησιμοποιήσαμε Γραμμική Παλινδρόμηση. Είχαμε ακόμη χρησιμοοιήσει επιλογή χαρακτηριστικών (attribute selection) με την μέθοδο M5. Είχαμε θέσει την παράμετρο της κορυφογραμμής στο 2.1E-7.

Πείραμα 16: Χρησιμοποιήσαμε Βηματική Παλινδρόμηση. Είχαμε θέσει τον εκτιμητή (estimator) να είναι ο Εμπειρικός Bayes.

Πείραμα 17: Χρησιμοποιήσαμε Βηματική Παλινδρόμηση. Είχαμε θέσει τον εκτιμητή (estimator) να είναι ο Εμφωλευμένος Επιλογέας Μοντέλου (Nested Model Selector).

Πείραμα 18: Χρησιμοποιήσαμε Βηματική Παλινδρόμηση. Είχαμε θέσει τον εκτιμητή (estimator) να είναι ο Επιλογέας Υποσυνόλου(subset selector).

Πείραμα 19: Χρησιμοποιήσαμε Βηματική Παλινδρόμηση. Είχαμε θέσει τον εκτιμητή (estimator) να είναι ο PACE2.

Πείραμα 20: Χρησιμοποιήσαμε Βηματική Παλινδρόμηση. Είχαμε θέσει τον εκτιμητή (estimator) να είναι ο PACE4.

Πείραμα 21: Χρησιμοποιήσαμε Βηματική Παλινδρόμηση. Είχαμε θέσει τον εκτιμητή (estimator) να είναι ο PACE6.

Πείραμα 22: Χρησιμοποιήσαμε Δίκτυο συναρτήσεων αξονικών βάσεων (RBF network). Είχαμε θέσει τον αριθμό των ομάδων (clusters) να είναι ίσο με 4, την κορυφογραμμή να είναι ίση με 1,0E-8 και την ελάχιστη τυπική απόκλιση να είναι 0.1 .

Πείραμα 23: Χρησιμοποιήσαμε Δίκτυο συναρτήσεων αξονικών βάσεων (RBF network). Είχαμε θέσει τον αριθμό των ομάδων (clusters) να είναι ίσο με 5, την κορυφογραμμή να είναι ίση με 1,0E-8 και την ελάχιστη τυπική απόκλιση να είναι 0.2 .

Πείραμα 24: Χρησιμοποιήσαμε Δίκτυο συναρτήσεων αξονικών βάσεων (RBF network). Είχαμε θέσει τον αριθμό των ομάδων (clusters) να είναι ίσο με 6, την κορυφογραμμή να είναι ίση με 1,0E-8 και την ελάχιστη τυπική απόκλιση να είναι 0.01 .

Πείραμα 25: Χρησιμοποιήσαμε Δίκτυο συναρτήσεων αξονικών βάσεων (RBF network). Είχαμε θέσει τον αριθμό των ομάδων (clusters) να είναι ίσο με 6, την κορυφογραμμή να είναι ίση με 1,0E-8 και την ελάχιστη τυπική απόκλιση να είναι 0.3 .

Πείραμα 26: Χρησιμοποιήσαμε Δίκτυο συναρτήσεων αξονικών βάσεων (RBF network). Είχαμε θέσει τον αριθμό των ομάδων (clusters) να είναι ίσο με 3, την κορυφογραμμή να είναι ίση με 1,0E-8 και την ελάχιστη τυπική απόκλιση να είναι 0.1 .

Πείραμα 27: Χρησιμοποιήσαμε Δίκτυο συναρτήσεων αξονικών βάσεων (RBF network). Είχαμε θέσει τον αριθμό των ομάδων (clusters) να είναι ίσο με 2, την κορυφογραμμή να είναι ίση με 1,0E-8 και την ελάχιστη τυπική απόκλιση να είναι 0.1 .

Πείραμα 28: Χρησιμοποιήσαμε Απλή Γραμμική Παλινδρόμηση.

Πείραμα 29: Χρησιμοποιήσαμε Sequential Minimal Optimization Παλινδρόμηση (SMO regression). Είχαμε θέσει την παράμετρο πολυπλοκότητας c ίση με 1, και κανονικοποιούσαμε τα δεδομένα πριν εκτελεστεί ο αλγόριθμος. Επίσης είχαμε θέσει πολυωνυμικό πυρήνα με εκθέτη 1, και βελτιστοποιητή Regression SMO Improved με παράμετρο epsilon ο οποίος χρησιμοποιείται στην μη ευαισθητοποιημένη loss function του e ίσο με 0,001. Ο παράγοντας ανεκτικότητας, ο οποίος χρησιμοποιείται, για να ελέγχεται το κριτήριο τερματισμού του αλγορίθμου είναι στα 0,001.

Πείραμα 30: Χρησιμοποιήσαμε Sequential Minimal Optimization Παλινδρόμηση (SMO regression). Είχαμε θέσει την παράμετρο πολυπλοκότητας c ίση με 1, και κανονικοποιούσαμε τα δεδομένα πριν εκτελεστεί ο αλγόριθμος. Επίσης είχαμε θέσει πολυωνυμικό πυρήνα με εκθέτη 1,2, και βελτιστοποιητή Regression SMO Improved με παράμετρο epsilon ο οποίος χρησιμοποιείται στην μη ευαισθητοποιημένη loss function του e ίσο με 0,001. Ο παράγοντας ανεκτικότητας, ο οποίος χρησιμοποιείται, για να ελέγχεται το κριτήριο τερματισμού του αλγορίθμου είναι στα 0,001.

Πείραμα 31: Χρησιμοποιήσαμε Sequential Minimal Optimization Παλινδρόμηση (SMO regression). Είχαμε θέσει την παράμετρο πολυπλοκότητας c ίση με 1, και κανονικοποιούσαμε τα δεδομένα πριν εκτελεστεί ο αλγόριθμος. Επίσης είχαμε θέσει πολυωνυμικό πυρήνα με εκθέτη 1, και βελτιστοποιητή Regression SMO Improved με παράμετρο epsilon ο οποίος χρησιμοποιείται στην μη ευαισθητοποιημένη loss function του e ίσο με 0,005. Ο παράγοντας ανεκτικότητας, ο οποίος χρησιμοποιείται, για να ελέγχεται το κριτήριο τερματισμού του αλγορίθμου είναι στα 0,001.

Πείραμα 32: Χρησιμοποιήσαμε Sequential Minimal Optimization Παλινδρόμηση (SMO regression). Είχαμε θέσει την παράμετρο πολυπλοκότητας c ίση με 1, και

κανονικοποιούσαμε τα δεδομένα πριν εκτελεστεί ο αλγόριθμος. Επίσης είχαμε θέσει πολυωνυμικό πυρήνα με εκθέτη 1, και βελτιστοποιητή Regression SMO Improved με παράμετρο epsilon ο οποίος χρησιμοποιείται στην μη ευαισθητοποιημένη loss function του e ίσο με 0,005. Ο παράγοντας ανεκτικότητας, ο οποίος χρησιμοποιείται, για να ελέγχεται το κριτήριο τερματισμού του αλγορίθμου είναι στα 0,009.

Πείραμα 33: Χρησιμοποιήσαμε ταξινομητή PLS Παλινδρόμηση με αριθμό των στοιχείων για να υπολογιστούν ίσο με 20.

5.3 Αποτελέσματα και συμπεράσματα των εκτελέσεων των πειραμάτων

Τα δεδομένα εισόδου στους αλγορίθμους, για κάθε μια από τις διμηνίες ήταν :

1. Τα δεδομένα της διμηνίας ως έχουν, χωρίς καμία προεπεξεργασία
2. Τα δεδομένα της διμηνίας, με διαγραφή του θορύβου (διαγραφή αδικαιολόγητα εξαιρετικά χαμηλών ή υψηλών ηλεκτρικών καταναλώσεων)
3. Τα δεδομένα της διμηνίας, με διαγραφή του θορύβου (διαγραφή αδικαιολόγητα εξαιρετικά χαμηλών ή υψηλών ηλεκτρικών καταναλώσεων) καθώς επίσης τα *αριθμητικά* χαρακτηριστικά του συνόλου δεδομένων, δηλαδή ο αριθμός δωματίων, ο αριθμός ενηλίκων και ανηλίκων κατοίκων και το μέγεθος του σπιτιού, διαχωρίζεται το καθένα σε τρεις κατηγορίες, και εξαλείφονται εντελώς τα αριθμητικά χαρακτηριστικά.

Παρουσιάζουμε για κάθε διμηνία μία γραφική παράσταση. Οι παραστάσεις αυτές έχουν τρεις διαφορετικές γραμμές, η κάθε μία δηλώνει την είσοδο ενός από τα πιο πάνω σύνολα δεδομένων για κάθε μία από τις 33 διαφορετικές εκτελέσεις των δέκα αλγορίθμων μας. Η κάθε μία από τις τιμές οι οποίες παρουσιάζονται πιο κάτω είναι *ο μέσος όρος* των δέκα εκτελέσεων του συγκεκριμένου πειράματος στο συγκεκριμένο σύνολο δεδομένων της διμηνίας.

Οι πιο κάτω γραφικές παραστάσεις, περιέχουν κάθετες γραμμές οι οποίες αναπαριστούν το τελευταίο πείραμα ενός συγκεκριμένου αλγορίθμου ο οποίος έχει χρησιμοποιηθεί.

Θα επαναλαμβάνουμε τους αλγορίθμους οι οποίοι έχουν εκτελεστεί κάτω από κάθε γραφική παράσταση, για να μπορούμε να παρατηρούμε ευκολότερα τις αποδόσεις των αλγορίθμων.

Πιο συγκεκριμένα:

Στα πειράματα 1 μέχρι 5, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στο πείραμα 6, είχε εκτελεστεί Ισοτονική Παλινδρόμηση.

Στα πειράματα 7 μέχρι 10, είχε εκτελεστεί ο αλγόριθμος του Ελαχίστου Μέσου Τετραγώνων.

Στα πειράματα 11 μέχρι 15, είχε εκτελεστεί Γραμμική Παλινδρόμηση.

Στα πειράματα 16 μέχρι 21, είχε εκτελεστεί Βηματική Παλινδρόμηση.

Στα πειράματα 22 μέχρι 27, είχε εκτελεστεί Δίκτυο RBF.

Στο πείραμα 28, είχε εκτελεστεί Απλή Γραμμική Παλινδρόμηση.

Στα πειράματα 29 μέχρι 32, είχε εκτελεστεί Παλινδρόμηση SMO.

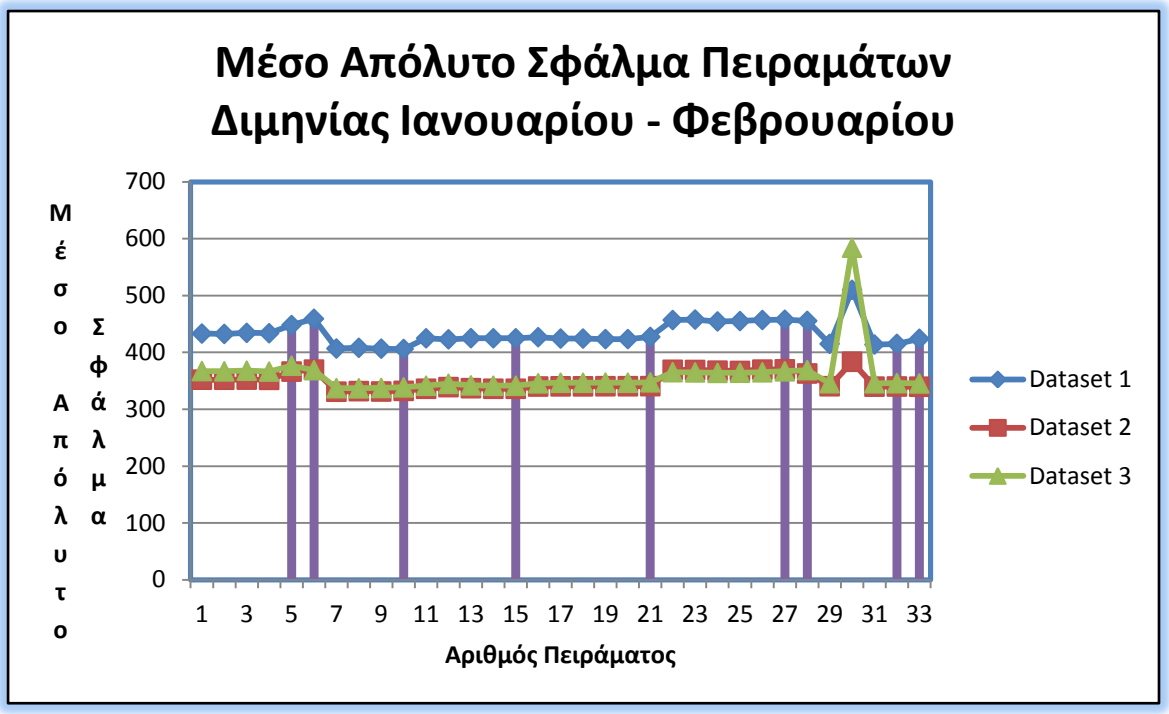
Στο πείραμα 33, είχε χρησιμοποιηθεί ο ταξινομητής PLS

5.3.1 Διμηνία Ιανουαρίου – Φεβρουαρίου

Μέσο απόλυτο σφάλμα διμηνίας Ιανουαρίου – Φεβρουαρίου			
Α/Α Συνόλου δεδομένων διμηνίας / Α/Α εκτέλεσης πειράματος	Σύνολο δεδομένων 1	Σύνολο δεδομένων 2	Σύνολο δεδομένων 3
1	433,24	352,12	367,3
2	432,59	352,08	366,79
3	434,46	352,27	367,93
4	433,83	351,85	366,41
5	448,19	366,37	376,72
6	459,24	369,79	368,93
7	406,96	330,97	337,26
8	408,18	332,01	336,96
9	406,64	331,31	337,6
10	406,15	332,43	338,64
11	424,9	336,11	341,53
12	423,07	339,05	344,89
13	425,17	337,15	342,08
14	424,9	336,11	341,53
15	424,9	336,11	341,53

16	426,85	340,05	345,76
17	424,43	340,39	346,74
18	424,43	340,39	346,74
19	423,49	340,6	346,89
20	423,52	340,57	346,78
21	427,35	340,71	347,57
22	457,03	369,45	365,31
23	457,93	369,1	364,92
24	454,56	367,96	364,09
25	455,23	367,48	364,33
26	457,12	369,77	365,15
27	457,52	370,24	366,88
28	455,57	363,01	368,93
29	415,17	340,6	345,92
30	510,18	383,64	584,31
31	413,98	339,89	345,88
32	415,17	340,11	346,03
33	424,1	339,74	345,56
Ελάχιστο μέσο απόλυτο σφάλμα	**406,15**	**330,97**	**336,96**
Ελάχιστο μέσο απόλυτο σφάλμα διμηνίας		**330,97**	

Πίνακας 5.1 – Μέσο απόλυτο σφάλμα για την διμηνία Ιανουαρίου - Φεβρουαρίου των 33 πειραμάτων για τα σύνολα δεδομένων 1,2 και 3.



Γράφημα 5.1 – Γραφική αναπαράσταση μέσου απολύτου σφάλματος για την διμηνία Ιανουαρίου - Φεβρουαρίου των 33 πειραμάτων για τα σύνολα δεδομένων 1,2 και 3

Για να κατανοήσουμε καλύτερα την γραφική παράσταση, παραθέτω σε συντομία τους αλγόριθμους οι οποίοι είχαν εκτελεστεί σε κάθε πείραμα, οι οποίοι έχουν ως εξής:

Στα πειράματα 1 μέχρι 5, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στο πείραμα 6, είχε εκτελεστεί Ισοτονική Παλινδρόμηση.

Στα πειράματα 7 μέχρι 10, είχε εκτελεστεί ο αλγόριθμος του Ελαχίστου Μέσου Τετραγώνων.

Στα πειράματα 11 μέχρι 15, είχε εκτελεστεί Γραμμική Παλινδρόμηση.

Στα πειράματα 16 μέχρι 21, είχε εκτελεστεί Βηματική Παλινδρόμηση.

Στα πειράματα 22 μέχρι 27, είχε εκτελεστεί Δίκτυο RBF.

Στο πείραμα 28, είχε εκτελεστεί Απλή Γραμμική Παλινδρόμηση.

Στα πειράματα 29 μέχρι 32, είχε εκτελεστεί Παλινδρόμηση SMO.

Στο πείραμα 33, είχε χρησιμοποιηθεί ο ταξινομητής PLS

Ανάλυση Αποτελεσμάτων: Μπορούμε να παρατηρήσουμε από την γραφική παράσταση το μέγεθος του μέσου απόλυτου σφάλματος, ανά εκτέλεση. Μπορούμε επίσης να παρατηρήσουμε διάφορα επίπεδα σφάλματος. Είναι σχεδόν ξεκάθαρο, ότι ανάλογα με τον αλγόριθμο ο οποίος τρέχει, δημιουργούνται τα επίπεδα αυτά, καθώς επίσης ότι τα δεδομένα του πρώτου συνόλου δεδομένων, έχουν περίπου γύρω στα 70 χειρότερο μέσο απόλυτο σφάλμα από τα υπόλοιπα δύο σύνολα δεδομένων. Μπορούμε να παρατηρήσουμε ότι στις πρώτες πέντε εκτελέσεις που εκτελείται ο αλγόριθμος των Γκαουσιανών Διαδικασιών, το

σφάλμα στο πρώτο σύνολο δεδομένων κυμαίνεται γύρω στα 435, ενώ γύρω στα 352 και 367 στο δεύτερο και τρίτο σύνολο δεδομένων αντίστοιχα. Στην εκτέλεση 6, όπου εκτελείται ο αλγόριθμος isotonic regression, παρατηρούμε παρόμοια αποτελέσματα, ενώ στις εκτελέσεις 7-10 όπου τρέχει ο αλγόριθμος του ελαχίστου τετραγωνικού μέσου, παρατηρούμε ότι λαμβάνουμε τα καλύτερα αποτελέσματα. Πιο συγκεκριμένα τα αποτελέσματα αυτά κυμαίνονται γύρω στα 407 στο πρώτο σύνολο δεδομένων, γύρω στα 331 στο δεύτερο σύνολο δεδομένων και γύρω στα 337 στο τρίτο σύνολο δεδομένων. Στις επαναλήψεις από 11 μέχρι και 15 εκτελείται ο αλγόριθμος γραμμικής παλινδρόμησης, ο οποίος και πάλι μας δίνει αρκετά καλά αποτελέσματα: 424 στο πρώτο σύνολο δεδομένων, 336 στο δεύτερο και 341 στο τρίτο. Στη συνέχεια οι εκτελέσεις της βηματικής παλινδρόμησης, έχουν ικανοποιητικά αποτελέσματα, αφού έχουν γύρω στα 424 μέσο απόλυτο σφάλμα στο πρώτο σύνολο δεδομένων, 340 στο δεύτερο και 346 στο τρίτο. Ακολουθούν από την εκτέλεση 27 μέχρι και την 32, τα δίκτυα RBF, τα οποία έχουν υψηλότερο μέσο σφάλμα από τους υπόλοιπους αλγορίθμους που θα μπορούσαν να θεωρηθούν ότι έχουν παράξει ικανοποιητικά αποτελέσματα. Επίσης μπορούμε να παρατηρήσουμε στη συνέχεια τα αποτελέσματα του αλγορίθμου παλινδρόμηση SMO τα οποία κυμαίνονται στα περίπου 414, 340 και 345 στα τρία δεδομένα εισόδου αντίστοιχα. Στο τέλος μπορούμε να παρατηρήσουμε τον ταξινομητή PLS ο οποίος έχει παρόμοια αποτελέσματα.

Επίσης, θα παρατηρήσουμε, ότι στο πείραμα 30 , όπου εκτελείται ο αλγόριθμος παλινδρόμηση SOM, υπάρχει μία κορυφή στην γραφική παράσταση. Παρόλο που τόσο το πείραμα 29, όσο και το πείραμα 31, είναι επίσης παλινδρόμηση SOM, έχουν σαφώς καλύτερα αποτελέσματα. Η διαφορά της συγκεκριμένης εκτέλεσης σε σχέση με τις άλλες, έγκειται στο γεγονός ότι είχαμε θέσει τον εκθέτη του πολυωνυμικού πυρήνα ίσο με 1,2 σε αντίθεση με τις υπόλοιπες εκτελέσεις του αλγορίθμου όπου ήταν 1. Έχουμε κυρίως αφήσει τα αποτελέσματα της συγκεκριμένης εκτέλεσης του αλγορίθμου στη γραφική παράσταση, για να μπορούμε να κατανοήσουμε καλύτερα το μέγεθος με το οποίο μία παράμετρος κάποιου αλγορίθμου, μπορεί να επηρεάζει δραστικά τις ικανότητες του να παράγει σωστά αποτελέσματα.

Γενικά μπορούμε να παρατηρήσουμε ότι, υπάρχουν καλύτερα αποτελέσματα στο σύνολο δεδομένων 2 από ότι στο τρία, καθώς επίσης ότι το σύνολο δεδομένων 1 δεν έχει τόσο καλά αποτελέσματα όσο τα υπόλοιπα δύο σύνολα. Επίσης μπορούμε να δούμε ότι τα καλύτερα αποτελέσματα έρχονται από τον αλγόριθμο ελάχιστου τετραγωνικού μέσου, ακολουθούμενου από την γραμμική παλινδρόμηση. Πιο συγκεκριμένα, όσον αφορά την διμηνία αυτή, έχουμε πάρει σαν ελάχιστο σφάλμα στο πρώτο σύνολο δεδομένων τα 406,15,

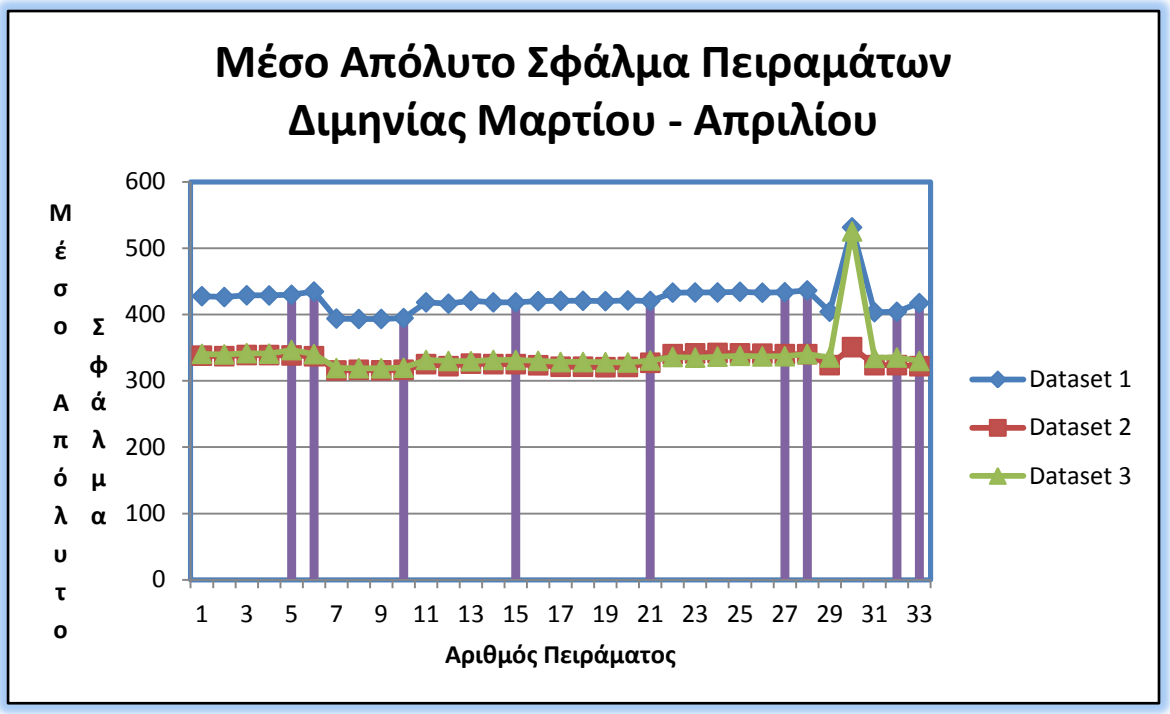
στο δεύτερο 330,97 καθώς στο τρίτο σύνολο 336,96. Έτσι το σύνολο που θα θέλαμε να χρησιμοποιήσουμε σαν είσοδο είναι το **δεύτερο σύνολο δεδομένων**, ενώ ο αλγόριθμος ο οποίος θα επιθυμούσαμε να χρησιμοποιήσουμε είναι ο **αλγόριθμος του ελαχίστου τετραγωνικού μέσου**. Έτσι έχουμε μέσο απόλυτο σφάλμα ***330,97***. Υπενθυμίζεται ότι στη διμηνία αυτή η **τυπική απόκλιση είναι 645,22**. Συνεπώς με την χρήση του αλγορίθμου αυτού, το σφάλμα είναι *το μισό της τυπικής απόκλισης*, πράγμα που σημαίνει ότι υπάρχει μάθηση μέσω του αλγορίθμου, και υπάρχουν ίσως όχι άριστα αποτελέσματα, αλλά αρκετά ικανοποιητικά

5.3.2 Διμηνία Μαρτίου - Απριλίου

Μέσο απόλυτο σφάλμα διμηνίας Μαρτίου – Απριλίου			
Α/Α Συνόλου δεδομένων διμηνίας / Α/Α εκτέλεσης πειράματος	Σύνολο δεδομένων 1	Σύνολο δεδομένων 2	Σύνολο δεδομένων 3
1	427,67	338,01	340,73
2	426,42	337,28	339,93
3	429,05	339,01	341,75
4	428,83	338,9	341,05
5	429,76	338,34	346,38
6	434,69	337,17	340,41
7	393,93	315,68	319,31
8	393,28	316,65	319,08
9	393,39	315,91	318,97
10	394,62	316,76	319,77
11	418,4	325,22	331,61
12	416,38	322,19	329,98
13	420,71	325,99	329,68
14	418,4	325,22	331,61
15	418,4	325,22	331,61
16	420,16	323,54	330,15
17	420,69	321,33	328,42
18	420,69	321,33	328,42

19	420,22	320,63	328,17
20	421,34	321	327,73
21	420,54	327,29	330,76
22	433,08	340,26	336
23	433,29	341,47	335,01
24	433,31	342,11	336,49
25	434,38	341,5	337,83
26	432,94	340,99	336,9
27	433,62	340,31	337,16
28	436,39	340,25	340,41
29	404,19	323,75	335,24
30	531,48	350,86	525,33
31	403,59	323,66	334,69
32	404,16	323,76	335,27
33	417,24	322,19	329,97
Ελάχιστο μέσο απόλυτο σφάλμα	**393,28**	**315,68**	**318,97**
Ελάχιστο μέσο απόλυτο σφάλμα διμηνίας		**315,68**	

Πίνακας 5.2 – Μέσο απόλυτο σφάλμα για την διμηνία Μαρτίου – Απριλίου των 33 πειραμάτων για τα σύνολα δεδομένων 1,2 και 3.



Γράφημα 5.2 – Γραφική αναπαράσταση μέσου απολύτου σφάλματος για την διμηνία Μαρτίου - Απριλίου των 33 πειραμάτων για τα σύνολα δεδομένων 1,2 και 3

Για να κατανοήσουμε καλύτερα την γραφική παράσταση, παραθέτω σε συντομία τους αλγόριθμους οι οποίοι είχαν εκτελεστεί σε κάθε πείραμα, οι οποίοι έχουν ως εξής:

Στα πειράματα 1 μέχρι 5, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στο πείραμα 6, είχε εκτελεστεί Ισοτονική Παλινδρόμηση.

Στα πειράματα 7 μέχρι 10, είχε εκτελεστεί ο αλγόριθμος του Ελαχίστου Μέσου Τετραγώνων.

Στα πειράματα 11 μέχρι 15, είχε εκτελεστεί Γραμμική Παλινδρόμηση.

Στα πειράματα 16 μέχρι 21, είχε εκτελεστεί Βηματική Παλινδρόμηση.

Στα πειράματα 22 μέχρι 27, είχε εκτελεστεί Δίκτυο RBF.

Στο πείραμα 28, είχε εκτελεστεί Απλή Γραμμική Παλινδρόμηση.

Στα πειράματα 29 μέχρι 32, είχε εκτελεστεί Παλινδρόμηση SMO.

Στο πείραμα 33, είχε χρησιμοποιηθεί ο ταξινομητής PLS

Ανάλυση Αποτελεσμάτων: Μπορούμε να παρατηρήσουμε από την γραφική παράσταση το μέγεθος του μέσου απόλυτου σφάλματος, ανά εκτέλεση. Σύμφωνα με τον αλγόριθμο τον οποίος τρέχει, παρατηρούνται διαφορετικά επίπεδα σφάλματος, καθώς επίσης ότι τα δεδομένα του πρώτου συνόλου δεδομένων, έχουν περίπου γύρω στα 70 χειρότερο μέσο απόλυτο σφάλμα από τα υπόλοιπα δύο σύνολα δεδομένων. Μπορούμε να παρατηρήσουμε ότι στις πρώτες πέντε εκτελέσεις που εκτελείται ο αλγόριθμος των Γκαουσιανών

Διαδικασιών, το σφάλμα στο πρώτο σύνολο δεδομένων κυμαίνεται γύρω στα 428, ενώ γύρω στα 338 και 340 στο δεύτερο και τρίτο σύνολο δεδομένων αντίστοιχα. Στην εκτέλεση 6, όπου εκτελείται ο αλγόριθμος isotonic regression, παρατηρούμε παρόμοια αποτελέσματα, ενώ στις εκτελέσεις 7-10 όπου τρέχει ο αλγόριθμος του ελαχίστου τετραγωνικού μέσου, παρατηρούμε ότι λαμβάνουμε τα καλύτερα αποτελέσματα. Πιο συγκεκριμένα τα αποτελέσματα αυτά κυμαίνονται γύρω στα 393 στο πρώτο σύνολο δεδομένων, γύρω στα 316 στο δεύτερο σύνολο δεδομένων και γύρω στα 319 στο τρίτο σύνολο δεδομένων. Στις επαναλήψεις από 11 μέχρι και 15 εκτελείται ο αλγόριθμος γραμμικής παλινδρόμησης, ο οποίος και πάλι μας δίνει αρκετά καλά αποτελέσματα: 418 στο πρώτο σύνολο δεδομένων, 325 στο δεύτερο και 330 στο τρίτο. Στη συνέχεια οι εκτελέσεις της βηματικής παλινδρόμησης, έχουν ικανοποιητικά αποτελέσματα, αφού έχουν γύρω στα 420 μέσο απόλυτο σφάλμα στο πρώτο σύνολο δεδομένων, 321 στο δεύτερο και 329 στο τρίτο. Ακολουθούν από την εκτέλεση 27 μέχρι και την 32, τα δίκτυα RBF, τα οποία έχουν υψηλότερο μέσο σφάλμα από τους υπόλοιπους αλγορίθμους που θα μπορούσαν να θεωρηθούν ότι έχουν παράξει ικανοποιητικά αποτελέσματα. Επίσης μπορούμε να παρατηρήσουμε στη συνέχεια τα αποτελέσματα του αλγορίθμου παλινδρόμηση SMO τα οποία κυμαίνονται στα περίπου 403, 323 και 335 στα τρία δεδομένα εισόδου αντίστοιχα. Στο τέλος μπορούμε να παρατηρήσουμε τον ταξινομητή PLS ο οποίος έχει παρόμοια αποτελέσματα.

Επίσης, θα παρατηρήσουμε, ότι στο πείραμα 30 , όπου εκτελείται ο αλγόριθμος παλινδρόμηση SOM, υπάρχει μία κορυφή στην γραφική παράσταση. Παρόλο που τόσο το πείραμα 29, όσο και το πείραμα 31, είναι επίσης παλινδρόμηση SOM, έχουν σαφώς καλύτερα αποτελέσματα. Η διαφορά της συγκεκριμένης εκτέλεσης σε σχέση με τις άλλες, έγκειται στο γεγονός ότι είχαμε θέσει τον εκθέτη του πολυωνυμικού πυρήνα ίσο με 1,2 σε αντίθεση με τις υπόλοιπες εκτελέσεις του αλγορίθμου όπου ήταν 1. Έχουμε κυρίως αφήσει τα αποτελέσματα της συγκεκριμένης εκτέλεσης του αλγορίθμου στη γραφική παράσταση, για να μπορούμε να κατανοήσουμε καλύτερα το μέγεθος με το οποίο μία παράμετρος κάποιου αλγορίθμου, μπορεί να επηρεάζει δραστικά τις ικανότητες του να παράγει σωστά αποτελέσματα.

