ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Implementation of power theft detection algorithms based on neural networks and fuzzy logic techniques

Υλοποίηση αλγορίθμων εντοπισμού ρευματοκλοπής με χρήση νευρωνικών δικτύων και ασαφούς λογικής

Όνομα : Τιτάκης Γεώργιος
Αριθμός Μητρώου : 2011030101

Χανιά, Νοέμβριος 2019

Acknowledgements

First and foremost, I have to thank my research supervisor, Mr. Stavrakakis Georgios. Without his assistance and dedicated involvement in every step throughout the process, this thesis would have never been accomplished. I would like to thank you very much for your support and understanding over these past two years.

Also, I would like to thank Prof. Zervakis Michalis and Dr Sergaki Eleftheria for their participation in the three party committee of this thesis.

I would also like to express my gratitude to Mr. Blazakis Konstantinos, who provided me with the energy consumption data, guided me with his advice through all my research and helped me to gain insight about the overall problem. His experience with power theft detection techniques and his previous avocation on the subject, have taught me more than I could ever give him credit here.

Finally, I would like to thank my family and friends for their encouragement and invaluable support throughout my education.

Ευχαριστίες

Πρώτα απ’ όλα, θέλω να ευχαριστήσω τον επιβλέποντα της διπλωματικής εργασίας μου, κ. Γεώργιο Σταυρακάκη, για την πολύτιμη βοήθεια και καθοδήγησή του κατά τη διάρκεια της δουλειάς μου. Επίσης, είμαι ευγνώμων στα υπόλοιπα μέλη της εξεταστικής επιτροπής της διπλωματικής εργασίας μου, τον κ. Ζερβάκη Μιχαήλ, καθηγητή ΗΜΜΥ και τη Δρ. Σεργάκη Ελευθερία, μέλος ΕΔΙΠ, για την προσεκτική ανάγνωση της εργασίας μου και για τις πολύτιμες υποδείξεις τους.

Θα ήθελα επίσης να ευχαριστήσω τον υποψήφιο διδάκτορα κ. Μπλαζάκη Κωνσταντίνο, ο οποίος μου παραχώρησε τα δεδομένα κατανάλωσης ηλεκτρικής ενέργειας, με καθοδήγησε με τις συμβουλές του καθόλη τη διάρκεια της ερευνάς και με βοήθησε να αποκτήσω μια εικόνα για το συνολικό πρόβλημα. Η εμπειρία του πάνω στην ανίχνευση ρευματοκλοπής και οι ερευνητικές εργασίες του πάνω στο θέμα, αποδείχθηκαν καταλυτικές για την ολοκλήρωση αυτής της διπλωματικής.

Πάνω απ’ όλα, είμαι ευγνώμων στους γονείς μου, για την ολόψυχη αγάπη και υποστήριξή τους όλα αυτά τα χρόνια. Αφιερώνω αυτή την εργασία στην μητέρα μου και στον πατέρα μου.

Abstract

The main issue for which we have written this thesis is non-technical losses detection in a smart electrical system. With the implementation of data mining techniques for analyzing the electricity consumption patterns of consumers, we identify the illegal consumers based on irregularities in their consumption data.

Distribution of electricity involves significant Technical Losses (TL) as well as Non-Technical Losses (NTL). Illegal consumption of electricity or electricity theft constitutes a major part of NTL. This thesis presents four different algorithmic approaches which are used to detect abnormalities in the received energy consumption readings.

First of all, we gathered the energy consumption data, taken from a real smart grid network in Ireland [50] and prepared them for each method. Applying features, normalization techniques and principal component analysis for the dimensional reduction of smart meter readings, we managed to adjust the energy consumption data to a better form to work with. Then we tested the five algorithmic methods in order to classify the consumers to legal or illegal.

The tools that used in this thesis in order to create neural networks for the classification of illegal and legal consumers, were Fast Artificial Neural Network Library (FANN) and Deep Learning Toolbox (or Neural Network Toolbox). For comparison, we implemented three more methods, which were not based on neural networks. The first used the LibOPF library for the design of optimum-path forest classifier, the second Support Vector Machines (SVM) and the third Neuro-Fuzzy Inference Systems.

Finally, we discussed the results of the above electricity theft detection techniques we experimented with. We observed that almost all the algorithmic methods which were used, gave us very satisfactory results for the detection of irregularities in energy consumption data.

Περίληψη

Η ρευματοκλοπή είναι ένα διαδεδομένο φαινόμενο όχι μόνο στις υποανάπτυκτες χώρες αλλά σε όλο τον κόσμο. Μελέτες δείχνουν αύξηση των ποσοστών κλοπής ηλεκτρικής ενέργειας σε πολλές χώρες και οι οικονομικές απώλειες για ορισμένα συστήματα είναι τόσο σημαντικές ώστε οι επιχειρήσεις κοινής ωφέλειας να βρίσκονται σε χρηματοπιστωτική αναταραχή. Σαν αποτέλεσμα, δεν είναι δυνατή η πραγματοποίηση επενδύσεων για τη βελτίωση του συστήματος και την προσθήκη πρόσθετης δυναμικότητας, όπως επίσης δεν είναι δυνατή η κάλυψη δανείων και πληρωμών και ο καταναλωτής αντιμετωπίζει αυξημένα τέλη ηλεκτρικής ενέργειας. Ακόμη και σε αποδοτικά συστήματα, οι απώλειες ηλεκτρικής ενέργειας λόγω κλοπής, μπορούν να αποφέρουν εκατομμύρια δολάρια ετησίως σε χαμένα έσοδα.

Τα μελλοντικά έξυπνα δίκτυα θα πρέπει να επιδείξουν καλύτερη εποπτεία σε διάφορους παράγοντες στα δίκτυα διανομής, να αυξήσουν την αξιοπιστία τους, την ευελιξία τους και να μειώσουν τις απώλειες μεταφοράς και διανομής. Με την ανάπτυξη προηγμένης τεχνολογίας μετρητών στο ηλεκτρικό δίκτυο, η κλοπή ενέργειας γίνεται πιο περίπλοκη και υιοθετούνται πολλές νέες τεχνολογίες για την επίλυση αυτού του προβλήματος. Η κλοπή ηλεκτρικής ενέργειας στις διάφορες μορφές της μπορεί να μειωθεί και να διατηρηθεί υπό έλεγχο μόνο από την ισχυρή και δυναμική δράση των οργανώσεων του ενεργειακού τομέα. Η στρατηγική και η δράση πρέπει να βασίζονται στη πλήρη κατανόηση της ειδικής φύσης του προβλήματος της κλοπής. Κάθε σύστημα ισχύος έχει τις δικές του μοναδικές ιδιότητες και μόνο γνωρίζοντας το σύστημα και το πρόβλημα μπορούν να σχεδιαστούν και να εφαρμοστούν αποτελεσματικές λύσεις.

Δεδομένου ότι ένα υψηλό επίπεδο κλοπής ηλεκτρικής ενέργειας συνδέεται με τη διαφθορά, η ανάλυση δεν μπορεί να περιοριστεί στις τεχνικές και διαχειριστικές προοπτικές και πρέπει να εφαρμοστεί διεπιστημονική προσέγγιση. Η κλοπή ως δραστηριότητα σε ορισμένα συστήματα συνδέεται στενά με τη διακυβέρνηση και με το κοινωνικό, οικονομικό και πολιτικό περιβάλλον. Σε μια γενική κουλτούρα διαφθοράς ως τρόπο ζωής, η κλοπή ηλεκτρισμού μπορεί να μειωθεί σε μέτρια επίπεδα με τεχνικές / μηχανικές μεθόδους, αλλά είναι μια άνιση μάχη για να μειωθεί δραστικά ο ρυθμός κλοπής ηλεκτρικού ρεύματος όσο συνεχίζεται η εκτεταμένη διαφθορά. Η μείωση της ρευματοκλοπής και η διατήρησή της εντός εύλογων ορίων είναι πιθανότερο να είναι επιτυχής σε συστήματα με μια καλή κυβερνητική πολιτική. Αυτό οφείλεται στο γεγονός ότι οι μηχανισμοί μείωσης ρευματοκλοπής βρίσκουν ένα φιλικό περιβάλλον για την εφαρμογή και τη λειτουργία τους.

Οι τεχνολογικές καινοτομίες (έξυπνοι μετρητές) καθιστούν αυτό το έργο πιο εύκολο αν υπάρχουν και οι ανάλογες διαχειριστικές δεξιότητες. Τα συστήματα ηλεκτρικής ενέργειας μπορούν να αναδιαρθρωθούν σε κάθε τομέα τους έτσι ώστε να λειτουργούν σε πλαίσια ευγενή ανταγωνισμού, όπου η αποδοτικότητα και η αποτελεσματικότητα στην παροχή υπηρεσιών να είναι τόσο αρετές όσο και ανάγκες.

Ο κύριος λόγος της συγγραφής αυτής της εργασίας είναι η ανίχνευση των μη τεχνικών απωλειών σε ένα έξυπνο ηλεκτρικό δίκτυο. Με τη χρήση αλγορίθμων-τεχνικών για την ανάλυση των μοτίβων κατανάλωσης των πελατών, αναγνωρίζουμε τους παράνομους καταναλωτές με βάση τις ανωμαλίες που εντοπίζονται στα δεδομένα κατανάλωσης τους.

Η διανομή και μεταφορά του ηλεκτρικού ρεύματος περιλαμβάνει σημαντικές τεχνικές όπως επίσης και μη τεχνικές απώλειες. Η παράνομη κατανάλωση του ηλεκτρικού ρεύματος αποτελεί σημαντικό μερίδιο των μη τεχνικών απωλειών. Με την έλευση προηγμένων τεχνολογιών μέτρησης, δεδομένα κατανάλωσης ενέργειας είναι διαθέσιμα στις υπηρεσίες σε πραγματικό χρόνο ,πράγμα που μπορεί να χρησιμοποιηθεί για την αναγνώριση των παράνομων πελατών. Η διπλωματική αυτή παρουσιάζει 5 διαφορετικές αλγοριθμικές προσεγγίσεις που χρησιμοποιούνται για την αναγνώριση ανωμαλιών στη κατανάλωση ενέργειας.

Πρώτα, διαβάζουμε τα δεδομένα κατανάλωσης ηλεκτρικής ενέργειας των καταναλωτών, τα οποία μας παραχωρήθηκαν από ένα πραγματικό ηλεκτρικό δίκτυο στην Ιρλανδία [50]. Τα δεδομένα αυτά, όπως μπορείτε να δείτε πιο αναλυτικά στο κεφάλαιο 3, περιείχαν 6 περιπτώσεις με διαφορετικά ποσοστά και είδη κλοπής ανά περίπτωση. Συγκεκριμένα, είχαμε

- Δεδομένα με μερική κλοπή 30-50%
- Δεδομένα με μερική κλοπή 50-70%
- Δεδομένα με υπερφόρτωση 40-60%
- Δεδομένα με υπερφόρτωση 60-80%
- Δεδομένα με υπερφόρτωση 80-100%
- Δεδομένα με υπερφόρτωση 60-80% και μερική κλοπή 50-70%

Ελέγξαμε κάθε περίπτωση κλοπής ηλεκτρικής ενέργειας από τις παραπάνω σε 30λεπτες (δηλαδή για κάθε καταναλωτή είχαμε μία μέτρηση ανά 30 λεπτά) μετρήσεις όπου είχαμε στη κατοχή μας. Μετατρέψαμε τις 30λεπτες μετρήσεις σε 8ωρες (δηλαδή για κάθε καταναλωτή είχαμε μία μέτρηση ανά 8 ώρες) και δουλέψαμε κυρίως με αυτές καθώς τα αποτελέσματα σε σύγκριση με τις 30λεπτες μετρήσεις ήταν σε γενικές γραμμές ίδια. Έπειτα, με τη εξαγωγή features από τα δεδομένα , τεχνικών κανονικοποίησης και τη μέθοδο PCA (principal component analysis) επεξεργαστήκαμε τα δεδομένα έτσι ώστε να είναι πιο αποδοτικά για τις 5 διαφορετικές αλγοριθμικές μεθόδους που ακολούθησαν με στόχο να κατηγοριοποιήσουν τους καταναλωτές σε νόμιμους ή παράνομους. Οι μεθόδοι που χρησιμοποιήθηκαν είναι οι εξής:

Multilayer Artificial Neural Networks με χρήση της Fast Artificial Neural Network (FANN) Library

Η FANN βιβλιοθήκη που χρησιμοποιήθηκε σε αυτή τη μέθοδο μέσω τερματικού Linux, ήταν εύκολη στη χρήση, γρήγορη (ταχύτερη από όλες τις άλλες μεθόδους εκτός από τη μέθοδο με χρήση της LibOPF βιβλιοθήκης) και ευέλικτη (μπορούσαμε να προσαρμόσουμε πολλές παραμέτρους και λειτουργίες σε κίνηση). Τα δεδομένα ηλεκτρικής ενέργειας χρειάστηκαν κάποια επεξεργασία για να μπορούν να χρησιμοποιηθούν ως είσοδο στη βιβλιοθήκη, αλλά τα αποτελέσματα αυτής της μεθόδου μας επιβραβεύουν.

Από όλες τις μεθόδους που δοκιμάστηκαν σε αυτή τη διπλωματική, η μέθοδος με multilayer artificial neural networks ήταν μια από τις καλύτερες τόσο στην απόδοση όσο και στην ταχύτητα. Η απόδοση εντοπισμού ρευματοκλοπής κυμαίνονταν από 89% -97% (ανάλογα με την περίπτωση που δοκιμάστηκε) και γι 'αυτό συνιστάται απόλυτα για την ανίχνευση ρευματοκλοπής.

Neural Networks με χρήση του Deep Learning Toolbox

Η μέθοδος αυτή έγινε με χρήση του λογισμικού της Matlab και συγκεκριμένα του εργαλείου Deep Learning Toolbox. Οπως μπορείτε να δείτε και στον πίνακα 0.1, λειτούργησε πολύ καλά στην αναγνώριση των παράνομων καταναλωτών με την απόδοση στην ανίχνευση ρευματοκλοπής να κυμαίνεται από 84%-93% . Ωστόσο υπάρχουν κάποια μειονεκτήματα σε αυτή τη μέθοδο:

• Αυτή η μέθοδος είναι μακράν η πιο χρονοβόρα μέθοδος που έχει δοκιμαστεί σε αυτή τη διπλωματική. Μια προσομοίωση με όλους τους πελάτες (3273) που είχαμε στη κατοχή μας, μπορούσε να διαρκέσει έως και 10 λεπτά ακόμη και με τη χρήση της principal component analysis, η οποία μειώνει δραστικά τον χρόνο μιας πλήρης εκτέλεσης (δεν δοκιμάστηκε χωρίς τη χρήση της pca καθώς απαιτούσε ακόμα περισσότερο χρόνο).
• Τα τυχαία βάρη για το νευρωνικό δίκτυο που επιλέγονται τυχαία από το pattern recognition app της Matlab, με το οποίο υλοποιήσαμε το νευρωνικό μας δίκτυο, οδηγούν μερικές φορές σε μέτριες αποδόσεις (σπάνια είχαμε μερικές προσομοιώσεις με ακρίβεια 74-80% λόγω αυτής της ενέργειας)

Φυσικά, αυτή η μέθοδος εκμεταλλεύεται το λογισμικό της Matlab από άποψη τεχνικών απεικόνισης (visualization), ευκολίας χρήσης και ευελιξίας. Συνολικά, όσον αφορά την απόδοση αυτή η μέθοδος ήταν η τρίτη καλύτερη που δοκιμάστηκε σε αυτή τη διπλωματική και αποτελεί μια πολύ καλή μέθοδο ανίχνευσης κλοπής ηλεκτρισμού.

Support Vector Machine Classification

Στην μέθοδο αυτή, χρησιμοποιήσαμε support vector machines classifier για την ανίχνευση των ανωμαλιών στις μετρήσεις των παράνομων καταναλωτών. Υλοποιήθηκε στο λογισμικό της Matlab καθώς προτιμήθηκε από την LibSVM λόγω ευκολίας χρήσης της Matlab. Οι support vector machines classifiers μπορούν να παράγουν ακριβή και εύρωστα αποτελέσματα classification σε μια σωστή θεωρητική βάση, ακόμη και όταν τα δεδομένα εισόδου είναι μη μονότονα και γραμμικά μη διαχωρίσιμα. Έτσι μπορούν να βοηθήσουν στην αξιολόγηση πιο σχετικών πληροφοριών με έναν βολικό τρόπο. Μάλιστα, η ακρίβεια των αποτελεσμάτων δεν βασίζεται στην ποιότητα της κρίσης της εμπειρογνωμοσύνης του ανθρώπου για τη βέλτιστη επιλογή της συνάρτησης γραμμικοποίησης των μη γραμμικών δεδομένων εισόδου και έτσι θεωρούνται χρήσιμο εργαλείο για την αποτελεσματική συμπλήρωση των πληροφοριών που αποκτώνται από τις κλασικές τεχνικές γραμμικού classification.

Το SVM και τα Τεχνητά Νευρωνικά Δίκτυα είναι δύο δημοφιλείς στρατηγικές για την εποπτευόμενη μηχανική μάθηση και classification. Οι SVM classifiers ,ως classifiers ευαίσθητος στο κόστος, μπορούν να λειτουργήσουν αποδοτικά ακόμα και σε μη ισορροπημένα δεδομένα. Όλα τα άλλα πλεονεκτήματα των SVM classifiers εξαρτώνται από τον τομέα και το είδος της εργασίας που επιζητείται. Οι SVM classifiers είναι καλοί σε σύγκριση με άλλους classsifiers, καθώς απαιτούν μικρότερη υπολογιστική πολυπλοκότητα ενώ παρουσιάζουν αυξημένη απόδοση στο classification σε σύγκριση με οποιονδήποτε άλλο μη γραμμικό classifier.

Στην περίπτωσή μας, όπως μπορείτε να δείτε από τις μετρήσεις απόδοσης στο πίνακα 0.1, η ακρίβεια κυμαινόταν από 79% έως 87% (ανάλογα με την περίπτωση που δοκιμάστηκε). Τα αποτελέσματα είναι αρκετά καλά αλλά δεν φτάνουν τα επίπεδα των δύο προηγούμενων μέθοδων που δοκιμάστηκαν. Συγκεκριμένα, στην 1η (υπερφόρτωσης 40% -60%) και στην 4η (μερικής κλοπής 30% -50%) περίπτωση, οι μετρήσεις απόδοσης μειώθηκαν λόγω του γεγονότος ότι στις περιπτώσεις αυτές τα δείγματα των καταναλωτών είναι παραπλήσια (δεν διαφέρουν αρκετά τα samples ενός νόμιμου καταναλωτή από αυτά ενός παράνομου καταναλωτή) μεταξύ τους και ο clasiffier δεν μπορούσε εύκολα να διακρίνει τον παράνομο από τον νόμιμο καταναλωτή. Εάν τα δεδομένα ήταν λιγότερο ισορροπημένα, τα αποτελέσματα θα ήταν πιθανώς καλύτερα, καθώς οι περισσότεροι classifiers 2-κλάσεων λειτουργούν καλύτερα με αυτόν τον τρόπο [52]. Ωστόσο, σε γενικά πλαίσια, τα αποτελέσματα αυτής της μεθόδου ήταν αρκετά ενθαρρυντικά για την ανίχνευση της ρευματοκλοπής στις περιπτώσεις που εξετάστηκαν.

Optimum Path Forest Classifiers με χρήση της LibOPF Library

Στην μέθοδο αυτή χρησιμοποιήσαμε optimum path forest classifier με τη χρήση της βιβλιοθήκης LibOPF σε τερματικό Linux, με σκοπό την ανίχνευση ρευματοκλοπής. Ωστόσο, όπως μπορείτε να δείτε και από τον πίνακα 0.1, τα αποτελέσματα αυτής της μεθόδου δεν ήταν ιδιαίτερα επιτυχή. Η ακρίβεια κυμαίνεται από 61,52% έως 80,91%, καθιστώντας αυτή τη μέθοδο όχι και τόσο αποδοτική. Αυτό το χάσμα στην ακρίβεια συμβαίνει πιθανώς επειδή οι OPF classifiers είναι ευαίσθητοι στο θόρυβο και σε μεγάλες αποκλίσεις των τιμών. Τα πρωτότυπα επιλέγονται με βάση το Minimum Spanning Tree (MST) και ως αποτέλεσμα επιλέγουν θορυβώδη δείγματα ή υπερβολικές υψηλές-χαμηλές τιμές για να γίνουν πρωτότυπα. Αυτά τα ‘κακά’ δείγματα έχουν μεγάλη επίδραση στις αποφάσεις του OPF classification.

Προκειμένου να κάνουμε τη μέθοδο αυτή να αποφέρει καλύτερα αποτελέσματα προβήκαμε σε αρκετούς πειραματισμούς. Προσπαθήσαμε να διαχωρίσουμε τα δεδομένα σε δεδομένα εκπαίδευσης (training data), δεδομένα επαλήθευσης (validation data), δεδομένα ελέγχου (test data) χειροκίνητα αλλά και με το προεπιλεγμένο πρόγραμμα της βιβλιοθήκης, προσπαθήσαμε να αλλάξουμε τα ποσοστά διαίρεσης των σετ δεδομένων που αναφέρθηκαν, πειραματιστήκαμε με διαφορετικές παραμέτρους, χρησιμοποιήσαμε διάφορα εναλλακτικά εργαλεία που προσέφερε η βιβλιοθήκη, χρησιμοποιήσαμε διαφορετικά features ως είσοδο αλλά πάντα καταλήγαμε σε παρόμοια αποτελέσματα.

Επιπρόσθετα, η μορφή αρχείου που η βιβλιοθήκη απαιτούσε ως είσοδο για να δουλέψει ήταν αρκετά δύσκολο να δημιουργηθεί από το αρχικό σύνολο δεδομένων που είχαμε στην κατοχή μας. Όλες οι πληροφορίες μεταξύ κάθε φάσης του classification αποθηκεύονταν σε αρχεία .dat και ως αποτέλεσμα δεν μπορούσαμε να εξαγάγουμε οτιδήποτε θέλαμε. Τέλος, τα εργαλεία-προγράμματα που χρησιμοποιεί η βιβλιοθήκη LibOPF για το classification είναι «κλειδωμένα» σε αρχεία και δεν μπορούσαμε να παρέμβουμε στον κώδικα τους. Αυτός είναι ο λόγος που δεν μπορούσαμε να υπολογίσουμε άλλες μετρήσεις επιδόσεις πέρα από την ακρίβεια που η βιβλιοθήκη υπολογίζει από μόνη της με το πρόγραμμα «opf_accuracy».

Συνολικά, αυτή η μέθοδος δεν ανταποκρίθηκε στις προσδοκίες που είχαμε όταν διαβάζαμε για αυτή στην έρευνά μας σχετικά με τις τεχνικές ανίχνευσης κλοπής ηλεκτρικής ενέργειας πριν ξεκινήσουμε αυτή τη διπλωματική. Τα αποτελέσματα μπορούν να χαρακτηριστούν μη ικανοποιητικά, παρόλο που όπως αποδείχτηκε είναι η ταχύτερη μέθοδος, σε σύγκριση με τις υπόλοιπες που δοκιμάστηκαν. Φυσικά, υπάρχει η πιθανότητα να μη υλοποιήθηκε σωστά από μέρους μας ή αυτή η μέθοδος να μην είναι τόσο κατάλληλη όσο οι άλλες για το είδος του προβλήματός μας ή για τα δεδομένα ηλεκτρικής ενέργειας που είχαμε. Σίγουρα αφιερώσαμε λιγότερο χρόνο σε αυτή τη μέθοδο από τις άλλες, αλλά αυτό οφείλεται στο γεγονός ότι τα αποτελέσματα δεν ήταν καθόλου ικανοποιητικά από την αρχή. Συμπερασματικά, μετά από όλες αυτές τις μεθόδους που έχουμε εξετάσει σε αυτή τη διπλωματική αυτή η μέθοδος σίγουρα δεν συνιστάται, τουλάχιστον από εμάς, για ανίχνευση της κλοπής ηλεκτρικής ενέργειας.

Neuro-Fuzzy System με χρήση του Neuro-Fuzzy Designer App

Αυτή η μέθοδος όπως μπορείτε να δείτε και από το πίνακα 0.1 απέδωσε εξαιρετικά αποτελέσματα. Η ακρίβεια κυμαινόταν από 92% ως 99% ανάλογα με τη περίπτωση κλοπής που δοκιμάστηκε. Ο Neuro-Fuzzy Designer της Matlab, που χρησιμοποιήθηκε για τη δημιουργία του μοντέλου FIS , ήταν εύκολος στη χρήση και αρκετά ευέλικτος (μπορούσαμε να προσαρμόσουμε οποιοδήποτε παράμετρο αρκετά εύκολα). Από όλες τις μεθόδους που εξετάστηκαν σε αυτή τη διπλωματική αυτή η μέθοδος ήταν η καλύτερη όσον αφορά την απόδοση. Σίγουρα δεν ήταν η πιο γρήγορη μέθοδος, καθώς η εκπαίδευση του μοντέλου FIS απαιτούσε αρκετό χρόνο αλλά τα αποτελέσματα της μεθόδου αυτής μας αντάμειψαν. Όπως αποδείχθηκε, ο συνδυασμός multilayer neural networks με fuzzy logic που υλοποιείται μέσω του FIS μοντέλου είναι από τις καλύτερες μεθόδους για τον εντοπισμό ρευματοκλοπής.

Παρακάτω βλέπουμε ένα πίνακα με τις επιδόσεις των μεθόδων που υλοποιήθηκαν σε αυτή τη διπλωματική σε κάθε διαφορετική περίπτωση που εξετάστηκαν.

Methods	Overload 40-60%	Overload 60-80%	Overload 80-100%	Partial Theft 30-50%	Partial Theft 50-70%	Partial Theft 50-70% & Overload 60-80%
FANN	89%	93%	95%	92%	97%	95%
Deep Learning Toolbox	86%	87,8%	89,2%	84,8%	92,6%	91,8%
SVM	80%	83%	85%	79%	87%	85%
LibOPF	61,52%	64,75%	69,15%	71,46%	80,91%	75,44%
Neuro-Fuzzy System	93%	97%	99%	96%	99%	92%

Table 0.1 *Απόδοση όλων των μεθόδων για κάθε περίπτωση που εξετάστηκε*

Εν κατακλείδι, η εργασία μας έδειξε ενθαρρυντικά αποτελέσματα για την ανίχνευση ρευματοκλοπής στο ηλεκτρικό δίκτυο. Η δουλειά μας βασίστηκε σε πραγματικά δεδομένα κατανάλωσης ενέργειας από ένα ηλεκτρικό δίκτυο στην Ιρλανδία. Από όλες τις μεθόδους, ξεχώρισαν, όσο αφορά την απόδοση, η μέθοδος με multilayer artificial neural networks με χρήση της FANN βιβλιοθήκης καθώς και η μέθοδος με Neuro-Fuzzy System. Αποδοτική, αν και αρκετά χρονοβόρα, ήταν επίσης η μέθοδος με neural networks με τη χρήση του Deep Learning Toolbox ενώ και η μέθοδος με SVM classifiers ήταν αξιόλογη. Αντιθέτως, η μέθοδος με Optimum Path Forest classifiers δεν απέδωσε αυτό που περιμέναμε όταν διαβάζαμε για αυτή στην ερευνά μας και δεν συνίσταται, συγκριτικά με τις άλλες.

Συνολικά, είμαστε πολύ ευχαριστημένοι με τα αποτελέσματα που επιτεύχθηκαν κατά τη διάρκεια αυτής της διπλωματικής και σίγουρα ανταποκρίθηκε τις προσδοκίες μας.

Contents

List of Abbreviations

AMI	Advanced Metering Infrastructure
AOC	Average Original Consumption
ADC	Average Daily Consumption
AMC	Average Monthly Consumtion
ATC	Average Total Consumption
ANNs	Artificial Neural Networks
CPC	Calculated Probable Consumption
CNNs	Convolutional Neural Networks
DE	Difference Expansion
DG	Distributed Generation
EDC	Electric Distribution Companies
FANN	Fast Artificial Neural Network
FNN	Fuzzy Neural Network
GPU	Graphic Processing Units
GIS	Geographic Information System
GNA	Gauss-Newton algorithm
HPC	High Performance Computers
IFT	Image Foresting Transform
LTSM	Long Short-Term Memory
LM	Levenberg-Marquardt
MST	Minimum Spanning Tree
NN	Neural Network
NNPR	Neural Network Pattern Recognition
NFS	Neuro-Fuzzy System
NTL	Non-Technical Losses
OPF	Optimum Path Forest
PCA	Principal Component Analysis
RTP	Real Time Pricing
SVM	Support Vector Machines
T&D	Transmission and Distribution
TBP	Time Base Pricing

1. Introduction

1.1 Problem Definition

The Generation, Transmission and Distribution (T&D) of electricity, involve many operational losses. These operational losses are divided in two categories, technical losses that cannot be avoided and non-technical losses due to electricity theft from illegal activities of consumers. The magnitude of these losses is rising at an alarming rate in several countries. With a view to enhance the economy of the utilities, as well as the efficiency and the security of the grid, we propose some methods of analyzing electricity consumption patterns of consumers and identifying illegal behaviours.

1.2 Dissertation Objectives and Scope

The power losses are a common reality for the electric utilities and can be classified as technical or non-technical losses. Losses that occur during generation can be technically defined, but Transmission and Distribution (T&D) losses cannot be quantified completely. The present transmission and distribution losses in several countries have been reported to be over 30% [11]. Substantial quantity of losses proves the involvement of Non-Technical Losses (NTL) in distribution networks. Total losses during T&D can be evaluated from the information like total load and the total energy billed, using established standards. NTL are caused mainly by factors external to the power system. Electricity theft constitutes a major chunk of the NTL. It is estimated that utilities worldwide lose more than $25 Billion every year due to illegal consumption of electricity [53]. It has also been identified that the illegal consumption of electricity by local business sectors is increasing. Total losses incurred by utilities due to electricity theft are huge. As the impact of these losses is huge, it is essential to force the implementation of a mechanism that reduces NTL and identifies illegal consumers.

Detection of illegal consumers is an extremely challenging problem nowadays due to the large amount of money that are not imputable to the state. This dissertation presents some methods that use energy consumption patterns to detect illegal consumers in a smart grid environment. To realize this solution, initially, this dissertation conducts an extensive survey on the methods implemented in pilfering electricity and technologies involved in smart energy meters.

In general, utilities collect real-time energy consumption information from their consumers several times a day. We had the chance to work with some real energy consumption data from an electrical grid in Ireland [50]. After some energy consumption data processing, we apply specific classification methods which are implemented to identify illegal and legal consumers and could lead to a motivated and strategic progression towards achieving objectives and features of a smart grid.

1.3 Literature Review

From our research on electricity theft we have found several techniques that were proposed to control illegal consumption or identify illegal consumers of electricity in the recent past. We present some of these methods :

[1] Bandim et al. (2003). Proposed utilization of a central observer meter at secondary terminals of distribution transformer. Value of energy read by the central observer meter is compared with the sum of energy consumption values read by all energy meters in range. These two values of current are compared to estimate the total quantity of electricity that is being consumed illegally.

[2] Anand and Naveen (2003). Vigilant energy metering system (VEMS) is an advanced energy metering system that can fight electricity theft. It provides the data acquisition, transfer and data processing capabilities among the energy meters, local area stations and the base. It also facilitates load forecasting and control, identifies potential areas of theft, losses and takes measures to rectify it.

[3] Nagi et al. (2009). Proposed a novel approach of using genetic algorithm-support vector machines (GA-SVM) in detecting electricity theft. Load consumption data of all customers is collected and data mining techniques are used to filter and group these customers based on their consumption patterns. Customers are grouped into different classes based on the extent of the abnormality in load profile and then the customers with high probability of theft are inspected.

[4] Nizar and Dong (2009). The main objective here is to base the investigation on comparing the efficacy of the Support Vector Machine (SVM) technique with the newly emerging techniques of Extreme Learning Machine (ELM) and its OS-ELM variant as means of classification and prediction in this context. Non-technical Losses (NTL) represent a significant proportion of electricity losses in both developing and developed countries. The ELM-based approach presented here uses customer load-profile information to expose abnormal behaviour that is known to be highly correlated with NTL activities. This approach provides a method of data mining for this purpose and it involves extracting patterns of customer behaviour from historical kWh consumption data. The results yield classification classes that are used to reveal whether any significant behaviour that emerges is due to irregularities in consumption. In this paper, ELM and online sequential-ELM (OS-ELM) algorithms are both used to achieve an improved classification performance and to increase accuracy of results. A comparison of this approach with other classification techniques, such as the Support Vector Machine (SVM) algorithm, is also undertaken and the ELM performance and accuracy in NTL analysis is shown to be superior.

[5] Pasdar and Mirzakuchaki (2007). Power line communications (PLC) presents an interesting and economical solution for automatic meter reading (AMR). If an AMR system via PLC is set in a power delivery system, a detection system for illegal electricity usage may be easily added in the existing PLC network. In the detection system, the second digitally energy meter chip is used and the value of energy is stored. The recorded energy is compared with the value at the main kilowatthour meter. In the case of

a difference between two recorded energy data, an error signal is generated and transmitted via PLC network. This paper describes a prototype of the detector system for illegal electricity usage using the power lines. The target of this study is to discover new and possible solutions for the lack of literature concerning this problem.

[6] Zeng et al. (2009) proposed a new method to monitor the street lamp power cables, as well as the theft of electric transmission cables. In this method, resonant frequency and equivalent capacitance of the cable are calculated by sending a frequency varying current signal to the street lamp power system. These values are used to detect the theft of electric transmission cables and then the location where the event happened.

[7] Jamil et al. (2004) proposed a microcontroller based energy meter which facilitates the utility company to monitor and control the power supply to its spatially distributed consumers. This meter acts as a check meter to detect the meter tampering. However, e-metering systems can collect and process data, and can detect abnormalities in load profiles indicating electricity theft (De et al., 2003)

[8] Saptarshi De, Rabul Anand and Sirat Moinuddin et al (2004) Proposes the e-metering as an efficient method of power measurement. The e-metering data possess unique characteristics and the network needs innovative technology to record, monitor and process the data. The data are processed to collect information of customers, power usage profile of an area and of a consumer, supply outage time and losses incurred in distribution system The system is capable of measuring and estimating the quality of power supplied to the consumers. It consists of data acquisition, transmission and processing components among the energy meters, local area stations and base stations. The application of the e-metering system is extended to streamline power distribution with online monitoring of power quality, real time theft detection and automatic billing.

[9] Mano et al. (2004) Suggest proper design and implementation of rules for investigation of illegal consumers. Revenue Assurance and Audit Process is targeted at improving revenues for utility companies by reducing commercial losses by about 20% each year.

[10] Perez et al. (2005). Suggest that proper implementation of strategies in deployment and maintenance of the distribution networks can control commercial losses. In addition, strengthening the transformers at substations with higher configuration ones is also suggested. These suggestions can also be implemented in other African countries with the similar situations in order to improve the efficiency in transmission and distribution system.

[11] Soma Shekara Sreenadh Reddy Depuru (2010). This paper proposes an architectural design of smart meter, external control station, harmonic generator, and filter circuit. Motivation of this work is to deject illegal consumers, and conserve and effectively utilize energy. If a considerable amount of NTL is detected, harmonic generator is operated at that feeder for introducing additional harmonic component for destroying appliances of the illegal consumers.

[12] Solomon Nunoo et al. (2011). The method includes receiving meter data of the measured power consumed by a customer, receiving delivered power data that includes data of the power delivered to the customer, determining a difference between the meter data and the delivered power data, determining that the difference between the meter data and the delivered power data is greater than a predetermined amount, and indicating a discrepancy if the difference between the meter data and the delivered power data is greater than a predetermined amount.

[13] Rong Jiang et al (2014). They discuss the background of Advanced Metering Infrastructure (AMI) and identity major security requirements that AMI should meet. Specifically, an attack tree based threat model is first presented to illustrate the energy-theft behaviors in AMI. Then, they summarize the current AMI energy-theft detection schemes into three categories, i.e., classification-based, state estimation-based and game theory-based ones, and make extensive comparisons and discussions on them.

[14] Soma Shekara Sreenadh Reddy Depuru (2012). This paper designs and implements an encoding procedure to simplify and modify customer energy consumption data for quicker analysis without compromising the quality or uniqueness of the data. This paper parallelizes overall customer classification process. The parallelized algorithms have resulted in appreciable results as displayed in the results section of the paper.

[15] P.kadurek et al.(2010). This paper provides insight into the illegal use or abstraction of electricity in the Netherlands. The importance and the economic aspects of theft detection are presented and the current practices and experiences are discussed. The paper also proposes a novel methodology for automated detection of illegal utilization of electricity in the future distribution networks equipped with smart metering infrastructure. The necessary data requirements for smart meters and distribution substations are defined, in order to unlock this feature in distribution network. The paper also proposes the measures, which should be undertaken by the smart metering standards.

[16] Shih-Che Huang et al (2013). In this paper, a state estimation based approach for distribution transformer load estimation is exploited to detect meter malfunction-tampering and provide quantitative evidence of non-technical loss (NTL). A measure of overall fit of the estimated values to the pseudo feeder bus injection measurements based on customer metering data aggregated at the distribution transformers is used to localize the electricity usage irregularity.

[17] Robert Czechowski and Anna Magdalena Kosek et al (2017). The paper presents not only the factors encouraging energy consumers to engage in dishonest behavior and the techniques they use to achieve the intended result, but also technical measures aimed at detecting such actions. The discussed technical issues proved useful in designing increasingly refined security measures and ways to detect electricity theft

[18] Caio C. O. Ramos , Andre N. Souza, Joao P. Papa, Alexandre X. Falcao et al (2010). This article proposes an innovative and accurate solution for non-technical losses identification using the

Optimum-Path Forest (OPF) classifier and its learning algorithm. Results in two datasets demonstrated that OPF outperformed the state of the art pattern recognition techniques and OPF with learning achieved better results for automatic nontechnical losses identification than recently ones obtained in the literature..

