ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ Μ/Υ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΝΑΥΤΙΛΙΑΣ ΚΑΙ ΒΙΟΜΗΧΑΝΙΑΣ
ΤΜΗΜΑΤΟΣ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ & ΤΕΧΝΟΛΟΓΙΑΣ
ΔΙΑΠΑΝΕΠΙΣΤΗΜΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΤΕΧΝΟ-ΟΙΚΟΝΟΜΙΚΑ ΣΥΣΤΗΜΑΤΑ»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Solar Production Forecasting using Data Analysis and Machine Learning / Πρόβλεψη παραγωγής ενέργειας σε φωτοβολταϊκά χρησιμοποιώντας Ανάλυση Δεδομένων και Μηχανική Μάθηση

Ιωάννης Χουστουλάκης

Επιβλέπων: Χάρης Δούκας, Αναπληρωτής Καθηγητής Ε.Μ.Π.

Υπεύθυνος: Ελισσαίος Β. Σαρμάς, Υποψήφιος Διδάκτωρ Ε.Μ.Π.

Αθήνα, Ιούνιος 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ Μ/Υ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΝΑΥΤΙΛΙΑΣ ΚΑΙ ΒΙΟΜΗΧΑΝΙΑΣ
ΤΜΗΜΑΤΟΣ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ & ΤΕΧΝΟΛΟΓΙΑΣ
ΔΙΑΠΑΝΕΠΙΣΤΗΜΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΤΕΧΝΟ-ΟΙΚΟΝΟΜΙΚΑ ΣΥΣΤΗΜΑΤΑ»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Solar production forecasting using Data Analysis and Machine Learning / Πρόβλεψη παραγωγής ενέργειας σε φωτοβολταϊκά χρησιμοποιώντας Ανάλυση Δεδομένων και Μηχανική Μάθηση

Ιωάννης Χουστουλάκης

Επιβλέπων: Χάρης Δούκας, Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 7η Ιουνίου 2022.

Χάρης Δούκας,	Ιωάννης Ψαρράς	Δημήτρης Ασκούνης
Αναπληρωτής Καθηγητής	Καθηγητής	Καθηγητής

Αθήνα, Ιούνιος 2022

Ιωάννης Χουστουλάκης

Διπλωματούχος Μηχανολόγος Μηχανικός Ε.Μ.Π.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Η εργασία αυτή εξετάζει τις τεχνικές ανάλυσης δεδομένων με τις οποίες μπορεί να γίνει πρόβλεψη της παραγόμενης ενέργειας και εντοπισμός αστοχιών σε μια φωτοβολταϊκή μονάδα. Εξετάζονται οι μέθοδοι που χρησιμοποιήθηκαν σε πρόσφατες έρευνες και παρουσιάζεται μια προτεινόμενη μεθοδολογία για την πρόβλεψη της παραγωγή.

Με τη χρήση αισθητήρων, ιστορικών και μετεωρολογικών δεδομένων, σε συνδυασμό με υπολογιστικές μεθόδους και μεθόδους μηχανικής μάθησης, μπορούν να γίνουν αναλύσεις που κάνουν προβλέψεις για την παραγωγή των φωτοβολταϊκών μονάδων σε διαφορετικούς χρονικούς ορίζοντες. Οι προβλέψεις αυτές είναι σημαντικές για τη διαχείριση της στοχαστικότητας της ηλιακής ενέργειας ώστε να μπορεί αυτή να αξιοποιηθεί σε μεγαλύτερο βαθμό στο ηλεκτρικό δίκτυο. Πραγματοποιήθηκε βιβλιογραφική έρευνα στις μεθόδους που χρησιμοποιούνται και παρουσιάζονται οι πρόσφατες δημοσιεύσεις που πειραματίζονται πάνω στην τεχνολογία αυτή. Εξετάζονται οι στατιστικές μέθοδοι του ARMA, της Παλινδρόμησης, της Εκθετικής Εξομάλυνσης και της Φυσικής Μεθόδου. Εξετάζονται επίσης οι μέθοδοι Μηχανικής Μάθησης και της Βαθιάς Μάθησης (Deep Learning), συμπεριλαμβανομένων των Νευρωνικών Δικτυών, των Δέντρων Αποφάσεων, των Μηχανών Διανυσμάτων Υποστήριξης και υβριδικών μεθόδων.

Αντίστοιχα, με μετρήσεις από διάφορους αισθητήρες και την ανάλυση των μετρήσεων αυτών στον υπολογιστή, μπορεί να γίνει εντοπισμός αστοχιών στα φωτοβολταϊκά στοιχεία ώστε να αποφευχθούν περαιτέρω ζημίες, να γίνουν οι απαραίτητες διορθώσεις και να εξασφαλιστεί η μέγιστη παραγωγικότητα της εγκατάστασης. Διεξήχθη αντίστοιχη έρευνα στις μεθόδους με τις όποιες γίνεται ο εντοπισμός αστοχιών και αναφέρονται δημοσιεύσεις που πειραματίζονται στο αντικείμενο αυτό. Εξετάζονται οι οπτικές μέθοδοι, οι μέθοδοι ηλεκτρικών χαρακτηριστικών, οι μέθοδοι απεικόνισης, οι μέθοδοι τεχνητής νοημοσύνης, οι μέθοδοι που στηρίζονται στη διάγνωση από εξειδικευμένες συσκευές, η προγνωστική συντήρηση με χρήση αισθητήρων σε πραγματικό χρόνο, καθώς και υβριδικές μέθοδοι.

Στην προτεινόμενη μεθοδολογία πρόβλεψης παραγωγής, αναπτύχθηκε λογισμικό στην γλώσσα Python το οποίο πραγματοποιεί προβλέψεις για την παραγωγή αξιοποιώντας μετεωρολογικά δεδομένα και ιστορικά δεδομένα παραγωγής από ένα ανώνυμο Case Study. Πραγματοποιήθηκε προ-επεξεργασία στα δεδομένα, διορθώνοντας τις εσφαλμένες τιμές και γεμίζοντας τις κενές τιμές. Έγινε προσαρμογή ώστε τα δεδομένα να μπορούν να εισαχθούν στους αλγορίθμους μηχανικής μάθησης. Χρησιμοποιήθηκαν οι μέθοδοι του Δένδρου Αποφάσεων, των Μηχανών Διανυσμάτων Υποστήριξης, της Γραμμικής Παλινδρόμησης και υβριδικός συνδυασμός αυτών. Αξιοποιήθηκε το λογισμικό scikit-learn. Χρησιμοποιήθηκε η μέθοδος VotingRegressor για τον συνδυασμό μεθόδων και η μέθοδος GridSearchCV για τη βελτιστοποίηση των παραμέτρων κάθε μοντέλου. Πραγματοποιήθηκαν εκτεταμένες δοκιμές στις οποίες προσδιορίστηκε η βέλτιστη επιλογή μεταβλητών εισόδου και η βέλτιστη επιλογή υπερπαραμέτρων σε κάθε μοντέλο. Τα μοντέλα δοκιμάστηκαν με δυο διαφορετικά μεγέθη δεδομένων εκπαίδευσης και συγκρίθηκε η ακρίβειά τους. Στην πρώτη δοκιμή τα δεδομένα

εκπαίδευσης ήταν το σύνολο των δεδομένων εκτός από την τελευταία εβδομάδα και στη δεύτερη δοκιμή αποτελούσαν το 50% των δεδομένων.

Λέξεις κλειδιά: Ανανεώσιμες Πηγές Ενέργειας, Παραγωγή Φωτοβολταϊκών, Εντοπισμός αστοχιών, Βραχυπρόθεσμες προβλέψεις, Μηχανική Μάθηση, Ανάλυση δεδομένων

Abstract

This thesis examines the data analysis techniques with which energy production forecasting and fault detection can be performed on a photovoltaic unit. Methods used in recent research are examined and a proposed methodology for production forecasting is presented.

Using sensors, historical and meteorological data, combined with computational methods and machine learning methods, analysis can be performed to predict the productivity of solar cells in various time ranges. These predictions are important in order to be able to respond to the variation of solar energy in order for it to be usable on the electrical grid to a larger degree. Bibliographical research was done on the methods used for forecasting and recent publications experimenting on this technology are presented. Statistical Methods are examined, which includes ARMA, Regression, Exponential Smoothing Method and the Physical Method. Machine Learning and Deep Learning Methods are also examined, which includes Neural Networks, Decision Trees, Support Vector Machines and hybrid methods.

Similarly, by collecting measurements from various sensors and analyzing these measurements on a computer, fault detection can be performed on Photovoltaic units in order to prevent further damage, perform the necessary maintenance and ensure the maximum productivity of the installation. Bibliographical research was done on the available fault detection methods and the publications experimenting on that field. The methods examined included the visual method, electrical characteristics methods, imaging techniques, artificial intelligence, device-based methods, predictive maintenance through real-time sensors and hybrid methods.

In the proposed methodology for production forecasting, software was developed in Python, which can make production forecasts using meteorological and historical production data from an anonymized case study. The data was pre-processed, correcting erroneous values and filling in missing values. The data was formatted and adjusted in order to be inputted to the machine learning algorithms. The methods of Decision Tree, Support Vector Machines, Linear Regression and hybrid combination of them were used. The scikit-learn software was utilized. The VotingRegressor method was used for the ensemble methods and the GridSearchCV method was used for the optimization of the parameters in each model. Extended tests were performed to determine the optimal selection of input features as well as the optimal hyperparameters for each model. The models were tested with two different train set sizes to compare their accuracy. During the first test the train data were composed of all available data except the final week and during the second test they were composed of 50% of the data.

Keywords: Renewable Energy Sources, Photovoltaic Production, Fault detection, Short-term Forecasting, Machine Learning, Data Analysis

Contents

Ευρεία Περίληψη

Κεφάλαιο 1: Εισαγωγή

Η ανάγκη για μορφές ενέργειας που να είναι φιλικότερες στο περιβάλλον έχει αναδείξει την παραγωγή ενέργειας από φωτοβολταϊκά στοιχεία κρίσιμη για την Ελληνική οικονομία.

Μέχρι πρόσφατα που η ενέργεια προερχόταν μόνο από θερμοηλεκτρικά εργοστάσια, η διαθέσιμη ηλεκτρική ισχύς στο δίκτυο είχε μια καθορισμένη τιμή που οριζόταν από ανθρώπους. Με τα νέα δεδομένα της ηλεκτροπαραγωγής από φωτοβολταϊκά στοιχεία και ανεμογεννήτριες, αυτό έχει αλλάξει. Η ηλιακή ακτινοβολία, όπως και η ταχύτητα του ανέμου, αποτελούν στοχαστικές μεταβλητές, απρόβλεπτες σε μεγάλο βαθμό κατά τη διάρκεια της ημέρας.

Κατά συνέπεια, είναι καταρχήν απρόβλεπτο πόση διαθέσιμη ισχύ θα έχει το ηλεκτρικό δίκτυο, και αν η ισχύς αυτή επαρκεί κάθε δεδομένη στιγμή για να καλύψει τη ζήτηση. Κατά συνέπεια, χωρίς κάποιο σύστημα υποστήριξης, η ηλεκτροπαραγωγή με Ανανεώσιμες Πηγές Ενέργειας αφήνει μια χώρα εκτεθειμένη σε διαρκείς απώλειες παροχής ρεύματος και πτώσεις τάσεις. Η πρόβλεψη της παραγωγής εξασφαλίζει ότι το δίκτυο θα λειτουργεί σταθερότερα σε διαφορετικές συνθήκες. Υπάρχει συνεπώς ανάγκη να γίνονται εκ των προτέρων εκτιμήσεις για την ποσότητα της παραγόμενης ενέργειας. Αυτό δίνει τη δυνατότητα να ληφθούν μέτρα προσαρμογής της ζήτησης του ρεύματος ώστε να ανταπεξέρχεται στην προσφορά. Τέτοιο μέτρο είναι η διακύμανση της τιμής του ρεύματος κατά τη διάρκεια της ημέρας. Η πρόβλεψη της παραγωγής δίνει επίσης τη δυνατότητα στο χειριστή του ηλεκτρικού δικτύου να αξιοποιήσει διαφορετικές πηγές ενέργειας που έχει στη διάθεσή του ώστε να ανταποκριθεί σε αυξήσεις ή μειώσεις στη διαθέσιμη ισχύ (1).

Η πρόγνωση καιρού είναι απαραίτητη για τον προσδιορισμό της διακύμανσης της παραγωγής, αλλά όχι καθ' εαυτή αρκετή. Η έρευνα απέδειξε ότι τα μετεωρολογικά δεδομένα δεν έχουν γραμμική σχέση με την παραγωγή ενέργειας. Τα δεδομένα της πρόγνωσης καιρού χρειάζονται περαιτέρω επεξεργασία ώστε να γίνει εκτίμηση της μελλοντικής παραγωγής.

Επίσης, η παροχή της ηλεκτρικής ενέργειας από τα φωτοβολταϊκά μπορεί να μειωθεί και από αστοχίες που μπορούν να προκύψουν στις διατάξεις των φωτοβολταϊκών καθώς και στους συλλέκτες. Τέτοιες αστοχίες πρέπει να προσδιοριστούν και να διορθωθούν εγκαίρως ώστε να εξασφαλιστεί η ομαλή λειτουργία του δικτύου. Είναι σημαντικό να εντοπίζονται εγκαίρως τα σφάλματα ώστε να ελαχιστοποιηθεί το κόστος συντήρησης και να μεγιστοποιηθεί η παραγωγή. Η πρόβλεψη της παραγωγής μπορεί επίσης να συνεισφέρει στον εντοπισμό αστοχιών.

Τόσο στο πρόβλημα της στοχαστικότητας των μετεωρολογικών δεδομένων όσο και στον εντοπισμό των αστοχιών ανταπεξέρχεται η ανάλυση δεδομένων και η μηχανική μάθηση.

Με τη χρήση υπολογιστικών μεθόδων, τα μετεωρολογικά δεδομένα μπορούν να αξιοποιηθούν με τέτοιο τρόπο ώστε να γίνει μια προσεγγιστική αντιστοίχιση τους με την παραγωγή ενέργειας. Επίσης, τα δεδομένα που συλλέγονται από αισθητήρες μπορούν να συγκριθούν με τις τιμές που έδειχναν οι αισθητήρες όταν είχε παρουσιαστεί ή επρόκειτο να παρουσιαστεί συγκεκριμένο σφάλμα, ώστε να γίνει εγκαίρως ο εντοπισμός και η διόρθωσή του όταν επανεμφανιστεί.

Στόχος και αντικείμενο της διπλωματικής

Υπάρχει μια ποικιλία μεθόδων που χρησιμοποιούνται στην πρόβλεψη της παραγωγής και τον εντοπισμό αστοχιών στα φωτοβολταϊκά. Η απόδοση και η ακρίβειά τους ποικίλουν έντονα μεταξύ διαφορετικών μελετών περίπτωσης.

Το αντικείμενο της διπλωματικής αυτής είναι να εξεταστεί η εφαρμογή των διαφόρων μεθόδων πρόβλεψης και να προσφέρει ένα πλαίσιο με το οποίο να μπορεί να γίνει σύγκριση στις πρόσφατες έρευνες για την εφαρμογή και την επίδοση των μεθόδων αυτών.

Αντίστοιχη έρευνα διεξήχθη στις μεθόδους εντοπισμού σφαλμάτων, ώστε να υπάρχει μια βάση αναφοράς για το ποιες μέθοδοι είναι αποτελεσματικές στον εντοπισμό διαφορετικών τύπων αστοχιών. Ερευνώνται επίσης οι πρόσφατες δημοσιεύσεις στο αντικείμενο.

Επιπλέον, η διπλωματική αυτή περιλαμβάνει μια προτεινόμενη μεθοδολογία για πρόβλεψη παραγωγής φωτοβολταϊκών, η οποία αναπτύχθηκε στην Python. Αναπτύχθηκαν διάφορα μοντέλα, τα οποία βελτιστοποιήθηκαν και στη συνέχεια συγκρίθηκαν στην απόδοσή τους, χρησιμοποιώντας ανώνυμα δεδομένα για να γίνει μια μελέτη περίπτωσης. Η απόδοση της προτεινόμενης μεθοδολογίας συγκρίθηκε στη συνέχεια με τα μοντέλα που χρησιμοποιούνται στη βιβλιογραφία.

Συνεισφορά και αξία της διπλωματικής

Η διπλωματική αυτή παρέχει μια καινούρια μεθοδολογία σχεδιασμένη να συγκρίνει διαφορετικά μοντέλα πρόβλεψης βάσει της απόδοσής τους και στη συνέχεια να παράγει προβλέψεις χρησιμοποιώντας ένα επιλεγμένο μοντέλο. Παρέχει μια βάση που επιτρέπει τη βελτιστοποίηση κάθε μοντέλου ώστε να μεγιστοποιήσει την απόδοσή του. Η βάση αυτή επιτρέπει επίσης να δοκιμαστεί κάθε μοντέλο με διαφορετικά δεδομένα.

Καταρχήν τα δεδομένα υπέστησαν κατάλληλη προ-επεξεργασία. Αναπτύχθηκε λογισμικό το οποίο διορθώνει τις εσφαλμένες τιμές και γεμίζει τις κενές τιμές στα δεδομένα. Στη συνέχεια, τα δεδομένα χρησιμοποιήθηκαν σε πέντε διαφορετικά μοντέλα, χρησιμοποιώντας Δέντρα Αποφάσεων, Μηχανές Διανυσμάτων Υποστήριξης , Γραμμική Παρεμβολή και δύο υβριδικά μοντέλα που χρησιμοποιούν συνδυασμό των προηγούμενων. Έγιναν εκτεταμένες δοκιμές για να προσδιοριστεί η βέλτιστη επιλογή των μεταβλητών εισόδου. Στη συνέχεια έγιναν περαιτέρω δοκιμές για να βελτιστοποιηθούν οι υπερπαράμετροι κάθε μοντέλου. Η απόδοση των μοντέλων σε κάθε δοκιμή καταγράφηκε και συγκρίθηκε με τα προηγούμενα αποτελέσματα με τη χρήση εξειδικευμένων δεικτών, συμπεριλαμβανομένης της τετραγωνικής ρίζας του μέσου τετραγωνικού σφάλματος (RMSE), του μέσου απόλυτου σφάλματος (MAE) και του συντελεστή προσδιορισμού (R-Squared). Δυο διαφορετικά μεγέθη

δεδομένων χρησιμοποιήθηκαν ως είσοδος στα μοντέλα. Η απόδοση των μοντέλων συγκρίθηκε με αυτόν τον τρόπο και για βραχυπρόθεσμες προβλέψεις.

Επιπλέον, η διπλωματική αυτή παρέχει μια γενική εικόνα για τις μεθόδους που χρησιμοποιούνται στην πρόβλεψη παραγωγής και τον εντοπισμό σφαλμάτων στα φωτοβολταϊκά. Περιγράφονται οι πιο συνήθεις μέθοδοι, οι βασικές τους έννοιες, η τεχνική ορολογία, ο τρόπος λειτουργίας, τα πλεονεκτήματα και μειονεκτήματα καθεμίας, καθώς και η συμβατότητά τους σε διαφορετικές εφαρμογές.

Παρουσιάζεται μια εκτεταμένη έρευνα στις δημοσιεύσεις σχετικά με τη μεθοδολογία στις προβλέψεις παραγωγής των φωτοβολταϊκών. Για κάθε δημοσίευση αναφέρεται η μέθοδος που χρησιμοποιήθηκε, οι μεταβλητές εισόδου, το χρονικό εύρος της πρόβλεψης, άλλες τυχόν λεπτομέρειες της εφαρμογής, μαζί με την απόδοση και τα αποτελέσματα κάθε περίπτωσης. Αυτό παρέχει ένα σημείο αναφοράς για μελλοντική έρευνα, καθώς μπορεί να συγκριθεί εύκολα η απόδοση πολλών διαφορετικών εφαρμογών.