Γενικά μπορούμε να παρατηρήσουμε ότι, υπάρχουν καλύτερα αποτελέσματα στο σύνολο δεδομένων 2 από ότι στο τρία, καθώς επίσης ότι το σύνολο δεδομένων 1 δεν έχει τόσο καλά αποτελέσματα όσο τα υπόλοιπα δύο σύνολα. Επίσης μπορούμε να δούμε ότι τα καλύτερα αποτελέσματα έρχονται από τον αλγόριθμο ελάχιστου τετραγωνικού μέσου, ακολουθούμενου από την γραμμική παλινδρόμηση. Πιο συγκεκριμένα, όσον αφορά την

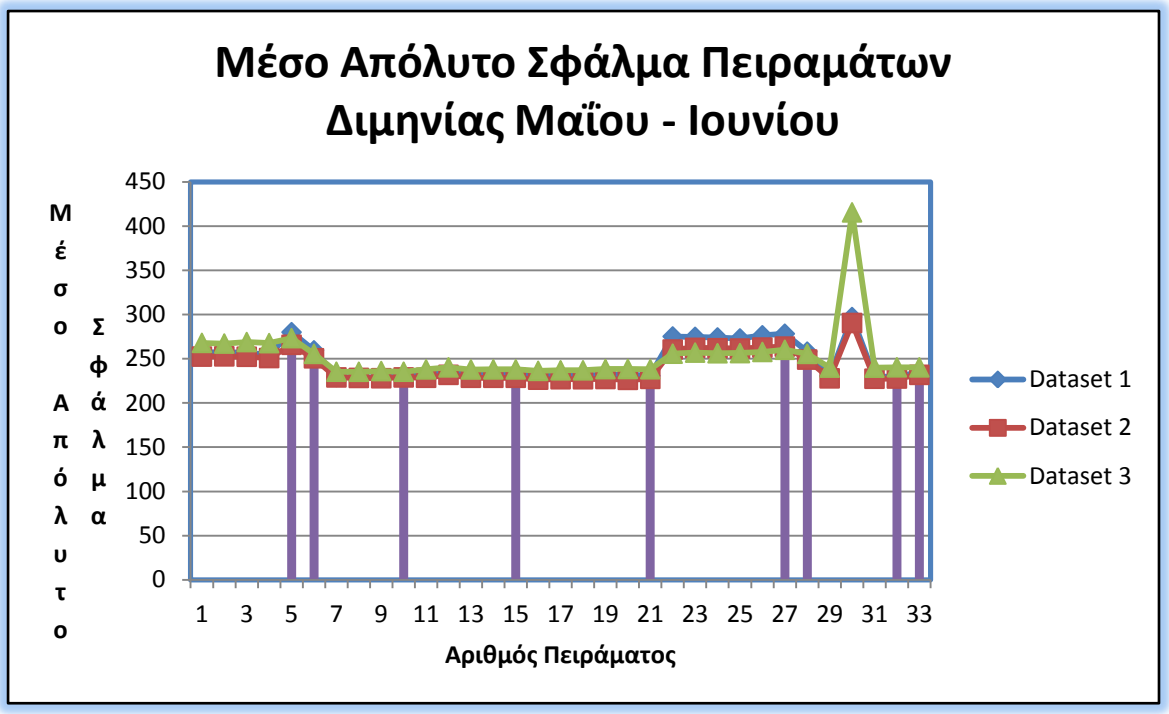
διμηνία αυτή, έχουμε πάρει σαν ελάχιστο σφάλμα στο πρώτο σύνολο δεδομένων τα 393,28, στο δεύτερο 315,68 καθώς στο τρίτο σύνολο 318,97. Έτσι το σύνολο που θα θέλαμε να χρησιμοποιήσουμε σαν είσοδο είναι το **δεύτερο σύνολο δεδομένων**, ενώ ο αλγόριθμος ο οποίος θα επιθυμούσαμε να χρησιμοποιήσουμε είναι ο **αλγόριθμος του ελαχίστου τετραγωνικού μέσου**. Έτσι έχουμε μέσο απόλυτο σφάλμα ***315,68***. Υπενθυμίζεται ότι στη διμηνία αυτή η **τυπική απόκλιση είναι 644,466**. Συνεπώς με την χρήση του αλγορίθμου αυτού, το σφάλμα είναι *το μισό της τυπικής απόκλισης*, πράγμα που σημαίνει ότι υπάρχει μάθηση μέσω του αλγορίθμου, και υπάρχουν ίσως όχι άριστα αποτελέσματα, αλλά αρκετά ικανοποιητικά

5.3.3 Διμηνία Μαΐου - Ιουνίου

Μέσο απόλυτο σφάλμα διμηνίας Μάιου – Ιουνίου			
Α/Α Συνόλου δεδομένων διμηνίας / Α/Α εκτέλεσης πειράματος	Σύνολο δεδομένων 1	Σύνολο δεδομένων 2	Σύνολο δεδομένων 3
1	257	252,47	267,98
2	257,66	252,74	267,21
3	256,48	252,29	268,91
4	254,96	250,84	267,69
5	280,03	265,81	273,38
6	259,5	250,56	255,54
7	229,46	228,89	235,37
8	229,25	228,34	235,31
9	229,2	227,99	236,09
10	229,77	228,91	235,37
11	233,43	228,51	238
12	234,25	231,84	240,32
13	232,79	228,52	237,91
14	233,43	228,51	238
15	233,43	228,51	238
16	230,25	226,38	236,35
17	231,39	226,88	236,96
18	231,39	226,88	236,96

19	232,15	227,12	238,36
20	232,15	226,16	238,46
21	230,78	227,26	238,18
22	275,22	260,43	255,75
23	274,77	262,31	257
24	274,11	261,9	256,26
25	272,96	261,54	256,43
26	276,55	262,66	257,72
27	278,29	263,74	260,17
28	258,02	249,08	255,54
29	230,39	227,88	240,05
30	297,18	290,46	415,55
31	229,65	227,42	240
32	229,69	227,66	240,37
33	234,86	231,83	240,31
Ελάχιστο μέσο απόλυτο σφάλμα	**229,2**	**226,16**	**235,31**
Ελάχιστο μέσο απόλυτο σφάλμα διμηνίας		**226,16**	

Πίνακας 5.3 – Μέσο απόλυτο σφάλμα για την διμηνία Απριλίου - Ιουνίου των 33 πειραμάτων για τα σύνολα δεδομένων 1,2 και 3.



Γράφημα 5.3 – Γραφική αναπαράσταση μέσου απολύτου σφάλματος για την διμηνία Μαΐου - Ιουνίου των 33 πειραμάτων για τα σύνολα δεδομένων 1,2 και 3

Για να κατανοήσουμε καλύτερα την γραφική παράσταση, παραθέτω σε συντομία τους αλγόριθμους οι οποίοι είχαν εκτελεστεί σε κάθε πείραμα, οι οποίοι έχουν ως εξής:

Στα πειράματα 1 μέχρι 5, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στο πείραμα 6, είχε εκτελεστεί Ισοτονική Παλινδρόμηση.

Στα πειράματα 7 μέχρι 10, είχε εκτελεστεί ο αλγόριθμος του Ελαχίστου Μέσου Τετραγώνων.

Στα πειράματα 11 μέχρι 15, είχε εκτελεστεί Γραμμική Παλινδρόμηση.

Στα πειράματα 16 μέχρι 21, είχε εκτελεστεί Βηματική Παλινδρόμηση.

Στα πειράματα 22 μέχρι 27, είχε εκτελεστεί Δίκτυο RBF.

Στο πείραμα 28, είχε εκτελεστεί Απλή Γραμμική Παλινδρόμηση.

Στα πειράματα 29 μέχρι 32, είχε εκτελεστεί Παλινδρόμηση SMO.

Στο πείραμα 33, είχε χρησιμοποιηθεί ο ταξινομητής PLS

Ανάλυση Αποτελεσμάτων: Μπορούμε να παρατηρήσουμε από την γραφική παράσταση το μέγεθος του μέσου απόλυτου σφάλματος, ανά εκτέλεση. Σύμφωνα με τον αλγόριθμο τον οποίος τρέχει, παρατηρούνται διαφορετικά επίπεδα σφάλματος, καθώς επίσης ότι τα δεδομένα και των τριών συνόλων δεδομένων έχουν παρόμοιο μέσο απόλυτο σφάλμα. Μπορούμε να παρατηρήσουμε ότι στις πρώτες πέντε εκτελέσεις που εκτελείται ο αλγόριθμος των Γκαουσιανών Διαδικασιών, το σφάλμα στο πρώτο σύνολο δεδομένων κυμαίνεται γύρω

στα 257, ενώ γύρω στα 252 και 268 στο δεύτερο και τρίτο σύνολο δεδομένων αντίστοιχα. Στην εκτέλεση 6, όπου εκτελείται ο αλγόριθμος isotonic regression, παρατηρούμε παρόμοια αποτελέσματα, ενώ στις εκτελέσεις 7-10 όπου τρέχει ο αλγόριθμος του ελαχίστου τετραγωνικού μέσου, παρατηρούμε ότι λαμβάνουμε τα καλύτερα αποτελέσματα. Πιο συγκεκριμένα τα αποτελέσματα αυτά κυμαίνονται γύρω στα 229 στο πρώτο σύνολο δεδομένων, γύρω στα 228,5 στο δεύτερο σύνολο δεδομένων και γύρω στα 235 στο τρίτο σύνολο δεδομένων. Στις επαναλήψεις από 11 μέχρι και 15 εκτελείται ο αλγόριθμος γραμμικής παλινδρόμησης, ο οποίος και πάλι μας δίνει αρκετά καλά αποτελέσματα: 233 στο πρώτο σύνολο δεδομένων, 228 στο δεύτερο και 238 στο τρίτο. Στη συνέχεια οι εκτελέσεις της βηματικής παλινδρόμησης, έχουν ικανοποιητικά αποτελέσματα, αφού έχουν γύρω στα 231 μέσο απόλυτο σφάλμα στο πρώτο σύνολο δεδομένων, 227 στο δεύτερο και 238 στο τρίτο. Ακολουθούν από την εκτέλεση 27 μέχρι και την 32, τα δίκτυα RBF, τα οποία έχουν υψηλότερο μέσο σφάλμα από τους υπόλοιπους αλγορίθμους που θα μπορούσαν να θεωρηθούν ότι έχουν παράξει ικανοποιητικά αποτελέσματα. Επίσης μπορούμε να παρατηρήσουμε στη συνέχεια τα αποτελέσματα του αλγορίθμου παλινδρόμηση SMO τα οποία κυμαίνονται στα περίπου 229, 228 και 240 στα τρία δεδομένα εισόδου αντίστοιχα. Στο τέλος μπορούμε να παρατηρήσουμε τον ταξινομητή PLS ο οποίος έχει παρόμοια αποτελέσματα.

Επίσης, θα παρατηρήσουμε, ότι στο πείραμα 30 , όπου εκτελείται ο αλγόριθμος παλινδρόμηση SOM, υπάρχει μία κορυφή στην γραφική παράσταση. Παρόλο που τόσο το πείραμα 29, όσο και το πείραμα 31, είναι επίσης παλινδρόμηση SOM, έχουν σαφώς καλύτερα αποτελέσματα. Η διαφορά της συγκεκριμένης εκτέλεσης σε σχέση με τις άλλες, έγκειται στο γεγονός ότι είχαμε θέσει τον εκθέτη του πολυωνυμικού πυρήνα ίσο με 1,2 σε αντίθεση με τις υπόλοιπες εκτελέσεις του αλγορίθμου όπου ήταν 1. Έχουμε κυρίως αφήσει τα αποτελέσματα της συγκεκριμένης εκτέλεσης του αλγορίθμου στη γραφική παράσταση, για να μπορούμε να κατανοήσουμε καλύτερα το μέγεθος με το οποίο μία παράμετρος κάποιου αλγορίθμου, μπορεί να επηρεάζει δραστικά τις ικανότητες του να παράγει σωστά αποτελέσματα.

Γενικά μπορούμε να παρατηρήσουμε ότι, υπάρχουν καλύτερα αποτελέσματα στο σύνολο δεδομένων 2 από ότι στα σύνολα 1 και 2. Επίσης μπορούμε να δούμε ότι τα καλύτερα αποτελέσματα έρχονται από τον αλγόριθμο βηματικής παλινδρόμησης, ακολουθούμενου από τον αλγόριθμο ελάχιστου τετραγωνικού μέσου, με εξαιρετικά μικρή διαφορά μεταξύ των δύο. Πιο συγκεκριμένα, όσον αφορά την διμηνία αυτή, έχουμε πάρει σαν ελάχιστο σφάλμα στο πρώτο σύνολο δεδομένων τα 229,2, στο δεύτερο 226,16 καθώς στο τρίτο σύνολο 235,31.

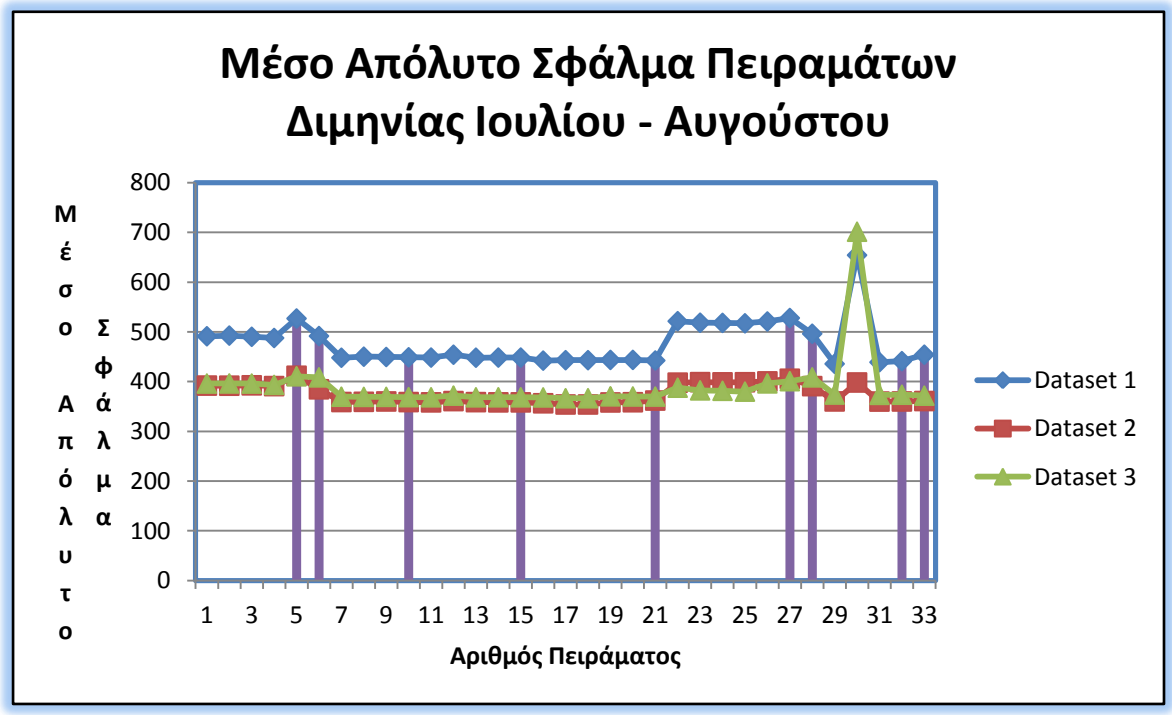
Έτσι το σύνολο που θα θέλαμε να χρησιμοποιήσουμε σαν είσοδο είναι το **δεύτερο σύνολο δεδομένων**, ενώ ο αλγόριθμος ο οποίος θα επιθυμούσαμε να χρησιμοποιήσουμε είναι ο **αλγόριθμος της βηματικής παλινδρόμησης**. Έτσι έχουμε μέσο απόλυτο σφάλμα ***235,31***. Υπενθυμίζεται ότι στη διμηνία αυτή η **τυπική απόκλιση είναι 381,491**. Συνεπώς με την χρήση του αλγορίθμου αυτού, το σφάλμα είναι *γύρω στα 38,32% λιγότερο της τυπικής απόκλισης*, πράγμα που σημαίνει ότι υπάρχει μάθηση μέσω του αλγορίθμου, και υπάρχουν ίσως όχι άριστα αποτελέσματα, αλλά αρκετά ικανοποιητικά

5.3.4 Διμηνία Ιουλίου - Αυγούστου

Μέσο απόλυτο σφάλμα διμηνίας Ιουλίου – Αυγούστου			
Α/Α Συνόλου δεδομένων διμηνίας / Α/Α εκτέλεσης πειράματος	Σύνολο δεδομένων 1	Σύνολο δεδομένων 2	Σύνολο δεδομένων 3
1	491,16	391,99	396,09
2	492,48	391,79	396,26
3	490,08	392,51	396,05
4	487,68	390,95	393,3
5	526,86	411,73	410,74
6	491,59	384,52	408,54
7	448	358,96	369,64
8	450,09	359,33	369,31
9	449,49	360,07	369,34
10	448,88	358,86	368,82
11	448,32	358,26	368,51
12	454,03	360,85	371,8
13	448,06	358,71	368,44
14	448,32	358,26	368,51
15	448,32	358,26	368,51
16	442,16	356,42	368,25
17	443,07	353,99	366,56
18	443,07	353,99	366,56
19	443,56	358,11	370,42
20	443,61	358,54	370,08

21	442,41	361,91	369,92
22	521,43	397,9	387,81
23	518,39	398,89	382,41
24	518,03	398,49	381,52
25	517,14	398,67	379,88
26	520,86	400,39	396,45
27	527,81	405,72	400,88
28	496,24	390,89	408,54
29	435,67	360,14	373,95
30	654,17	398,2	701,36
31	438,99	359,86	373,39
32	441,13	360,06	373,63
33	454,1	360,91	371,75
Ελάχιστο μέσο απόλυτο σφάλμα	**435,67**	**353,99**	**366,56**
Ελάχιστο μέσο απόλυτο σφάλμα διμηνίας		**353,99**	

Πίνακας 5.4 – Μέσο απόλυτο σφάλμα για την διμηνία Ιουλίου - Αυγούστου των 33 πειραμάτων για τα σύνολα δεδομένων 1,2 και 3.



Γράφημα 5.4 – Γραφική αναπαράσταση μέσου απολύτου σφάλματος για την διμηνία Ιουλίου - Αυγούστου των 33 πειραμάτων για τα σύνολα δεδομένων 1,2 και 3

Για να κατανοήσουμε καλύτερα την γραφική παράσταση, παραθέτω σε συντομία τους αλγόριθμους οι οποίοι είχαν εκτελεστεί σε κάθε πείραμα, οι οποίοι έχουν ως εξής:

Στα πειράματα 1 μέχρι 5, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στο πείραμα 6, είχε εκτελεστεί Ισοτονική Παλινδρόμηση.

Στα πειράματα 7 μέχρι 10, είχε εκτελεστεί ο αλγόριθμος του Ελαχίστου Μέσου Τετραγώνων.

Στα πειράματα 11 μέχρι 15, είχε εκτελεστεί Γραμμική Παλινδρόμηση.

Στα πειράματα 16 μέχρι 21, είχε εκτελεστεί Βηματική Παλινδρόμηση.

Στα πειράματα 22 μέχρι 27, είχε εκτελεστεί Δίκτυο RBF.

Στο πείραμα 28, είχε εκτελεστεί Απλή Γραμμική Παλινδρόμηση.

Στα πειράματα 29 μέχρι 32, είχε εκτελεστεί Παλινδρόμηση SMO.

Στο πείραμα 33, είχε χρησιμοποιηθεί ο ταξινομητής PLS

Ανάλυση Αποτελεσμάτων: Μπορούμε να παρατηρήσουμε από την γραφική παράσταση το μέγεθος του μέσου απόλυτου σφάλματος, ανά εκτέλεση. Σύμφωνα με τον αλγόριθμο τον οποίος τρέχει, παρατηρούνται διαφορετικά επίπεδα σφάλματος, καθώς επίσης ότι τα δεδομένα του πρώτου συνόλου δεδομένων, έχουν περίπου γύρω στα 90 χειρότερο μέσο απόλυτο σφάλμα από τα υπόλοιπα δύο σύνολα δεδομένων. Μπορούμε να παρατηρήσουμε ότι στις πρώτες πέντε εκτελέσεις που εκτελείται ο αλγόριθμος των Γκαουσιανών Διαδικασιών, το σφάλμα στο πρώτο σύνολο δεδομένων κυμαίνεται γύρω στα 491, ενώ γύρω στα 391 και 396 στο δεύτερο και τρίτο σύνολο δεδομένων αντίστοιχα. Στην εκτέλεση 6, όπου εκτελείται ο αλγόριθμος isotonic regression, παρατηρούμε παρόμοια αποτελέσματα, ενώ στις εκτελέσεις 7-10 όπου τρέχει ο αλγόριθμος του ελαχίστου τετραγωνικού μέσου, παρατηρούμε ότι λαμβάνουμε τα καλύτερα αποτελέσματα. Πιο συγκεκριμένα τα αποτελέσματα αυτά κυμαίνονται γύρω στα 448 στο πρώτο σύνολο δεδομένων, γύρω στα 359 στο δεύτερο σύνολο δεδομένων και γύρω στα 368 στο τρίτο σύνολο δεδομένων. Στις επαναλήψεις από 11 μέχρι και 15 εκτελείται ο αλγόριθμος γραμμικής παλινδρόμησης, ο οποίος και πάλι μας δίνει αρκετά καλά αποτελέσματα: 448 στο πρώτο σύνολο δεδομένων, 358 στο δεύτερο και 368 στο τρίτο. Στη συνέχεια οι εκτελέσεις της βηματικής παλινδρόμησης, έχουν ικανοποιητικά αποτελέσματα, αφού έχουν γύρω στα 443 μέσο απόλυτο σφάλμα στο πρώτο σύνολο δεδομένων, 356 στο δεύτερο και 368 στο τρίτο. Ακολουθούν από την εκτέλεση 27 μέχρι και την 32, τα δίκτυα RBF, τα οποία έχουν υψηλότερο μέσο σφάλμα από τους υπόλοιπους αλγορίθμους που θα μπορούσαν να

θεωρηθούν ότι έχουν παράξει ικανοποιητικά αποτελέσματα. Επίσης μπορούμε να παρατηρήσουμε στη συνέχεια τα αποτελέσματα του αλγορίθμου παλινδρόμηση SMO τα οποία κυμαίνονται στα περίπου 440, 359 και 373 στα τρία δεδομένα εισόδου αντίστοιχα. Στο τέλος μπορούμε να παρατηρήσουμε τον ταξινομητή PLS ο οποίος έχει παρόμοια αποτελέσματα.

Επίσης, θα παρατηρήσουμε, ότι στο πείραμα 30 , όπου εκτελείται ο αλγόριθμος παλινδρόμηση SOM, υπάρχει μία κορυφή στην γραφική παράσταση. Παρόλο που τόσο το πείραμα 29, όσο και το πείραμα 31, είναι επίσης παλινδρόμηση SOM, έχουν σαφώς καλύτερα αποτελέσματα. Η διαφορά της συγκεκριμένης εκτέλεσης σε σχέση με τις άλλες, έγκειται στο γεγονός ότι είχαμε θέσει τον εκθέτη του πολυωνυμικού πυρήνα ίσο με 1,2 σε αντίθεση με τις υπόλοιπες εκτελέσεις του αλγορίθμου όπου ήταν 1. Έχουμε κυρίως αφήσει τα αποτελέσματα της συγκεκριμένης εκτέλεσης του αλγορίθμου στη γραφική παράσταση, για να μπορούμε να κατανοήσουμε καλύτερα το μέγεθος με το οποίο μία παράμετρος κάποιου αλγορίθμου, μπορεί να επηρεάζει δραστικά τις ικανότητες του να παράγει σωστά αποτελέσματα.

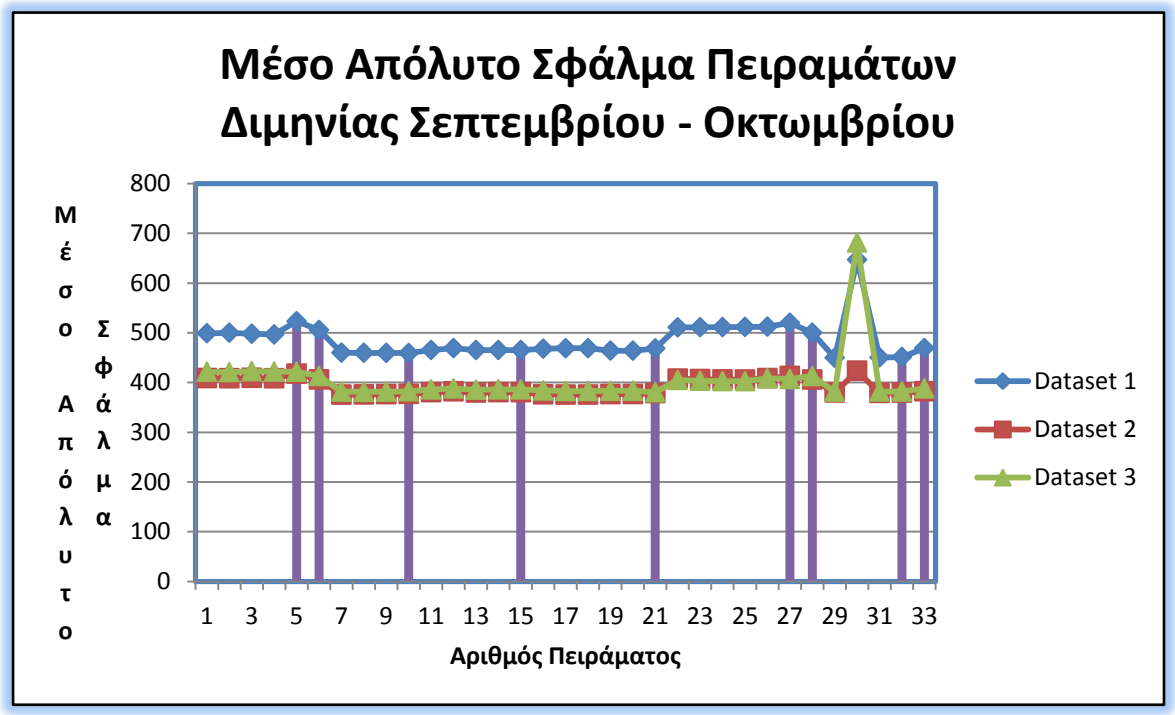
Γενικά μπορούμε να παρατηρήσουμε ότι, υπάρχουν καλύτερα αποτελέσματα στο σύνολο δεδομένων 2 από ότι στο τρία, καθώς επίσης ότι το σύνολο δεδομένων 1 δεν έχει τόσο καλά αποτελέσματα όσο τα υπόλοιπα δύο σύνολα. Επίσης μπορούμε να δούμε ότι τα καλύτερα αποτελέσματα έρχονται από τον αλγόριθμο της βηματικής παλινδρόμησης, ακολουθούμενου από τον αλγόριθμο των ελαχίστων τετραγώνων των μέσων. Πιο συγκεκριμένα, όσον αφορά την διμηνία αυτή, έχουμε πάρει σαν ελάχιστο σφάλμα στο πρώτο σύνολο δεδομένων τα 435,67, στο δεύτερο 353,99 καθώς στο τρίτο σύνολο 366,56. Έτσι το σύνολο που θα θέλαμε να χρησιμοποιήσουμε σαν είσοδο είναι το **δεύτερο σύνολο δεδομένων**, ενώ ο αλγόριθμος ο οποίος θα επιθυμούσαμε να χρησιμοποιήσουμε είναι ο **αλγόριθμος της βηματικής παλινδρόμησης**. Έτσι έχουμε μέσο απόλυτο σφάλμα ***353,99***. Υπενθυμίζεται ότι στη διμηνία αυτή η **τυπική απόκλιση είναι 741,27**. Συνεπώς με την χρήση του αλγορίθμου αυτού, το σφάλμα είναι *το μικρότερο του μισού της τυπικής απόκλισης*, πράγμα που σημαίνει ότι υπάρχει μάθηση μέσω του αλγορίθμου, και υπάρχουν ίσως όχι άριστα αποτελέσματα, αλλά αρκετά ικανοποιητικά

5.3.5 Διμηνία Σεπτέμβρη - Οκτωβρίου

Μέσο απόλυτο σφάλμα διμηνίας Σεπτεμβρίου – Οκτωβρίου			
Α/Α Συνόλου δεδομένων διμηνίας / Α/Α εκτέλεσης πειράματος	Σύνολο δεδομένων 1	Σύνολο δεδομένων 2	Σύνολο δεδομένων 3
1	498,89	408,82	421,73
2	499,99	408,48	420,93
3	497,88	409,28	422,77
4	496,49	408,45	421,96
5	523,1	417,72	421,64
6	505,57	406,08	413
7	460,01	375,57	381,68
8	459,65	375,94	381,23
9	459,5	376,49	381,53
10	459,1	377,09	382,46
11	465,29	380,55	386,15
12	468,99	382,23	388
13	465,18	379,75	385,53
14	465,29	380,55	386,15
15	465,29	380,55	386,15
16	467,89	376,17	384,04
17	468,79	375,6	382,96
18	468,79	375,6	382,96
19	464,05	376,56	383,27
20	463,97	376,73	383,51
21	468,38	378,49	380,39
22	510,76	407,67	405,16
23	511,04	406,78	403,45
24	511,26	406,02	402,81
25	511,7	405,81	402,57
26	512,25	408,72	408,46
27	520,05	413,02	406,97
28	499,98	405,85	413

29	449,73	379,78	381,65
30	646,83	423,6	681,51
31	450,3	378,97	381,75
32	451,23	379,65	381,88
33	469,12	382,33	388,13
Ελάχιστο μέσο απόλυτο σφάλμα	**449,73**	**375,57**	**380,39**
Ελάχιστο μέσο απόλυτο σφάλμα διμηνίας		**375,57**	

Πίνακας 5.5 – Μέσο απόλυτο σφάλμα για την διμηνία Σεπτεμβρίου – Οκτωβρίου των 33 πειραμάτων για τα σύνολα δεδομένων 1,2 και 3.



Γράφημα 5.5 – Γραφική αναπαράσταση μέσου απολύτου σφάλματος για την διμηνία Σεπτεμβρίου - Οκτωβρίου των 33 πειραμάτων για τα σύνολα δεδομένων 1,2 και 3

Για να κατανοήσουμε καλύτερα την γραφική παράσταση, παραθέτω σε συντομία τους αλγόριθμους οι οποίοι είχαν εκτελεστεί σε κάθε πείραμα, οι οποίοι έχουν ως εξής:

Στα πειράματα 1 μέχρι 5, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στο πείραμα 6, είχε εκτελεστεί Ισοτονική Παλινδρόμηση.

Στα πειράματα 7 μέχρι 10, είχε εκτελεστεί ο αλγόριθμος του Ελαχίστου Μέσου Τετραγώνων.
Στα πειράματα 11 μέχρι 15, είχε εκτελεστεί Γραμμική Παλινδρόμηση.
Στα πειράματα 16 μέχρι 21, είχε εκτελεστεί Βηματική Παλινδρόμηση.
Στα πειράματα 22 μέχρι 27, είχε εκτελεστεί Δίκτυο RBF.
Στο πείραμα 28, είχε εκτελεστεί Απλή Γραμμική Παλινδρόμηση.
Στα πειράματα 29 μέχρι 32, είχε εκτελεστεί Παλινδρόμηση SMO.
Στο πείραμα 33, είχε χρησιμοποιηθεί ο ταξινομητής PLS

Ανάλυση Αποτελεσμάτων: Μπορούμε να παρατηρήσουμε από την γραφική παράσταση το μέγεθος του μέσου απόλυτου σφάλματος, ανά εκτέλεση. Σύμφωνα με τον αλγόριθμο τον οποίος τρέχει, παρατηρούνται διαφορετικά επίπεδα σφάλματος, καθώς επίσης ότι τα δεδομένα του πρώτου συνόλου δεδομένων, έχουν περίπου γύρω στα 85 χειρότερο μέσο απόλυτο σφάλμα από τα υπόλοιπα δύο σύνολα δεδομένων. Μπορούμε να παρατηρήσουμε ότι στις πρώτες πέντε εκτελέσεις που εκτελείται ο αλγόριθμος των Γκαουσιανών Διαδικασιών, το σφάλμα στο πρώτο σύνολο δεδομένων κυμαίνεται γύρω στα 498, ενώ γύρω στα 408 και 421 στο δεύτερο και τρίτο σύνολο δεδομένων αντίστοιχα. Στην εκτέλεση 6, όπου εκτελείται ο αλγόριθμος isotonic regression, παρατηρούμε παρόμοια αποτελέσματα, ενώ στις εκτελέσεις 7-10 όπου τρέχει ο αλγόριθμος του ελαχίστου τετραγωνικού μέσου, παρατηρούμε ότι λαμβάνουμε τα καλύτερα αποτελέσματα. Πιο συγκεκριμένα τα αποτελέσματα αυτά κυμαίνονται γύρω στα 459 στο πρώτο σύνολο δεδομένων, γύρω στα 375 στο δεύτερο σύνολο δεδομένων και γύρω στα 381 στο τρίτο σύνολο δεδομένων. Στις επαναλήψεις από 11 μέχρι και 15 εκτελείται ο αλγόριθμος γραμμικής παλινδρόμησης, ο οποίος και πάλι μας δίνει αρκετά καλά αποτελέσματα: 465 στο πρώτο σύνολο δεδομένων, 380 στο δεύτερο και 386 στο τρίτο. Στη συνέχεια οι εκτελέσεις της βηματικής παλινδρόμησης, έχουν ικανοποιητικά αποτελέσματα, αφού έχουν γύρω στα 467 μέσο απόλυτο σφάλμα στο πρώτο σύνολο δεδομένων, 375 στο δεύτερο και 382 στο τρίτο. Ακολουθούν από την εκτέλεση 27 μέχρι και την 32, τα δίκτυα RBF, τα οποία έχουν υψηλότερο μέσο σφάλμα από τους υπόλοιπους αλγορίθμους που θα μπορούσαν να θεωρηθούν ότι έχουν παράξει ικανοποιητικά αποτελέσματα. Επίσης μπορούμε να παρατηρήσουμε στη συνέχεια τα αποτελέσματα του αλγορίθμου παλινδρόμηση SMO τα οποία κυμαίνονται στα περίπου 450, 378 και 381 στα τρία δεδομένα εισόδου αντίστοιχα. Στο τέλος μπορούμε να παρατηρήσουμε τον ταξινομητή PLS ο οποίος έχει παρόμοια αποτελέσματα.

Επίσης, θα παρατηρήσουμε, ότι στο πείραμα 30 , όπου εκτελείται ο αλγόριθμος παλινδρόμηση SOM, υπάρχει μία κορυφή στην γραφική παράσταση. Παρόλο που τόσο το

πείραμα 29, όσο και το πείραμα 31, είναι επίσης παλινδρόμηση SOM, έχουν σαφώς καλύτερα αποτελέσματα. Η διαφορά της συγκεκριμένης εκτέλεσης σε σχέση με τις άλλες, έγκειται στο γεγονός ότι είχαμε θέσει τον εκθέτη του πολυωνυμικού πυρήνα ίσο με 1,2 σε αντίθεση με τις υπόλοιπες εκτελέσεις του αλγορίθμου όπου ήταν 1. Έχουμε κυρίως αφήσει τα αποτελέσματα της συγκεκριμένης εκτέλεσης του αλγορίθμου στη γραφική παράσταση, για να μπορούμε να κατανοήσουμε καλύτερα το μέγεθος με το οποίο μία παράμετρος κάποιου αλγορίθμου, μπορεί να επηρεάζει δραστικά τις ικανότητες του να παράγει σωστά αποτελέσματα.