[19] Breno C. Costa, Bruno. L. A. Alberto, André M. Portela, W. Maduro, Esdras O. Eler et al (2013). This paper proposes the use of the knowledge-discovery in databases based on artificial neural networks applied to the classifying process of consumers to be inspected. An experiment performed in a Brazilian electric power distribution company indicated an improvement of over 50% of the proposed approach if compared to the previous methods used by that company.

[20] Patrick Glauner et al (2019). This thesis compares expert knowledge-based decision making systems to automated statistical decision making. It proposes a method for visualizing prediction results at various granularity levels in a spatial hologram. This approach allows domain experts to put the classification results into the context of the data and to incorporate their knowledge for making the final decisions of which customers to inspect. Moreover, it presents a machine learning framework that classifies customers into NTL or non-NTL using a variety of features derived from the customers' consumption data as well as a selection of master data. The methodology used is specifically tailored to the level of noise in the data.

[21] Awais Khan and Wei Xie et al (2018). This research project modeled and designed the hardware prototype of automated anti-theft electricity distribution system. The proposed system detects the illegal load and burns it by sending high voltage signals from a capacitor bank. The legal load is made safe and uninterrupted during execution of illegal load.

[22] Ali Akbar Ghasemi and Mohsen Gitizadeh et al (2018). In this paper, a combined method is proposed to detect both types of illegal consumptions. Customer energy consumption pattern classification method based on probabilistic neural network and mathematical model based on Levenberg-Marquardt method are used to detect the first and second type of illegal consumption, respectively. Moreover, the impact of Distributed Generation (DG) sources on illegal consumption of electricity is analyzed and proposed detection algorithm is modified to compensate it. Experimental results are presented to show the effectiveness of this method in detection of both types of illegal consumption.

[23] Jawad Nagi, Keem Siah Yap, Sieh Kiong Tiong, Syed Khaleel Ahmed and Malik Mohamad et al (2010). This paper presents a new approach towards nontechnical loss (NTL) detection in power utilities using an artificial intelligence based technique, support vector machine (SVM). The fraud detection model (FDM) developed in this research study preselects suspected customers to be inspected onsite fraud based on irregularities in consumption behavior. This approach provides a method of data mining, which involves feature extraction from historical customer consumption data. This SVM based approach uses customer load profile information and additional attributes to expose abnormal behavior that is known to be highly correlated with NTL activities. The result yields customer classes which are

used to shortlist potential suspects for onsite inspection based on significant behavior that emerges due to fraud activities.

[24] Konstantinos Blazakis and Georgios Stavrakakis et al (2019). This article presents a computational method of analyzing and identifying electricity consumption patterns of consumers based on data mining techniques in order to identify illegal residential consumers. Combining principal component analysis (PCA) with mean shift algorithm for different power theft scenarios. The overall research has shown encouraging results in residential consumers power theft detection that will help utilities to improve the reliability, security and operation of power network.

1.4 Thesis Outline

Chapter 1: Discusses the problem definition and scope of dissertation and also provides a literature review in power theft detection techniques.

Chapter 2: Provides an insight to methods of stealing electricity, methodology for tamper detection but also the factors and consequences of electricity theft.

Chapter 3: Describes the energy consumption data we worked with in our thesis, the features that have been tested and some accuracy metrics.

Chapter 4: Outlines the first electricity theft detection method based on multilayer artificial neural networks (Fast Artificial Neural Network Library).

Chapter 5: Expounds the second electricity theft detection method based on neural networks (Deep Learning Toolbox).

Chapter 6: Introduces the third electricity theft detection method based on Support Vector Machines classification.

Chapter 7: Presents the final electricity theft detection method based on Optimum Path Forest classifiers (LibOPF library).

Chapter 8: Discusses the experimental results and further research for improving power theft detection techniques.

2. Methods, Factors and Consequences of Power Theft

Generally, electricity consumers may be classified as genuine-legal consumers, partial illegal consumers, and completely illegal consumers. This chapter refers to technical and non-technical losses in a power grid, presents several simple and sophisticated methods used in pilfering electricity, discusses factors that influence illegal consumers to steal electricity and discusses consequences of electricity theft.

2.1 Technical and Non Technical Losses

Generation, Transmission, and Distribution of electricity involve many losses. The losses can technically be classified as direct losses and indirect losses (technical and non-technical losses).

Technical losses occur naturally due to power dissipation in transmission lines, power transformers and measurement systems . These losses are normally computed based on the information about the load on the grid and the energy billed to the consumer. The most common examples of technical losses include the power dissipated in transmission lines and transformers due to their internal electrical resistance. Technical losses are possible to compute and control, provided by the power system infrastructure. Computation tools for calculating power flow, losses, and equipment status in power systems have been developed for some time. Improvements in information technology and data acquisition have also made the calculation and verification of technical losses easier. These losses are calculated based on the natural properties of components in the power system, which include resistance, reactance, capacity, voltage, and current. Loads are not included in technical losses because they are actually intended to receive as much energy as possible. Two major sources contribute to technical losses: load losses which consist of the ohmic losses I^2R and impedance losses I^2X of the various system elements, and no-load losses which are independent of the actual load served by the power system. The majority of the no-load losses are due to transformer core losses resulting from excitation current flows. The technical losses are generally computed at 4–12% depending on various factors including delivery systems, networks and the technology that is used. There is a need to manage and reduce these losses through efficient management and operation [11],[54],[57].

Non technical Losses (NTLs) refer to losses that occur independently of technical losses in power systems. NTLs are caused by actions external to the power system and also by the loads and conditions that technical losses computations fail to take into account. Non-technical losses account for about 4–40%, depending on their generating capacity and geographical area. However, computing non-technical losses is complicated and can only be estimated. Non-technical losses are related to energy theft, component breakdowns , faulty meters or metering components, billing cycles on the customer, etc. Of all the non-technical losses, electricity theft takes the major chunk and includes illegal tapping, bypassing and tampering meters and several other physical methods including defaulting and evading payment to utility companies. No power system is 100% safe and secure from theft. Energy theft, especially electricity is a common problem and is predominant across the globe. It is fundamentally due to these inherent problems, electricity theft, cannot be computed precisely but can only be quantified as an estimate [54].

2.2 Methods of Stealing Electricity

The power distribution monitoring is an important factor in electric power systems and electricity theft defence is one of the chief steps in distribution network reconstruction. Even though the power supply departments have made huge investments of manpower and material, the phenomenon of electricity theft has increased and not abated, and the methods of stealing electricity is continuously improving.

The calculation of electricity quantity measured by single-phase electric energy meters is mainly according to the relationship with voltage, electric current and power-factor angle. If any factor is changed, the electric energy meter turns slowly, stalls and will even cause a reversal, so the purpose of electricity stealing is attained. According to the investigation analysis, there have been dozens of electricity-stealing tricks tampering single-phase electric energy meter. The methods could be approximately divided into under voltage, undercurrent, phase-shifted and difference expansion (DE) to their principles [57],[58],[25].

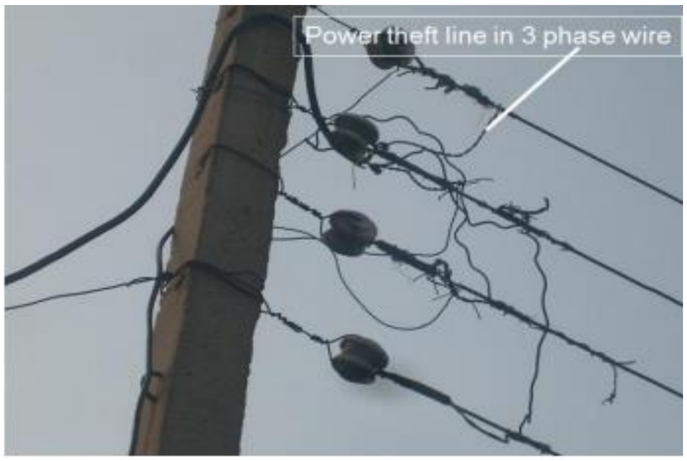


Figure 2.1 *Method of stealing electricity.*

A. Stealing electricity by under voltage technology

Electricity thieves adopt all kinds of technology to deliberately change the wire splice of voltage circuit which is measured by electrical energy, or cause a malfunction in measuring a voltage circuit and voltage curve of pressure loss, thus measuring less electric power [57]. Here are some common tricks:

1) Unhooking technology of electricity-stealing. Secretly destroy the lead sealing of electric energy meter, open voltage hook of terminal in junction box, and make no electric current through, all using quantity of electricity steal.

2) Loose zero curve technology. Open input zero curve of meter, ground output zero curve of meter.

3) One fire one ground technology. Take the ground wire as naught line, generally take the water pipe or caliduct as ground wire, the risk is bigger.

4) Violated wire connection.

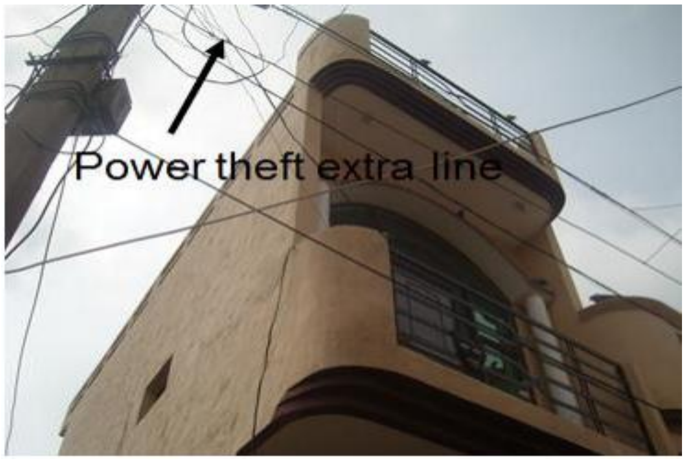


Figure 2.2 *Method of stealing electricity.*

<u>B. Stealing Electricity by Undercurrent Technology</u>

Electricity thieves adopt all kinds of technology to deliberately change the wire splice of voltage circuit which is measured by electrical energy, or cause a malfunction in measuring a voltage circuit and voltage curve of pressure loss, thus measuring less electric power [57]. Here are some common tricks

1) Loop of short electric current, which makes the electric energy meter shift slow.

2) Bypassing meter with a wire in order to reel across electrical energy meter, which allows no or little electric current through.

3) Exchange fire wire and zero wire.

Figure 2.3 *Method of stealing electricity.*

<u>C. Stealing Electricity by Phase-Shifted Technology</u>

Electricity thieves adopt all kinds of technology to deliberately change normal wire connection of electric energy meter, take use of specifically connecting methods of inductance or capacitance to change the normal phase relationship of voltage and current in the loop of electric energy meter, thus making the electric energy meter shift slow and even causing reversal [57]. Here are some common tricks:

1) Reverse the in and out fire wire. Make electric current in the current coil reverse, and the electric energy meter reverse.
2) Make the electric meter reverse by using external power supply. Adopt hand generator with voltage and current output or inverter power supply to join into the electric meter, make the electric energy meter reverse rapidly.

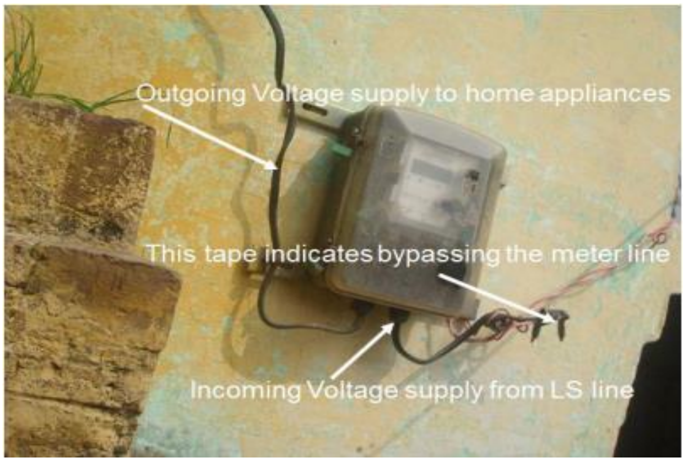

Outgoing Voltage supply to home appliances
This tape indicates bypassing the meter line
Incoming Voltage supply from LS line

Figure 2.4 *Method of stealing electricity.*

D. Stealing Electricity by Difference Expansion (DE) Technology
Electricity thieves disconnect electric energy meter privately, adopt all kinds of technology to change inner structural performance of electric energy meter to cause itself error increase. They make use of current or mechanical force to destroy electric energy meter, and change its installation conditions in order to make less electric energy records.

Besides the four electricity-stealing methods listed above , the worst is that the user illegally and improperly draws or connects a wire in order to steal electricity without a reporting meter. This stealing method with no meter always causes injuries, fire accidents and so on. These people should be brought to justice and punished strictly [57].

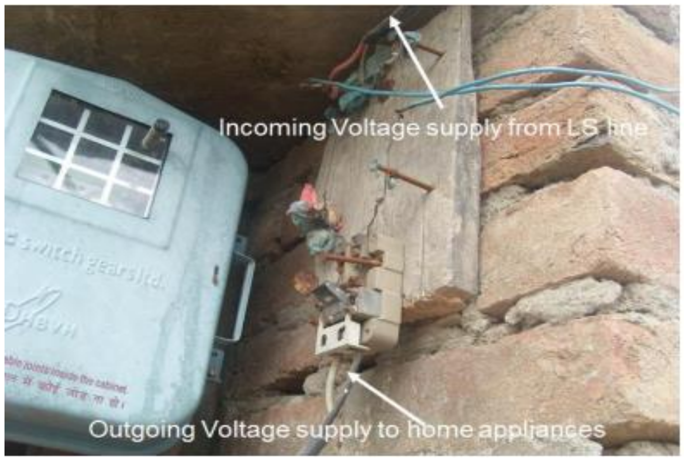

Incoming Voltage supply from LS line
Outgoing Voltage supply to home appliances

Figure 2.5 *Method of stealing electricity.*

2.3 Methodology for Tamper Detection

A system for detection of tampering with a utility meter provides not only an indication that tampering has occurred but also sufficient information to enable an estimation of actual consumption to be made as opposed to the tampered meter consumption.

Neutral Tampering
This tampering is done by disconnecting the neutral wire in the domestic electricity meter. With the Neutral disconnected, there is no voltage input and thus no output would be generated by the power supply. The component used for the neutral tampering detection circuit is an Optocoupler connected in parallel to the Voltage coil. When the neutral wire from the electricity meter is disconnected, the power supply to the LED is also disconnected and hence it does not glow.

Solution: As long as the neutral and the phase are connected to the voltage coil, the parallely connected Optocoupler LED produces a high output thereby biasing the base of the photo transistor which produces an output at the emitter. As soon as the LED fails to glow, the photodiode is unbiased and its output drops to low level or logic “0” signal and it is sent to the micro controller [58].

Magnetic Tampering
Magnetic tampering technique is done by bringing a high powered magnet in close proximity of the domestic electricity meter. When this happens, the rotor disc is exposed to a high magnetic field. Then the resultant opposing magnetic field to the rotor is highly increased leading to slowing down of rotor or perfect stopping of the disc rotation. The idea is to saturate the core of the sensors or distort the flux in the core so that output is erroneous. This effectively results in less billing.

Solution: When a high power magnet is brought in proximity of the electricity meter in the presence of the Reed switch two leads come in contact with each other and result in the closed circuit thereby helping in detection of the signal by the micro controller. Thus the tampering is detected effectively and the microcontroller receives the information, which tis passed on to the LCD unit and the GSM modem [58].

Reverse Current
Reverse Current occurs when the phase and neutral are wired to the wrong inputs, causing current to flow in the direction opposite to normal. Figure shows the Neutral Wire connection is swapped thus causing current IN to flow in the reverse direction. Due to the reverse current flow through Neutral, metering firmware will show wrong signs in active power readings. The polarity of current transformer (CT) changes when any of the two currents has an opposite sign to the one expected.

Solution: To overcome this, metering firmware always uses the absolute value of active power for driving the energy pulse, thus reverse current has no effect on energy calculation or accurate billing [58].

Partial Earth Fault Condition

Partial earth fault means some of the load has been connected to another ground potential and not the neutral wire. In normal condition current going through the Phase wire is the same as coming out of the neutral wire (IP = IN). In case of partial Earth Fault Condition, the current in the neutral wire IN is less than that in the Phase or live wire IP.

Solution: To detect this condition, firmware monitors the currents on both energy wires - Phase and Neutral, and compares them. If they differ significantly, the firmware uses the larger of the two currents to determine the amount of energy to be billed and signals a "fault" condition [58].

2.4 Consequences of Power Theft

In daily basis operation utilities attempt to maintain a good power factor, flat voltage profile and sufficient reactive power along the feeders. These operations may become difficult to perform due to dynamic and inadequate load flow information. Although, illegal consumption of electricity might affect the performance of appliances connected to grid.

Primarily, electricity theft affects the utility company and then its customers. Electricity theft places severe impacts on utility companies and the consumers in general. It adversely affects the quality of power supply, overloads the generation units thereby resulting in overall losses as the utility companies cannot explicitly estimate its net supply to the consumer be it legal or illegal, forcing utility companies to buy more electricity from the market thereby increasing the load on the generators to produce more. Though the generation unit keeps tracks of the amount of power generated at any given instance, it makes it difficult for the production units to keep track of the exact amount of energy transmitted to the grid to the utility companies. The increase of power overload may also result in high voltage or low voltage affecting the performances and even result in damages to appliances of the consumers. This erratic and unpredictable additional loads will lead to blackouts and burnouts during peak load periods In addition to erratic power distribution to the consumers, theft affects economies of the utility companies adversely. Utility companies earn their money as a return for the service provided. About 10–40% of revenue is lost due to NTL. These losses need to be compensated, if the salaries of employees, overhead costs, and maintenance costs need to be met, forcing the electricity companies to increase the tariff on the consumer, thus impacting the real consumers directly. Theft also raises many safety concerns like electric feeder shocks resulting in the death of a person tapping the electricity illegally, the risk of electrical fire which may cause a major disaster. Improper handling of the distribution feeder might pose danger to the whole community, as these wires might start sparking and may cause fire during extreme weather conditions [30],[55],[28].

2.5 Factors That Influence Illegal Consumers

The factors that influence consumers to steal electricity depend upon various local parameters that fall into multiple categories like social, political, financial, literacy, law, managerial, infrastructural, and economical. Of these factors, socio-economic ones influence people to a greater extent in stealing electricity [25],[28].

In essence, electricity theft is proportional to the socio-economic conditions of the consumer. The most important factors are:

- Higher energy prices, unemployment or weak economic situation of the consumers.
- The belief that it is dishonest to steal something from a neighbor but not from a utility (public or large entity which have a lot of money).
- Some consumers might not be literate about the issues, laws and offenses related to the energy theft.
- Corrupt politicians and employees of the utilities are responsible for billing irregularities. In some cases, total money spent on bribing utility employees is less than the money that would have been paid for consuming the same amount of electricity legally.
- Reasons to hide total energy consumption (e.g. Consumers who grow marijuana illegally or small-scale industries to hide overall production).
- Tax Purposes. Different tax in an electrified house.
- Weak accountability and enforcement of law.

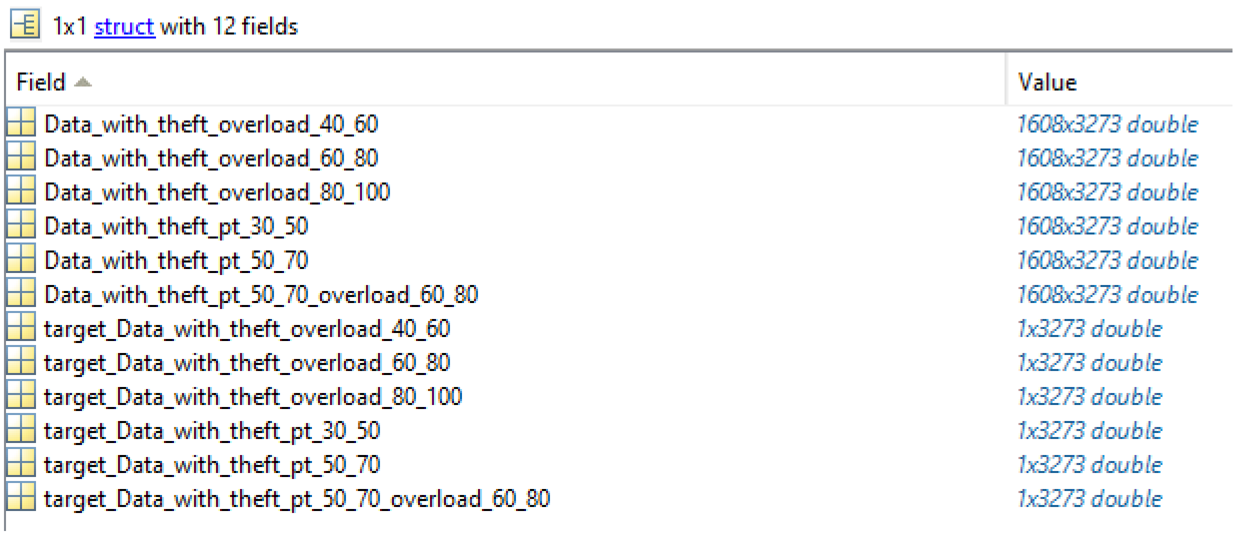
3. Energy Consumption Data Manipulation, Features and Metrics

With the rapid development of electrical grids, the power industries have to face the increasingly severe problem of electricity theft. Theft of electricity has negative effects on many socioeconomic aspects, including the impact on the economy for power enterprises and social development. The traditional anti-theft means require officers checking the integrity of kilowatt-hour meter and the correctness of wiring house by house, which requires enormous manpower and material resources. With the advent of modern information collecting technology, power enterprises now possess relatively complete database of power consumption. As a result, performing data mining on existing database and identifying abnormal users has become a hot topic in the field of information technology.

Nowadays, smart meters have been installed in over 60% of the most developed countries' households, Energy consumption data is more widely available than ever before and is delivering benefits to consumers, including more accurate billing and more reliable forecasts and alerts. Electric utilities are also using this granular data to improve operations in several ways.

3.1 Energy Consumption Data Description

For the implementation of this thesis we had the chance to obtain some real energy consumption data from an electricity consumer behavior trial by the Irish Commission for Energy Regulation (CER). This data is based on the real data from approximately 5000 Irish households monitored for one and a half years [50]. The work of this thesis is based on these data. The real energy consumption measurements we had in our possession was a struct of data (you can see in figure 3.1) with different kinds and percentages of electricity theft in each data table. These data had also the corresponding "target" data, which is data that define if the customer is fraudster or non fraudster.

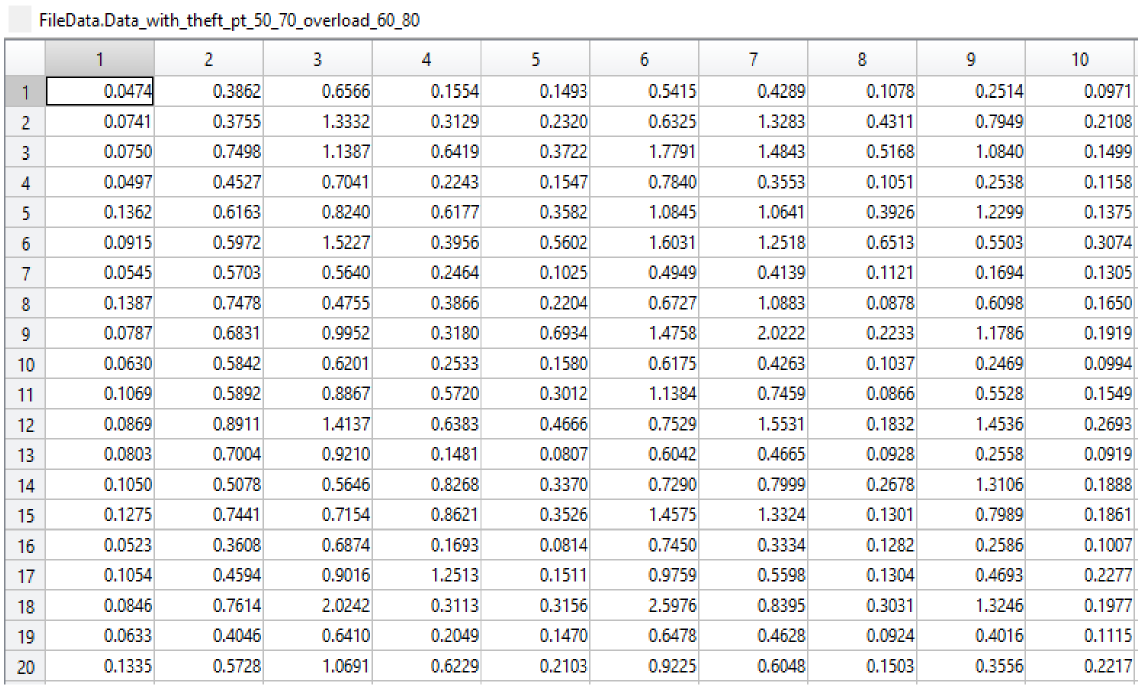


1x1 struct with 12 fields

Field	Value
Data_with_theft_overload_40_60	1608x3273 double
Data_with_theft_overload_60_80	1608x3273 double
Data_with_theft_overload_80_100	1608x3273 double
Data_with_theft_pt_30_50	1608x3273 double
Data_with_theft_pt_50_70	1608x3273 double
Data_with_theft_pt_50_70_overload_60_80	1608x3273 double
target_Data_with_theft_overload_40_60	1x3273 double
target_Data_with_theft_overload_60_80	1x3273 double
target_Data_with_theft_overload_80_100	1x3273 double
target_Data_with_theft_pt_30_50	1x3273 double
target_Data_with_theft_pt_50_70	1x3273 double
target_Data_with_theft_pt_50_70_overload_60_80	1x3273 double

Figure 3.1 *Energy Consumption Data.*

Every data table has 1608 rows and 3273 columns. Each column from these data tables is a different consumer and each row is an 8 hour measurement in kw/h for a specific time. The initial data had one measurement per 30 minutes for each consumer, but we averaged them with one measurement per 8 hours. Of course, this change didn't affect the performance of our methods. So, in total, we have 536 days of measurements (3 measurements per day, so 1608/3=536) and 3273 customers. You can see in figure 3.2 the form of the energy consumption data.


FileData.Data_with_theft_pt_50_70_overload_60_80

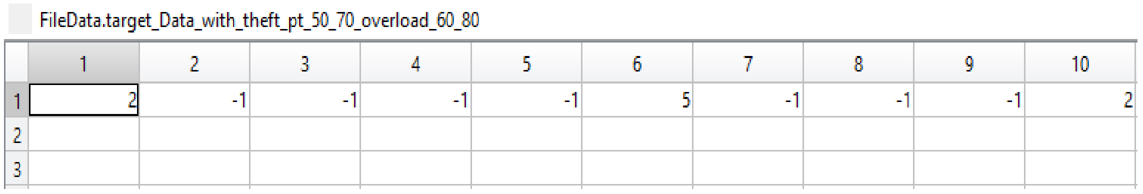
	1	2	3	4	5	6	7	8	9	10
1	0.0474	0.3862	0.6566	0.1554	0.1493	0.5415	0.4289	0.1078	0.2514	0.0971
2	0.0741	0.3755	1.3332	0.3129	0.2320	0.6325	1.3283	0.4311	0.7949	0.2108
3	0.0750	0.7498	1.1387	0.6419	0.3722	1.7791	1.4843	0.5168	1.0840	0.1499
4	0.0497	0.4527	0.7041	0.2243	0.1547	0.7840	0.3553	0.1051	0.2538	0.1158
5	0.1362	0.6163	0.8240	0.6177	0.3582	1.0845	1.0641	0.3926	1.2299	0.1375
6	0.0915	0.5972	1.5227	0.3956	0.5602	1.6031	1.2518	0.6513	0.5503	0.3074
7	0.0545	0.5703	0.5640	0.2464	0.1025	0.4949	0.4139	0.1121	0.1694	0.1305
8	0.1387	0.7478	0.4755	0.3866	0.2204	0.6727	1.0883	0.0878	0.6098	0.1650
9	0.0787	0.6831	0.9952	0.3180	0.6934	1.4758	2.0222	0.2233	1.1786	0.1919
10	0.0630	0.5842	0.6201	0.2533	0.1580	0.6175	0.4263	0.1037	0.2469	0.0994
11	0.1069	0.5892	0.8867	0.5720	0.3012	1.1384	0.7459	0.0866	0.5528	0.1549
12	0.0869	0.8911	1.4137	0.6383	0.4666	0.7529	1.5531	0.1832	1.4536	0.2693
13	0.0803	0.7004	0.9210	0.1481	0.0807	0.6042	0.4665	0.0928	0.2558	0.0919
14	0.1050	0.5078	0.5646	0.8268	0.3370	0.7290	0.7999	0.2678	1.3106	0.1888
15	0.1275	0.7441	0.7154	0.8621	0.3526	1.4575	1.3324	0.1301	0.7989	0.1861
16	0.0523	0.3608	0.6874	0.1693	0.0814	0.7450	0.3334	0.1282	0.2586	0.1007
17	0.1054	0.4594	0.9016	1.2513	0.1511	0.9759	0.5598	0.1304	0.4693	0.2277
18	0.0846	0.7614	2.0242	0.3113	0.3156	2.5976	0.8395	0.3031	1.3246	0.1977
19	0.0633	0.4046	0.6410	0.2049	0.1470	0.6478	0.4628	0.0924	0.4016	0.1115
20	0.1335	0.5728	1.0691	0.6229	0.2103	0.9225	0.6048	0.1503	0.3556	0.2217

Figure 3.2 *Measurements from the energy consumption data.*

Every target table has 1 row and 3273 columns. Like before, each column from these target tables is a different consumer. These target tables reveal the identity of each consumer, accordingly to the number that corresponds to him. If the number is:

- -1, then he is Genuine-Legal Consumer
- 2, then he is Illegal Consumer, Partial Theft
- 5, then he is Illegal Consumer, Overload

You can see a target table in the figure 3.3


FileData.target_Data_with_theft_pt_50_70_overload_60_80

	1	2	3	4	5	6	7	8	9	10
1	2	-1	-1	-1	-1	5	-1	-1	-1	2
2										
3										

Figure 3.3 *Target table.*

The target tables are necessary because we need to validate the results of our methods with the information of the tables in order to see how good our detection of electricity theft was. Moreover, the neural network methods that were tested in this thesis, demanded a target table as output for validation.

3.2 Data Manipulation and Features

In order to use the data table that described in the previous sub-chapter we had to use data manipulation techniques and some features for better results.

Feature extraction involves reducing the amount of resources required to describe a large set of data. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power, also it may cause a classification algorithm to overfit to training samples and generalize poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy. Many machine learning practitioners believe that properly optimized feature extraction is the key to effective model construction [31],[28],[37].

Below we present all the features and the methods we used in our work. Some of the features below used in combination and some others used only for some tests. Notice that the data manipulation was done in Matlab. For the theft detection techniques which were implemented outside of Matlab software, we just extracted the manipulated data into csv, binary or txt files depending on the method.

3.2.1 Features

Average

Relative mean based technique of feature extraction is a robust technique used for content identification.

The initial data had one measurement per 30 minutes for each consumer but we averaged them to one measurement per 8 hours. In our code, we created several averaged data tables from the original data tables. Specifically, we created one table with one measurement per day for each customer (average of 3 measurements, since one measurement is per 8 hours),one table with one measurement per month for each customer (average of 90 measurements, since one day has 3 measurements), one table with one measurement per 5 days for each consumer (average of 15 measurements, since one day is 3 measurements) and one table with one measurement per 10 days for each customer (average of 30 measurements, since one day is 3 measurements). We experimented with all these averaged tables and we ended up using the table with one measurement per day.

Correlation Coefficient

A correlation coefficient is a numerical measure of some type of correlation, meaning a statistical relationship between two variables. The variables may be two columns of a given data set of observations, often called a sample, or two components of a multivariate random variable with a known distribution.

Several types of correlation coefficient exist, each with their own definition and own range of usability and characteristics. They all assume values in the range from −1 to +1, where ±1 indicates the strongest possible agreement and 0 the strongest possible disagreement. As tools of analysis, correlation coefficients present certain problems, including the propensity of some types to be distorted by outliers and the possibility of incorrectly being used to infer a causal relationship between the variables [39].

The correlation coefficient of a data table was computed with Matlab function:

```
CC = corrcoef(x),
```

where x is the data table.

Range

In statistics, the range of a set of data is the difference between the largest and smallest values. However, in descriptive statistics, this concept of range has a more complex meaning. The range is the size of the smallest interval which contains all the data and provides an indication of statistical dispersion. It is measured in the same units as the data. Since it only depends on two of the observations, it is most useful in representing the dispersion of small data sets [38].

The range of the measurements for each customer was computed in Matlab, with the use of min-max functions :

```
Range_Data(i)=max(Data(:,i))-min(Data(:,i)),
```

where Data is our data table

Mean

In probability and statistics, the population mean, or expected value, are a measure of the central tendency either of a probability distribution or of the random variable characterized by that distribution. In the case of a discrete probability distribution of a random variable X, the mean is equal to the sum over every possible value weighted by the probability of that value; that is, it is computed by taking the product of each possible value x of X and its probability p(x), and then adding all these products together. An analogous formula applies to the case of a continuous probability distribution.
For a finite population, the population mean of a property is equal to the arithmetic mean of the given property while considering every member of the population. For example, the population mean height is equal to the sum of the heights of every individual divided by the total number of individuals. The sample mean may differ from the population mean, especially for small samples. The law of large numbers dictates that the larger the size of the sample, the more likely it is that the sample mean will be close to the population mean [40].

The mean of the measurements for each customer was computed with Matlab function :

```
Mean_data=mean(Data),
```
where Data is our data table.

Variance

In probability theory and statistics, variance is the expectation of the squared deviation of a random variable from its mean. Informally, it measures how far a set of (random) numbers are spread out from their average value. Variance has a central role in statistics, where some ideas that use it include descriptive statistics, statistical inference, hypothesis testing, goodness of fit, and Monte Carlo sampling. Variance is an important tool in the sciences, where statistical analysis of data is common [42].

The variance is the square of the standard deviation, the second central moment of a distribution, and the covariance of the random variable with itself, and it is computed with Matlab function:

```
Variance_Data=var(Data),
```
where Data is our data table.

Skewness

Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive , negative or undefined. For a unimodal distribution, negative skew commonly indicates that the tail is on the left side of the distribution, and positive skew indicates that the tail is on the right. Skewness in a data series may sometimes be observed not only graphically but by simple inspection of the values. This is how we used skewness as a feature in our work.

The skewness of the measurements of each consumer was computed with Matlab function:

```
SK_data=skewness(Data(:,C)),
```
where Data is our data table and C is a consumer.

Standar Deviation

In statistics, the standard deviation (SD, also represented by the lower case Greek letter sigma σ or the Latin letter s) is a measure that is used to quantify the amount of variation or dispersion of a set of data values. A low standard deviation indicates that the data points tend to be close to the mean (also called the expected value) of the set, while a high standard deviation indicates that the data points are spread out over a wider range of values.

The standard deviation of a random variable, statistical population, data set, or probability distribution is the square root of its variance. It is algebraically simpler, though in practice less robust, than the average absolute deviation. A useful property of the standard deviation is that, unlike the variance, it is expressed in the same units as the data.

In addition to express variability of a population, the standard deviation is commonly used to measure confidence in statistical conclusions. For example, the margin of error in polling data is determined by calculating the expected standard deviation in the results if the same poll were to be conducted multiple times. This derivation of a standard deviation is often called the "standard error" of the estimate or "standard error of the mean" when referring to a mean. It is computed as the standard deviation of all the means that would be computed from that population if an infinite number of samples were drawn and a mean for each sample were computed.

It is very important to note that the standard deviation of a population and the standard error of a statistic derived from that population (such as the mean) are quite different but related (related by the inverse of the square root of the number of observations). The reported margin of error of a poll is computed from the standard error of the mean (or alternatively from the product of the standard deviation of the population and the inverse of the square root of the sample size, which is the same thing) and is typically about twice the standard deviation—the half-width of a 95 percent confidence interval [41].

The standard deviation of the data table was computed with Matlab's function :

```
STD_Data=std(Data),
```

where Data is our data table.

3.2.2 Data Mining Techniques

Data Normalization

Data normalization is one of the main strategies that needs to be employed before performing classification. The basic purpose of data normalization is to keep check on the attributes having greater range of numeric values, so that the smaller sized values are not neglected. Normalization is a rescaling of the data from the original range so that all values are within the range of 0 and 1. Normalization requires that you know or are able to accurately estimate the minimum and maximum observable values. The minimum and maximum value of a table can be found with Matlab max() and min() functions.
A value is normalized as follows:

```
y=(x-min)/(max-min),
```

where the minimum and maximum values pertain to the value-table x that are being normalized.

Principal Component Analysis (PCA)

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance, and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric. PCA is sensitive to the relative scaling of the original variables.

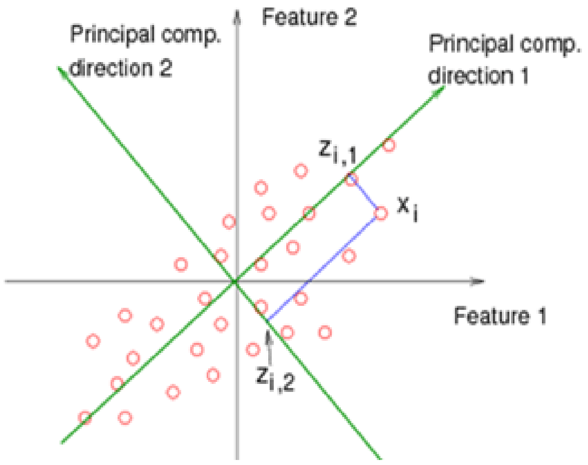


Figure 3.4 *Principal Component Analysis*

PCA is mostly used as a tool in exploratory data analysis and for making predictive models. PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of a data matrix, usually after mean centering (and normalizing or using Z-scores) the data matrix for each attribute. The results of a PCA are usually discussed in terms of component scores, sometimes called factor scores (the transformed variable values corresponding to a particular data point), and loadings (the weight by which each standardized original variable should be multiplied to get the component score).