Αντίστοιχη έρευνα διεξήχθη στις μεθόδους εντοπισμού αστοχιών. Αναφέρονται οι βασικές έννοιες, ο τρόπος λειτουργίας και η συμβατότητά κάθε μεθόδου για τον εντοπισμό διαφορετικών σφαλμάτων, καθώς και τα αποτελέσματα πρόσφατων ερευνών στις μεθόδους αυτές.

Δομή της διπλωματικής

Η διπλωματική έχει 6 κεφάλαια και 2 παραρτήματα.

Το κεφάλαιο 1 περιέχει μια εισαγωγή και τους λόγους που υπάρχει ανάγκη για έρευνα στο αντικείμενο αυτό. Εξηγείται ο στόχος και το αντικείμενο της διπλωματικής, μαζί με τη συνεισφορά και την αξία της. Τέλος, εξηγείται η δομή της διπλωματικής.

Το κεφάλαιο 2 περιέχει μια ανάλυση στις μεθόδους πρόβλεψης παραγωγής που περιγράφονται στη βιβλιογραφία, μαζί με μια έρευνα στις πρόσφατες δημοσιεύσεις που έγιναν στην πρόβλεψη παραγωγής φωτοβολταϊκών, μαζί με τις μεθόδους και την απόδοση κάθε εφαρμογής. Αναλύονται η μέθοδος εμμονής, οι στατιστικές μέθοδοι και οι μέθοδοι μηχανικής μάθησης.

Το κεφάλαιο 3 περιγράφει τις μεθόδους εντοπισμού αστοχιών στους ηλιακούς συλλέκτες. Εξηγείται ο τρόπος λειτουργίας κάθε μεθόδου και το είδος των αστοχιών που μπορεί να εντοπίσει. Εξετάζονται οι εφαρμογές σε πρόσφατες δημοσιεύσεις. Περιγράφονται οι οπτικές μέθοδοι, οι μέθοδοι ηλεκτρικών χαρακτηριστικών, οι μέθοδοι απεικόνισης, οι μέθοδοι τεχνητής νοημοσύνης, οι μέθοδοι που βασίζονται σε εξειδικευμένες συσκευές, οι μέθοδοι που χρησιμοποιούν αισθητήρες σε πραγματικό χρόνο και οι υβριδικές μέθοδοι.

Το κεφάλαιο 4 παρουσιάζει την προτεινόμενη μεθοδολογία. Γίνονται προβλέψεις χρησιμοποιώντας ιστορικά δεδομένα παραγωγής και μετεωρολογικά δεδομένα, με τη χρήση Δέντρων αποφάσεων, Μηχανών Διανυσμάτων Υποστήριξης, Γραμμικής Παλινδρόμησης και δύο υβριδικών μοντέλων που χρησιμοποιούν συνδυασμούς των προηγούμενων. Εξηγείται ο τρόπος λειτουργίας των επιλεγμένων μοντέλων. Αναλύεται η σχετική θεωρία με κατάλληλα παραδείγματα, διαγράμματα και μαθηματικούς τύπους. Εξηγείται η μέθοδος προ-

επεξεργασίας των δεδομένων. Εξετάζεται η συσχέτιση μεταξύ των διαθέσιμων δεδομένων και της μετρούμενης ισχύος εξόδου ώστε να γίνει εκτίμηση του ποιες μεταβλητές θα είναι χρήσιμες κατά τη διάρκεια των δοκιμών.

Το κεφάλαιο 5 αναφέρει αναλυτικά τις δοκιμές που έγιναν και τα αποτελέσματα που παράχθηκαν. Αυτό περιλαμβάνει μια σειρά δοκιμών με σκοπό να προσδιοριστεί η βέλτιστη επιλογή μεταβλητών εισόδου και μια σειρά δοκιμών με σκοπό να προσδιοριστούν οι βέλτιστες τιμές των υπερ-παραμέτρων για κάθε μοντέλο. Η βελτιστοποίηση έγινε καταρχήν χειροκίνητα, αναφέροντας την ακρίβεια κάθε μοντέλου σε κάθε δοκιμή και επιλέγοντας τις καλύτερες παραμέτρους χειροκίνητα. Στη συνέχεια έγιναν δοκιμές αυτοματοποιημένα, με τη βοήθεια του GridSearchCV, από τα εργαλεία του scikit-learn. Έγιναν δοκιμές με δύο διαφορετικά μεγέθη δεδομένων εισόδου. Αναλύονται λεπτομερώς τα αποτελέσματα .

Το κεφάλαιο 6 παρουσιάζει τα συμπεράσματα από τις δοκιμές και προτείνει μελλοντικές εργασίες με τις οποίες μπορεί να συνεχιστεί η έρευνα αυτής της διπλωματικής.

Το παράρτημα Α παρουσιάζει τον κώδικα που αναπτύχθηκε για να εφαρμοστεί η προτεινόμενη μεθοδολογία. Περιλαμβάνει στατιστική ανάλυση και μηχανική μάθηση με τη χρήση του scikit-learn. Ο κώδικας συνοδεύεται από σχόλια που εξηγούν τις λειτουργίες του.

Το παράρτημα Β παρουσιάζει τον κώδικα που χρησιμοποιήθηκε για την προ-επεξεργασία των δεδομένων, που περιλάμβανε τις απαραίτητες προσαρμογές στη μορφοποίηση, το γέμισμα των κενών τιμών και τη διόρθωση των εσφαλμένων τιμών που προκλήθηκαν από το θόρυβο στο ηλεκτρικό δίκτυο.

Κεφάλαιο 2

Στο κεφάλαιο αυτό αναλύονται οι μέθοδοι πρόβλεψης που υπάρχουν στη βιβλιογραφία. Εξηγείται η μέθοδος εμμονής και ο τρόπος που χρησιμοποιείται. Αναλύονται οι στατιστικές μέθοδοι, συμπεριλαμβανομένης της ARMA, της παλινδρόμησης, της εκθετικής εξομάλυνσης και της φυσικής μεθόδου. Εξηγείται ο τρόπος λειτουργίας τους και οι συνθήκες στις οποίες αποδίδουν καλύτερα.

$$X(t) = \sum_{i=1}^{p} \alpha_i X(t-i) + \sum_{j=1}^{q} \beta_j e(t-j)$$

Εξίσωση 1 – Μαθηματικός τύπος ARMA (2):

Περιγράφονται επίσης οι μέθοδοι μηχανικής μάθησης και βαθιάς μάθησης, όπως τα Νευρωνικά Δίκτυα, οι Μηχανές Διανυσμάτων Υποστήριξης και οι υβριδικές μέθοδοι. Γίνεται εκτεταμένη ανάλυση στις πρόσφατες έρευνες που χρησιμοποιούν διαφορετικές μεθόδους πρόβλεψης, αναφέροντας την ακρίβεια καθεμίας.

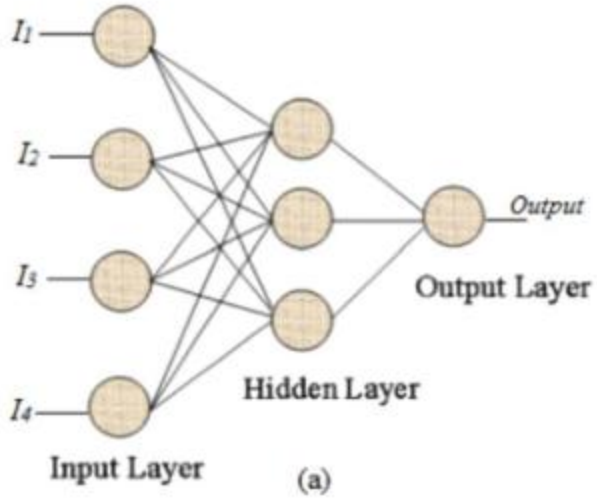

Figure 1 – Βασική δομή Νευρωνικού Δικτύου (3)

Το κεφάλαιο αυτό οδήγησε στα παρακάτω συμπεράσματα:

- Οι μέθοδοι μηχανικής μάθησης έχουν μεγαλύτερη ακρίβεια από τις καθαρά στατιστικές μεθόδους.
- Οι πιο δημοφιλείς μέθοδοι μηχανικής μάθησης και βαθιάς μάθησης είναι οι διάφορες μορφές των Νευρωνικών Δικτύων, ακολουθούμενων από τις Μηχανές Διανυσμάτων Υποστήριξης και τις υβδρικές μεθόδους.
- Στη βιβλιογραφία η μέγιστη ακρίβεια επιτεύχθηκε με τη χρήση Νευρωνικών Δικτύων (ANN), αλλά σε άλλες περιπτώσεις η μέθοδος αυτή είχε κακή απόδοση.
- Οι περισσότερες προβλέψεις στη βιβλιογραφία είναι βραχυπρόθεσμες, μεταξύ τιμών κάτω της μίας ώρας μέχρι και μια ημέρα.
- Η ακρίβεια των μοντέλων επηρεάζεται έντονα από τον καιρό. Ο ηλιόλουστος καιρός οδηγεί σε πολλές περιπτώσεις σε πιο ακριβείς προβλέψεις.
- Η χρήση διαφορετικών μετεωρολογικών δεδομένων ποικίλει έντονα στη βιβλιογραφία. Κάποιοι ερευνητές βασίζονται αποκλειστικά στα ιστορικά δεδομένα παραγωγής και άλλοι χρησιμοποιούν διάφορες παραμέτρους, όπως την κάλυψη από τα σύννεφα ή τη θερμοκρασία, για να κάνουν προβλέψεις. Το συμπέρασμα ήταν ότι, για την έρευνα σε αυτή τη διπλωματική, έπρεπε να δοκιμαστούν διαφορετικές μεταβλητές εισόδου ώστε να προσδιοριστεί η μέγιστη απόδοση. Ο λόγος είναι ότι στη βιβλιογραφία δεν υπήρχε συμπέρασμα του ποιες μεταβλητές είναι χρήσιμες σε όλες τις περιπτώσεις.

Κεφάλαιο 3

Σε αυτό το κεφάλαιο εξετάζονται οι μέθοδοι εντοπισμού αστοχιών στα φωτοβολταϊκά. Περιγράφεται ο τρόπος λειτουργίας κάθε μεθόδου και το είδος των αστοχιών που μπορεί να εντοπίσει. Ερευνήθηκαν και εξηγούνται εφαρμογές κάθε μεθόδου σε πρόσφατες έρευνες.

Εξηγείται η οπτική μέθοδος, που συνήθως περιλαμβάνει τον εντοπισμό αστοχιών με ανθρώπινη παρατήρηση. Εξηγούνται τα πρότυπα της διαδικασίας, τα είδη των λαθών που μπορούν να εντοπιστούν και τα μειονεκτήματα της μεθόδου.

Αναλύονται οι μέθοδοι που βασίζονται στην ανάλυση ηλεκτρικών χαρακτηριστικών των φωτοβολταϊκών εγκαταστάσεων. Αυτό περιλαμβάνει την MBDM, την RDM, τη μέτρηση I-V, την OSA, PLA και CDI. Αναφέρονται τα αποτελέσματα πρόσφατων ερευνών.

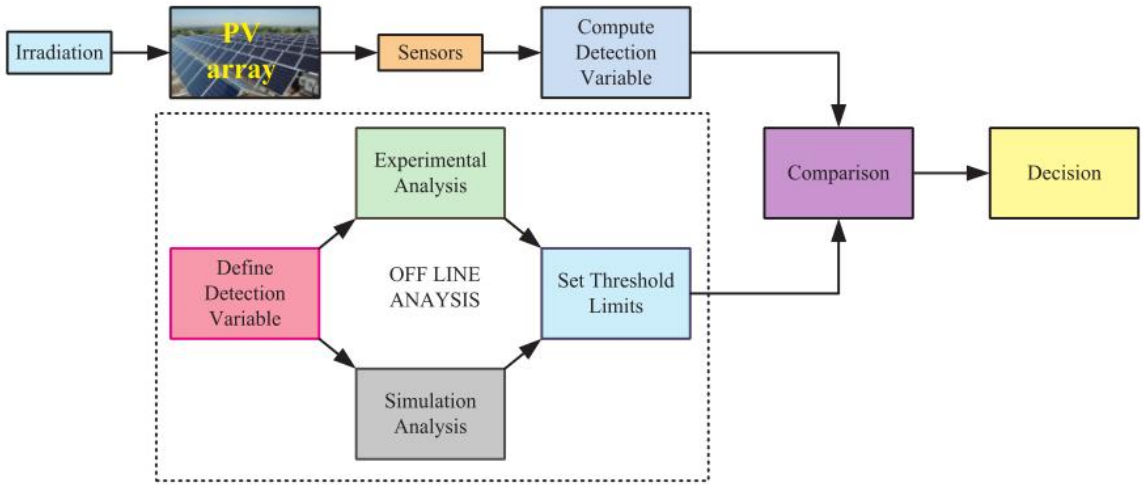

Figure 2 – Διαδικασία RDM για εντοπισμό αστοχιών (4)

Περιγράφονται οι τεχνικές απεικόνισης, που περιλαμβάνουν τη χρήση υπερύθρων, υπερήχων, ηλεκτροφωταύγεια και τη μέθοδο Lock in Thermography.

Αναλύονται οι μέθοδοι που χρησιμοποιούν τεχνητή νοημοσύνη και δεδομένα αισθητήρων για να εντοπίσουν αστοχίες. Αυτό περιλαμβάνει μεθόδους μηχανικής μάθησης, όπως Μηχανές Διανυσμάτων υποστήριξης, Deep Learning και συνελικτικά νευρωνικά δίκτυα. Περιγράφονται οι μέθοδοι που χρησιμοποιούν εξειδικευμένες συσκευές και τα είδη των συσκευών που υπάρχουν για τον εντοπισμό διαφορετικών αστοχιών. Εξηγούνται οι μέθοδοι που χρησιμοποιούν αισθητήρες που παρακολουθούνται σε πραγματικό χρόνο, ώστε να εντοπιστεί ένα πιθανό σφάλμα προτού προκύψει. Περιγράφονται τέλος και οι υβδρικές μέθοδοι που συνδυάζουν έναν αριθμό τεχνικών από τις παραπάνω.

Το κεφάλαιο αυτό κατέληξε στα παρακάτω συμπεράσματα:

- Ο οπτικός έλεγχος είναι χρήσιμος σαν προκαταρκτικός για να εντοπιστεί αν χρειάζεται περαιτέρω έλεγχος.
- Οι μετρήσεις ηλεκτρικών χαρακτηριστικών εντοπίζουν αστοχίες ανοικτού και κλειστού κυκλώματος, hotspots, αστοχίες γείωσης, ηλεκτρικού τόξου, σκίασης και σφάλματα γήρανσης.

- Οι μέθοδοι απεικόνισης μπορούν να εντοπίσουν ρωγμές και μικρό-ρωγμές, αστοχίες διασύνδεσης και καλωδίωσης, hotspots, αύξηση αντίστασης σειράς, ελλαττωματικές διόδους παράκαμψης, μη-ομοιόμορφο ρεύμα και διαφορές στην ενεργειακή απόδοση μεταξύ συλλεκτών.
- Η μηχανική μάθηση αξιοποιεί δεδομένα τόσο από απεικόνιση όσο και ηλεκτρικές μετρήσεις και μπορεί να εντοπίσει και αστοχίες που έχουν ήδη προκύψει αλλά και αστοχίες που θα προκύψουν στο μέλλον.
- Για κάθε κατηγορία αστοχιών υπάρχει μια εξειδικευμένη συσκευή που μπορεί να την εντοπίσει.

Κεφάλαιο 4

Σε αυτό το κεφάλαιο παρουσιάζεται η προτεινόμενη μεθοδολογία. Αναπτύχθηκε ένα πρόγραμμα στην Python με το οποίο έγιναν προβλέψεις για την παραγωγή με τη χρήση ιστορικών δεδομένων παραγωγής και μετεωρολογικών δεδομένων.

Χρησιμοποιήθηκαν οι μέθοδοι του Δέντρου Αποφάσεων, των Μηχανών Διανυσμάτων Υποστήριξης, της Γραμμικής Παλινδρόμησης και δύο υβριδικών συνδυασμών των προηγουμένων. Εξηγείται ο τρόπος λειτουργίας των μοντέλων αυτών αναλύοντας τη σχετική θεωρία και παρουσιάζοντας παραδείγματα, διαγράμματα και τους μαθηματικούς τύπους κάθε μοντέλου.

Αναλύεται βήμα προς βήμα η μέθοδος προ-επεξεργασίας των δεδομένων. Τα δεδομένα έπρεπε να καθαριστούν από κάποια σφάλματα μέτρησης και να μορφοποιηθούν κατάλληλα για να είναι χρησιμοποιήσιμα από τα μοντέλα μηχανικής μάθησης.

Ελέγχθηκε η συσχέτιση μεταξύ των διαθέσιμων δεδομένων και της μετρούμενης ισχύος εξόδου ώστε να εκτιμηθεί ποιες μεταβλητές θα είναι χρήσιμες σαν δεδομένα εισόδου κατά τη διάρκεια των δοκιμών.

Επιλέχθηκαν τα μοντέλα του Δέντρου Αποφάσεων, των Μηχανών Διανυσμάτων Υποστήριξης και της Γραμμικής Παλινδρόμησης για διάφορους λόγους. Ένας σημαντικός παράγοντας ήταν τα διαθέσιμα εργαλεία του scikit-learn που έχουν καλή υποστήριξη στο διαδίκτυο και προσφέρουν πρόσβαση στα μοντέλα αυτά. Ένας άλλος λόγος για τη χρήση του Δέντρου Αποφάσεων και των Μηχανών Διανυσμάτων Υποστήριξης ήταν η ποικιλία των υπερπαραμέτρων που διαθέτουν, καθώς αυτό δίνει τη δυνατότητα να βελτιστοποιηθούν τα αποτελέσματα. Υπάρχει επίσης έλλειψη στη βιβλιογραφία από εφαρμογές Δέντρου Αποφάσεων και Γραμμικής παλινδρόμησης για προβλέψεις παραγωγής φωτοβολταϊκών.

Ένας ακόμα λόγος ήταν η ανάγκη να ελεγχθεί η γραμμικότητα του εξεταζόμενου προβλήματος. Από άποψη φυσικής, η σχέση μεταξύ άμεσης ηλιακής ακτινοβολίας και παραγόμενης ισχύος από τα φωτοβολταϊκά θα έπρεπε θεωρητικά να είναι γραμμική. Αν προστίθεντο και άλλες μεταβλητές εισόδου, αυτό θα άλλαζε. Οι Μηχανές Διανυσμάτων Υποστήριξης μπορούν να λειτουργήσουν και με γραμμικό και με μη γραμμικό kernel. Το Δέντρο Αποφάσεων είναι μη-γραμμικός αλγόριθμος, ενώ η Γραμμική Παλινδρόμηση είναι γραμμική.

Με διαφορετικές δοκιμές, συγκρίνοντας τις μεθόδους αυτές, μπορεί να βρεθεί η βέλτιστη λύση είτε το πρόβλημα είναι τελικά γραμμικό είτε όχι. Ο συνδυασμός ενός γραμμικού και ενός μη γραμμικού μοντέλου μπορεί ενδεχομένως να αντιμετωπίσει ένα πρόβλημα που περιέχει και γραμμικές και μη-γραμμικές τάσεις.