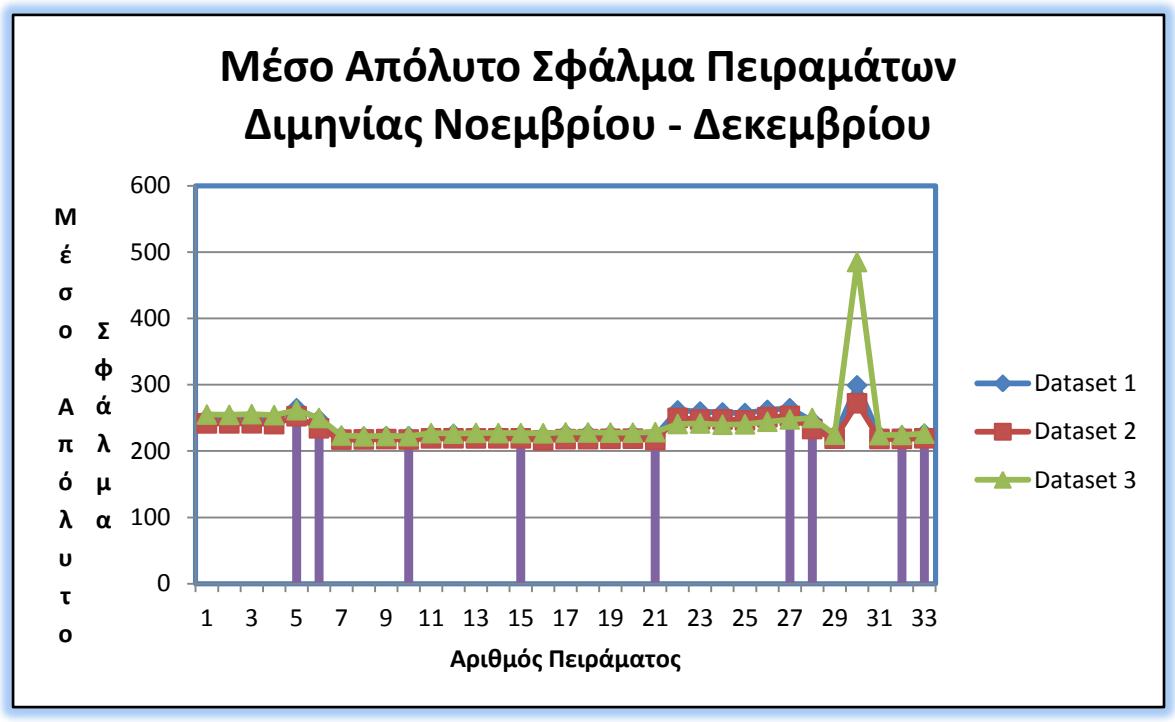
Γενικά μπορούμε να παρατηρήσουμε ότι, υπάρχουν καλύτερα αποτελέσματα στο σύνολο δεδομένων 2 από ότι στο τρία, καθώς επίσης ότι το σύνολο δεδομένων 1 δεν έχει τόσο καλά αποτελέσματα όσο τα υπόλοιπα δύο σύνολα. Επίσης μπορούμε να δούμε ότι τα καλύτερα αποτελέσματα έρχονται από τον αλγόριθμο των ελαχίστων τετραγωνικών μέσων, ακολουθούμενου από τον αλγόριθμο της βηματικής παλινδρόμησης. Πιο συγκεκριμένα, όσον αφορά την διμηνία αυτή, έχουμε πάρει σαν ελάχιστο σφάλμα στο πρώτο σύνολο δεδομένων τα 449,73 , στο δεύτερο 375,57 καθώς στο τρίτο σύνολο 380,39. Έτσι το σύνολο που θα θέλαμε να χρησιμοποιήσουμε σαν είσοδο είναι το **δεύτερο σύνολο δεδομένων**, ενώ ο αλγόριθμος ο οποίος θα επιθυμούσαμε να χρησιμοποιήσουμε είναι ο **αλγόριθμος των ελαχίστων τετραγωνικών μέσων**. Έτσι έχουμε μέσο απόλυτο σφάλμα ***375,57***. Υπενθυμίζεται ότι στη διμηνία αυτή η **τυπική απόκλιση είναι 725,86**. Συνεπώς με την χρήση του αλγορίθμου αυτού, το σφάλμα είναι *γύρω στο μισό της τυπικής απόκλισης*, πράγμα που σημαίνει ότι υπάρχει μάθηση μέσω του αλγορίθμου, και υπάρχουν ίσως όχι άριστα αποτελέσματα, αλλά αρκετά ικανοποιητικά

5.3.6 Διμηνία Νοέμβρη – Δεκέμβρη

Μέσο απόλυτο σφάλμα διμηνίας Νοεμβρίου – Δεκεμβρίου			
Α/Α Συνόλου δεδομένων διμηνίας / Α/Α εκτέλεσης πειράματος	Σύνολο δεδομένων 1	Σύνολο δεδομένων 2	Σύνολο δεδομένων 3
1	249,75	241,73	255,05
2	249,86	241,8	254,59
3	249,84	241,92	255,66
4	248,42	240,67	254,18

5	264,5	252,52	261,71
6	247,52	234,29	249,91
7	221,94	217,28	223,48
8	221,52	217,57	222,19
9	222,25	218,06	222,77
10	222,26	217,45	222,54
11	224,06	219,16	227,02
12	225,56	219,25	226,26
13	224,17	219,1	227,41
14	224,06	219,16	227,02
15	224,06	219,16	227,02
16	222,55	216,02	226,84
17	225,3	217,96	228,04
18	225,3	217,96	228,04
19	224,84	218,42	227,74
20	224,66	218,71	227,73
21	222,51	216,78	228,52
22	261,3	249,94	240,85
23	259,61	247,53	241,64
24	258,25	247,67	239,67
25	257,25	246,39	240,32
26	262,1	251,09	244,23
27	264,6	252,91	247,68
28	245,1	233,22	249,91
29	223,02	218,67	224,78
30	299,06	272,08	485,07
31	222,73	218,24	224,13
32	222,88	218,36	224,33
33	226,01	219,25	226,26
Ελάχιστο μέσο απόλυτο σφάλμα	**221,52**	**216,02**	**222,19**
Ελάχιστο μέσο απόλυτο σφάλμα διμηνίας		**216,02**	

Πίνακας 5.6 – Μέσο απόλυτο σφάλμα για την διμηνία Νοεμβρίου - Δεκεμβρίου των 33 πειραμάτων για τα σύνολα δεδομένων 1,2 και 3.



Γράφημα 5.6 – Γραφική αναπαράσταση μέσου απολύτου σφάλματος για την διμηνία Νοεμβρίου - Δεκεμβρίου των 33 πειραμάτων για τα σύνολα δεδομένων 1,2 και 3

Για να κατανοήσουμε καλύτερα την γραφική παράσταση, παραθέτω σε συντομία τους αλγόριθμους οι οποίοι είχαν εκτελεστεί σε κάθε πείραμα, οι οποίοι έχουν ως εξής:

Στα πειράματα 1 μέχρι 5, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στο πείραμα 6, είχε εκτελεστεί Ισοτονική Παλινδρόμηση.

Στα πειράματα 7 μέχρι 10, είχε εκτελεστεί ο αλγόριθμος του Ελαχίστου Μέσου Τετραγώνων.

Στα πειράματα 11 μέχρι 15, είχε εκτελεστεί Γραμμική Παλινδρόμηση.

Στα πειράματα 16 μέχρι 21, είχε εκτελεστεί Βηματική Παλινδρόμηση.

Στα πειράματα 22 μέχρι 27, είχε εκτελεστεί Δίκτυο RBF.

Στο πείραμα 28, είχε εκτελεστεί Απλή Γραμμική Παλινδρόμηση.

Στα πειράματα 29 μέχρι 32, είχε εκτελεστεί Παλινδρόμηση SMO.

Στο πείραμα 33, είχε χρησιμοποιηθεί ο ταξινομητής PLS

Ανάλυση Αποτελεσμάτων: Μπορούμε να παρατηρήσουμε από την γραφική παράσταση το μέγεθος του μέσου απόλυτου σφάλματος, ανά εκτέλεση. Σύμφωνα με τον αλγόριθμο τον

οποίος τρέχει, παρατηρούνται διαφορετικά επίπεδα σφάλματος, καθώς επίσης ότι τα δεδομένα των τριών συνόλων δεδομένων έχουν παρόμοιο μέσο απόλυτο σφάλμα. Μπορούμε να παρατηρήσουμε ότι στις πρώτες πέντε εκτελέσεις που εκτελείται ο αλγόριθμος των Γκαουσιανών Διαδικασιών, το σφάλμα στο πρώτο σύνολο δεδομένων κυμαίνεται γύρω στα 249, ενώ γύρω στα 241 και 255 στο δεύτερο και τρίτο σύνολο δεδομένων αντίστοιχα. Στην εκτέλεση 6, όπου εκτελείται ο αλγόριθμος isotonic regression, παρατηρούμε παρόμοια αποτελέσματα, ενώ στις εκτελέσεις 7-10 όπου τρέχει ο αλγόριθμος του ελαχίστου τετραγωνικού μέσου, παρατηρούμε ότι λαμβάνουμε τα καλύτερα αποτελέσματα. Πιο συγκεκριμένα τα αποτελέσματα αυτά κυμαίνονται γύρω στα 221 στο πρώτο σύνολο δεδομένων, γύρω στα 217,5 στο δεύτερο σύνολο δεδομένων και γύρω στα 222 στο τρίτο σύνολο δεδομένων. Στις επαναλήψεις από 11 μέχρι και 15 εκτελείται ο αλγόριθμος γραμμικής παλινδρόμησης, ο οποίος και πάλι μας δίνει αρκετά καλά αποτελέσματα: 224 στο πρώτο σύνολο δεδομένων, 219 στο δεύτερο και 227 στο τρίτο. Στη συνέχεια οι εκτελέσεις της βηματικής παλινδρόμησης, έχουν ικανοποιητικά αποτελέσματα, αφού έχουν γύρω στα 225 μέσο απόλυτο σφάλμα στο πρώτο σύνολο δεδομένων, 218 στο δεύτερο και 228 στο τρίτο. Ακολουθούν από την εκτέλεση 27 μέχρι και την 32, τα δίκτυα RBF, τα οποία έχουν υψηλότερο μέσο σφάλμα από τους υπόλοιπους αλγορίθμους που θα μπορούσαν να θεωρηθούν ότι έχουν παράξει ικανοποιητικά αποτελέσματα. Επίσης μπορούμε να παρατηρήσουμε στη συνέχεια τα αποτελέσματα του αλγορίθμου παλινδρόμηση SMO τα οποία κυμαίνονται στα περίπου 222, 218 και 224 στα τρία δεδομένα εισόδου αντίστοιχα. Στο τέλος μπορούμε να παρατηρήσουμε τον ταξινομητή PLS ο οποίος έχει παρόμοια αποτελέσματα.

Επίσης, θα παρατηρήσουμε, ότι στο πείραμα 30 , όπου εκτελείται ο αλγόριθμος παλινδρόμηση SOM, υπάρχει μία κορυφή στην γραφική παράσταση. Παρόλο που τόσο το πείραμα 29, όσο και το πείραμα 31, είναι επίσης παλινδρόμηση SOM, έχουν σαφώς καλύτερα αποτελέσματα. Η διαφορά της συγκεκριμένης εκτέλεσης σε σχέση με τις άλλες, έγκειται στο γεγονός ότι είχαμε θέσει τον εκθέτη του πολυωνυμικού πυρήνα ίσο με 1,2 σε αντίθεση με τις υπόλοιπες εκτελέσεις του αλγορίθμου όπου ήταν 1. Έχουμε κυρίως αφήσει τα αποτελέσματα της συγκεκριμένης εκτέλεσης του αλγορίθμου στη γραφική παράσταση, για να μπορούμε να κατανοήσουμε καλύτερα το μέγεθος με το οποίο μία παράμετρος κάποιου αλγορίθμου, μπορεί να επηρεάζει δραστικά τις ικανότητες του να παράγει σωστά αποτελέσματα.

Γενικά μπορούμε να παρατηρήσουμε ότι, υπάρχουν καλύτερα αποτελέσματα στο σύνολο δεδομένων 2 από ότι στο τρία, καθώς επίσης ότι το σύνολο δεδομένων 3 αυτή τη φορά δεν

έχει τόσο καλά αποτελέσματα όσο τα υπόλοιπα δύο σύνολα. Επίσης μπορούμε να δούμε ότι τα καλύτερα αποτελέσματα έρχονται από τον αλγόριθμο της βηματικής παλινδρόμησης μέσων, ακολουθούμενου από τον αλγόριθμο των ελαχίστων τετραγώνων των μέσων. Πιο συγκεκριμένα, όσον αφορά την διμηνία αυτή, έχουμε πάρει σαν ελάχιστο σφάλμα στο πρώτο σύνολο δεδομένων τα 221,52 , στο δεύτερο 216,02 καθώς στο τρίτο σύνολο 222,19. Έτσι το σύνολο που θα θέλαμε να χρησιμοποιήσουμε σαν είσοδο είναι το **δεύτερο σύνολο δεδομένων**, ενώ ο αλγόριθμος ο οποίος θα επιθυμούσαμε να χρησιμοποιήσουμε είναι ο **αλγόριθμος της βηματικής παλινδρόμησης**. Έτσι έχουμε μέσο απόλυτο σφάλμα ***216,02***. Υπενθυμίζεται ότι στη διμηνία αυτή η **τυπική απόκλιση είναι 360,2**. Συνεπώς με την χρήση του αλγορίθμου αυτού, το σφάλμα μειώνεται κατά 40% *της τυπικής απόκλισης*, πράγμα που σημαίνει ότι υπάρχει μάθηση μέσω του αλγορίθμου, και υπάρχουν ίσως όχι άριστα αποτελέσματα, αλλά αρκετά ικανοποιητικά

5.4 Μελέτη βελτίωσης των αποτελεσμάτων των αλγορίθμων συγκρίσει του αριθμού των δεδομένων

Για να μελετήσουμε την επίδραση που έχει η αύξηση του μεγέθους του συνόλου των δεδομένων μας, στους αλγορίθμους που χρησιμοποιούμε, έχουμε εκτελέσει ένα σύνολο πειραμάτων. Παρακάτω, παραθέτουμε το σύνολο των πειραμάτων τα οποία είχαμε εκτελέσει, καθώς και τα αποτελέσματά τους για κάθε διμηνία. Πιο συγκεκριμένα, είχαμε εκτελέσει ένα σύνολο διαφορετικών αλγορίθμων, για να δούμε την επίδραση στον κάθε ένα αλγόριθμο ξεχωριστά (ίσως κάποιοι αλγόριθμοι, να μαθαίνουν με λιγότερα δεδομένα από κάποιους άλλους), για όλες τις έξι διμηνίες που μελετούμε. Αυτό γιατί κάθε μια διμηνία έχει διαφορετική τυπική απόκλιση, και εξαρτάται από διαφορετικά χαρακτηριστικά σε σχέση οποιαδήποτε άλλη.

Τα πειράματα τα οποία είχαμε χρησιμοποιήσει είναι τα παρακάτω:

Πείραμα 1: Χρησιμοποιήσαμε Γκαουσιανές διαδικασίες. Είχαμε θέσει την παράμετρο του θορύβου στο 1,1, και είχαμε κανονικοποιήσει τα δεδομένα. Επίσης ως συνάρτηση πυρήνα, είχαμε χρησιμοποιήσει τον πυρήνα τύπου RBF με παράμετρο gamma ίση με 1.

Πείραμα 2: Χρησιμοποιήσαμε Γκαουσιανές διαδικασίες. Είχαμε θέσει την παράμετρο του θορύβου στο 0,9, και είχαμε κανονικοποιήσει τα δεδομένα. Επίσης ως συνάρτηση πυρήνα, είχαμε χρησιμοποιήσει τον πυρήνα τύπου RBF με παράμετρο gamma ίση με 1.

Πείραμα 3: Χρησιμοποιήσαμε Γραμμική Παλινδρόμηση. Είχαμε ακόμη χρησιμοοιήσει επιλογή χαρακτηριστικών (attribute selection) με την μέθοδο M5. Είχαμε θέσει την παράμετρο της κορυφογραμμής στο 1.0E-8.

Πείραμα 4: Χρησιμοποιήσαμε Γραμμική Παλινδρόμηση. Δεν είχαμε χρησιμοοιήσει επιλογή χαρακτηριστικών και έτσι όλα τα χαρακτηριστικά εισήχθησαν στο μοντέλλο. Είχαμε θέσει την παράμετρο της κορυφογραμμής στο 1.0E-8.

Πείραμα 5: Χρησιμοποιήσαμε Γραμμική Παλινδρόμηση. Είχαμε ακόμη χρησιμοοιήσει επιλογή χαρακτηριστικών (attribute selection) με την άπλυστη μέθοδο. Είχαμε θέσει την παράμετρο της κορυφογραμμής στο 1.0E-8.

Πείραμα 6: Χρησιμοποιήσαμε Βηματική Παλινδρόμηση. Είχαμε θέσει τον εκτιμητή (estimator) να είναι ο Εμπειρικός Bayes.

Πείραμα 7: Χρησιμοποιήσαμε Βηματική Παλινδρόμηση. Είχαμε θέσει τον εκτιμητή (estimator) να είναι ο Εμφωλευμένος Επιλογέας Μοντέλου (Nested Model Selector).

Πείραμα 8: Χρησιμοποιήσαμε Βηματική Παλινδρόμηση. Είχαμε θέσει τον εκτιμητή (estimator) να είναι ο PACE2.

Πείραμα 9: Χρησιμοποιήσαμε Δίκτυο συναρτήσεων αξονικών βάσεων (RBF network). Είχαμε θέσει τον αριθμό των ομάδων (clusters) να είναι ίσο με 4, την κορυφογραμμή να είναι ίση με 1,0E-8 και την ελάχιστη τυπική απόκλιση να είναι 0.1 .

Πείραμα 10: Χρησιμοποιήσαμε Δίκτυο συναρτήσεων αξονικών βάσεων (RBF network). Είχαμε θέσει τον αριθμό των ομάδων (clusters) να είναι ίσο με 3, την κορυφογραμμή να είναι ίση με 1,0E-8 και την ελάχιστη τυπική απόκλιση να είναι 0.1 .

Πείραμα 11: Χρησιμοποιήσαμε Δίκτυο συναρτήσεων αξονικών βάσεων (RBF network). Είχαμε θέσει τον αριθμό των ομάδων (clusters) να είναι ίσο με 2, την κορυφογραμμή να είναι ίση με 1,0E-8 και την ελάχιστη τυπική απόκλιση να είναι 0.1 .

5.4.1 Αποτελέσματα και συμπεράσματα των εκτελέσεων των πειραμάτων

Τα δεδομένα εισόδου στους αλγορίθμους, για κάθε μια από τις διμηνίες ήταν τα δεδομένα της διμηνίας, με διαγραφή του θορύβου (διαγραφή αδικαιολόγητα εξαιρετικά χαμηλών ή υψηλών ηλεκτρικών καταναλώσεων) όπως δηλαδή και στο σύνολο δεδομένων 2 στο υποκεφάλαιο 5.3. Υπήρχαν 4 διαφορετικά σύνολα δεδομένων, τα οποία εξετάστηκαν με τους πιο πάνω αλγορίθμους. Πιο συγκεκριμένα το σύνολο δεδομένων 1, περιείχε 100 τυχαίες εγγραφές, το σύνολο δεδομένων 2, περιείχε 200 τυχαίες εγγραφές, το σύνολο δεδομένων 3, περιείχε 300 τυχαίες εγγραφές, και το σύνολο δεδομένων 4, περιείχε 400 τυχαίες εγγραφές.

Παρουσιάζουμε για κάθε διμηνία τα αποτελέσματά της, μία γραφική παράσταση, καθώς και συμπεράσματα για τη γραφική παράσταση αυτή. Οι παραστάσεις αυτές έχουν τέσσερις διαφορετικές γραμμές, η κάθε μία δηλώνει την είσοδο ενός από τα πιο πάνω σύνολα δεδομένων για κάθε μία από τις 11 διαφορετικές εκτελέσεις των αλγορίθμων μας. Η κάθε μία από τις τιμές οι οποίες παρουσιάζονται στους πίνακες και στις γραφικές παραστάσεις είναι *ο μέσος όρος* των δέκα εκτελέσεων του συγκεκριμένου πειράματος στο συγκεκριμένο σύνολο δεδομένων της διμηνίας.

Οι πιο κάτω γραφικές παραστάσεις, περιέχουν κάθετες γραμμές οι οποίες αναπαριστούν το τελευταίο πείραμα ενός συγκεκριμένου αλγορίθμου ο οποίος έχει χρησιμοποιηθεί.

Θα επαναλαμβάνουμε τα ονόματα των αλγορίθμων οι οποίοι έχουν εκτελεστεί, κάτω από κάθε γραφική παράσταση, για να μπορούμε να παρατηρούμε ευκολότερα τις αποδόσεις των αλγορίθμων.

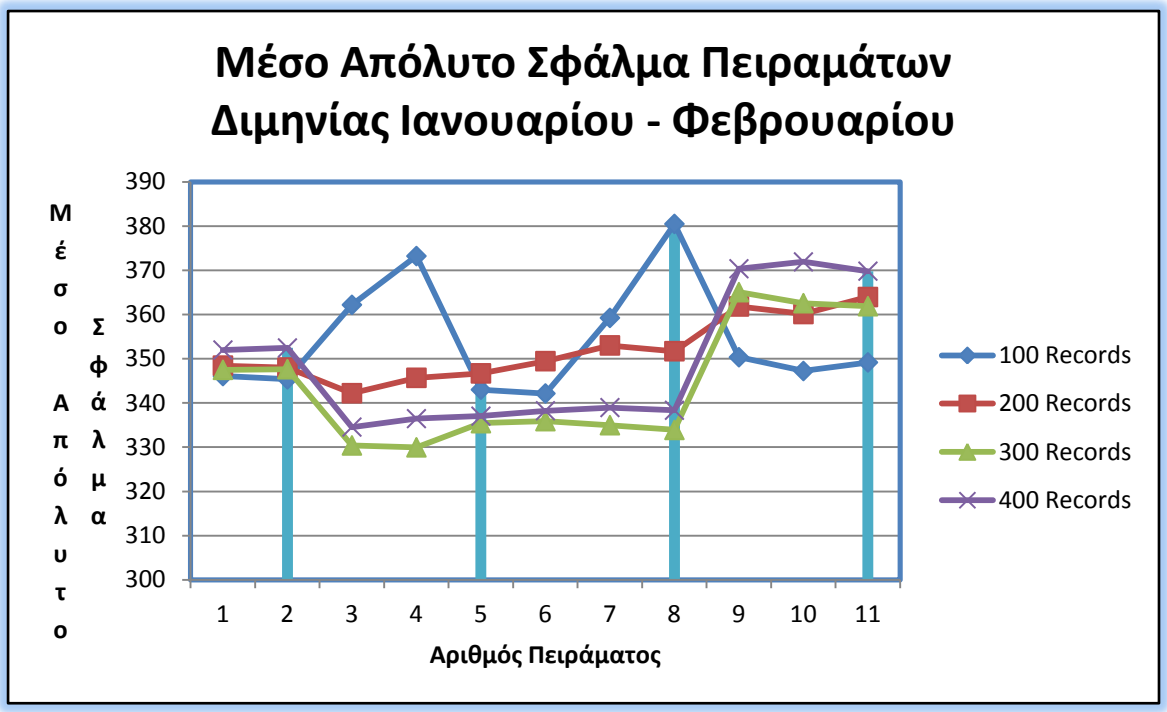
Πιο συγκεκριμένα:

Στα πειράματα 1 και 2, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.
Στα πειράματα 3 μέχρι 5, είχε εκτελεστεί Γραμμική Παλινδρόμηση.
Στα πειράματα 6 μέχρι 8, είχε εκτελεστεί Βηματική Παλινδρόμηση.
Στα πειράματα 9 μέχρι 11, είχε εκτελεστεί Δίκτυο RBF.

5.4.1.1 Διμηνία Ιανουαρίου – Φεβρουαρίου

Μέσο απόλυτο σφάλμα διμηνίας Ιανουαρίου – Φεβρουαρίου				
Α/Α Συνόλου δεδομένων διμηνίας / Α/Α εκτέλεσης πειράματος	Σύνολο δεδομένων 1	Σύνολο δεδομένων 2	Σύνολο δεδομένων 3	Σύνολο δεδομένων 4
1	346,09	348,45	347,5	351,99
2	345,35	348,02	347,64	352,46
3	362,2	342,21	330,4	334,55
4	373,23	345,69	329,97	336,46
5	343,02	346,7	335,44	337,02
6	342,15	349,46	335,9	338,25
7	359,25	353,02	334,98	338,96
8	380,5	351,71	333,91	338,35
9	350,35	361,83	365,07	370,37
10	347,23	360,12	362,57	371,92
11	349,18	363,96	361,89	369,81
Ελάχιστο μέσο απόλυτο σφάλμα	**342,15**	**342,21**	**329,97**	**334,55**
Ελάχιστο μέσο απόλυτο σφάλμα διμηνίας		**329,97**		

Πίνακας 5.7 – Μέσο απόλυτο σφάλμα για την διμηνία Ιανουαρίου - Φεβρουαρίου των 11 πειραμάτων για τα σύνολα δεδομένων 1,2,3 και 4.



Γράφημα 5.7 – Γραφική αναπαράσταση μέσου απολύτου σφάλματος για την διμηνία Ιανουαρίου - Φεβρουαρίου των 11 πειραμάτων για τα σύνολα δεδομένων 1,2,3 και 4

Για να κατανοήσουμε καλύτερα την γραφική παράσταση, παραθέτω σε συντομία τους αλγόριθμους οι οποίοι είχαν εκτελεστεί σε κάθε πείραμα, οι οποίοι έχουν ως εξής:

Στα πειράματα 1 και 2, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στα πειράματα 3 μέχρι 5, είχε εκτελεστεί Γραμμική Παλινδρόμηση.

Στα πειράματα 6 μέχρι 8, είχε εκτελεστεί Βηματική Παλινδρόμηση.

Στα πειράματα 9 μέχρι 11, είχε εκτελεστεί Δίκτυο RBF.

Ανάλυση Αποτελεσμάτων:

Στην συγκεκριμένη ανάλυση, δεν θα ασχοληθούμε με την απόδοση του κάθε αλγορίθμου στο κάθε σύνολο δεδομένων καθώς έχει μελετηθεί στο υποκεφάλαιο 5.3, αλλά θα δούμε την συνολική απόδοση των αλγορίθμων στα 4 σύνολα δεδομένων, για να μελετήσουμε πως η αύξηση δεδομένων επιφέρει καλύτερα αποτελέσματα εάν γίνεται αυτό.

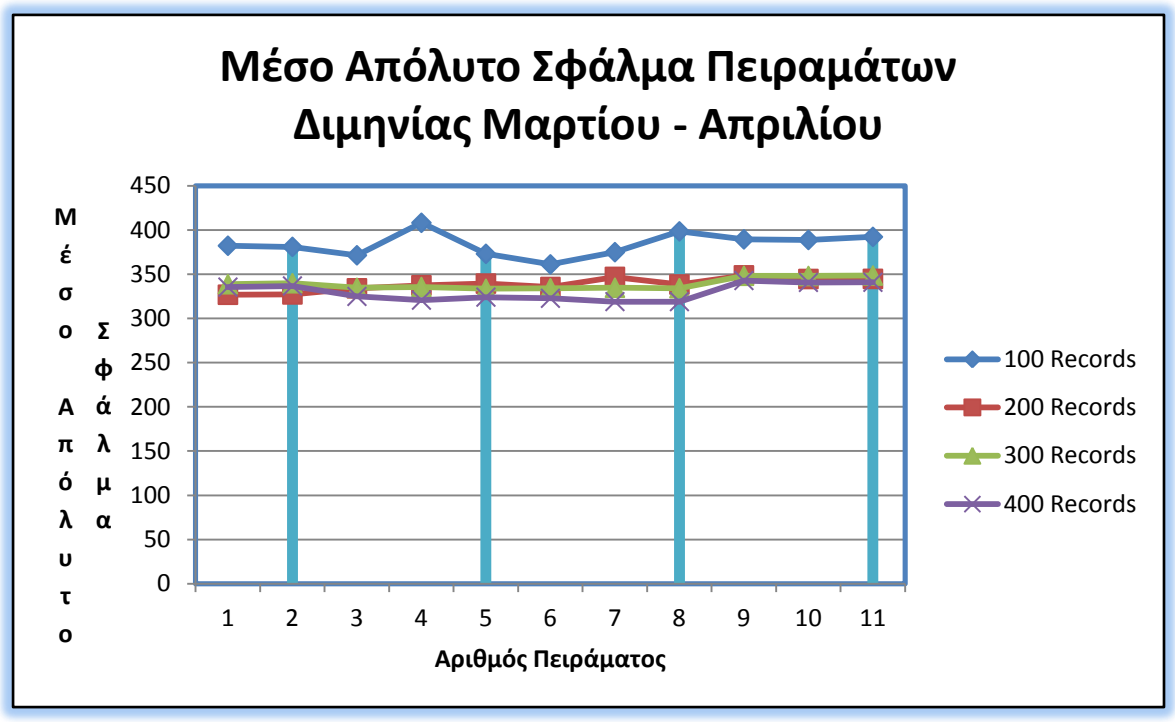
Από την γραφική παράσταση αυτή, μπορούμε να δούμε αρκετή αστάθεια μεταξύ των αλγορίθμων. Τα χειρότερα αποτελέσματα, και τα πιο ασταθή, υπάρχουν στο σύνολο των 100 εγγραφών, όπως και αναμένεται. Αυτό δηλώνει ότι οι αλγόριθμοι έχουν κάποια δυσκολία να μάθουν έναντι των τριών άλλων συνόλων δεδομένων. Στην συνέχεια βλέπουμε ότι το σύνολο δεδομένων με 200 εγγραφές, έχει σαφώς πιο ομαλή μορφή, και καλύτερα αποτελέσματα από ότι στο σύνολο των 100 εγγραφών. Επίσης μπορούμε να μελετήσουμε ότι οι αλγόριθμοι, στα 300 και 400 δεδομένα δεν έχουν πολλή διαφορά, και παρατηρούμε ότι με 300 δεδομένα

υπάρχουν ελαφρώς καλύτερα αποτελέσματα από ότι στα 400. Αυτό δεν είναι το αναμενόμενο, και θα δούμε μελετώντας τις επόμενες γραφικές παραστάσεις ότι αυτό δεν ισχύει γενικά. Πιο συγκεκριμένα, η διαφορά μεταξύ του συνόλου των 100 εγγραφών και του συνόλου των 200 είναι γύρω στις 3 μονάδες. Μεταξύ του συνόλου των 200 εγγραφών και των υπολοίπων δύο, η διαφορά κυμαίνεται γύρω στις 5 μονάδες, ενώ μεταξύ του συνόλου δεδομένων με 300 εγγραφές και του συνόλου δεδομένων με 400 εγγραφές η διαφορά είναι επίσης γύρω στις 5 μονάδες.

5.4.1.2 Διμηνία Μαρτίου - Απριλίου

Μέσο απόλυτο σφάλμα διμηνίας Μαρτίου – Απριλίου				
Α/Α Συνόλου δεδομένων διμηνίας / Α/Α εκτέλεσης πειράματος	Σύνολο δεδομένων 1	Σύνολο δεδομένων 2	Σύνολο δεδομένων 3	Σύνολο δεδομένων 4
1	382,2	326,77	338,68	335,41
2	380,92	327,18	339,8	336,75
3	371,41	334,03	334,64	324,87
4	408,13	337,46	335,57	320,62
5	373,05	339,39	333,54	324,07
6	361,11	335,52	334,23	322,75
7	374,94	346,92	334,71	319,04
8	398,42	338,35	334,17	318,96
9	389,42	348,69	348,22	342,83
10	388,71	344,57	348,01	340,73
11	392,22	344,71	348,49	340,85
Ελάχιστο μέσο απόλυτο σφάλμα	**361,11**	**326,77**	**333,54**	**318,96**
Ελάχιστο μέσο απόλυτο σφάλμα διμηνίας		**318,96**		

Πίνακας 5.8 – Μέσο απόλυτο σφάλμα για την διμηνία Μαρτίου - Απριλίου των 11 πειραμάτων για τα σύνολα δεδομένων 1,2,3 και 4.



Γράφημα 5.8 – Γραφική αναπαράσταση μέσου απολύτου σφάλματος για την διμηνία Μαρτίου - Απριλίου των 11 πειραμάτων για τα σύνολα δεδομένων 1,2,3 και 4

Για να κατανοήσουμε καλύτερα την γραφική παράσταση, παραθέτω σε συντομία τους αλγόριθμους οι οποίοι είχαν εκτελεστεί σε κάθε πείραμα, οι οποίοι έχουν ως εξής:

Στα πειράματα 1 και 2, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στα πειράματα 3 μέχρι 5, είχε εκτελεστεί Γραμμική Παλινδρόμηση.

Στα πειράματα 6 μέχρι 8, είχε εκτελεστεί Βηματική Παλινδρόμηση.

Στα πειράματα 9 μέχρι 11, είχε εκτελεστεί Δίκτυο RBF.

Ανάλυση Αποτελεσμάτων:

Στην συγκεκριμένη ανάλυση, δεν θα ασχοληθούμε με την απόδοση του κάθε αλγορίθμου στο κάθε σύνολο δεδομένων καθώς έχει μελετηθεί στο υποκεφάλαιο 5.3, αλλά θα δούμε την συνολική απόδοση των αλγορίθμων στα 4 σύνολα δεδομένων, για να μελετήσουμε πως η αύξηση δεδομένων επιφέρει καλύτερα αποτελέσματα εάν γίνεται αυτό.

Από την γραφική παράσταση αυτή, μπορούμε να δούμε ότι τα αποτελέσματα των αλγορίθμων βελτιώνονται καθώς αυξάνονται τα δεδομένα. Τα χειρότερα αποτελέσματα, και

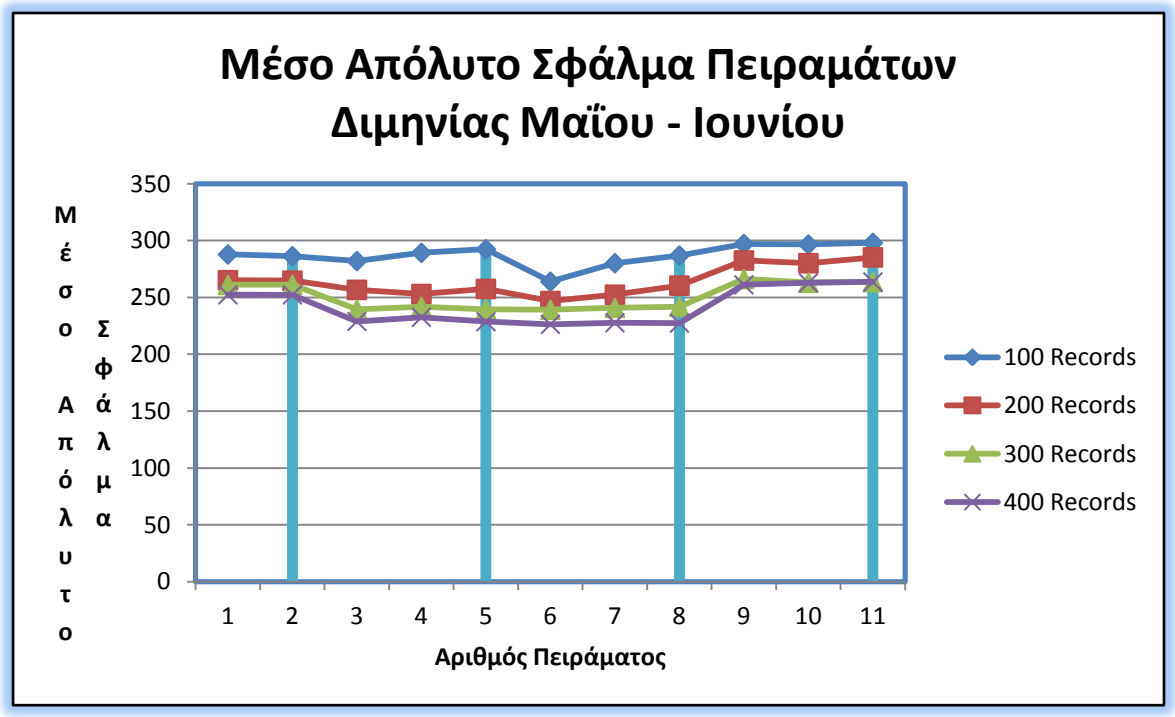
τα πιο ασταθή, υπάρχουν στο σύνολο των 100 εγγραφών, όπως και αναμένεται. Αυτό δηλώνει ότι οι αλγόριθμοι έχουν κάποια δυσκολία να μάθουν έναντι των τριών άλλων συνόλων δεδομένων. Στην συνέχεια βλέπουμε ότι το σύνολο δεδομένων με 200 εγγραφές, έχει σαφώς πιο ομαλή μορφή, και καλύτερα αποτελέσματα από ότι στο σύνολο των 100 εγγραφών. Επίσης μπορούμε να μελετήσουμε ότι οι αλγόριθμοι, στα 400 δεδομένα παρουσιάζουν καλύτερα αποτελέσματα από ότι τα 300 δεδομένα. Πιο συγκεκριμένα, η διαφορά μεταξύ του συνόλου των 100 εγγραφών και του συνόλου των 200 είναι γύρω στις 45 μονάδες. Μεταξύ του συνόλου των 200 εγγραφών και του συνόλου των 300 εγγραφών, η διαφορά κυμαίνεται γύρω στη μία μονάδα, ενώ μεταξύ του συνόλου δεδομένων με 300 εγγραφές και του συνόλου δεδομένων με 400 εγγραφές η διαφορά είναι γύρω στις 10 μονάδες. Από αυτό μπορούμε να συμπεράνουμε ότι με την αύξηση των δεδομένων, έχουμε καλύτερα αποτελέσματα

5.4.1.3 Διμηνία Μαΐου - Ιουνίου

Μέσο απόλυτο σφάλμα διμηνίας Μαΐου – Ιουνίου				
Α/Α Συνόλου δεδομένων διμηνίας / Α/Α εκτέλεσης πειράματος	Σύνολο δεδομένων 1	Σύνολο δεδομένων 2	Σύνολο δεδομένων 3	Σύνολο δεδομένων 4
1	287,91	265,25	261,12	252,46
2	286,38	264,83	261,18	252,32
3	282,02	256,74	239,5	228,99
4	289,3	253,08	241,91	232,61
5	292,44	257,57	239,7	228,75
6	263,94	246,92	239,29	226,4
7	280,16	252,45	240,91	227,81
8	286,92	260,47	241,93	227,5
9	297,13	282,59	266,29	261,21
10	296,63	280,22	263,09	262,89
11	298,15	285,28	263,66	263,64
Ελάχιστο μέσο απόλυτο σφάλμα	**263,94**	**246,92**	**239,29**	**226,4**
Ελάχιστο μέσο		**226,4**		

απόλυτο σφάλμα διμηνίας

Πίνακας 5.9 – Μέσο απόλυτο σφάλμα για την διμηνία Μαΐου - Ιουνίου των 11 πειραμάτων για τα σύνολα δεδομένων 1,2,3 και 4.