PCA is the simplest of the true eigenvector-based multivariate analyses. Often, its operation can be thought of as revealing the internal structure of the data in a way that best explains the variance in the data. If a multivariate dataset is visualized as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA can supply the user with a lower-dimensional picture, a projection or "shadow" of this object when viewed from its (in some sense) most informative viewpoint. This is done by using only the first few principal components so that the dimensionality of the transformed data is reduced, while maintaining great variance [44],[28].

The principal component analysis of our data table was computed with Matlab’s function :

```
PCA_Data=pca(Data),
```
where Data is our data table.

3.3 Performance Metrics

Once you have built your model, the most important question that arises is how efficient it is. So, evaluating your model is the most important task in the data science project which delineates how satisfying your predictions are. Below we describe the most important metrics that we have used in our work. Some of them were not used in all the techniques we worked with, due to the programming environment or the limited tools the method could offer [49].

True positives and true negatives are the observations that are correctly predicted . We want to minimize false positives and false negatives so they are shown in red color. These terms are a bit confusing. So let's take each term one by one and understand it fully.

True Positives (TP)- These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. when a consumer is classified as illegal (say class I) and he is actually an illegal consumer (belongs to class I).

True Negatives (TN)- These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. when a consumer is classified as legal (say class L) and he is actually a legal consumer (belongs to class L).

False positives and false negatives, these values occur when your actual class contradicts with the predicted class.

False Positives (FP)– When actual class is no and predicted class is yes. E.g. when a consumer is classified as illegal (say class I) but he is actually a legal consumer (belongs to class L).

False Negatives (FN)– When actual class is yes but predicted class in no. E.g. when a consumer is classified as legal (say class L) but he is actually an illegal consumer (belongs to class I).

After these four parameters, we can calculate Accuracy, Precision, Recall and F1 score, which are the most important performance metrics.

Accuracy- Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost the same. Therefore, you have to look at other parameters to evaluate the performance of your model.

$$\text{Accuracy} = \frac{True\ Negatives + True\ Positives}{True\ Negatives + True\ Positives + False\ Positives + False\ Negatives}$$

Precision- Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate.

$$\text{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to all the observations in actual class.

$$\text{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

F1 score- F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances). Intuitively it is not so easy to perceive the accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have a similar cost. If the cost of false positives and false negatives is very different, it's better to look at both Precision and Recall. High precision but lower recall is extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better the performance of our model is. Mathematically, it can be expressed as:

$$\text{F1 Score} = \frac{2 * (Recall * Precision)}{Recall + Precision}$$

Mean Squared Error

Mean Squared Error (MSE) is quite similar to Mean Absolute Error, the only difference being that MSE takes the average of the square of the difference between the original values and the predicted values. The advantage of MSE being that it is easier to compute the gradient, whereas Mean Absolute Error requires complicated linear programming tools to compute the gradient. As, we take square of the error, the effect of larger errors become more pronounced than smaller error, hence the model can now focus more on the larger errors. This is computed in most cases with default functions.

Confusion Matrix

Confusion Matrix, as the name suggests, gives us a matrix as output and describes the complete performance of the model. As you can see from the figure 3.5, the green squares shows the true positives (square [1.1]) and true negatives (square [2.2]) while the red squares shows the false positives (square [1.2]) and false negatives (square [2.1]). The grey square shows the accuracy of the classification. You can see many of them in chapter 5, where we plot them with Matlab Deep Learning Toolbox in each case of electricity theft that was tested.

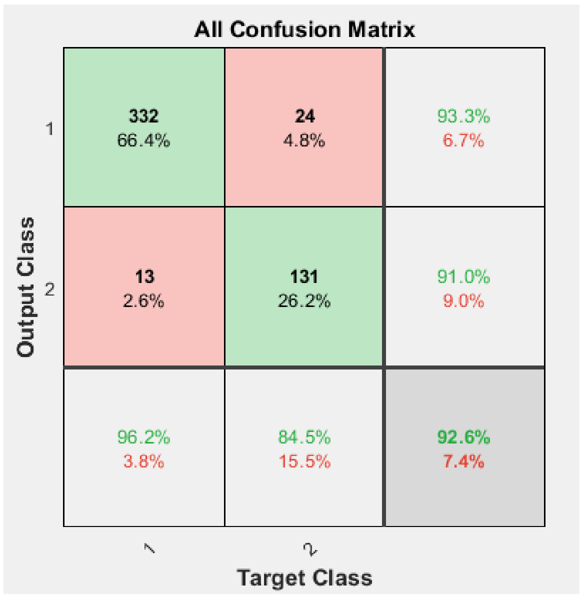


Figure 3.5 *Confusion Matrix*

4. Power Theft Detection Via Multilayer Artificial Neural Networks

This chapter describes the implementation of an artificial neural network with the use of the Fast Artificial Neural Network Library, called FANN. The library is written in ANSI C and its work is to implement multilayer feedforward networks with support for both fully connected and sparse connected networks. The library is designed to be fast, versatile and easy to use and as it turned out it offers great results for the detection of electricity theft.

4.1 About Artificial Neural Networks

First of all, lets talk about artificial neural networks which are the main reason of using this library. It is not possible (not at the moment at least) to make an artificial brain, but it is possible to make simplified artificial neurons and artificial neural networks (ANNs). These ANNs can be made in many different ways and can try to mimic the brain in many different ways. ANNs are not intelligent, but they are good for recognizing patterns and making simple rules for complex problems. They also have excellent training capabilities which are often used in artificial intelligence research. ANNs are good at generalizing from a set of training data. E.g. this means an ANN given data about a set of animals connected to a fact identifying them as mammals or not, is able to predict whether an animal outside the original set is a mammal from its data. This is a very desirable feature of ANNs, because it is not necessary to know the characteristics defining a mammal, the ANN will find out by itself. In our case, we use the ANN for the detection of illegal consumers of electricity.

This ANN library that I have chosen to use, implements multilayer feedforward ANNs, which is the most common kind of ANN. In a multilayer feedforward ANN, the neurons are ordered in layers, starting with an input layer and ending with an output layer. Between these two layers are a number of hidden layers. Connections in these kinds of network only go forward from one layer to the next. In the figure below we can see a fully connected multilayer feedforward network with one hidden layer. As you can see, all the neurons in each layer are connected to all the neurons in the next layer. This is called a fully connected network and although ANNs do not need to be fully connected, they often are [61],[62].

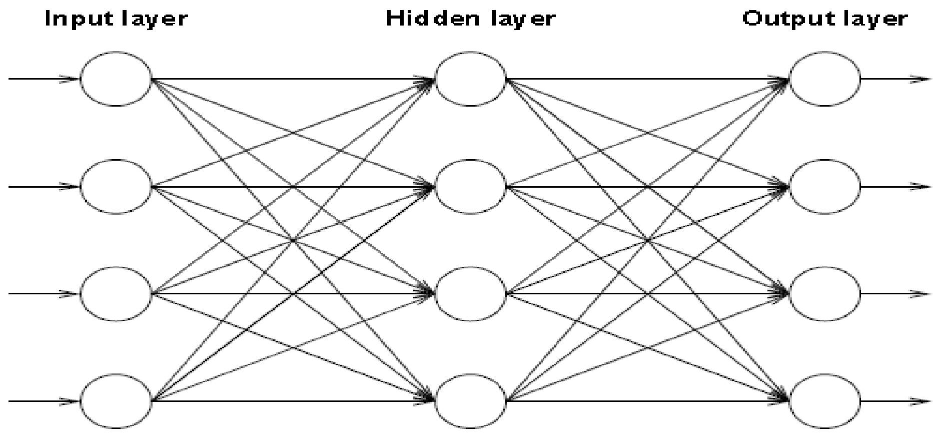


Figure 4.1 *A fully connected multilayer feedforward network with one hidden layer.*

Multilayer feedforward ANNs have two different phases: A training phase (sometimes also referred to as the learning phase) and an execution phase. In the training phase the ANN is trained to return a specific output when given a specific input, this is done by continuous training on a set of training data. In the execution phase the ANN returns outputs on the basis of inputs. The way the execution of a feedforward ANN functions is: An input is presented to the input layer, the input is propagated through all the layers until it reaches the output layer, where the output is returned. In a feedforward ANN an input can easily be propagated through the network and evaluated to an output. It is more difficult to compute a clear output from a network where connections are allowed in all directions (like in the brain), since this will create loops [61].

Two different kinds of parameters can be adjusted during the training of an ANN, the weights and the t value in the activation functions. This is impractical and it would be easier if only one of the parameters should be adjusted. To cope with this problem a bias neuron is invented. The bias neuron lies in one layer, is connected to all the neurons in the next layer, but to none in the previous layer and it always emits 1. Since the bias neuron emits 1 the weights, connected to the bias neuron, are added directly to the combined sum of the other weights, just like the t value in the activation functions. A modified equation for the neuron, where the weight for the bias neuron is represented as wn+1, is shown in equation 4.1:

$$g(x) = \frac{1}{1 + e^{-2sx}} \tag{4.1}$$

Adding the bias neuron allows us to remove the t value from the activation function, only leaving the weights to be adjusted, when the ANN is being trained. A modified version of the sigmoid function is shown in equation 4.2:

$$y(x) = g\left(w_{n+1} \sum_{i=0}^{n} w_i x_i \right) \tag{4.2}$$

We cannot remove the t value without adding a bias neuron, since this would result in a zero output from the sum function if all inputs where zero, regardless of the values of the weights. Some ANN libraries do, however, remove the t value without adding bias neurons, counting on the subsequent layers to get the right results. An ANN with added bias neurons is shown in the figure below.

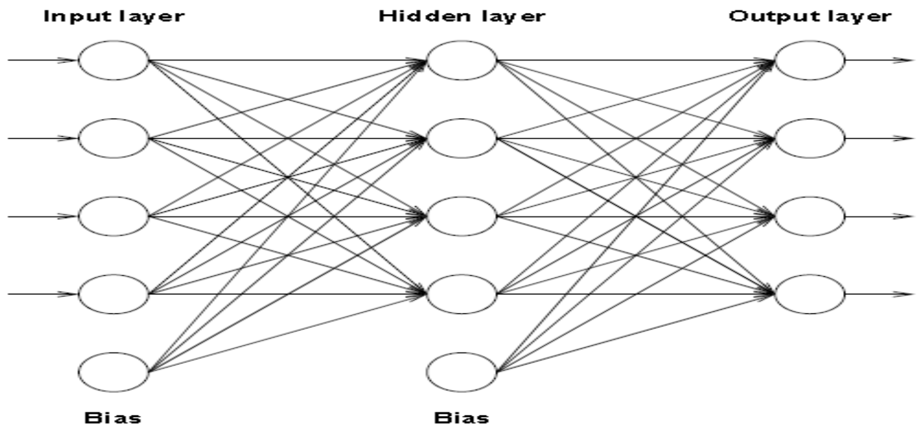


Figure 4.2 *A fully connected multilayer feedforward network with one hidden layer and bias neurons.*

When training an ANN with a set of input and output data, we wish to adjust the weights in the ANN, to make the ANN give the same outputs as seen in the training data. On the other hand, we have to avoid making the ANN too specific, because it will give precise results for the training data, but incorrect results for all other data. When this happens, the ANN has been over-fitted. The training process can be seen as an optimization problem, where we wish to minimize the mean square error of the entire set of training data. This problem can be solved in many different ways, ranging from standard optimization heuristics like simulated annealing, through more special optimization techniques like genetic algorithms to specialized gradient descent algorithms like backpropagation. The most used algorithm is the backpropagation algorithm [61].

Backpropagation algorithms are a family of methods used to efficiently train artificial neural networks (ANNs) following a gradient descent approach that exploits the chain rule. The main feature of backpropagation is its iterative, recursive and efficient method for calculating the weights updates to improve the network until it is able to perform the task for which it is being trained.

The backpropagation algorithm works in much the same way as the name suggests. After propagating an input through the network, the error is calculated and the error is propagated back through the network while the weights are adjusted in order to make the error smaller. Although we want to minimize the mean square error for all the training data, the most efficient way of doing this with the backpropagation algorithm, is to train on data sequentially one input at a time, instead of training on the combined data. However, this means that the order the data is given in is of importance, but it also provides a very efficient way of avoiding getting stuck in a local minima [61],[62].

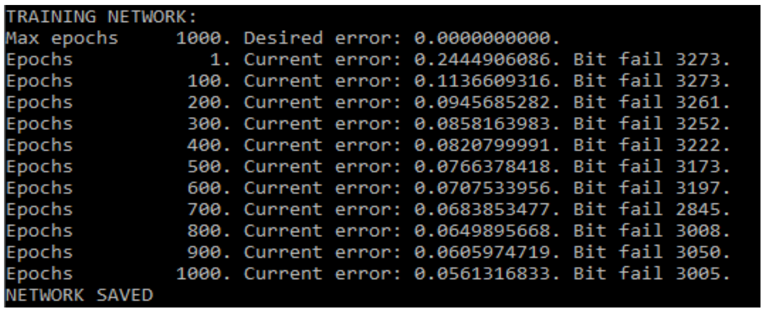
4.2 Neural Network Design

When creating a network it is necessary to define how many layers, neurons and connections it should have. If the network becomes too large, the ANN will have difficulties learning and it will tend eventually to over-fit, meaning that it will be fitted precisely to this set of training data and thereby loose generalization. If the network becomes too small, it will not be able to represent the rules needed to learn the problem and it will never gain a sufficiently low error rate.

The number of input and output layers can be defined by the data of the project and its requests. However, the number of hidden layers is also very important and it is a parameter that can only be specified with experimentation. Generally speaking, if the problem is simple it is often enough to have one or two hidden layers, but as the problems get more complex, so does the need for more layers.

In order to measure the quality of our ANN implementation, we have to check the mean square error values it can produce during a fixed period of training. The mean square error value is calculated while the ANN is being trained. Some functions of FANN library are implemented, to use and manipulate this error value. It is always hard to test the quality of an ANN implementation. How well a library performs on a given problem is a combination of a number of factors, including the initial weights, the training algorithm, the activation function and the parameters for this function. Especially the initial weights are tricky because they are set at random. For this reason two training sessions with the same data can give different results. Another problem is finding good datasets. ANN libraries perform differently on different datasets, meaning that just because one library is better at one problem does not mean that it is better at another problem [61]. For this reason, quality benchmarks were tested on several different datasets with different kinds and percentages of electricity theft in each data table. You can see all the data tables that have been used in the figure 3.1 in chapter 3.1.

In our implementation of the neural network for electricity theft detection using the FANN library we made a lot of experiments in order to conclude to the “optimal” neural network parameters. For the network sizes, the datasets are separated with 50% for training, 25% for validation and 25% for testing. However I didn’t do validation while training and for this reason I decided to use both the validation and test sets for testing. I did the quality benchmarks by training ANNs with training sets for a fixed period. Specifically, I was stopping the training phase every 100 echoes in order to write information about the mean square error for the training data and the testing data. For the calculation of the mean square error I used the same function on all the different libraries, to make sure that differences in this calculation does not affect the result. In figure 4.3 we display information about the mean square error for the training data during the network’s training phase.



```
TRAINING NETWORK:
Max epochs     1000. Desired error: 0.0000000000.
Epochs            1. Current error: 0.2444906086. Bit fail 3273.
Epochs          100. Current error: 0.1136609316. Bit fail 3273.
Epochs          200. Current error: 0.0945685282. Bit fail 3261.
Epochs          300. Current error: 0.0858163983. Bit fail 3252.
Epochs          400. Current error: 0.0820799991. Bit fail 3222.
Epochs          500. Current error: 0.0766378418. Bit fail 3173.
Epochs          600. Current error: 0.0707533956. Bit fail 3197.
Epochs          700. Current error: 0.0683853477. Bit fail 2845.
Epochs          800. Current error: 0.0649895668. Bit fail 3008.
Epochs          900. Current error: 0.0605974719. Bit fail 3050.
Epochs         1000. Current error: 0.0561316833. Bit fail 3005.
NETWORK SAVED
```

Figure 4.3 *Display information about mse in training phase.*

4.3 Neural Network Parameters and Input Data

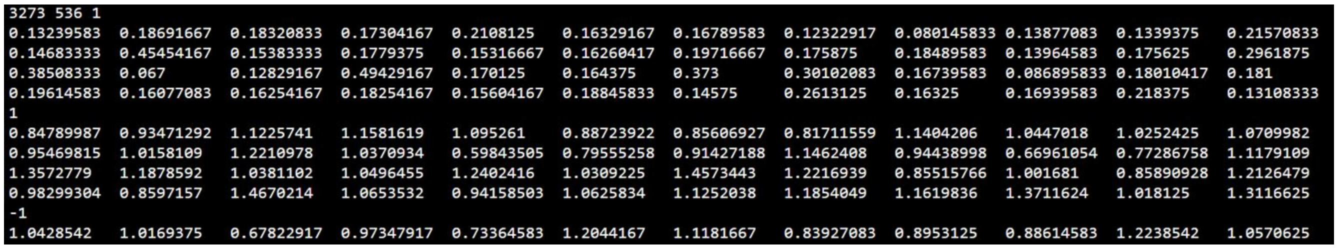
Creating the neural network architecture therefore means coming up with values for the number of layers of each type and the number of nodes in each of these layers.

The number of input layers is completely and uniquely determined once you know the shape of your training data. In our case, is depended of the feature extracted from our data and used as input. After a lot of experiments we conclude that the best input for our neural network was a normalized average of the electricity consumption measurements. Each customer's measurement was averaged in order to represent one day of electricity consumption. In total, we had 1608 measurements for each customer (3 measurements per day) which were used to create 536 (1608/3) measurements (on average). So, the number of input layers for our neural network was 536.

Like the input layers, every NN has also output layers. Its size (number of neurons) is completely determined by the chosen model configuration. Our NN runs in *Regression Mode* (the ML convention of using a term that is also used in statistics but assigning a different meaning to it is very confusing). In the neural networks that run in Regression Mode like ours, the output layer has a single node and returns only one value. This value defines if the consumer is fraudster (in this case the output will be close to -1) or non-fraudster (in this case the output will be close to 1) [46],[43].

The number of hidden layers is something that cannot be defined in theory. For most problems, one could probably get decent performance by setting the hidden layer configuration using just two rules: (i) number of hidden layers equals one and (ii) the number of neurons in that layer is the mean of the neurons in the input and output layers. However, none of these actually gave good results for our neural network. So after a lot of experiments with different numbers of hidden layer, we ended up using 35 hidden layers.

The FANN Library couldn't use the Matlab original energy consumption data with their default form. So, we had to export the data (the energy consumption data) in a .data file format, while making some adjustments to it. Firstly, we had to write to the first line the number of samples, the number of input layers and the number of output layers. Then we have the measurements-samples of each consumer followed by the target's table element (defines if the consumer is legal or illegal and it is used for validation of the results) that corresponds to the consumer. You can see figure 4.4 for a better understanding:



```
3273 536 1
0.13239583  0.18691667  0.18320833  0.17304167  0.2108125   0.16329167  0.16789583  0.12322917  0.080145833 0.13877083  0.1339375   0.21570833
0.14683333  0.45454167  0.15383333  0.1779375   0.15316667  0.16260417  0.19716667  0.175875    0.18489583  0.13964583  0.175625    0.2961875
0.38508333  0.067       0.12829167  0.49429167  0.170125    0.164375    0.373       0.30102083  0.16739583  0.086895833 0.18010417  0.181
0.19614583  0.16077083  0.16254167  0.18254167  0.15604167  0.18845833  0.14575     0.2613125   0.16325     0.16939583  0.218375    0.13108333
1
0.84789987  0.93471292  1.1225741   1.1581619   1.095261    0.88723922  0.85606927  0.81711559  1.1404206   1.0447018   1.0252425   1.0709982
0.95469815  1.0158109   1.2210978   1.0370934   0.59843505  0.79555258  0.91427188  1.1462408   0.94438998  0.66961054  0.77286758  1.1179109
1.3572779   1.1878592   1.0381102   1.0496455   1.2402416   1.0309225   1.4573443   1.2216939   0.85515766  1.001681    0.85890928  1.2126479
0.98299304  0.8597157   1.4670214   1.0653532   0.94158503  1.0625834   1.1252038   1.1854049   1.1619836   1.3711624   1.018125    1.3116625
-1
1.0428542   1.0169375   0.67822917  0.97347917  0.73364583  1.2044167   1.1181667   0.83927083  0.8953125   0.88614583  1.2238542   1.0570625
```

Figure 4.4 *Form of energy consumption data for input to FANN library.*

4.4 Creating, Training and Testing the Neural Network

The whole procedure of creating, training and testing the neural network achieved with FANN Library on a Linux terminal using the gcc compiler. After the data processing we mentioned in the previous subchapter, we used FANN's functions with some improving adjustments in order to create, train and test the neural network. Specifically, we set our parameters (layers size, epochs etc.) for the neural network and we created a makefile which is used to "run" all the functions which implement the neural network. These functions are:

Training phase

fann_create : Creates the ANN with a connection rate (1 for a fully connected network), a learning rate (0.7 is a reasonable default and the one we used) and a parameter saying how many layers the network should consist of (including the input and output layer). After this parameter follows one parameter for each layer (starting with the input layer) saying how many neurons should be in each layer.

fann_train_on _file : Trains the ANN for a maximum of max_epochs (we used the default price of 1000 epochs) epochs, or until the mean square error is lower than the desired error (we set the desired error to 0, a price that is actually unreliable). A status line is written every 100 epochs between reports epoch. We noticed that by increasing the number of max_echoes the neural network results was much better, but in this case the neural network tends to over-fit to specific data losing generalization.

fann_save : Saves the ANN to a file, in order to use it later during execution.

fann_destroy : Destroys the ANN and deallocates the memory it used. The conguration file saved by fann_save contains all the information we needed in order to recreate the network.

Execution phase

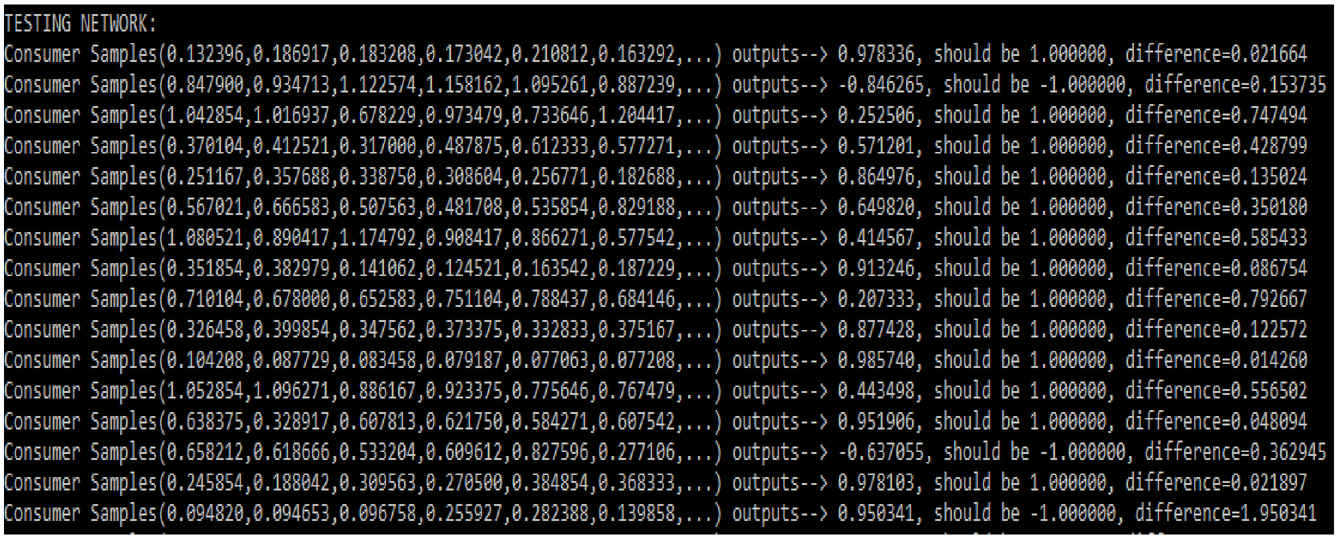
fann_create_from _file : Creates the network from a conguration file, which has earlier been saved by the training program with fann_save function.

fann_run : Executes the input on the ANN and returns the output from the ANN.

fann_type : Is the type used internally by the fann library. This type is float when including floatfann.h, double when including doublefann.h and int when including fixedfann.h. In our case we used double type (including doublefann.h).

The above six are the basic (with the fann_type described) functions of the FANN library. Of course, we worked with more library functions inside the code but not as basic as the above six.

The test of the artificial neural network gives an output in the range [-1,1] like you can see in figure 4.5 below:



```
TESTING NETWORK:
Consumer Samples(0.132396,0.186917,0.183208,0.173042,0.210812,0.163292,...) outputs--> 0.978336, should be 1.000000, difference=0.021664
Consumer Samples(0.847900,0.934713,1.122574,1.158162,1.095261,0.887239,...) outputs--> -0.846265, should be -1.000000, difference=0.153735
Consumer Samples(1.042854,1.016937,0.678229,0.973479,0.733646,1.204417,...) outputs--> 0.252506, should be 1.000000, difference=0.747494
Consumer Samples(0.370104,0.412521,0.317000,0.487875,0.612333,0.577271,...) outputs--> 0.571201, should be 1.000000, difference=0.428799
Consumer Samples(0.251167,0.357688,0.338750,0.308604,0.256771,0.182688,...) outputs--> 0.864976, should be 1.000000, difference=0.135024
Consumer Samples(0.567021,0.666583,0.507563,0.481708,0.535854,0.829188,...) outputs--> 0.649820, should be 1.000000, difference=0.350180
Consumer Samples(1.080521,0.890417,1.174792,0.908417,0.866271,0.577542,...) outputs--> 0.414567, should be 1.000000, difference=0.585433
Consumer Samples(0.351854,0.382979,0.141062,0.124521,0.163542,0.187229,...) outputs--> 0.913246, should be 1.000000, difference=0.086754
Consumer Samples(0.710104,0.678000,0.652583,0.751104,0.788437,0.684146,...) outputs--> 0.207333, should be 1.000000, difference=0.792667
Consumer Samples(0.326458,0.399854,0.347562,0.373375,0.332833,0.375167,...) outputs--> 0.877428, should be 1.000000, difference=0.122572
Consumer Samples(0.104208,0.087729,0.083458,0.079187,0.077063,0.077208,...) outputs--> 0.985740, should be 1.000000, difference=0.014260
Consumer Samples(1.052854,1.096271,0.886167,0.923375,0.775646,0.767479,...) outputs--> 0.443498, should be 1.000000, difference=0.556502
Consumer Samples(0.638375,0.328917,0.607813,0.621750,0.584271,0.607542,...) outputs--> 0.951906, should be 1.000000, difference=0.048094
Consumer Samples(0.658212,0.618666,0.533204,0.609612,0.827596,0.277106,...) outputs--> -0.637055, should be -1.000000, difference=0.362945
Consumer Samples(0.245854,0.188042,0.309563,0.270500,0.384854,0.368333,...) outputs--> 0.978103, should be 1.000000, difference=0.021897
Consumer Samples(0.094820,0.094653,0.096758,0.255927,0.282388,0.139858,...) outputs--> 0.950341, should be -1.000000, difference=1.950341
```

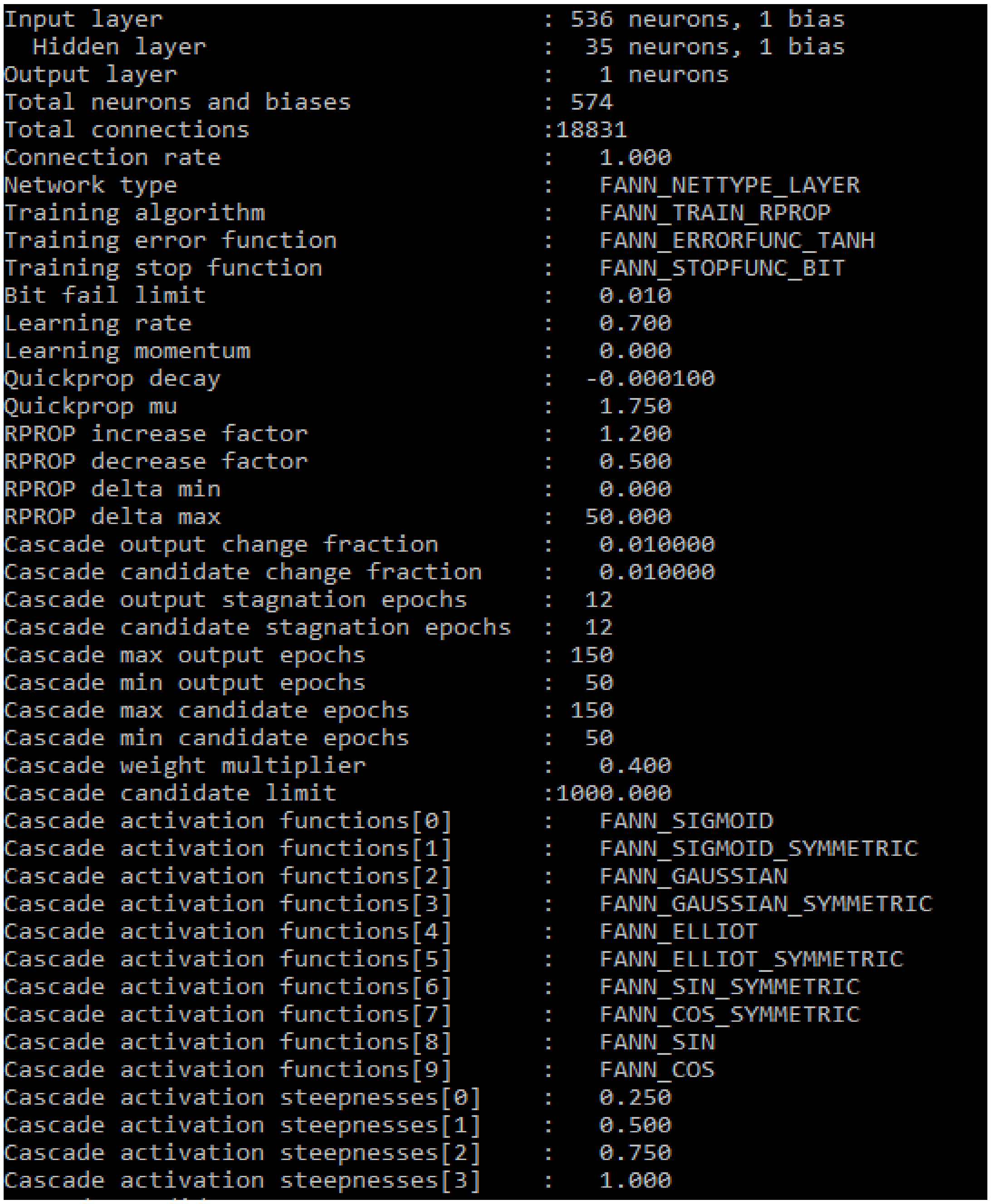
Figure 4.5 *Testing phase of the neural network.*

In order to find the accuracy of the neural network we implement a “rule” model which decides, depending on the output of the neural network, if the consumer is fraudster or non-fraudster. As you can see in the image above we have a price called difference, which tells the numerical difference between the output of the neural network and the desired outcome (which is known from the target data table). As we have already said in the previous chapter the value 1 corresponds to a legal consumer and the value -1 corresponds to an illegal consumer. So we use this difference value in order to see if the neural network output is closer to 1 or -1 and depending on that we make the final decision about the consumer. In most cases this rule model works great but as you can see in the last line of the image above the neural network had made a fault estimation about the consumer (so the rule model based on this estimation will decide wrongly). Luckily cases like this one are minimal and the performance of the neural network remains very good.

4.5 Results

As we mentioned in the beginning of chapter 3, the real-time energy consumption measurements we have in our possession is a struct of data with different kinds and percentages of electricity theft in each data table. We tested the performance of our neural network in each of these cases as you can see in the images below. The procedure was very fast with this method, so we used all the consumers (3273) for the classification. The accuracy is representative of the average accuracy in each case. Each case tested 5-10 times in order to end up to the images below.

An overview of the neural network:

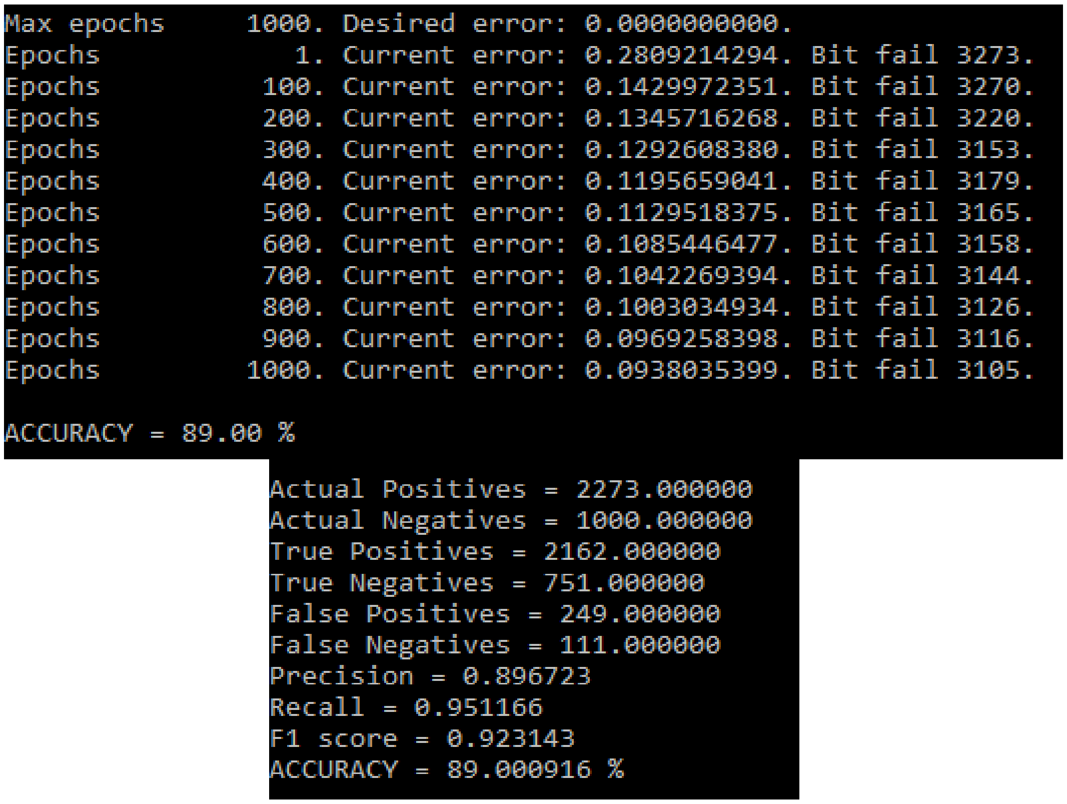


```
Input layer                          : 536 neurons, 1 bias
  Hidden layer                       :  35 neurons, 1 bias
Output layer                         :   1 neurons
Total neurons and biases             : 574
Total connections                    :18831
Connection rate                      :   1.000
Network type                         :   FANN_NETTYPE_LAYER
Training algorithm                   :   FANN_TRAIN_RPROP
Training error function              :   FANN_ERRORFUNC_TANH
Training stop function               :   FANN_STOPFUNC_BIT
Bit fail limit                       :   0.010
Learning rate                        :   0.700
Learning momentum                    :   0.000
Quickprop decay                      :  -0.000100
Quickprop mu                         :   1.750
RPROP increase factor                :   1.200
RPROP decrease factor                :   0.500
RPROP delta min                      :   0.000
RPROP delta max                      :  50.000
Cascade output change fraction       :   0.010000
Cascade candidate change fraction    :   0.010000
Cascade output stagnation epochs     :  12
Cascade candidate stagnation epochs  :  12
Cascade max output epochs            : 150
Cascade min output epochs            :  50
Cascade max candidate epochs         : 150
Cascade min candidate epochs         :  50
Cascade weight multiplier            :   0.400
Cascade candidate limit              :1000.000
Cascade activation functions[0]      :   FANN_SIGMOID
Cascade activation functions[1]      :   FANN_SIGMOID_SYMMETRIC
Cascade activation functions[2]      :   FANN_GAUSSIAN
Cascade activation functions[3]      :   FANN_GAUSSIAN_SYMMETRIC
Cascade activation functions[4]      :   FANN_ELLIOT
Cascade activation functions[5]      :   FANN_ELLIOT_SYMMETRIC
Cascade activation functions[6]      :   FANN_SIN_SYMMETRIC
Cascade activation functions[7]      :   FANN_COS_SYMMETRIC
Cascade activation functions[8]      :   FANN_SIN
Cascade activation functions[9]      :   FANN_COS
Cascade activation steepnesses[0]    :   0.250
Cascade activation steepnesses[1]    :   0.500
Cascade activation steepnesses[2]    :   0.750
Cascade activation steepnesses[3]    :   1.000
```

Figure 4.6 *Overview of the neural network*

Below we quote some images with the accuracy metrics of the neural network in each case we experimented with. Because this method was implemented in a Linux terminal we could not have visual presentation for the results as we did with the methods in Matlab.