Η ηλιακή ακτινοβολία των προηγούμενων ωρών έδειξε πολύ υψηλή συσχέτιση με την παραγωγή ενέργειας. Το αποτέλεσμα αυτό ταιριάζει με το γεγονός ότι η αυτοσυσχέτιση της ακτινοβολίας ήταν υψηλή μέχρι και 4 ώρες πριν. Αυτό οδήγησε στο συμπέρασμα ότι οι τιμές της ηλιακής ακτινοβολίας των περασμένων 4 με 5 ωρών πρέπει να εξεταστούν στις δοκιμές. Η κάλυψη από σύννεφα είχε σχεδόν μηδενική συσχέτιση με την παραγωγή και εξαιρέθηκε από τις δοκιμές.

Η χρονοσειρά των δεδομένων εμφάνισε έναν εποχιακό παράγοντα, πράγμα που οδήγησε στο συμπέρασμα ότι στοιχεία της ημερομηνίας, όπως η εποχή ή ο μήνας, πρέπει να συμπεριληφθούν στις δοκιμές ως μεταβλητές εισόδου.

Κεφάλαιο 5

Σε αυτό το κεφάλαιο αναφέρονται οι δοκιμές που έγιναν στα πλαίσια αυτής της έρευνας και σχολιάζονται τα αποτελέσματα. Έγινε καταρχήν μια σειρά δοκιμών για να προσδιοριστεί η βέλτιστη επιλογή μεταβλητών εισόδου για κάθε μοντέλο. Η απόδοση κάθε μοντέλου εμφανίζεται για κάθε δοκιμή με τους ακόλουθους δείκτες: 1) Την τετραγωνική ρίζα του μέσου τετραγωνικού σφάλματος, 2) Το μέσο απόλυτο σφάλμα και 3) Το συντελεστή προσδιορισμού (R-Squared), με δύο διαφορετικούς τρόπους υπολογισμού.

Έγιναν στη συνέχεια δοκιμές για τη βελτιστοποίηση των υπερπαραμέτρων κάθε μοντέλου. Η βελτιστοποίηση έγινε καταρχήν χειροκίνητα, αναφέροντας την ακρίβεια κάθε μοντέλου και επιλέγοντας τις καλύτερες παραμέτρους με το χέρι, καθώς και αυτόματα, με τη χρήση της λειτουργίας GridSearchCV του scikit-learn.

Χρησιμοποιήθηκαν οι καλύτερες παράμετροι που βρέθηκαν από τις προηγούμενες δοκιμές για να δοκιμαστούν τα μοντέλα με μικρότερο πλήθος δεδομένων εισόδου, εξετάζοντας την απόδοσή τους όταν υπάρχουν λιγότερες πληροφορίες. Τα αποτελέσματα συγκρίθηκαν με αυτά των προηγούμενων δοκιμών.

Τα συμπεράσματα από τις δοκιμές ήταν τα ακόλουθα:

- Τα μετεωρολογικά δεδομένα εκτός της ηλιακής ακτινοβολίας δεν φάνηκαν να είναι χρήσιμα για να γίνουν προβλέψεις στη συγκεκριμένη μελέτη περίπτωσης. Επιβεβαιώνεται από τη βιβλιογραφία ότι η χρησιμότητα των διάφορων μετεωρολογικών παραμέτρων εξαρτάται από τη συγκεκριμένη περίπτωση, ανάλογα με διάφορες συνθήκες. Τα δεδομένα που συνείσφεραν στην πρόβλεψη ήταν οι τιμές της ηλιακής ακτινοβολίας των προηγούμενων ωρών, η ώρα και ο μήνας.

- Για τα συγκεκριμένα δεδομένα της μελέτης περίπτωσης, το Δέντρο Αποφάσεων παρατηρήθηκε πως ήταν το πιο ακριβές μοντέλο όταν έγιναν οι προβλέψεις με λιγότερα δεδομένα (50% των δεδομένων, περίπου ενάμισι έτος), ενώ ο συνδυασμός Μηχανών Διανυσμάτων Υποστήριξης και Δέντρου Αποφάσεων ήταν το πιο ακριβές μοντέλο όταν υπήρχαν περισσότερα δεδομένα (3 έτη περίπου). Στη δοκιμή με το μικρότερο σύνολο δεδομένων εισόδου, η διακύμανση της ακρίβειας μεταξύ των διαφορετικών μοντέλων γίνεται μικρότερη. Η Γραμμική Παλινδρόμηση έχει σχεδόν την ίδια ακρίβεια και για τα δύο μεγέθη εισόδου, πράγμα που είναι αναμενόμενο καθώς δεν μαθαίνει από μεγαλύτερες ποσότητες δεδομένων από ένα σημείο και πέρα.

- Τα βέλτιστα αποτελέσματα για κάθε μοντέλο ήταν τα ακόλουθα:

	Δέντρο Αποφάσεων	**SVM**	**Γραμμική Παλινδρόμηση**	**Δέντρο & Παλινδρόμηση**	**Δέντρο & SVM**
	Μεγάλο Dataset				
nRMSE	17,11%	24,34%	51,15%	23,06%	15,56%
	Μικρότερο Dataset				
nRMSE	24,78%	25,71%	34,27%	25,71%	25,62%

- Η μέση τιμή της παραγωγής ενέργειας στα δεδομένα ήταν 58973 Wh ενώ το βελτιστοποιημένο δέντρο αποφάσεων είχε Μέσο Απόλυτο Σφάλμα 6901 Wh όταν το 50% των δεδομένων χρησιμοποιήθηκαν σαν είσοδος, που σημαίνει ότι παρουσιάζει ένα μέσο σφάλμα της τάξης του 11,7 %. Όταν έγιναν προβλέψεις μόνο για την τελευταία εβδομάδα, αξιοποιώντας σχεδόν το σύνολο των διαθέσιμων δεδομένων, η μέση τιμή παραγωγής των πραγματικών δεδομένων εξόδου ήταν 38953 Wh και ο συνδυασμός Μηχανών Διανυσμάτων Υποστήριξης και Δέντρου Αποφάσεων είχε Μέσο Απόλυτο Σφάλμα 3144, δίνοντας ένα μέσο σφάλμα 8,3%.
- Τα αποτελέσματα της προτεινόμενης μεθοδολογίας ήταν ικανοποιητικά σε σύγκριση με αυτά της βιβλιογραφίας, που σε ορισμένες περιπτώσεις φτάνουν και τις τιμές της τάξης του 41,20% RMSE.

Κεφάλαιο 6

Όσον αφορά τη συγκεκριμένη μελέτη περίπτωσης, μπορούν να γίνουν περισσότερες εκτεταμένες δοκιμές, αξιοποιώντας ένα ακόμη μικρότερο μέγεθος δεδομένων εισόδου, ώστε να δοκιμαστεί η αξιοπιστία των μοντέλων όταν υπάρχουν ακόμα λιγότερες πληροφορίες. Μπορούν επίσης να γίνουν δοκιμές στα συνδυαστικά μοντέλα με άλλες μεταβλητές εισόδου από τις βέλτιστες των Μηχανών Διανυσμάτων Υποστήριξης ώστε να συγκριθεί η απόδοσή τους.

Θα μπορούσαν να είναι χρήσιμα επίσης και διαφορετικά μοντέλα σε περαιτέρω δοκιμές, όπως οι αλγόριθμοι Deep Learning. Τα δεδομένα θα μπορούσαν να παράξουν ακριβείς προβλέψεις και με άλλα μοντέλα από αυτά που χρησιμοποιήθηκαν.

Θα μπορούσε επίσης να αναπτυχθεί λογισμικό είτε σαν διαδικτυακή εφαρμογή είτε σαν συμβατική εφαρμογή offline, ώστε να δημιουργηθεί ένα διαδραστικό περιβάλλον που να προσφέρει πρόσβαση στις λειτουργίες του προγράμματος σε κάποιον που δεν ξέρει να χειρίζεται την Python.

Chapter 1: Introduction

1.1 Introduction

The need for energy sources that are more friendly to the environment has made energy production from photovoltaics critical for the Greek economy.

Until recently, with energy coming just from thermal power plants, the amount of available electric power on the grid had a fixed value that was defined by humans. This has changed, as the energy is now produced by PV and wind turbines. Solar radiation and wind speed are stochastic variables and are to a large degree unpredictable through the duration of the day.

Because of that, it cannot be initially predicted how much power will the electric grid have available, and whether that power is enough in a given moment to satisfy the demand. As such, if no support system exists, power generation using Renewable Energy Sources leaves a country exposed to constant blackouts and voltage drops. Production forecast can ensure that the electric grid will be more reliable under a variety of conditions. Consequently, there is a need for predictions on the energy that will be produced in the near future. That way appropriate measures can be taken to adjust the energy demand so it can match the energy supply. One such measure is adjusting the energy price during the day. Energy forecasts also give the grid operator the ability to utilize different energy sources to respond to increases or decreases in the available power (1).

Weather forecasting is necessary to determine the fluctuation of energy production, but not enough by itself. Research has shown that weather data do not have a linear relationship with the energy production. Weather forecast data require further processing in order to make an assessment of the future energy production.

Also, power generation from photovoltaics can be decreased by faults that can occur in PV strings, including the collectors themselves. Such failures need to be determined and corrected in time to ensure the smooth operation of the grid. Timely detection of faults is important in order to minimize maintenance costs and maximize productivity.

Both the issue of production variance due to the weather as well as the issue of fault detection can be addressed with data analysis and machine learning. Using computational methods, weather data can be utilized in such a way that they can be linked to energy production. Additionally, data collected from various sensors can be compared with their values when a particular failure had occurred or was about to occur, in order to detect and fix future failures in time. Forecasting can also contribute to fault detection.

There is a need to determine the optimal methods for both production forecasting and fault detecting in PV in order to maximize their efficiency.

1.2 Thesis Target and Objective

There is a variety of methods that are used for PV production forecasting and fault detection. Their efficiency and accuracy vary greatly in each case study.

The subject of this thesis is to investigate the application of the various methods in production forecasting and provide a frame of reference on recent research on their application performance.

A similar investigation is made on PV fault detection methods, in order to provide a general frame of reference for which techniques are efficient in detecting different types of faults. The findings of recent research on the subject are also reported.

Additionally, this thesis presents a proposed methodology for PV production forecasting, which was implemented in Python. Different models were developed, optimized and then compared, using anonymized data as a case study. The performance of the proposed methodology was then compared with the models used in literature.

1.3 Thesis Contribution and Value

This thesis provides an original methodology designed to compare different forecasting models on their efficiency and subsequently produce forecasts using a chosen model. It provides a framework that can optimize each model in order to maximize its performance. It also has the ability to test each model using different data sets.

First, the data was pre-processed appropriately. Software for correcting erroneous values and missing values was developed. Then, the data was used in five different models, using Decision Tree, Support Vector Machines, Linear Regression, and two hybrid models of the aforementioned. Extended tests were made to determine the optimal set of input features. Afterwards, more testing was performed to optimize the hyperparameters of each model. The performance of the models in each test was recorded and compared to the previous results using specialized indicators such as Root Mean Squared Error, Mean Absolute Error and Coefficient of determination. Two different dataset sizes were tested as input. This way the efficiency of the models was compared for different amounts of information available.

Additionally, this thesis provides a general overview of the methods used in PV production forecasting as well as fault detection. The most commonly used methods are described in their basic concept, technical terminology and way of operation, their advantages and shortcomings, as well as their suitability for different applications.

An extensive review on literature is provided on the methodology used for PV production forecasting. For each research paper, the method is reported, along with the input features, forecast time range, and other specifics of the application, together with the performance and results in each case. This provides a frame of reference for future research, as the performance of many different applications can be quickly compared.

A similar review was performed on the methods used for fault detection in PV. Basic concepts, way of operation and suitability for detecting different faults are reported for each method, as well as the results of recent research on it.

1.4 Thesis Structure

This thesis is comprised of 6 chapters and 2 appendices.

Chapter 1 provides an introduction and the reasons that research is needed on the subject. The thesis target and objective are explained, together with its contribution and value. Finally, the thesis structure is explained.

Chapter 2 provides an analysis of forecasting methods described in literature as well as a review of recent research papers in PV production forecasting, together with the methods used and performance of each application. Persistence method, statistical methods and machine learning methods are discussed.

Chapter 3 describes the methods for detecting faults in Solar Cells. The way each method works and the type of faults that can be detected by it are explained. Applications in recent research are reviewed. Visual, electrical characteristics, imaging, artificial intelligence, device-based, hybrid methods, as well as methods using real-time sensors, are described.

Chapter 4 presents the proposed methodology. Forecasting is performed using collected historical energy production and weather data, using Decision Trees, SVM, Linear Regression, and two hybrid models using a combination of the above. The way the selected models work is explained. The related theory is analyzed with the appropriate examples, diagrams and formulas. The method of pre-processing the data is explained step-by-step. The correlation between the available data and the measured power output is examined in order to assess which variables will be useful as input features during the testing.

Chapter 5 reports in detail the tests performed and results obtained. This includes a series of tests to determine the best set of input features and tests to optimize the hyperparameters of each model. The optimization was performed both manually, reporting the accuracy of each model and selecting the best set by hand, as well as automatically, with the help of the GridSearchCV function of the scikit-learn tools. Tests using two different input set sizes were made. The results are discussed in detail.

Chapter 6 presents the conclusions made by the results and proposes future work to continue the research of this thesis.

Appendix A presents the source code that was developed to apply the proposed methodology. It includes statistical analysis and machine learning utilizing sci-kit learn. It is accompanied by comments explaining the functions.

Appendix B presents the source code that was used for pre-processing the data, which included making the necessary formatting adjustments, filling in missing values and correcting erroneous values caused by noise in the electrical grid.

Chapter 2: Review of PV Production Forecasting methods

2.1. Introduction

In this chapter the forecasting methods described in literature are discussed. Each section describes a different category of methods.

Section 2.2 briefly describes the Persistence method and its applicability.

Section 2.3 analyses methods utilizing statistical analysis, including ARMA, Regression, exponential smoothing and the physical method. Their way of working is described as well as the conditions in which they perform well.

Section 2.4 describes Machine-Learning methods, such as ANN, SVM and hybrid methods.

Section 2.5 contains a literature review over a large number of papers utilizing different methods for PV power forecasting, displaying the accuracy in each case.

Section 2.6 includes the conclusions made from the gathered information.

Forecasting methods for PV power generation are divided to direct and indirect models. In Indirect models, solar irradiance is forecasted by using techniques such as numerical weather prediction (NWP), image-based methods and hybrid artificial neural networks (ANN) (3). The forecasted solar irradiance is inputted to software such as TRNSYSM, PVFORM and HOMER (5) (3) to get the power generation forecast (3). In direct models, instead of forecasting solar irradiance, historical data samples of the power output and the weather data are used to directly forecast PV power generation. A study by Mitsuru e al. (6) implemented both indirect and direct methods for PV power to forecast production in a 1-day scope and concluded that the direct method is better (3).

The accuracy of the forecasting depends greatly on the forecasting horizon. Short-term horizons used are the following (7):

	Intra-hour	**Intra-day**	**Day Ahead**
Time range	15-120 minutes	1 to 6 hours	1 to 3 days
Time step	30 sec to 5 min	1 hour	1 hour
Used for	Anticipating large short-term fluctuations	Anticipating grid load	Schedule Drafting
Model used	Timeseries/Total sky imager	NWP/Satellite image	NWP/Satellite image

PV production depends mainly on solar irradiance. Other weather parameters, like temperature, humidity and wind speed, are potential factors influencing production and subsequently potential input features to be used in a forecast, but their impact and correlation

with PV production depends on geographical location (3). For that reason, each of them needs to be examined in every case study to determine whether it should be used or not in the forecast. For example, wind speed does not affect PV power output when the temperature is very low. Solar radiation has high correlation with PV production in any weather conditions, even though the correlation is smaller in cloudy or rainy conditions (8).

It's important to preprocess the input data, because historical data can have outlier values, non-stationary components due to the weather, or missing values due to recording errors, all of which can result in a lower forecast accuracy. Preprocessing methods include historical lag identification, normalization (9), trend-free time series, wavelet transform (WT) and self-organizing map (SOM) (3). Normalization is the most commonly used preprocessing method in PV production forecasting (3), and it consists of converting each of the individual data values into the ratio of their value divided by the difference between the maximum and minimum value of the data (deviation). In case of pre-processing, post-processing is also required before the accuracy of the forecasting model can be estimated. Most common methods include anti-normalization, if normalization was used in the forecasting model, (9) and wavelet construction, if wavelet decomposition was used.

PV production forecasting methods are categorized based by the way the forecast horizon and the data are used. Based on the forecast horizon, forecasting is divided to short-term (1 hour up to 7 days), medium-term (one week to one month) and long-term (one month to one year). The forecast error increases together with the forecast horizon. Short-term forecasting is used for energy management and scheduling of electrical power. Medium-term forecasting is used for planning the power system and maintenance schedule. Long-term forecasting is used for planning the electricity generation. Almost all of the literature analyzed in this study concerns short-term forecasting. Categorization is also based on the way historical data of the PV output and weather data are used. This includes the persistence, statistical, machine-learning and hybrid method models.

2.2 Persistence model

The persistence model doesn't use weather data and equates the forecasted power output to the value of the previous day at the same time of that day. This model is used for benchmarking in order to compare other methods' efficiency. It is used generally to forecast the power output for the next hour, and the model's accuracy depends on how stable the weather conditions are. If the weather conditions are the same as the previous day, the power output of the previous day at the same time is a good indicator of the currently expected output. When making forecasts on a longer time range, the accuracy decreases (10).

2.3 Statistical methods

These methods use statistical analysis on the data, using previous time-series data. They are normally used for short-term forecasting. Using more recent data as input increases the accuracy of the prediction.

Such a method is the Autoregressive moving average (ARMA) model, which uses a combination of the autoregression and moving average models to forecast PV generation from a defined time-series. The formula for the ARMA model is the following (2):

$$X(t) = \sum_{i=1}^{p} \alpha_i X(t-i) + \sum_{j=1}^{q} \beta_j e(t-j)$$

In the above formula X(t) is the forecasted PV output, adding together the AR and MA functions. P and q indicate the order, α_i is the AR coefficient, β_j the MA coefficient, e(t) produces random variables with zero mean and constant variance (white noise) (11). The reasons for this model's popularity are that: 1) it adopts the Box-Jenkins method and 2) it has the ability to extract statistical properties (12). An extension of this model, the AR integrated MA (ARIMA), removes any non-stationarity from the input (13). Its disadvantage is that the time series data have to be stationary (14).

Another statistical method is the regression method, which is used to establish a relationship between the input and output data. In this case the weather data is considered to be a set of explanatory variables and the forecasted PV output is set as the dependent variable. The PV power forecast can be calculated with either a simple linear regression or with multiple linear regressions (15). Using both temperature and solar irradiance as input gives a better regression model than using either on its own, although the efficiency in using temperature as an input depends on the geographical location. In order to use this method, a mathematical model and a number of explanatory variables are needed.

The exponential smoothing method makes forecasts by smoothing time series data using the exponential window function, in which weights are assigned on past data and diminish exponentially the further they go into the past. The simple exponential smoothing method (EWMA) has the following formula (3):

$$\hat{Y}_{t+1} = \alpha Y_t + (1 - \alpha)\hat{Y}_t = \hat{Y}_t + \alpha(Y_t - \hat{Y}_t)$$

In the above formula, α is the smoothing constant, ranging from 0 to 1. An initial Y_0 value needs to be set by estimation to begin the iterations. EWMA is similar to the Moving Average method and is best used with time series that are trendless (stationary), as the forecast would show earlier values instead of the actual trend otherwise.