Γράφημα 5.9 – Γραφική αναπαράσταση μέσου απολύτου σφάλματος για την διμηνία Μαΐου - Ιουνίου των 11 πειραμάτων για τα σύνολα δεδομένων 1,2,3 και 4

Για να κατανοήσουμε καλύτερα την γραφική παράσταση, παραθέτω σε συντομία τους αλγόριθμους οι οποίοι είχαν εκτελεστεί σε κάθε πείραμα, οι οποίοι έχουν ως εξής:

Στα πειράματα 1 και 2, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στα πειράματα 3 μέχρι 5, είχε εκτελεστεί Γραμμική Παλινδρόμηση.

Στα πειράματα 6 μέχρι 8, είχε εκτελεστεί Βηματική Παλινδρόμηση.

Στα πειράματα 9 μέχρι 11, είχε εκτελεστεί Δίκτυο RBF.

Ανάλυση Αποτελεσμάτων:

Στην συγκεκριμένη ανάλυση, δεν θα ασχοληθούμε με την απόδοση του κάθε αλγορίθμου στο κάθε σύνολο δεδομένων καθώς έχει μελετηθεί στο υποκεφάλαιο 5.3, αλλά θα δούμε την

συνολική απόδοση των αλγορίθμων στα 4 σύνολα δεδομένων, για να μελετήσουμε πως η αύξηση δεδομένων επιφέρει καλύτερα αποτελέσματα εάν γίνεται αυτό.

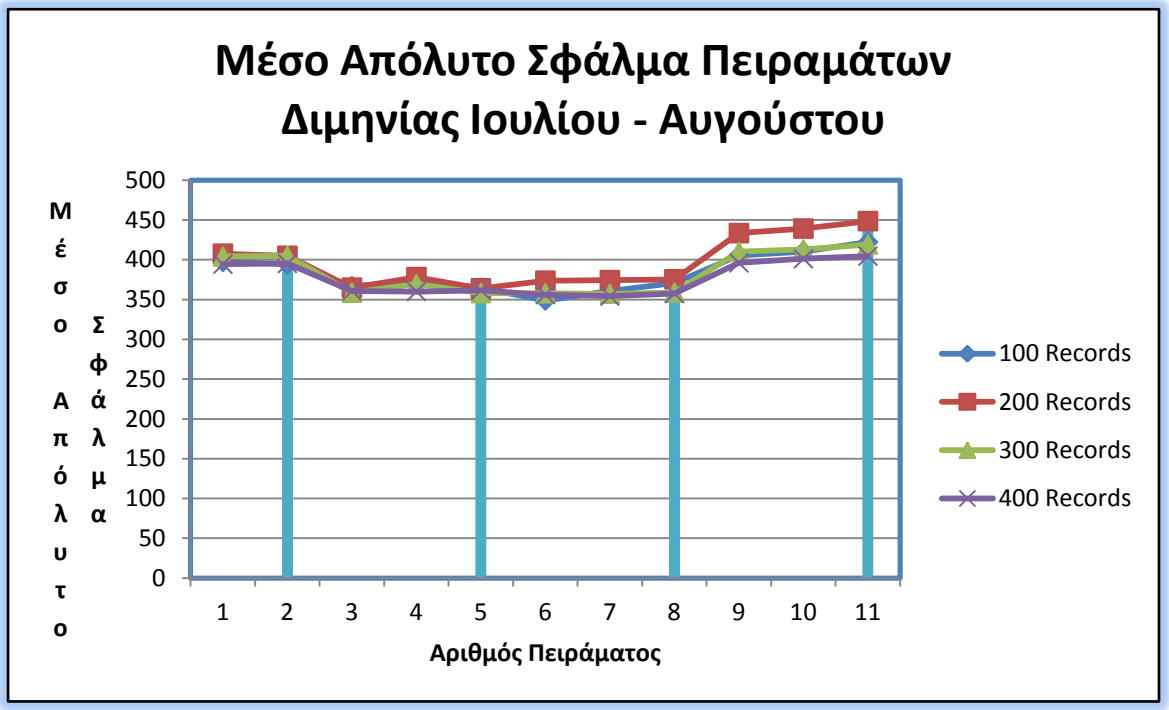
Από την γραφική παράσταση αυτή, μπορούμε να δούμε ότι τα αποτελέσματα των αλγορίθμων βελτιώνονται καθώς αυξάνονται τα δεδομένα. Τα χειρότερα αποτελέσματα, και τα πιο ασταθή, υπάρχουν στο σύνολο των 100 εγγραφών, όπως και αναμένεται. Αυτό δηλώνει ότι οι αλγόριθμοι έχουν κάποια δυσκολία να μάθουν έναντι των τριών άλλων συνόλων δεδομένων. Στην συνέχεια βλέπουμε ότι το σύνολο δεδομένων με 200 εγγραφές, έχει σαφώς πιο ομαλή μορφή, και καλύτερα αποτελέσματα από ότι στο σύνολο των 100 εγγραφών. Επίσης μπορούμε να μελετήσουμε ότι οι αλγόριθμοι, στα 400 δεδομένα παρουσιάζουν καλύτερα αποτελέσματα από ότι τα 300 δεδομένα. Πιο συγκεκριμένα, η διαφορά μεταξύ του συνόλου των 100 εγγραφών και του συνόλου των 200 είναι γύρω στις 22 μονάδες. Μεταξύ του συνόλου των 200 εγγραφών και του συνόλου των 300 εγγραφών, η διαφορά κυμαίνεται γύρω στις 14 μονάδες, ενώ μεταξύ του συνόλου δεδομένων με 300 εγγραφές και του συνόλου δεδομένων με 400 εγγραφές η διαφορά είναι γύρω στις 8 μονάδες. Από αυτό μπορούμε να συμπεράνουμε ότι με την αύξηση των δεδομένων, τα αποτελέσματά μας βελτιώνονται.

5.4.1.4 Διμηνία Ιουλίου - Αυγούστου

Μέσο απόλυτο σφάλμα διμηνίας Ιανουαρίου – Φεβρουαρίου				
Α/Α Συνόλου δεδομένων διμηνίας / Α/Α εκτέλεσης πειράματος	Σύνολο δεδομένων 1	Σύνολο δεδομένων 2	Σύνολο δεδομένων 3	Σύνολο δεδομένων 4
1	397,24	407,54	404,01	394,33
2	394,36	404,88	404,86	395,18
3	366,63	364,9	358,6	360,8
4	373,32	378,15	369,55	360,05
5	364,67	364,01	358,04	361,53
6	348,86	373,61	357,85	356,05
7	360,65	374,32	357,25	354,14
8	370,68	375,4	358,84	357,37
9	405,31	433,41	410,05	396,04
10	410,35	439,18	413,1	401,16
11	422,23	448,49	419,25	404,51

Ελάχιστο μέσο απόλυτο σφάλμα	**348,86**	**364,01**	**357,25**	**354,14**
Ελάχιστο μέσο απόλυτο σφάλμα διμηνίας		**348,86**		

Πίνακας 5.10 – Μέσο απόλυτο σφάλμα για την διμηνία Ιουλίου - Αυγούστου των 11 πειραμάτων για τα σύνολα δεδομένων 1,2,3 και 4.



Γράφημα 5.10 – Γραφική αναπαράσταση μέσου απολύτου σφάλματος για την διμηνία Ιουλίου - Αυγούστου των 11 πειραμάτων για τα σύνολα δεδομένων 1,2,3 και 4

Για να κατανοήσουμε καλύτερα την γραφική παράσταση, παραθέτω σε συντομία τους αλγόριθμους οι οποίοι είχαν εκτελεστεί σε κάθε πείραμα, οι οποίοι έχουν ως εξής:

Στα πειράματα 1 και 2, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στα πειράματα 3 μέχρι 5, είχε εκτελεστεί Γραμμική Παλινδρόμηση.

Στα πειράματα 6 μέχρι 8, είχε εκτελεστεί Βηματική Παλινδρόμηση.

Στα πειράματα 9 μέχρι 11, είχε εκτελεστεί Δίκτυο RBF.

Ανάλυση Αποτελεσμάτων:

Στην συγκεκριμένη ανάλυση, δεν θα ασχοληθούμε με την απόδοση του κάθε αλγορίθμου στο κάθε σύνολο δεδομένων καθώς έχει μελετηθεί στο υποκεφάλαιο 5.3, αλλά θα δούμε την συνολική απόδοση των αλγορίθμων στα 4 σύνολα δεδομένων, για να μελετήσουμε πως η αύξηση δεδομένων επιφέρει καλύτερα αποτελέσματα εάν γίνεται αυτό.

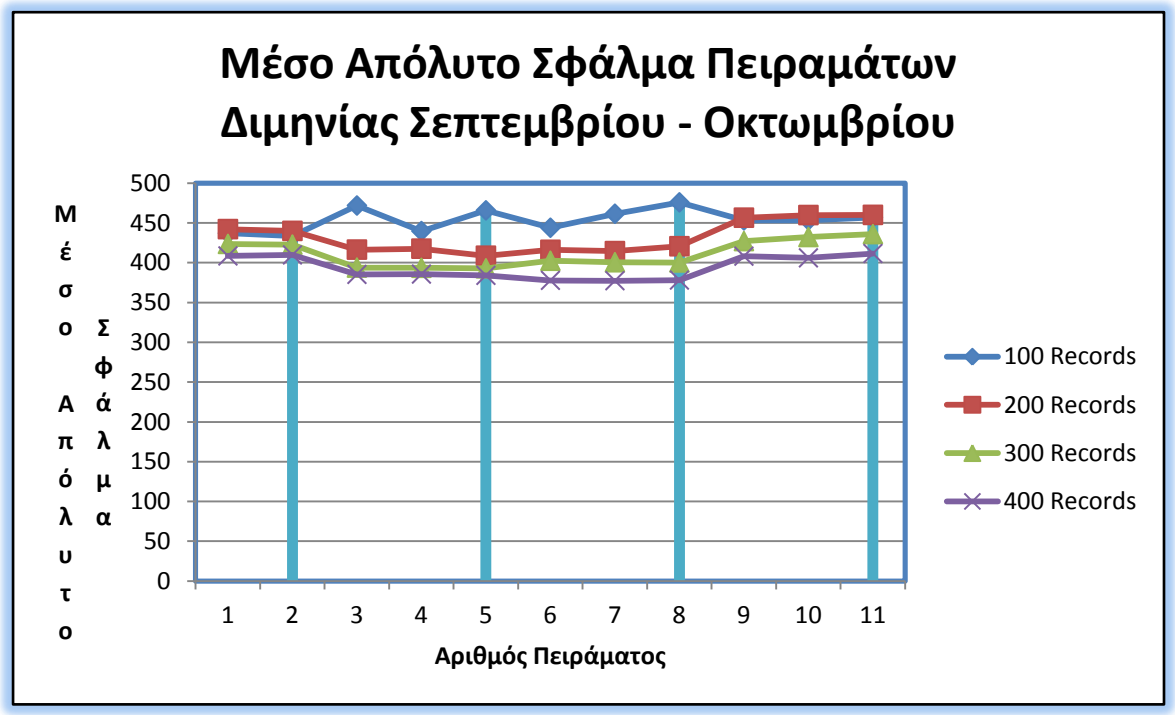
Από την γραφική παράσταση αυτή, μπορούμε να δούμε ότι τα αποτελέσματα των αλγορίθμων βελτιώνονται καθώς αυξάνονται τα δεδομένα. Τα χειρότερα αποτελέσματα, και τα πιο ασταθή, στην περίπτωση αυτή υπάρχουν στο σύνολο των 200 εγγραφών σε αντίθεση με τις άλλες διμηνίες που τα χειρότερα αποτελέσματα υπάρχουν στο σύνολο των 100 εγγραφών. Αυτό δηλώνει ότι οι αλγόριθμοι έχουν κάποια δυσκολία να μάθουν έναντι των τριών άλλων συνόλων δεδομένων. Στην συνέχεια βλέπουμε ότι το σύνολο δεδομένων με 100 εγγραφές, έχει σαφώς πιο ομαλή μορφή, και καλύτερα αποτελέσματα από ότι στο σύνολο των 200 εγγραφών. Επίσης μπορούμε να μελετήσουμε ότι οι αλγόριθμοι, στα 400 δεδομένα παρουσιάζουν καλύτερα αποτελέσματα από ότι τα 300 δεδομένα. Πιο συγκεκριμένα, η διαφορά μεταξύ του συνόλου των 100 εγγραφών και του συνόλου των 200 είναι γύρω στις 10 μονάδες. Μεταξύ του συνόλου των 200 εγγραφών και του συνόλου των 300 εγγραφών, η διαφορά κυμαίνεται γύρω στις 14 μονάδες, ενώ μεταξύ του συνόλου δεδομένων με 300 εγγραφές και του συνόλου δεδομένων με 400 εγγραφές η διαφορά είναι γύρω στις 6 μονάδες. Από αυτό μπορούμε να συμπεράνουμε ότι με την αύξηση των δεδομένων, τα αποτελέσματά μας βελτιώνονται.

5.4.1.5 Διμηνία Σεπτεμβρίου - Οκτωβρίου

Μέσο απόλυτο σφάλμα διμηνίας Σεπτεμβρίου - Οκτωβρίου				
Α/Α Συνόλου δεδομένων διμηνίας / Α/Α εκτέλεσης πειράματος	Σύνολο δεδομένων 1	Σύνολο δεδομένων 2	Σύνολο δεδομένων 3	Σύνολο δεδομένων 4
1	436,97	442,15	423,57	408,44
2	433,46	439,91	422,71	409,71
3	471,78	416,42	393,82	385,23
4	439,71	417,58	393,55	385,62
5	465,72	408,69	392,86	384,02
6	443,87	416,13	402,38	377,75
7	461,46	414,48	400,54	377,33

8	475,77	420,64	400,25	378,1
9	453,06	456,51	427,1	408,12
10	453,23	459,83	432,24	406
11	456,85	460,01	435,88	411,5
Ελάχιστο μέσο απόλυτο σφάλμα	**433,46**	**408,69**	**392,86**	**377,33**
Ελάχιστο μέσο απόλυτο σφάλμα διμηνίας		**377,33**		

Πίνακας 5.11 – Μέσο απόλυτο σφάλμα για την διμηνία Σεπτεμβρίου-Οκτωμβρίου των 11 πειραμάτων για τα σύνολα δεδομένων 1,2,3 και 4.



Γράφημα 5.11 – Γραφική αναπαράσταση μέσου απολύτου σφάλματος για την διμηνία Σεπτεμβρίου - Οκτωβρίου των 11 πειραμάτων για τα σύνολα δεδομένων 1,2,3 και 4

Για να κατανοήσουμε καλύτερα την γραφική παράσταση, παραθέτω σε συντομία τους αλγόριθμους οι οποίοι είχαν εκτελεστεί σε κάθε πείραμα, οι οποίοι έχουν ως εξής:

Στα πειράματα 1 και 2, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στα πειράματα 3 μέχρι 5, είχε εκτελεστεί Γραμμική Παλινδρόμηση.

Στα πειράματα 6 μέχρι 8, είχε εκτελεστεί Βηματική Παλινδρόμηση.

Στα πειράματα 9 μέχρι 11, είχε εκτελεστεί Δίκτυο RBF.

Ανάλυση Αποτελεσμάτων:

Στην συγκεκριμένη ανάλυση, δεν θα ασχοληθούμε με την απόδοση του κάθε αλγορίθμου στο κάθε σύνολο δεδομένων καθώς έχει μελετηθεί στο υποκεφάλαιο 5.3, αλλά θα δούμε την συνολική απόδοση των αλγορίθμων στα 4 σύνολα δεδομένων, για να μελετήσουμε πως η αύξηση δεδομένων επιφέρει καλύτερα αποτελέσματα εάν γίνεται αυτό.

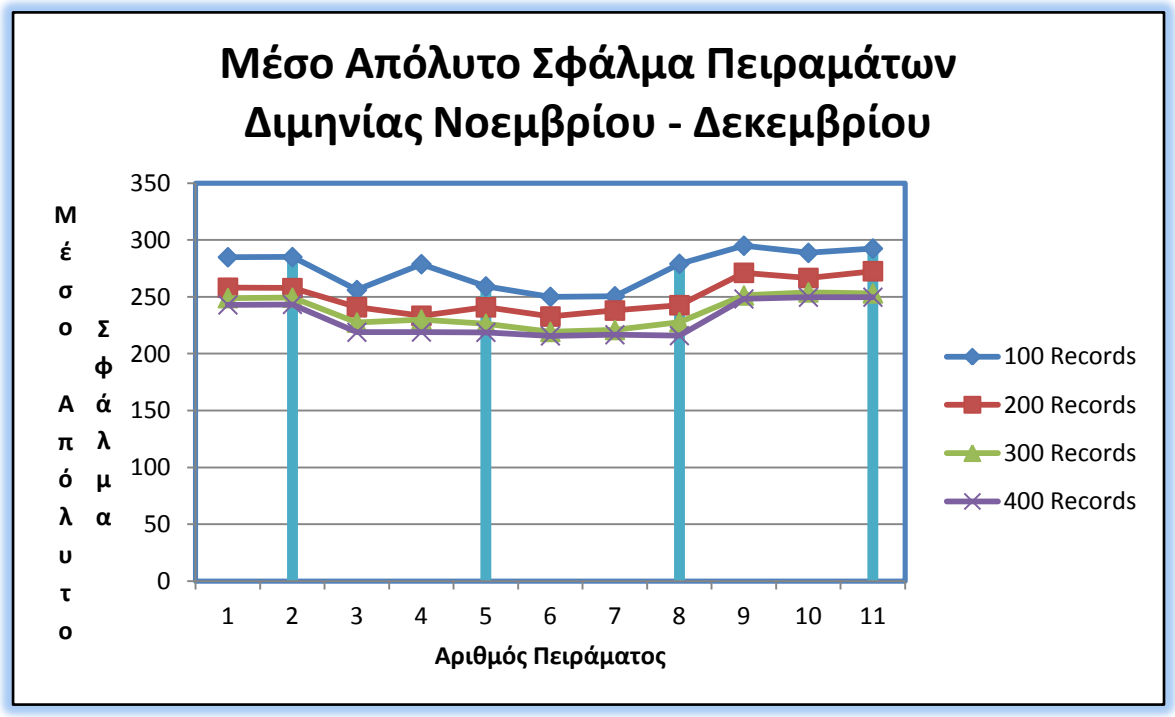
Από την γραφική παράσταση αυτή, μπορούμε να δούμε ότι τα αποτελέσματα των αλγορίθμων βελτιώνονται καθώς αυξάνονται τα δεδομένα. Τα χειρότερα αποτελέσματα, και τα πιο ασταθή, υπάρχουν στο σύνολο των 100 εγγραφών, όπως και αναμένεται. Αυτό δηλώνει ότι οι αλγόριθμοι έχουν κάποια δυσκολία να μάθουν έναντι των τριών άλλων συνόλων δεδομένων. Στην συνέχεια βλέπουμε ότι το σύνολο δεδομένων με 200 εγγραφές, έχει σαφώς πιο ομαλή μορφή, και καλύτερα αποτελέσματα από ότι στο σύνολο των 100 εγγραφών. Επίσης μπορούμε να μελετήσουμε ότι οι αλγόριθμοι, στα 400 δεδομένα παρουσιάζουν καλύτερα αποτελέσματα από ότι τα 300 δεδομένα. Πιο συγκεκριμένα, η διαφορά μεταξύ του συνόλου των 100 εγγραφών και του συνόλου των 200 είναι γύρω στις 21 μονάδες. Μεταξύ του συνόλου των 200 εγγραφών και του συνόλου των 300 εγγραφών, η διαφορά κυμαίνεται γύρω στις 21 μονάδες, ενώ μεταξύ του συνόλου δεδομένων με 300 εγγραφές και του συνόλου δεδομένων με 400 εγγραφές η διαφορά είναι γύρω στις 18 μονάδες. Από αυτό μπορούμε να συμπεράνουμε ότι με την αύξηση των δεδομένων, τα αποτελέσματά μας βελτιώνονται.

5.4.1.6 Διμηνία Νοεμβρίου - Δεκεμβρίου

Μέσο απόλυτο σφάλμα διμηνίας Νοεμβρίου - Δεκεμβρίου				
Α/Α Συνόλου δεδομένων διμηνίας / Α/Α εκτέλεσης πειράματος	Σύνολο δεδομένων 1	Σύνολο δεδομένων 2	Σύνολο δεδομένων 3	Σύνολο δεδομένων 4
1	284,92	258,05	248,97	242,98
2	285,05	257,74	249,34	243,35
3	256,05	240,94	227,47	218,89
4	278,81	233,43	230,07	219,08

5	259,24	240,87	226,39	218,86
6	250,07	232,77	219,27	215,69
7	250,57	238,02	220,95	216,75
8	279,11	242,71	227,82	215,94
9	295,04	271,2	251,32	248,22
10	288,74	266,68	254,02	249,7
11	292,54	272,64	253,16	249,77
Ελάχιστο μέσο απόλυτο σφάλμα	**250,07**	**232,77**	**219,27**	**215,69**
Ελάχιστο μέσο απόλυτο σφάλμα διμηνίας		**215,69**		

Πίνακας 5.12 – Μέσο απόλυτο σφάλμα για την διμηνία Νοεμβρίου - Δεκεμβρίου των 11 πειραμάτων για τα σύνολα δεδομένων 1,2,3 και 4.



Γράφημα 5.12 – Γραφική αναπαράσταση μέσου απολύτου σφάλματος για την διμηνία Νοεμβρίου-Δεκεμβρίου των 11 πειραμάτων για τα σύνολα δεδομένων 1,2,3 και 4

Για να κατανοήσουμε καλύτερα την γραφική παράσταση, παραθέτω σε συντομία τους αλγόριθμους οι οποίοι είχαν εκτελεστεί σε κάθε πείραμα, οι οποίοι έχουν ως εξής:

Στα πειράματα 1 και 2, είχαν εκτελεστεί οι Γκαουσιανές διαδικασίες.

Στα πειράματα 3 μέχρι 5, είχε εκτελεστεί Γραμμική Παλινδρόμηση.

Στα πειράματα 6 μέχρι 8, είχε εκτελεστεί Βηματική Παλινδρόμηση.

Στα πειράματα 9 μέχρι 11, είχε εκτελεστεί Δίκτυο RBF.

Ανάλυση Αποτελεσμάτων:

Στην συγκεκριμένη ανάλυση, δεν θα ασχοληθούμε με την απόδοση του κάθε αλγορίθμου στο κάθε σύνολο δεδομένων καθώς έχει μελετηθεί στο υποκεφάλαιο 5.3, αλλά θα δούμε την συνολική απόδοση των αλγορίθμων στα 4 σύνολα δεδομένων, για να μελετήσουμε πως η αύξηση δεδομένων επιφέρει καλύτερα αποτελέσματα εάν γίνεται αυτό.

Από την γραφική παράσταση αυτή, μπορούμε να δούμε ότι τα αποτελέσματα των αλγορίθμων βελτιώνονται καθώς αυξάνονται τα δεδομένα. Τα χειρότερα αποτελέσματα, και τα πιο ασταθή, υπάρχουν στο σύνολο των 100 εγγραφών, όπως και αναμένεται. Αυτό δηλώνει ότι οι αλγόριθμοι έχουν κάποια δυσκολία να μάθουν έναντι των τριών άλλων συνόλων δεδομένων. Στην συνέχεια βλέπουμε ότι το σύνολο δεδομένων με 200 εγγραφές, έχει σαφώς πιο ομαλή μορφή, και καλύτερα αποτελέσματα από ότι στο σύνολο των 100 εγγραφών. Επίσης μπορούμε να μελετήσουμε ότι οι αλγόριθμοι, στα 400 δεδομένα παρουσιάζουν καλύτερα αποτελέσματα από ότι τα 300 δεδομένα. Πιο συγκεκριμένα, η διαφορά μεταξύ του συνόλου των 100 εγγραφών και του συνόλου των 200 είναι γύρω στις 24 μονάδες. Μεταξύ του συνόλου των 200 εγγραφών και του συνόλου των 300 εγγραφών, η διαφορά κυμαίνεται γύρω στις 13 μονάδες, ενώ μεταξύ του συνόλου δεδομένων με 300 εγγραφές και του συνόλου δεδομένων με 400 εγγραφές η διαφορά είναι γύρω στις 7 μονάδες. Από αυτό μπορούμε να συμπεράνουμε ότι με την αύξηση των δεδομένων, τα αποτελέσματά μας βελτιώνονται.

Συμπεράσματα υποκεφαλαίου: Από τα πειράματα τα οποία είχαμε διεξάγει με 4 διαφορετικούς τύπους αλγορίθμων, σε 4 διαφορετικά σύνολα δεδομένων, και στις έξι διμηνίες του 2012, είχαμε παρατηρήσει, ότι όσο περισσότερα είναι τα δεδομένα, τόσο περισσότερο μειώνεται το μέσο απόλυτο σφάλμα. Αυτό σημαίνει ότι στην περίπτωση που αποκτήσουμε ακόμη περισσότερα δεδομένα, οι αλγόριθμοί μας, θα είναι σε θέση να ανακαλύπτουν ακόμη καλύτερα τις σχέσεις μεταξύ των δεδομένων μας, και τις σχέσεις μεταξύ των δεδομένων εισόδου, και επιθυμητής εξόδου

Κεφάλαιο 6

Συμπεράσματα

6.1 Σύνοψη και συμπεράσματα μελέτης

Στην διπλωματική αυτή εργασία, είχαμε συλλέξει στατικά χαρακτηριστικά μιας οικίας, όπως για παράδειγμα το μέγεθός της, ή τις συσκευές κλιματισμού και θέρμανσής τις οποίες διαθέτει, και προσπαθήσαμε μέσω αυτών να εκτιμήσουμε την ηλεκτρική κατανάλωση της οικίας.

Είχαμε χρησιμοποιήσει σωρεία από αλγορίθμους μηχανικής μάθησης, και είχαμε εκτελέσει δεκάδες χιλιάδες πειράματα, ούτως ώστε να καλύψουμε όσο το δυνατό περισσότερες περιπτώσεις, και να πάρουμε όσο το δυνατό πιο ορθά αποτελέσματα. Σαν είσοδο στους αλγορίθμους είχαμε περάσει τρία διαφορετικά σύνολα δεδομένων για κάθε διμηνία, τα οποία περιγράφονται στο κεφάλαιο 5.

Για αρχή, είχαμε χρησιμοποιήσει δύο διαφορετικά Δίκτυα MLP τα οποία είχα υλοποιήσει στην γλώσσα προγραμματισμού JAVA, ένα για παραγωγή συνάρτησης, και ένα για κατηγοριοποίηση των δεδομένων. Είχαμε βρει ότι με το νευρωνικό δίκτυο συνάρτησης, για την διμηνία Ιανουαρίου – Φεβρουαρίου -η οποία έχει τυπική απόκλιση 645- μέσο απόλυτο σφάλμα 307 με την χρήση του συνόλου δεδομένων 2. Για την επόμενη διμηνία η οποία έχει τυπική απόκλιση 644- με την χρήση νευρωνικού δικτύου, είχαμε βρει σφάλμα 306 με την χρήση του τρίτου συνόλου δεδομένων. Στην συνέχεια είχαμε εκτελέσει το νευρωνικό δίκτυο για την διμηνία Μαΐου – Ιουνίου, η οποία έχει τυπική απόκλιση 381. Εκεί είχαμε βρει μέσο απόλυτο σφάλμα 205,91. Όσον αφορά την διμηνία Ιουλίου – Αυγούστου, η οποία έχει τυπική απόκλιση 741, το μέσο απόλυτο σφάλμα ήταν στα 319,68 με την χρήση του δεύτερου συνόλου δεδομένων. Η διμηνία Σεπτεμβρίου – Οκτωβρίου, με την χρήση του τρίτου συνόλου

δεδομένων είχε μέσο απόλυτο σφάλμα 369,78, ενώ η τυπική της απόκλιση είναι 725. Στην τελευταία διμηνία, η οποία είχε και τυπική απόκλιση 360, με την χρήση του τρίτου συνόλου, είχαμε πετύχει μέσο απόλυτο σφάλμα 218,21.

Είχαμε ακόμη σκεφτεί, στα πλαίσια των πειραμάτων μας, ότι ίσως να μην ενδιαφέρει σε τόσο μεγάλο βαθμό τον χρήστη να πάρει από τους αλγορίθμους μία τιμή της εκτίμησης της κατανάλωσης του σε κιλοβατώρες, αλλά να τον κατατάξει σε μια κατηγορία καταναλώσεων. Αυτό σημαίνει ότι, αντί για παράδειγμα να αναφέρει ο αλγόριθμος στον χρήστη, ότι αναμένεται να καταναλώσει 1119KW, να του αναφέρει ότι είναι στην κατηγορία 1000-1200 KW. Αυτό ίσως να είναι πιο χρήσιμο για τον χρήστη, αλλά ίσως και να είχε καλύτερα αποτελέσματα, αφού έτσι, δεν θα αναμένεται από τον αλγόριθμο να παράξει μία συγκεκριμένη τιμή αλλά μία κατηγορία. Έτσι, είχαμε ακόμη εκτελέσει πειράματα για κατηγοριοποίηση των δεδομένων σε 4 κατηγορίες με την χρήση του δεύτερου νευρωνικού το οποίο είχαμε κατασκευάσει. Παρόλα αυτά δεν υπήρξαν καλά ποσοστά επιτυχίας, και άρα κατηγοριοποίησης της σωστής οικίας στην σωστή ενεργειακή κατηγορία. Τα αποτελέσματα του αλγορίθμου αυτού, έχουν παρουσιαστεί στην διπλωματική αυτή εργασία, λόγω του ότι θέλουμε να δείξουμε τον τρόπο ακριβώς με τον οποίο είχαμε εργαστεί, καθώς επίσης να διαφανεί ότι είχαμε μελετήσει και στραφεί και άλλες προσεγγίσεις εκτός από την παλινδρόμηση (regression) για επίλυση του προβλήματός μας. Ταυτόχρονα μπορούμε να παρατηρήσουμε ότι η προσέγγιση μηχανικής μάθησης με την οποία αναλύουμε ένα πρόβλημα, είναι εξαιρετικά σημαντική, καθώς επίσης επηρεάζει τα αποτελέσματά μας στην συνέχεια.

Έτσι στην συνέχεια, είχαμε εκτελέσει πειράματα με την χρήση των αλγορίθμων των Γκαουσιανών Διαδικασιών, Ισοτονική Παλινδρόμηση, Ελαχίστου τετραγώνου των μέσων, γραμμικής παλινδρόμησης, βηματικής παλινδρόμησης, δίκτυα RBF, Απλή γραμμική παλινδρόμηση, καθώς και ταξινομητή PLS. Από τις χιλιάδες εκτελέσεις των αλγορίθμων αυτών, είχαμε βρει ότι όχι τόσο καλά αποτελέσματα είχαν το δίκτυο RBF, ενώ τα καλύτερα είχαν ο αλγόριθμος του Ελαχίστου τετραγώνου των μέσων καθώς επίσης και η βηματική παλινδρόμηση

Μέσω των πειραμάτων αυτών, είχαμε βρει ότι το σύνολο δεδομένων 2, το οποίο περιέχει τα δεδομένα ως έχουν, με διαγραφή του «θορύβου», είχε καλύτερα αποτελέσματα τόσο από το σύνολο 1 όσο και από το σύνολο 3. Επίσης θα πρέπει να αναφέρουμε και τα αποτελέσματα τα οποία προκύπτουν.

Για την διμηνία Ιανουαρίου – Φεβρουαρίου χρησιμοποιήσαμε τον αλγόριθμο του ελαχίστου μέσου και είχαμε μέσο ελάχιστο σφάλμα 330,97.

Για την διμηνία Μαρτίου - Απριλίου χρησιμοποιήσαμε τον αλγόριθμο ελαχίστου τετραγωνικού μέσου και είχαμε μέσο ελάχιστο σφάλμα 315,68
Για την διμηνία Μαΐου-Ιουνίου χρησιμοποιήσαμε τον αλγόριθμο βηματικής παλινδρόμησης και είχαμε μέσο ελάχιστο σφάλμα 235,31.
Για την διμηνία Ιουλίου - Αυγούστου χρησιμοποιήσαμε τον αλγόριθμο της βηματικής παλινδρόμησης και είχαμε μέσο ελάχιστο σφάλμα 353,99.
Για την διμηνία Σεπτεμβρίου - Οκτωβρίου χρησιμοποιήσαμε τον αλγόριθμο ελάχιστου τετραγωνικού μέσου και είχαμε μέσο ελάχιστο σφάλμα 375,57.
Για την διμηνία Νοεμβρίου - Δεκεμβρίου χρησιμοποιήσαμε τον αλγόριθμο της βηματικής παλινδρόμησης και είχαμε μέσο ελάχιστο σφάλμα 216,02.

Έτσι μπορούμε να δούμε ότι είχαμε καταφέρει να πάρουμε αποτελέσματα αρκετά καλά, τα οποία είναι και πολλές φορές καλύτερα και από το μισό της τυπικής απόκλισης.

Τα αποτελέσματα των αλγορίθμων αυτών, δηλαδή οι συναρτήσεις οι οποίες έχουν κατασκευαστεί από τους αλγορίθμους έχουν υλοποιηθεί στην γλώσσα C#, και αποτελούν πλέον μέρος της λειτουργίας προσωπικές συγκρίσεις της εφαρμογής κοινωνικός ηλεκτρισμός.