<u>1st CASE : Theft with overload 40%-60%</u>

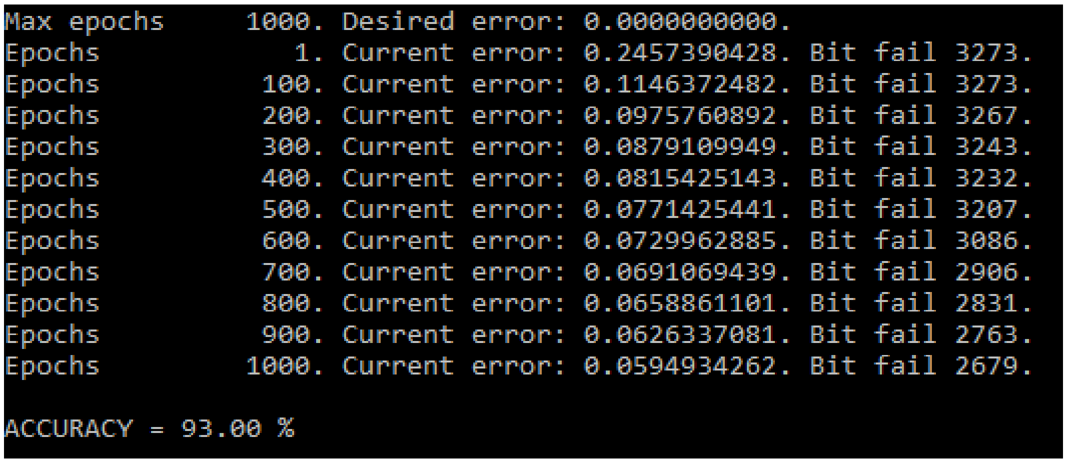


```
Max epochs     1000. Desired error: 0.0000000000.
Epochs            1. Current error: 0.2809214294. Bit fail 3273.
Epochs          100. Current error: 0.1429972351. Bit fail 3270.
Epochs          200. Current error: 0.1345716268. Bit fail 3220.
Epochs          300. Current error: 0.1292608380. Bit fail 3153.
Epochs          400. Current error: 0.1195659041. Bit fail 3179.
Epochs          500. Current error: 0.1129518375. Bit fail 3165.
Epochs          600. Current error: 0.1085446477. Bit fail 3158.
Epochs          700. Current error: 0.1042269394. Bit fail 3144.
Epochs          800. Current error: 0.1003034934. Bit fail 3126.
Epochs          900. Current error: 0.0969258398. Bit fail 3116.
Epochs         1000. Current error: 0.0938035399. Bit fail 3105.

ACCURACY = 89.00 %
```

```
Actual Positives = 2273.000000
Actual Negatives = 1000.000000
True Positives = 2162.000000
True Negatives = 751.000000
False Positives = 249.000000
False Negatives = 111.000000
Precision = 0.896723
Recall = 0.951166
F1 score = 0.923143
ACCURACY = 89.000916 %
```

Figure 4.7 *Performance Metrics for Case 1*

<u>2nd CASE : Theft with overload 60%-80%</u>



```
Max epochs     1000. Desired error: 0.0000000000.
Epochs            1. Current error: 0.2457390428. Bit fail 3273.
Epochs          100. Current error: 0.1146372482. Bit fail 3273.
Epochs          200. Current error: 0.0975760892. Bit fail 3267.
Epochs          300. Current error: 0.0879109949. Bit fail 3243.
Epochs          400. Current error: 0.0815425143. Bit fail 3232.
Epochs          500. Current error: 0.0771425441. Bit fail 3207.
Epochs          600. Current error: 0.0729962885. Bit fail 3086.
Epochs          700. Current error: 0.0691069439. Bit fail 2906.
Epochs          800. Current error: 0.0658861101. Bit fail 2831.
Epochs          900. Current error: 0.0626337081. Bit fail 2763.
Epochs         1000. Current error: 0.0594934262. Bit fail 2679.

ACCURACY = 93.00 %
```

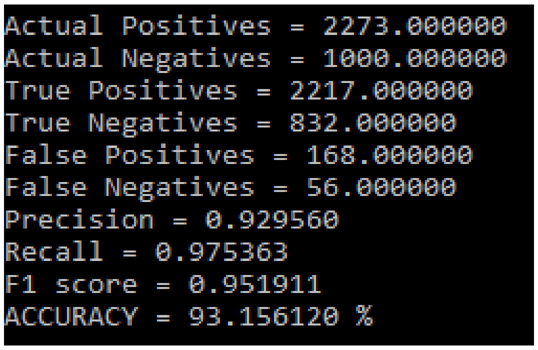

```
Actual Positives = 2273.000000
Actual Negatives = 1000.000000
True Positives = 2217.000000
True Negatives = 832.000000
False Positives = 168.000000
False Negatives = 56.000000
Precision = 0.929560
Recall = 0.975363
F1 score = 0.951911
ACCURACY = 93.156120 %
```

Figure 4.8 *Performance Metrics for Case 2*

3rd CASE : Theft with overload 80%-100%

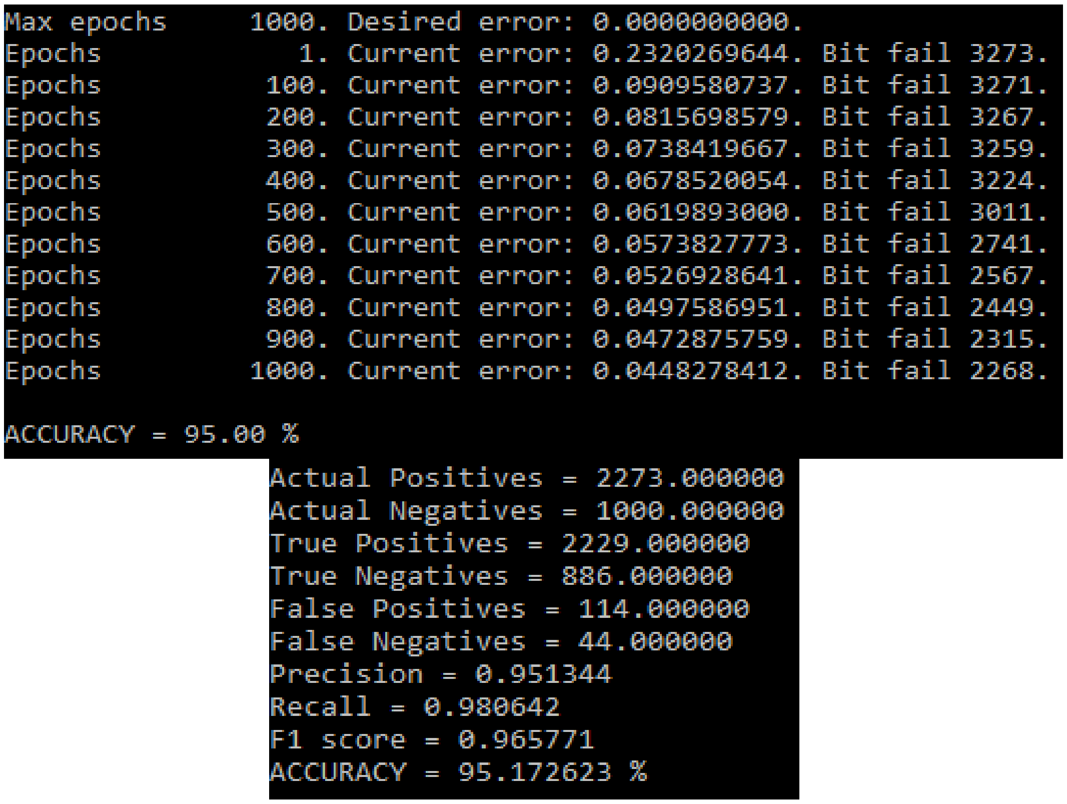
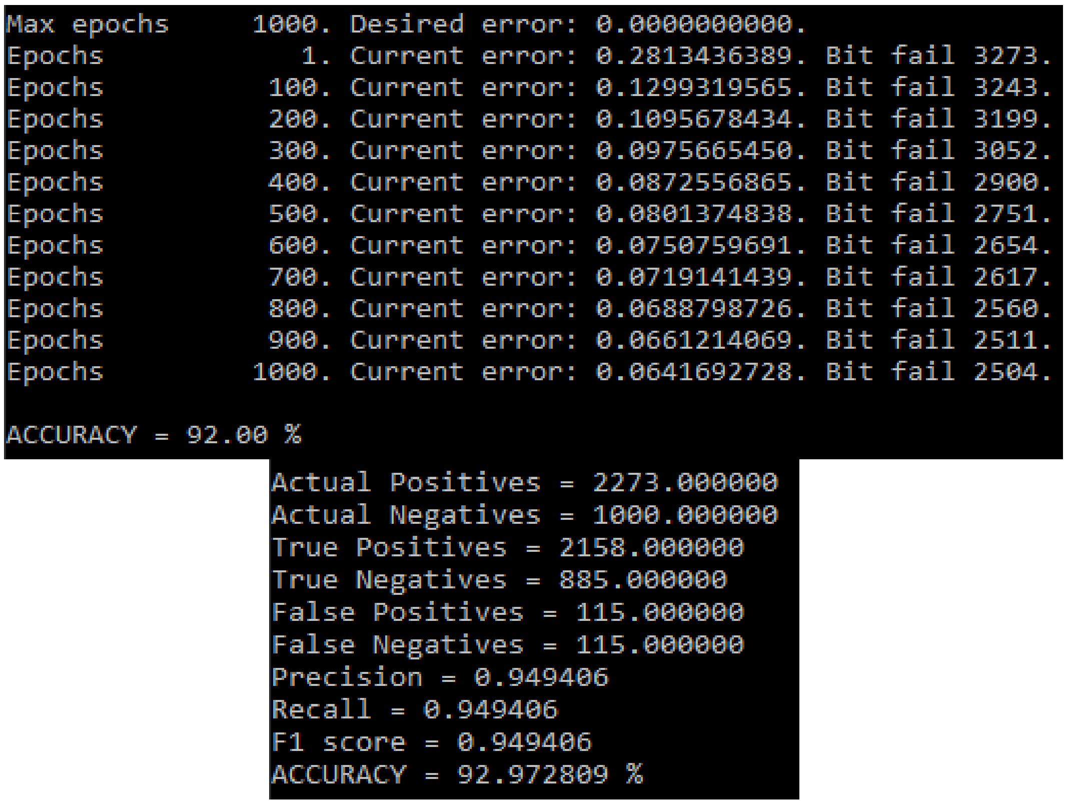

```
Max epochs     1000. Desired error: 0.0000000000.
Epochs            1. Current error: 0.2320269644. Bit fail 3273.
Epochs          100. Current error: 0.0909580737. Bit fail 3271.
Epochs          200. Current error: 0.0815698579. Bit fail 3267.
Epochs          300. Current error: 0.0738419667. Bit fail 3259.
Epochs          400. Current error: 0.0678520054. Bit fail 3224.
Epochs          500. Current error: 0.0619893000. Bit fail 3011.
Epochs          600. Current error: 0.0573827773. Bit fail 2741.
Epochs          700. Current error: 0.0526928641. Bit fail 2567.
Epochs          800. Current error: 0.0497586951. Bit fail 2449.
Epochs          900. Current error: 0.0472875759. Bit fail 2315.
Epochs         1000. Current error: 0.0448278412. Bit fail 2268.

ACCURACY = 95.00 %
```
```
Actual Positives = 2273.000000
Actual Negatives = 1000.000000
True Positives = 2229.000000
True Negatives = 886.000000
False Positives = 114.000000
False Negatives = 44.000000
Precision = 0.951344
Recall = 0.980642
F1 score = 0.965771
ACCURACY = 95.172623 %
```

Figure 4.9 *Performance Metrics for Case 3*

4th CASE : Partial theft 30%-50%

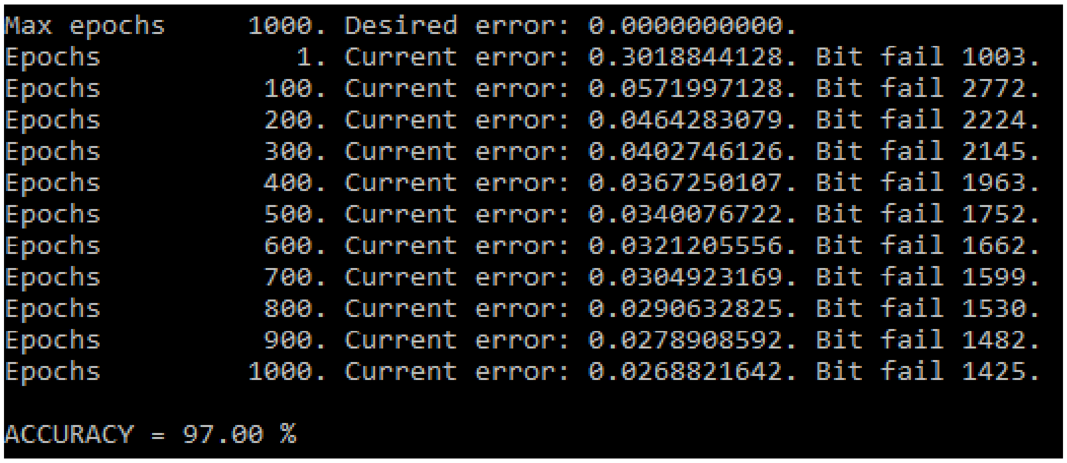


```
Max epochs     1000. Desired error: 0.0000000000.
Epochs            1. Current error: 0.2813436389. Bit fail 3273.
Epochs          100. Current error: 0.1299319565. Bit fail 3243.
Epochs          200. Current error: 0.1095678434. Bit fail 3199.
Epochs          300. Current error: 0.0975665450. Bit fail 3052.
Epochs          400. Current error: 0.0872556865. Bit fail 2900.
Epochs          500. Current error: 0.0801374838. Bit fail 2751.
Epochs          600. Current error: 0.0750759691. Bit fail 2654.
Epochs          700. Current error: 0.0719141439. Bit fail 2617.
Epochs          800. Current error: 0.0688798726. Bit fail 2560.
Epochs          900. Current error: 0.0661214069. Bit fail 2511.
Epochs         1000. Current error: 0.0641692728. Bit fail 2504.

ACCURACY = 92.00 %
```

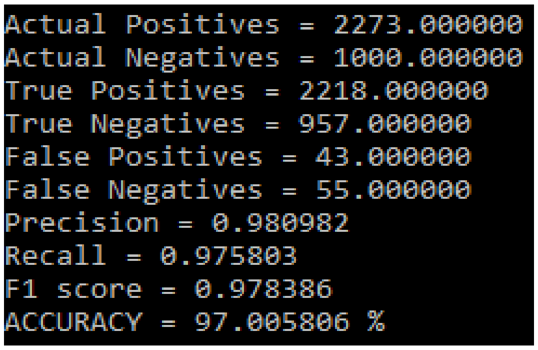
```
Actual Positives = 2273.000000
Actual Negatives = 1000.000000
True Positives = 2158.000000
True Negatives = 885.000000
False Positives = 115.000000
False Negatives = 115.000000
Precision = 0.949406
Recall = 0.949406
F1 score = 0.949406
ACCURACY = 92.972809 %
```

Figure 4.10 *Performance Metrics for Case 4*

5th CASE : Partial theft 50%-70%



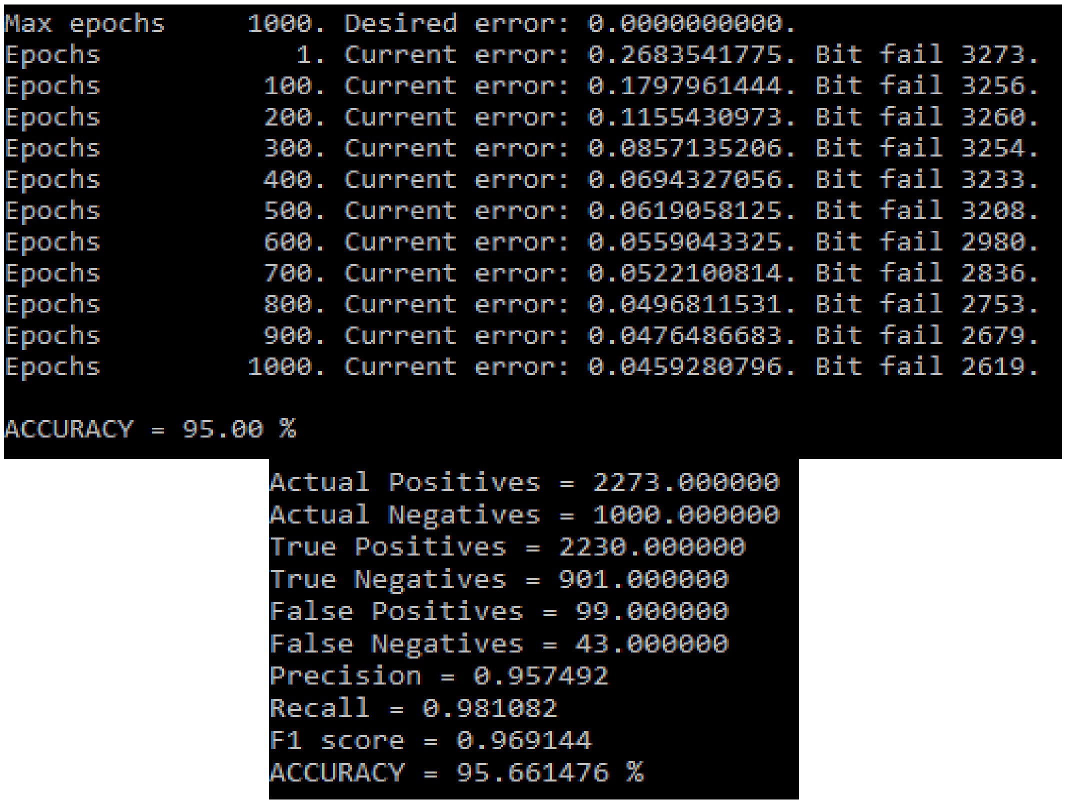
```
Max epochs     1000. Desired error: 0.0000000000.
Epochs            1. Current error: 0.3018844128. Bit fail 1003.
Epochs          100. Current error: 0.0571997128. Bit fail 2772.
Epochs          200. Current error: 0.0464283079. Bit fail 2224.
Epochs          300. Current error: 0.0402746126. Bit fail 2145.
Epochs          400. Current error: 0.0367250107. Bit fail 1963.
Epochs          500. Current error: 0.0340076722. Bit fail 1752.
Epochs          600. Current error: 0.0321205556. Bit fail 1662.
Epochs          700. Current error: 0.0304923169. Bit fail 1599.
Epochs          800. Current error: 0.0290632825. Bit fail 1530.
Epochs          900. Current error: 0.0278908592. Bit fail 1482.
Epochs         1000. Current error: 0.0268821642. Bit fail 1425.

ACCURACY = 97.00 %
```



```
Actual Positives = 2273.000000
Actual Negatives = 1000.000000
True Positives = 2218.000000
True Negatives = 957.000000
False Positives = 43.000000
False Negatives = 55.000000
Precision = 0.980982
Recall = 0.975803
F1 score = 0.978386
ACCURACY = 97.005806 %
```

Figure 4.11 *Performance Metrics for Case 5*

6th CASE : Theft with overload 60%-80% and partial theft 50%-70%



```
Max epochs     1000. Desired error: 0.0000000000.
Epochs            1. Current error: 0.2683541775. Bit fail 3273.
Epochs          100. Current error: 0.1797961444. Bit fail 3256.
Epochs          200. Current error: 0.1155430973. Bit fail 3260.
Epochs          300. Current error: 0.0857135206. Bit fail 3254.
Epochs          400. Current error: 0.0694327056. Bit fail 3233.
Epochs          500. Current error: 0.0619058125. Bit fail 3208.
Epochs          600. Current error: 0.0559043325. Bit fail 2980.
Epochs          700. Current error: 0.0522100814. Bit fail 2836.
Epochs          800. Current error: 0.0496811531. Bit fail 2753.
Epochs          900. Current error: 0.0476486683. Bit fail 2679.
Epochs         1000. Current error: 0.0459280796. Bit fail 2619.

ACCURACY = 95.00 %
```

```
Actual Positives = 2273.000000
Actual Negatives = 1000.000000
True Positives = 2230.000000
True Negatives = 901.000000
False Positives = 99.000000
False Negatives = 43.000000
Precision = 0.957492
Recall = 0.981082
F1 score = 0.969144
ACCURACY = 95.661476 %
```

Figure 4.12 *Performance Metrics for Case 6*

As you can see from table 4.1 the performance metrics of this method were very satisfying in all cases. These results are largely due to the “rule” model we applied at the output of the neural network and based on that, we decided about each consumer’s legality with great success.

CASES	**Accuracy**	**Recall**	**Precision**	**F1 Score**	**True Positives**	**True Negatives**	**False Positives**	**False Negatives**	**Time (s)**
1st Case	89%	0,9511	0,8967	0,9231	2162	751	249	111	105
2nd Case	93%	0,9753	0,9295	0,9519	2217	832	168	56	111
3rd Case	95%	0,9806	0,9513	0,9657	2229	886	114	44	121
4th Case	92%	0,9494	0,9494	0,9494	2158	885	115	115	117
5th Case	97%	0,9758	0,9809	0,9783	2218	957	43	55	126
6th Case	95%	0,9810	0,9574	0,9691	2230	901	99	43	129

Table 4.1 *Performance Metrics*

Generally, the FANN library that was used in this method, was easy to use, fast (faster than all the other methods, except of the method with optimum path forest classifiers) and versatile (we could adjust many parameters easily).

From all the methods that were tested in this thesis, the method with the multilayer artificial neural networks was one of the best, both in performance and speed and for this reason is totally recommended for electricity theft detection.

5. Power Theft Detection Via Neural Networks

This chapter describes the implementation of a neural network with Deep Learning Toolbox in Matlab. Deep Learning Toolbox (formerly Neural Network Toolbox) provides a framework for designing and implementing neural networks with algorithms, pretrained models, and apps. You can use convolutional neural networks (ConvNets, CNNs) and long short-term memory (LSTM) networks to perform classification and regression on image, time-series, and text data.

5.1 About Deep Learning Toolbox

Matlab and Deep Learning Toolbox provide command-line functions and apps for creating, training, and simulating neural networks. The apps make it easy to develop neural networks for tasks such as classification, regression (including time-series regression), clustering and plots help you visualize activations, edit network architectures, and monitor training progress.

The Deep Learning Toolbox has 4 categories of "wizards-apps", each one of those is suitable for solving a different kind of problem. Specifically these apps are:

• Fitting app: In fitting problems, the neural network have to map between a data set of numeric inputs and a set of numeric targets. Examples of this type of problem include estimating house prices from such input variables as tax rate, pupil/teacher ratio in local schools and crime rate (house_dataset) estimating engine emission levels based on measurements of fuel consumption and speed (engine_dataset) or predicting a patient's bodyfat level based on body measurements (bodyfat_dataset). The Neural Fitting app has everything needed for data selection, creating-training a network and evaluating its performance with mean square error and regression analysis.

• Pattern Recognition app: In pattern recognition problems, the neural network have to classify inputs into a set of target categories. For example, recognize the vineyard that a particular bottle of wine came from, based on chemical analysis (wine_dataset) or classify a tumor as benign or malignant, based on uniformity of cell size, clump thickness, mitosis (cancer_dataset). The Neural Pattern Recognition app has everything needed for data selection, creating-training a network and evaluating its performance with cross-entropy and confusion matrices.

• Clustering app: In clustering problems, the neural network have to group data by similarity. For example: market segmentation done by grouping people according to their buying patterns, data mining can be done by partitioning data into related subsets, a bioinformatic analysis has to group the genes with related expression patterns. The Neural Clustering app includes everything needed for data selection, creating-training a network and evaluating its performance with a variety of visualization tools.

• Time Series app: Prediction is a kind of dynamic filtering, in which past values of one or more time series are used to predict future values. Dynamic neural networks, which include tapped delay lines, are used for nonlinear filtering and prediction. There are many applications for prediction. For example, a financial analyst might want to predict the future value of a stock, bond or other financial instrument. An engineer might want to predict the impending failure of a jet engine. Predictive models are also used for system identification (or dynamic modeling), in which you build dynamic models of physical systems. These dynamic models are important for analysis, simulation, monitoring and control of a variety of systems, including manufacturing systems, chemical processes, robotics and aerospace systems.

For the detection of electricity theft in this method, we used the Pattern Recognition app. As the results showed, this app detect to a great extend the patterns in electricity consumption irregularities.

5.2 Neural Network Parameters and Input Data

Creating the neural network means coming up with values for the number of layers of each type and the number of nodes in each of these layers.

The number of input layers is completely and uniquely determined once you know the shape of your training data. In our case, it depends on the feature we extracted from our data and used as input. After a lot of experiments we conclude that the best input for our neural network was the principal component analysis of the daily average of electricity consumption measurements. Each consumer's measurement was averaged in order to represent one day of electricity consumption. In total, we had 1608 measurements for each customer (3 measurements per day) which were used to create 536 (1608/3) measurements (on average). Then, we used the principal component analysis into our averaged data. This is done by using only the first 20 principal components (88% variance) so that the dimensionality of the transformed data is reduced.

Like the input layers, every NN has also output layers. Its size (number of neurons) is completely determined by the chosen model configuration. In our case, we have 2 output layers. The output layer value defines if the consumer is fraudster (in this case the output will be 2) or non-fraudster (in this case the output will be 1). Actually, the pattern recognition app outputs a vector with values 1 or 2 depending on the predicted result.

The number of hidden neurons is something that cannot be defined in theory. For most problems, one could probably get decent performance by setting the hidden neuron configuration using just two rules: (i) number of hidden neurons equals one and (ii) the number of neurons in that layer is the mean of the neurons in the input and output layers [46]. However, none of these actually gave good results for our neural network. So after a lot of experiments with different numbers of hidden neurons, we ended up changing the number of hidden neurons depending on the dataset we worked with. As we described in chapter 3, the data was a struct of datasets with different kinds and percentages of electricity theft. For each dataset we tested multiple times the neural network in order to conclude to the number of hidden neurons for each dataset. So we have:

- For the dataset with partial theft 30%-50% we used 28 hidden neurons
- For the dataset with partial theft 50%-70% we used 32 hidden neurons
- For the dataset with overload 40%-60% we used 37 hidden neurons
- For the dataset with overload 60%-80% we used 23 hidden neurons
- For the dataset with overload 80%-100% we used 37 hidden neurons
- For the dataset with overload 60%-80% and partial theft 50%-70% we used 32 hidden neurons

Preprocessing the network inputs, targets and parameters is very important in the implementation of neural networks as it improves the efficiency of neural network training and as a result its performance.

Moreover we had to increase the neural network's ability to generalize, which helps to prevent overfitting, a common problem in neural network design. Overfitting occurs when a network has memorized the training set and does not generalize new data inputs. Overfitting produces a relatively small error on the training set but a much larger error when new data is presented to the network.

Two solutions to improve generalization include [63]:

- Regularization modifies the network's performance function (the measure of error that the training process minimizes). By including the sizes of the weights and biases, regularization produces a network that performs well with the training data and exhibits smoother behavior when presented with new data.
- Early stopping uses two different data sets: the training set, to update the weights and biases, and the validation set, to stop training when the network begins to overfit the data.

5.3 Creating, Training and Testing the Neural Network

Each neural network application we described in chapter 5.1 is unique, but developing the neural network typically follows these steps:

- Preparing your data and configuring the network's inputs and outputs

As we already mentioned, we averaged the electricity consumption measurements, so that each measurement corresponds to one day. We combine this averaged data with principal component analysis in order to reduce the dimensions of our datasets (while keeping great variance), speed up the neural network and improve the performance of the neural network. The neural network inputs and outputs are determined from the "shape" of our dataset and the hidden layer size chosen by multiple experiments.

- Creating the neural network

After setting all the parameters of the neural network we used Matlab function *patternnet which is a function that creates a* feedforward neural network (or pattern recognition network in this case) that can be trained to classify inputs according to target classes.

- Training the neural network

Before training the neural network (or pattern recognition app) we had to choose the neural network training function that updates weight and bias values. We experimented with different training function before concluding to Matlab function *trainlm. trainlm is a neural network training function that updates weight and bias values according to Levenberg-Marquardt optimization. It is* the fastest backpropagation algorithm in the deep learning toolbox, and is highly recommended as a first-choice supervised algorithm, although it does require more memory than other algorithms.

Then, after multiple experiments we decided about the setup division of data for training, testing and validation. We divided up every sample randomly (with Matlab) and concluded that the best configuration was to use 80% for training, 10% for testing and 10% for validation.

We chose the *msereg* (mean squared error with regularization performance function) as the neural network's performance function . It measures network performance as the weight sum of two factors: the mean squared error and the mean squared weights and biases.

Lastly we used the Matlab's function *train()* in order to train the neural network.

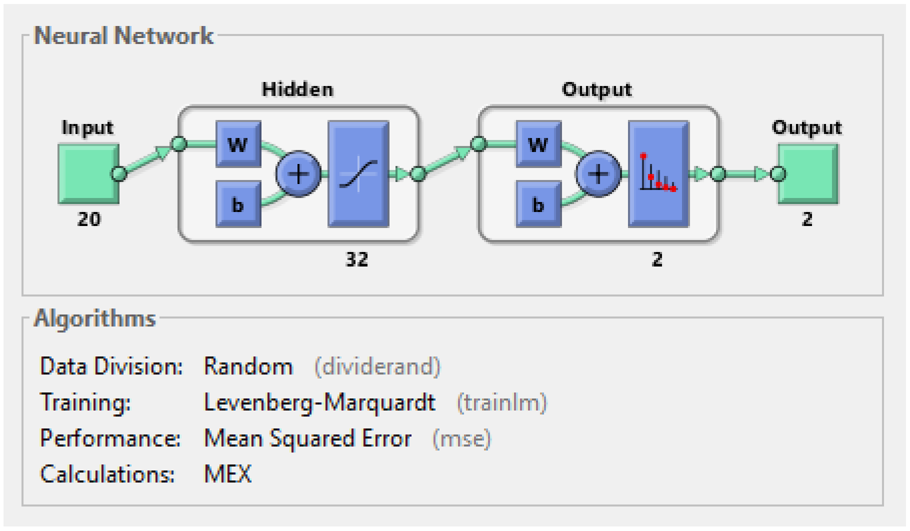
- <u>Testing, validating and plotting the neural network</u>

We used Matlab function *perform()* for testing the neural network and recalculating training, testing and validation performance. Then we computed some neural network's accuracy metrics like confusion matrices, precision, recall and F1. In the end we used some plots in order to view the neural network, the confusion matrices, the performance etc.

5.4 Results

As we mentioned in the beginning of chapter 3, the real-time energy consumption measurements we have in our possession is a struct of data with different kinds and percentages of electricity theft in each data table. We tested the performance of our neural network in each of these cases as you can see in figure 6.1. The accuracy is representative of the average accuracy in each case. Each case was tested 5-10 times, which lead to the images below.

An overview of the neural network:



Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

Figure 6.1 *Overview of the neural network*

Below we quote some images with the performance, the error histograms and the confusion matrices of the neural network in each case we experimented with. The images below is from simulations with 500 consumers in order to save time. A full simulation of the 3273 consumers could take up to 10 minutes.