The physical method uses mathematical formulas and Numerical Weather Prediction models (NWP) that relate weather conditions to the PV production, using parameters such as the geographical location, the orientation of the panels and weather variables (3). This model can be either simple or complicated, depending on the number of weather variables it uses. It is more accurate in good weather.

2.4 Machine-learning methods

Machine-learning methods require a larger dataset than statistical methods in order to be accurate, but can make predictions using linear, non-linear and non-stationary input data.

2.4.1 Artificial Neural Network

The Artificial Neural Network (ANN) is a machine learning method. It is the most effective method for PV production forecasting, and it is used in most research papers when the weather data is non-linear, as statistical methods are less accurate when there is a complicated non-linear bonding between the data (3). An ANN consists of input layers, hidden layers, output layers, connections and neurons. The input layer takes in the input information, which is analyzed by the hidden layer. The output layer provides the output, having received the analyzed results from the hidden layer. The connections link together the neurons between different layers, as shown in Figure 3. A neuron cell consists of the combination function, which adds the inputs together, and the activation function, which transfers the input in the form of an output. The general ANN formula is the following (16):

$$U_N = b + \sum_{j=1}^{N} (W_j \times I_j)$$

In the above formula U_N is the final network output, b the bias weight, N the number of inputs, W_j the connection weight and I_j and the network input.

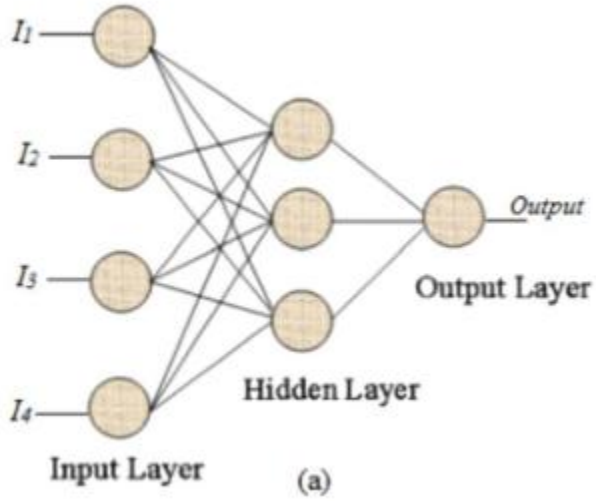

Figure 3 - Basic ANN architecture (3)

The two basic functions of a Neural Network are Training and Testing. During Training, the learning algorithm tries to discover the relationship between the input and the output by changing the weight values of the synapses. After comparing the actual output with the predicted output, the error is calculated and the weight and bias values of the NN are updated based on that error. This is repeated until the predicted and actual output match. The final

output is calculated from the weight values and input data that was used for the testing, and varies when the input, the activation function or the architecture change.

The most common activation functions for PV forecasting are the Gaussian radial basis, the sigmoid and the hyperbolic tangent sigmoid functions (3). They are differentiable, continuous and non-linear (17). While most problems can be solved by single layer Neural Networks, when more complicated relationships are present between the input and output variables, other types of NN are needed, such as the Multilayer Perceptron, the Radial Basis function NN, the Multi-Layer Feed-Forward NN, the Recurrent Neural Network (RNN), the Adaptive Neuro-Fuzzy Interface Systems (ANFIS) and the General Regression Neural Network.

The Multilayer Perceptron Neural Network (MLPNN) is composed of one or more hidden layers, the number of which can be adjusted depending on how complex a problem is, although more than two are rarely needed (18). It is a supervised feed forward ANN (19) and it is used for PV power forecasting, among other applications.

The Radial Basis Function Neural Network has two layers and the training process is divided into two stages. It needs less computing time than other Neural Networks and has good accuracy. It is simple in structure and it is also used in PV power forecasting (20).

A Multi-Layer Feed-Forward Neural Network is relatively less complex than MLPNN, as the information is transferred from the input to the direction of the output only, without a feedback loop. It is used in several forecasting applications (21).

The Recurrent Neural Network is good for time-series data forecasting, as it is suitable for predicting complex relationships between input and output variables (20). It is more accurate than feed-forward neural networks (22).

Adaptive Neuro-Fuzzy Interface Systems are a type of adaptive Multi-Layer Feed-Forward Neural Network. They are the most commonly used fuzzy system as they are transparent and require less computational power. Their drawback is that they require a large amount of data and their predictions have a tendency of overfitting.

The General Regression Neural Network is also effective in solving non-linear problems but requires a lot computational power as it grows in size. As the name suggests, it is designed for regression tasks.

The Back-propagation Neural Network is also commonly used because it can solve complex regression problems. It is a very accurate supervised learning algorithm (23). A modified version exists to fix certain limitations of its original version.

2.4.2 Support Vector Machine (SVM)

SVM is a supervised machine learning method that was originally designed for classification, but was expanded to be used in regression problems as well. It is based on the structural risk minimization principle and can minimize the error of the training data by minimizing an upper boundary of the expected risk.

Support Vector Regression (SVR) is the application of SVM in a time series regression, which is used in PV power forecasting (24). It maps the time series data into a higher dimensional feature space using non-linear mapping and then performs linear regression on that space (3), converting the non-linear regression to linear regression, as shown in Figure 4.

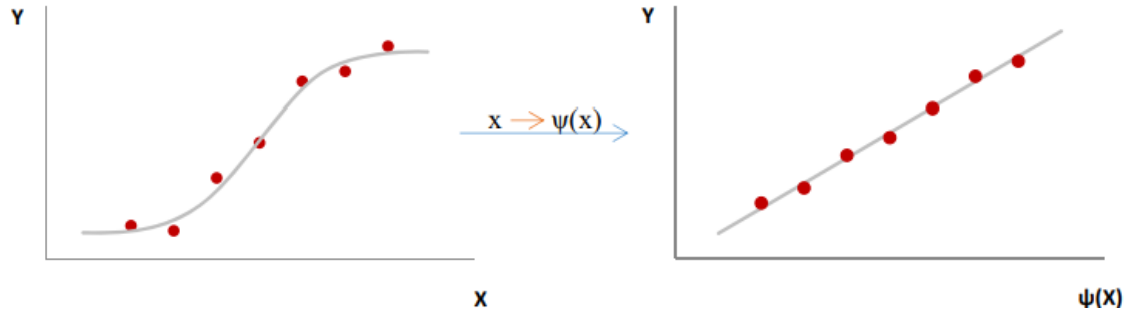

Figure 4 - Converting non-linear regression to linear regression (3)

The estimation function for SVR is as follows (25):

$$y_f = f(x) = w \times \psi(x) + b$$

In the above function x is the input weather data, y is the PV power output, w is the weight vector and b is the bias term. These values are approximated through minimization of the regularized risk function (25):

$$R(C) = C\frac{1}{N}\sum_{i=1}^{N} L_\varepsilon(y_i, f_i) + \frac{1}{2}\|W\|^2$$

In the above, $L_\varepsilon(y_i, f_i) = \begin{cases} |y_i - f_i| - \varepsilon, if\ |y_i - f_i| \geq \varepsilon \\ 0, otherwise \end{cases}$ is the ε-insensitive loss function, C and ε are user-defined parameters, y_i is the real value at period I and f_i is its respective forecasted value. If the forecasted value is within the ε-tube the loss will be zero. The term $\frac{1}{2}\|W\|^2$ measures how flat the function is.

The accuracy of SVR depends greatly on the choice of the kernel function and the corresponding parameters (9). The most common functions include: a) the linear function, which can only be used when the training data has only small fluctuations, b) the polynomial, which represents how similar the training samples are in a feature space, c) the Gaussian RBF, which can be used to an infinite dimensional feature space and has been extensively used for PV power forecasting, and d) the sigmoid function. The kernel parameters control the complexity of the model and set how the high dimensional feature space is structured. For PV forecasting, the parameters that need to be defined in an SVR model include the tube radius ε, which determines the range of data which are not to be taken into account for the regression, the penalty C, which defines the penalties for the estimation errors, and the kernel function's parameter (3). These parameters determine how accurate the model will be (3).

2.4.3 Hybrid Models

Hybrid models use of a combination of the methods like the ones described above. Hybrid models are more accurate as they combine the advantages of different models and overcome the limitations of their stand-alone versions. Hybrid models are also more accurate specifically in PV power forecasting. Such a model is the Fuzzy inference model with RNN (26). The fuzzy inference model is used to smoothen the input weather data. The hybrid fuzzy-GA forecasting model is also very accurate in PV power forecasting (27). Wavelet transform (WT) is also used in many hybrid models in order to de-noise the input data, followed by the application of ANN and SVM models, resulting in a minimized prediction error (28).

Utilizing more than one model makes the technique more complicated for the computer to process, increasing computational power requirements. The accuracy of the hybrid model depends on the performance of the consisting models. Should a consisting single model have inadequate performance, the total performance of the hybrid model suffers (3).

2.5 Research done recently on PV power forecasting

Authors and References	Forecast horizon	Forecasting model	Forecast error	Year	Description of contribution
Mayer and Grof (29)	Up to 48h	Physical model	nRMSE 16,8%	2021	Physical PV forecasting from NWP data.
Theocharides et al. (30)	1 d ahead	SVR, BN, RT	nRMSE 4,53%, MAPE 3,17%	2021	Input features for the forecast included the historical PV power production, irradiance and ambient temperature, NWP, solar elevation and azimuth angles.
Nespoli et al. (31)	24h ahead	ANN, Hybrid ANN	NMAE 1-2% for ANN, 2-5,3% for HANN in Sunny days	2019	This study compared an ANN and a Hybrid ANN method, the first using only historical climatic and PV system parameters, the second also using the daily weather forecast. The hybrid model has better accuracy on some days but the results vary more.
Soumyabrata et al (32)	20 min ahead	Triple exponential smoothing	0,13 MAE	2018	Input features for the forecast included time-series data of the measured solar irradiance and the clear-sky solar irradiance.
Theocharides (33)	1 d ahead	ANN, SVR, RT	nRMSE 0,76% ANN, 1,13% SVR and 1,33% RT	2018	In this study, ANN outperformed SVR and RT. The input variables were global irradiance, ambient temperature, relative humidity, Azimuth, Elevation and wind direction.
Alfadda et al. (34)	1h ahead	SVR, Polynomial Regression and Lasso	RMSE 5,3%	2017	Input features included weather conditions (sky condition, weather temperature, module temperature, solar irradiance, wind velocity, wind direction, dew point, relative humidity, visibility and cloud cover), power generated in the last few hours, as well as day and time information. Not all features improved accuracy. This

Authors and References	Forecast horizon	Forecasting model	Forecast error	Year	Description of contribution
					study found SVR to be more accurate than PR and Lasso.
Cervone et al. (35)	72 h ahead	ANN and Analog Ensemble	RMSE 8,66%	2017	This study presented a method combining ANN and AnEn which performed better than using either method individually. Statistical analysis was made based on observations of the PV solar farm output and atmospheric NWP model data. The analysis was done based on simulated solar farms.
Grimmacia et al. (36)	1 d ahead	PHANN	NMAE 12,6 %	2017	This study concluded that the settings minimizing the NMAE were an ensemble composed of 10 trials and 120 neurons in a single layer ANN configuration. The input dataset included one year of PV power output measurements and corresponding weather data including ambient temperature, global horizontal solar radiation wind, etc. as well as the deterministic global solar radiation under clear sky conditions.
Kumar Das et al (37)	1 d ahead	SVR, ANN	3,08 % nRMSE	2017	Input features used were the historical PV power output and corresponding meteorological data. Weather was categorized as either normal (clear sky) or abnormal (cloudy or rainy). The study concluded that the SVR model was more accurate than the corresponding ANN one.
Theocharides et al. (38)	1 d ahead	ANN	0,71% nRMSE	2017	Input Variables used were the Numerical Weather Prediction (NWP), Satellite images and Sky images. A conventional Feedforward NN was designed, consisting of 7 inputs and 12 hidden nodes. An ensemble meta-

Authors and References	Forecast horizon	Forecasting model	Forecast error	Year	Description of contribution
					algorithm was developed based on the optimally identified network and was used to increase accuracy.
Fentis et al. (39)	Short-term	SVR	RRMSE 15,23 %, R-squared 0,96%	2016	Input features included samples of solar radiation every 15 minutes and the plate temperature. Three months' worth of samples were used.
Vagropoulos et at. (40)	1 d ahead and hourly	SARIMAX, SARIMA, modified SARIMA and ANN	nRMSE 12,89% for SARIMA, 11,12% for modified SARIMA, 10,25% for optimized combined model	2016	The SARIMAX, SARIMA, modified SARIMA and ANN models were compared, using solar radiation data (real measurements and forecasts) obtained from a weather station.
Wolff et al. (41)	15 min and 2 h	SVR, Physical model	RMSE 10,5%	2016	Input features used were the PV power measurements, numerical weather prediction and cloud motion data.
Zhaoxuan Li et al. (42)	15 min, 1 h and 24 h ahead	ANN and SVR	15 min: ANN 13,2% RMSE, SVR 13,3 %, 24h: 41,2%	2016	Input data used were the ambient temperature, wind speed, wind direction, solar zenith angle (without atmospheric correction), solar zenith angle, degrees from zenith, refracted, cosine of solar incidence angle on panel, cosine refraction corrected solar zenith angle, solar elevation (no atmospheric correction) and the solar elevation angle (degrees from horizon, refracted).

Authors and References	Forecast horizon	Forecasting model	Forecast error	Year	Description of contribution
Almeida et al. (43)	1 d ahead	Non–parametric model	cv-Mean bias error (cv MBE) < 1.3%	2015	A non-parametric forecasting model was developed using NWP data, treating the PV system as a black box.
Chu et al. (44)	5, 10, and 15 min ahead	GA optimized ANN	Minimum mean absolute error (MAE) 21.02%	2015	This study proposed an ANN-based real-time smart reforecasting model, using GA optimization for better accuracy, to make forecasts within 1 hour and less. GA defined the optimized input features, the number of hidden layers and neurons per layer. The results showed that the proposed model had better results than the cloud tracking-based deterministic model, the ARMA model, and the k-nearest neighbor (kNN) model.
De Leone (45)	1 d ahead	v-SVR	11,43% MAPE	2015	Input features included the historical data on solar irradiance, the environmental temperature and past energy production.
Dolara et al. (46)	24 h ahead	Physical model	Normalized MAE (NMAE) < 1%, and weighted MAE (WMAE) < 2%	2015	This study proposed three physical models for forecasting using recorded weather data. The accuracy of the model depends on the calculation of the cell temperature and the data used for its calibration. Unlike the ANN-based forecasting models, it did not need a training period.
Dolara et al. (47)	1 d ahead	Physical Hybrid ANN	nRMSE 4,98% on a sunny day, 17,9 on an unstable day, 33,3 on a cloudy day	2015	A Physical model and a Clear Sky Solar Radiation Model were combined with ANN. The theoretical Solar Radiation was calculated and used as input along with weather forecast data (Temperature, Humidity, etc).

Authors and References	Forecast horizon	Forecasting model	Forecast error	Year	Description of contribution
Leva et al. (48)	24 h ahead	ANN	Normalized RMSE 12,5% -36,9%	2015	The PV forecast was made with an ANN-based model. The model's performance was analyzed during several days of varying cloud coverage. The study showed that the model's accuracy depends strictly on the pre-processing of the input data and how accurate the input dataset is.
Liu et al. (49)	24 h ahead	Back-propagation (BP) based ANN model	Mean absolute percentage error (MAPE) 7,65%	2015	An innovative PV forecast model was developed that added aerosol index (AI) as an input factor. AI was used together with seasonal weather classification and a BP-based ANN model to predict the PV power generation of the next 24h. The conclusion was that the new model outperformed the conventional ANN model.
Ramsami and Oree (50)	24 h ahead	Stepwise regression, GRNN, FFNN, and MLR	RMSE 2,74%	2015	This study used a model based on Stepwise regression to determine which input features had the strongest correlation with the PV output. Then, inputs were used in models using GRNN, FFNN, MLR, and their respective hybrid models. The stepwise regression-FFNN hybrid model outperformed the other models. All the hybrid models slightly outperformed their respective single-stage models.
Rana et al. (51)	30 min	NN, X-Means algorithm, and an iterative methodology	Mean relative error (MRE) 16,92%-17,58%	2015	Three models were developed based on ensembles of NN to predict the 30-min PV power generation for the next day. One model was iterative and two were non-iterative. They were compared using four yearly solar datasets. The iterative model was the most accurate. The ensemble NN

Authors and References	Forecast horizon	Forecasting model	Forecast error	Year	Description of contribution
					models were also found to be more accurate than a single NN.
Yang et al. (52)	1–2 h ahead	Spatial and temporal correlations	RMSE- 81.5 for 1 h ahead and 136.3 for 2 h ahead	2015	An innovative multiple time-scale, data-driven forecasting model was developed, using ARX-based ST. Spatial and temporal correlations were made between neighboring solar sites. The new model outperformed the conventional persistence model.
Yang et al. (53)	Monthly	Exponential Smoothing method with decompositions	nRMSE 11,99% to 22,35%	2015	Input variables used included the time series of the Global horizontal irradiance and the cloud cover index. Each was tested separately.
Zhu et al. (54)	1 d ahead	Hybrid model (wavelet decomposition (WD) and ANN)	RMSE 7.193%−19.663%	2015	A hybrid model using wavelet decomposition and ANN was proposed to a make forecasts using less computing power. Wavelet decomposition was used on recorded PV output data which was then used on ANN.
A. Gandelli et al. (55)	24 h ahead	Physical Hybrid ANN	nRMSE 10,51%	2014	This study proposed a Physical Hybrid Artificial Neural Network, based on ANN and basic Physical constraints of the PV plant. Input variables used were the Day, Hour, Environmental Temperature, Wind Speed, Humidity, Pressure and Cloud cover. The proposed method was more accurate than a simple ANN.

Authors and References	Forecast horizon	Forecasting model	Forecast error	Year	Description of contribution
De Giorgi et al. (56)	1–24 h ahead	Statistical methods based on multiple regression (MR) analysis and Elman ANN	NMAE 6,50%−19,49%, nRMSE 10,91%−23,99%	2014	In the study, multiple regression was used examining different input features. Elman ANN was used for forecasting. The model is most accurate when all weather data and the power output are used as input features.
Junior et al. (57)	24 h ahead	SVR and principal component analysis (PCA)	RMSE 10,24%	2014	Three models were tested using past PV output data and SVR to make daily forecasts. Preprocessing with PCA was found to improve performance significantly.
Tao and Chen (58)	24 h ahead	GA-based NN	Error 8.00%	2014	In this study the weights and thresholds of back-propagation NN (BPNN) were optimized using the GA approach to make the model more accurate. The results of the model were better than the BPNN-based forecasting model.
Yang et al. (59)	1 d ahead	SOM, learning vector quantization (LVQ), SVR, and fuzzy inference	MRE 3,295%	2014	A hybrid forecasting model based on meteorological data was proposed in this study. Historical data was categorized using SOM and LVQ based on the weather. The model was trained using SVR. Sub-models were chosen using fuzzy inference. The results of the new model were more accurate than those of the simple SVR and ANN.