Εμπειρική ανάλυση των αποτελεσμάτων και σχολιασμός

Τα δεδομένα όπως και μπορούμε να παρατηρήσουμε στο Κεφάλαιο 2, είναι αραιά (sparse). Για παράδειγμα, ο αριθμός των ανθρώπων που έχουν πισίνα είναι κατά πολύ λιγότερος σε σχέση με τους ανθρώπους που δεν έχουν, και το σύνολο των εγγραφών μας δείχνει ότι τα σπίτια περιέχουν πολύ λιγότερα χαρακτηριστικά από όσα υπάρχουν.

Μπορούμε ακόμη να παρατηρήσουμε, στο Κεφάλαιο 5, ότι οι αλγόριθμοι οι οποίοι έχουν το πιο ψηλό μέσο απόλυτο σφάλμα, είναι οι αλγόριθμοι οι οποίοι δεν παράγουν μια ευθεία γραμμή, επίπεδο ή πολυεπίπεδο ως συνάρτηση των δεδομένων εισόδου με τα επιθυμητά αποτελέσματα, αλλά μια συνάρτηση, η οποία προσαρμόζεται περισσότερο στα δεδομένα. Τέτοιοι αλγόριθμοι, οι οποίοι ξεχωρίζουν αρκετά για την ελαφρώς χειρότερη απόδοσή τους, είναι οι Γκαουσιανές Διαδικασίες, καθώς επίσης και τα Δίκτυα RBF. Αλγόριθμοι όπως την γραμμική παλινδρόμηση, ή την βηματική παλινδρόμηση, οι οποίες δημιουργούν μία γραμμική συνάρτηση, σε δυσδιάστατο χώρο, επίπεδο ή πολυεπίπεδο έχουν αρκετά καλύτερα αποτελέσματα. Αυτό ίσως να μην φαντάζει τόσο λογικό, λόγω της φύσης των δεδομένων

μας, τα οποία δεν φαίνονται να είναι γραμμικά σε σχέση με την ηλεκτρική κατανάλωση μίας οικίας.

Από την άλλη, το νευρωνικό δίκτυο το οποίο έχω αναπτύξει εγώ έχει καλύτερα αποτελέσματα από κάθε άλλο αλγόριθμο, είτε παράγει ευθεία συνάρτηση, είτε προσαρμόζεται στα δεδομένα και δημιουργεί μη ευθείες συναρτήσεις. Επίσης να αναφέρουμε ότι το νευρωνικό δίκτυο δεν παράγει ευθείες συναρτήσεις (εκτός αν εμείς το θέσουμε έτσι), αλλά παράγει συναρτήσεις τις οποίες ακολουθούν την μορφή των δεδομένων.

Τα δεδομένα μας, πράγματι δεν είναι γραμμικά, και για τον λόγο αυτό το μέσο απόλυτο σφάλμα το οποίο προκύπτει από το νευρωνικό δίκτυο είναι αρκετά μικρότερο. Συνήθως οι αλγόριθμοι οι οποίοι προσπαθούν να προσαρμοστούν θα δεδομένα, έχουν ένα πολύ μεγάλο σύνολο παραμέτρων οι οποίες τα διέπουν. Για παράδειγμα για το Δίκτυο RBF, πρέπει να καθοριστούν αρκετές διαφορετικές μεταβλητές, από τον αρχιτέκτονα του δικτύου όπως τα κέντρα για κάθε νευρώνα, οι αριθμοί των νευρώνων, η τιμή της διασποράς (σ) η οποία και μεταβάλει το μέγεθος της επιρροής που έχει κάθε σημείο του συνόλου εκπαίδευσης στον τελικό σχηματισμό της συνάρτησης που απαιτούμε, καθώς επίσης τρεις ρυθμούς μάθησης, οι οποίοι ρυθμίζουν τον ρυθμό με τον οποίο μεταβάλλονται τα βάρη, η διασπορά καθώς επίσης τα κέντρα. Συνεπώς υπάρχει ένα μεγάλο διάνυσμα μεταβλητών, το οποίο πρέπει να επιλεγεί σωστά, και θα πρέπει να προσέχουμε ακριβώς τι αντίκτυπο έχει η κάθε παράμετρος στην όλη επίδοση του δικτύου. Οι παράμετροι τις οποίες χρειάζεται το δίκτυο RBF, είναι ικανές να το βοηθήσουν να μάθει σε αρκετά καλό βαθμό τα δεδομένα, και την σχέση τους με το επιθυμητό αποτέλεσμα.

Έτσι θα αναρωτηθούμε, γιατί αυτοί οι αλγόριθμοι έχουν χαμηλότερη απόδοση από τους γραμμικούς, σε δεδομένα τα οποία ίσως δεν έχουν γραμμική σχέση με την έξοδο, ενώ επίσης τα νευρωνικά δίκτυα τα οποία είχα υλοποιήσει έχουν καλύτερα αποτελέσματα. Μία υπόθεση η οποία δίνει μία πολύ λογική εξήγηση για αυτό, έγκειται στο γεγονός ότι οι αλγόριθμοι τους οποίους είχαμε τρέξει στο κεφάλαιο 5, ήταν αλγόριθμοι οι οποίοι μας δίνονταν ως έχει από το εργαλείο WEKA. Οι εξωτερικές επιδράσεις καθώς επίσης και οι παράμετροι τις οποίες δέχονταν οι αλγόριθμοι της WEKA, ήταν ελάχιστες, και δεν μπορούσαμε να μελετήσουμε ακριβώς πως επιδρά κάθε παράμετρος του αλγορίθμου στα δεδομένα μας. Οι περισσότερες παράμετροι, είτε δεν είχαν υλοποιηθεί στους αλγορίθμους της WEKA (π.χ. μεταβολή της διασποράς), είτε οι παράμετροι ήταν ενσωματωμένοι (hardcoded) στον κώδικα (πχ τα βάρη - μία από τις πλέον σημαντικότερες παραμέτρους του δικτύου RBF - διαλέγονταν από την WEKA, και όχι από τον αρχιτέκτονα του δικτύου, δηλαδή εμάς). Έτσι δεν μας δινόταν η

ευκαιρία να μελετήσουμε την συμπεριφορά του δικτύου σε βάθος, και το δίκτυο δεν μπορούσε να βρει την βέλτιστη λύση. Στο Νευρωνικό Δίκτυο τύπου MLP λόγω του ότι αναπτύχθηκε από εμάς, είχαμε αυτή την δυνατότητα, και την είχαμε αξιοποιήσει πλήρως, στο να πάρουμε όσο δυνατό καλύτερα αποτελέσματα.

Συνεπώς το γεγονός ότι είχαμε υλοποιήσει και τους δικούς μας αλγορίθμους, είχε πολύ θετικό αντίκτυπο, μιας και μπορούσαμε να το προσαρμόζουμε όπως χρειάζεται, για να μας παράξει την καλύτερη λύση μέσα από τα δεδομένα μας.

Επίσης, οι εγγραφές τις οποίες είχαμε αγγίζουν τις 500. Σε ένα πρόβλημα, το οποίο απαιτεί την χρήση μηχανικής μάθησης για να επιλυθεί, ο αριθμός των εγγραφών τις οποίες είχαμε εμείς, είναι αρκετά μικρός. Είχαμε δει ακόμη, μέσα από την διπλωματική εργασία αυτή, ότι τα αποτελέσματα τα οποία παράγονται από τους αλγορίθμους έχουν σημαντική βελτίωση με την αύξηση των δεδομένων. Συνεπώς έχουμε δείξει ότι, πράγματι μπορεί ένας αλγόριθμος μάθησης να συντελέσει στην εκμάθηση και στην σύνδεση της ηλεκτρικής κατανάλωσης μίας οικίας με τα χαρακτηριστικά της, με σκοπό να μπορεί να αναπαράξει την τιμή της κατανάλωσης βασισμένος στη σχέση αυτή σε μία νέα οικία.

Επίσης το πρόβλημα αυτό, ήταν ιδιαίτερα δύσκολο να λυθεί, και ίσως είναι ένας από τους λόγους, που δεν είχαμε ακόμη καλύτερα αποτελέσματα. Αυτό επειδή δεν είναι τα χαρακτηριστικά μίας οικίας ως επί το πλείστον που καθορίζουν την ηλεκτρική της κατανάλωση, αλλά ιδιαίτερα ο ανθρώπινος παράγοντας. Με αυτό εννοώ ότι, όλοι ίσως να έχουμε μεγάλο ή μικρό σπίτι. Ποιοι όμως για παράδειγμα όταν απομακρύνονται από ένα δωμάτιο του σπιτιού τους σβήνουν και τα φώτα? Αυτό αποτελεί την ενεργειακή συνείδηση του κάθε ανθρώπου, ένα παράγοντα ο οποίος δεν είναι μετρήσιμος, και έτσι δεν έχει ληφθεί υπόψην στις μελέτες μας.

Επίσης το γεγονός ότι δεν έχουμε δυναμικά δεδομένα για κάθε σπίτι, όπως για παράδειγμα δεδομένα τα οποία θα μας έδιναν αισθητήρες, είτε θερμοκρασίας είτε η πληροφορία για κάθε στιγμή πόσοι κάτοικοι βρίσκονται στο σπίτι, καθώς επίσης το γεγονός ότι έχουμε τα δεδομένα σε μόνο 6 σύνολα μέσα στον χρόνο (τις διμηνίες) δυσχεραίνει τις δυνατότητές μας, στο να πάρουμε ακόμη καλύτερες τιμές. Εφόσον με στατικά δεδομένα είχαμε τέτοια ποσοστά επιτυχίας, τότε με realtime δεδομένα, είτε κατανάλωσης, είτε μεταβλητών χαρακτηριστικών της οικίας, τότε θα είχαμε ακόμη καλύτερα αποτελέσματα.

6.2 Μελλοντική Εργασία

Η εργασία αυτή, έχει καταφέρει να δείξει, ότι η μηχανική μάθηση, είναι σε θέση να εκτιμήσει την ηλεκτρική κατανάλωση μίας οικίας με βάση στατικά δεδομένα ενός σπιτιού. Οι εκπαιδεύσεις των αλγορίθμων, έγιναν στην ουσία σε έξι ανεξάρτητα σύνολα δεδομένων, το κάθε ένα από τα οποία αφορούσε τις καταναλώσεις για μία συγκεκριμένη διμηνία.

Τι θα γινόταν όμως, εάν στην διάθεσή μας, υπήρχαν δυναμικά δεδομένα? Στην περίπτωση για παράδειγμα που είχαμε τα δεδομένα για μία οικία καθημερινά, ή ακόμη και σε πραγματικό χρόνο, τότε στην εκπαίδευση των αλγορίθμων μας, θα έμπενε ακόμη μία παράμετρος. Ο χρόνος. Η παράμετρος αυτή είναι ιδιαίτερα σημαντική, γιατί πλέον η συνάρτησή μας, θα χρονική πλέον, η οποία θα πάραγε ακόμη πιο ακριβή αποτελέσματα. Αυτό σημαίνει ότι αντί να είναι σε θέση ο αλγόριθμος να παράγει μία στατική τιμή για την παρούσα διμηνία, να παράγει και αποτελέσματα ακόμη και για τις **μελλοντικές ημέρες,** όπως δηλαδή παράγουν μεγάλες εταιρείες την ζήτηση για πετρέλαιο ή φυσικό αέριο για τις επόμενες μέρες με βάση έξυπνα συστήματα.

Για να το θέσω πιο απλά. Για κάθε οικία, είχα 6 τιμές, οι οποίες λόγω του εξαιρετικά μικρού τους αριθμού δεν μπορούν να συνδυαστούν μεταξύ τους. Παράγετε δηλαδή μια συσχέτιση χαρακτηριστικών ενός σπιτιού, με μια τιμή ηλεκτρικής κατανάλωσης. Εάν για παράδειγμα είχαμε στην διάθεση μας την ηλεκτρική κατανάλωση μίας οικίας για κάθε μέρα, ή ακόμη και πραγματικού χρόνου, θα είχαμε *μία ακολουθία από ηλεκτρικές καταναλώσεις* της οικίας. Οι ακολουθίες περιέχουν επίσης μεγάλα ποσά πληροφορίας, και ένας αλγόριθμος θα ήταν σε θέση να παράγει μία τιμή ηλεκτρικής κατανάλωσης την οποία εκτιμάται να έχει το σπίτι ακόμη και για τις επόμενες μέρες. Δηλαδή θα μπορούσε ο αλγόριθμος να βρει τα trends[35] της ηλεκτρικής κατανάλωσης της οικίας.

Αυτό θα ήταν ένα μεγάλο προνόμιο και για την εφαρμογή Κοινωνικός Ηλεκτρισμός. Είναι *η ιδανική λειτουργία*, η οποία μπορεί να παρέχεται από την εφαρμογή, και η οποία δίνει φυσικά ένα εξαιρετικά σημαντικό κίνητρο σε οποιοδήποτε, να συνδέεται με την εφαρμογή, ή ακόμη να την έχει εγκατεστημένη ακόμη και στο προσωπικό του κινητό τηλέφωνο. Αυτό ίσως να μην γίνεται στην Κύπρο ακόμη, αλλά στο εξωτερικό η άμεση ενημέρωση μίας οικίας είναι ήδη γεγονός[36], ή πρόκειται να αναπτυχθεί στα επόμενα χρόνια. Άρα αυτό αποτελεί μία

μοναδική ευκαιρία για μελλοντική εργασία, και συνέχεια του έργου του οποίου έχω επιτελέσει, καθώς επίσης για την ανάπτυξη και ευρεία χρήση της εφαρμογής.

Εκτός της ανάλυσης δυναμικών δεδομένων της οικίας, ή της κατανάλωσης της, θα μπορούσε κάλλιστα να επεκταθεί η ανάλυση μεταβλητών και σε άλλες μεταβλητές, οι οποίες ίσως και να εμπλέκουν την ιδιότητα της ασάφειας-υποκειμενικότητας, όπως για παράδειγμα μεταβλητές οι οποίες έχουν σχέση με την προσωπική αντίληψη του χρήστη, για να είμαστε σε θέση να ενσωματώσουμε στην μελέτη αυτή και τον ανθρώπινο παράγοντα. Τέτοιου είδους μεταβλητές – ερωτήσεις συμπεριλαμβάνουν τις ακόλουθες ερωτήσεις:
-Σε τι βαθμό πιστεύετε ότι είστε ενεργειακά συνειδητοποιημένος?
-Πόσο πολύ πιστεύετε ότι είστε σε θέση να μειώσετε την ηλεκτρική σας ενέργεια παίρνοντας συγκεκριμένες δράσεις?

Έτσι ίσως να μπορούσαμε να επεκτείνουμε την ανάλυση του προβλήματος χρησιμοποιώντας ακόμη περισσότερους αλγορίθμους, όπως της ασαφής λογικής (fuzzy logic), ή ακόμη να μελετήσουμε περεταίρω το πρόβλημα με ακόμη περισσότερους αλγορίθμους, αφού στο πεδίο της μηχανικής μάθησης μπορούν να γίνονται συνεχώς βελτιστοποιήσεις, καθώς επίσης μπορούν να συνδυαστούν περισσότεροι αλγόριθμοι.

Η μελέτη δυναμικών δεδομένων και κατανάλωσης μίας οικίας και η μελέτη περισσότερων μεταβλητών που αφορούν τον ανθρώπινο παράγοντα, δίνουν σημαντικές πιθανότητες για βελτιστοποίηση των αποτελεσμάτων που έχουν προκύψει από την έρευνα αυτή. Επίσης η μελέτη του προβλήματος εμπλέκοντας ή συνδυάζοντας περισσότερους αλγορίθμους, καθώς επίσης και η συλλογή ακόμη περισσότερων δεδομένων αναμένεται ότι θα είναι σε θέση να αποδώσουν σημαντικούς καρπούς τόσο στην κοινωνία του Κοινωνικού Ηλεκτρισμού όσο και ευρύτερα αφού ο χρήστης της εφαρμογής, θα έχει τη ευχέρεια να γνωρίζει την ηλεκτρική του κατανάλωση εκ των προτέρων από την ομάδα του Κοινωνικού Ηλεκτρισμού για κάθε Κύπριο, και όχι μόνο χρήστη της εφαρμογής.

Βιβλιογραφία

[1] Phil Simon (March 18, 2013). *Too Big to Ignore: The Business Case for Big Data*. Wiley. p 89. ISBN 978-1118638170

[2] Κρόκου Α, 2008. Αυτοματοποίηση ελέγχου λογισμικού με την χρήση γράφου ροής δεδομένων και γενετικών αλγορίθμων. Πανεπιστήμιο Κύπρου

[3] Ιακώβου Α. 2010 Πρόβλεψη ποδοσφαιρικών αποτελεσμάτων με την χρήση τεχνητών νευρωνικών Δικτύων. Πανεπιστήμιο Κύπρου

[4] Chrysostomou C.,2006 Fuzzy logic based aqm congestion control in TCP/IP Networks. University of Cyprus

[5] Department Of Computer Science, Μεγάλη επιτυχία του Τμήματος Πληροφορικής, Πανεπιστήμιο Κύπρου: Πρώτο Βραβείο σε παγκόσμιο διαγωνισμό από το ITU στο "Social Electricity"2012-last update [Homepage of Department of Computer Science, University of Cyprus], [Online]. Available: http://www.cs.ucy.ac.cy/el/component/content/article/225-first-award-itu-social-electricity [01/03,2013].

[6] Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function Mathematics of Control, Signals, and Systems, 2(4), 303 - 314

[7] Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations. MIT Press, 1986

[8] Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961

[9] Christodoulou, C., 2012. Lectures Notes of Course EPL442, Computational Learning Systems.

[10] Wikipedia, *Gaussian process*, http://en.wikipedia.org/wiki/Gaussian_process (as of May 30, 2013, 14:07 GMT).

[11] CARL, R., February 23rd, 2011, 2011-last update. Available: http://www.gaussianprocess.org/ [10/05,2012].

[12] Rasmussen, C. E. (2004). "Gaussian Processes in Machine Learning". Advanced Lectures on Machine Learning. Lecture Notes in Computer Science **3176**. pp. 63–71.

[13] Η εικόνα είναι παρμένη από την σελίδα http://en.wikipedia.org/wiki/File:Isotonic_regression.svg

[14] Wikipedia, *Isotonic regression*, http://en.wikipedia.org/wiki/Isotonic_regression (as of May 30, 2013, 14:17 GMT).

[15] Wikipedia, *Linear regression*, http://en.wikipedia.org/wiki/Linear_regression (as of May 30, 2013, 14:18 GMT).

[16] Wang, Y. & Witten, I.H. (1999). Pace Regression. (Working paper 99/12). Hamilton, New Zealand: University of Waikato, Department of Computer Science.

[17] Pentaho Community, PaceRegression [Homepage of Atlassian Confluence], [Online]. Available: http://wiki.pentaho.com/display/DATAMINING/PaceRegression [02/02, 2013].

[18] Bullinaria J, 2004, Introduction to Neural Networks, Lecture 12, Radial Basis Functions: Introduction

[19] Bors A, University of York, Introduction of the Radial Basis Function Networks

[20] Wikipedia, *Radial basis function Network* http://en.wikipedia.org/wiki/Radial_basis_function_network (as of May 30, 2013, 14:22 GMT).

[21] Wikipedia, *Simplelinear regression*, http://en.wikipedia.org/wiki/Simple_linear_regression (as of May 30, 2013, 14:22 GMT).

[22] Platt, John (1998), *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*

[23] Wang, Y (2000). A new approach to fitting linear models in high dimensional spaces. Hamilton, New Zealand.

[24] Wang, Y., Witten, I. H.: Modeling for optimal probability prediction. In: Proceedings of the Nineteenth International Conference in Machine Learning, Sydney, Australia, 650-657, 2002.

[25] David J.C. Mackay (1998). Introduction to Gaussian Processes. Dept. of Physics, Cambridge University, UK.

[26] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy: Improvements to the SMO Algorithm for SVM Regression. In: IEEE Transactions on Neural Networks, 1999.

[27] A.J. Smola, B. Schoelkopf (1998). A tutorial on support vector regression.

[28] Peter J. Rousseeuw, Annick M. Leroy (1987). Robust regression and outlier detection.

[29] le Cessie, S., van Houwelingen, J.C. (1992). Ridge Estimators in Logistic Regression. Applied Statistics. 41(1):191-201.

[30] Rousseeuw P. Least Median of Squares Regression

[31] Machine Learning Group At The University Of Waikato, , Weka Machine Learning Project [Homepage of University of Waikato], [Online]. Available: http://www.cs.waikato.ac.nz/ml/index.html [09/12, 2012].

[32] Rapidminer Team, Rapidminer. Available: http://rapid-i.com/content/view/181/190/ [10/12, 2012].

[33] Eaton, J., , GNU Octave [Homepage of GNU], [Online]. Available: http://www.gnu.org/software/octave/index.html [07/31, 2012].

[34] The R Foundation For Statistical Computing, The R Project for Statistical Computing. Available: http://www.r-project.org/ [9/10, 2012].

[35] Google, 2013-last update, Google Trends [Homepage of Google], [Online]. Available: http://www.google.com/trends/ [03/04, 2013].

[36] National Grid, 2013-last update, National Grid: Electricity - Real Time Operational data . Available: http://www.nationalgrid.com/uk/Electricity/Data/Realtime/ [05/08,12] .

Παράρτημα Α

Εδώ υπάρχει ο κώδικας που έχω γράψει και χρησιμοποιήσει για την υλοποίηση αλγορίθμων. (Backpropagation MLP function and classifier)

Α.1 Κώδικας MLP with Backpropagation το οποίο παράγει τιμή

```
//Programmer: Andreas Loizou
//Task: Implement a neural network which can estimate electricity
consumption
//What for:


import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Date;

public class ArtificialNeuralNetwork {
    // TODO sindesi ton weigth table bias me sigmoid function

    //The following three constants are used to help with the indexes of
the tables which will be responcible for
    // the data normalisation
     public static final int max=1;
     public static final int min=0;
     public static final int sum=2;
     public static final int average=2;


    private float[][] normalisationTable=null;
    private float outputMax = -Float.MAX_VALUE;
    private float outputMin = Float.MAX_VALUE;
    private int numHiddenLayerOneNeurons; // How many are the neurons of
hidden layer one
```

```
    private int numHiddenLayerTwoNeurons; // How many are the neurons of
hidden layer two
    private int numInputNeurons; // How many are the input neurons
    private int numOutputNeurons; // How many are the output neurons
    private float learningRate = 0; // The learning rate
    private float momentum; // The momentum
    private int maxIterations; // The number of iterations
    private String trainFile = ""; // The name of the file with training
patterns
    private String testFile; // The name of the file with the testing
patterns

    boolean watchTime = true; //This variable is to watch the time the
network needs to be trained
    boolean debug = false; // This is a variable to help the programmer to
see some "extra info" at the output.
    // This variable is by default at false

    private int successPatterns = 0;
    private Neuron[] hiddenLayerOneNeurons; // The neurons of hidden layer
one
    private Neuron[] hiddenLayerTwoNeurons; // The neurons of hidden layer
two
    private Neuron[] outputNeurons; // The output neurons

    // The weight tables
    private float[][] weightTableFirst;
    private float[][] weightTableSecond;
    private float[][] weightTableThird;

    // The weight tables with the previous weights to be used with momentum
    private float[][] previousWeightTableFirst;
    private float[][] previousWeightTableSecond;
    private float[][] previousWeightTableThird;

    private float[][] backupWeightTableFirst;
    private float[][] backupWeightTableSecond;
    private float[][] backupWeightTableThird;

    // The number of patterns wich are used for training or testing
    int trainingDataNo = 0;
    int testingDataNo = 0;

    // The weight table of the bias
```

```
    float weightTableBiasInputLayer[];
    float weightTableBiasHiddenLayerOne[];
    float weightTableBiasHiddenLayerTwo[];

    // The weight table with the previous values to be used with momentum
    float previousWeightTableBiasInputLayer[];
    float previousWeightTableBiasHiddenLayerOne[];
    float previousWeightTableBiasHiddenLayerTwo[];

    // The weight table with the previous values to be used with momentum
    float backupWeightTableBiasInputLayer[];
    float backupWeightTableBiasHiddenLayerOne[];
    float backupWeightTableBiasHiddenLayerTwo[];

    // The outputs of the every layer
    private float[] outputsLayerOne;
    private float[] outputsLayerTwo;
    private float[] outputsLayerOutput; // These are the final outputs

    // The outputs we expect and our inputs ***FOR ONLY A PATTERN***
    private float[] wantedOutputs;
    private float[] inputs;

    // The number of layers
    private int levels = 3;

    // It returns a number from -1 to 1
    private static float floatFromMinusOneToOne() {
        int prosimo = 1;
        // We use the random to choose the sign and the random again to
choose a number from 0 to 1
        if (Math.random() < 0.5) {
            prosimo = -1;
        } else
            prosimo = 1;
        return (float) Math.random() * prosimo;

    }

    // This function is used to set bias to random values
    public void randomizeBias(float[] table) {
        int i;
        for (i = 0; i < table.length; i++) {
            table[i] = floatFromMinusOneToOne();
```

```
        }
    }

    private float normalisedValue(float value, int column){
        return ((float)value-
this.normalisationTable[min][column])/Math.abs(this.normalisationTable[max]
[column]-this.normalisationTable[min][column]);


    }
    private float normalisedOutput(float value){

        return ((float)value-this.outputMin)/Math.abs(this.outputMax-
this.outputMin);



    }
    public void normaliseData(){
        FileInputStream fstream;
        int i;
        int j;
        try {
            fstream = new FileInputStream(this.trainFile);
            // Get the object of DataInputStream
            DataInputStream in = new DataInputStream(fstream);
            BufferedReader br = new BufferedReader(new
InputStreamReader(in));
            for(i=0;i<this.trainingDataNo;i++){
                this.readLineOfDataBefore(br);
                for (j=0;j<this.numInputNeurons;j++){
                    if (this.normalisationTable[max][j]<this.inputs[j]){
                        this.normalisationTable[max][j] = this.inputs[j];
                    }
                    if (this.normalisationTable[min][j]>this.inputs[j]){
                        this.normalisationTable[min][j] = this.inputs[j];
                    }

                    if (this.outputMax<this.wantedOutputs[0]){
                        this.outputMax = this.wantedOutputs[0];

                    }
                    if (this.outputMin>this.wantedOutputs[0]){
                        this.outputMin = this.wantedOutputs[0];
```

```
                    }
this.normalisationTable[sum][j]+=inputs[j];
                }


            }
            in.close();

            fstream = new FileInputStream(this.testFile);
            // Get the object of DataInputStream
            in = new DataInputStream(fstream);
            br = new BufferedReader(new InputStreamReader(in));

            for(i=0;i<this.testingDataNo;i++){
                this.readLineOfDataBefore(br);
                for (j=0;j<this.numInputNeurons;j++){
                    if (this.normalisationTable[max][j]<this.inputs[j])
                        this.normalisationTable[max][j] = this.inputs[j];
                    if (this.normalisationTable[min][j]>this.inputs[j])
                        this.normalisationTable[min][j] = this.inputs[j];

                    if (this.outputMax<this.wantedOutputs[0]){
                        this.outputMax = this.wantedOutputs[0];

                    }
                    if (this.outputMin>this.wantedOutputs[0]){
                        this.outputMin = this.wantedOutputs[0];

                    }
this.normalisationTable[sum][j]+=inputs[j];
                }


            }
            in.close();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

```
    }

    // This function is used to save the w(t-1) weights to be used with
momentum
    // We copy the current weights to some tables
    private void copyFromBackupTablesToOlder() {
        int i, j;
        if (previousWeightTableFirst != null) {
            for (i = 0; i < previousWeightTableFirst.length; i++) {
                for (j = 0; j < previousWeightTableFirst[i].length; j++) {
                    this.previousWeightTableFirst[i][j] =
backupWeightTableFirst[i][j];
                }
            }
        }
        if (weightTableSecond != null) {
            for (i = 0; i < weightTableSecond.length; i++) {
                for (j = 0; j < weightTableSecond[i].length; j++) {
                    previousWeightTableSecond[i][j] =
backupWeightTableSecond[i][j];
                }
            }
        }
        if (weightTableThird != null) {
            for (i = 0; i < weightTableThird.length; i++) {
                for (j = 0; j < weightTableThird[i].length; j++) {
                    previousWeightTableThird[i] =
backupWeightTableThird[i];
                }
            }
        }
        if (weightTableBiasHiddenLayerOne != null) {
            for (i = 0; i < this.weightTableBiasHiddenLayerOne.length; i++)
{
                previousWeightTableBiasHiddenLayerOne[i] =
backupWeightTableBiasHiddenLayerOne[i];
            }
        }
        if (weightTableBiasHiddenLayerTwo != null) {
            for (i = 0; i < weightTableBiasHiddenLayerTwo.length; i++) {
                previousWeightTableBiasHiddenLayerTwo[i] =
backupWeightTableBiasHiddenLayerTwo[i];
            }
        }
```

```
        if (weightTableBiasInputLayer != null) {
            for (i = 0; i < this.weightTableBiasInputLayer.length; i++) {
                previousWeightTableBiasInputLayer[i] =
backupWeightTableBiasInputLayer[i];
            }
        }

    }


    // This function is used to save the w(t-1) weights to be used with
momentum
    // We copy the current weights to some tables
    private void copyFromNewWeightTablesTobackup() {
        int i, j;
        if (backupWeightTableFirst != null) {
            for (i = 0; i < backupWeightTableFirst.length; i++) {
                for (j = 0; j < backupWeightTableFirst[i].length; j++) {
                    this.backupWeightTableFirst[i][j] =
weightTableFirst[i][j];
                }
            }
        }
        if (weightTableSecond != null) {
            for (i = 0; i < weightTableSecond.length; i++) {
                for (j = 0; j < weightTableSecond[i].length; j++) {
                    backupWeightTableSecond[i][j] =
weightTableSecond[i][j];
                }
            }
        }
        if (weightTableThird != null) {
            for (i = 0; i < weightTableThird.length; i++) {
                for (j = 0; j < weightTableThird[i].length; j++) {
                    backupWeightTableThird[i] = weightTableThird[i];
                }
            }
        }
        if (weightTableBiasHiddenLayerOne != null) {
            for (i = 0; i < this.weightTableBiasHiddenLayerOne.length; i++)
{
                backupWeightTableBiasHiddenLayerOne[i] =
weightTableBiasHiddenLayerOne[i];
            }
```

```
        }
        if (weightTableBiasHiddenLayerTwo != null) {
            for (i = 0; i < weightTableBiasHiddenLayerTwo.length; i++) {
                backupWeightTableBiasHiddenLayerTwo[i] =
weightTableBiasHiddenLayerTwo[i];
            }
        }
        if (weightTableBiasInputLayer != null) {
            for (i = 0; i < this.weightTableBiasInputLayer.length; i++) {
                backupWeightTableBiasInputLayer[i] =
weightTableBiasInputLayer[i];
            }
        }

    }


    // This function is used to create the tables and objects we want, and
to give them their initial values
    public void initialize() {
        // IMPORTANT. WE CREATE ONLY THE TABLES WE WANT. FOR EXAMPLE IF WE
WANT TWO LAYERS, WE EILL CREATE ONLY TWO LAYERS
        inputs = new float[this.numInputNeurons];
        // TODO create the bias (an extra row at each table)
        this.wantedOutputs = new float[this.numOutputNeurons];
        this.outputsLayerOne = new float[this.numHiddenLayerOneNeurons];
        this.outputsLayerTwo = new float[this.numHiddenLayerTwoNeurons];
        this.outputsLayerOutput = new float[this.numOutputNeurons];

        if (numHiddenLayerOneNeurons > 0) { // If we have a layer one, then
we create it
            hiddenLayerOneNeurons = new Neuron[numHiddenLayerOneNeurons];
        }
        if (numHiddenLayerTwoNeurons > 0) { // If we have a layer two, then
we create it
            hiddenLayerTwoNeurons = new Neuron[numHiddenLayerTwoNeurons];

        }

        outputNeurons = new Neuron[numOutputNeurons]; // We always create
an output level

        weightTableBiasInputLayer = null;
        weightTableBiasHiddenLayerOne = null;
```

```
        weightTableBiasHiddenLayerTwo = null;

        if (numHiddenLayerOneNeurons == 0 && numHiddenLayerTwoNeurons == 0)
{ // one level
            weightTableFirst = new
float[numInputNeurons][numOutputNeurons];
            weightTableSecond = null;
            weightTableThird = null;

            previousWeightTableFirst = new
float[numInputNeurons][numOutputNeurons];
            previousWeightTableSecond = null;
            previousWeightTableThird = null;

            backupWeightTableFirst = new
float[numInputNeurons][numOutputNeurons];
            backupWeightTableSecond = null;
            backupWeightTableThird = null;


            randomizeWeights(weightTableFirst, numInputNeurons,
numOutputNeurons);

            createLevel(outputNeurons, 1);

            this.levels = 1;
            weightTableBiasInputLayer = new float[numOutputNeurons];
            previousWeightTableBiasInputLayer = new
float[numOutputNeurons];
            backupWeightTableBiasInputLayer = new float[numOutputNeurons];
            randomizeBias(weightTableBiasInputLayer);
        } else if (numHiddenLayerOneNeurons > 0 && numHiddenLayerTwoNeurons
== 0) { // Two levels
            weightTableFirst = new
float[numInputNeurons][numHiddenLayerOneNeurons];
            weightTableSecond = new
float[numHiddenLayerOneNeurons][numOutputNeurons];
            weightTableThird = null;

            previousWeightTableFirst = new
float[numInputNeurons][numHiddenLayerOneNeurons];
            previousWeightTableSecond = new
float[numHiddenLayerOneNeurons][numOutputNeurons];
            previousWeightTableThird = null;
```

```
            backupWeightTableFirst = new
float[numInputNeurons][numHiddenLayerOneNeurons];
            backupWeightTableSecond = new
float[numHiddenLayerOneNeurons][numOutputNeurons];
            backupWeightTableThird = null;

            randomizeWeights(weightTableFirst, numInputNeurons,
numHiddenLayerOneNeurons);
            randomizeWeights(weightTableSecond, numHiddenLayerOneNeurons,
numOutputNeurons);

            createLevel(hiddenLayerOneNeurons, 1);
            createLevel(outputNeurons, 2);

            weightTableBiasInputLayer = new
float[numHiddenLayerOneNeurons];
            weightTableBiasHiddenLayerOne = new float[numOutputNeurons];
            previousWeightTableBiasInputLayer = new
float[numHiddenLayerOneNeurons];
            previousWeightTableBiasHiddenLayerOne = new
float[numOutputNeurons];
            backupWeightTableBiasInputLayer = new
float[numHiddenLayerOneNeurons];
            backupWeightTableBiasHiddenLayerOne = new
float[numOutputNeurons];
            randomizeBias(weightTableBiasInputLayer);
            randomizeBias(weightTableBiasHiddenLayerOne);

            this.levels = 2;
        } else { // Three levels
            weightTableFirst = new
float[numInputNeurons][numHiddenLayerOneNeurons];
            weightTableSecond = new
float[numHiddenLayerOneNeurons][numHiddenLayerTwoNeurons];
            weightTableThird = new
float[numHiddenLayerTwoNeurons][numOutputNeurons];

            previousWeightTableFirst = new
float[numInputNeurons][numHiddenLayerOneNeurons];
            previousWeightTableSecond = new
float[numHiddenLayerOneNeurons][numHiddenLayerTwoNeurons];
```