1st CASE : Theft with overload 40%-60%

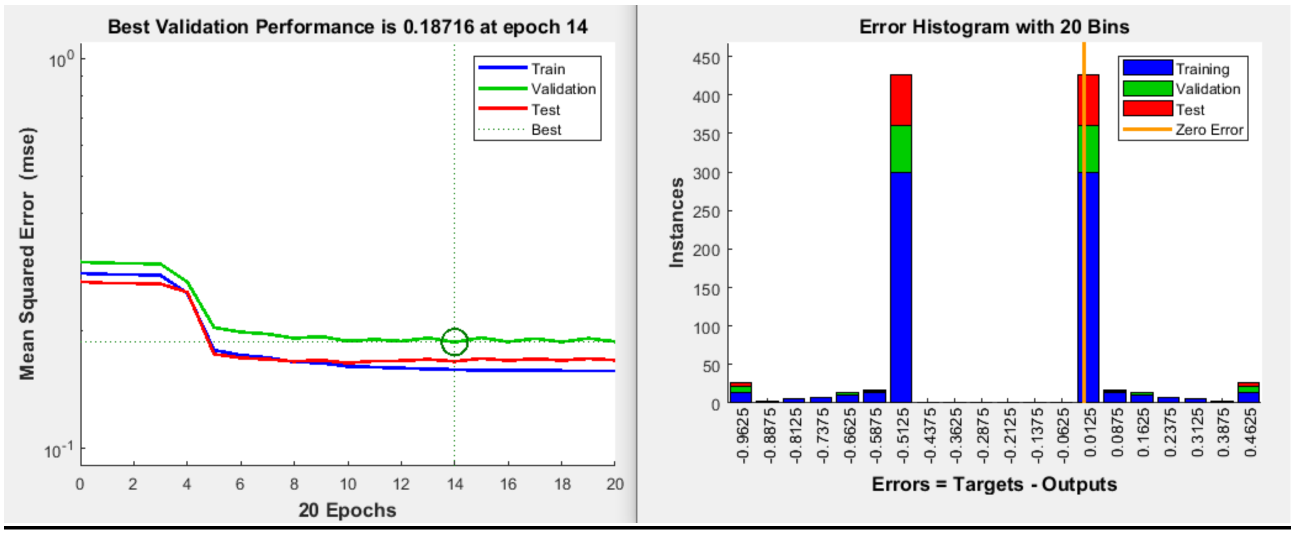


Figure 6.2 *Performance and Error Histogram 1st CASE*

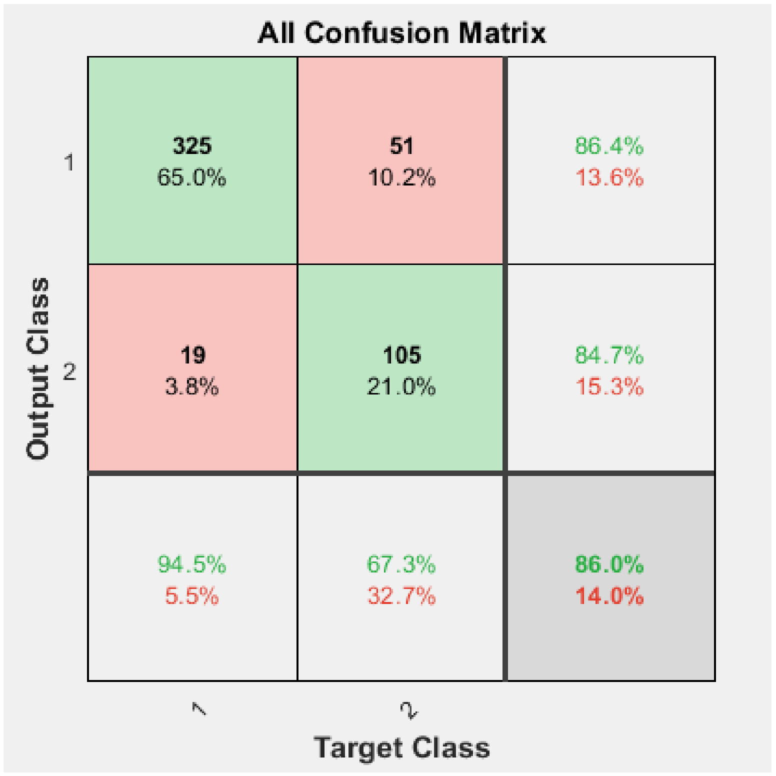


Figure 6.3 *Confusion Matrix 1st CASE*

2nd CASE : Theft with overload 60%-80%

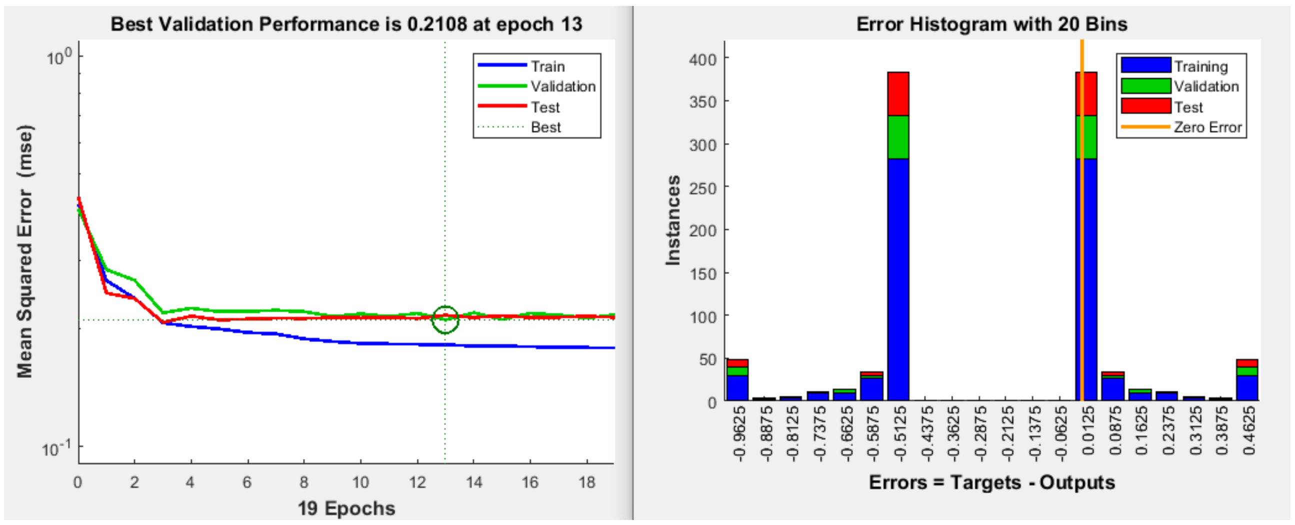


Figure 6.4 *Performance and Error Histogram 2nd CASE*

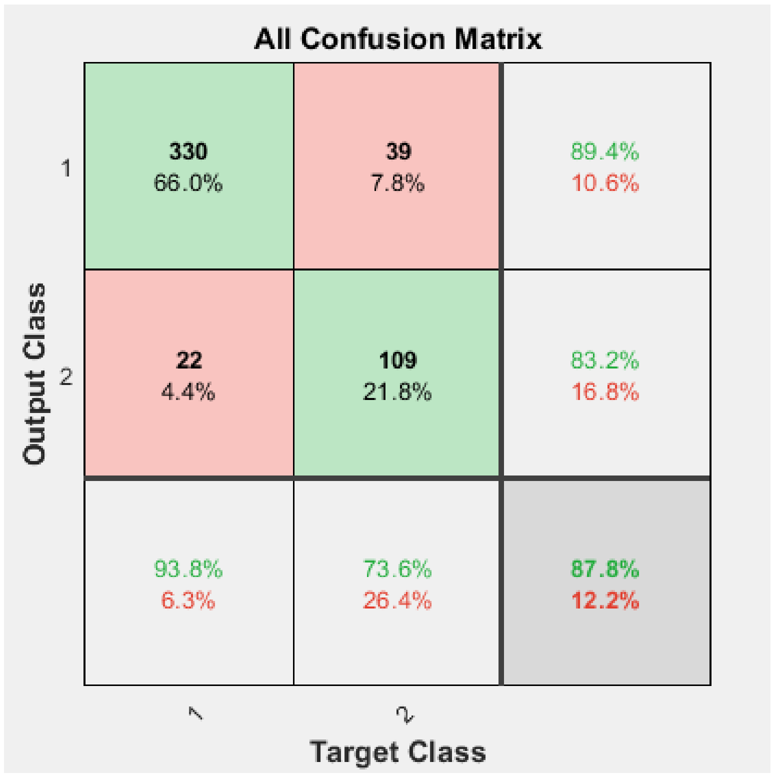


Figure 6.5 *Confusion Matrix 2nd CASE*

3rd CASE : Theft with overload 80%-100%

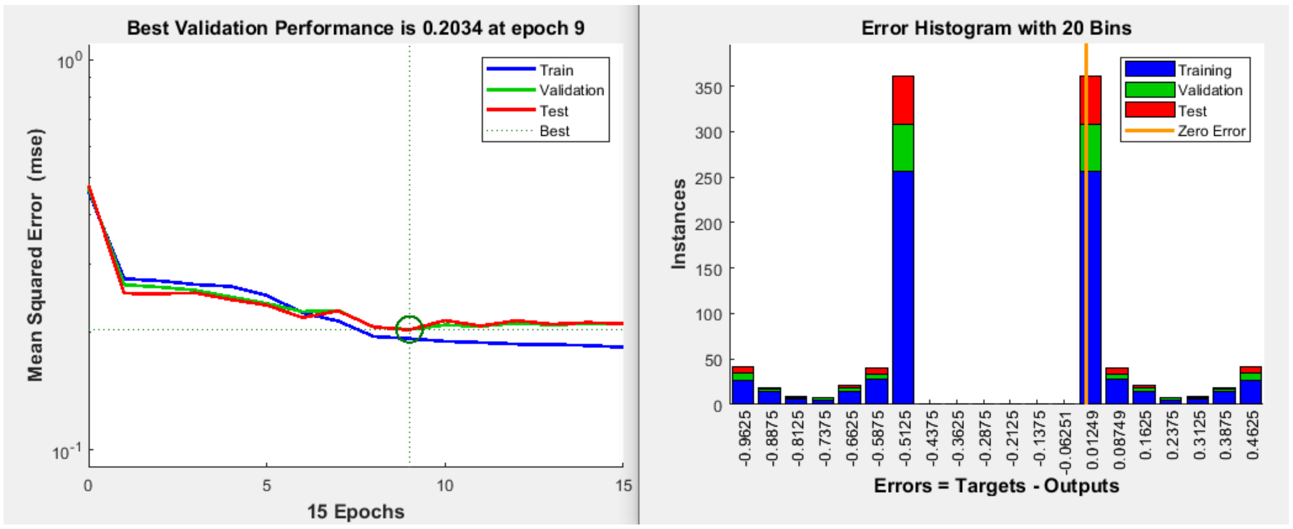


Figure 6.6 *Performance and Error Histogram 3rd CASE*

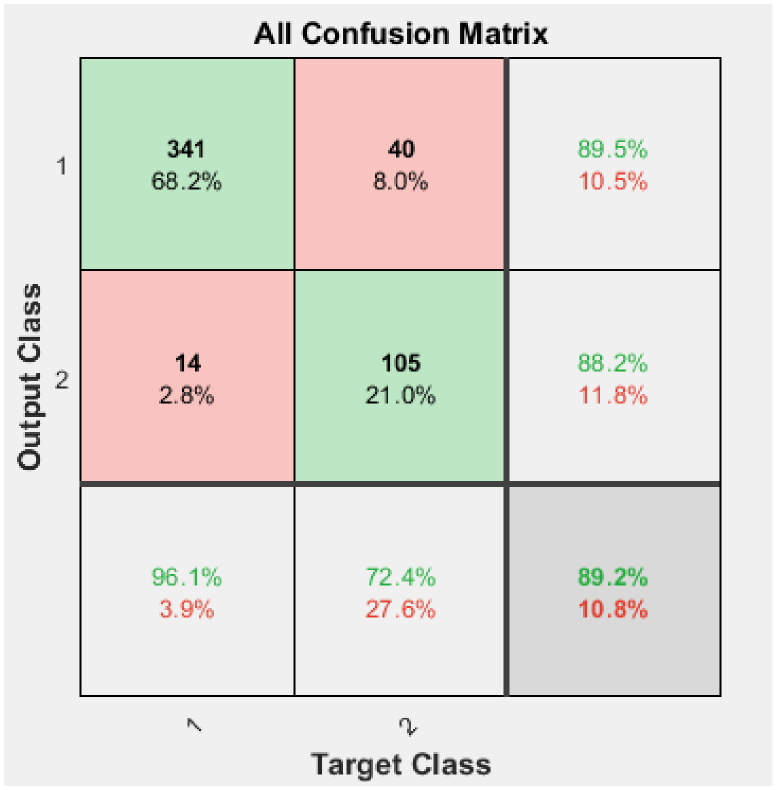


Figure 6.7 *Confusion Matrix 3rd CASE*

4th CASE : Partial theft 30%-50%

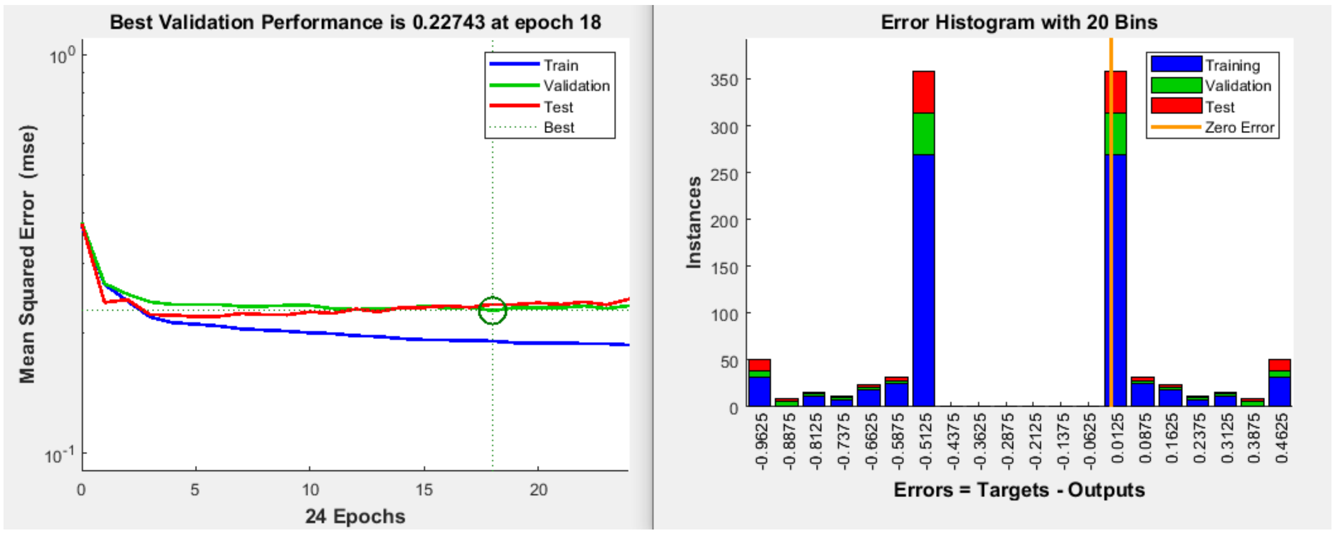


Figure 6.8 *Performance and Error Histogram 4th CASE*

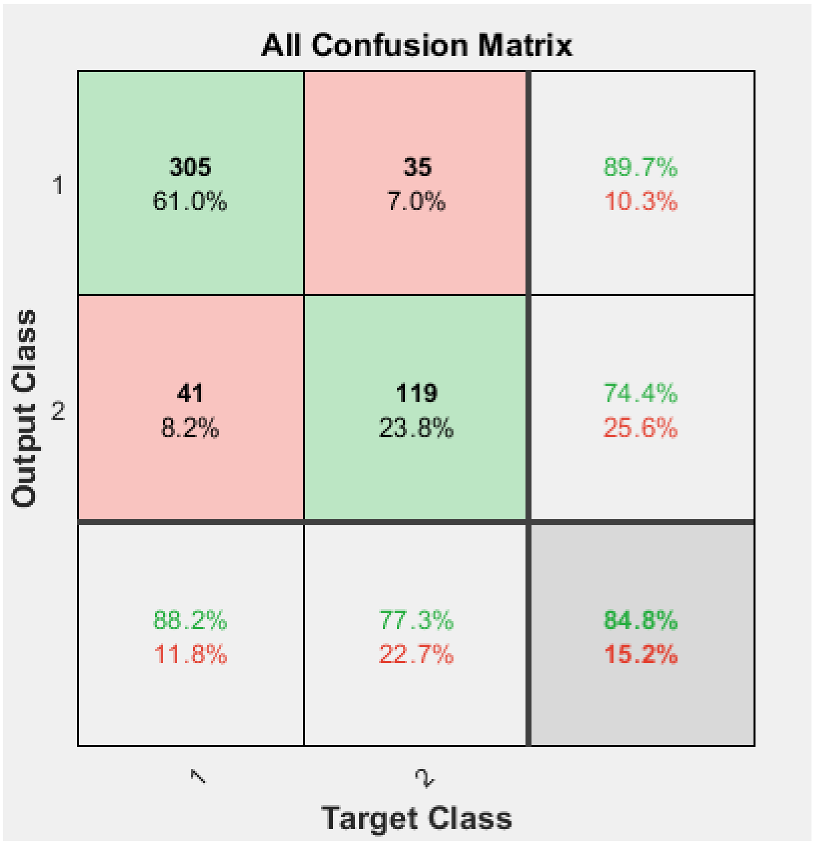


Figure 6.9 *Confusion Matrix 4th CASE*

5th CASE : Partial theft 50%-70%

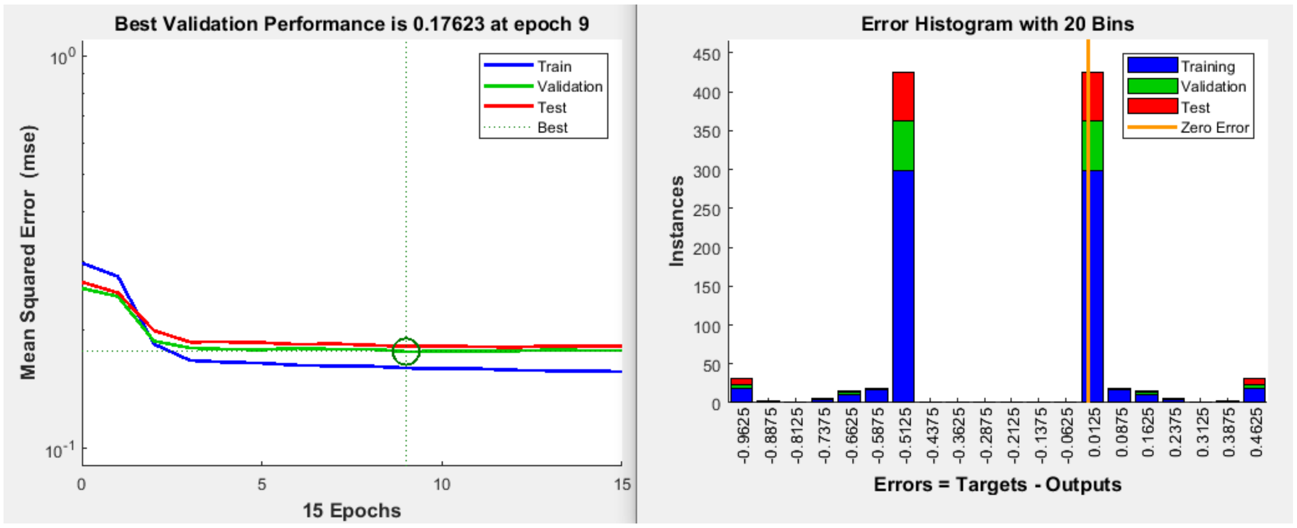


Figure 6.10 *Performance and Error Histogram 5th CASE*

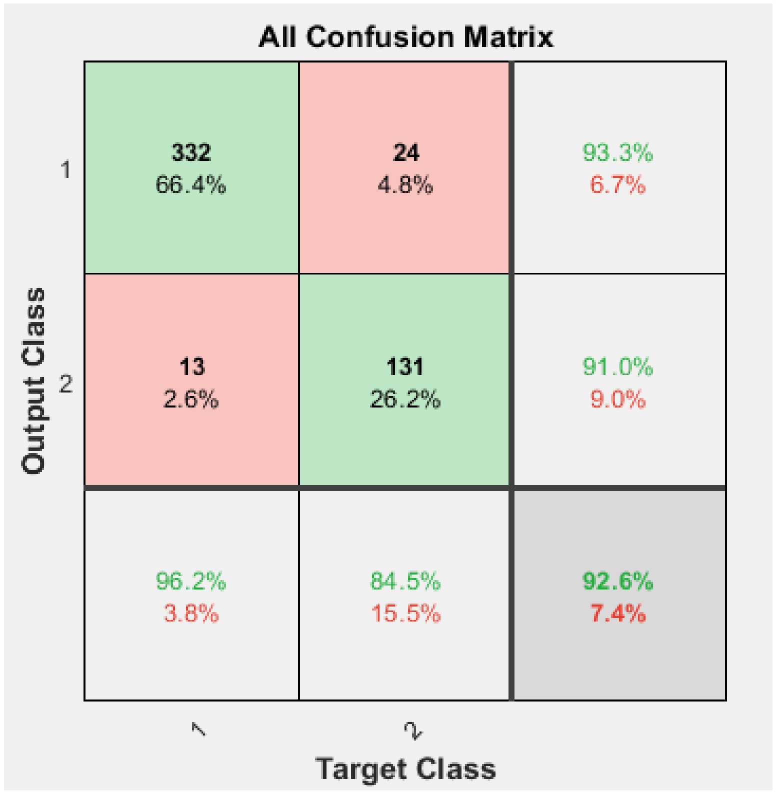


Figure 6.11 *Confusion Matrix 5th CASE*

6th CASE : Theft with overload 60%-80% and partial theft 50%-70%

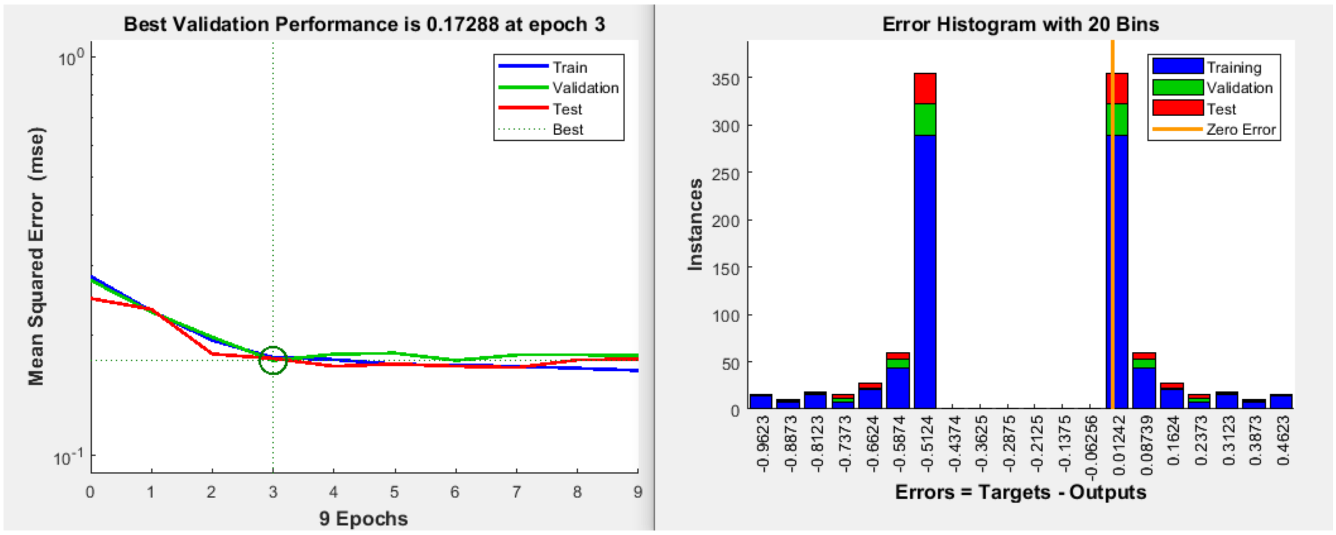
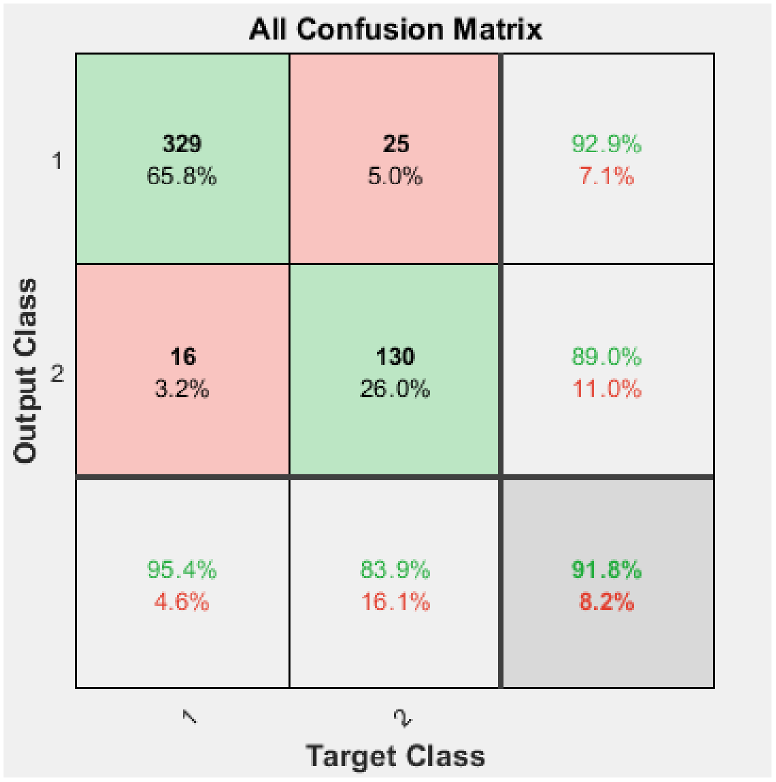


Figure 6.12 *Performance and Error Histogram 6th CASE*



	1	2	
1	**329** 65.8%	**25** 5.0%	92.9% 7.1%
2	**16** 3.2%	**130** 26.0%	89.0% 11.0%
	95.4% 4.6%	83.9% 16.1%	**91.8%** **8.2%**

Figure 6.11 *Confusion Matrix 6th CASE*

The table 6.1 shows some accuracy metrics we computed from the simulations of each case that have been examined. Of course these numbers differentiated with each simulation due to the random initial weights of the neural network. The performance metrics that you can see below represent the average simulations.

CASES	Accuracy	Recall	Precision	F1	True Positives	True Negatives	False Positives	False Negatives	Time (s)
1st Case	86%	0,8644	0,9448	0,9028	325	105	19	51	218
2nd Case	87,8%	0,8799	0,9574	0,9170	337	102	15	46	223
3rd Case	89,2%	0,8950	0,9606	0,9266	341	105	14	40	215
4th Case	84,8%	0,8971	0,8815	0,8892	305	119	41	35	229
5th Case	92,6%	0,9326	0,9623	0,9472	332	131	13	24	236
6th Case	91,8%	0,9294	0,9536	0,9413	329	130	16	25	211

Table 6.1 *Total Performance Metrics*

Generally, this method as you can see from the accuracy metrics in table 6.1, yielded good results. The accuracy metrics ranged from 84%-93% (depending on the case that was tested). However, it has some disadvantages:

- This method is by far the most time consuming method that has been tested in this thesis. A simulation with all the customers (3273) could take up to 35-40 minutes even with the use of principal component analysis, which drastically reduces the time of a full simulation.
- The random weights that the pattern recognition app chooses by default sometimes lead to moderate classifications (we rarely had some simulations with 76-80% accuracy due to that act)

Of course, this method takes advantage of Matlab software in terms of visualization, facility and versatility. As you can see from the images in this subchapter we plotted a lot of things like confusion matrices, error histograms and performance.

In total, in terms of performance this method was one of the best that was tested in this thesis and constitutes one very good method for electricity theft detection.

6. Power Theft Detection with Support Vector Machine Classifiers

Support Vector Machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. In this chapter we used support vector machine classifier in order to classify the consumers of electricity into fraudster or non-fraudster. The support-vector machines classification implemented in Matlab.

6.1 SVM definition

Classifying data is a common task in machine learning. Let’s assume some given data points each belong to one of two classes, and the goal is to decide into which class a new data point will belong to. In the case of support-vector machines, a data point is viewed as a dimensional vector (a list of numbers), and we want to know whether we can separate such points with a dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum margin hyperplane and the linear classifier it defines is known as maximum margin classifier or equivalently, the perception of optimal stability.

More formally, a support-vector machine constructs a hyperplane or set of hyperplanes in a high or infinite-dimensional space, which can be used for classification, regression or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

Whereas the original problem may be stated in a finite-dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products of pairs of input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $k(x,y)$ selected to suit the problem. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant, where such a set of vectors is an orthogonal (and thus minimal) set of vectors that defines a hyperplane. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters of images of feature vectors that occur in the data base. With this choice of a hyperplane, the points in the feature space that are mapped into the hyperplane are defined by the relation:

$$\sum_i \alpha_i k(x_i, x) = \text{constant} \tag{6.1}$$

Note that if *k(x,y)* becomes small as *y* grows further away from *x* , each term in the sum measures the degree of closeness of the test point *x* to the corresponding data base point x_i. In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points *x* mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets that are not convex at all in the original space [47].

6.2 SVM applications

SVMs can be used to solve various real-world problems:

• SVMs are helpful in text and hypertext categorization, as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings. Some methods for shallow semantic parsing are based on support vector machines.

• Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true for image segmentation systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik.

• Hand-written characters can be recognized using SVM.

• The SVM algorithm has been widely applied in Biology and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models. Support-vector machine weights have also been used to interpret SVM models in the past. Posthoc interpretation of support-vector machine models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences [47].

6.3 Classification Procedure

In order to implement an SVM classification we had to follow some steps for achieving better results:

1) We prepared the data loading the struct of datasets with different kinds and percentages of electricty theft. Then we divided the data to train and test set. Specifically, we used 80% of the data as train set and 20% of the data as test set. The selection of the samples for the train/test sets was totally random using Matlab's functions.

2) We prepared validation set out of training set (which will be used later for k-fold CV) with Matlab function called *cvpartition()* which, depending on the arguments we gave, will give a number of sets for the cross validation (we gave 10, so we had 10 sets for cross validation).

3) Then we used *fitcsvm()* function in order to train a support vector machine classifier for two class learning(fraudster or non-fraudster). Afterwards we used *predict()* function for comparing the results of trained model we created with *fitcsvm()* with the actual results and see how wrong the prediction was. The prediction in this situation will not be so good, as we did not choose the best parameters (we executed this step just for training the classifier and used it for the next step to find the best hyperparameters)

4) We used the trained model from the previous step in order to select specific features from the dataset (chose the consumer's samples which could be more useful for the classification) with Matlab's function *sequentialfs()*. The number of the features-columns we are going to extract was depending on the argument we used in the function. In our case we used 2,4 or 10 features-columns of all the dataset, according to the case of electricity theft that we examine. Then using these features that were best for the classification we used *fitcsvm()* again in order to train the model again, but this time with the best parameters or hyperparameters.

5) Finally, using the best parameters for the test set, we compared with predict() function the trained model (created with fitcsvm() in the previous step) with the prediction of the function and the results were greatly improved. We also used matlab's functions *resubLoss()* and *kfoldLoss()* to find the accuracy. Then using some plots functions we visualize the hyperplane with samples from our work. You can see the results in the next chapter.

6.4 Results

As we mentioned in the beginning of chapter 3, the real-time energy consumption measurements we have in our possession is a struct of data with different kinds and percentages of electricity theft in each data table. We tested the performance of our support vector machine classification in each of these cases, as you can see in the images below. The accuracy is representative of the average accuracy in each case. Each case was tested at least 5 times in order to lead to the images below. We tested the svm classification with all customers but the images below are from simulations with 500 consumers for time saving. With 500 consumers a full classification could take approximately 4 minutes but with all the consumers it could take up to 20 minutes.

In the images below we choose to present you the hyperplane and the graph of minimum objective vs function evaluations. The matlab’s command window that you can see, shows the features-columns (2-4 or 10 depending on the case) that *sequentialfs()* function choose from the dataset in each case in order to use it for the classification as well as the optimization process. The 30 iterations (you can see only 20 for space brevity) conclude to best estimated feasible points for BoxConstraint and KernelScale.

For BoxConstraint, the basic idea is that when the data is not perfectly separable, the training algorithm must allow some mis-classification in the training set. In this case it is applying a cost to the misclassification. The higher the box-constraint the higher the cost of the misclassified points, leading to a more strict separation of the data.

The KernelScale is also called gamma. Gamma is related to how spread out the data points are. If they are very far from each other (which would happen in a very high dimensional space for example), then you don't want the kernel to drop off quickly, so you would use a small gamma.

1st CASE : Theft with overload 40%-60%

CASE	Accuracy	Recall	Precision	F1	True Positives	True Negatives	False Positives	False Negatives	Time (s)
1st Case	80%	0,9571	0,7976	0,8701	67	13	17	3	261

Table 6.1 *Performance Metrics*

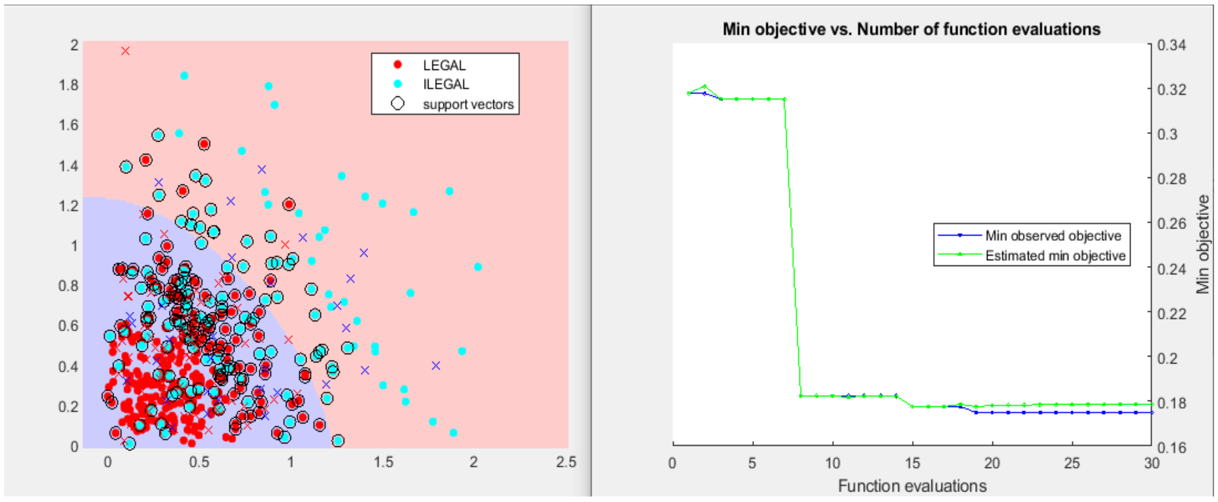
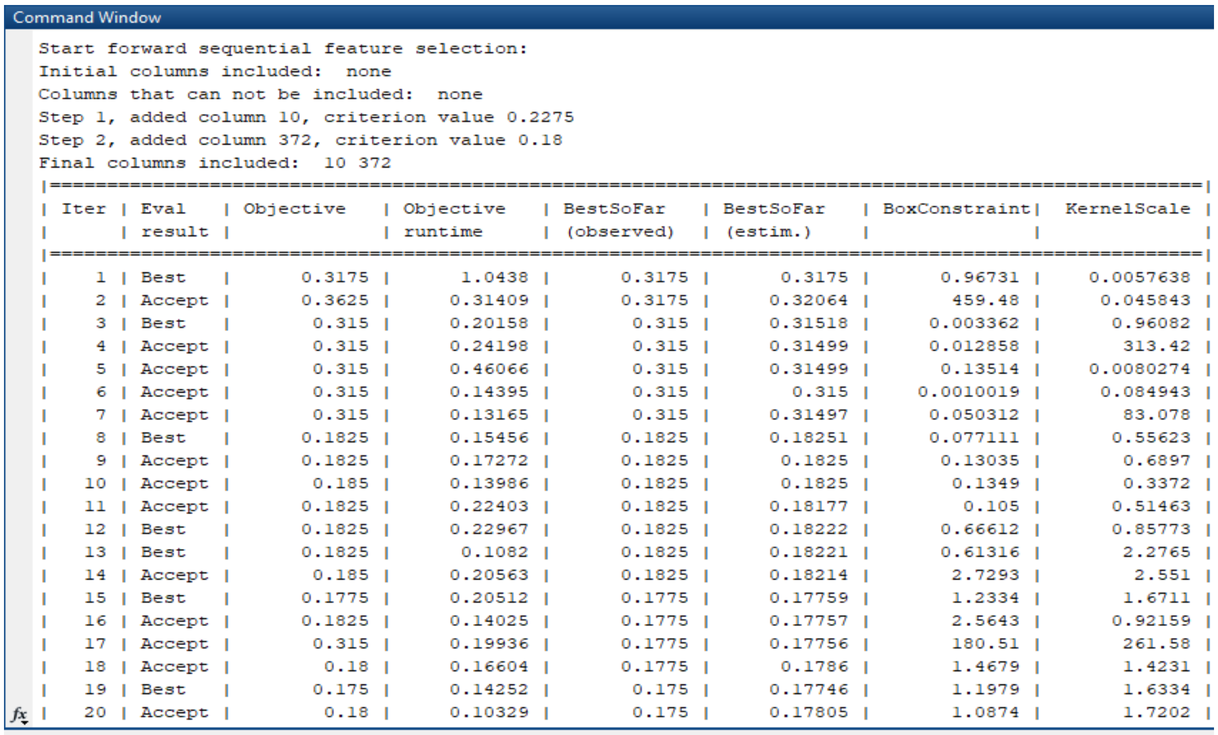


Figure 6.1 *Hyperplane and Function evaluations*



```
Start forward sequential feature selection:
Initial columns included:  none
Columns that can not be included:  none
Step 1, added column 10, criterion value 0.2275
Step 2, added column 372, criterion value 0.18
Final columns included:  10 372
|==========================================================================================================|
| Iter | Eval   | Objective    | Objective    | BestSoFar    | BestSoFar    | BoxConstraint|  KernelScale |
|      | result |              | runtime      | (observed)   | (estim.)     |              |              |
|==========================================================================================================|
|    1 | Best   |       0.3175 |       1.0438 |       0.3175 |       0.3175 |      0.96731 |    0.0057638 |
|    2 | Accept |       0.3625 |      0.31409 |       0.3175 |      0.32064 |       459.48 |     0.045843 |
|    3 | Best   |        0.315 |      0.20158 |        0.315 |      0.31518 |     0.003362 |      0.96082 |
|    4 | Accept |        0.315 |      0.24198 |        0.315 |      0.31499 |     0.012858 |       313.42 |
|    5 | Accept |        0.315 |      0.46066 |        0.315 |      0.31499 |      0.13514 |    0.0080274 |
|    6 | Accept |        0.315 |      0.14395 |        0.315 |        0.315 |    0.0010019 |     0.084943 |
|    7 | Accept |        0.315 |      0.13165 |        0.315 |      0.31497 |     0.050312 |       83.078 |
|    8 | Best   |       0.1825 |      0.15456 |       0.1825 |      0.18251 |     0.077111 |      0.55623 |
|    9 | Accept |       0.1825 |      0.17272 |       0.1825 |       0.1825 |      0.13035 |       0.6897 |
|   10 | Accept |        0.185 |      0.13986 |       0.1825 |       0.1825 |       0.1349 |       0.3372 |
|   11 | Accept |       0.1825 |      0.22403 |       0.1825 |      0.18177 |        0.105 |      0.51463 |
|   12 | Best   |       0.1825 |      0.22967 |       0.1825 |      0.18222 |      0.66612 |      0.85773 |
|   13 | Best   |       0.1825 |       0.1082 |       0.1825 |      0.18221 |      0.61316 |       2.2765 |
|   14 | Accept |        0.185 |      0.20563 |       0.1825 |      0.18214 |       2.7293 |        2.551 |
|   15 | Best   |       0.1775 |      0.20512 |       0.1775 |      0.17759 |       1.2334 |       1.6711 |
|   16 | Accept |       0.1825 |      0.14025 |       0.1775 |      0.17757 |       2.5643 |      0.92159 |
|   17 | Accept |        0.315 |      0.19936 |       0.1775 |      0.17756 |       180.51 |       261.58 |
|   18 | Accept |         0.18 |      0.16604 |       0.1775 |       0.1786 |       1.4679 |       1.4231 |
|   19 | Best   |        0.175 |      0.14252 |        0.175 |      0.17746 |       1.1979 |       1.6334 |
|   20 | Accept |         0.18 |      0.10329 |        0.175 |      0.17805 |       1.0874 |       1.7202 |
```

Figure 6.2 *Command Window with sequential feature selection and parameters optimization.*

2nd CASE : Theft with overload 60%-80%

CASE	Accuracy	Recall	Precision	F1	True Positives	True Negatives	False Positives	False Negatives	Time (s)
2nd Case	83%	0,9718	0,8214	0,8903	69	14	15	2	276

Table 6.2 *Performance Metrics*

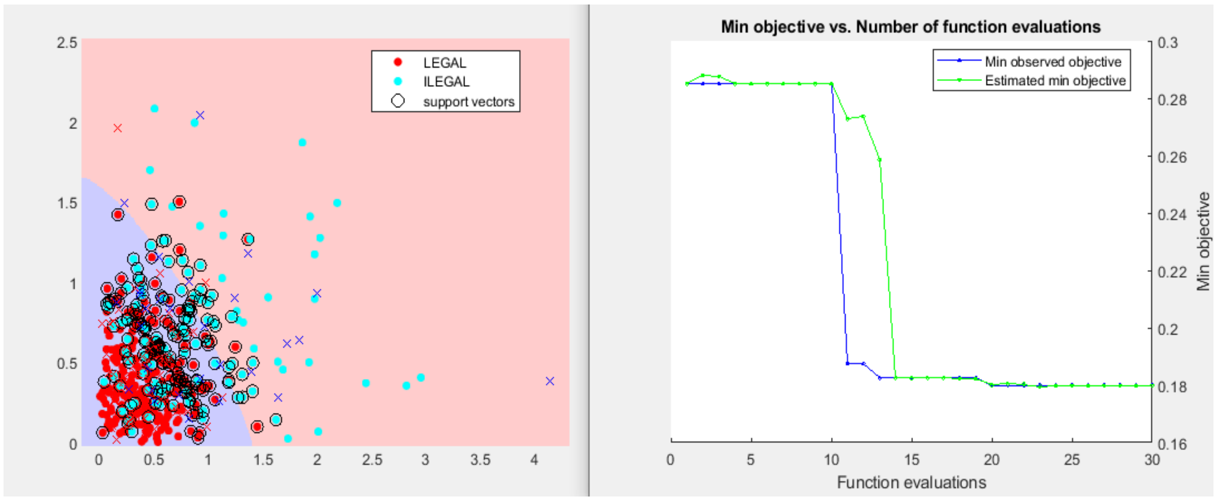
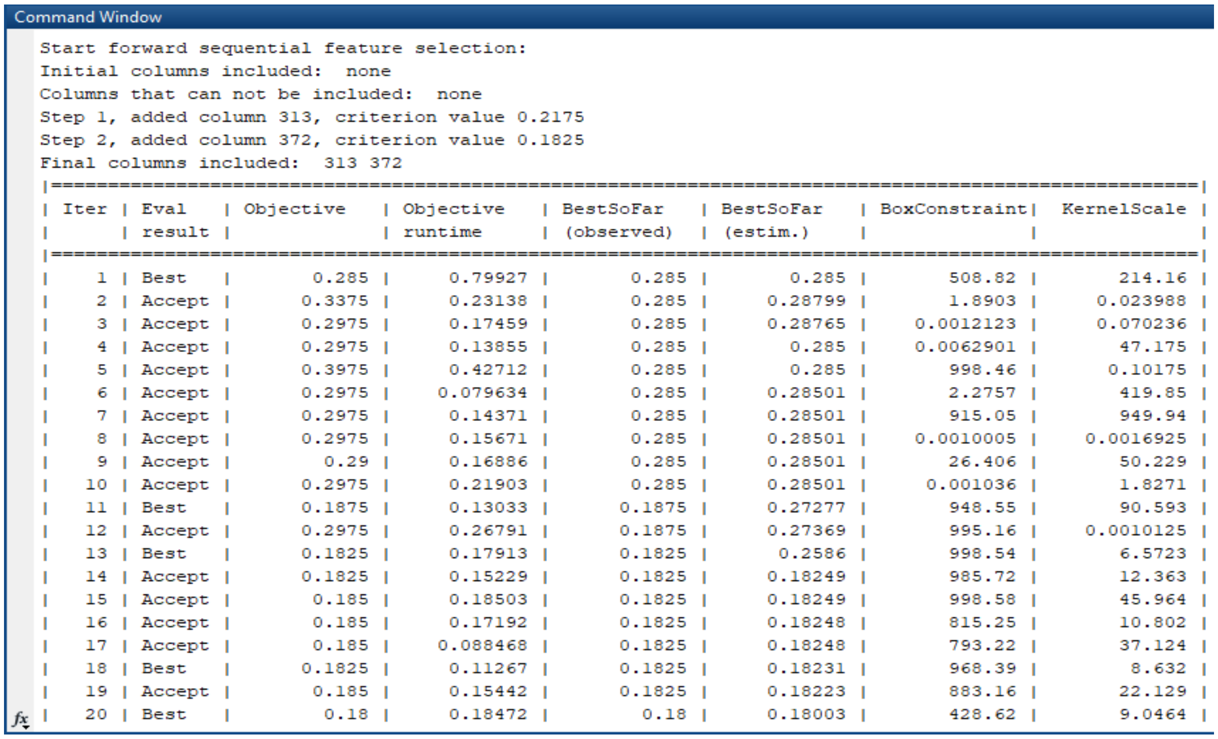