Authors and References	Forecast horizon	Forecasting model	Forecast error	Year	Description of contribution
Yantig et al. (60)	1 d ahead	ARMAX	MAPE 38,88% for training data and 82,69 for validation data	2014	The proposed model is based on ARIMA and uses temperature, precipitation, insolation duration and humidity. It does not use a forecast for solar irradiance.
Dong (61)	5 to 60 min	Exponential Smoothing method	nRMSE 26,5% for 5 min 49,5% for 60 min	2013	This study proposed a Fourier trend model along with a KPSS test.
Haque et al. (62)	1 d ahead	WT, fuzzy ARTMAP (FA), and firefly (FF)	MAPE 3.38%−11.83%, nRMSE 12.11%−13.13%	2013	A hybrid model for PV power forecasting was proposed, making a daily forecast based on meteorological data in which a seasonal classification was performed. WT was used to filter out the outlier values in the data. FA was used for forecasting, and firefly was used for optimization.
Tuyishimire et al. (63)	1 d ahead	Kalman predictor	N/A	2013	A model was proposed based on real-time forecasting from a multiple-rate Kalman predictor. This model applied statistical analysis on historical data. One model provided steady-state variance and a second provided transient-following capability.
Yona et al. (64)	24 h ahead	Fuzzy theory (FT) and NN	Maximum MAE 2,76 kW	2013	Models using FT and RNN were proposed for forecasting in this study. The forecasted solar irradiance and weather data were used. FT used irradiance forecast data to smoothen the RNN training stage.
Mandal et al. (65)	1 h ahead	WT and RBFNN	MAPE 2,38% (sunny day) and	2012	A forecasting model was proposed combining RBFNN and WT. The correlation between solar irradiance,

			4,08% (cloudy day)		temperature and PV output was calculated. WT was used to remove the outliers in the time-series of the PV output and weather. RBFNN tracked the nonlinearity of PV power data. The proposed model was more accurate than the single RBFNN, although it was not as accurate for rainy days.
Mori and Takahashi (66)	24 h ahead	Hybrid Method (generalized RBF network, Deterministic annealing, and evolutionary particle swarm optimization (EPSO))	Maximum error 0,228 pu	2012	In this study, the center and width of the RBF was calculated in a GRBFN using DA. Because the data was non-linear, a weight decay technique was used to avoid overfitting. Evolutionary particle swarm optimization was used for selecting the optimal neuron weights in GRBFN. GRBFN was applied for forecasting. The model outperformed the MLP, RBFN, and GRBFN and other ANN models.
Pedro and Coimbra (67)	1 h and 2 h ahead	ANNs optimized by GA (GAs/ANN)	nRMSE 13,07% for 1 h and 18,71% for 2 h ahead	2012	Five models were compared using ARIMA, persistent, ANN, ANN optimized by GA and kNN. Input features outside the system, like solar radiation telemetry, were not used. ANN models had the best accuracy, particularly GAs/ANN. All the models' performance was heavily affected by the seasonality of the solar irradiance.
Shi et al. (68)	1 d ahead	Weather classification and SVM	MRE 8,64%	2012	A forecasting model was proposed using SVM and weather classification. Four SVM sub-models were created, each for a different classification of the weather conditions.

Authors and References	Forecast horizon	Forecasting model	Forecast error	Year	Description of contribution
Xu et al. (69)	1 h ahead	Similar day selection algorithm and weighted SVM (WSVM)	Mean square error (MSE) 21,8	2012	The proposed forecasting model used weighted SVM. Five days which were the most similar to the day ahead were used for the training, using the similarities to set the weights for the weighted SVM. The model was shown to outperform the ANN.
Chen et al. (70)	24 h ahead	SOM and RBFNN	MAPE 8,29%−10,80% (sunny day)	2011	This study proposed a model using RBFNN for forecasting. The accuracy of the model depended on cloud cover conditions, for which classification was made using SOM. This model had good accuracy in forecasting in sunny and cloudy days and acceptable accuracy in rainy days.
Chupong and Plangklang (71)	1 d ahead	Elman NN	MAPE 16,83%	2011	A PV forecasting model using Elman NN. Used as inputs of this model were the calculated solar irradiance in sunny weather and the predicted meteorological data. Solar radiation was calculated instead of measured due to lower cost.
Cococcioni et al. (72)	24 h ahead	Time series analysis and feed-forward NN (FFNN) with tapped delay lines	MAPE < 5,0%	2011	A model for daily forecast was proposed based on an ANN with tapped delay lines. This model used a NARX time-series analysis model and had good performance during experimentation.

Authors and References	Forecast horizon	Forecasting model	Forecast error	Year	Description of contribution
Ding et al. (73)	24 h ahead	ANN and similar day selection algorithm	MAPE 10,06% (sunny day) and 18,89% (rainy day)	2011	An ANN model based on an improved BP learning algorithm was proposed in this study to overcome the shortcomings of the standard BP learning algorithm, which has slow convergence and a tendency to fall into the local minimum. The technique does not need complex modeling or complex calculations. The Similar day selection algorithm was used to improve performance.
Kang et al. (74)	24 h ahead	K-means clustering method	MAPE 11%	2011	This study used at the first stage the K-means clustering method. Analysis and classification were performed on weather data concerning the chance of rain. The data was then used in a forecasting model to make predictions. In order to monitor cloud coverage, the use of a camera was recommended.

2.6 Conclusions

The following conclusions were made:

- According to the gathered literature, machine learning methods outperform statistical methods in general.
- Variations of Neural Networks are the most popular method, followed by SVR and hybrid models.
- The best performance was displayed by ANN in certain papers, although in others ANN methods performed poorly.
- Most forecasts in literature are generally short-term, usually ranging from less than an hour up to one day.
- The effectiveness of a model greatly varies with the weather. Sunny weather gives much better accuracy in several cases.
- The usage of different weather data greatly varies in literature. Some researchers rely solely on historical production data and others utilize parameters such as cloud coverage or temperature to make forecasts. It was concluded that, for the purposes of this research, different sets of input features needed to be tested to determine the best performance, because, in the literature, there was no conclusion of particular variables being helpful in every case.

Chapter 3: Review of PV Fault detection methods

3.1 Introduction

In this chapter the methods for detecting faults in Solar Cells are examined. Each section describes the way a method works and the type of faults that can be detected by it. Applications of each method in recent research have been investigated and explained.

Section 3.2 describes the visual method for detecting faults, which primarily includes detecting a failure by manual observation. Procedure standards, detectable fault types and shortcomings of the method are explained.

Section 3.3 describes the methods based on analyzing the electrical characteristics of PV installations. This includes MBDM, RDM, Electrical I-V measurement, OSA, PLA and CDI. Each method is explained along with related diagrams and the basic formulas each method uses. The results of related research are discussed.

Section 3.4 describes imaging techniques, including Thermal/IR, Ultrasonic, Electroluminescence and Lock in Thermography.

Section 3.5 analyses methods based on artificial intelligence, which analyze sensor data to detect failures. This includes Machine learning methods, such as SVM, and Deep Learning methods, such as Convolutional Neural Networks (CNN).

Section 3.6 describes Device-based methods, and the specialized devices that exist to diagnose different types of failures.

Section 3.7 describes methods using real-time sensors connected to computer software to detect a possible failure before it occurs.

Section 3.8 describes hybrid techniques, combining a number of methods described in the previous sections.

Section 3.9 lists the conclusions reached from this chapter.

3.2 Visual method

Visual inspections are the first step for fault detection in order to determine whether further tests are needed. They are usually conducted regularly (75). A checklist for the visual inspection of fielded modules has been developed by NREL/IEA and the US Department of Energy (76). IEC-61215 standards (International Electrotechnical Commission, 1987) state that the inspection should be conducted at 1000 lux and from different angles to avoid reflections, as they can give defective images (77). Faults that can be detected this way include discoloration, bubbles, burning marks, delamination, browning, cracked glass or cells, dirt point, loose wiring or wiring exposed to damage, rusted or corroded interconnections, snail tails and damaged pieces, as well as soiling by e.g. snow or fallen leaves (78). The shortcoming of visual inspection is that it depends on human capabilities, which can be unreliable. Also, faults may be detected too late (75). Visual inspections also include shade analysis, which can be performed by tools

like the Solmetric Suneye and can detect possible mismatch issues due to e.g.new trees growing or new buildings constructed (79).

3.3 Electrical Characteristics Methods

3.3.1 Model-Based Difference Measurement (MBDM)

PV modeling emulates virtually the real-time operating characteristics of a PV system (4). MBDM works by comparing real-time parameters with the results predicted in the model, in order to detect faults in the system (80). Real-time output parameters like the operating voltage and current change during faulty operation (81). Theoretical parameters are calculated using real-time measurements of irradiance and temperature, which are compared to the actual measured parameters, in order to detect faults. The most commonly used models are the single diode (SD) and double diode (DD) models (82) (83) (84). The accuracy of the model depends on the accuracy of the method used for the extraction of the model parameters (85) (86) (87). The procedure is displayed on Figure 5.

A number of techniques based on MBDM have been developed. A method for automatic failure detection in units connected to the grid was developed by Chine et al. (88), which was based on MBDM. The method compared the forecasted power ratio with the power ratio measured in real-time. The difference between the two was utilized to calculate the absolute power ratio error (APRE) to perform fault detection. Kymakis et al. (89) used the error in Rc and Rv in order to detect whether there is an inverter fault, a string fault or a general fault. The string current was divided by the AC power and the result was used to find the location of the fault. Also, Silvestre et al. (90) used the difference between the real-time system losses and the ones predicted by the model in order to detect faults, and compared the real with the predicted DC current and DC voltage to distinguish between shading faults and other system faults. Davarifar et al. (91) used only the difference between the actual and the forecasted power as a parameter for detecting open circuit faults, hotspots and ground faults, utilizing a Wald test to prevent false positives. The faults were diagnosed using a flash test device that can inspect I-V and P-V curves.

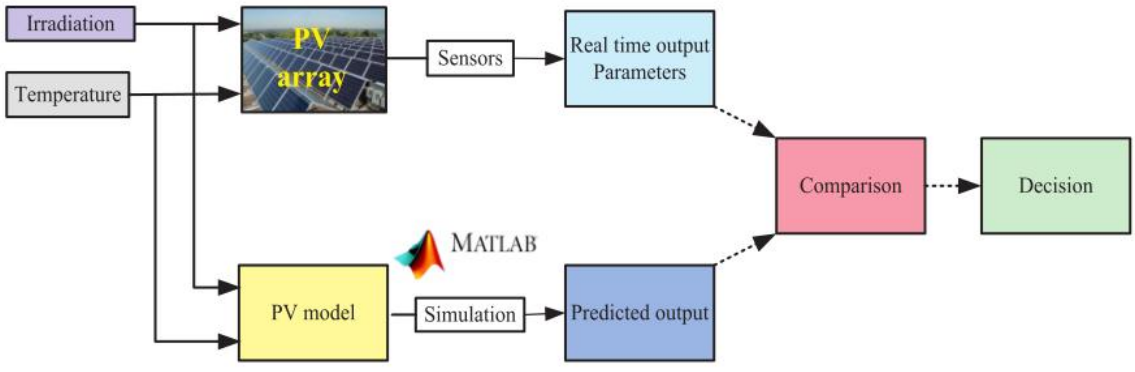

Figure 5 - MBDM-based fault detection (80)

3.3.2. Real-time difference measurement (RDM)

RDM detects faults by comparing real-time values with their threshold limits (4). The threshold limits are set by using either PV modeling or real-time experimentations. It is similar to MBDM, but doesn't need a real-time PV model to detect faults. RDM detects faults more quickly than MBDM, which is important in order to isolate and repair faulty panels in time (4). The accuracy of the threshold limits determine how accurate RDM is. The limits are set based on a PV model, so the accuracy of the model determines the accuracy of RDM. A diagram of the procedure is displayed on Figure 6.

Shimakage et al. (92) used three ways to detect faults using RDM: 1) By measuring the AC output power and comparing it with the threshold limit, 2) By calculating the difference in power between the present and previous instant and 3) By calculating the performance ratio and comparing it with the previous instant. The most critical is the third way as the performance ratio has a strong correlation with the environmental variations (93). Fault detection is needed if the output power of the PV system has more than 6% loss (92) (94). Xu et al (95) developed a method for detecting shading faults. In this method the module voltages as well as the current between particular groups of panels were measured using several sensors. The momentary difference between current and voltage values was used as the detection criteria. The severity of faults was measured using a three-level alarm system.

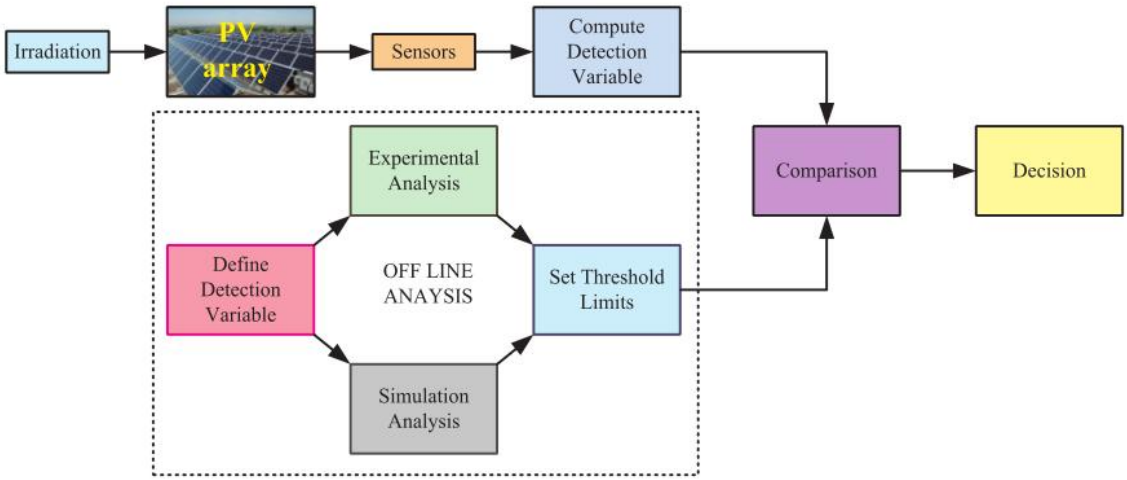

Figure 6 - RDM for fault detection (4)

3.3.3 Electrical current–voltage (I-V) Measurement

Electrical I-V measurements are usually done at the PV strings, as the PV array generally has several parallel PV strings connecting at the junction box which can be measured individually (75). The voltage and current of the string output is checked at the junction box to detect faults such as disconnection or degradation, as such faults reduce the string output power or affect the string I-V curve (96). The actual location of the fault cannot be determined using methods like I-V or Voc measurement (97), so a method for automatic fault detection was

developed, which compared the measured I-V characteristics of the string with its characteristics under various fault cases. Miwa et al. (98) determined the (-dI/DV)-V characteristic using the I-V characteristics. The peak appearance of the calculated characteristics was used to detect the partial shadow phenomenon (75). The values of series resistances, shunt resistances (Rs and Rsh) and fill factor (FF) were calculated from the I-V characteristics and then used to estimate the performance of the PV system (99). The fill factor is determined by the formula below:

$$FF = \frac{P_{max}}{I_{SC} \cdot V_{OC}}$$

3.3.4 Output Signal Analysis (OSA)

OSA methods analyze the output signal on the time domain to detect faults. They can detect various faults, including shade faults, short-circuit faults and ground faults (81), but are primarily used for detecting arc faults. Failures in the PV system can be detected by the terminal output characteristics, especially when distortions appear in the output current and voltage (4). This is possible regardless of the environmental conditions. Threshold limits are calculated and fault check is performed when they are exceeded (80). Oscillations and distortions in the voltage and current waveforms can indicate arcing faults. The anomalies can be detected in the waveform by using Fast Fourier Transforms (FFT). A diagram of the procedure is displayed on Figure 7.

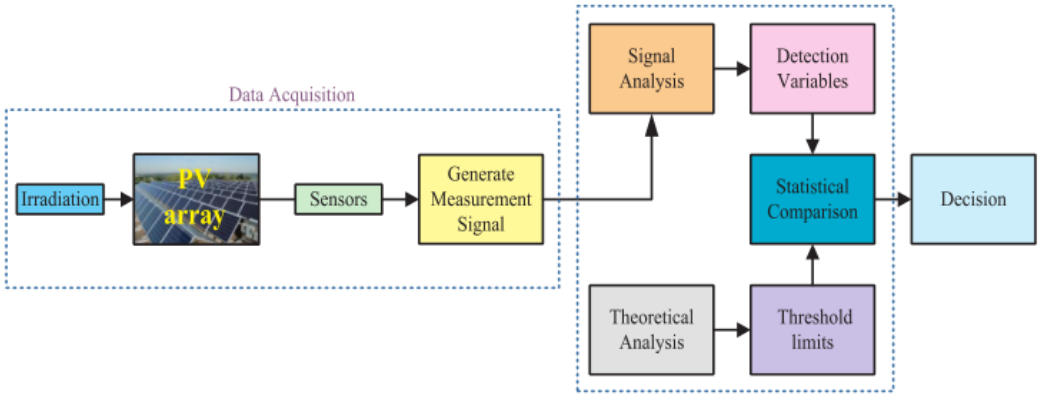

Figure 7 - OSA method for fault detection (4)

Zhao et al. (100) used OSA to analyze the output current signal of a PV string to detect outliers (101). The outlier detection rules that were tested included the Three Sigma Rule, Hampel Identifier and Box Plot Rule. Box Plot Rule was found to be the most accurate in detecting faults for all operating conditions. A quantitative approach based on Local Outlier Factor (LOF) was proposed to improve accuracy in outlier detection so that line to line faults can be detected (102) (103). This helped to filter out false positives that occur when basic statistical outlier rules are used.

3.3.5 Power loss analysis (PLA)

This method analyzes power losses in the system to detect and classify faults (75). Real-time PV system behavior is simulated by using climate data and PV system parameters calculated from monitored data. The simulated data are compared to the monitored data in order to detect power losses in the system and determine the type of a fault. A method to automatically monitor and detect faults based on PLA has been developed (75). The method defined two indicators, the current ratio and the voltage ratio, to compare the monitored DC variables with the corresponding simulated values. The ratios are calculated as follows:

$$RC = \frac{I_{PV-simulated}}{I_{PV-measured}}$$

$$RV = \frac{V_{PV-simulated}}{V_{PV-measured}}$$

The method was proven effective in detecting partial shading faults, faulty modules and strings, aging and MPPT errors (104). Another method includes comparing real-time monitored data with the simulated values, analyzing the DC current and voltage losses and comparing them with defined error thresholds to classify an error (90). Madeti and Singh (105) proposed a fault detection technique which analyzes the values at the terminals in order to determine inconsistencies on faulty PV strings and their corresponding array. The algorithm of the approach compared the monitored values with the voltage of a healthy PV string in order to detect unhealthy strings and faulty modules. The method embeds voltage sensors in key locations instead of using current sensors. The method was proven to be effective.

3.3.6 Climatic Data Independent (CDI)

The most commonly used CDI methods are the time domain reflector (TDR) and the earth capacitor measurement (ECM) (75). As the name implies, CDI methods do not use climate data (such as solar irradiance, temperature, humidity and wind speed) for fault detection. PV circuit parameters are measured with devices like the LCR meter (inductance (L), capacitance (C) and resistance (R)). In TDR, faults like the increase of the series resistance between modules and PV string positions are detected by the delay between the injected and the reflected signal in waveform changes. It measures the electrical characteristics of the transmission line to locate faults and impedance change from degradation (106). TDR should be used in set time intervals to detect array degradation. Takashima et al. (107) did experimental work using TDR. A different study by Takashima et al. (97) used ECM to detect the location where the PV string was disconnected. The location of the disconnected module n in a string of M modules can be approximated by the following formula:

$$\mathrm{n} = (\mathrm{C}_x/\mathrm{C}_D)\mathrm{M}$$

In the above C_D is the normal string capacitance and C_X is the capacitance of the defective string. ECM was to be used when the inspection was complete. Vergura et al. (108) developed a statistical method based on analysis of variance (ANOVA) and non-parametric Kruskale Wallis

(KW) (105). The method analyses the behavior of the individual parts of the PV system to detect faults and abnormal operation.