```
            previousWeightTableThird = new
float[numHiddenLayerTwoNeurons][numOutputNeurons];

            backupWeightTableFirst = new
float[numInputNeurons][numHiddenLayerOneNeurons];
            backupWeightTableSecond = new
float[numHiddenLayerOneNeurons][numHiddenLayerTwoNeurons];
            backupWeightTableThird = new
float[numHiddenLayerTwoNeurons][numOutputNeurons];

            randomizeWeights(weightTableFirst, numInputNeurons,
numHiddenLayerOneNeurons);
            randomizeWeights(weightTableSecond, numHiddenLayerOneNeurons,
numHiddenLayerTwoNeurons);
            randomizeWeights(weightTableThird, numHiddenLayerTwoNeurons,
numOutputNeurons);

            createLevel(hiddenLayerOneNeurons, 1);
            createLevel(hiddenLayerTwoNeurons, 2);
            createLevel(outputNeurons, 3);

            weightTableBiasInputLayer = new
float[numHiddenLayerOneNeurons];
            weightTableBiasHiddenLayerOne = new
float[numHiddenLayerTwoNeurons];
            weightTableBiasHiddenLayerTwo = new float[numOutputNeurons];
            previousWeightTableBiasInputLayer = new
float[numHiddenLayerOneNeurons];
            previousWeightTableBiasHiddenLayerOne = new
float[numHiddenLayerTwoNeurons];
            previousWeightTableBiasHiddenLayerTwo = new
float[numOutputNeurons];

            backupWeightTableBiasInputLayer = new
float[numHiddenLayerOneNeurons];
            backupWeightTableBiasHiddenLayerOne = new
float[numHiddenLayerTwoNeurons];
            backupWeightTableBiasHiddenLayerTwo = new
float[numOutputNeurons];

            randomizeBias(weightTableBiasInputLayer);
            randomizeBias(weightTableBiasHiddenLayerOne);
            randomizeBias(weightTableBiasHiddenLayerTwo);
            this.levels = 3;
```

```
        }
        this.normalisationTable = new float[3][this.numInputNeurons];

for( int i=0;i<this.normalisationTable[min].length;i++){
    this.normalisationTable[min][i] = Float.MAX_VALUE;
    this.normalisationTable[max][i] = -Float.MAX_VALUE;
}
    }

    // Getters and setters
    public int getNumHiddenLayerOneNeurons() {
        return numHiddenLayerOneNeurons;
    }

    public void setNumHiddenLayerOneNeurons(int numHiddenLayerOneNeurons) {
        this.numHiddenLayerOneNeurons = numHiddenLayerOneNeurons;
    }

    public int getNumHiddenLayerTwoNeurons() {
        return numHiddenLayerTwoNeurons;
    }

    public void setNumHiddenLayerTwoNeurons(int numHiddenLayerTwoNeurons) {
        this.numHiddenLayerTwoNeurons = numHiddenLayerTwoNeurons;
    }

    public int getNumInputNeurons() {
        return numInputNeurons;
    }

    public void setNumInputNeurons(int numInputNeurons) {
        this.numInputNeurons = numInputNeurons;
    }

    public int getNumOutputNeurons() {
        return numOutputNeurons;
    }

    public void setNumOutputNeurons(int numOutputNeurons) {
        this.numOutputNeurons = numOutputNeurons;
    }

    public float getLearningRate() {
        return learningRate;
```

```
    }

    public void setLearningRate(float learningRate) {
        this.learningRate = learningRate;
    }

    public float getMomentum() {
        return momentum;
    }

    public void setMomentum(float momentum) {
        this.momentum = momentum;
    }

    public int getMaxIterations() {
        return maxIterations;
    }

    public void setMaxIterations(int maxIterations) {
        this.maxIterations = maxIterations;
    }

    public String getTrainFile() {
        return trainFile;
    }

    public void setTrainFile(String trainFile) {
        this.trainFile = trainFile;
    }

    public String getTestFile() {
        return testFile;
    }

    public void setTestFile(String testFile) {
        this.testFile = testFile;
    }

    // This function returns the weighted sum wich will be the input to a
neuron
    public float weightedSum(Neuron neuron) {
        float sum = 0;
        int i;
        float[][] weightTable;
```

```
        float[] inputTable;
        float[] biasTable;
        int neuronNumber = neuron.getNumberInLevel();

        // We choose the tables we want
        if (levels == 3) {
            if (neuron.getLevel() == 1) {
                biasTable = weightTableBiasInputLayer;
                weightTable = weightTableFirst;
                inputTable = this.inputs;
            } else if (neuron.getLevel() == 2) {
                biasTable = weightTableBiasHiddenLayerOne;
                weightTable = weightTableSecond;
                inputTable = outputsLayerOne;
            } else {
                biasTable = weightTableBiasHiddenLayerTwo;
                weightTable = weightTableThird;
                inputTable = outputsLayerTwo;
            }
        } else if (levels == 2) {
            if (neuron.getLevel() == 1) {
                biasTable = weightTableBiasInputLayer;
                weightTable = weightTableFirst;
                inputTable = this.inputs;
            } else {
                biasTable = weightTableBiasHiddenLayerOne;
                weightTable = weightTableSecond;
                inputTable = this.outputsLayerOne;

            }
        } else {// Levels=1;
            biasTable = weightTableBiasInputLayer;
            weightTable = weightTableFirst;
            inputTable = this.inputs;
        }

        // We add the values to find the sum
        for (i = 0; i < inputTable.length; i++) {
            sum += (float) inputTable[i] * (float)
weightTable[i][neuronNumber];
        }

        sum = sum + biasTable[neuron.getNumberInLevel()];
        return sum;
```

```
    }

    // This function prduces the output Y of a neuron
    public float neuronOutput(Neuron neuron) {

        float weightedSum = weightedSum(neuron);
        float Y = 0;

        // The following is the sigmoid function
        Y = (float) ((float) 1 / (float) (1 + (Math.pow(Math.E, -(float)
learningRate * weightedSum))));

        // We then save the output in our tables
        if (neuron.getLevel() == this.levels) {
            outputsLayerOutput[neuron.getNumberInLevel()] = Y;
            return Y;
        }
        if (neuron.getLevel() == 1) {
            outputsLayerOne[neuron.getNumberInLevel()] = Y;

        } else if (neuron.getLevel() == 2) {
            outputsLayerTwo[neuron.getNumberInLevel()] = Y;

        }
        return Y;
    }

    // We create a level of neurons
    private void createLevel(Neuron[] levelNeurons, int level) {
        int i;
        float neuronThreshold = 0;
        for (i = 0; i < levelNeurons.length; i++) {
            neuronThreshold = floatFromMinusOneToOne();
            levelNeurons[i] = new Neuron(neuronThreshold, level, i);
        }

    }

    // We randomize the initial weights
    private void randomizeWeights(float[][] table, int rows, int columns) {
        int i;
        int j;

        for (i = 0; i < rows; i++) {
```

```
            for (j = 0; j < columns; j++) {
                table[i][j] = floatFromMinusOneToOne();
            }

        }

    }

    // This function was copied from the Internet and it was changed to fit
my demands
    // "http://www.roseindia.net/java/beginners/java-read-file-line-by-
line.shtml"
    // I use this function to count the lines of a file, in order to know
how mutch are my patters
    public int dataNo(String Filename) {

        int linesNo = 0;
        try {
            // Open the file that is the first
            // command line parameter
            FileInputStream fstream = new FileInputStream(Filename);
            // Get the object of DataInputStream
            DataInputStream in = new DataInputStream(fstream);
            BufferedReader br = new BufferedReader(new
InputStreamReader(in));
            String strLine;
            // Read File Line By Line
            while ((strLine = br.readLine()) != null) {
                linesNo++;
            }
            // Close the input stream
            in.close();
        } catch (Exception e) {// Catch exception if any
            System.err.println("Error: " + e.getMessage());
        }
        return linesNo;
    }

    // This function reads a line from the file and calculates the outputs
for all the neurons
    // It does the forward phase of training or testing
    private void forwardPass(BufferedReader br) {
        int k;
        // Ignore the following lines. They are just for debugging
```

```
        if (debug == true) {

System.out.println("=========================================================
==============");
            System.out.println("Weighttable first " +
this.weightTableFirst[0][0] + " " + this.weightTableFirst[0][1] + " " +
this.weightTableFirst[1][0] + " " + this.weightTableFirst[1][1] + " ");
            System.out.println("Weighttable second " +
this.weightTableSecond[0][0] + " " + weightTableSecond[1][0] + " ");
            System.out.println("Weighttable third " +
this.weightTableThird[0][0]);
            System.out.println("bias first " +
this.weightTableBiasInputLayer[0] + " " + this.weightTableBiasInputLayer[1]
+ " ");
            System.out.println("bias second" +
this.weightTableBiasHiddenLayerOne[0]);
            System.out.println("bias third" +
this.weightTableBiasHiddenLayerTwo[0]);
        }
        // Read the next line of input
        this.readLineOfData(br);

        // Calculate the outputs of every neuron
        if (levels > 1 && this.hiddenLayerOneNeurons.length > 0) {

            for (k = 0; k < hiddenLayerOneNeurons.length; k++) {
                this.neuronOutput(hiddenLayerOneNeurons[k]);
            }
        }
        if (levels > 2 && this.hiddenLayerTwoNeurons.length > 0) {

            for (k = 0; k < hiddenLayerTwoNeurons.length; k++) {
                this.neuronOutput(hiddenLayerTwoNeurons[k]);
            }
        }

        for (k = 0; k < this.outputsLayerOutput.length; k++) {
            this.neuronOutput(this.outputNeurons[k]);
        }
        if (debug == true) {
            System.out.println(this.hiddenLayerOneNeurons[0].getDelta() + "
" + this.outputsLayerOne[0]);
            System.out.println(this.hiddenLayerOneNeurons[1].getDelta() + "
" + this.outputsLayerOne[1]);
```

```
            System.out.println(this.hiddenLayerTwoNeurons[0].getDelta() + "
" + this.outputsLayerTwo[0]);
            System.out.println(this.outputNeurons[0].getDelta() + " " +
this.outputsLayerOutput[0]);
        }
//this.printOutputs();
    }

    // This function is responcible for one epoch of training.
    // It makes the forward pass, and the backward pass and it calculates
the error and writes it to the files
    private void makeOneEpochOfTraining(int i) {

        int j;
        int k;
float error = 0;
        FileWriter fstream = null;
        BufferedWriter out = null;
        try {
            fstream = new FileWriter("errors.txt", true); // The file wich
will be used for writing the errors
            out = new BufferedWriter(fstream);
        } catch (IOException e1) {

            e1.printStackTrace();
        }
        FileInputStream fstreamin;
        DataInputStream in;
        BufferedReader br = null;
        try {
            fstreamin = new FileInputStream(this.trainFile); // The file
wich will be used to read our patters
            in = new DataInputStream(fstreamin);
            br = new BufferedReader(new InputStreamReader(in));
        } catch (FileNotFoundException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
        }
        float trainingError = 0;
        for (j = 0; j < this.trainingDataNo; j++) {

            this.forwardPass(br); // We make a forward pass
```

```
            trainingError = trainingError + this.findError(); // We
calculate the error of one pattern

            for (k = 0; k < this.outputsLayerOutput.length; k++) {
                this.delta(this.outputNeurons[k]); // We calculate the new
delta

            }

            if (this.hiddenLayerTwoNeurons != null &&
this.hiddenLayerTwoNeurons.length > 0) {

                for (k = 0; k < hiddenLayerTwoNeurons.length; k++) {
                    this.delta(hiddenLayerTwoNeurons[k]);
                }
            }

            if (this.hiddenLayerOneNeurons!=null &&
this.hiddenLayerOneNeurons.length > 0) {

                for (k = 0; k < hiddenLayerOneNeurons.length; k++) {
                    this.delta(hiddenLayerOneNeurons[k]);
                }
            }

            //

            this.copyFromNewWeightTablesTobackup();
            this.updateAllWeights(); // We update the weights
            this.copyFromBackupTablesToOlder(); // We copy the weight
Tables to other tables to be saved for momentum
error = error + Math.abs(this.wantedOutputs[0]*(this.outputMax-
this.outputMin)+this.outputMin -
(this.outputsLayerOutput[0]*(this.outputMax-
this.outputMin)+this.outputMin));
        }
        error = error / ((float) (this.trainingDataNo));
        trainingError = trainingError / ((float) (this.trainingDataNo)); //
We calculate the errors
        try {
            out.write(i + "\t" + trainingError  + "\tTR: " + error); // We
write the error to the file
        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
        try {
            out.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    // This function is responsible for one phase of testing
    // It makes the forward pass and calculates the error
    private void makeOneEpochOfTesting(int i) {

        int j;
        float iterationSumError = 0; // Finds the sum of an iteration
        float error = 0;
        FileWriter fstream = null;
        BufferedWriter out = null;
        FileInputStream fstreamin;
        DataInputStream in;
        BufferedReader br = null;
        try {
            fstream = new FileWriter("errors.txt", true); // The file in
which the error will be written
            out = new BufferedWriter(fstream);
        } catch (IOException e1) {
            e1.printStackTrace();
        }

        try {
            fstreamin = new FileInputStream(this.testFile); // The file in
which we will read our patterns
            in = new DataInputStream(fstreamin);
            br = new BufferedReader(new InputStreamReader(in));
        } catch (FileNotFoundException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
        }
        float testingError = 0;
        for (j = 0; j < this.testingDataNo; j++) {
            this.forwardPass(br); // We make our forward pass
            if (i==this.maxIterations-1){
```

```
                this.printOutputs();
            }
            error = error + Math.abs(this.wantedOutputs[0]*(this.outputMax-
this.outputMin)+this.outputMin -
(this.outputsLayerOutput[0]*(this.outputMax-
this.outputMin)+this.outputMin));
            iterationSumError = (float) iterationSumError + (float)
this.findError(); // We calculate error of the pattern
        }
        testingError = (float) iterationSumError / ((float)
(this.testingDataNo)); // We calculate the error of the epoch
        try {
            out.write("\t" + testingError  + "\tTS: "+error/ ((float)
(this.testingDataNo))+ "\n"); // We write the error to the file
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        try {
            out.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    // We find the error according to the lectures
    public float findError() {
        float forAnInputTrainingError = 0;
        // This is the type of the lecture
        for (int i = 0; i < this.numOutputNeurons; i++) {
            forAnInputTrainingError = (float) (forAnInputTrainingError +
Math.pow(this.wantedOutputs[i] - this.outputsLayerOutput[i], 2));
        }
        // System.out.println("Error is:" + (float) (0.5 *
forAnInputTrainingError));

        // The following is used to distinguish ones from zeros outputs.
        // It will be used to XOR problem

        int r = 1;

        int counterOfSuccessfulOutputNeurons=0;
```

```
        int k;
        for(k=0;k<this.numOutputNeurons;k++){


            if (this.outputsLayerOutput[k] > 0.5) {
                r = 1;
            } else
                r = 0;
if (this.wantedOutputs[k]==r){
    counterOfSuccessfulOutputNeurons++;
}


//System.out.print("[" + this.wantedOutputs[k] + "," +
this.outputsLayerOutput[k]+"] ");
        }

//if (counterOfSuccessfulOutputNeurons==this.numOutputNeurons)
    //      this.successPatterns++;

//if (this.findTheAce()==this.maxThesis(this.outputsLayerOutput)){

//  this.successPatterns++;
//}
        // The following prints the two inputs the output we want to get
and the output we get
        //System.out.println(this.inputs[0] + " " + this.inputs[1] + "wa: "
+ this.wantedOutputs[0] + "ao : " + r);
        return (float) (0.5 * forAnInputTrainingError); // Return the error
    }

    int findTheAce(){
        for (int i=0;i<this.wantedOutputs.length;i++)
                    if (this.wantedOutputs[i]==1){
                        return i;
                    }
        return 0;
    }
    // This function is used to make the training and the testing of the
network
    public void trainAndEvaluate() {
        int i;
        FileWriter fstream;
```

```
        FileWriter fstreamSuccessTraining;
        BufferedWriter out;
        try {
            fstreamSuccessTraining = new FileWriter("successrate.txt",
false);
            fstreamSuccessTraining.close();
            fstreamSuccessTraining = new FileWriter("successrate.txt",
true);
            out = new BufferedWriter(fstreamSuccessTraining);

            // We erase any previous data which are might be written at the
error files
            fstream = new FileWriter("errors.txt", false);

            // A training epoch is run and then a testing epoch is run
            for (i = 0; i < this.maxIterations; i++) {
                fstreamSuccessTraining = new FileWriter("successrate.txt",
true);
                out = new BufferedWriter(fstreamSuccessTraining);
                if(watchTime){
                System.out.println("Starting Training No "+i);
                }
                this.successPatterns = 0;
                this.makeOneEpochOfTraining(i);
                //out.write(i + "\t" +
(((float)this.successPatterns))/((float)this.trainingDataNo) );
                this.successPatterns = 0;
                if(watchTime){
                System.out.println("Starting Testing No "+i);
                }
                this.makeOneEpochOfTesting(i);
                //out.write("\t" +
(((float)this.successPatterns))/((float)this.testingDataNo) + "\n");
                out.close();
            }

            out.close();
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } // The file in which the error will be written
```

```
    }

    // This function is responcible to update the weights of all the
vertices.
    // We use momentum as well to update the weights
    public void updateAllWeights() {

        // It has to find the valid tables before to make any changes to
their weights
        // This means to find update the weights only to the tables wich
refer to levels that exist
        int i, j;
        Neuron[] table;
        if (levels == 1)
            table = outputNeurons;
        else
            table = hiddenLayerOneNeurons;

        for (i = 0; i < weightTableFirst.length; i++) {
            for (j = 0; j < table.length; j++) {
                weightTableFirst[i][j] = weightTableFirst[i][j] -
learningRate * table[j].getDelta() * inputs[i] + this.momentum *
(weightTableFirst[i][j] - previousWeightTableFirst[i][j]);
            }
        }
        for (i = 0; i < weightTableBiasInputLayer.length; i++) {

            weightTableBiasInputLayer[i] = weightTableBiasInputLayer[i] -
learningRate * table[i].getDelta() + this.momentum *
(weightTableBiasInputLayer[i] - previousWeightTableBiasInputLayer[i]);

        }

        if (levels >= 2) {
            if (levels == 2)
                table = outputNeurons;
            else
                table = hiddenLayerTwoNeurons;

            for (i = 0; i < weightTableSecond.length; i++) {
                for (j = 0; j < table.length; j++) {
                    weightTableSecond[i][j] = weightTableSecond[i][j] -
learningRate * table[j].getDelta() * outputsLayerOne[i] + this.momentum
```

```
                            * (weightTableSecond[i][j] -
previousWeightTableSecond[i][j]);

                }
            }

            for (i = 0; i < weightTableBiasHiddenLayerOne.length; i++) {

                weightTableBiasHiddenLayerOne[i] =
weightTableBiasHiddenLayerOne[i] - learningRate * table[i].getDelta() +
this.momentum
                        * (weightTableBiasHiddenLayerOne[i] -
previousWeightTableBiasHiddenLayerOne[i]);

            }

        } else {
            return;
        }

        if (levels == 3) {
            for (i = 0; i < weightTableThird.length; i++) {
                for (j = 0; j < outputNeurons.length; j++) {
                    weightTableThird[i][j] = weightTableThird[i][j] -
learningRate * outputNeurons[j].getDelta() * outputsLayerTwo[i] +
this.momentum
                            * (weightTableThird[i][j] -
previousWeightTableThird[i][j]);

                }
            }

            for (i = 0; i < weightTableBiasHiddenLayerTwo.length; i++) {

                weightTableBiasHiddenLayerTwo[i] =
weightTableBiasHiddenLayerTwo[i] - learningRate *
outputNeurons[i].getDelta() + this.momentum
                        * (weightTableBiasHiddenLayerTwo[i] -
previousWeightTableBiasHiddenLayerTwo[i]);

            }

        } else
            return;
```

```
}

// It calculates the delta of a neuron
public void delta(Neuron neuron) {
    float[] outputsTable;
    float neuronOutput;
    float deltaValue;
    float y;
    float t;
    float rightPart = 0;
    float[][] weightTable;
    Neuron[] nextLevel;
    int i;

    // It finds its level
    if (this.levels == 3) {

        if (neuron.getLevel() == 1) {
            outputsTable = outputsLayerOne;
        } else if (neuron.getLevel() == 2) {
            outputsTable = outputsLayerTwo;
        } else {
            outputsTable = outputsLayerOutput;
        }

    } else if (this.levels == 2) {

        if (neuron.getLevel() == 1) {
            outputsTable = outputsLayerOne;
        } else {
            outputsTable = outputsLayerOutput;
        }
    } else {

        outputsTable = outputsLayerOutput;

    }

    // delta is calculated

    neuronOutput = outputsTable[neuron.getNumberInLevel()];
    y = neuronOutput;
```

```
        // If its at the output level we use the output we want to get
        if (neuron.getLevel() == levels) {
            t = wantedOutputs[neuron.getNumberInLevel()];
            rightPart = y - t;
            // trainingError += y - t;
        } else {
            // If it is not at the output level we use the delta of the
neurons of the next level
            if (neuron.getLevel() == 1) {

                weightTable = weightTableSecond;
                if (levels == 2) {

                    nextLevel = this.outputNeurons;

                } else {
                    nextLevel = hiddenLayerTwoNeurons;
                }
            } else { // Level =2 and total levels = 3
                weightTable = weightTableThird;
                nextLevel = outputNeurons;
            }
            for (i = 0; i < weightTable[0].length; i++) {
                rightPart += weightTable[neuron.getNumberInLevel()][i] *
(nextLevel[i].getDelta());

            }
        }

        // Calculate delta
        deltaValue = y * (1 - y) * rightPart;
        neuron.setDelta(deltaValue);
    }

    // Prints the outputs of a pattern
    public void printOutputs(){
        System.out.print("Want: " + this.wantedOutputs[0] + " returns: " +
this.outputsLayerOutput[0] +
"["+Math.round(this.wantedOutputs[0]*(this.outputMax-
this.outputMin)+this.outputMin)+","+"" +
Math.round(this.outputsLayerOutput[0]*(this.outputMax-
this.outputMin)+this.outputMin)+"]");
        System.out.println(" Outputs [wanted output,real output] ");
        for (int i = 0;i<this.outputNeurons.length;i++){
```

```
            System.out.print("[" + this.wantedOutputs[i] + "," +
this.outputsLayerOutput[i] + "]" + " ");
        }
                System.out.println();
    }
    // It reads a pattern and saves the input into the input table and the
output we want to get into the wantedOutputs table
    public void readLineOfData(BufferedReader br) {
        String strLine;
        String[] data;
        int j;

        try {
            strLine = br.readLine();
            data = strLine.split(",");
            this.wantedOutputs[0] =
this.normalisedOutput(Float.parseFloat(data[1]));
            /*int b=0;
                    for(b=0;b<wantedOutputs.length;b++){
                        System.out.print(wantedOutputs[b] + " ");
                    }

                    System.out.println();
                    */
            for (j = 2; j < data.length; j++) {
                inputs[j-2] =
this.normalisedValue(Float.parseFloat(data[j]), j-2);

            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

    // It reads a pattern and saves the input into the input table and the
output we want to get into the wantedOutputs table
    public void readLineOfDataBefore(BufferedReader br) {
        String strLine;
        String[] data;
        int j;

        try {
```

```
            strLine = br.readLine();
            data = strLine.split(",");
            this.wantedOutputs[0] = Float.parseFloat(data[1]);
            /*int b=0;
                    for(b=0;b<wantedOutputs.length;b++){
                        System.out.print(wantedOutputs[b] + " ");
                    }

                    System.out.println();
                    */

            //We start from 2 because the first column is the ID
            for (j = 2; j < data.length; j++) {
                inputs[j-2] = Float.parseFloat(data[j]);

            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }



    /**
     * @param args
     */

    // Our main function
    public static void main(String[] args) {
        ArtificialNeuralNetwork NN; // Our neural network
        String strLine;
        BufferedReader br = null;
        Date date = null;
        NN = new ArtificialNeuralNetwork();
        if(NN.watchTime){
              date = new Date();

               // display time and date using toString()
               System.out.println("Time and date started: " +
date.toString());
            }
        FileInputStream fstream;
        try {
```

```
            fstream = new FileInputStream("parameters.txt"); // We want to
read perameters file
            DataInputStream in = new DataInputStream(fstream);
            br = new BufferedReader(new InputStreamReader(in));
        } catch (FileNotFoundException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
        }
        try {

            // We read the info we want from the parameters file
            strLine = br.readLine();

NN.setNumHiddenLayerOneNeurons(Integer.parseInt(strLine.substring("numHidde
nLayerOneNeurons ".length())));
            strLine = br.readLine();

NN.setNumHiddenLayerTwoNeurons(Integer.parseInt(strLine.substring("numHidde
nLayerTwoNeurons ".length())));
            strLine = br.readLine();

NN.setNumInputNeurons(Integer.parseInt(strLine.substring("numInputNeurons
".length())));
            strLine = br.readLine();

NN.setNumOutputNeurons(Integer.parseInt(strLine.substring("numOutputNeurons
".length())));
            strLine = br.readLine();

NN.setLearningRate(Float.parseFloat(strLine.substring("learningRate
".length())));
            strLine = br.readLine();
            NN.setMomentum(Float.parseFloat(strLine.substring("momentum
".length())));
            strLine = br.readLine();

NN.setMaxIterations(Integer.parseInt(strLine.substring("maxIterations
".length())));
            strLine = br.readLine();
            NN.setTrainFile(strLine.substring("trainFile ".length()));
            strLine = br.readLine();
            NN.setTestFile(strLine.substring("testFile ".length()));

        } catch (IOException e1) {
```

```
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        NN.trainingDataNo = NN.dataNo(NN.getTrainFile()); // We calculate
the number of training patters
        NN.testingDataNo = NN.dataNo(NN.getTestFile()); // We calculate the
number of testing patterns

        NN.initialize(); // We initialize the network

        NN.normaliseData();
        int i;
        System.out.print(NN.outputMin + " ");
        for(i=0;i<NN.normalisationTable[min].length;i++){

            System.out.print(NN.normalisationTable[min][i] + " ");
        }
        System.out.println();
        System.out.print(NN.outputMax + " ");
        for(i=0;i<NN.normalisationTable[max].length;i++){

            System.out.print(NN.normalisationTable[max][i] + " ");
        }
        System.out.println();
    NN.trainAndEvaluate(); // We train and test the network

        if (NN.watchTime){
           Date datefinished = new Date();
            NN.printWeights(NN);
           // display time and date using toString()
           System.out.println("Time and date started: " + date.toString());
           System.out.println("Time and date finished: " +
datefinished.toString());
        }


        System.out.println("***Finished***");
    }

    public void printWeights(ArtificialNeuralNetwork NN){
        int i,j;
        System.out.println("Weighttable first " );
        for (i=0;i<NN.weightTableFirst.length;i++){
            for (j=0;j<NN.weightTableFirst[0].length;j++    )
```

```
            {System.out.print(weightTableFirst[i][j] + "\t");
            }
            System.out.println();

        }
        System.out.println();
        System.out.println("Weighttable second ");
        for (i=0;i<NN.weightTableSecond.length;i++){
            for (j=0;j<NN.weightTableSecond[0].length;j++   )
            {System.out.print(weightTableSecond[i][j] + "\t");
            }
            System.out.println();

        }
        System.out.println();
System.out.println("Weighttable Third" );
for (i=0;i<NN.weightTableThird.length;i++){
    for (j=0;j<NN.weightTableThird[0].length;j++    )
    {System.out.print(weightTableThird[i][j] + "\t");
    }
    System.out.println();

}
System.out.println();

System.out.println("bias first " );
for (i=0;i<NN.weightTableBiasInputLayer.length;i++){

    System.out.print(weightTableBiasInputLayer[i] + "\t");

}System.out.println();
System.out.println("bias second" );
for (i=0;i<NN.weightTableBiasHiddenLayerOne.length;i++){

    System.out.print(weightTableBiasHiddenLayerOne[i] + "\t");

}
System.out.println();
System.out.println("bias Third" );
for (i=0;i<NN.weightTableBiasHiddenLayerTwo.length;i++){

    System.out.print(weightTableBiasHiddenLayerTwo[i] + "\t");

}
```

```
        }
    }
```

Α.2 Κλάση Neuron. Χρησιμοποιείται στα MLP Νευρωνικά Δίκτυα

```
//Programmer: Andreas Loizou
//Task: Implement a neuron class for an artificial Neural Network
public class Neuron {

private float threshold = 0;
private int level = 0;
private int numberInLevel;
private float delta = 0;

public Neuron(float threshold, int level, int numberInLevel){
    this.threshold = threshold;
    this.level = level;
    this.numberInLevel = numberInLevel;
}

public int getLevel(){

    return this.level;
}

public float getDelta(){

    return this.delta;
}

public int getNumberInLevel(){

    return this.numberInLevel;
}

public float getThreshold(){

    return this.threshold;
}

public String toString(){

    return "[Level " + level + ", numberInLevel " + numberInLevel + "] ;
Threshold = " + threshold;
```

```
}

public void setDelta(float delta){

    this.delta = delta;

}
}
```

A3 – Νευρωνικό Δίκτυο Κατηγοριοποίησης

```
//Programmer: Andreas Loizou
//Task: Implement a neural network which can classify a record
// containing the characteristics of a house at 4 different consumption
classes


import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Date;

public class ArtificialNeuralNetwork {
    // TODO sindesi ton weigth table bias me sigmoid function

    //The following three constants are used to help with the indexes of
the tables which will be responcible for
    // the data normalisation
     public static final int max=1;
     public static final int min=0;
     public static final int sum=2;
     public static final int average=2;


    private float[][] normalisationTable=null;
    private int numHiddenLayerOneNeurons; // How many are the neurons of
hidden layer one
    private int numHiddenLayerTwoNeurons; // How many are the neurons of
hidden layer two
    private int numInputNeurons; // How many are the input neurons
    private int numOutputNeurons; // How many are the output neurons
    private float learningRate = 0; // The learning rate
    private float momentum; // The momentum
    private int maxIterations; // The number of iterations
    private String trainFile = ""; // The name of the file with training
patterns
```

```
    private String testFile; // The name of the file with the testing
patterns

    boolean watchTime = true; //This variable is to watch the time the
network needs to be trained
    boolean debug = false; // This is a variable to help the programmer to
see some "extra info" at the output.
    // This variable is by default at false

    private int successPatterns = 0;
    private Neuron[] hiddenLayerOneNeurons; // The neurons of hidden layer
one
    private Neuron[] hiddenLayerTwoNeurons; // The neurons of hidden layer
two
    private Neuron[] outputNeurons; // The output neurons

    // The weight tables
    private float[][] weightTableFirst;
    private float[][] weightTableSecond;
    private float[][] weightTableThird;

    // The weight tables with the previous weights to be used with momentum
    private float[][] previousWeightTableFirst;
    private float[][] previousWeightTableSecond;
    private float[][] previousWeightTableThird;

    private float[][] backupWeightTableFirst;
    private float[][] backupWeightTableSecond;
    private float[][] backupWeightTableThird;

    // The number of patterns wich are used for training or testing
    int trainingDataNo = 0;
    int testingDataNo = 0;

    // The weight table of the bias
    float weightTableBiasInputLayer[];
    float weightTableBiasHiddenLayerOne[];
    float weightTableBiasHiddenLayerTwo[];

    // The weight table with the previous values to be used with momentum
    float previousWeightTableBiasInputLayer[];
    float previousWeightTableBiasHiddenLayerOne[];
    float previousWeightTableBiasHiddenLayerTwo[];
```

```
    // The weight table with the previous values to be used with momentum
    float backupWeightTableBiasInputLayer[];
    float backupWeightTableBiasHiddenLayerOne[];
    float backupWeightTableBiasHiddenLayerTwo[];

    // The outputs of the every layer
    private float[] outputsLayerOne;
    private float[] outputsLayerTwo;
    private float[] outputsLayerOutput; // These are the final outputs

    // The outputs we expect and our inputs ***FOR ONLY A PATTERN***
    private float[] wantedOutputs;
    private float[] inputs;

    // The number of layers
    private int levels = 3;

    // It returns a number from -1 to 1
    private static float floatFromMinusOneToOne() {
        int prosimo = 1;
        // We use the random to choose the sign and the random again to
choose a number from 0 to 1
        if (Math.random() < 0.5) {
            prosimo = -1;
        } else
            prosimo = 1;
        return (float) Math.random() * prosimo;

    }

    // This function is used to set bias to random values
    public void randomizeBias(float[] table) {
        int i;
        for (i = 0; i < table.length; i++) {
            table[i] = floatFromMinusOneToOne();
        }
    }

    private float normalisedValue(float value, int column){
        return ((float)value-
this.normalisationTable[min][column])/Math.abs(this.normalisationTable[max]
[column]-this.normalisationTable[min][column]);
```

```
    }
    public void normaliseData(){
        FileInputStream fstream;
        int i;
        int j;
        try {
            fstream = new FileInputStream(this.trainFile);
            // Get the object of DataInputStream
            DataInputStream in = new DataInputStream(fstream);
            BufferedReader br = new BufferedReader(new
InputStreamReader(in));
            for(i=0;i<this.trainingDataNo;i++){
                this.readLineOfDataBefore(br);
                for (j=0;j<this.numInputNeurons;j++){
                    if (this.normalisationTable[max][j]<this.inputs[j]){
                        this.normalisationTable[max][j] = this.inputs[j];
                    }
                    if (this.normalisationTable[min][j]>this.inputs[j]){
                        this.normalisationTable[min][j] = this.inputs[j];
                    }
this.normalisationTable[sum][j]+=inputs[j];
                }


            }
            in.close();

            fstream = new FileInputStream(this.testFile);
            // Get the object of DataInputStream
            in = new DataInputStream(fstream);
            br = new BufferedReader(new InputStreamReader(in));

            for(i=0;i<this.testingDataNo;i++){
                this.readLineOfDataBefore(br);
                for (j=0;j<this.numInputNeurons;j++){
                    if (this.normalisationTable[max][j]<this.inputs[j])
                        this.normalisationTable[max][j] = this.inputs[j];
                    if (this.normalisationTable[min][j]>this.inputs[j])
                        this.normalisationTable[min][j] = this.inputs[j];
this.normalisationTable[sum][j]+=inputs[j];
                }


            }
```

```
            in.close();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    public float[] letterToBinaryTable(char c){
        float[] binaryTable = new float[this.numOutputNeurons];
        binaryTable[c-'A'] = 1;
        return binaryTable;

    }
    // This function is used to save the w(t-1) weights to be used with
momentum
    // We copy the current weights to some tables
    private void copyFromBackupTablesToOlder() {
        int i, j;
        if (previousWeightTableFirst != null) {
            for (i = 0; i < previousWeightTableFirst.length; i++) {
                for (j = 0; j < previousWeightTableFirst[i].length; j++) {
                    this.previousWeightTableFirst[i][j] =
backupWeightTableFirst[i][j];
                }
            }
        }
        if (weightTableSecond != null) {
            for (i = 0; i < weightTableSecond.length; i++) {
                for (j = 0; j < weightTableSecond[i].length; j++) {
                    previousWeightTableSecond[i][j] =
backupWeightTableSecond[i][j];
                }
            }
        }
        if (weightTableThird != null) {
            for (i = 0; i < weightTableThird.length; i++) {
                for (j = 0; j < weightTableThird[i].length; j++) {
                    previousWeightTableThird[i] =
backupWeightTableThird[i];
                }
```