Figure 6.3 *Hyperplane and Function evaluations*



```
Command Window
Start forward sequential feature selection:
Initial columns included:  none
Columns that can not be included:  none
Step 1, added column 313, criterion value 0.2175
Step 2, added column 372, criterion value 0.1825
Final columns included:  313 372
|=====================================================================================================|
| Iter | Eval   | Objective    | Objective    | BestSoFar    | BestSoFar    | BoxConstraint|  KernelScale |
|      | result |              | runtime      | (observed)   | (estim.)     |              |              |
|=====================================================================================================|
|    1 | Best   |        0.285 |      0.79927 |        0.285 |        0.285 |       508.82 |       214.16 |
|    2 | Accept |       0.3375 |      0.23138 |        0.285 |      0.28799 |       1.8903 |     0.023988 |
|    3 | Accept |       0.2975 |      0.17459 |        0.285 |      0.28765 |    0.0012123 |     0.070236 |
|    4 | Accept |       0.2975 |      0.13855 |        0.285 |        0.285 |    0.0062901 |       47.175 |
|    5 | Accept |       0.3975 |      0.42712 |        0.285 |        0.285 |       998.46 |      0.10175 |
|    6 | Accept |       0.2975 |     0.079634 |        0.285 |      0.28501 |       2.2757 |       419.85 |
|    7 | Accept |       0.2975 |      0.14371 |        0.285 |      0.28501 |       915.05 |       949.94 |
|    8 | Accept |       0.2975 |      0.15671 |        0.285 |      0.28501 |    0.0010005 |    0.0016925 |
|    9 | Accept |         0.29 |      0.16886 |        0.285 |      0.28501 |       26.406 |       50.229 |
|   10 | Accept |       0.2975 |      0.21903 |        0.285 |      0.28501 |     0.001036 |       1.8271 |
|   11 | Best   |       0.1875 |      0.13033 |       0.1875 |      0.27277 |       948.55 |       90.593 |
|   12 | Accept |       0.2975 |      0.26791 |       0.1875 |      0.27369 |       995.16 |    0.0010125 |
|   13 | Best   |       0.1825 |      0.17913 |       0.1825 |       0.2586 |       998.54 |       6.5723 |
|   14 | Accept |       0.1825 |      0.15229 |       0.1825 |      0.18249 |       985.72 |       12.363 |
|   15 | Accept |        0.185 |      0.18503 |       0.1825 |      0.18249 |       998.58 |       45.964 |
|   16 | Accept |        0.185 |      0.17192 |       0.1825 |      0.18248 |       815.25 |       10.802 |
|   17 | Accept |        0.185 |     0.088468 |       0.1825 |      0.18248 |       793.22 |       37.124 |
|   18 | Best   |       0.1825 |      0.11267 |       0.1825 |      0.18231 |       968.39 |        8.632 |
|   19 | Accept |        0.185 |      0.15442 |       0.1825 |      0.18223 |       883.16 |       22.129 |
|   20 | Best   |         0.18 |      0.18472 |         0.18 |      0.18003 |       428.62 |       9.0464 |
```

Figure 6.4 *Command Window with sequential feature selection and parameters optimization.*

3rd CASE : Theft with overload 80%-100%

CASE	Accuracy	Recall	Precision	F1	True Positives	True Negatives	False Positives	False Negatives	Time (s)
3rd Case	85%	0,9594	0,8554	0,9006	71	14	12	3	293

Table 6.3 *Performance Metrics*

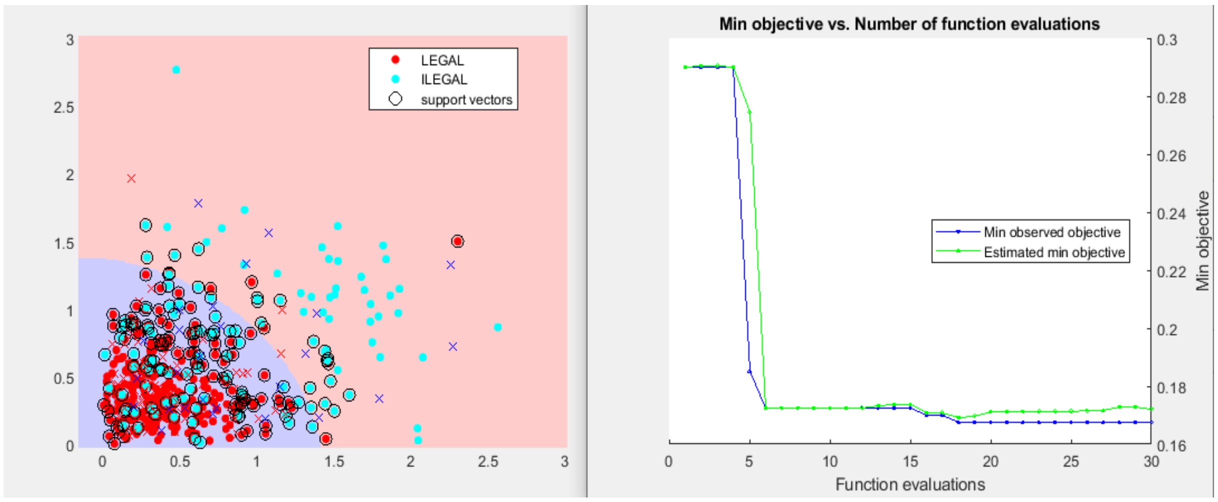
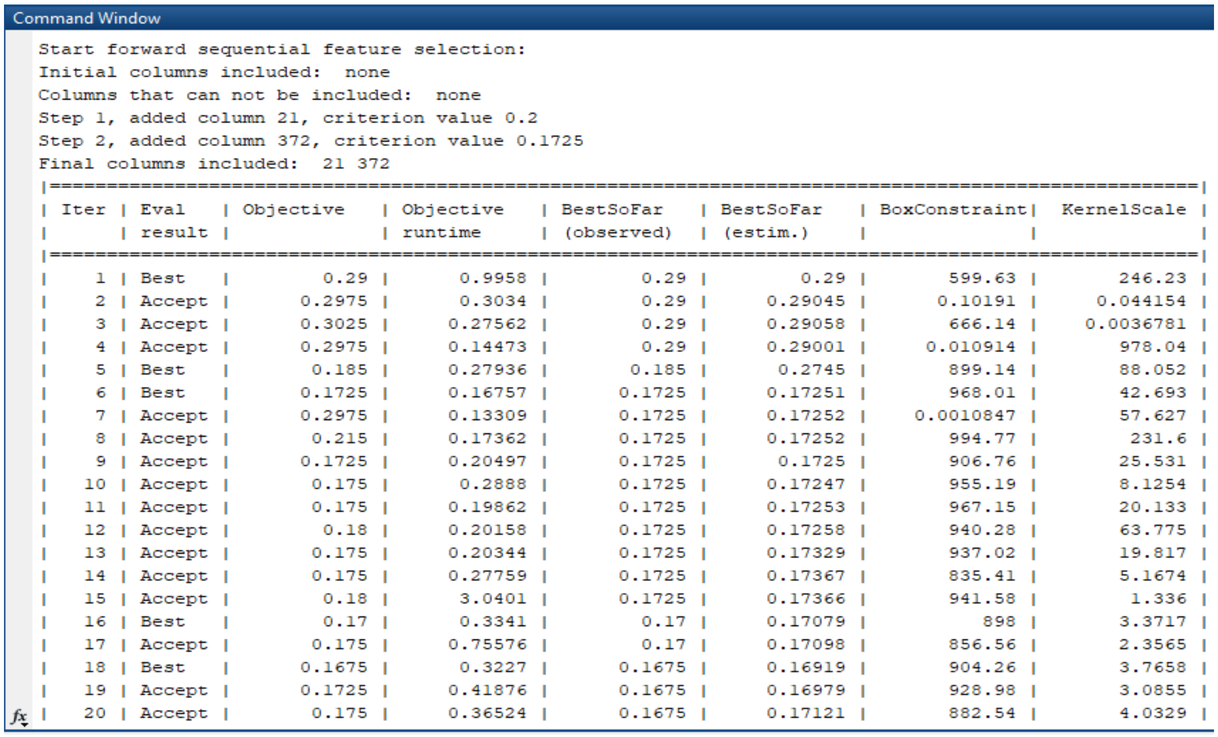


Figure 6.5 *Hyperplane and Function evaluations*



```
Command Window
Start forward sequential feature selection:
Initial columns included:  none
Columns that can not be included:  none
Step 1, added column 21, criterion value 0.2
Step 2, added column 372, criterion value 0.1725
Final columns included:  21 372
|====================================================================================================|
| Iter | Eval   | Objective    | Objective    | BestSoFar    | BestSoFar    | BoxConstraint|  KernelScale |
|      | result |              | runtime      | (observed)   | (estim.)     |              |              |
|====================================================================================================|
|    1 | Best   |         0.29 |       0.9958 |         0.29 |         0.29 |       599.63 |       246.23 |
|    2 | Accept |       0.2975 |       0.3034 |         0.29 |      0.29045 |      0.10191 |     0.044154 |
|    3 | Accept |       0.3025 |      0.27562 |         0.29 |      0.29058 |       666.14 |    0.0036781 |
|    4 | Accept |       0.2975 |      0.14473 |         0.29 |      0.29001 |     0.010914 |       978.04 |
|    5 | Best   |        0.185 |      0.27936 |        0.185 |       0.2745 |       899.14 |       88.052 |
|    6 | Best   |       0.1725 |      0.16757 |       0.1725 |      0.17251 |       968.01 |       42.693 |
|    7 | Accept |       0.2975 |      0.13309 |       0.1725 |      0.17252 |    0.0010847 |       57.627 |
|    8 | Accept |        0.215 |      0.17362 |       0.1725 |      0.17252 |       994.77 |        231.6 |
|    9 | Accept |       0.1725 |      0.20497 |       0.1725 |       0.1725 |       906.76 |       25.531 |
|   10 | Accept |        0.175 |       0.2888 |       0.1725 |      0.17247 |       955.19 |       8.1254 |
|   11 | Accept |        0.175 |      0.19862 |       0.1725 |      0.17253 |       967.15 |       20.133 |
|   12 | Accept |         0.18 |      0.20158 |       0.1725 |      0.17258 |       940.28 |       63.775 |
|   13 | Accept |        0.175 |      0.20344 |       0.1725 |      0.17329 |       937.02 |       19.817 |
|   14 | Accept |        0.175 |      0.27759 |       0.1725 |      0.17367 |       835.41 |       5.1674 |
|   15 | Accept |         0.18 |       3.0401 |       0.1725 |      0.17366 |       941.58 |        1.336 |
|   16 | Best   |         0.17 |       0.3341 |         0.17 |      0.17079 |          898 |       3.3717 |
|   17 | Accept |        0.175 |      0.75576 |         0.17 |      0.17098 |       856.56 |       2.3565 |
|   18 | Best   |       0.1675 |       0.3227 |       0.1675 |      0.16919 |       904.26 |       3.7658 |
|   19 | Accept |       0.1725 |      0.41876 |       0.1675 |      0.16979 |       928.98 |       3.0855 |
|   20 | Accept |        0.175 |      0.36524 |       0.1675 |      0.17121 |       882.54 |       4.0329 |
```

Figure 6.6 *Command Window with sequential feature selection and parameters optimization.*

4th CASE : Partial theft 30%-50%

CASE	Accuracy	Recall	Precision	F1	True Positives	True Negatives	False Positives	False Negatives	Time (s)
4th Case	79%	0,9305	0,8072	0,8644	67	12	16	5	242

Table 6.4 *Performance Metrics*

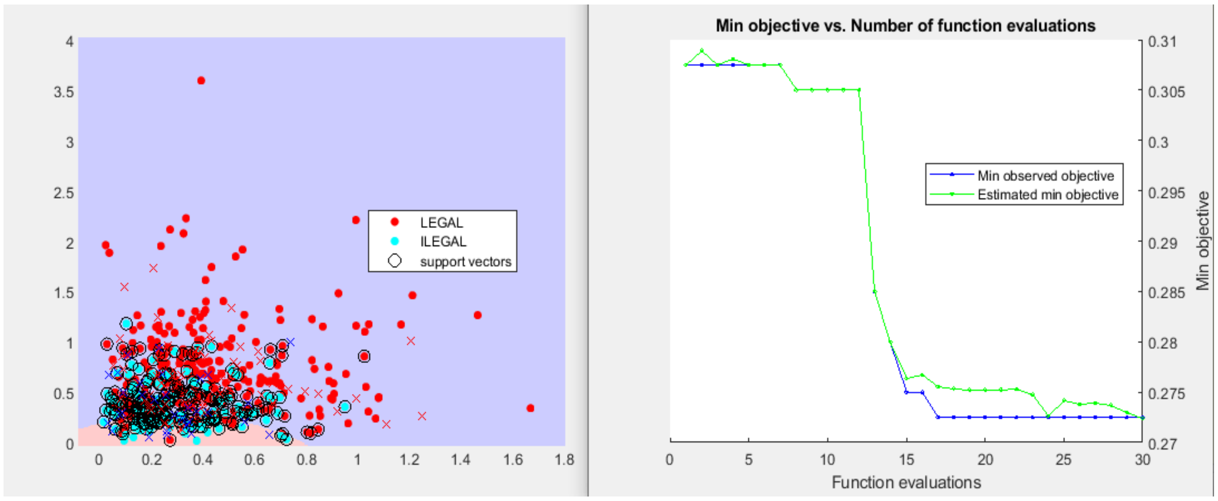
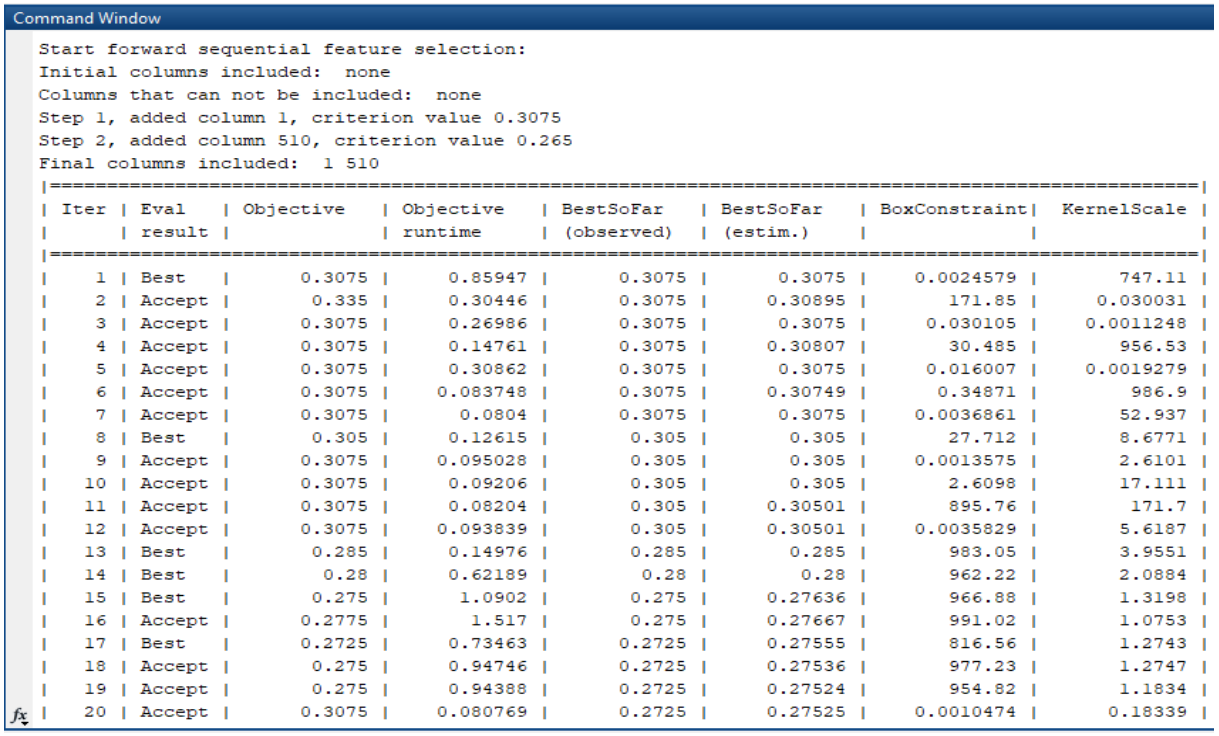


Figure 6.7 *Hyperplane and Function evaluations*



```
Start forward sequential feature selection:
Initial columns included:  none
Columns that can not be included:  none
Step 1, added column 1, criterion value 0.3075
Step 2, added column 510, criterion value 0.265
Final columns included:  1 510
|====================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   | BoxConstraint|  KernelScale |
|      | result |             | runtime     | (observed)  | (estim.)    |              |              |
|====================================================================================================|
|    1 | Best   |      0.3075 |     0.85947 |      0.3075 |      0.3075 |    0.0024579 |       747.11 |
|    2 | Accept |       0.335 |     0.30446 |      0.3075 |     0.30895 |       171.85 |     0.030031 |
|    3 | Accept |      0.3075 |     0.26986 |      0.3075 |      0.3075 |     0.030105 |    0.0011248 |
|    4 | Accept |      0.3075 |     0.14761 |      0.3075 |     0.30807 |       30.485 |       956.53 |
|    5 | Accept |      0.3075 |     0.30862 |      0.3075 |      0.3075 |     0.016007 |    0.0019279 |
|    6 | Accept |      0.3075 |    0.083748 |      0.3075 |     0.30749 |      0.34871 |        986.9 |
|    7 | Accept |      0.3075 |      0.0804 |      0.3075 |      0.3075 |    0.0036861 |       52.937 |
|    8 | Best   |       0.305 |     0.12615 |       0.305 |       0.305 |       27.712 |       8.6771 |
|    9 | Accept |      0.3075 |    0.095028 |       0.305 |       0.305 |    0.0013575 |       2.6101 |
|   10 | Accept |      0.3075 |     0.09206 |       0.305 |       0.305 |       2.6098 |       17.111 |
|   11 | Accept |      0.3075 |     0.08204 |       0.305 |     0.30501 |       895.76 |        171.7 |
|   12 | Accept |      0.3075 |    0.093839 |       0.305 |     0.30501 |    0.0035829 |       5.6187 |
|   13 | Best   |       0.285 |     0.14976 |       0.285 |       0.285 |       983.05 |       3.9551 |
|   14 | Best   |        0.28 |     0.62189 |        0.28 |        0.28 |       962.22 |       2.0884 |
|   15 | Best   |       0.275 |      1.0902 |       0.275 |     0.27636 |       966.88 |       1.3198 |
|   16 | Accept |      0.2775 |       1.517 |       0.275 |     0.27667 |       991.02 |       1.0753 |
|   17 | Best   |      0.2725 |     0.73463 |      0.2725 |     0.27555 |       816.56 |       1.2743 |
|   18 | Accept |       0.275 |     0.94746 |      0.2725 |     0.27536 |       977.23 |       1.2747 |
|   19 | Accept |       0.275 |     0.94388 |      0.2725 |     0.27524 |       954.82 |       1.1834 |
|   20 | Accept |      0.3075 |    0.080769 |      0.2725 |     0.27525 |    0.0010474 |      0.18339 |
```

Figure 6.8 *Command Window with sequential feature selection and parameters optimization.*

<u>5th CASE : Partial theft 50%-70%</u>

CASE	Accuracy	Recall	Precision	F1	True Positives	True Negatives	False Positives	False Negatives	Time (s)
5th Case	87%	0,8904	0,9286	0,9091	65	22	5	8	246

***Table 6.5** Performance Metrics*

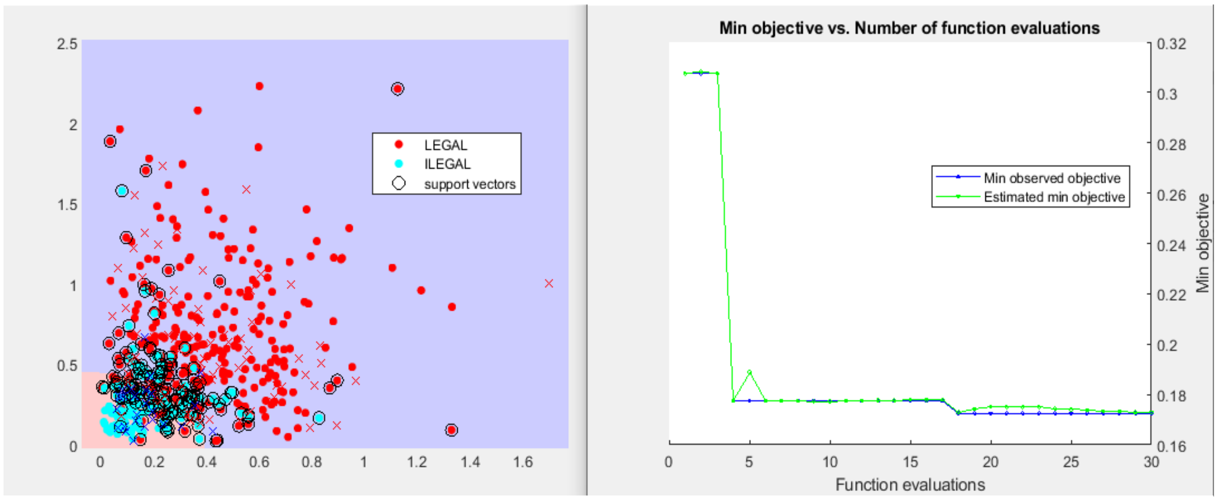


***Figure 6.9** Hyperplane and Function evaluations*

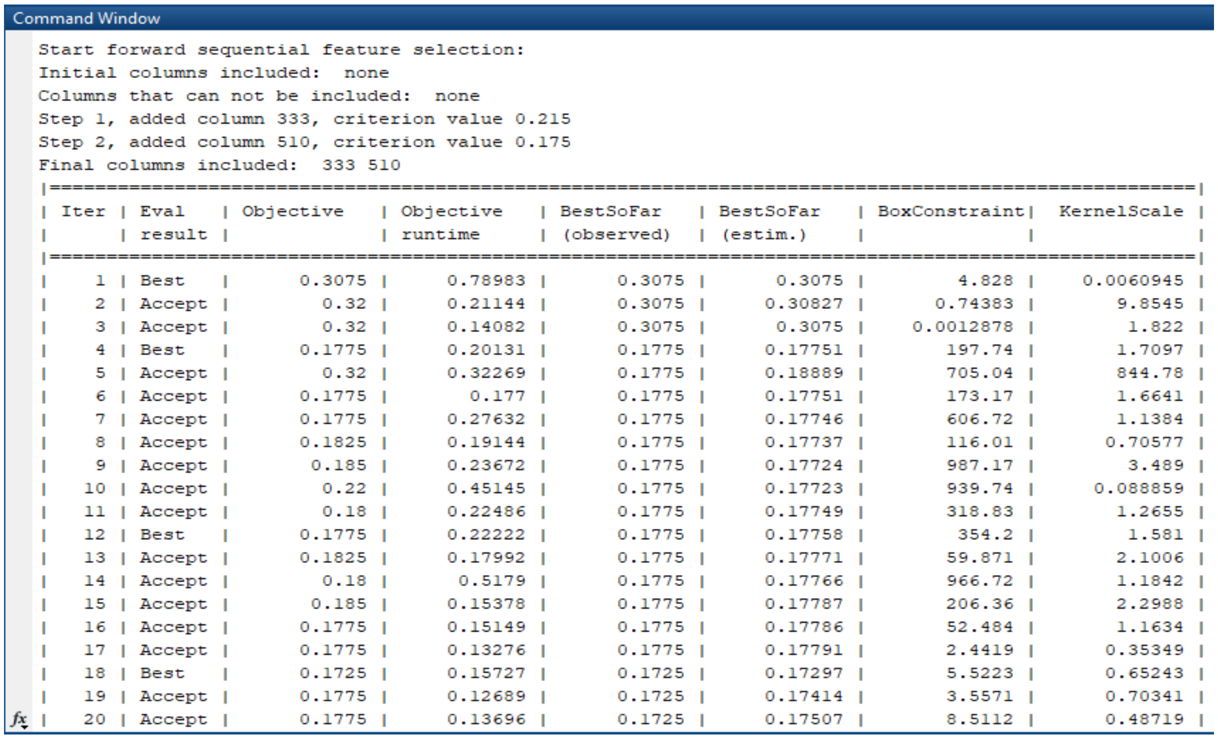

```
Command Window
Start forward sequential feature selection:
Initial columns included:  none
Columns that can not be included:  none
Step 1, added column 333, criterion value 0.215
Step 2, added column 510, criterion value 0.175
Final columns included:  333 510
|===================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   | BoxConstraint|  KernelScale |
|      | result |             | runtime     | (observed)  | (estim.)    |              |              |
|===================================================================================================|
|    1 | Best   |      0.3075 |     0.78983 |      0.3075 |      0.3075 |        4.828 |    0.0060945 |
|    2 | Accept |        0.32 |     0.21144 |      0.3075 |     0.30827 |      0.74383 |       9.8545 |
|    3 | Accept |        0.32 |     0.14082 |      0.3075 |      0.3075 |    0.0012878 |        1.822 |
|    4 | Best   |      0.1775 |     0.20131 |      0.1775 |     0.17751 |       197.74 |       1.7097 |
|    5 | Accept |        0.32 |     0.32269 |      0.1775 |     0.18889 |       705.04 |       844.78 |
|    6 | Accept |      0.1775 |       0.177 |      0.1775 |     0.17751 |       173.17 |       1.6641 |
|    7 | Accept |      0.1775 |     0.27632 |      0.1775 |     0.17746 |       606.72 |       1.1384 |
|    8 | Accept |      0.1825 |     0.19144 |      0.1775 |     0.17737 |       116.01 |      0.70577 |
|    9 | Accept |       0.185 |     0.23672 |      0.1775 |     0.17724 |       987.17 |        3.489 |
|   10 | Accept |        0.22 |     0.45145 |      0.1775 |     0.17723 |       939.74 |     0.088859 |
|   11 | Accept |        0.18 |     0.22486 |      0.1775 |     0.17749 |       318.83 |       1.2655 |
|   12 | Best   |      0.1775 |     0.22222 |      0.1775 |     0.17758 |        354.2 |        1.581 |
|   13 | Accept |      0.1825 |     0.17992 |      0.1775 |     0.17771 |       59.871 |       2.1006 |
|   14 | Accept |        0.18 |      0.5179 |      0.1775 |     0.17766 |       966.72 |       1.1842 |
|   15 | Accept |       0.185 |     0.15378 |      0.1775 |     0.17787 |       206.36 |       2.2988 |
|   16 | Accept |      0.1775 |     0.15149 |      0.1775 |     0.17786 |       52.484 |       1.1634 |
|   17 | Accept |      0.1775 |     0.13276 |      0.1775 |     0.17791 |       2.4419 |      0.35349 |
|   18 | Best   |      0.1725 |     0.15727 |      0.1725 |     0.17297 |       5.5223 |      0.65243 |
|   19 | Accept |      0.1775 |     0.12689 |      0.1725 |     0.17414 |       3.5571 |      0.70341 |
|   20 | Accept |      0.1775 |     0.13696 |      0.1725 |     0.17507 |       8.5112 |      0.48719 |
```

***Figure 6.10** Command Window with sequential feature selection and parameters optimization.*

6th CASE : Theft with overload 60%-80% and partial theft 50%-70%

CASE	Accuracy	Recall	Precision	F1	True Positives	True Negatives	False Positives	False Negatives	Time (s)
6th Case	85%	0,9315	0,8718	0,9007	68	17	10	5	298

***Table 6.6** Performance Metrics*

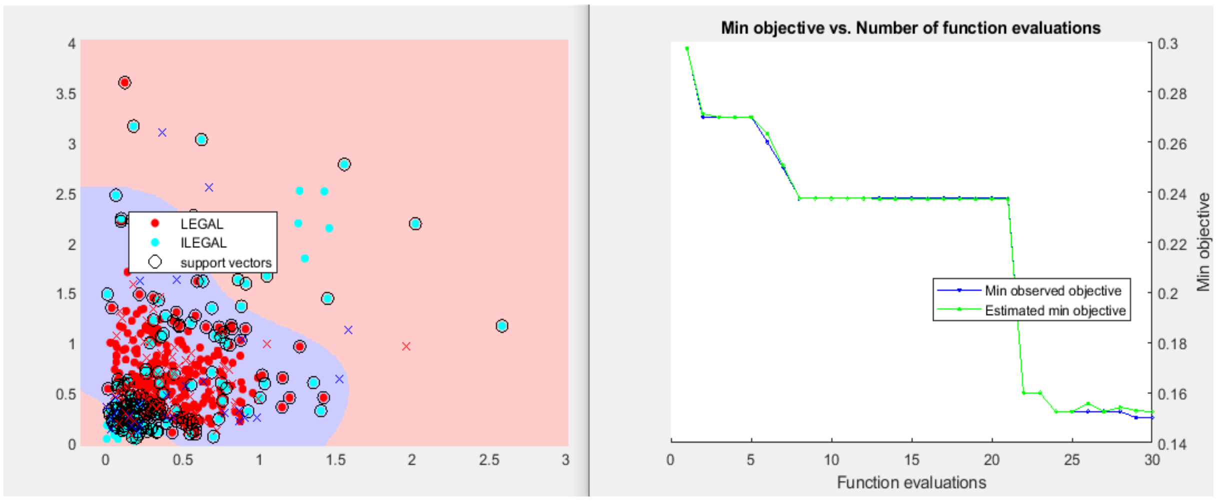
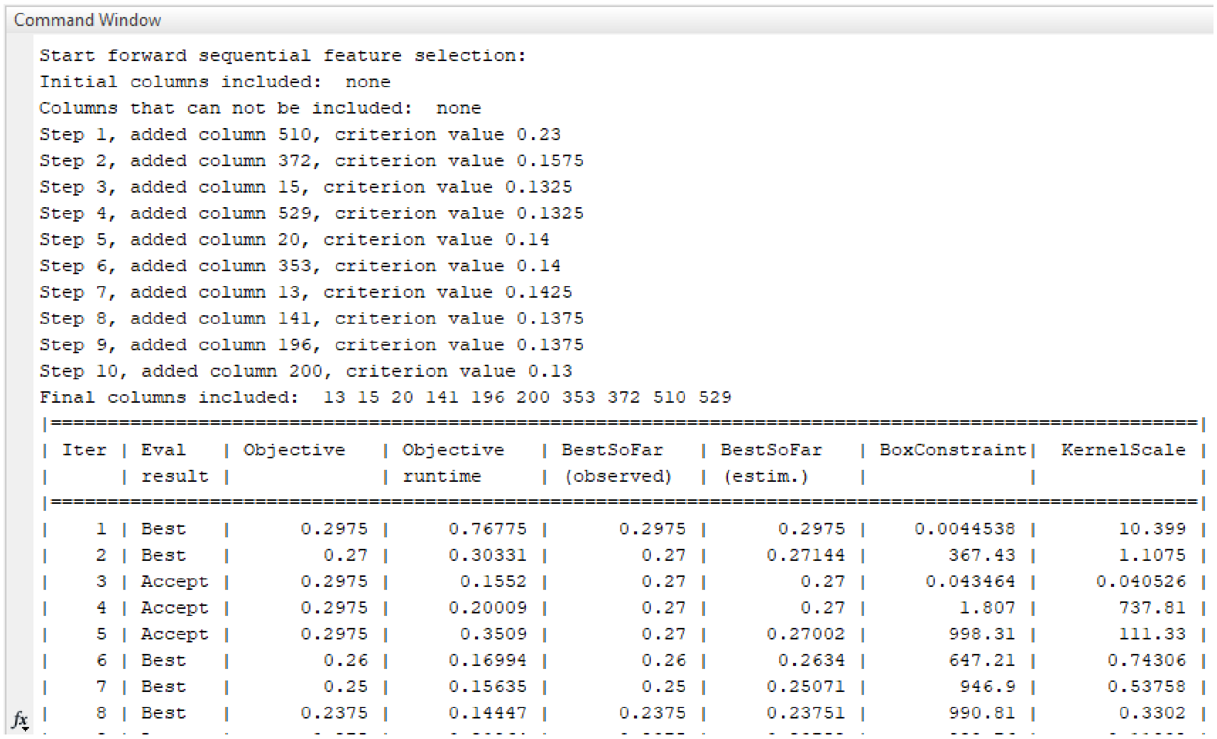


***Figure 6.11** Hyperplane and Function evaluations*



```
Start forward sequential feature selection:
Initial columns included:  none
Columns that can not be included:  none
Step 1, added column 510, criterion value 0.23
Step 2, added column 372, criterion value 0.1575
Step 3, added column 15, criterion value 0.1325
Step 4, added column 529, criterion value 0.1325
Step 5, added column 20, criterion value 0.14
Step 6, added column 353, criterion value 0.14
Step 7, added column 13, criterion value 0.1425
Step 8, added column 141, criterion value 0.1375
Step 9, added column 196, criterion value 0.1375
Step 10, added column 200, criterion value 0.13
Final columns included:  13 15 20 141 196 200 353 372 510 529
|====================================================================================================|
| Iter | Eval   | Objective    | Objective    | BestSoFar    | BestSoFar    | BoxConstraint|  KernelScale |
|      | result |              | runtime      | (observed)   | (estim.)     |              |              |
|====================================================================================================|
|    1 | Best   |       0.2975 |      0.76775 |       0.2975 |       0.2975 |    0.0044538 |       10.399 |
|    2 | Best   |         0.27 |      0.30331 |         0.27 |      0.27144 |       367.43 |       1.1075 |
|    3 | Accept |       0.2975 |       0.1552 |         0.27 |         0.27 |     0.043464 |     0.040526 |
|    4 | Accept |       0.2975 |      0.20009 |         0.27 |         0.27 |        1.807 |       737.81 |
|    5 | Accept |       0.2975 |       0.3509 |         0.27 |      0.27002 |       998.31 |       111.33 |
|    6 | Best   |         0.26 |      0.16994 |         0.26 |       0.2634 |       647.21 |      0.74306 |
|    7 | Best   |         0.25 |      0.15635 |         0.25 |      0.25071 |        946.9 |      0.53758 |
|    8 | Best   |       0.2375 |      0.14447 |       0.2375 |      0.23751 |       990.81 |       0.3302 |
```

***Figure 6.12** Command Window with sequential feature selection and parameters optimization.*

The table 6.7 shows the accuracy metrics we computed from the simulations of each case that have been examined. Of course these numbers differentiated slightly with each simulation. The performance metrics that you can see below represent the average simulations.

CASES	Accuracy	Recall	Precision	F1	True Positives	True Negatives	False Positives	False Negatives	Time (s)
1st Case	80%	0,9571	0,7976	0,8701	67	13	17	3	261
2nd Case	83%	0,9718	0,8214	0,8903	69	14	15	2	276
3rd Case	85%	0,9594	0,8554	0,9006	71	14	12	3	293
4th Case	79%	0,9305	0,8072	0,8644	67	12	16	5	242
5th Case	87%	0,8904	0,9286	0,9091	65	22	5	8	246
6th Case	85%	0,9315	0,8718	0,9007	68	17	10	5	298

Table 6.7 *Total Performance Metrics*

Support Vector Machines can produce accurate and robust classification results on a sound theoretical basis, even when input data are non-monotone and non-linearly separable. So they can help to evaluate more relevant information in a convenient way. Since they linearize data on an implicit basis by means of kernel transformation, the accuracy of results does not rely on the quality of human expertise judgement for the optimal choice of the linearization function of non-linear input data. For these reasons SVMs are regarded as a useful tool for effectively complementing the information gained from classical linear classification techniques.

SVM and Artificial Neural Networks are two popular strategies for supervised machine learning and classification. SVM benefits depend on a particular project. The SVM classifier is a great choice for unbalanced data. As a cost-sensitive classifier it can solve the problem of unbalanced data. All other benefits really depend on the domain and task. But even if the number of positive and negative examples are not similar SVM would work fine if we normalize the data or may be projected into the space of the decision boundary which separates the two classes. SVM quite good compared to other classifiers as the computational complexity is reduced and classification efficiency is increased when compared to any other non linear classifier [52].

As you can see from the performance metrics above the support vector machine classification achieved satisfying results but not as good as the previous two methods that have been tested. Specifically, in the 1st and the 4th case the performance metrics dropped due to the fact that in these cases the consumers’s samples are very similar to each other and the classifier could not distinguish easily the fraudster from the non-fraudster. If the data was less balanced the results would probably be better as most of 2-class classifier works better that way, but generally the support vector machine classification was not so bad for the detection of electricity theft.

7. Power Theft Detection with Optimum Path Forest Classifiers

Optimum Path Forest is another interesting framework that leads to a very powerful tool for pattern recognition classification with graph-based methods. Basically, such methods model the machine learning task as a problem formulated in the graph theory: the dataset samples, which are represented by their corresponding feature vectors, are the graph nodes, that are further connected by an adjacency relation. Without loss of generality, a graph-based method aims at removing or adding edges using some heuristic in order to create connected components, which stand for a group of samples that share some similar characteristics [32].

In our work the optimum-path forest (OPF) clustering algorithm has been employed to identify irregular and regular profiles of consumers obtained from a Irish electrical power company. The optimum path forest classification has been implemented with the use of LibOPF library in a Linux terminal.

7.1 OPF Classifier

Optimum Path Forest (OPF) is a new framework for graph-based pattern recognition, which addresses the graph partition task as a competition process among some key (prototype) samples in order to conquer the remaining nodes according to a path-cost function. The idea is based on the Image Foresting Transform (IFT), which works similarly to OPF, but in the context of designing image processing-like operators. Both OPF and IFT follow the idea of the ordered communities formation, in which an individual(node) will belong to the community (cluster) that gives him/her the best reward (path-cost function value).