3.4 Imaging Techniques

3.4.1 Infrared (IR) or thermal imaging

In Infrared (IR) or thermal imaging, an infrared camera scans the PV array during operation. It measures differences in temperature on the surfaces of cells and modules, which can be the result of failures such as interconnection failures, module wiring failures, hot spots from internal short circuits, defective bypass diodes, change in the series resistance value, snail trails, cell mismatch and cell cracks (75). Localized heat generation can be due to shunted cells, poor contacts and short circuits. Also, when PV cells are connected in series, cells generating less power than others become reverse biased, act as a resistor and dissipate heat (105). Thermal imaging is divided into forward bias (FBI) and reverse bias (RBI). In FBI the PV module is connected to a power supply in a forward biased condition. Then a current twice as large as the short circuit current of the module (which is defined by the manufacturer) is passed through the module, causing it to heat up. Then, images are captured with an IR camera and future image processing is performed to detect and classify failures. This method reliably detects faults like hot spot, loose connections and increases in series resistance. In RBI, the procedure is the same but the power supply is in reversed biased condition, which can help detect ohmic shunts (105).

Thermal imaging is better for inspecting large PV installations, and is often used in combination with unmanned aerial vehicles (UAVs) (78). Aerial IR thermography costs more, but it is more accurate in assessing the performance of PV panels (109). Methods combining thermal imaging and visual cameras have been developed, and they are cheaper and more reliable than other methods (110). Thermal imaging is also used in most PV panel manufacturing processes to detect potential issues (111). Automated procedures for fault detection and classification (FDC) using thermal imaging have been proposed (112). Aerial thermal images have some issues confirming when a failure is actually detected or not from a visual perspective. There are also image quality issues caused by UAV altitude, observation angle and velocity (113).

Vergura et al. (114) and Guerriero et al. (115) researched FDC using thermal imaging, where an IR camera captures images that are processed to locate and classify failures. Vergura et al. calculate the mean value and deviation of the temperature for each cell and classify cells based on their temperature mean in order to detect faulty cells. Guerriero et al. used thermal gradient analysis and developed a method to recognize the edges of PV modules by spotting sharp differences between the temperatures of the metal frame and the nearby solar cells. Aspects that affected the performance of FDC using this method were the positioning of the IR camera, the distance from the PV array and the overlapping between consecutive pictures (75).

3.4.2 Ultrasonic Inspection

This method analyses ultrasonic vibrations that follow an excitation, in order to detect cracks and micro-cracks in the cells of a PV-module (105). It can detect unbounded cells and is

also used during the production phase to detect voids and de-bonded lamination structures (75). The PV module is scanned with a moving ultrasonic transducer along with the X-Y indicator. There are two types of ultrasonic inspection, the pulse-echo method and the transmission method.

In the pulse-echo method the PV module is scanned with ultrasonic pulses and the defects reflected back are recorded. This provides the location of a defect in all three dimensions and determines the causes of the degradation (75). This method can detect debonding of cells (105) and it is the most commonly used for this purpose.

The transmission method entails recording the attenuated ultrasonic signal. This method can also detect the size and location of a defect. Hund and King (116) proposed a failure analysis of PV modules which had field exposure for a long time. The study helped to identify the causes of module degradation. Both methods are shown in Figure 8.

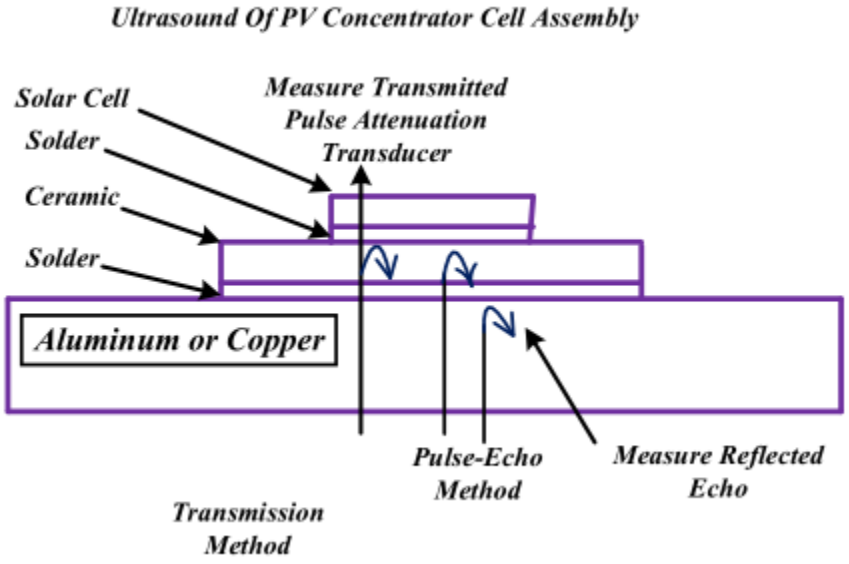

Figure 8 - Ultrasonic inspection methods (75)

3.4.3 Electroluminescence Imaging

In Electroluminescence imaging (EL imaging), ramped voltage is injected to the module and the resulting electroluminescence shows any defects. It is based on recording the photons emitted when excited carriers recombine into a PV cell (117). This can also be achieved by a radiation emission over the PV cell, where the light comes from photoluminescence (PL) (118). The larger the density of the current and the lifetime of the carrier, the more intense the emission of the EL images is. For that reason, it is used frequently for failure analysis (105). It can reveal non-uniform current, broken gates, small cracks and differences in conversion efficiency between cells. In the case of a crack in a cell, the image appears dark and it doesn't extend through the whole cell. It is a costly method and cannot be conducted while the module is in operation (75). The steps in EL analysis are displayed in Figure 9.

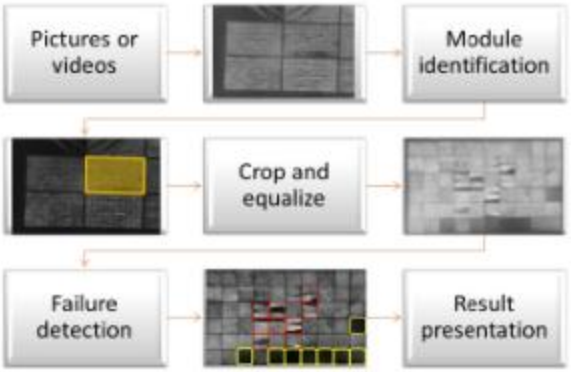

Figure 9 - Electroluminescence analysis (75)

3.4.4 Lock in Thermography (LIT)

In lock-in thermography the PV cells are checked for power loss (75). An excitation device together with a power supply inject pulse current in the PV module. Cells with defected shunts heat up from the current and then located using a thermographic camera. Alternations in the duration or magnitude of the electric pulses can help detect different types of shunt defects. This method detects small defects (119) and is generally not performed on modules while in operation. The test is performed either in the dark or under illumination (105).

3.5 Artificial Intelligence Methods

3.5.1 Machine Learning (ML)

Machine learning is commonly included in FDC for PV systems. This includes diagnosis and predictive diagnosis. Predictive modeling includes a) Data preparation, b) Training, c) Results post-processing and d) Validation (120). The steps are shown in more detail in Figure 10.

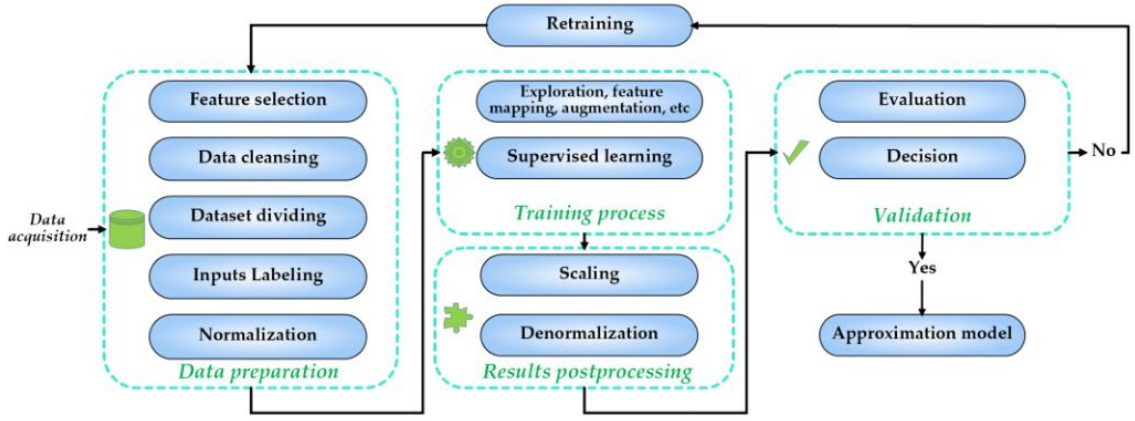

Figure 10 - Flow diagram for setting up a ML predictive model (120)

The type of the training model needed depends on how complex to problem is (121). Theoretically, an accurate predictive model must have training and testing data from the same probability distribution, but that is not possible in real-life applications (120). Machine learning

models are divided into a) Conventional ML, b) Advanced deep learning and c) Recent-knowledge driven methods like TL or GANs. All methods use different learning principles, like reinforcement learning, hybrid and ensemble. The classification is shown in Figure 11.

Conventional ML predictive models are created aiming to make the closest achievable approximation between the inputs and the outputs. They have ordinary representation and they have no deeper nonlinear abstractions. Such models are the Multilayer Perceptron (MLP), the Support Vector Machine (SVM) and the K-nearest Neighbor (KNN) (122). PV condition monitoring using conventional ML is divided to ordinary sensor-based and image acquisition-based (120).

Ordinary sensors include I-V, P-V, radiation and temperature sensors, and they are popular in applications with conventional ML for PV condition monitoring. For example, a Probabilistic Neural Network (PNN) was developed to detect irregular operating conditions (123). I-V signals from the DC converter were used to detect if they deviate when compared with previously recorded values. Learning data was collected from a simulation model. This study (123) analyzed four types of short-circuit fault modes.

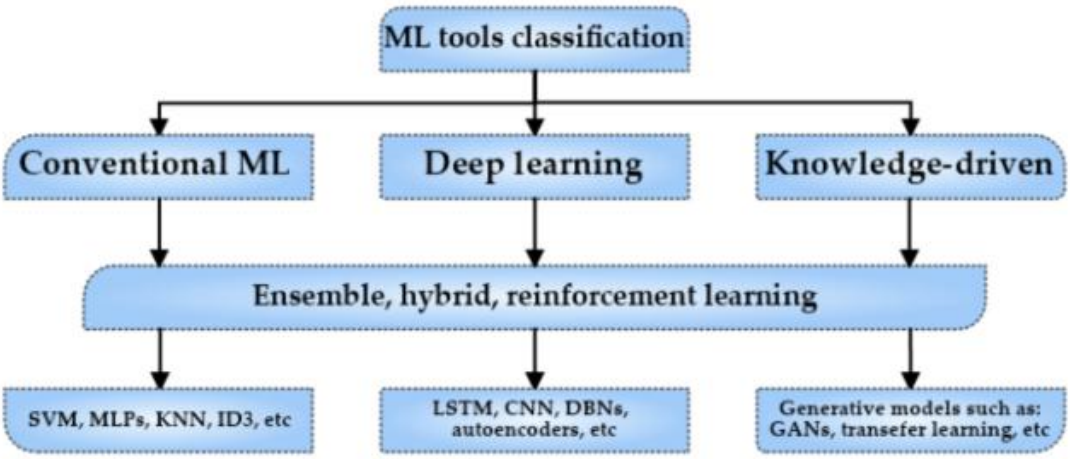

Figure 11 - ML tools classification (120)

Conventional ML for image acquisition requires analysis on larger datasets than the ones based used on ordinary sensor analysis. Few studies have been made on the use of conventional ML for higher dimensional image processing (120). Ali et al. (124) proposed the use of infrared thermographic images to detect degradation and shading faults from hotspots on PV modules. Several methods for extracting features have been proposed to get cleaner data. Data preprocessing includes image processing patterns, like RGB, texture, local binary pattern (LBP) and Oriented Gradient Histogram (OGH). SVM has shown the best results in this study (124).

3.5.2 Deep Learning (DL)

Deep learning (DL) is a type of Machine Learning with focus on representations and feature mappings. The representations become more meaningful as the improved feature space enlarges (120). Fault detection and diagnosis (FDD) using big data becomes more effective and

automated using the representation learning ability of DL (125). Its effectiveness relies on the mathematical tool used as well as the process models of the plant (126). Deep networks extract abstract and high-level features from the input data. The results become more accurate when the effective feature representation in the data is extracted (124).

Convolutional Neural Networks (CNN) are composed of a convolutional layer, a pooling layer and a fully connected layer (127), as shown in Figure 12. In the convolutional layer, features from the input data are extracted, matrix element multiplication is added to them in the perceptual field and then the deviation is added (128). The extraction of local spatial correlation features in the input data depends on the size of the convolution kernel, and can enhance the features of the signal and reduce the noise (127). The pooling layer reduces the spatial size of the convolved feature. It uses dimensionality reduction schemes to decrease the computational power needed and it can also extract relevant features that have no variation in rotation or in position, so it can train the model effectively (129). A fully connected layer can be added to train the model in non-linear combinations of the high-level features. Using CNN in fault diagnosis has the benefit that the data comes from many sources (130), as CNN inputs can be spectrograms (131), images (132) or time series (133). Another benefit is that the extracted CNN input has translation invariance (134) which is important for the generalization of the algorithm as complex systems can have high temperatures and magnetic interference (125).

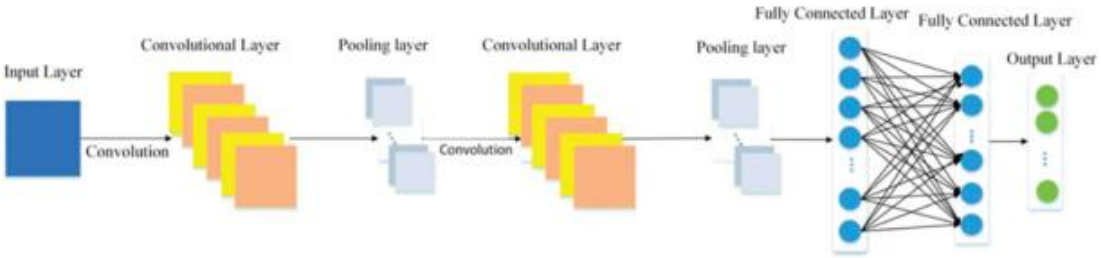

Figure 12 - Architecture of a CNN (125)

In Recurrent Neural Networks (RNN), the nodes are linked in a chain and the input features are time series (135). RNNs use previous states of the network in order to learn sequences that change over time (136). Popular RNNs include Gated Recurrent Unit (GRU) networks and Long Short-Term Memory networks (LSTM) (125), which are designed for long sequence prediction, as RNN itself has a length limit (137). Gates allow the recurrent units to learn how different time scales are dependent, in order to avoid long-term dependence (138). The benefit of the RNN in PV fault detection is that it is capable of making predictions for dynamic systems, as the input features are time-series and the depth is proportionate to the input length (125). Also, the sampling in PV systems varies in length; the predictions need to be unaffected by that; RNN has that advantage (125). Another benefit is that RNN can properly work with dynamic non-linear systems because it is Turing complete (125).

Stacked Auto Encoder networks are neural networks with multiple hidden layers, which are stacked together (139). The layers' inputs and outputs are connected in chain (140).

Networks consist of: 1) An encoder, which converts the input into the hidden layer representation, and 2) a decoder, which turns the representation back into an input (125). These networks use multiple nonlinear mappings to calculate higher order input representations (141). Their advantages for PV fault diagnosis include: 1) That they are capable of processing 1D signals like the data from PV systems, 2) that they are capable of unsupervised training needed for unlabeled PV data and 3) that SAE networks' layer-chaining way of operation can deal with high order nonlinear input without dispersing the deep network (125).

Deep Belief Networks (DBN) are models composed of 1) multiple hidden layers within Restricted Boltzmann Machines (RBM) (which are networks composed of a visible and a hidden layer) as well as 2) an output layer. An RBM has independent activation conditions 1) for each hidden layer, when features are inputted, as well as 2) for the visible layers, when the hidden state is inputted (142). An RBM uses an energy function to show the high order interaction between variables (125). A DBN in PV fault diagnostics has the benefit that it can predict probability distributions without restrictive assumptions. It is also capable of simulating nonlinear systems with multiple variables as it uses feature grouping sequences for activation value sets. Finally, it is capable of generating more samples, in case the number of samples is limited, because it uses unsupervised learning.

3.6 Device-based Techniques

There are several standard devices for protection and fault detection in PV systems. Each can detect specific types of faults and has its own advantages and disadvantages.

Ground Fault Detector Interrupters (GFDI) analyze the DC rating of the inverter and can detect single and double ground faults. They are cheap and easy to implement but they are sensitive to leakage currents and are unable to see blind spots (4).

Overcurrent protective devices (OCPD) analyze the current rating of a PV string and can detect double ground faults and line-to-line faults. They are also cheap and operate passively, but they are only effective for grounded PV units and do not respond fast enough to dangerous fault currents (4).

A Residual Current Device (RCD) analyzes both the current and the voltage of a PV string or array to measure residual current. It can detect single ground faults. Unlike OCPD, it can operate in ungrounded PV systems as well as grounded ones, and it is also sensitive to high impedance faults. Its disadvantages include that it can exhibit false positives due to external noise, it is only accurate when combined with a GFDI, and it also consists a shock hazard (4).

Insulation monitoring devices measure insulation resistance based on the rated voltage of the PV array. It can detect double ground faults. They are reliable and can perform tests during the night as well as during the day. However, the insulation resistance can vary with the environmental conditions and any inverters and fuses need to be isolated to be tested in grounded systems (4).

Arc fault detector (AFD) and Arc Fault Circuit Interrupter (AFCI) devices analyze the waveform of the output current and voltage. They can reliably detect series arc faults and protect a PV unit against them. They cannot, however, detect parallel arcs, and can show false positives due to interference from the converter switching (4).

Earth capacitor measurement (ECM) is a technique that analyzes capacitance to detect open circuit faults. It is accurate and doesn't depend on the level of solar irradiance to work but the PV system needs to be offline for the experiment to be performed. A Line checker can also detect open circuit faults without the need for the unit to go offline. However, it is more time consuming (4).

3.7 Predictive Maintenance through Real-time Sensors

Predictive maintenance through real-time sensors is the most accurate and most costly fault detection method (78). In this case, fault detection is performed through real time sensors, which can also be wireless (143). Proposed systems used sensors to measure temperature (ambient and on the module), solar irradiance, open circuit voltage, short-circuit current, fill factor and panel efficiency (78), (144), (143). The sensors that are used include the thermocouple for temperature, pyranometer for irradiance, shunt resistor for current, and voltage divider for voltage (78). The use of the sensors gives a global view of the PV system and a view of the way the inverter interacts when a shadow is present. Software used included Visual Basic (144) and LabVIEW (145). A PV analyzer must be used, like the Solmetric I-V curve tracer. An Arduino can be used as a PV analyzer with similar results (146). FDD algorithms are also used, which compare a number of parameters with the limits observed in a healthy system, in order to detect partial and total loss of productivity (90). A system for FDD using wireless sensors has also been proposed (147) which analyzes the data in MATLAB, allowing efficiency optimization.

3.8 Hybrid detection techniques (HDT)

Hybrid detection methods are a combination of two separate methods and are developed to increase accuracy, reduce demands in computing power, give the ability to tell between two different failures with the same signature, and detect when more than one fault occurs at once.