```
            }
        }
        if (weightTableBiasHiddenLayerOne != null) {
            for (i = 0; i < this.weightTableBiasHiddenLayerOne.length; i++)
{
                previousWeightTableBiasHiddenLayerOne[i] =
backupWeightTableBiasHiddenLayerOne[i];
            }
        }
        if (weightTableBiasHiddenLayerTwo != null) {
            for (i = 0; i < weightTableBiasHiddenLayerTwo.length; i++) {
                previousWeightTableBiasHiddenLayerTwo[i] =
backupWeightTableBiasHiddenLayerTwo[i];
            }
        }
        if (weightTableBiasInputLayer != null) {
            for (i = 0; i < this.weightTableBiasInputLayer.length; i++) {
                previousWeightTableBiasInputLayer[i] =
backupWeightTableBiasInputLayer[i];
            }
        }

    }


    // This function is used to save the w(t-1) weights to be used with
momentum
    // We copy the current weights to some tables
    private void copyFromNewWeightTablesTobackup() {
        int i, j;
        if (backupWeightTableFirst != null) {
            for (i = 0; i < backupWeightTableFirst.length; i++) {
                for (j = 0; j < backupWeightTableFirst[i].length; j++) {
                    this.backupWeightTableFirst[i][j] =
weightTableFirst[i][j];
                }
            }
        }
        if (weightTableSecond != null) {
            for (i = 0; i < weightTableSecond.length; i++) {
                for (j = 0; j < weightTableSecond[i].length; j++) {
                    backupWeightTableSecond[i][j] =
weightTableSecond[i][j];
                }
```

```
            }
        }
        if (weightTableThird != null) {
            for (i = 0; i < weightTableThird.length; i++) {
                for (j = 0; j < weightTableThird[i].length; j++) {
                    backupWeightTableThird[i] = weightTableThird[i];
                }
            }
        }
        if (weightTableBiasHiddenLayerOne != null) {
            for (i = 0; i < this.weightTableBiasHiddenLayerOne.length; i++)
{
                backupWeightTableBiasHiddenLayerOne[i] =
weightTableBiasHiddenLayerOne[i];
            }
        }
        if (weightTableBiasHiddenLayerTwo != null) {
            for (i = 0; i < weightTableBiasHiddenLayerTwo.length; i++) {
                backupWeightTableBiasHiddenLayerTwo[i] =
weightTableBiasHiddenLayerTwo[i];
            }
        }
        if (weightTableBiasInputLayer != null) {
            for (i = 0; i < this.weightTableBiasInputLayer.length; i++) {
                backupWeightTableBiasInputLayer[i] =
weightTableBiasInputLayer[i];
            }
        }

    }


    // This function is used to create the tables and objects we want, and
to give them their initial values
    public void initialize() {
        // IMPORTANT. WE CREATE ONLY THE TABLES WE WANT. FOR EXAMPLE IF WE
WANT TWO LAYERS, WE EILL CREATE ONLY TWO LAYERS
        inputs = new float[this.numInputNeurons];
        // TODO create the bias (an extra row at each table)
        this.wantedOutputs = new float[this.numOutputNeurons];
        this.outputsLayerOne = new float[this.numHiddenLayerOneNeurons];
        this.outputsLayerTwo = new float[this.numHiddenLayerTwoNeurons];
        this.outputsLayerOutput = new float[this.numOutputNeurons];
```

```
        if (numHiddenLayerOneNeurons > 0) { // If we have a layer one, then
we create it
            hiddenLayerOneNeurons = new Neuron[numHiddenLayerOneNeurons];
        }
        if (numHiddenLayerTwoNeurons > 0) { // If we have a layer two, then
we create it
            hiddenLayerTwoNeurons = new Neuron[numHiddenLayerTwoNeurons];

        }

        outputNeurons = new Neuron[numOutputNeurons]; // We always create
an output level

        weightTableBiasInputLayer = null;
        weightTableBiasHiddenLayerOne = null;
        weightTableBiasHiddenLayerTwo = null;

        if (numHiddenLayerOneNeurons == 0 && numHiddenLayerTwoNeurons == 0)
{ // one level
            weightTableFirst = new
float[numInputNeurons][numOutputNeurons];
            weightTableSecond = null;
            weightTableThird = null;

            previousWeightTableFirst = new
float[numInputNeurons][numOutputNeurons];
            previousWeightTableSecond = null;
            previousWeightTableThird = null;

            backupWeightTableFirst = new
float[numInputNeurons][numOutputNeurons];
            backupWeightTableSecond = null;
            backupWeightTableThird = null;


            randomizeWeights(weightTableFirst, numInputNeurons,
numOutputNeurons);

            createLevel(outputNeurons, 1);

            this.levels = 1;
            weightTableBiasInputLayer = new float[numOutputNeurons];
            previousWeightTableBiasInputLayer = new
float[numOutputNeurons];
```

```
            backupWeightTableBiasInputLayer = new float[numOutputNeurons];
            randomizeBias(weightTableBiasInputLayer);
        } else if (numHiddenLayerOneNeurons > 0 && numHiddenLayerTwoNeurons
== 0) { // Two levels
            weightTableFirst = new
float[numInputNeurons][numHiddenLayerOneNeurons];
            weightTableSecond = new
float[numHiddenLayerOneNeurons][numOutputNeurons];
            weightTableThird = null;

            previousWeightTableFirst = new
float[numInputNeurons][numHiddenLayerOneNeurons];
            previousWeightTableSecond = new
float[numHiddenLayerOneNeurons][numOutputNeurons];
            previousWeightTableThird = null;


            backupWeightTableFirst = new
float[numInputNeurons][numHiddenLayerOneNeurons];
            backupWeightTableSecond = new
float[numHiddenLayerOneNeurons][numOutputNeurons];
            backupWeightTableThird = null;

            randomizeWeights(weightTableFirst, numInputNeurons,
numHiddenLayerOneNeurons);
            randomizeWeights(weightTableSecond, numHiddenLayerOneNeurons,
numOutputNeurons);

            createLevel(hiddenLayerOneNeurons, 1);
            createLevel(outputNeurons, 2);

            weightTableBiasInputLayer = new
float[numHiddenLayerOneNeurons];
            weightTableBiasHiddenLayerOne = new float[numOutputNeurons];
            previousWeightTableBiasInputLayer = new
float[numHiddenLayerOneNeurons];
            previousWeightTableBiasHiddenLayerOne = new
float[numOutputNeurons];
            backupWeightTableBiasInputLayer = new
float[numHiddenLayerOneNeurons];
            backupWeightTableBiasHiddenLayerOne = new
float[numOutputNeurons];
            randomizeBias(weightTableBiasInputLayer);
            randomizeBias(weightTableBiasHiddenLayerOne);
```

```
            this.levels = 2;
        } else { // Three levels
            weightTableFirst = new
float[numInputNeurons][numHiddenLayerOneNeurons];
            weightTableSecond = new
float[numHiddenLayerOneNeurons][numHiddenLayerTwoNeurons];
            weightTableThird = new
float[numHiddenLayerTwoNeurons][numOutputNeurons];

            previousWeightTableFirst = new
float[numInputNeurons][numHiddenLayerOneNeurons];
            previousWeightTableSecond = new
float[numHiddenLayerOneNeurons][numHiddenLayerTwoNeurons];
            previousWeightTableThird = new
float[numHiddenLayerTwoNeurons][numOutputNeurons];

            backupWeightTableFirst = new
float[numInputNeurons][numHiddenLayerOneNeurons];
            backupWeightTableSecond = new
float[numHiddenLayerOneNeurons][numHiddenLayerTwoNeurons];
            backupWeightTableThird = new
float[numHiddenLayerTwoNeurons][numOutputNeurons];

            randomizeWeights(weightTableFirst, numInputNeurons,
numHiddenLayerOneNeurons);
            randomizeWeights(weightTableSecond, numHiddenLayerOneNeurons,
numHiddenLayerTwoNeurons);
            randomizeWeights(weightTableThird, numHiddenLayerTwoNeurons,
numOutputNeurons);

            createLevel(hiddenLayerOneNeurons, 1);
            createLevel(hiddenLayerTwoNeurons, 2);
            createLevel(outputNeurons, 3);

            weightTableBiasInputLayer = new
float[numHiddenLayerOneNeurons];
            weightTableBiasHiddenLayerOne = new
float[numHiddenLayerTwoNeurons];
            weightTableBiasHiddenLayerTwo = new float[numOutputNeurons];
            previousWeightTableBiasInputLayer = new
float[numHiddenLayerOneNeurons];
            previousWeightTableBiasHiddenLayerOne = new
float[numHiddenLayerTwoNeurons];
```

```
            previousWeightTableBiasHiddenLayerTwo = new
float[numOutputNeurons];

            backupWeightTableBiasInputLayer = new
float[numHiddenLayerOneNeurons];
            backupWeightTableBiasHiddenLayerOne = new
float[numHiddenLayerTwoNeurons];
            backupWeightTableBiasHiddenLayerTwo = new
float[numOutputNeurons];

            randomizeBias(weightTableBiasInputLayer);
            randomizeBias(weightTableBiasHiddenLayerOne);
            randomizeBias(weightTableBiasHiddenLayerTwo);
            this.levels = 3;
        }
        this.normalisationTable = new float[3][this.numInputNeurons];

for( int i=0;i<this.normalisationTable[min].length;i++){
    this.normalisationTable[min][i] = Float.MAX_VALUE;
    this.normalisationTable[max][i] = Float.MIN_VALUE;
}
    }

    // Getters and setters
    public int getNumHiddenLayerOneNeurons() {
        return numHiddenLayerOneNeurons;
    }

    public void setNumHiddenLayerOneNeurons(int numHiddenLayerOneNeurons) {
        this.numHiddenLayerOneNeurons = numHiddenLayerOneNeurons;
    }

    public int getNumHiddenLayerTwoNeurons() {
        return numHiddenLayerTwoNeurons;
    }

    public void setNumHiddenLayerTwoNeurons(int numHiddenLayerTwoNeurons) {
        this.numHiddenLayerTwoNeurons = numHiddenLayerTwoNeurons;
    }

    public int getNumInputNeurons() {
        return numInputNeurons;
    }
```

```
public void setNumInputNeurons(int numInputNeurons) {
    this.numInputNeurons = numInputNeurons;
}

public int getNumOutputNeurons() {
    return numOutputNeurons;
}

public void setNumOutputNeurons(int numOutputNeurons) {
    this.numOutputNeurons = numOutputNeurons;
}

public float getLearningRate() {
    return learningRate;
}

public void setLearningRate(float learningRate) {
    this.learningRate = learningRate;
}

public float getMomentum() {
    return momentum;
}

public void setMomentum(float momentum) {
    this.momentum = momentum;
}

public int getMaxIterations() {
    return maxIterations;
}

public void setMaxIterations(int maxIterations) {
    this.maxIterations = maxIterations;
}

public String getTrainFile() {
    return trainFile;
}

public void setTrainFile(String trainFile) {
    this.trainFile = trainFile;
}
```

```
    public String getTestFile() {
        return testFile;
    }

    public void setTestFile(String testFile) {
        this.testFile = testFile;
    }

    // This function returns the weighted sum wich will be the input to a
neuron
    public float weightedSum(Neuron neuron) {
        float sum = 0;
        int i;
        float[][] weightTable;
        float[] inputTable;
        float[] biasTable;
        int neuronNumber = neuron.getNumberInLevel();

        // We choose the tables we want
        if (levels == 3) {
            if (neuron.getLevel() == 1) {
                biasTable = weightTableBiasInputLayer;
                weightTable = weightTableFirst;
                inputTable = this.inputs;
            } else if (neuron.getLevel() == 2) {
                biasTable = weightTableBiasHiddenLayerOne;
                weightTable = weightTableSecond;
                inputTable = outputsLayerOne;
            } else {
                biasTable = weightTableBiasHiddenLayerTwo;
                weightTable = weightTableThird;
                inputTable = outputsLayerTwo;
            }
        } else if (levels == 2) {
            if (neuron.getLevel() == 1) {
                biasTable = weightTableBiasInputLayer;
                weightTable = weightTableFirst;
                inputTable = this.inputs;
            } else {
                biasTable = weightTableBiasHiddenLayerOne;
                weightTable = weightTableSecond;
                inputTable = this.outputsLayerOne;

            }
```

```
        } else {// Levels=1;
            biasTable = weightTableBiasInputLayer;
            weightTable = weightTableFirst;
            inputTable = this.inputs;
        }

        // We add the values to find the sum
        for (i = 0; i < inputTable.length; i++) {
            sum += (float) inputTable[i] * (float)
weightTable[i][neuronNumber];
        }

        sum = sum + biasTable[neuron.getNumberInLevel()];
        return sum;
    }

    // This function prduces the output Y of a neuron
    public float neuronOutput(Neuron neuron) {

        float weightedSum = weightedSum(neuron);
        float Y = 0;

        // The following is the sigmoid function
        Y = (float) ((float) 1 / (float) (1 + (Math.pow(Math.E, -(float)
learningRate * weightedSum))));

        // We then save the output in our tables
        if (neuron.getLevel() == this.levels) {
            outputsLayerOutput[neuron.getNumberInLevel()] = Y;
            return Y;
        }
        if (neuron.getLevel() == 1) {
            outputsLayerOne[neuron.getNumberInLevel()] = Y;

        } else if (neuron.getLevel() == 2) {
            outputsLayerTwo[neuron.getNumberInLevel()] = Y;

        }
        return Y;
    }

    // We create a level of neurons
    private void createLevel(Neuron[] levelNeurons, int level) {
        int i;
```

```
        float neuronThreshold = 0;
        for (i = 0; i < levelNeurons.length; i++) {
            neuronThreshold = floatFromMinusOneToOne();
            levelNeurons[i] = new Neuron(neuronThreshold, level, i);
        }

    }

    // We randomize the initial weights
    private void randomizeWeights(float[][] table, int rows, int columns) {
        int i;
        int j;

        for (i = 0; i < rows; i++) {
            for (j = 0; j < columns; j++) {
                table[i][j] = floatFromMinusOneToOne();
            }

        }

    }

    // This function was copied from the Internet and it was changed to fit
my demands
    // "http://www.roseindia.net/java/beginners/java-read-file-line-by-
line.shtml"
    // I use this function to count the lines of a file, in order to know
how mutch are my patters
    public int dataNo(String Filename) {

        int linesNo = 0;
        try {
            // Open the file that is the first
            // command line parameter
            FileInputStream fstream = new FileInputStream(Filename);
            // Get the object of DataInputStream
            DataInputStream in = new DataInputStream(fstream);
            BufferedReader br = new BufferedReader(new
InputStreamReader(in));
            String strLine;
            // Read File Line By Line
            while ((strLine = br.readLine()) != null) {
                linesNo++;
            }
```

```
            // Close the input stream
            in.close();
        } catch (Exception e) {// Catch exception if any
            System.err.println("Error: " + e.getMessage());
        }
        return linesNo;
    }

    // This function reads a line from the file and calculates the outputs
for all the neurons
    // It does the forward phase of training or testing
    private void forwardPass(BufferedReader br) {
        int k;
        // Ignore the following lines. They are just for debugging
        if (debug == true) {

System.out.println("==========================================================
==============");
            System.out.println("Weighttable first " +
this.weightTableFirst[0][0] + " " + this.weightTableFirst[0][1] + " " +
this.weightTableFirst[1][0] + " " + this.weightTableFirst[1][1] + " ");
            System.out.println("Weighttable second " +
this.weightTableSecond[0][0] + " " + weightTableSecond[1][0] + " ");
            System.out.println("Weighttable third " +
this.weightTableThird[0][0]);
            System.out.println("bias first " +
this.weightTableBiasInputLayer[0] + " " + this.weightTableBiasInputLayer[1]
+ " ");
            System.out.println("bias second" +
this.weightTableBiasHiddenLayerOne[0]);
            System.out.println("bias third" +
this.weightTableBiasHiddenLayerTwo[0]);
        }
        // Read the next line of input
        this.readLineOfData(br);

        // Calculate the outputs of every neuron
        if (levels > 1 && this.hiddenLayerOneNeurons.length > 0) {

            for (k = 0; k < hiddenLayerOneNeurons.length; k++) {
                this.neuronOutput(hiddenLayerOneNeurons[k]);
            }
        }
        if (levels > 2 && this.hiddenLayerTwoNeurons.length > 0) {
```

```
            for (k = 0; k < hiddenLayerTwoNeurons.length; k++) {
                this.neuronOutput(hiddenLayerTwoNeurons[k]);
            }
        }

        for (k = 0; k < this.outputsLayerOutput.length; k++) {
            this.neuronOutput(this.outputNeurons[k]);
        }
        if (debug == true) {
            System.out.println(this.hiddenLayerOneNeurons[0].getDelta() + "
" + this.outputsLayerOne[0]);
            System.out.println(this.hiddenLayerOneNeurons[1].getDelta() + "
" + this.outputsLayerOne[1]);
            System.out.println(this.hiddenLayerTwoNeurons[0].getDelta() + "
" + this.outputsLayerTwo[0]);
            System.out.println(this.outputNeurons[0].getDelta() + " " +
this.outputsLayerOutput[0]);
        }
this.printOutputs();
    }

    // This function is responcible for one epoch of training.
    // It makes the forward pass, and the backward pass and it calculates
the error and writes it to the files
    private void makeOneEpochOfTraining(int i) {

        int j;
        int k;

        FileWriter fstream = null;
        BufferedWriter out = null;
        try {
            fstream = new FileWriter("errors.txt", true); // The file wich
will be used for writing the errors
            out = new BufferedWriter(fstream);
        } catch (IOException e1) {

            e1.printStackTrace();
        }
        FileInputStream fstreamin;
        DataInputStream in;
        BufferedReader br = null;
        try {
```

```
            fstreamin = new FileInputStream(this.trainFile); // The file
wich will be used to read our patters
            in = new DataInputStream(fstreamin);
            br = new BufferedReader(new InputStreamReader(in));
        } catch (FileNotFoundException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
        }
        float trainingError = 0;
        for (j = 0; j < this.trainingDataNo; j++) {

            this.forwardPass(br); // We make a forward pass
            trainingError = trainingError + this.findError(); // We
calculate the error of one pattern

            for (k = 0; k < this.outputsLayerOutput.length; k++) {
                this.delta(this.outputNeurons[k]); // We calculate the new
delta

            }

            if (this.hiddenLayerTwoNeurons != null &&
this.hiddenLayerTwoNeurons.length > 0) {

                for (k = 0; k < hiddenLayerTwoNeurons.length; k++) {
                    this.delta(hiddenLayerTwoNeurons[k]);
                }
            }

            if (this.hiddenLayerOneNeurons!=null &&
this.hiddenLayerOneNeurons.length > 0) {

                for (k = 0; k < hiddenLayerOneNeurons.length; k++) {
                    this.delta(hiddenLayerOneNeurons[k]);
                }
            }

            //

            this.copyFromNewWeightTablesTobackup();
            this.updateAllWeights(); // We update the weights
            this.copyFromBackupTablesToOlder(); // We copy the weight
Tables to other tables to be saved for momentum
```

```
        }
        trainingError = trainingError / ((float) (this.trainingDataNo)); //
We calculate the errors
        try {
            out.write(i + "\t" + trainingError ); // We write the error to
the file
        } catch (IOException e) {

            e.printStackTrace();
        }
        try {
            out.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    // This function is responsible for one phase of testing
    // It makes the forward pass and calculates the error
    private void makeOneEpochOfTesting(int i) {

        int j;
        float iterationSumError = 0; // Finds the sum of an iteration
        FileWriter fstream = null;
        BufferedWriter out = null;
        FileInputStream fstreamin;
        DataInputStream in;
        BufferedReader br = null;
        try {
            fstream = new FileWriter("errors.txt", true); // The file in
which the error will be written
            out = new BufferedWriter(fstream);
        } catch (IOException e1) {
            e1.printStackTrace();
        }

        try {
            fstreamin = new FileInputStream(this.testFile); // The file in
which we will read our patterns
            in = new DataInputStream(fstreamin);
            br = new BufferedReader(new InputStreamReader(in));
        } catch (FileNotFoundException e2) {
```

```
            // TODO Auto-generated catch block
            e2.printStackTrace();
        }
        float testingError = 0;
        for (j = 0; j < this.testingDataNo; j++) {
            this.forwardPass(br); // We make our forward pass
            iterationSumError = (float) iterationSumError + (float)
this.findError(); // We calculate error of the pattern
        }
        testingError = (float) iterationSumError / ((float)
(this.testingDataNo)); // We calculate the error of the epoch
        try {
            out.write("\t" + testingError + "\n"); // We write the error to
the file
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        try {
            out.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    // We find the error according to the lectures
    public float findError() {
        float forAnInputTrainingError = 0;
        // This is the type of the lecture
        for (int i = 0; i < this.numOutputNeurons; i++) {
            forAnInputTrainingError = (float) (forAnInputTrainingError +
Math.pow(this.wantedOutputs[i] - this.outputsLayerOutput[i], 2));
        }
        // System.out.println("Error is:" + (float) (0.5 *
forAnInputTrainingError));

        // The following is used to distinguish ones from zeros outputs.
        // It will be used to XOR problem

        int r = 1;

        int counterOfSuccessfulOutputNeurons=0;
```

```
        int k;
        for(k=0;k<this.numOutputNeurons;k++){


            if (this.outputsLayerOutput[k] > 0.5) {
                r = 1;
            } else
                r = 0;
if (this.wantedOutputs[k]==r){
    counterOfSuccessfulOutputNeurons++;
}


//System.out.print("[" + this.wantedOutputs[k] + "," +
this.outputsLayerOutput[k]+"] ");
        }

//if (counterOfSuccessfulOutputNeurons==this.numOutputNeurons)
    //      this.successPatterns++;

if (this.findTheAce()==this.maxThesis(this.outputsLayerOutput)){

    this.successPatterns++;
}
        // The following prints the two inputs the output we want to get
and the output we get
        //System.out.println(this.inputs[0] + " " + this.inputs[1] + "wa: "
+ this.wantedOutputs[0] + "ao : " + r);
        return (float) (0.5 * forAnInputTrainingError); // Return the error
    }

    int findTheAce(){
        for (int i=0;i<this.wantedOutputs.length;i++)
                    if (this.wantedOutputs[i]==1){
                        return i;
                    }
        return 0;
    }
    // This function is used to make the training and the testing of the
network
    public void trainAndEvaluate() {
        int i;
        FileWriter fstream;
```

```
        FileWriter fstreamSuccessTraining;
        BufferedWriter out;
        try {
            fstreamSuccessTraining = new FileWriter("successrate.txt",
false);
            fstreamSuccessTraining.close();
            fstreamSuccessTraining = new FileWriter("successrate.txt",
true);
            out = new BufferedWriter(fstreamSuccessTraining);

            // We erase any previous data which are might be written at the
error files
            fstream = new FileWriter("errors.txt", false);

            // A training epoch is run and then a testing epoch is run
            for (i = 0; i < this.maxIterations; i++) {
                fstreamSuccessTraining = new FileWriter("successrate.txt",
true);
                out = new BufferedWriter(fstreamSuccessTraining);
                if(watchTime){
                System.out.println("Starting Training No "+i);
                }
                this.successPatterns = 0;
                this.makeOneEpochOfTraining(i);
                out.write(i + "\t" +
(((float)this.successPatterns))/((float)this.trainingDataNo) );
                this.successPatterns = 0;
                if(watchTime){
                System.out.println("Starting Testing No "+i);
                }
                this.makeOneEpochOfTesting(i);
                out.write("\t" +
(((float)this.successPatterns))/((float)this.testingDataNo) + "\n");
                out.close();
            }

            out.close();
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } // The file in which the error will be written
```

```
    }

    // This function is responcible to update the weights of all the
vertices.
    // We use momentum as well to update the weights
    public void updateAllWeights() {

        // It has to find the valid tables before to make any changes to
their weights
        // This means to find update the weights only to the tables wich
refer to levels that exist
        int i, j;
        Neuron[] table;
        if (levels == 1)
            table = outputNeurons;
        else
            table = hiddenLayerOneNeurons;

        for (i = 0; i < weightTableFirst.length; i++) {
            for (j = 0; j < table.length; j++) {
                weightTableFirst[i][j] = weightTableFirst[i][j] -
learningRate * table[j].getDelta() * inputs[i] + this.momentum *
(weightTableFirst[i][j] - previousWeightTableFirst[i][j]);
            }
        }
        for (i = 0; i < weightTableBiasInputLayer.length; i++) {

            weightTableBiasInputLayer[i] = weightTableBiasInputLayer[i] -
learningRate * table[i].getDelta() + this.momentum *
(weightTableBiasInputLayer[i] - previousWeightTableBiasInputLayer[i]);

        }

        if (levels >= 2) {
            if (levels == 2)
                table = outputNeurons;
            else
                table = hiddenLayerTwoNeurons;

            for (i = 0; i < weightTableSecond.length; i++) {
                for (j = 0; j < table.length; j++) {
                    weightTableSecond[i][j] = weightTableSecond[i][j] -
learningRate * table[j].getDelta() * outputsLayerOne[i] + this.momentum
```

```
                            * (weightTableSecond[i][j] -
previousWeightTableSecond[i][j]);

                }
            }

            for (i = 0; i < weightTableBiasHiddenLayerOne.length; i++) {

                weightTableBiasHiddenLayerOne[i] =
weightTableBiasHiddenLayerOne[i] - learningRate * table[i].getDelta() +
this.momentum
                        * (weightTableBiasHiddenLayerOne[i] -
previousWeightTableBiasHiddenLayerOne[i]);

            }

        } else {
            return;
        }

        if (levels == 3) {
            for (i = 0; i < weightTableThird.length; i++) {
                for (j = 0; j < outputNeurons.length; j++) {
                    weightTableThird[i][j] = weightTableThird[i][j] -
learningRate * outputNeurons[j].getDelta() * outputsLayerTwo[i] +
this.momentum
                            * (weightTableThird[i][j] -
previousWeightTableThird[i][j]);

                }
            }

            for (i = 0; i < weightTableBiasHiddenLayerTwo.length; i++) {

                weightTableBiasHiddenLayerTwo[i] =
weightTableBiasHiddenLayerTwo[i] - learningRate *
outputNeurons[i].getDelta() + this.momentum
                        * (weightTableBiasHiddenLayerTwo[i] -
previousWeightTableBiasHiddenLayerTwo[i]);

            }

        } else
            return;
```

```
}

// It calculates the delta of a neuron
public void delta(Neuron neuron) {
    float[] outputsTable;
    float neuronOutput;
    float deltaValue;
    float y;
    float t;
    float rightPart = 0;
    float[][] weightTable;
    Neuron[] nextLevel;
    int i;

    // It finds its level
    if (this.levels == 3) {

        if (neuron.getLevel() == 1) {
            outputsTable = outputsLayerOne;
        } else if (neuron.getLevel() == 2) {
            outputsTable = outputsLayerTwo;
        } else {
            outputsTable = outputsLayerOutput;
        }

    } else if (this.levels == 2) {

        if (neuron.getLevel() == 1) {
            outputsTable = outputsLayerOne;
        } else {
            outputsTable = outputsLayerOutput;
        }
    } else {

        outputsTable = outputsLayerOutput;

    }

    // delta is calculated

    neuronOutput = outputsTable[neuron.getNumberInLevel()];
    y = neuronOutput;
```

```
        // If its at the output level we use the output we want to get
        if (neuron.getLevel() == levels) {
            t = wantedOutputs[neuron.getNumberInLevel()];
            rightPart = y - t;
            // trainingError += y - t;
        } else {
            // If it is not at the output level we use the delta of the
neurons of the next level
            if (neuron.getLevel() == 1) {

                weightTable = weightTableSecond;
                if (levels == 2) {

                    nextLevel = this.outputNeurons;

                } else {
                    nextLevel = hiddenLayerTwoNeurons;
                }
            } else { // Level =2 and total levels = 3
                weightTable = weightTableThird;
                nextLevel = outputNeurons;
            }
            for (i = 0; i < weightTable[0].length; i++) {
                rightPart += weightTable[neuron.getNumberInLevel()][i] *
(nextLevel[i].getDelta());

            }
        }

        // Calculate delta
        deltaValue = y * (1 - y) * rightPart;
        neuron.setDelta(deltaValue);
    }

    // Prints the outputs of a pattern
    public void printOutputs(){
        System.out.print("Want: " + ((char) ('A'+this.findTheAce())) + "
returns: " + ((char)('A' + this.maxThesis(this.outputsLayerOutput))));
        System.out.println(" Outputs [wanted output,real output] ");
        for (int i = 0;i<this.outputNeurons.length;i++){

            System.out.print("[" + this.wantedOutputs[i] + "," +
this.outputsLayerOutput[i] + "]" + " ");
        }
```

```
                System.out.println();
    }
    // It reads a pattern and saves the input into the input table and the
output we want to get into the wantedOutputs table
    public void readLineOfData(BufferedReader br) {
        String strLine;
        String[] data;
        int j;

        try {
            strLine = br.readLine();
            data = strLine.split(",");
            this.wantedOutputs =
this.letterToBinaryTable(data[0].charAt(0));
            /*int b=0;
                    for(b=0;b<wantedOutputs.length;b++){
                        System.out.print(wantedOutputs[b] + " ");
                    }

                    System.out.println();
                    */
            for (j = 1; j < data.length; j++) {
                inputs[j-1] =
this.normalisedValue(Float.parseFloat(data[j]), j-1);

            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

    // It reads a pattern and saves the input into the input table and the
output we want to get into the wantedOutputs table
    public void readLineOfDataBefore(BufferedReader br) {
        String strLine;
        String[] data;
        int j;

        try {
            strLine = br.readLine();
            data = strLine.split(",");
            this.wantedOutputs =
this.letterToBinaryTable(data[0].charAt(0));
```

```
            /*int b=0;
                    for(b=0;b<wantedOutputs.length;b++){
                        System.out.print(wantedOutputs[b] + " ");
                    }

                    System.out.println();
                    */


            for (j = 1; j < data.length; j++) {
                inputs[j-1] = Float.parseFloat(data[j]);

            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
    float maxThesis(float[] table){
        int thesis=0;
        float tempMax=0;
        for (int i=0;i<table.length;i++){
            if (tempMax<table[i]){
                tempMax=table[i];
                thesis = i;
            }

        }
        return thesis;
    }


    /**
     * @param args
     */

    // Our main function
    public static void main(String[] args) {
        ArtificialNeuralNetwork NN; // Our neural network
        String strLine;
        BufferedReader br = null;
        Date date = null;
        NN = new ArtificialNeuralNetwork();
        if(NN.watchTime){
```

```
                date = new Date();

                 // display time and date using toString()
                 System.out.println("Time and date started: " +
date.toString());
            }
        FileInputStream fstream;
        try {
            fstream = new FileInputStream("parameters.txt"); // We want to
read perameters file
            DataInputStream in = new DataInputStream(fstream);
            br = new BufferedReader(new InputStreamReader(in));
        } catch (FileNotFoundException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
        }
        try {

            // We read the info we want from the parameters file
            strLine = br.readLine();

NN.setNumHiddenLayerOneNeurons(Integer.parseInt(strLine.substring("numHidde
nLayerOneNeurons ".length())));
            strLine = br.readLine();

NN.setNumHiddenLayerTwoNeurons(Integer.parseInt(strLine.substring("numHidde
nLayerTwoNeurons ".length())));
            strLine = br.readLine();

NN.setNumInputNeurons(Integer.parseInt(strLine.substring("numInputNeurons
".length())));
            strLine = br.readLine();

NN.setNumOutputNeurons(Integer.parseInt(strLine.substring("numOutputNeurons
".length())));
            strLine = br.readLine();

NN.setLearningRate(Float.parseFloat(strLine.substring("learningRate
".length())));
            strLine = br.readLine();
            NN.setMomentum(Float.parseFloat(strLine.substring("momentum
".length())));
            strLine = br.readLine();
```

```
NN.setMaxIterations(Integer.parseInt(strLine.substring("maxIterations
".length())));
            strLine = br.readLine();
            NN.setTrainFile(strLine.substring("trainFile ".length()));
            strLine = br.readLine();
            NN.setTestFile(strLine.substring("testFile ".length()));

        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        NN.trainingDataNo = NN.dataNo(NN.getTrainFile()); // We calculate
the number of training patters
        NN.testingDataNo = NN.dataNo(NN.getTestFile()); // We calculate the
number of testing patterns

        NN.initialize(); // We initialize the network

        NN.normaliseData();
        int i;
        for(i=0;i<NN.normalisationTable[min].length;i++){

            System.out.print(NN.normalisationTable[min][i] + " ");
        }
        System.out.println();
        for(i=0;i<NN.normalisationTable[max].length;i++){

            System.out.print(NN.normalisationTable[max][i] + " ");
        }
        System.out.println();
    NN.trainAndEvaluate(); // We train and test the network


/*System.out.println("=================================================
================");
            System.out.println("Weighttable first " +
NN.weightTableFirst[0][0] + " " + NN.weightTableFirst[0][1] + " " +
NN.weightTableFirst[1][0] + " " + NN.weightTableFirst[1][1] + " ");
            System.out.println("Weighttable second " +
NN.weightTableSecond[0][0] + " " + NN.weightTableSecond[1][0] + " ");
            System.out.println("bias first " +
NN.weightTableBiasInputLayer[0] + " " + NN.weightTableBiasInputLayer[1] + "
");
```

```
            System.out.println("bias second" +
NN.weightTableBiasHiddenLayerOne[0]);
            */
        if (NN.watchTime){
           Date datefinished = new Date();
            NN.printWeights(NN);
           // display time and date using toString()
           System.out.println("Time and date started: " + date.toString());
           System.out.println("Time and date finished: " +
datefinished.toString());
        }


        System.out.println("***Finished***");
    }

    public void printWeights(ArtificialNeuralNetwork NN){
        int i,j;
        System.out.println("Weighttable first " );
        for (i=0;i<NN.weightTableFirst.length;i++){
            for (j=0;j<NN.weightTableFirst[0].length;j++    )
            {System.out.print(weightTableFirst[i][j] + "\t");
            }
            System.out.println();

        }
        System.out.println();
        System.out.println("Weighttable second ");
        for (i=0;i<NN.weightTableSecond.length;i++){
            for (j=0;j<NN.weightTableSecond[0].length;j++   )
            {System.out.print(weightTableSecond[i][j] + "\t");
            }
            System.out.println();

        }
        System.out.println();
System.out.println("Weighttable Third" );
for (i=0;i<NN.weightTableThird.length;i++){
    for (j=0;j<NN.weightTableThird[0].length;j++    )
    {System.out.print(weightTableThird[i][j] + "\t");
    }
    System.out.println();

}
```

```
System.out.println();

System.out.println("bias first " );
for (i=0;i<NN.weightTableBiasInputLayer.length;i++){

    System.out.print(weightTableBiasInputLayer[i] + "\t");

}System.out.println();
System.out.println("bias second" );
for (i=0;i<NN.weightTableBiasHiddenLayerOne.length;i++){

    System.out.print(weightTableBiasHiddenLayerOne[i] + "\t");

}
System.out.println();
System.out.println("bias Third" );
for (i=0;i<NN.weightTableBiasHiddenLayerTwo.length;i++){

    System.out.print(weightTableBiasHiddenLayerTwo[i] + "\t");

}
    }
}
```

Παράρτημα Β

Εδώ παραθέτω τα SQL επερωτήματα, τα οποία έδιναν από την βάση διαφορετικές μορφές δεδομένων

Β1. Ερώτημα το οποίο δίνει όλα τα δεδομένα ως έχουν στη βάση χωρίζοντας τα τυχαία σε δύο σύνολα.