Generally, this is how optimum a path forest classifier works: Let Z1 be training set, Z2 evaluation set and Z3 test set with |Z1|, |Z2| and |Z3| samples of a given dataset. This division of the dataset is necessary to validate the classifier and evaluate its learning capacity from the errors. Z1 is used to project the classifier and Z3 kept unseen during the project. A pseudo-test on Z2 is used to teach the classifier by randomly interchanging samples of Z1 with misclassified samples of Z2. After learning, an improvement in accuracy on Z3 is expected . Let $\lambda(s)$ be the function that assigns the correct label i, i=1, 2,…,c, to any sample $s \in Z1 \cup Z2 \cup Z3$, $S \subset Z1$ be a set of prototypes from all classes, and v be an algorithm which extracts n features from any sample $s \in Z1 \cup Z2 \cup Z3$ and returns a vector v(s) r . The distance $d(s,t) \geq 0$ between two samples, s and t, is the one between their corresponding feature vectors v(s) r and v(t) r . One can use any distance function suitable for the extracted features. The most common is the Euclidean one. Our problem consists of projecting a classifier which can predict the correct label $\lambda(s)$ of any sample $s \in Z3$. Training consists of finding a special set $S^* \subset Z1$ of prototypes and a discrete optimal partition of Z1 in the feature space (i.e., an optimum-path forest rooted in S*). The classification of a sample $s \in Z3$ (or $s \in Z2$) is done by evaluating the optimum paths incrementally, as though it were part of the forest, and assigning to it the label of the most strongly connected prototype [32].

7.2 Datasets extraction

First of all, we had to convert the datasets with electricity measurements to the file format that the LibOPF demand. Specifically, the original dataset and its parts training, evaluation and test sets must be in the following BINARY file format:

```
<# of samples> <# of labels> <# of features>
<0> <label> <feature 1 from element 0> <feature 2 from element 0> ...
<1> <label> <feature 1 from element 1> <feature 2 from element 1> ...
.
.
<i> <label> <feature 1 from element i> <feature 2 from element i> ...
<i+1> <label> <feature 1 from element i+1> <feature 2 from element i+1> ...
.
.
<n-1> <label> <feature 1 from element n-1> <feature 2 from element n-1> ...
```

The first line of the file, <# of samples> <# of labels> <# of features> , contains, respectively,
the dataset size, the number of labels (classes) and the number of features in the feature vectors.
The first number of each line, <0>, <1>, ... <n-1>, is a sample identifier (for n samples in the dataset), which is used in the case of precomputed distances. However, the identifier must be specified anyway. The <label> is the kind of class the samples belongs to and <features> are the samples we used.

So we shaped the datasets into this format in Matlab and exported it as a .txt file. Then using LibOPF tool *txt2opf* we managed to convert our dataset into the binary file format (.dat) that the library needs for reading the datasets as input.

7.3 Classification Procedure

As we mentioned in the beginning of chapter 3, the real energy consumption measurements we have in our possession is a struct of data with different kinds and percentages of electricity theft in each data table. We tested the performance of our optimum path forest classifier in each of this cases as you can see in the images below. The accuracy is representative of the average accuracy in each case. Each case was tested at least 5 times and the results are shown in the images below .We tested the classification of the opf classifer with all the customers we had in our possession (3273) as the classification with the LibOPF Library was very fast even with all the customers. The accuracy metrics, however, were not so good.

LibOPF library has all the tools that are needed for the opf classification with default files and functions which did all the work.

First of all, we used the ‘*opf_split’*, which depending on the arguments we gave , randomly splits the dataset into training, testing and/or evaluation set. It also normalizes the dataset according to our selection (the features are normalized with the following equation: *N_i = (F_i - M_i)/S_i*, where F_i, M_i and S_i are, respectively, the feature i, the average of F_i and the standard deviation of F_i in the dataset.)

Secondly, we used the ‘*opf_train*’, which takes over the training procedure of the classifier (execute the training phase.). The program designs a classifier and outputs it in a file, which is used by '*opf_classify*' for testing the dataset. Instead of ‘opf_train’ we used ‘*opf_learn*’ when the examined datasets was large (thousands of samples). The difference between ‘*opf_train*’ and ‘*opf_learn*’ is that the last learns from the classification errors in the evaluation set without increasing the training set size, and outputs a final classifier in a file, which is used for testing by the program '*opf_classify*' like the ‘*opf_train*’ do.

Thirdly, we used the '*opf_classify*'. This uses the file with the classifier, which ‘*opf_train*’ or ‘*opf_learn*’ exported in the previous step, and executes the test phase by classifying the test set. It outputs a file which contains the predicted labels and will be used for computing the accuracy in the next step

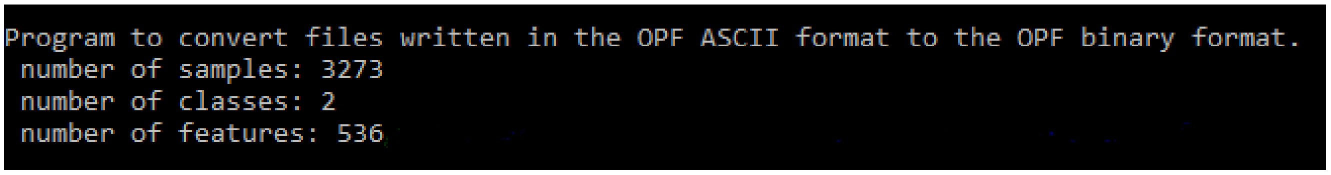
Lastly, we used the *'opf_accuracy'* ,which *is a program to compute the accuracy over training and/or test set. The ‘opf_accuracy’ will look for a classified file, which has been exported from the 'opf_classify'* in order to compute the accuracy of that classification. *We had added some extra code here in 'opf_accuracy'* file in order to compute the other accuracy metrics too.

This is the procedure we follow in order to simulate an opf classification model. Of course we change a lot of things in each tool-file-program in order to set it to the parameters that have the best results.All in all, the opf classification does not have such good results as the other methods but this is something you will see in the next subchapter.

7.4 Results

As we mentioned in the beginning of chapter 3, the real-time energy consumption measurements we have in our possession is a struct of data with different kinds and percentages of electricity theft in each data table. We tested the performance of our optimum path forest classifier in each of these cases, as you can see in the images below. The accuracy is representative of the average accuracy in each case. Each case was tested at least 5 times in order to lead to the images below. This method, just as the method with the FANN Library, is very fast and we didn’t have limitations in the number of consumers. So we used all the consumers (3273) for the opf classification. The results though are not as encouraging as the results of the other methods.

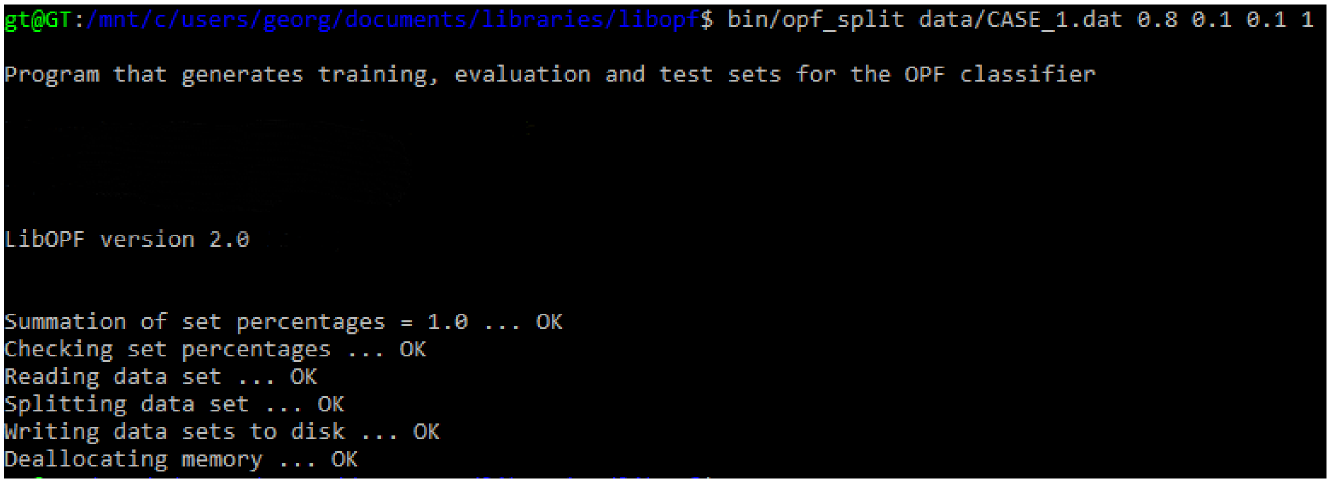
Following the procedure we described in the previous subchapter, we convert the energy consumption data, exported from Matlab, to BINARY file format using the *txt2opf* tool.



```
Program to convert files written in the OPF ASCII format to the OPF binary format.
 number of samples: 3273
 number of classes: 2
 number of features: 536
```

Figure 7.1 *Original file conversion to BINARY file format*

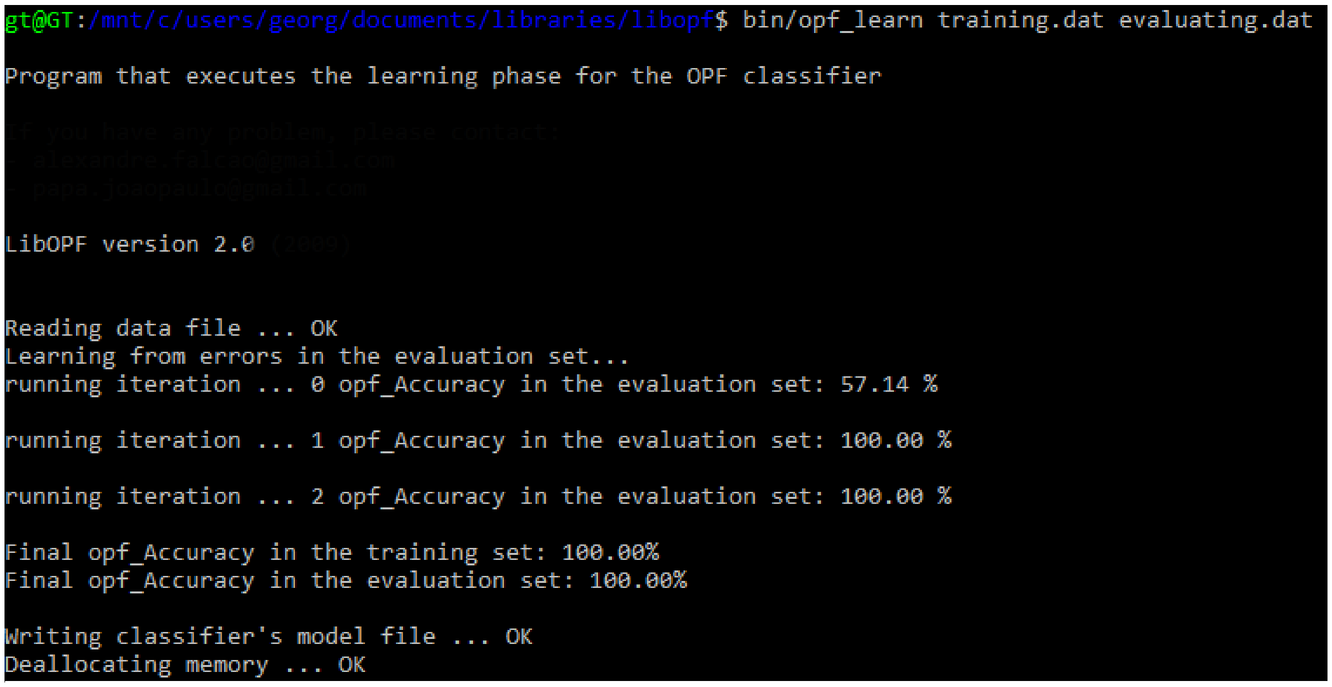
Then, using the *opf_split* tool we split the data set into training, evaluation and test set. We split the dataset as you can see in 80% training,10% testing and 10% evaluating.



```
gt@GT:/mnt/c/users/georg/documents/libraries/libopf$ bin/opf_split data/CASE_1.dat 0.8 0.1 0.1 1

Program that generates training, evaluation and test sets for the OPF classifier



LibOPF version 2.0


Summation of set percentages = 1.0 ... OK
Checking set percentages ... OK
Reading data set ... OK
Splitting data set ... OK
Writing data sets to disk ... OK
Deallocating memory ... OK
```

Figure 7.2 *Data division to training, evaluation and test sets*

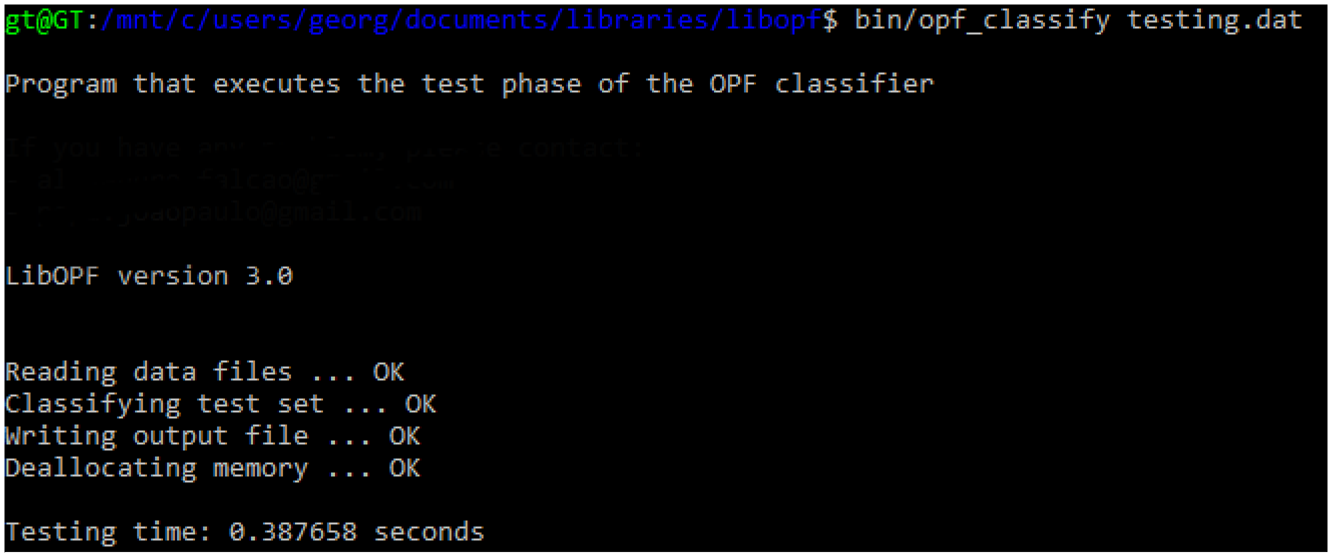
Next, using the *opf_learn* tool instead of *opf_train* (because opf_learn is for large datasets) we trained the classifier. As you can see the classifier keeps learning from errors in the evaluation set with multiple iteration until the accuracy of evaluation set becomes 100%.



```
gt@GT:/mnt/c/users/georg/documents/libraries/libopf$ bin/opf_learn training.dat evaluating.dat

Program that executes the learning phase for the OPF classifier

If you have any problem, please contact:
- alexandra.falcao@gmail.com
- papa.joaopaulo@gmail.com

LibOPF version 2.0 (2009)


Reading data file ... OK
Learning from errors in the evaluation set...
running iteration ... 0 opf_Accuracy in the evaluation set: 57.14 %

running iteration ... 1 opf_Accuracy in the evaluation set: 100.00 %

running iteration ... 2 opf_Accuracy in the evaluation set: 100.00 %

Final opf_Accuracy in the training set: 100.00%
Final opf_Accuracy in the evaluation set: 100.00%

Writing classifier's model file ... OK
Deallocating memory ... OK
```

Figure 7.3 *Training Phase*

Then, using the classifier we just trained, we execute the test phase of the opf classifier.

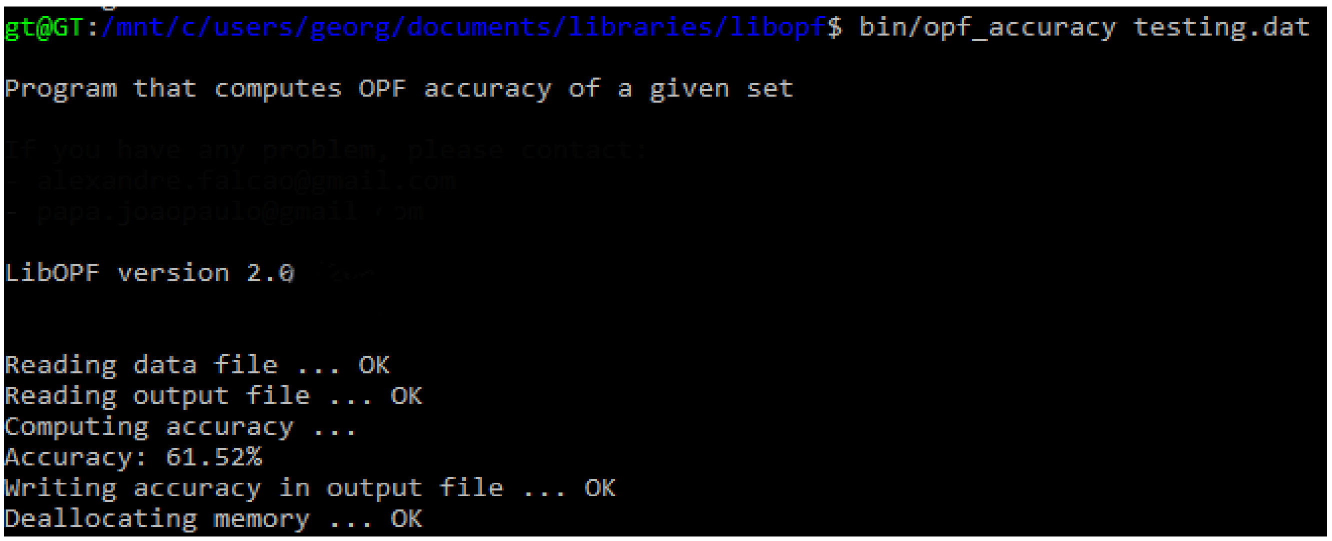


```
gt@GT:/mnt/c/users/georg/documents/libraries/libopf$ bin/opf_classify testing.dat

Program that executes the test phase of the OPF classifier

If you have an[illegible], please contact:
- al[illegible]falcao@g[illegible].com
- [illegible]joaopaulo@gmail.com

LibOPF version 3.0


Reading data files ... OK
Classifying test set ... OK
Writing output file ... OK
Deallocating memory ... OK

Testing time: 0.387658 seconds
```

Figure 7.4 *Testing phase*

Lastly, using the opf_accuracy tool we compute the accuracy of the classification in each case of electricity theft that have been examined. All the previous steps were executed each time we examined each case of electricity theft but for brevity we only show you one time. Below you can see the accuracy of each case.

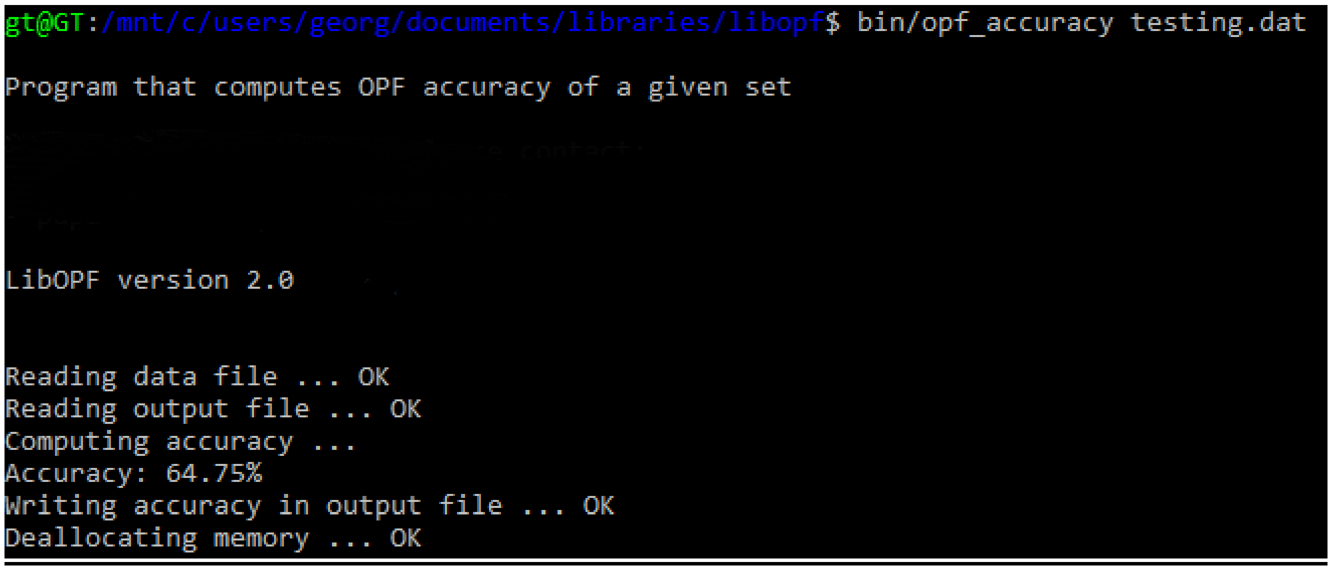
1st CASE : Theft with overload 40%-60%



```
gt@GT:/mnt/c/users/georg/documents/libraries/libopf$ bin/opf_accuracy testing.dat

Program that computes OPF accuracy of a given set

[redacted]

LibOPF version 2.0 [redacted]


Reading data file ... OK
Reading output file ... OK
Computing accuracy ...
Accuracy: 61.52%
Writing accuracy in output file ... OK
Deallocating memory ... OK
```

Figure 7.5 *OPF classification accuracy*

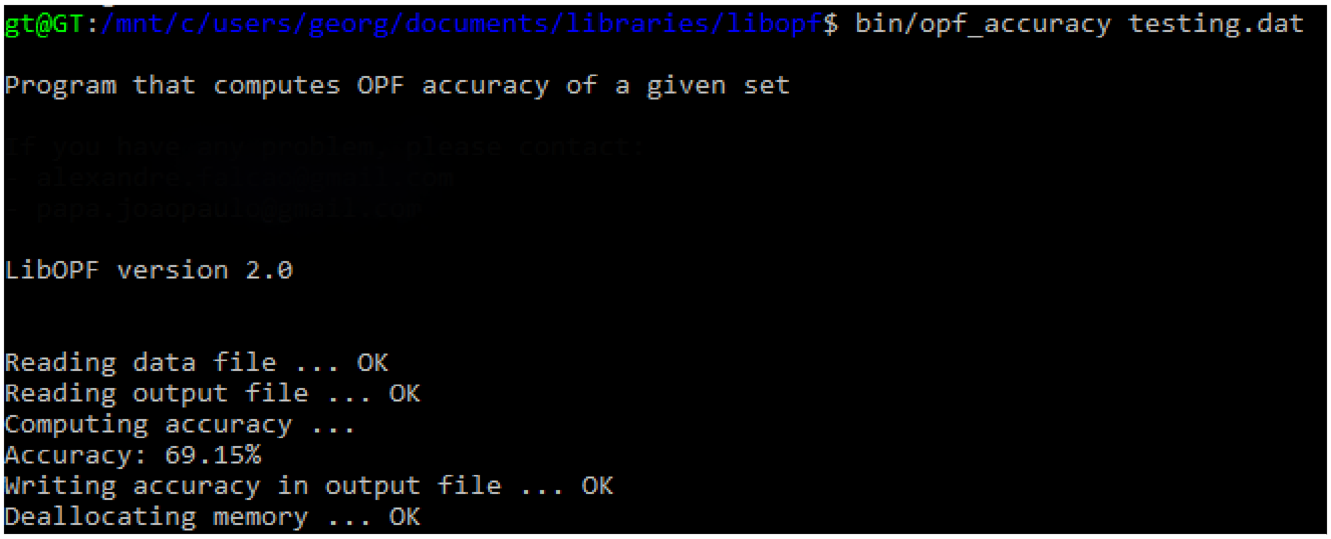
2nd CASE : Theft with overload 60%-80%



```
gt@GT:/mnt/c/users/georg/documents/libraries/libopf$ bin/opf_accuracy testing.dat

Program that computes OPF accuracy of a given set

[redacted]

LibOPF version 2.0 [redacted]


Reading data file ... OK
Reading output file ... OK
Computing accuracy ...
Accuracy: 64.75%
Writing accuracy in output file ... OK
Deallocating memory ... OK
```

Figure 7.6 *OPF classification accuracy*

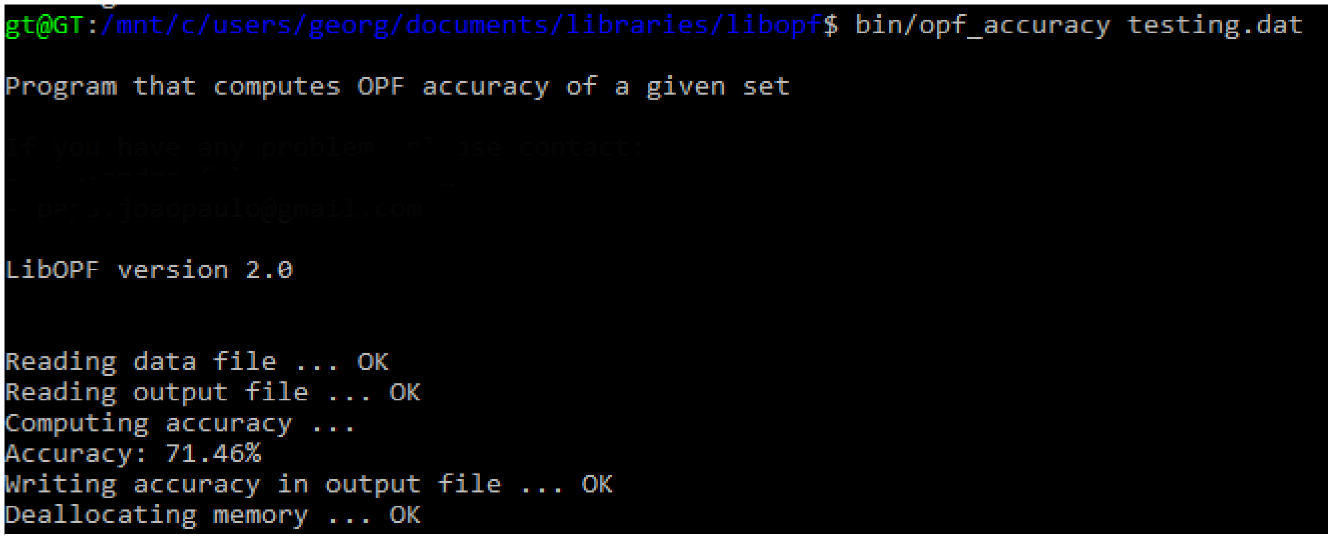
3rd CASE : Theft with overload 80%-100%



```
gt@GT:/mnt/c/users/georg/documents/libraries/libopf$ bin/opf_accuracy testing.dat

Program that computes OPF accuracy of a given set

If you have any problem, please contact:
- alexandre.falcao@gmail.com
- papa.joaopaulo@gmail.com

LibOPF version 2.0


Reading data file ... OK
Reading output file ... OK
Computing accuracy ...
Accuracy: 69.15%
Writing accuracy in output file ... OK
Deallocating memory ... OK
```

Figure 7.7 *OPF classification accuracy*

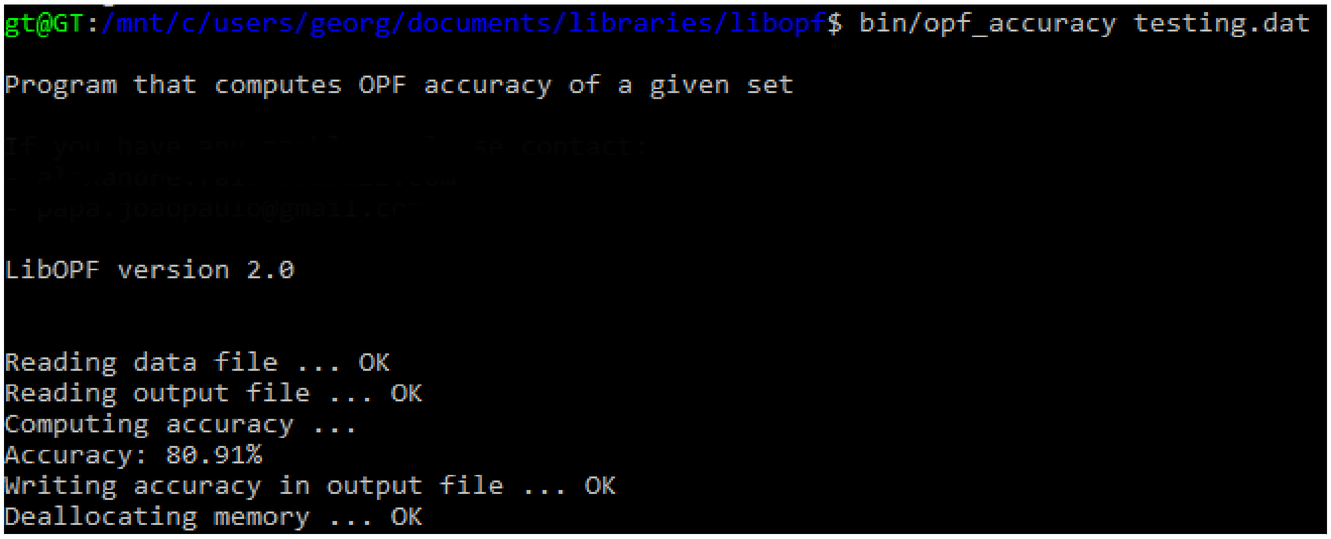
4th CASE : Partial theft 30%-50%



```
gt@GT:/mnt/c/users/georg/documents/libraries/libopf$ bin/opf_accuracy testing.dat

Program that computes OPF accuracy of a given set

If you have any problem, [illegible]se contact:
[illegible]
- [illegible].joaopaulo@gmail.com

LibOPF version 2.0


Reading data file ... OK
Reading output file ... OK
Computing accuracy ...
Accuracy: 71.46%
Writing accuracy in output file ... OK
Deallocating memory ... OK
```

Figure 7.8 *OPF classification accuracy*

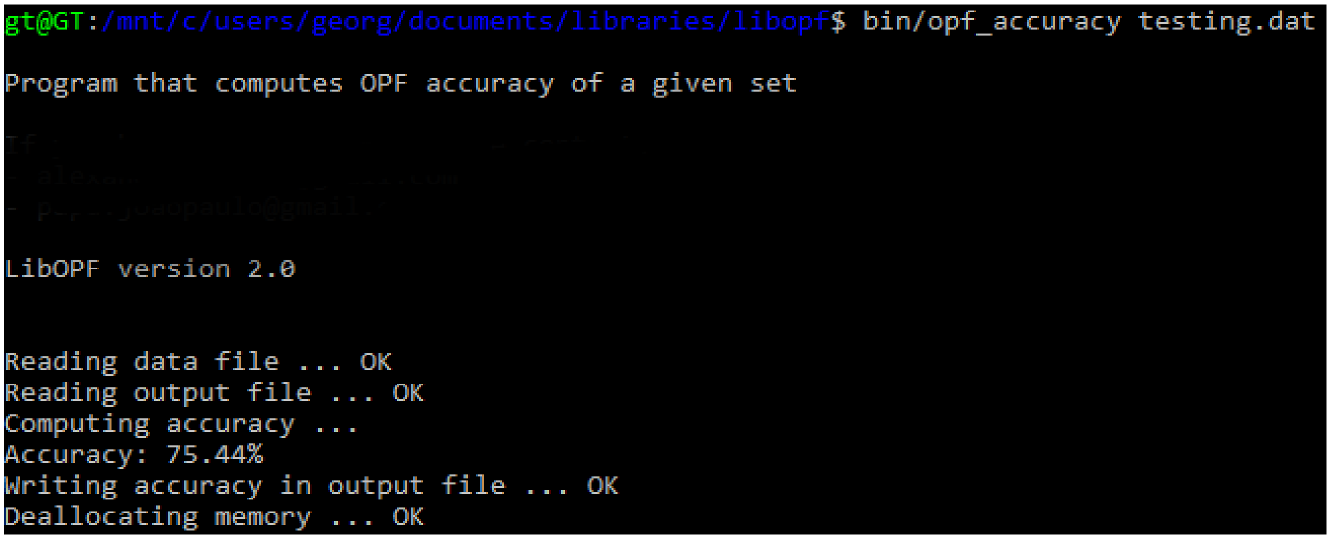
5th CASE : Partial theft 50%-70%



```
gt@GT:/mnt/c/users/georg/documents/libraries/libopf$ bin/opf_accuracy testing.dat

Program that computes OPF accuracy of a given set

[redacted]

LibOPF version 2.0


Reading data file ... OK
Reading output file ... OK
Computing accuracy ...
Accuracy: 80.91%
Writing accuracy in output file ... OK
Deallocating memory ... OK
```

Figure 7.9 *OPF classification accuracy*

6th CASE : Theft with overload 60%-80% and partial theft 50%-70%



```
gt@GT:/mnt/c/users/georg/documents/libraries/libopf$ bin/opf_accuracy testing.dat

Program that computes OPF accuracy of a given set

[redacted]

LibOPF version 2.0


Reading data file ... OK
Reading output file ... OK
Computing accuracy ...
Accuracy: 75.44%
Writing accuracy in output file ... OK
Deallocating memory ... OK
```

Figure 7.10 *OPF classification accuracy*

The table below shows the accuracy metrics we computed from the simulations of each case that have been examined with the optimum path forest classifier. Of course, these numbers differentiated slightly with each simulation. The performance metrics that you can see below represent the average simulations.

CASES	Accuracy	Time (s)
1st Case - overload_40_60	61,52%	33
2nd Case - overload_60_80	64,75%	37
3rd Case - overload_80_100	69,15%	31
4th Case - partial_theft_30_50	71,46%	41
5th Case - partial_theft_50_70	80,91%	36
6th Case- overload_60_80_&_partial_theft_50_70	75,44%	43

Table 7.1 *Accuracy and time of execution of all the cases*

As you can see from the table above, the performance of this method is quite disappointing. The accuracy ranges from 61,52% to 80,91% and that is actually a great chasm. This chasm probably happened because OPF is sensible to noise and outliers(since the prototypes choosing were based on the MST) and so it chooses noisy samples or outliers to become prototypes. These samples have great influence on OPF's classification.

As much as we experimented with this method we could not make this technique to achieve better results. We tried splitting the dataset manually and with library's default program, we tried changing the division of the training, testing and evaluating set, we experimented with different parameters, we used multiple tools that the library offered, we used different features as input, but always ended up with similar disappointing results.

Apart from the disappointing results, the file format that the library demanded as input in order to work was very hard to be composited from the original dataset we had in our possession. All the information between each phase of this procedure was saved in .dat files and as a result we couldn't extract anything we wanted. Lastly, the tools-programs that the LibOPF library use for the classification is 'locked' into files and you cannot intervene into their code. That is the reason that we couldn't compute any other metrics besides accuracy, which was computed with library's program '*opf_accuracy*'.

All in all, this method did not reach the expectations we had when we carried out our research about electricity theft detection techniques before starting this thesis. The results are not encouraging, even though as you can see from the table 7.1, it is the faster method compared to the others. Of course, there is a possibility that we did something wrong when using this method or this method is not as appropriate as the others for the energy consumption data we had in our possesion. I dedicated less time on this method than on the others, but this is because the results were not desirable from the beginning. In conclusion, after all these methods that I have experimented within this thesis, this method is definitely not recommended for detection of electricity theft.

8. Power Theft Detection Via Neuro-Fuzzy System (NFS)

Neuro-fuzzy hybridization results in a hybrid intelligent system that synergizes these two techniques by combining the human-like reasoning style of fuzzy systems with the learning and connectionist structure of neural networks. Neuro-fuzzy hybridization is widely termed as fuzzy neural network (FNN) or neuro-fuzzy system (NFS) in the literature. Neuro-fuzzy system (the more popular term is used henceforth) incorporates the human-like reasoning style of fuzzy systems through the use of fuzzy sets and a linguistic model consisting of a set of "if-then" fuzzy rules. The main strength of neuro-fuzzy systems is that they are universal approximators with the ability to solicit interpretable "if-then" rules [68].

8.1 Architecture of Neuro-Fuzzy Inference System

A neuro-fuzzy inference system or network-based fuzzy inference system is a kind of artificial neural network that is based on Takagi–Sugeno fuzzy inference system [67]. Since it integrates both neural networks and fuzzy logic principles, it has potential to capture the benefits of both in a single framework. Its inference system corresponds to a set of fuzzy "if–then" rules that have learning capability to approximate nonlinear functions. Hence, neuro-fuzzy inference system is considered to be a universal estimator. It has uses in intelligent situational aware energy management system [65].

It is possible to identify two parts in the network structure, namely premise and consequence parts. In more details, the architecture is composed by five layers. The first layer takes the input values and determines the membership functions belonging to them. It is commonly called fuzzification layer. The membership degrees of each function are computed by using the premise parameter set, namely {a,b,c}. The second layer is responsible of generating the firing strengths for the rules. Due to its task, the second layer is denoted as "rule layer". The role of the third layer is to normalize the computed firing strengths, by diving each value for the total firing strength. The fourth layer takes as input the normalized values and the consequence parameter set {p,q,r}. The values returned by this layer are the defuzzificated ones and those values are passed to the last layer to return the final output [65].