Chine et al. (148) proposed a combination of RDM with MLT for fault detection. When failures with unique fault signatures were detected, RDM was used, and when failures sharing the same signature were detected, MLT using ANN was used. Hu et al (149) developed a model using energy balance with the conventional cell model to correlate the electrical and thermal properties of a system. The ITH method was used to make the hybrid model. The model was used to measure panel temperature in order to make predictions. The criterion to detect a fault was how much U_{pv} changed among different panels. Kase and Nishikawa (150) used ITH to locate hotspots and injected reverse DC bias voltage in the PV string to detect open circuit failures. Yi and Etemadi (151) used SVM to detect line to line faults, combining OSA and MLT.

MSD wavelet packets were used for the SVM training stage. The method did not detect accurately line to line faults that had less than 20% mismatch level (4).

3.9 Conclusions

The conclusions reached in this chapter were the following:

- Visual inspections are useful as a preliminary test to determine whether further testing is needed.
- Electrical Characteristics measurements detect open and short-circuit faults, hotspots, ground faults, arc faults, shade faults and aging errors.
- Imaging methods can detect cracks and micro-cracks, interconnection and wiring failures, hotspots, increase in series resistance, defective bypass diodes, non-uniform current and differences in conversion efficiency between cells.
- Machine learning uses data collected from both imaging and electrical measurements and can detect both failures that have already occurred and failures that can occur in the future.
- Specialized devices exist to detect specific types of failures.

Chapter 4: Proposed methodology

4.1 Intro

In this chapter, the proposed methodology for this research is explained. A program was developed in Python to perform forecasting using collected data from historical energy production and weather data. The models used were the Decision Tree, SVM, Linear Regression, and two hybrid models using a combination of the above.

In section 4.2, the way the selected models work is explained. The related theory is analyzed with the appropriate examples, diagrams and formulas.

In section 4.3, the method of pre-processing the data is explained step-by-step. The data needed to be filtered of some measurement errors and formatted appropriately in order to be usable by the machine learning models.

In section 4.4, the correlation between the available data and the measured power output is examined in order to assess which variables will be useful as input features during the testing.

In section 4.5, conclusions are made on the preliminary analysis of the data and reasoning is provided for the selection of the models.

The proposed methodology can be summarized in the following flow diagram:

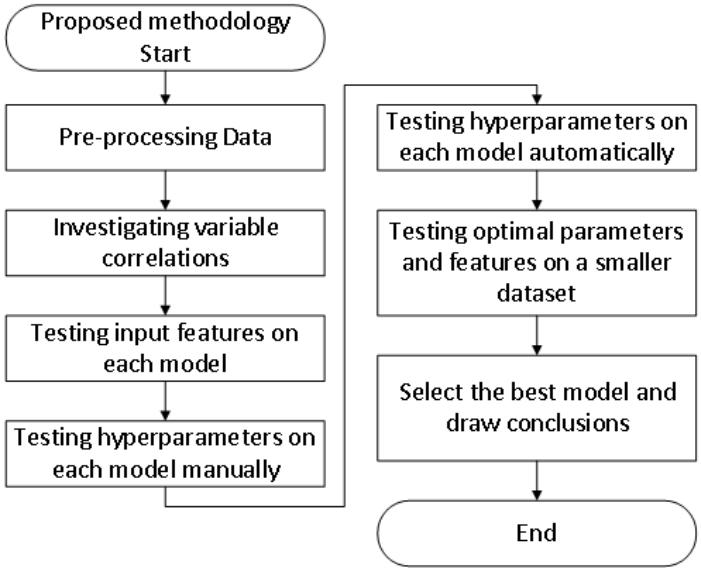

Figure 13 - Flow Chart of the Proposed methodology

4.2 Utilized ML models

4.2.1 Decision trees

Decision trees are a supervised learning method (152). It can be applied for regression, as is the case in this study, as well as classification tasks. It is based on the idea of partitioning a

complex decision into multiple simpler decisions. It uses a structure similar to a tree in the steps performed to make decisions. The tree is composed of nodes. At the root node, all data is included. In each following internal node, a binary decision is made to split the data between different classes or groups of classes. In the case of a regression task the split is made so that the data in the separate classes will have the minimum deviation from the mean value in their respective class (153). This is repeated until the terminal, "leaf" nodes are reached, which represent the final separate classes. In the case of regression, leaf nodes are defined by the number of training samples at the node. This is the top-down approach. In order to optimize the cost function, a bottom-up approach and a hybrid approach can also be used.

Decision trees assume that the relationship between the input and output variables is either linear or non-linear. This method is suitable for handling non-linear relationships. The inputs that possess the most information are used for the classification and the rest are rejected. As such, only a small amount of inputs is used in the procedure. The size of the tree needs to be carefully adjusted to avoid over-fitting as well as under-fitting. In order to avoid overfitting, pruning is also performed to reduce the variance of the output variable (153).

The tree structure also makes the output easy to interpret, unlike the output of Neural Networks. The trees can also be visualized. The input does not need standardization or extensive preparation, although the input may need to be balanced or it can result in biased predictions (152). It can be used for problems that have multiple dependent variables. The results can be tested using statistics. On the negative side, decision trees are sensitive to small variations in the data, producing very different trees a result of such variations. They are also unsuited for extrapolating as their predictions are not continuous. Decision trees are best used in an ensemble as they are not reliable in calculating the optimal tree on their own.

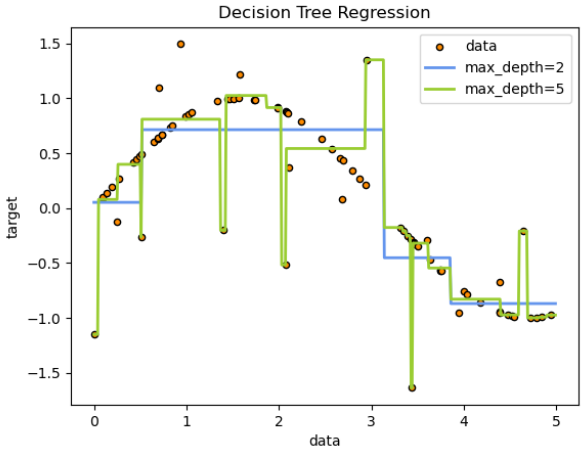

Figure 14 - Over-fitting in decision tree regression as seen in the green line (152)

4.2.2 Support Vector Machines

Support Vector Machines are also a supervised learning method for classification, regression and outlier detection. It is based on the concept that each class must have the

maximum possible distance from the rest. When used for regression, it is called Support Vector Regression (SVR) (154). SVR is a machine learning method in which a model learns the importance of a variable by defining the relationship between the input features and the dependent variable (155). Depending on the kernel function used, it can be linear or non-linear (154). Linear kernel functions for SVR are like support vector machines, but use an approximation tolerance margin (ε).

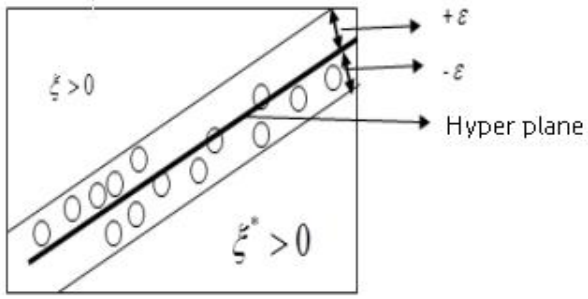

Figure 15 - Support Vector Linear Regression (154)

The kernel function for linear SVR is:

$$y = w.x + b$$

Where y is the input space, w.x is the vector product and b is a constant. The error function is:

$$min\frac{1}{2}||w^2|| + c\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

Where:

$$z_i - (w.x + b) \leq \varepsilon + \xi_i$$

$$(w.x) + b - z_i \leq \varepsilon + \xi_i^*$$

$$\xi_i \xi_i^* \geq 0$$

The figure and the kernel for non-linear vector regression are presented below:

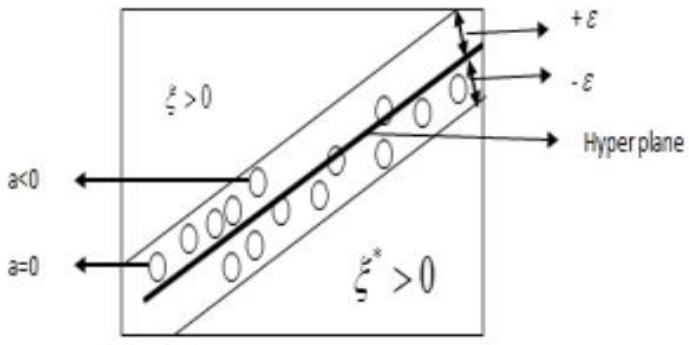

Figure 16 Support Vector Non-Linear Regression (154)

$$max \begin{cases} \frac{1}{2}\sum_{i,j=1}^{n}(a_i - a_i^*)(a_j - a_j^* k)(x_i x_j) \\ -\varepsilon\sum_{i=1}^{n}(a_i - a_i^*) + \sum_{i,j=1}^{n} y(a_i - a_i^*) \end{cases}$$

With the limitations: $\sum_{i=1}^{n}(a_i - a_i^*) = 0, 0 \leq a_i, a_i^* \leq C$

SVM is effective in high dimensional spaces as well as in cases where the samples are fewer than the dimensions. Out of all the training points, it only uses the support vectors, so it uses less memory than other algorithms. There is a tendency for over-fitting when there are more inputs than samples, which requires fine-tuning in choosing kernel functions and regularization to compensate (156).

4.2.3 Linear regression

Linear regression uses the Least Squares method to minimize the error in the correlation between the inputs with the output in a linear relationship. A number of coefficients are used for that purpose (157). The general concept of a linear model is expressed in the following equation:

$$y(w, x) = w_0 + w_1 x_1 + \cdots + + w_n x_n$$

where y is the output, x_i are the inputs and w_i the coefficients. In linear regression using the Least Squares method, the aim is to minimize the following function:

$$\min \ ||Xw - y||_2^2$$

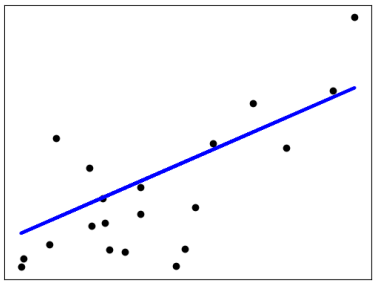

Figure 17 – Simple Linear Regression (157)

Linear regression models, like decision trees, can be trained quickly and are easy to interpret (158). The input features' deviation affects the model's accuracy, and the output variable needs to follow Gaussian distribution in relation to each input feature. This model cannot handle non-linear relationships between input and output variables. Categorical variables need to be converted to binary or constant in order to be used in linear regression. The coefficients assume the inputs are independent from each other; if they are correlated the results would deviate greatly due to random errors (157). It is also required that the variables are continuous with no significant outliers.

4.3 Data pre-processing

The weather and energy production data received required some pre-processing to be used in forecasting. The meter measuring the production data presented errors which resulted in missing values in the data, and in two cases that extended to a range of several months. Another error resulted in some values in a particular date to repeat twice for specific times. In the original dataset, negative production values due to reverse power flow were present, as the meters are connected to the grid. Also, due to noise from the grid, non-zero values were present at night, which do not represent actual energy production and therefore needed to be trimmed out. The weather data had also some individual missing values. Another issue was that the weather data were given in hourly values, while the production data were given in values with 10-minute intervals.

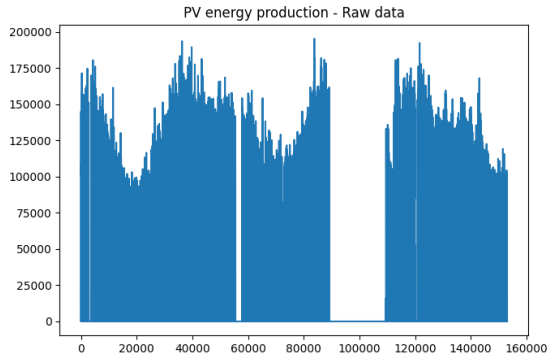

Figure 18 - PV production data before pre-processing

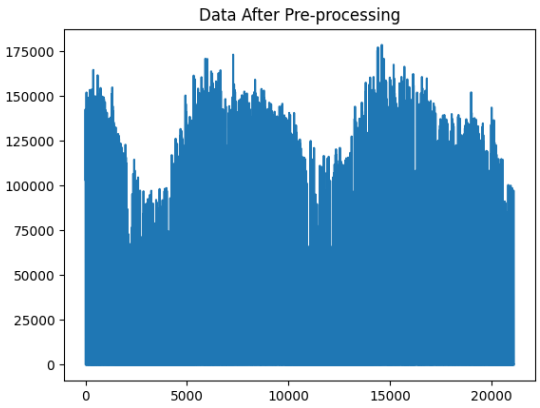

Figure 19 PV production data after pre-processing

The biggest gaps in production data were too large to fill in, so their respective time ranges were omitted from this analysis. However, the timing is checked for hourly consistency in the code, so in order to omit the time range with the missing values, the data was split into two parts, which were pre-processed separately and then joined together at the end of the procedure.

```
for i in range(89222): # Seperate the part before the missing values
    pvdata1.append(pvdata[i])
for j in range(112310, len(pvdata)): # Separate the part after the
missing values
    pvdata2.append(pvdata[j])
```

The gaps that last only a few days were filled in with the average value between the next and the previous week at the same hour and day of each value. In one case, where the gap lasted 16 days, the average value between the previous and the next month was used. The gaps appeared as a set of days with zero energy production and were manually detected, as automatic detection could potentially fill in values where there was actually no actual production. As such, large ranges in which data was to be filled were defined manually in the code:

```
if datetime.datetime.fromisoformat('2019-09-09 00:00:00') >=
datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) >=
datetime.datetime.fromisoformat('2019-08-23 00:00:00') and correctingpv
== 1: # If specific dates with known missing values match, use the
average data from the previous and next month

      bhournew.append([(datetime.datetime.fromtimestamp(time.mktime(bho
      ur[i][0]))).timetuple(),(bhour[i-30*24][1]+bhour[i+30*24][1])/2])

elif datetime.datetime.fromisoformat('2018-08-27 00:00:00') >=
datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) >=
datetime.datetime.fromisoformat('2018-08-22 00:00:00') and correctingpv
== 1: # If specific dates with known missing values match, use the
average data from the previous and next week

      bhournew.append([(datetime.datetime.fromtimestamp(time.mktime(bho
      ur[i][0]))).timetuple(),(bhour[i-7*24][1]+bhour[i+7*24][1])/2])

elif datetime.datetime.fromisoformat('2021-05-09 00:00:00') >=
datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) >=
datetime.datetime.fromisoformat('2021-05-05 00:00:00') and correctingpv
== 1: # If specific dates with known missing values match, use the
average data from the previous and next week

      bhournew.append([(datetime.datetime.fromtimestamp(time.mktime(bho
      ur[i][0]))).timetuple(),(bhour[i-7*24][1]+bhour[i+7*24][1])/2])
```

Any zero values besides those manually detected as abnormal were treated as normal. However, there were also missing time ranges in smaller ranges, in which the entire timestamp was missing and there was no value at all for a particular time. In order to correct that, the dataset was initially converted from 10-minute intervals to hourly values - which was necessary anyway because the timestamps need to match with the weather data. To perform the

conversion, all 10-minute values of a particular hour are added together and divided by the amount of values they consist of. Normally that would be six, but the code was designed to calculate the average with less in case a number of values was missing. This way individual 10-minute values are compensated for in case they are missing.

```
for i in range(len(btime)-1): # Omit the last element of the list to
solve the out of index on btime[i+1] error
      hourlyproduction += buildingdata[i][2]
      hourcounter += 1 # Some elements are missing, not always 6 per
      hour, so they are counted to calculate Wh
      if btime[i].tm_hour != btime[i+1].tm_hour:
            currenthour = list(btime[i])
            currenthour[4] = 0 # Set the timestamp minutes to zero
            currenthour[5] = 0 # Set the timestamp seconds to zero

            bhourly.append([time.struct_time(tuple(currenthour)),hourly
            production/hourcounter])
            hourcounter = 0
            hourlyproduction = 0
```

Afterwards the entire dataset is checked for coherence. Every single value must have the previous hour as its previous value and the next hour as its next value. If this is not the case, the value is filled in, if the gap is smaller than 24 hours. Otherwise, the code will display an error, allowing the user to manually handle the issue:

```
if ((datetime.datetime.fromtimestamp(time.mktime(bhour[i+1][0])) -
datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])))  >
datetime.timedelta(days=1)):
     print('Error, missing values!', bhour[i][0])
     print(bhour[i][0], 'is where the value is missing')
     sys.exit("Error message")
```

In case it is smaller than 24 hours, there are two ways to handle it:

A) If the current time is the same as the previous time and the next time is two hours later, that means the gap only appears because of the Daylight-Savings Time and the way Python handles it, so it should be left as is.

```
if bhour[i][0] == bhour[i-1][0] and
((datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) -
datetime.datetime.fromtimestamp(time.mktime(bhour[i+1][0]))) ==
datetime.timedelta(hours=2)):

   bhournew.append([(datetime.datetime.fromtimestamp(time.mktime(bhour[
   i][0])) + datetime.timedelta(hours=1)).timetuple(),bhour[i][1]])
```

B) If A) is not the case, each missing value is filled with the average between the value appearing 24 hours before and 24 hours after. This is repeated until there is no gap between the current and the next timestamp.

```
if ((datetime.datetime.fromtimestamp(time.mktime(bhour[i+1][0])) -
datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])))  >
datetime.timedelta(hours=2)):
      hours_missing =
      datetime.datetime.fromtimestamp(time.mktime(bhour[i+1][0])) -
      datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) -
      datetime.timedelta(hours=1)
      counter1 = 0
      bhournew.append(bhour[i])
      while hours_missing > datetime.timedelta(hours=0):
            fillervalue =
            [(datetime.datetime.fromtimestamp(time.mktime(bhour[i][0]))
            +
            datetime.timedelta(hours=1+counter1)).timetuple(),(bhour[i-
            24+counter1][1]+bhour[i+24+counter1][1])/2]

            bhournew.append(fillervalue)
            counter1 += 1
            hours_missing -= datetime.timedelta(hours=1)
```

In the original dataset, cleanup was also performed for non-zero values at night as well as negative values. An updated dataset was provided later on which had performed this clean-up beforehand:

```
for i in range(len(pv)):
      if pv[i][2] < 0:
            pv[i][2] = -pv[i][2] # Retrieve absolute power values to
            correct reverse power flow values
      if pv[i][2] < 100 and ( pvtime[i].tm_hour > 17 or
      pvtime[i].tm_hour < 7 ):
            pv[i][2] = 0 # Eliminates noise - Power produced after 5pm
            and before 7am less than 100 Watt is set to zero
```

The weather data did not have missing timestamps, only individual NaN values at specific times, which were filled using the value of the previous hour:

```
if np.isnan(weatdata[i][j]):
    weatdata[i][j] = weatdata[i-1][j]
```