```
SELECT CASE WHEN row_id <= 375
THEN 'Training'
ELSE 'Test'
END AS NNFlag,
t.*
FROM
( SELECT @rank := @rank + 1 AS ROW_ID ,H.*
FROM(SELECT ID,TOTAL_CONSUMPTION,URBAN,RURAL,
MOUNTAINOUS,PROPRIETARY,HOUSE, MAISONETTE,POOL,JACUZZI_SPA,INSULATION,
HOT_AIR_CONDITIONER,STORAGE_HEATERS, CENTRAL_HEATING_PETROLEUM,RADIATOR,
HALOGEN_HEATERS,FLOOR_HEATING,GAS_HEATERS, FIREPLACE, NONE_HEATING,
COLD_AIR_CONDITIONER,FAN, NONE_COOLING,
SQUARE_METERS,TOTAL_NUMBERS,ADULTS,UNDERAGE
FROM house_information_new as D, cons_01_12 as C
where D.Owner_ID = C.ID
and SQUARE_METERS!=-1 and TOTAL_NUMBERS!=-1 and ADULTS!=-1 and UNDERAGE !=-
1
ORDER BY RAND()) as H, (Select @rank := 0) AS ObligatoryAlias
) AS t
```

Β2. Ερώτημα το οποίο δημιουργεί το στοιχείο FLAT και διαγράφει τις αδικαιολόγητα αποκλίνουσες τιμές

```
SELECT CASE WHEN row_id <= 360
THEN 'Training'
ELSE 'Test'
END AS NNFlag,
t.*
FROM
( SELECT @rank := @rank + 1 AS ROW_ID ,H.*
```

```
FROM(SELECT ID,TOTAL_CONSUMPTION,URBAN,RURAL,
MOUNTAINOUS,PROPRIETARY,HOUSE,1-HOUSE-MAISONETTE as FLAT,
MAISONETTE,POOL,JACUZZI_SPA,INSULATION,
HOT_AIR_CONDITIONER,STORAGE_HEATERS, CENTRAL_HEATING_PETROLEUM,RADIATOR,
HALOGEN_HEATERS,FLOOR_HEATING,GAS_HEATERS, FIREPLACE, NONE_HEATING,
COLD_AIR_CONDITIONER,FAN, NONE_COOLING,
SQUARE_METERS,TOTAL_NUMBERS,ADULTS,UNDERAGE
FROM house_information_new as D, cons_01_12 as C
where D.Owner_ID = C.ID and TOTAL_CONSUMPTION>250 and TOTAL_CONSUMPTION<
UPLIMIT
and SQUARE_METERS!=-1 and TOTAL_NUMBERS!=-1 and ADULTS!=-1 and UNDERAGE !=-
1
ORDER BY RAND()) as H, (Select @rank := 0) AS ObligatoryAlias
) AS t
```

B3. Δημιουργία κατηγοριών στα αριθμητικά χαρακτηριστικά πλην της κατανάλωσης

```
SELECT CASE WHEN row_id <= 360
THEN 'Training'
ELSE 'Test'
END AS NNFlag,
t.*
FROM
( SELECT @rank := @rank + 1 AS ROW_ID ,H.*
FROM(SELECT ID,TOTAL_CONSUMPTION,URBAN,RURAL,
MOUNTAINOUS,PROPRIETARY,HOUSE,1-HOUSE-MAISONETTE as FLAT,
MAISONETTE,POOL,JACUZZI_SPA,INSULATION,
HOT_AIR_CONDITIONER,STORAGE_HEATERS, CENTRAL_HEATING_PETROLEUM,RADIATOR,
HALOGEN_HEATERS,FLOOR_HEATING,GAS_HEATERS, FIREPLACE, NONE_HEATING,
COLD_AIR_CONDITIONER,FAN, NONE_COOLING, CASE WHEN SQUARE_METERS < 150 THEN
1
ELSE 0
END as SQUARE_LT_150,
CASE WHEN SQUARE_METERS >= 150 AND SQUARE_METERS <= 200 THEN 1
ELSE 0
END as SQUARE_BETWEEN_150_200,
CASE WHEN SQUARE_METERS > 200 THEN 1
ELSE 0
END as SQUARE_GT_200,
CASE WHEN TOTAL_NUMBERS < 5 THEN 1
ELSE 0
END as TOTAL_NUMBERS_LT_5,
```

```
CASE WHEN TOTAL_NUMBERS >= 5 AND TOTAL_NUMBERS <= 9 THEN 1
ELSE 0
END as TOTAL_NUMBERS_BETWEEN_5_9,
CASE WHEN TOTAL_NUMBERS > 9 THEN 1
ELSE 0
END as TOTAL_NUMBERS_GT_9,
CASE WHEN ADULTS < 2 THEN 1
ELSE 0
END as ADULTS_LT_2,
CASE WHEN ADULTS >= 2 AND ADULTS <= 3 THEN 1
ELSE 0
END as ADULTS_BETWEEN_2_3,
CASE WHEN ADULTS > 3 THEN 1
ELSE 0
END as ADULTS_GT_3,
CASE WHEN UNDERAGE < 1 THEN 1
ELSE 0
END as UNDERAGE_LT_1,
CASE WHEN UNDERAGE >= 1 AND UNDERAGE <= 3 THEN 1
ELSE 0
END as UNDERAGE_BETWEEN_1_3,
CASE WHEN UNDERAGE > 3 THEN 1
ELSE 0
END as UNDERAGE_GT_3

FROM house_information_new as D, cons_01_12 as C
where D.Owner_ID = C.ID and TOTAL_CONSUMPTION>250 and TOTAL_CONSUMPTION<
UPLIMIT
and SQUARE_METERS!=-1 and TOTAL_NUMBERS!=-1 and ADULTS!=-1 and UNDERAGE !=-
1
ORDER BY RAND()) as H, (Select @rank := 0) AS ObligatoryAlias
) AS t
```

Β4 – Δημιουργία Κατηγοριών στην κατανάλωση αλλά όχι στα υπόλοιπα αριθμητικά χαρακτηριστικά

```
SELECT CASE WHEN row_id <= 360
THEN 'Training'
ELSE 'Test'
END AS NNFlag,
t.*
FROM
( SELECT @rank := @rank + 1 AS ROW_ID ,H.*
```

```
FROM(SELECT ID,
CASE WHEN TOTAL_CONSUMPTION < 800 THEN 1
ELSE 0
END as TOTAL_CONSUMPTION_BETWEEN_580_870,
CASE WHEN TOTAL_CONSUMPTION > 800 AND TOTAL_CONSUMPTION <= 1160 THEN 1
ELSE 0
END as TOTAL_CONSUMPTION_BETWEEN_870_1160,
CASE WHEN TOTAL_CONSUMPTION > 1160 AND TOTAL_CONSUMPTION <= 1500 THEN 1
ELSE 0
END as TOTAL_CONSUMPTION_BETWEEN_1160_1450,
CASE WHEN TOTAL_CONSUMPTION > 1500 THEN 1
ELSE 0
END as TOTAL_CONSUMPTION_GT_1500,
URBAN,RURAL, MOUNTAINOUS,PROPRIETARY,HOUSE,1-HOUSE-MAISONETTE as FLAT,
MAISONETTE,POOL,JACUZZI_SPA,INSULATION,
HOT_AIR_CONDITIONER,STORAGE_HEATERS, CENTRAL_HEATING_PETROLEUM,RADIATOR,
HALOGEN_HEATERS,FLOOR_HEATING,GAS_HEATERS, FIREPLACE, NONE_HEATING,
COLD_AIR_CONDITIONER,FAN, NONE_COOLING, SQUARE_METERS , TOTAL_NUMBERS ,
ADULTS, UNDERAGE

FROM house_information_new as D, cons_01_12 as C
where D.Owner_ID = C.ID and TOTAL_CONSUMPTION>250 and
TOTAL_CONSUMPTION<UPLIMIT
and SQUARE_METERS!=-1 and TOTAL_NUMBERS!=-1 and ADULTS!=-1 and UNDERAGE !=-
1
ORDER BY RAND()) as H, (Select @rank := 0) AS ObligatoryAlias
) AS t
```

B5 – Δημιουργία κατηγοριών στα αριθμητικά χαρακτηριστικά συμπεριλαμβανομένου της κατανάλωσης

```
SELECT CASE WHEN row_id <= 360
THEN 'Training'
ELSE 'Test'
END AS NNFlag,
t.*
FROM
( SELECT @rank := @rank + 1 AS ROW_ID ,H.*
FROM(SELECT ID,
CASE WHEN TOTAL_CONSUMPTION < 800 THEN 1
ELSE 0
```

```
END as TOTAL_CONSUMPTION_BETWEEN_580_870,
CASE WHEN TOTAL_CONSUMPTION > 800 AND TOTAL_CONSUMPTION <= 1160 THEN 1
ELSE 0
END as TOTAL_CONSUMPTION_BETWEEN_870_1160,
CASE WHEN TOTAL_CONSUMPTION > 1160 AND TOTAL_CONSUMPTION <= 1500 THEN 1
ELSE 0
END as TOTAL_CONSUMPTION_BETWEEN_1160_1450,
CASE WHEN TOTAL_CONSUMPTION > 1500 THEN 1
ELSE 0
END as TOTAL_CONSUMPTION_GT_1500,
URBAN,RURAL, MOUNTAINOUS,PROPRIETARY,HOUSE,1-HOUSE-MAISONETTE as FLAT,
MAISONETTE,POOL,JACUZZI_SPA,INSULATION,
HOT_AIR_CONDITIONER,STORAGE_HEATERS, CENTRAL_HEATING_PETROLEUM,RADIATOR,
HALOGEN_HEATERS,FLOOR_HEATING,GAS_HEATERS, FIREPLACE, NONE_HEATING,
COLD_AIR_CONDITIONER,FAN, NONE_COOLING, CASE WHEN SQUARE_METERS < 150 THEN
1
ELSE 0
END as SQUARE_LT_150,
CASE WHEN SQUARE_METERS >= 150 AND SQUARE_METERS <= 200 THEN 1
ELSE 0
END as SQUARE_BETWEEN_150_200,
CASE WHEN SQUARE_METERS > 200 THEN 1
ELSE 0
END as SQUARE_GT_200,
CASE WHEN TOTAL_NUMBERS < 5 THEN 1
ELSE 0
END as TOTAL_NUMBERS_LT_5,
CASE WHEN TOTAL_NUMBERS >= 5 AND TOTAL_NUMBERS <= 9 THEN 1
ELSE 0
END as TOTAL_NUMBERS_BETWEEN_5_9,
CASE WHEN TOTAL_NUMBERS > 9 THEN 1
ELSE 0
END as TOTAL_NUMBERS_GT_9,
CASE WHEN ADULTS < 2 THEN 1
ELSE 0
END as ADULTS_LT_2,
CASE WHEN ADULTS >= 2 AND ADULTS <= 3 THEN 1
ELSE 0
END as ADULTS_BETWEEN_2_3,
CASE WHEN ADULTS > 3 THEN 1
ELSE 0
END as ADULTS_GT_3,
CASE WHEN UNDERAGE < 1 THEN 1
ELSE 0
```

```
END as UNDERAGE_LT_1,
CASE WHEN UNDERAGE >= 1 AND UNDERAGE <= 3 THEN 1
ELSE 0
END as UNDERAGE_BETWEEN_1_3,
CASE WHEN UNDERAGE > 3 THEN 1
ELSE 0
END as UNDERAGE_GT_3

FROM house_information_new as D, cons_01_12 as C
where D.Owner_ID = C.ID and TOTAL_CONSUMPTION>250 and
TOTAL_CONSUMPTION<UPLIMIT
and SQUARE_METERS!=-1 and TOTAL_NUMBERS!=-1 and ADULTS!=-1 and UNDERAGE !=-
1
ORDER BY RAND()) as H, (Select @rank := 0) AS ObligatoryAlias
) AS t
```

Παράρτημα Γ

Στο παράρτημα αυτό παραθέτω τα έντυπα τα οποία έχουν διαμοιραστεί στο πλήθος για συλλογή δεδομένων. Επίσης παραθέτω ένα στιγμιότυπο της ιστοσελίδας που έχω κάνει για να προωθήσουμε το έργο, καθώς επίσης για να συλλέξουμε περισσότερα δεδομένα

Γ1 Ελληνικό έντυπο

ΕΝΤΥΠΟ ΕΘΕΛΟΝΤΙΚΗΣ ΣΥΜΜΕΤΟΧΗΣ ΣΤΟ ΕΡΕΥΝΗΤΙΚΟ ΠΡΟΓΡΑΜΜΑ SOCIAL ELECTRICITY

University of Cyprus

Εισαγωγή:

Η εφαρμογή **Κοινωνικός Ηλεκτρισμός** (Social Electricity) είναι μια νέα εφαρμογή στο Facebook που μας βοηθά να αντιληφθούμε την ηλεκτρική ενέργεια που καταναλώνουμε, μέσω συγκρίσεων με την αντίστοιχη ενέργεια που καταναλώνουν οι φίλοι μας και οι γείτονες μας. Πρόκειται για μια συνεργασία του Πανεπιστημίου Κύπρου και της Αρχής Ηλεκτρισμού Κύπρου (ΑΗΚ), με στόχο την μείωση της κατανάλωσης ενέργειας στη Κύπρο και της συνειδητοποίησης των Κύπριων πολιτών για θέματα ενέργειας.

Οι συγκρίσεις με γείτονες και φίλους μας προσφέρουν μια ένδειξη σχετικά με τα ποσά ηλεκτρικής ενέργειας που καταναλώνουμε, δεν μπορούν όμως να είναι επακριβείς, επειδή το κάθε άτομο ζει σε διαφορετική οικία με διαφορετικά χαρακτηριστικά.

Στη προσπάθεια μας να κάνουμε την εφαρμογή πιο αποτελεσματική για τους Κύπριους πολίτες, θα θέλαμε να προσθέσουμε την δυνατότητα να καταχωρεί ο κάθε Κύπριος τα στοιχεία της οικίας του και να μπορεί να συγκρίνει αυτόματα την ηλεκτρική του κατανάλωση μόνο με Κύπριους πολίτες που πληρούν παρόμοια χαρακτηριστικά. Έτσι, θα γνωρίζει το κάθε άτομο αν καταναλώνει χαμηλά/μέτρια/ψηλά ποσά ηλεκτρικής ενέργειας, μέσω συγκρίσεων μεταξύ ανθρώπων που ζουν οπουδήποτε στη Κύπρο, αλλά η οικία τους προσφέρει παρόμοια χαρακτηριστικά με αυτούς!

Για να είναι σε θέση όμως η εφαρμογή να παρέχει αυτή τη δυνατότητα, πρέπει να ενημερωθεί για τις σχέσεις μεταξύ πραγματικών παραμέτρων που σχετίζονται με την κατανάλωση ηλεκτρικής ενέργειας, όπως το μέγεθος του σπιτιού ή ο αριθμός των κατοίκων του.

Γι' αυτό τον λόγο ζητούμε την δική σας συμβολή. Σας ζητούμε να αφιερώσετε **μόνο 5 λεπτά** από τον χρόνο σας, για να μας βοηθήσετε εθελοντικά να προσφέρουμε αυτή την δυνατότητα στους Κύπριους καταναλωτές. Με αυτό τον τρόπο, αφενός βοηθάτε τον τόπο σας, επειδή συνεισφέρετε σε μια υπηρεσία που στόχο έχει την ορθολογιστική χρήση ηλεκτρικής ενέργειας και αφετέρου συμμετέχετε σε μια ερευνητική πρωτοβουλία με καινοτομία ανά το παγκόσμιο!

Σας ευχαριστούμε εκ των προτέρων για τον πολύτιμο σας χρόνο.

Σημείωση:

***Οι πληροφορίες που θα μας δώσετε είναι* πλήρως εμπιστευτικές,** και θα χρησιμοποιηθούν **μόνο για ερευνητικούς σκοπούς** από την ερευνητική ομάδα της εφαρμογής **Social Electricity**. Σας εγγυόμαστε και δεσμευόμαστε ότι οι πληροφορίες αυτές δεν θα δοθούν σε τρίτους και ότι θα σεβαστούμε πλήρως την ιδιωτικότητα σας. Επίσης σας παρακαλούμε όπως συμπληρώσετε ορθά όλα τα παρακάτω στοιχεία. Εάν τα στοιχεία είναι εσφαλμένα, ελλιπή ή ανακριβή, τότε η εφαρμογή δεν θα λειτουργεί με τον επιθυμητό τρόπο.

1. Τα παρακάτω στοιχεία χρησιμοποιούνται για εύρεση της ηλεκτρικής σας κατανάλωσης από την ΑΗΚ:

Ονοματεπώνυμο ατόμου στο οποίο είναι καταχωρημένος ο λογαριασμός της ΑΗΚ:
……

Οδός κατοικίας: ……

Ταχυδρομικός Κώδικας: ……

Χωριό:…… Πόλη: ……

2. Παρακαλώ σημειώστε «Χ» στο κουτί της επιλογής σας:

Περιοχή:	☐ Αστική	☐ Αγροτική	☐ Βουνό
Το σπίτι σας είναι:	☐ Ιδιόκτητο	☐ Ενοικιαζόμενο	
Τύπος σπιτιού:	☐ Κατοικία	☐ Διαμέρισμα	☐ Μεζονέτα
Το σπίτι έχει πισίνα:	☐ Ναι	☐ Όχι	
Το σπίτι έχει Jacuzzi/SPA:	☐ Ναι	☐ Όχι	
Το σπίτι είναι θερμομονωτικό:	☐ Ναι	☐ Όχι	

3. Τύποι θέρμανσης που υπάρχουν στο σπίτι: (Σημειώστε με «**Χ**» στα κουτιά των επιλογών που ισχύουν**)**

☐ Κεντρική θέρμανση (πετρελαίου) ☐ Καλοριφέρ ☐ Θερμάνσεις αλογόνου ☐ Άλλο (.....)

☐ Θερμαινόμενο πάτωμα ☐ Σόμπες Γκαζιού ☐ Τζάκι ☐ Κανένας

4. Τύποι Κλιματισμού που υπάρχουν στο σπίτι:

☐ Air-condition (Σύστημα κλιματισμού) ☐ Ανεμιστήρας ☐ Άλλο (.....) ☐ Κανένας

5. Συμπληρώστε τα κουτιά με τον αντίστοιχο αριθμό που εφαρμόζεται στην κατοικία σας:

Τετραγωνικά μέτρα σπιτιού: ☐ Συνολικός αριθμός δωματίων: ☐

Ενήλικες κάτοικοι : ☐ Ανήλικοι κάτοικοι: ☐

6. Πιστεύετε ότι υπάρχει κάποιος άλλος σημαντικός παράγοντας που σχετίζεται με την ηλεκτρική κατανάλωση και έπρεπε να ερωτηθεί? Αν ναι, ποιος είναι αυτός?

.....

Υπογράφοντας συναινώ στην παραχώρηση των στοιχείων της κατανάλωσης μου και μόνο, από την ΑΗΚ προς την ομάδα ανάπτυξης του **Social Electricity**, **μόνο για ερευνητικούς σκοπούς**.

_____ _____

Όνομα (ολογράφως) Υπογραφή

Γ2 Αγγλικό έντυπο

FORM FOR VOLUNTARY PARTICIPATION IN SOCIAL ELECTRICITY RESEARCH PROGRAM

University of Cyprus

Introduction:

Social Electricity is a Facebook application that helps people to perceive their electricity footprint, by means of comparison with the corresponding electrical energy consumed by their friends and neighbors. This application is collaboration between the University of Cyprus and the Electricity Authority of Cyprus (EAC), and aims to raise the energy awareness of Cypriot citizens, in order to conserve electricity.

Although the comparisons with neighbors and friends may provide an indication of the consumed amount of electricity, they may not be as accurate, because each person lives in a different house with different physical characteristics.

In our effort to make the application more effective for Cypriot citizens, we intend to add, for every Cypriot, the ability to list the main features of his house, and then compare his energy consumption only with other Cypriot citizens whose houses have similar characteristics. Therefore, each person will understand whether he consumes low/moderate/high amounts of electricity, through comparisons with other people who may live anywhere in Cyprus, but their houses offer similar features and characteristics!

To provide this new service, **Social Electricity** must be aware of the actual features/parameters of houses around Cyprus (e.g. the size of the house, the number of inhabitants), and associate these characteristics with the electrical consumption in these houses through the year.

For this reason, we ask for your help. We please you to dedicate **only 5 minutes** of your time to help us offer this opportunity to Cypriot consumers. By supporting our research project, not only you contribute to a great initiative for conserve electricity in Cyprus, but you also become part of an innovative attempt worldwide!

Thank you in advance for your valuable time.

Note:

The information you provide in this questionnaire is absolutely confidential and will be solely used **for research purposes** by the **Social Electricity** research team. We stress that this information will not be given to any third parties for any reason, and that your privacy will be highly respected.

Please fill in all the information below correctly. If the information is incorrect, incomplete or inaccurate, then our project will not perform as desired.

1. The following details are used to locate your electrical consumption by EAC:

Name and surname of the person who owns the account which is registered in EAC:
……

Address: ……

Postal Code: .....

Village: ..... Town: .....

2. Please mark with an "X" the boxes of your choice:

Region:	☐ Urban	☐ Rural	☐ Mountainous
Your residence is:	☐ Proprietary	☐ Rented	
Type of residence:	☐ House	☐ Flat	☐ Maisonette
Residence has a pool?	☐ Yes	☐ No	
Residence has Jacuzzi/SPA?	☐ Yes	☐ No	
Residence is insulated?	☐ Yes	☐ No	

3. Types of heating in the home: (Please mark with an "**X**" the options that apply**)**

☐ Central heating (Petroleum) ☐ Radiator ☐ Halogen heaters ☐ Other (.....)

☐ Floor Heating ☐ Gas heaters ☐ Fireplace ☐ None

4. Types of cooling in the home:

☐ Air-conditioner ☐ Fan ☐ Other (.....) ☐ None

5. Fill in the boxes with the number that applies to your residence:

Square meters of the house: ☐ Total Number of Rooms: ☐

Adult Residents: ☐ Underage Residents: ☐

6. Do you believe there are any other important factors associated with domestic electrical consumption which should be considered? If so, which are they?

.....

By signing I consent to the grant of my electrical consumption from EAC to the **Social Electricity** development team for **research purposes only**.

_____ _____

Name (Written out in full) Signature

Γ3 Ανανεωμένη μορφή Ελληνικού εντύπου

ΕΝΤΥΠΟ ΕΘΕΛΟΝΤΙΚΗΣ ΣΥΜΜΕΤΟΧΗΣ ΣΤΟ ΕΡΕΥΝΗΤΙΚΟ ΠΡΟΓΡΑΜΜΑ SOCIAL ELECTRICITY

University of Cyprus

Εισαγωγή:

Η εφαρμογή **Κοινωνικός Ηλεκτρισμός** (Social Electricity) είναι μια νέα εφαρμογή στο Facebook που μας βοηθά να αντιληφθούμε την ηλεκτρική ενέργεια που καταναλώνουμε, μέσω συγκρίσεων με την αντίστοιχη ενέργεια που καταναλώνουν οι φίλοι μας και οι γείτονες μας. Πρόκειται για μια συνεργασία του Πανεπιστημίου Κύπρου και της Αρχής Ηλεκτρισμού Κύπρου (ΑΗΚ), με στόχο την μείωση της κατανάλωσης ενέργειας στη Κύπρο και της συνειδητοποίησης των Κύπριων πολιτών για θέματα ενέργειας.

Οι συγκρίσεις με γείτονες και φίλους μας προσφέρουν μια ένδειξη σχετικά με τα ποσά ηλεκτρικής ενέργειας που καταναλώνουμε, δεν μπορούν όμως να είναι επακριβείς, επειδή το κάθε άτομο ζει σε διαφορετική οικία με διαφορετικά χαρακτηριστικά.

Στη προσπάθεια μας να κάνουμε την εφαρμογή πιο αποτελεσματική για τους Κύπριους πολίτες, θα θέλαμε να προσθέσουμε την δυνατότητα να καταχωρεί ο κάθε Κύπριος τα στοιχεία της οικίας του και να μπορεί να συγκρίνει αυτόματα την ηλεκτρική του κατανάλωση μόνο με Κύπριους πολίτες που πληρούν παρόμοια χαρακτηριστικά. Έτσι, θα γνωρίζει το κάθε άτομο αν καταναλώνει χαμηλά/μέτρια/ψηλά ποσά ηλεκτρικής ενέργειας, μέσω συγκρίσεων μεταξύ ανθρώπων που ζουν οπουδήποτε στη Κύπρο, αλλά η οικία τους προσφέρει παρόμοια χαρακτηριστικά με αυτούς!

Για να είναι σε θέση όμως η εφαρμογή να παρέχει αυτή τη δυνατότητα, πρέπει να ενημερωθεί για τις σχέσεις μεταξύ πραγματικών παραμέτρων που σχετίζονται με την κατανάλωση ηλεκτρικής ενέργειας, όπως το μέγεθος του σπιτιού ή ο αριθμός των κατοίκων του.

Γι' αυτό τον λόγο ζητούμε την δική σας συμβολή. Σας ζητούμε να αφιερώσετε **μόνο 5 λεπτά** από τον χρόνο σας, για να μας βοηθήσετε εθελοντικά να προσφέρουμε αυτή την δυνατότητα στους Κύπριους καταναλωτές. Με αυτό τον τρόπο, αφενός βοηθάτε τον τόπο σας, επειδή συνεισφέρετε σε μια υπηρεσία που στόχο έχει την ορθολογιστική χρήση ηλεκτρικής ενέργειας και αφετέρου συμμετέχετε σε μια ερευνητική πρωτοβουλία με καινοτομία ανά το παγκόσμιο!

Σας ευχαριστούμε εκ των προτέρων για τον πολύτιμο σας χρόνο.

Σημείωση:

***Οι πληροφορίες που θα μας δώσετε είναι* πλήρως εμπιστευτικές,** και θα χρησιμοποιηθούν **μόνο για ερευνητικούς σκοπούς** από την ερευνητική ομάδα της εφαρμογής **Social Electricity**. Σας εγγυόμαστε και δεσμευόμαστε ότι οι πληροφορίες αυτές δεν θα δοθούν σε τρίτους και ότι θα σεβαστούμε πλήρως την ιδιωτικότητα σας. Επίσης σας παρακαλούμε όπως συμπληρώσετε ορθά όλα τα παρακάτω στοιχεία. Εάν τα στοιχεία είναι εσφαλμένα, ελλιπή ή ανακριβή, τότε η εφαρμογή δεν θα λειτουργεί με τον επιθυμητό τρόπο.

1. Τα παρακάτω στοιχεία χρησιμοποιούνται για εύρεση της ηλεκτρικής σας κατανάλωσης από την ΑΗΚ:

Ονοματεπώνυμο ατόμου στο οποίο είναι καταχωρημένος ο λογαριασμός της ΑΗΚ:
……

Οδός κατοικίας: ……

Ταχυδρομικός Κώδικας: …… Χωριό:…… Πόλη: ……

Τηλέφωνο: …… Αρ Ταυτότητας:……

Αρ. Λογαριασμού Οικίας στην ΑΗΚ (Εντεκαψήφιος αριθμός ο οποίος βρίσκεται πάνω στις καταστάσεις ηλεκτρικής κατανάλωσης που στέλνει η ΑΗΚ ανά διμηνία) ……

2. Παρακαλώ σημειώστε «Χ» στο κουτί της επιλογής σας:

Περιοχή:	☐ Αστική	☐ Αγροτική	☐ Βουνό
Το σπίτι σας είναι:	☐ Ιδιόκτητο	☐ Ενοικιαζόμενο	
Τύπος σπιτιού:	☐ Κατοικία	☐ Διαμέρισμα	☐ Μεζονέτα
Το σπίτι έχει πισίνα:	☐ Ναι	☐ Όχι	
Το σπίτι έχει Jacuzzi/SPA:	☐ Ναι	☐ Όχι	
Το σπίτι είναι θερμομονωτικό:	☐ Ναι	☐ Όχι	

3. Τύποι θέρμανσης που υπάρχουν στο σπίτι: (Σημειώστε με «**Χ**» στα κουτιά των επιλογών που ισχύουν**)**

☐ Θερμοσυσσωρευτές ΑΗΚ ☐ Θερμαινόμενο πάτωμα ☐ Θερμάνσεις αλογόνου ☐ Κεντρική θέρμανση (πετρελαίου)

☐ Καλοριφέρ ☐ Σόμπες Γκαζιού ☐ Τζάκι ☐ Δεν υπάρχει

☐ Air-condition (Σύστημα κλιματισμού με ζεστό αέρα) ☐ Άλλο (.....)

4. Τύποι Κλιματισμού που υπάρχουν στο σπίτι:

☐ Air-condition (Σύστημα κλιματισμού) ☐ Ανεμιστήρας ☐ Άλλο (.....) ☐ Κανένας

5. Συμπληρώστε τα κουτιά με τον αντίστοιχο αριθμό που εφαρμόζεται στην κατοικία σας:

* Για να είναι ορθά τα αποτελέσματα της έρευνας, τα πιο κάτω στοιχεία είναι απαραίτητο να συμπληρωθούν.

Τετραγωνικά μέτρα σπιτιού*: ☐ Ενήλικες κάτοικοι*: ☐ Αριθμός υπνοδωματίων*: ☐

Συνολικός αριθμός δωματίων*: ☐ Ανήλικοι κάτοικοι* : ☐

6. Σε ποιο βαθμό πιστεύετε ότι είστε ενεργειακά συνειδητοποιημένοι?

☐ Πάρα πολύ ☐ Πολύ ☐ Αρκετά ☐ Λίγο ☐ Καθόλου

7. **Πιστεύετε ότι υπάρχει κάποιος άλλος σημαντικός παράγοντας που σχετίζεται με την ηλεκτρική κατανάλωση και έπρεπε να ερωτηθεί? Αν ναι, ποιος είναι αυτός?**

.....

Υπογράφοντας συναινώ στην παραχώρηση των στοιχείων της κατανάλωσης μου και μόνο, από την ΑΗΚ προς την ομάδα ανάπτυξης του **Social Electricity**, **μόνο για ερευνητικούς σκοπούς**.

_____ _____

Όνομα (ολογράφως) Υπογραφή

Γ4 Ιστοσελίδα προώθησης έρευνας

Social Electricity

Ελληνικά English

Social Electricity Συμμετοχή σε έρευνα του Social Electricity

Η εφαρμογή Κοινωνικός Ηλεκτρισμός (Social Electricity) είναι μια νέα εφαρμογή στο Facebook που μας βοηθά να αντιληφθούμε την ηλεκτρική ενέργεια που καταναλώνουμε, μέσω συγκρίσεων με την αντίστοιχη ενέργεια που καταναλώνουν οι φίλοι μας και οι γείτονες μας.

Πρόκειται για μια συνεργασία του Πανεπιστημίου Κύπρου και της Αρχής Ηλεκτρισμού Κύπρου (ΑΗΚ), με στόχο την μείωση της κατανάλωσης ενέργειας στη Κύπρο και της συνειδητοποίησης των Κύπριων πολιτών για θέματα ενέργειας. Οι συγκρίσεις με γείτονες και φίλους μας προσφέρουν μια ένδειξη σχετικά με τα ποσά ηλεκτρικής ενέργειας που καταναλώνουμε, δεν μπορούν όμως να είναι επακριβείς, επειδή το κάθε άτομο ζει σε διαφορετική οικία με διαφορετικά χαρακτηριστικά.

Στη προσπάθεια μας να κάνουμε την εφαρμογή πιο αποτελεσματική για τους Κύπριους πολίτες, θα θέλαμε να προσθέσουμε την δυνατότητα να καταχωρεί ο κάθε Κύπριος τα στοιχεία της οικίας του και να μπορεί να συγκρίνει αυτόματα την ηλεκτρική του κατανάλωση μόνο με Κύπριους πολίτες που πληρούν παρόμοια χαρακτηριστικά. Έτσι, θα γνωρίζει το κάθε άτομο αν καταναλώνει χαμηλά/μέτρια/ψηλά ποσά ηλεκτρικής ενέργειας, μέσω συγκρίσεων μεταξύ ανθρώπων που ζουν οπουδήποτε στη Κύπρο, αλλά η οικία τους προσφέρει παρόμοια χαρακτηριστικά με αυτούς! Για να είναι σε θέση όμως η εφαρμογή να παρέχει αυτή τη δυνατότητα, πρέπει να ενημερωθεί για τις σχέσεις μεταξύ πραγματικών παραμέτρων που σχετίζονται με την κατανάλωση ηλεκτρικής ενέργειας, όπως το μέγεθος του σπιτιού ή ο αριθμός των κατοίκων του.

Γι' αυτό τον λόγο ζητούμε την δική σας συμβολή. Σας ζητούμε να αφιερώσετε μόνο 5 λεπτά από τον χρόνο σας, για να μας βοηθήσετε εθελοντικά να προσφέρουμε αυτή την δυνατότητα στους Κύπριους καταναλωτές. Με αυτό τον τρόπο, αφενός βοηθάτε τον τόπο σας, επειδή συνεισφέρετε σε μια υπηρεσία που στόχο έχει την ορθολογιστική χρήση ηλεκτρικής ενέργειας και αφετέρου συμμετέχετε σε μια ερευνητική πρωτοβουλία με καινοτομία ανά το παγκόσμιο! Σας ευχαριστούμε εκ των προτέρων για τον πολύτιμο σας χρόνο.

Σημείωση: Οι πληροφορίες που θα μας δώσετε είναι πλήρως εμπιστευτικές, και θα χρησιμοποιηθούν μόνο για ερευνητικούς σκοπούς από την ερευνητική ομάδα της εφαρμογής Social Electricity. Σας εγγυόμαστε και δεσμευόμαστε ότι οι πληροφορίες αυτές δεν θα δοθούν σε τρίτους και ότι θα σεβαστούμε πλήρως την ιδιωτικότητα σας. Επίσης σας παρακαλούμε όπως συμπληρώσετε ορθά όλα τα παρακάτω στοιχεία. Εάν τα στοιχεία είναι εσφαλμένα, ελλιπή ή ανακριβή, τότε η εφαρμογή δεν θα λειτουργεί με τον επιθυμητό τρόπο.

Στα αριστερά, μπορείτε να δείτε ένα στιγμιότυπο της εφαρμογής Social Electricity στο Facebook.

Ελληνικό Έντυπο

Κατεβάστε το ελληνικό έντυπο από εδώ

Παρακαλούμε στείλτε μας τη συμπληρωμένη φόρμα στο info@socialelectricity.net

FOLLOW US