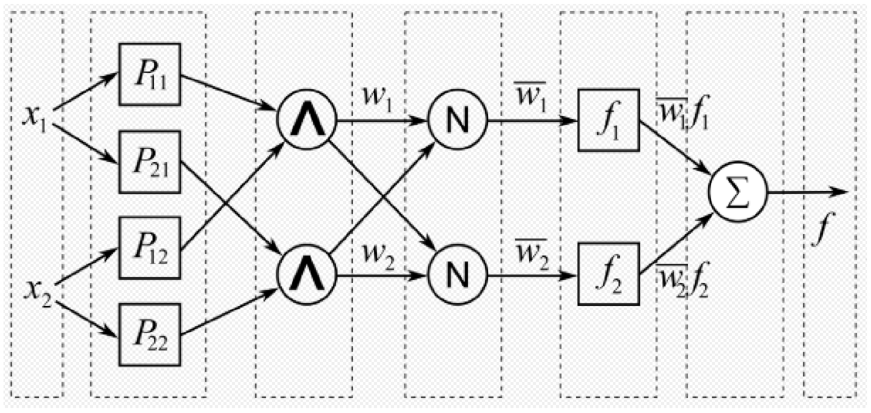


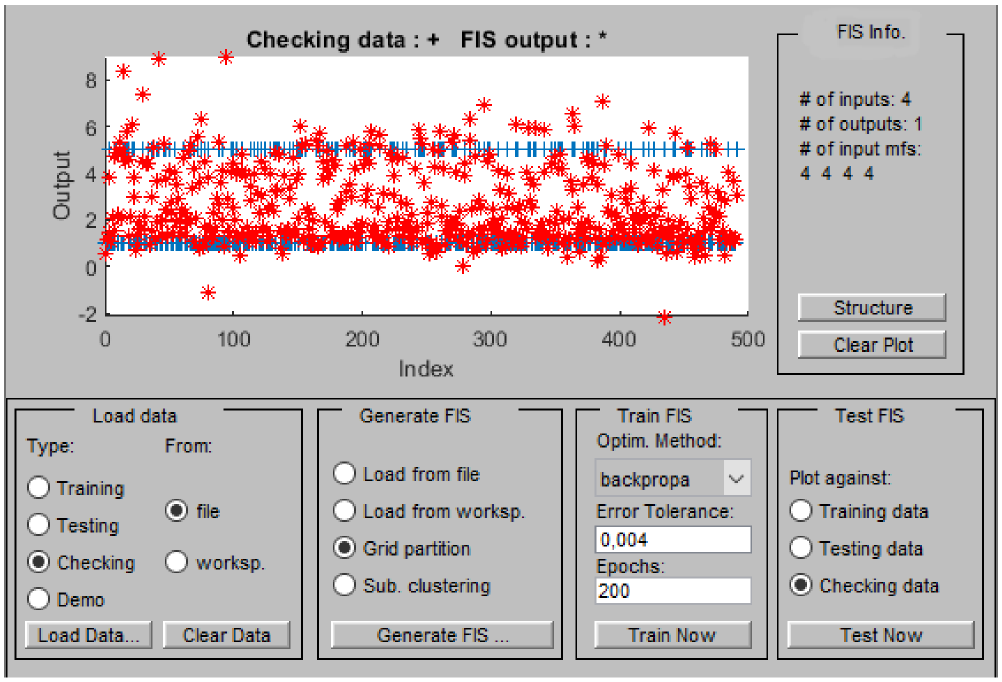
Figure 8.1 *A Neuro-Fuzzy System.*

8.2 Neuro-Fuzzy Designer App

Our work in this method was based on the Neuro-Fuzzy Designer app, a Matlab tool which let you design, train and test Takagi-Sugeno fuzzy inference systems using input/output data. Using Neuro-Fuzzy Designer app, we could:

- Tune membership function parameters of Takagi-Sugeno type fuzzy inference systems.
- Automatically generate an initial inference system structure based on our training data.
- Modify the inference system structure before tuning.
- Prevent overfitting to the training data using additional checking data.
- Test the generalization ability of our tuned system using testing data.
- Export our tuned fuzzy inference system to the Matlab workspace.

Despite the complexity of the neuro-fuzzy inference systems, the Neuro-Fuzzy Designer app was easy to use and versatile and provided many advantages for the implementation of neuro-fuzzy inference systems. In figure 8.2 you can see the neuro-fuzzy designer app which we used for the analysis and classification of consumers to normal and fraudulent respectively.



# of inputs: 4
# of outputs: 1
# of input mfs:
4 4 4 4

Type: Training [✓ Checking] Testing Demo
From: [✓ file] worksp.

Load from file
Load from worksp.
[✓ Grid partition]
Sub. clustering

Optim. Method: backpropa
Error Tolerance: 0,004
Epochs: 200

Plot against: Training data Testing data [✓ Checking data]

Figure 8.2 *Neuro-Fuzzy Designer App.*

8.3 Classification Procedure

As we mentioned in the beginning of chapter 3, the real energy consumption measurements we have in our possession is a struct of data with different kinds and percentages of electricity theft in each data table.

For every case of electricity theft the most common features were tested. These features were : Average (average of daily measurements), Mean, Skewness, Variance, Standard Deviation, Correlation Coefficient and Range. After a lot of experimentations, we ended up using as inputs, the 4 features which maximized the accuracy of the classification process. These features are: the Average, the Mean, the Skewness and the Standar Deviation.

Afterwards, the dataset for each case of electricity theft were split into training, checking and testing datasets with a ratio of 70% (2291 consumers), 15% (491 consumers) and 15% (491 consumers), respectively. We exported these datasets as .dat files (.dat is the type of file that neuro-fuzzy designer app demand for training, checking and testing datasets) and loaded them to the neuro-fuzzy designer app. Then we generated and trained the FIS model with the configurations you can see in table 8.1.

MF type	Generalized bell-shaped
Number of MFs	4
Output MF	Constant
Optimization Method	Backpropagation Algorithm
Error Tolerance	0.004
Number of epochs	200

Table 8.1 *Configurations of FIS Model*

Finally, we tested the FIS model and estimated the performance based on its predicted values. As you can see in the figure 8.3, the red “stars” (*) represent the predicted values and the blue “crosses” (+) represent the actual classes (1 for legal consumers and 5 for fraudsters).

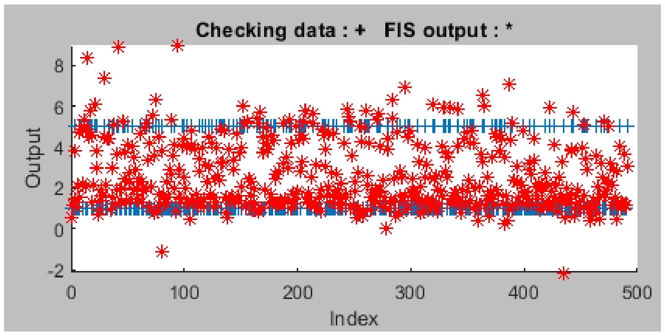


Figure 8.3 *FIS Predicted Values*

Based on the FIS predicted values we make the final decision about the consumer:

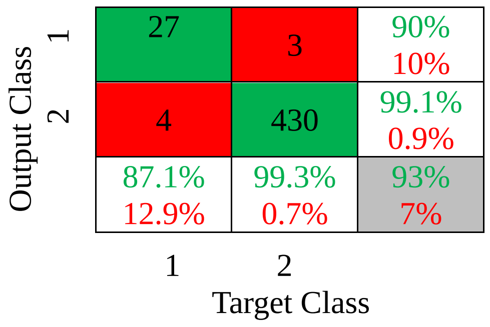
- ❖ If the predicted value is closer to 1, we classify the consumer as legal.
- ❖ If the predicted value is closer to 5, we classify the consumer as illegal.

Of course, the predicted values don’t lead always to correct estimations about the consumers. Luckily, cases like these are minimal and the performance of the FIS model remains very good.

8.4 Results

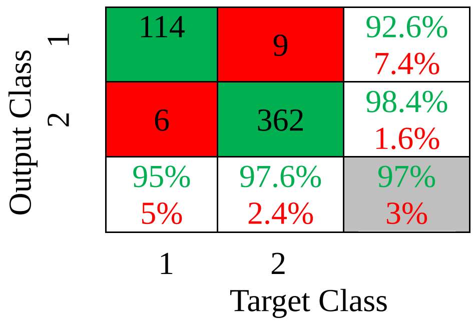
Below you can see the confusion matrices for each case of electricity theft that was tested.

<u>1st CASE : Theft with overload 40%-60%</u>



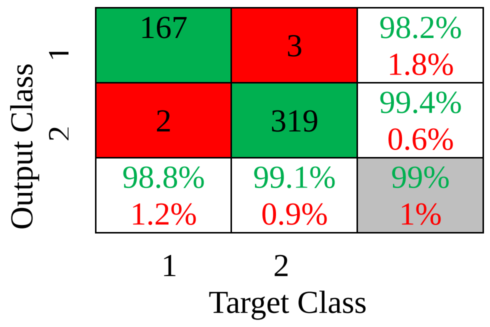
Output Class	Target Class 1	Target Class 2	
1	27	3	90% 10%
2	4	430	99.1% 0.9%
	87.1% 12.9%	99.3% 0.7%	93% 7%

<u>2nd CASE : Theft with overload 60%-80%</u>



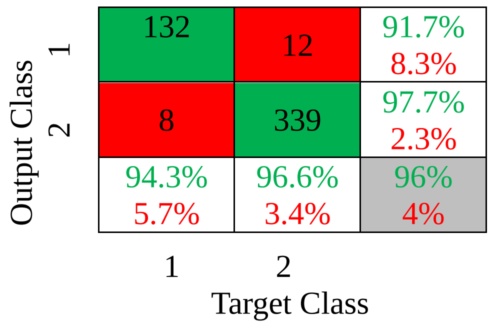
Output Class	Target Class 1	Target Class 2	
1	114	9	92.6% 7.4%
2	6	362	98.4% 1.6%
	95% 5%	97.6% 2.4%	97% 3%

3rd CASE : Theft with overload 80%-100%



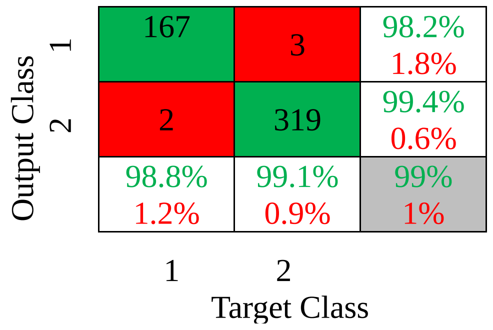
Output Class \ Target Class	1	2	
1	167	3	98.2% 1.8%
2	2	319	99.4% 0.6%
	98.8% 1.2%	99.1% 0.9%	99% 1%

4th CASE : Partial theft 30%-50%



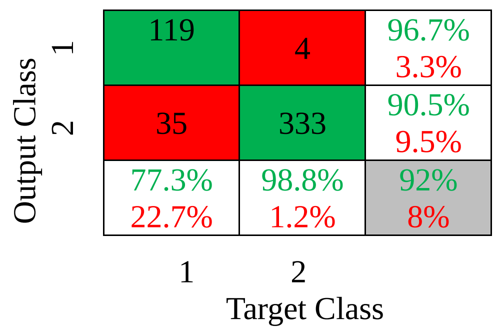
Output Class \ Target Class	1	2	
1	132	12	91.7% 8.3%
2	8	339	97.7% 2.3%
	94.3% 5.7%	96.6% 3.4%	96% 4%

5th CASE : Partial theft 50%-70%



Output Class \ Target Class	1	2	
1	167	3	98.2% 1.8%
2	2	319	99.4% 0.6%
	98.8% 1.2%	99.1% 0.9%	99% 1%

6th CASE : Theft with overload 60%-80% and partial theft 50%-70%



As you can see from table 8.2 the performance metrics of this method were very satisfying in all cases.

CASES	**Accuracy**	**F1 Score**	**Precision**	**Recall**
1st Case - overload_40_60	0.93	0.88	0.90	0.87
2nd Case - overload_60_80	0.97	0.95	0.93	0.95
3rd Case - overload_80_100	0.99	0.99	0.98	0.99
4th Case - partial_theft_30_50	0.96	0.93	0.92	0.94
5th Case - partial_theft_50_70	0.99	0.99	0.98	0.99
6th Case - overload_60_80_&_partial_theft_50_70	0.92	0.86	0.97	0.77

***Table 8.2** Total Performance Metrics*

Generally, this method as you can see from the accuracy metrics in table 8.2 yielded great results. The accuracy metrics ranged from 92%-99% (depending on the case that was tested). The Neuro-Fuzzy Designer that was used in this method for the implementation of the FIS model, was easy to use and versatile (we could adjust many parameters easily). From all the methods that were tested in this thesis, the method of this chapter was the best in performance. Certainly, it was not the faster method, as the training of the fis model took quite some time but the results were very positive. This combination of multilayer artificial neural networks with fuzzy logic proved the best method for detection of electricity theft.

9. Conclusions

Power theft is a widespread phenomenon not only in undeveloped countries, but all around the world. The evidence points to the increasing levels of power theft in many countries and the financial losses for some systems are so immense that the utilities are in financial turmoil. Investment in improving the system and adding additional capacity cannot be undertaken, loans and payments cannot be met, and the consumer faces increased electricity charges. Even in efficient systems, theft losses can account for millions of dollars each year in lost revenue.

The transition toward future smart networks should enhance the insight into the distribution networks, increase their reliability, flexibility and reduce the transmission and distribution losses. With the development of advanced metering infrastructure in smart grid, a more complicated situation in energy theft has emerged and many new technologies are adopted to try to solve this problem. Electricity theft in its various forms can be reduced and kept in check only by the strong and assertive action of power sector organizations. The strategy and the action should be based upon a thorough understanding of the specific nature of the theft problem. A strong case can be made that each power system (including consumer’s attitudes and behavior) has its own unique qualities and only by knowing the system and the problem can effective solutions be designed and implemented. Since a high level of power theft is linked with corruption, the analysis cannot be confined to technical and managerial perspectives and needs to be multi-disciplinary in approach. Theft as an activity in some systems is closely intertwined with governance and with the social, economic and political environment. Corruption and electricity theft thrives off each other. In an overall culture of corruption as a way of life, electricity theft can be reduced to moderate levels by technical/engineering methods. But it is an uphill battle to reduce the electricity theft rate drastically as long as extensive corruption continues. Reduction in power theft and keeping it within reasonable bounds is more likely to be successful in systems with a good governance culture. This is because the theft reduction mechanisms find a friendly environment for initiation and implementation. As part of generating and sustaining good governance in communities, electric power systems have the opportunity to take the lead in promoting sound corporate governance. The technological innovations make this task easier, provided that the managerial skills and desire exist. Electric power systems can be restructured to make power sector organizations operate in competitive environments where efficiency and effectiveness in service delivery are both virtues and necessities. Smart meters are essential for this restruction. The theft detection should be considered as one of the important aspects of future distribution networks, as well.

The main contribution of this thesis is the proposal of four methods for automated detection of illegal use of electricity in the low voltage distribution networks :

<u>Multilayer Artificial Neural Networks with FANN Library</u>

The FANN library that was used in this method, was easy to use, fast (faster than all the other methods except of LibOPF) and versatile (we could adjust many parameters and features easily). The dataset needed some fairly easy preparations in order to use as input into the library.

From all the methods that were tested in this thesis, the method with the FANN Library was one of the best, both in performance and speed. The accuracy metrics ranged from 89%-97% (depending on the case that was tested) and that’s why it is totally recommended for electricity theft detection.

Neural Networks with Deep Learning Toolbox

This method, as you can see from the accuracy metrics in table 9.1, worked very well. The accuracy metrics ranged from 84%-93% (depending on the case that was tested). However, it has some disadvantages:

- This method is by far the most time consuming method that has been tested in this thesis. A simulation with all the customers (3273) could take up to 10 minutes even with the use of principal component analysis, which drastically reduce the time of a full simulation.
- The random weights that the pattern recognition app chooses by default sometimes lead to moderate classifications (rarely we had some simulations with 76-80% accuracy due to that act).

Of course, this method takes advantage of Matlab software in terms of visualization, facility and versatility. In total, in terms of performance this method was the third best that was tested in this thesis and constitutes one very good method for electricity theft detection.

Support Vector Machine Classification

Support Vector Machines can produce accurate and robust classification results on a sound theoretical basis, even when input data are non-monotone and non-linearly separable. So they can help to evaluate more relevant information in a convenient way. Since they linearize data on an implicit basis by means of kernel transformation, the accuracy of the results does not rely on the quality of human expertise judgment for the optimal choice of the linearization function of non-linear input data. For these reasons SVMs are regarded as a useful tool for effectively complementing the information gained from classical linear classification techniques.

SVM and Artificial Neural Networks are two popular strategies for supervised machine learning and classification. SVM benefits depend on a particular project. The SVM classifier is a great choice for unbalanced data. As a cost-sensitive classifier it can solve the problem of unbalanced data. All other benefits are really depending on the domain and task. But even if the number of positive and negative examples are not similar, SVM would work fine if we normalize the data or may be projected into the space of the decision boundary which separates the two classes. SVM is quite good compared to other classifiers as the computational complexity is reduced and classification efficiency is increased when compared to any other non linear classifier [52].

As you can see from the performance metrics the support vector machine classification achieved satisfying results but not as good as the previous two methods that have been tested. The svm classification’s accuracy ranged from 79% to 87% (depending on the case that was tested). Certainly, the results are not negative but not as great as the previous two methods that were tested. Specifically, in the 1st (overload 40%-60%) and the 4th (partial theft 30%-50%) case the performance metrics dropped due to the fact that in these cases the consumers’ samples are very similar to each other and the classifier could not distinguish easily the fraudster from the non-fraudster. If the data was less balanced the results will probably be better as most of 2-class classifier works better that way but generally the support vector machine classification was encouraging for the detection of electricity theft.

Optimum Path Forest Classification

As you can see from the table 9.1, the performance of this method is not encouraging. The accuracy ranges from 61,52% to 80,91% and that is actually a great chasm. This chasm is probably happening because OPF is sensible to noise and outliers, since the prototypes choosing were based on the Minimum Spanning Tree (MST), it chooses noisy samples or outliers to become prototypes and these samples have great influence on OPF's classification decision.

As much as we experimented with this method we could not make this technique achieve better results. We tried splitting the dataset manually and with the library default program, we tried changing the division of the training, testing and evaluating set, we experimented with different parameters, we used multiple tools that the library offered, we used different features as input but always ended up with similar disappointing results.

Apart from the disappointing results, the file format that the library demanded as input in order to work was very hard to be composited from the original dataset we had in our possession. All the information between each phase of this procedure was saved in .dat files and as a result we couldn't extract anything we wanted. Lastly, the tools-programs that the LibOPF library uses for the classification is 'locked' into files and you cannot intervene into their code. That is the reason that we couldn't compute any other metrics besides accuracy, which was computed with library's program '*opf_accuracy*'.

All in all, this method did not reach the expectations we had when we carried out our research about electricity theft detection techniques before starting this thesis. The results are not encouraging, even though as you can see from table 7.1, it is the faster method compared to the others. Of course, there is a possibility that we did something wrong when using this method or this method is not as appropriate as the others for the energy consumption data we had in our possesion. I dedicated less time on this method than the others but this is because the results were not desirable from the beginning. In conclusion, after all these methods that I have experimented with in this thesis, this method is definitely not recommended for detection of electricity theft.

Fuzzy Inference System with Neuro-Fuzzy Designer App

This method as you can see from the accuracy metrics in table 9.1 yielded great results. The accuracy metrics ranged from 92%-99% (depending on the case that was tested). The Neuro-Fuzzy Designer that was used in this method for the implementation of the FIS model, was easy to use and versatile (we could adjust many parameters easily). From all the methods that were tested in this thesis, the method of this chapter was the best in performance. Certainly, it was not the faster method, as the training of the FIS model took quite some time but the results were very positive. The combination of multilayer artificial neural networks with fuzzy logic, through the FIS model, proved to be one of best method for electricity theft detection.

Below you can see a general table with the accuracy of each method that was implemented in this thesis in each case of electricity theft that was tested.

Methods	Overload 40-60%	Overload 60-80%	Overload 80-100%	Partial Theft 30-50%	Partial Theft 50-70%	Partial Theft 50-70% & Overload 60-80%
FANN	89%	93%	95%	92%	97%	95%
Deep Learning Toolbox	86%	87,8%	89,2%	84,8%	92,6%	91,8%
SVM	80%	83%	85%	79%	87%	85%
LibOPF	61,52%	64,75%	69,15%	71,46%	80,91%	75,44%
Neuro-Fuzzy System	93%	97%	99%	96%	99%	92%

Table 9.1 *Total Performance Metrics*

In conclusion, the overall research has shown encouraging results for the detection of abnormalities in the electrical grid. This thesis illustrates various cases, issues and setbacks in the operation of electricity theft controlling devices. Our work was based on real energy consumption data from an electrical grid in Ireland. We tested each case of electricity theft with 8-hour and 30-minutes measurements but the results were quite similar and we didn’t notice great differences. Overall, we are very pleased with the results that have been achieved over the course of this thesis and we have surely met our expectations.

9.1 Further Research

This thesis showed an attractive approach for the identification of energy consumption irregularities in a smart grid. However, there are a number of gaps in our knowledge and deficiencies in the utilities around electricity theft, that would benefit from further research in order to extend and further test what we have developed here:

• Detection algorithms can be enhanced by introducing more real-world parameters and variables (both technical and non-technical), so that the process of mapping energy patterns into irregularities is further strengthened. Integration of these new technical and non-technical parameters into classification algorithms may come into light after complete implementation of smart grid. This inclusion will ensure the impact of such parameters on algorithms for identification of illegal consumers. Real world parameters and variables can be the economic situation of a person or illegal activity in the past.

- Enhancements of the analysis on the impact of Time Base Pricing (TBR) and Distributed Generation (DG) on customer energy consumption patterns in collaboration with DSM and smart home management tools can be made. Study on the impact of Time Base Pricing and Real Time Pricing (RTP) on illegal consumption of electricity will be one of the most significant factors of future home energy management. This kind of analysis will benefit both genuine customers and utilities.

- Illegal consumption of electricity may be controlled by focusing on cyber security (designing stronger firewalls or enhancing firmware in view of cyber security). For example, complicating the process of meter tampering and reducing the hackings of smart meters through the establishment of certain standards. These standards are envisioned to control the flexibility of illegal consumers in installing external devices or software or firmware updates from third-party developers on their smart meters.

- Economically examine the scenario to place more than one sum meters in a line so as to localize more accurately the illegal consumers.
- Localization of a customer who does not have a smart meter and steal electricity from the power line directly.

- Integration of Geographic Information System (GIS) for real-time visualization of a customer’s energy consumption, pattern of irregularities and power generated from DG sources on customer premises. This portion of the work can be further extended to thematic mapping, e.g. economy and illegal consumption of electricity or literacy and illegal consumption of electricity. Thematic mapping provides an extensive understanding of the impact of illegal consumption on a large geographic entity.

- Definition of the most dangerous groups for power theft.

- Inclusion of the impact of illegal consumption in tools that perform dynamic optimization of Voltage and Vars are used to reduce the load on the grid.

- Artificial Intelligence (AI) algorithms which will sift through energy consumption readings in order to detect abnormal usage.

Bibliography

[1] C.J. Bandim , J.E.R. Alves, A.V. Pinto, F.C. Souza, M.R.B. Loureiro, C.A. Magalhaes, F. Galvez-Durand, ”Identification of energy theft and tampered meters using a central observer meter: a mathematical approach”, In : Proceedings of the IEEE PES Transmission and Distribution Conference and Exposition, Brazil, Vol. 1, pp. 163-168, 2003

[2] Anand R., S. De, Naveen S.A., “Design and development of vigilant energy metering system (VEMS) and its applications”, Student Conference on Research and Development, New Delhi, India, Vol. 11, pp 15-18, August 2003.

[3] Nagi J., Yap K.S.,Tiong S.K., Ahmed S.K., Mohammad A.M., “Detection of abnormalities and electricity theft using genetic support vector machines.” In: Proceedings of the IEEE Region 10 Conference TENCON, Hyderabad, India, Vol. 23, pp. 1-6, January 2009.

[4] Nizar A.H., Dong Z.Y., “Identification and detection of electricity customer behavior irregularities.” In : Proceedings of the IEEE/PES Power Systems Conference and Exposition, Seattle, Washington, Vol. 23 , pp. 1-10, March 2009.

[5] Pasdar A., Mirzakuchaki S.A. “A solution to remote detecting of illegal electricity usage based on smart metering.” In : Proceedings of the International Workshop on Soft Computing Applications, Oradea, Romania, Vol. 12, pp 163-167, August 2007.

[6] Zeng X., Jin G., Jin W., Xu Y., “Anti theft and monitoring system of street lamp power cables.” In : Proceedings of the Asia-Pacific Power and Energy Engineering Conference, Wuhan, China,Vol. 2, pp. 1-4, March 2009.

[7] Jamil M., Munir F., Khan A.A., Mirza A., “Telemetering and billing system for spatially distributed electrical power clients.” In: Proceedings of the E-Tech 2004, Karachi, Pakistan, pp. 35-40, July 2004.

[8] De S., Anand R., Naveen A., Moinuddin S., “E-metering solution for checking energy thefts and streamlining revenue collection in India” In: Proceedings of the IEEE PES Transmision and Distribution Conference and Exposition, Dallas Texas, Vol 3, No. 9, pp. 654-658, September 2003.

[9] Mano R., Cespedes R., Maia D., “Protecting revenue in the distribution industry: a new approach with the revenue assurance and audit process.” In: Proceedings of the IEEE/PES
Transmission and Distribution Conference and Exposition, Latin America, Kema, Brazil, pp. 218-223, November 2004

[10] Perez E.H., NKanka B.N., Ngulumingi C.V., Gimeno A., Kazadi A.B., “Analysis of technical losses in distribution networks of large cities in underdeveloped African countries.” In: Proceedings of the IEEE Power Engineering Society Inaugural Conference and Exposition in Africa, Durban, South Africa, pp. 92-96, July 2005.

[11] Soma Shekara Sreenadh Reddy Depuru, Lingfeng Wang, Vijay Devabhaktuni, “Electricity theft : Overview, issues, prevention and a smart meter based approach to control theft.”, Department of Electrical Engineering and Computer Science, University of Toledo, Toledo, OH 43606, USA, Vol. 39, pp. 1007-1015, February 2011.

[12] Solomon Nunoo, Joseph Attachie, “A methodology for the design of an electricity theft monitoring system.” In: Journal of Theoretical and Applied Information Technology, Vol. 26, pp. 112-117, April 2011

[13] Rong Jiang, Rongxing Lu, Ye Wang, Jun Luo, Changxiang Shen, and Xuemin (Sherman) Shen, “Energy-Theft Detection Issues for Advanced Metering Infrastructure in Smart Grid.” In: Proceedings of the IEEE Transactions on Power Systems, Vol. 19, pp. 105-120, 2009.

[14] Soma Shekara Sreenadh Reddy Depuru, Lingfeng Wang, Vijay Devabhaktumi, “Enhanced Encoding Technique for Identifying Abnormal Energy Usage Pattern.” In : EECS Department, 2012.

[15] P.Kadurek, J. Blom, J.F.G. Cobben, W.L. Kling, “Theft detection and smart metering practices and expectations in the Netherlands” In: Innovative Smart Grid Technologies Conference Europe (ISGT Europe), Vol. 2, pp. 654-658, June 2010.

[16] Shih-Che Huang, Yuan-Liang Lo, Chan-Nan Lu, “Non-Technical loss Detection Using State Estimation and Analysis of Variance.” In: IEEE Transactions on Power Systems, Vol. 19, No. 4, pp. 1663-1667, June 2013.

[17] Robert Czechowski, Anna Magdalena Kosek, “The Most Frequent Energy Theft Techniques and Hazards in Present Power Energy Consumption.”, IEEE, Vol. 3, April 2016.

[18] Caio C. O. Ramos , Andre N. Souza, Joao P. Papa, Alexandre X. Falcao, “Learning to Identify Non-Technical Losses with Optimum-Path Forest” at IWSSIP - 17th International Conference on Systems, Signals and Image Processing, 2010.
https://pdfs.semanticscholar.org/d66b/a8ac378771e3edbe001f18856bb65e993c2e.pdf

[19] Breno C. Costa, Bruno L.A. Alberto, Andre M. Portela, W. Maduro, Esdras O. Eler, “Fraud Detection in Electric Power Distribution Networks Using an ANN-based Knowledge” at International Journal of Artificial Intelligence & Applications (IJAIA), Vol. 4, No. 6, November 2013.

[20] Patrick Glauner ,”Artificial Intelligence for the Detection of Electricity Theft and Irregular Power Usage in Emerging Markets” at Karlsruhe University of Applied Sciences, January 2019.
https://www.researchgate.net/publication/330485084_Artificial_Intelligence_for_the_Detection_of_Electricity_Theft_and_Irregular_Power_Usage_in_Emerging_Markets

[21] Awais Khan and Wei Xie , “Designing and Modeling of Automated Anti-theft Electricity Distribution System” *in* MATEC Web of Conferences, Vol. 160, January 2018.
https://www.researchgate.net/publication/324344675_Designing_and_Modeling_of_Automated_Anti-theft_Electricity_Distribution_System

[22] Ali Akbar Ghasemi and Mohsen Gitizadeh, ”Detection of illegal consumers using pattern classification approach combined with Levenberg-Marquardt method in smart grid” at Department of Electronics and Electrical Engineering, Shiraz University of Technology, Shiraz, Iran,Vol. 99, pp. 363-375, July 2018.

[23] Jawad Nagi, Keem Siah Yap, Sieh Kiong Tiong, Syed Khaleel Ahmed and Malik Mohamad : “Nontechnical Loss Detection for Metered Customers in Power Utility Using Support Vector Machines” at IEEE Transactions on Power Delivery, Vol. 25, No. 2, April 2010

[24] Konstantinos, Blazakis, and Stavrakakis Georgios. "Efficient Power Theft Detection for Residential Consumers Using Mean Shift Data Mining Knowledge Discovery Process." at International Journal of Artificial Intelligence and Applications (IJAIA), Vol. 10, No.1, January 2019

[25] Soma Shekara Sreenadh Reddy Depuru, “Modeling, Detection, and Prevention of Electricity Theft for Enhanced Performance and Security of Power Grid” at University of Toledo, January 2012.

[26] Jawad Nagi ,”An Intelligent System For Detection of Non-technical Losses in Tenaga Nasional Berhad Malaysia Low Voltage Distribution Network” at College of Graduate Studies University Tenaga Nasional, 2009

[27] Syifaul Fuada, “Proposed An Intelligent System for Electricity Theft Detector at Smart City Scenarios” at University Pendidikan, Indonesia, Vol. 5, No. 1, pp. 51-58, April 2016.
https://www.researchgate.net/publication/319377388_Proposed_An_Intelligent_System_for_Electricity_Theft_Detector_at_Smart_City_Scenarios

[28] Blazakis Konstantinos, “Development of Novel Approaches for Smart Electric Grids Measurements Processing” at Technical University of Crete, January 2016
https://www.researchgate.net/publication/330823982_Efficient_Power_Theft_Detection_for_Residential_Consumers_Using_Mean_Shift_Data_Mining_Knowledge_Discovery_Process?fbclid=IwAR1Xwz74tMBWozLsIFWCaYWnQAr8flSBSzZFkJD65okDMLA_mvsjsqpYpoM

[29] Maxime Guymard, “Modeling of Technical Losses in the Senegalese Transmission and Distribution Grids and Determination of Non-technical Losses” at Royal Institute of Technlogy, Stockholm, December 2012.

[30] Diego Ponce de Leon Barido and Deepa Madan “The Unintended Consequences of Power Theft and Restructure: A Case Study of Gujarat” at Electric Power Systems Conference, India, 2012.
http://dleonb.com/wp-content/uploads/2016/05/The-Unintended-Consequences-of-Power-Theft-and-Restructure-A-Case-Study-of-Gujarat-India.pdf

[31] Yichi Yang, Guangzi Xu, “Theft of Electricity Detection Based on Supervised Learning” at Hangzhou Foreign Languages School, August 2017.
http://archive.ymsc.tsinghua.edu.cn/pacm_download/232/8892-1_paper_1128.pdf

[32] João Paulo Papa, Silas Evandro Nachif Fernandes, Alexandre Xavier Falcão ,”Optimum-Path Forest based on k-connectivity” at Department of Computing, São Paulo State University, Brazil, Vol. 87, pp.117-126, February 2017.
https://reader.elsevier.com/reader/sd/pii/S0167865516302057?token=8993925881373E988364C8D2CE6B13FE0AAF86119BC953C827CC675E2DE97B8BCD1D9E3E5CDB613AE3C8DAEB5E2790A2

[33] João P. Papa, Alexandre X. Falcão, “A New Variant of the Optimum-Path Forest Classifier” at International Symposium on Visual Computing,Vol. 53, pp.935-944, 2008.
https://link.springer.com/chapter/10.1007%2F978-3-540-89639-5_89

[34] João P. Papa, Alexandre X. Falcão ”A Learning Algorithm for the Optimum-Path Forest Classifier” at International Workshop on Graph-Based Representations in Pattern Recognition,Vol 55, pp.195-204, 2009.
https://link.springer.com/chapter/10.1007%2F978-3-642-02124-4_20

[35] João P.Papa, Alexandre X.Falcão, Victor Hugo C. de Albuquerquec, João Manuel R.S. “Efficient supervised optimum-path forest classification for large datasets” at UNESP – Univ Estadual Paulista, Departamento de Computação, Bauru, Brazil, Vol. 45, No. 1, pp.512-520, January 2012.
https://www.sciencedirect.com/science/article/abs/pii/S0031320311003013

[36] J. P. Papa A. X. Falcão C. T. N. Suzuki, “Supervised pattern classification based on optimum-path forest” at International Journal of Imaging Systems and Technology, Vol. 19, No. 2, pp. 120-131, May 2009.
https://onlinelibrary.wiley.com/doi/abs/10.1002/ima.20188

[37] Future extraction Theory:
https://en.wikipedia.org/wiki/Feature_extraction

[38] Range Theory:
https://en.wikipedia.org/wiki/Range_(statistics)

[39] Correlation Coefficient Theory:
https://en.wikipedia.org/wiki/Correlation_coefficient

[40] Mean Algorithm Theory :
https://en.wikipedia.org/wiki/Mean

[41] Standar Deviation Theory:
https://en.wikipedia.org/wiki/Standard_deviation

[42] Variance Theory :
https://en.wikipedia.org/wiki/Variance

[43] FANN Webpage :
http://leenissen.dk/fann/wp/

[44] Principal Component Analysis Theory:
https://en.wikipedia.org/wiki/Principal_component_analysis

[45] Levenberg-Marquardt Algorithm :
https://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt_algorithm

[46] Number NN Layers :
https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw

[47] Support Vector Machines Theory :
https://en.wikipedia.org/wiki/Support-vector_machine

[48] Electricity Loss Reduction & Theft Management Summit 2019:
http://www.equip-global.com/electricity-loss-reduction-and-theft-management-2019uae

[49] Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures
https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

[50] Irish Social Science Data Archive, "Commission for energy regulation
http://www.ucd.ie/issda/data/commissionforenergyregulationcer/, 2012

[51] What consumers want from energy usage data.
https://www.smart-energy.com/industry-sectors/smart-meters/survey-secc-consumers-energy-data/

[52] Why does SVM not perform well for imbalanced data?
https://www.quora.com/Why-does-SVM-not-perform-well-for-imbalanced-data

[53] Ρευματοκλοπή. Ορισμός, διαδικασία εντοπισμού και συνέπειες :
http://www.odigostoupoliti.eu/reymatoklopi-orismos-diadikasia-entopismou-synepeies/

[55] Sampath Kumar V., Jagdish Prasadb, Ravi Samikannuc , "Overview, issues and prevention of energy theft in smart grids and virtual power plants in Indian context" at Department of Electrical Engineering, Botswana International University of Science and Technology (BIUST),Vol. 110, pp. 365-374, November 2017.

[56] Deepa Warudkar, Priyamvada Chandel, B.A.Sawale , "Anti-tamper Features in Electronic energy Meters" at Central Power Research Institute, Bhopal, India, Vol 2, No. 5, May 2014.

[57] Muhammad Shazor , Attiya Baqai , Aamir Ali, "Anti-Electricity Theft System" at Department of Electronics & Biomedical Engineering, Mehran University of Engineering and Technology, Pakistan, March 2011.

[58] Bigyan Basnet, Ramesh Dileep Kumar Appana , "Power Theft Prevention System via Remote Monitoring" at Department of Electrical Engineering, Advanced College of Engineering, Nepal, September 2016.

[59] Mohit Arora, "Prevent tampering in energy meters" at European Documentary Network, February 2009

[60] Joaquim L. Viegas, Paulo R. Esteves, R. Melício, V.M.F. Mendes, Susana M. Vieira, "Solutions for detection of non-technical losses in the electricity grid: A review" at IDMEC, Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal, Vol. 80, pp.1256-1268, December 2017.

[61] Neural Network Theory:
http://leenissen.dk/fann/report/node4.html

[62] Backpropagation Thery:
https://en.wikipedia.org/wiki/Backpropagation

[63] Improving neural network generalization and avoid overfitting:
https://www.mathworks.com/help/deeplearning/ug/improve-neural-network-generalization-and-avoid-overfitting.html;jsessionid=5c146661ceb1f6fd81c3b9aa6ea6

[64] Konstantinos V. Blazakis, Theodoros N. Kapetanakis, and George S. Stavrakakis, “Electricity Theft Detection in Distribution Power Grids Using an Adaptive Neuro Fuzzy Inference System,” submitted to IEEE Transactions on Power Delivery.

[65] Adaptive neuro fuzzy inference system Theory:
https://en.wikipedia.org/wiki/Adaptive_neuro_fuzzy_inference_system

[66] J-S.R. Jang , “ANFIS: adaptive-network-based fuzzy inference system” at IEEE Transactions on Fuzzy Systems, Vol. 23, No. 3, June 2005.

[67] Takaki-Sugeno Fuzzy Model Theory:
http://researchhubs.com/post/engineering/fuzzy-system/takagi-sugeno-fuzzy-model.html

[68] Neuro-Fuzzy Theory:
https://en.wikipedia.org/wiki/Neuro-fuzzy