The time range of the weather data did not overlap entirely with that of the PV production data. Only the time range existing in both datasets could be used. So, for each of the PV datasets, a weather dataset was selected with the same time range:

```
def keepmatchingdatesonly(pvhour,wdata): # Keep only the data for each
set corresponding to the same dates and times

    pvcommondata = []
    wcommondata = []
    lateststart = 0
    earliestfinish = 0
    lateststart =
    max(time.mktime(pvhour[1][0]),time.mktime(wdata[1][0]))
    earliestfinish = min(time.mktime(pvhour[-1][0]),time.mktime(wdata[-
    1][0]))

    for i in range(len(pvhour)):
        if earliestfinish >= time.mktime(pvhour[i][0]) >= lateststart:
            pvcommondata.append(pvhour[i])
    for i in range(len(wdata)):
        if earliestfinish >= time.mktime(wdata[i][0]) >= lateststart:
            wcommondata.append(wdata[i])
    return pvcommondata, wcommondata

# Get weather timestamps
weatherdata1 = weatherdata
weatherdata2 = weatherdata

# Keep only the common dates among all data sets
pvhourly1, weatherdata1 = keepmatchingdatesonly(pvhourly1,
weatherdata1)
pvhourly2, weatherdata2 = keepmatchingdatesonly(pvhourly2,
weatherdata2)
```

A detail which was important to the processing of the data was the way the time and date was stored. The timestamp data was converted to a time structure in order to process it further. The weather data did not have time zone information while the PV production data did, which presented an incompatibility between the two. It was assumed that the weather data had the same time zone as the production data, +01:00. As such, the timestamps for the weather were imported using:

```
wtimestring = str(int(weatdata[i][0])) + "+01:00"
weatdata[i][0] =
datetime.datetime.strptime(wtimestring,"%Y%m%d%H%M%z").timetuple()
```

And the production data were imported using:

```
btime.append(datetime.datetime.strptime(btimestring,"%Y-%m-
%dT%H:%M:%S%z").timetuple())
```

In order to prepare the input features for the prediction process, the timestamps of the weather data were cross-referenced with the production data to ensure the input of each value corresponds to the same timestamp for both sets. If a mismatch is detected, the program stops displaying an error, showing the time where the mismatch was detected. The input features that were prepared included humidity, wind speed, cloud cover, solar radiation, time, day of the month, month, season, hour and weekday.

```
if pvhourly[i][0] == weatherdata[i][0]:
    try: # Will raise an error when trying the time when DST starts,
ignore that value -will omit a (erroneous) value of the original index
      wvalues2humid.append(weatherdata[i][2])
      wvalues3wind.append(weatherdata[i][3])
      wvalues4cloud.append(weatherdata[i][4])
      wvalues5rad.append(weatherdata[i][5])

      datesinstring.append(time.strftime('%Y-%m-%d %H:%M:%S',
pvhourly[i][0]))
      day.append(time.strftime('%d', pvhourly[i][0]))
      month.append(time.strftime('%m', pvhourly[i][0]))
            # Season
      if 3 > int(time.strftime('%m', pvhourly[i][0])) >= 6:
          season.append(2) #Spring
      elif 6 > int(time.strftime('%m', pvhourly[i][0])) >= 9:
          season.append(1) #Summer
      elif 9 > int(time.strftime('%m', pvhourly[i][0])) >= 12:
          season.append(3) #Autumn
      else:
          season.append(4) #Winter
      year.append(time.strftime('%Y', pvhourly[i][0]))
      hour.append(time.strftime('%H', pvhourly[i][0]))
      weekday.append(time.strftime('%w', pvhourly[i][0]))
      pvvalues.append(pvhourly[i][1])

      wvalues1temp.append(weatherdata[i][1])
else:
      print("Error: Date mismatch")
      print(pvhourly[i][0])
```

The pre-processing methodology can be summarized in the following flow diagram:

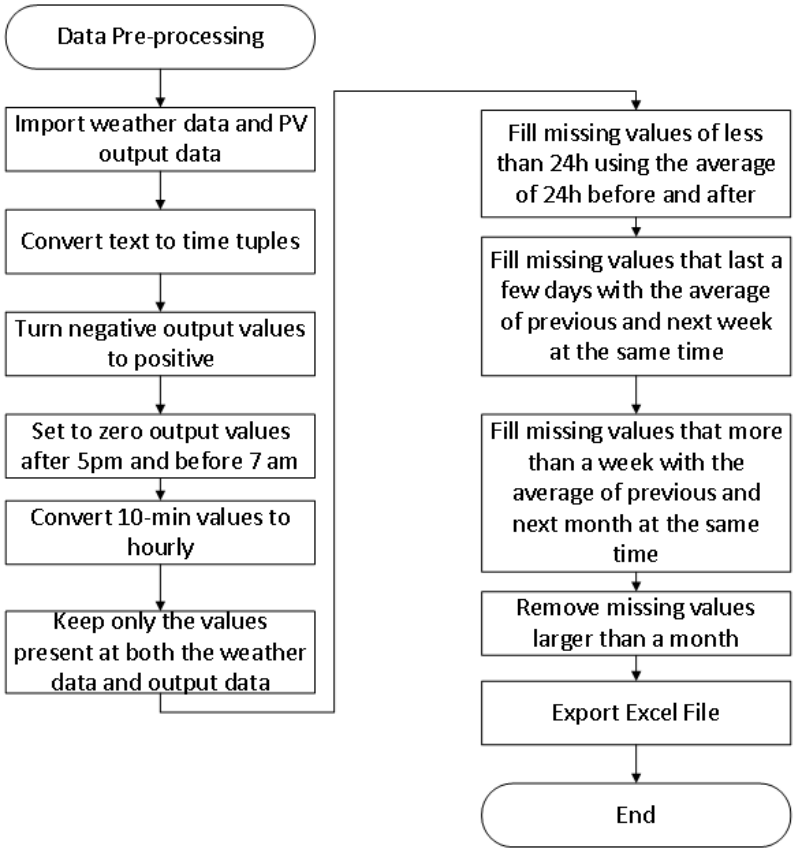

Figure 20 - Pre-processing methodology flow chart

4.4 Investigating input feature correlations

Autocorrelation was tested in the energy production data using the following script:

```
plt.figure(3)
plt.title("Autocorrelation of PV energy production")
plt.acorr(pvvalues,maxlags=10)
```

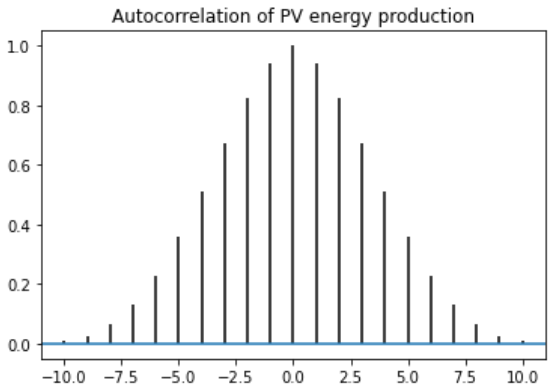

Figure 21 Autocorrelation of PV energy production

It was observed that autocorrelation is over 0,5 for up to 4 lags/hours. As such, up to 4 hours lag elements were to be investigated as possible input features for the forecasting.

In order to select which input features should be used for prediction, the correlation of each with the output variable should be checked. The following code was used to draw a correlation matrix:

```
correlation_mat = data.corr()
ax = sns.heatmap(correlation_mat, annot = True,fmt='.2f')
ax.figure.subplots_adjust(left = 0.3, bottom = 0.3)
plt.show()
```

The result was the following:

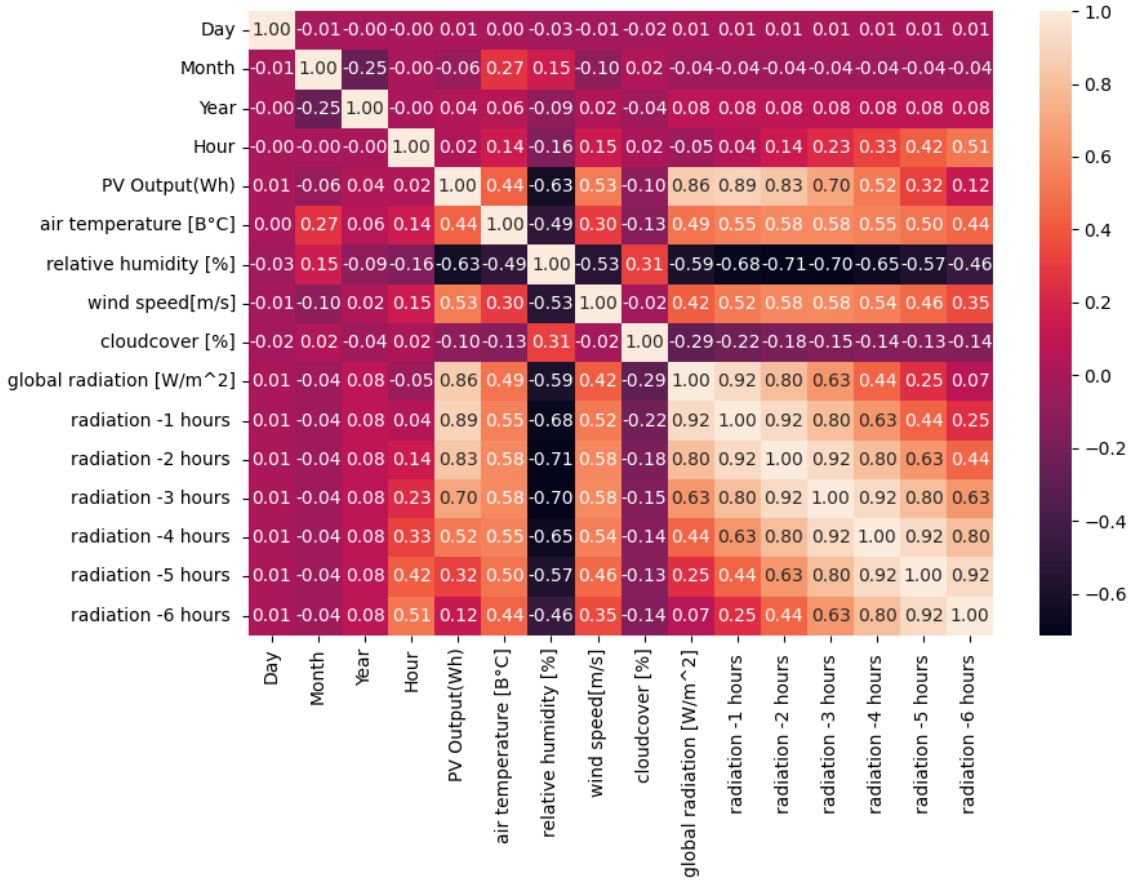

Figure 22 - Correlation matrix of the available variables

For a more aimed analysis at the PV output, the numpy command corrcoef was used. So, for example, the following command:

```
np.corrcoef(wvalues1temp,pvvalues)[0, 1]
```

was used to calculate the correlation between the ambient temperature and PV energy output. The following results were retrieved:

Temperature-PV Production Correlation	0,436661
Humidity-PV Production Correlation	-0,6306
Wind Speed-PV Production Correlation	0,527246
Cloud Cover-PV Production Correlation	-0,09673
Solar radiation-PV Production Correlation	0,85641
Solar radiation – Previous hour	0,89107
Solar radiation – 2 hours before	0,82966
Solar radiation – 3 hours before	0,69894
Solar radiation – 4 hours before	0,52197
Solar radiation – 5 hours before	0,32232

It was observed from these results that cloud cover has next to zero correlation so it should not be used for predictions. The Temperature also appears to be poorly correlated to PV output in this case study. Solar radiation, as expected, has very high correlation with the output.

Time series decomposition was also performed to identify trend and seasonal components on the PV production data.

```
seasonalpv = stm.tsa.seasonal.seasonal_decompose(pvvalues,
period=24*30) # Monthly intervals
statsmodels.tsa.seasonal.DecomposeResult.plot(seasonalpv,
observed=True, seasonal=True, trend=True, resid=True, weights=False)
```

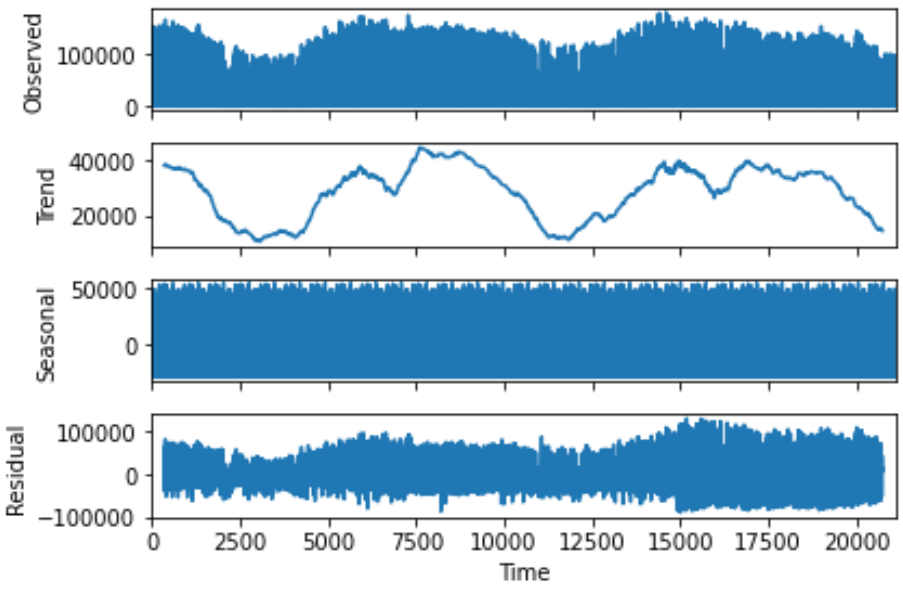

Figure 23 - Time series decomposition of PV production data

It was observed that the displayed trend was actually a seasonal loop throughout each year. Production peaks during summertime and is reduced in winter. It was concluded that the categorical features of Season and Month should be included in the input variables that were to be tested. Season was also tested, derived from the Month of the Year; for example, measurements with Month values 4, 5 and 6 (April, May and June) were assigned a Season value of “Spring”. The Hour of the day, ranging from 0 to 23, was also used as an input feature.

The Hour and Month features were tested both as an integer and as a categorical feature using OneHotEnconder.

4.5 Conclusions

Decision Tree, SVR and Linear regression were selected for this research for several reasons. An important factor was the well-supported tools available in scikit-learn which provide the ability to use those models reliably. Another factor was the large range of hyperparameters of the Decision Tree and SVR, which provided the ability to optimize the results.

While publications utilizing a form of ANN were common in literature, research utilizing Decision Trees and Linear Regression was uncommon. Testing their efficiency against the more commonly used SVR was deemed to contribute to a subject less explored in literature.

Another factor was the need to test the linearity of the problem that is being investigated. From a physics standpoint, the relation between direct solar radiation and PV power output could theoretically be linear. If other input features are added, that would change. SVM can operate both in a linear and a non-linear kernel. Decision Tree is a non-linear algorithm, while Linear Regression is linear. Different tests comparing all three models can ensure an optimal model is found both for the eventuality of the problem being linear and otherwise. An ensemble of a linear and a non-linear model can potentially tackle a problem that contains both linear and non-linear components.

Radiation data of the previous hours showed a very high correlation with the energy output. This is supported by the fact that the production autocorrelation was high for up to 4 hours before. This led to the conclusion that the solar radiation values of the past 4 to 5 hours must be examined in the test. Cloud cover had almost zero correlation and was excluded from testing.

The time series decomposition displayed a seasonal factor that determined that elements of the timestamp, such as month or season, must be included as input features.

Chapter 5: Results

5.1 Introduction

In this chapter, the series of tests performed in this research is reported and commented on.

In section 5.2, a series of tests is performed to determine the best set of input features for each model. The performance of each model in each case is displayed using different indicators.

In section 5.3, tests were performed to optimize the hyperparameters of each model. The optimization was performed first manually, reporting the accuracy of each model and selecting the best set by hand. Afterwards it was performed automatically, with the help of the GridSearchCV function of the scikit-learn tools.

In section 5.4, tests using the best parameters are repeated using a smaller test set and the results are compared with those of the models using the larger dataset.

In section 5.5, the collected results are discussed and conclusions are made.

5.2 Input feature optimization

Three error values were used to assess the results: Root Mean Square Error (RMSE), mean absolute error (MAE), and the coefficient of determination (R^2 or R2). Using scikit-learn, there were two ways to calculate R2: Either using *sklearn.metrics.r2_score*, or using the "*score*" method of the individual prediction models, such as *sklearn.tree.DecisionTreeRegressor.score.* Both results are included here as they sometimes varied significantly. The *metrics* result is referred to as R2 1 and the result from the individual models is referred to as R2 2. The methods that were used for the first tests were:

1) A Decision Tree Regression model with squared error criterion, maximum depth of the tree set to 15, the minimum number of samples required to split an internal node set to 4 and the minimum number of samples required to be at a leaf node set to 1. The rest of the values were set at default.
2) A Support Vector Regression model with C (Regularization parameter) set to 20, epsilon set to 0,1 and a linear kernel.
3) A Linear Regression model.

For the models 2 and 3 the input features were normalized using StandardScaler. The available data was divided to a train set and a test set. For the first part of the testing, the train set was comprised of all the data except the final week (168 hourly values). The final week was used as a test set. The results are presented below.

Using Solar Radiation alone as an input feature:

	Decision Tree Method	SVR	Linear Regression
RMSE	13453 W	12236 W	20874 W
MAE	6511 W	6047 W	13661 W
R2 1	0,767	0,686	0,698
R2 2	0,780	0,818	0,470

Using Solar Radiation and Temperature as input features:

	Decision Tree Method	SVR	Linear Regression
RMSE	17636 Wh	12236 Wh	20696 Wh
MAE	7472 Wh	6047 Wh	13661 Wh
R2 1	0,708	0,686	0,702
R2 2	0,622	0,818	0,480

It is observed that using Temperature as an input feature makes the results of the Decision Tree much worse, does not affect SVR and very marginally improves Linear Regression. Therefore, it was concluded that it should not be used for the tests that followed.

Using Solar Radiation and Humidity as input features:

	Decision Tree Method	SVR	Linear Regression
RMSE	19755 Wh	12236 Wh	20401 Wh
MAE	8569 Wh	6047 Wh	14491 Wh
R2 1	0,351	0,686	0,701
R2 2	0,526	0,818	0,494

Similar to Temperature, Humidity is not helpful for the forecasting. This is not out of the ordinary, as the literature states the correlation of weather data with PV production varies depending on the case study.

Using Solar Radiation and Wind as input features:

	Decision Tree Method	SVR	Linear Regression
RMSE	21576 Wh	12235 Wh	22144 Wh
MAE	9442 Wh	6056 Wh	14483 Wh
R2 1	0,414	0,686	0,672
R2 2	0,435	0,818	0,404

Similar to the previous weather data, it reduces the accuracy of the models.

Using Solar Radiation for the current and previous hour:

	Decision Tree Method	SVR	Linear Regression
RMSE	13986 Wh	9771 Wh	19925 Wh
MAE	5905 Wh	4864 Wh	13331 Wh
R2 1	0,843	0,812	0,749
R2 2	0,762	0,884	0,518

While the Radiation of the previous hour does not affect the Decision Tree significantly, it considerably improves the results of the other models. This is expected from the correlation values. As such, this variable was included in all following tests.

Using Radiation for the current hour, 1 hour and 2 hours before:

	Decision Tree Method	SVR	Linear Regression
RMSE	10628 Wh	9913 Wh	20070 Wh
MAE	4505 Wh	5086 Wh	13338 Wh
R2 1	0,888	0,807	0,744
R2 2	0,863	0,881	0,511

This time the Decision Tree model is improved while the other two models have slightly reduced accuracy. Still the improvement is considerable so the variable was included.

Using Radiation for the current hour, 1, 2 and 3 hours before:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	10225 Wh	9770 Wh	20059 Wh	9378 Wh	7847 Wh
MAE	4155 Wh	4985 Wh	13331 Wh	5410 Wh	3753 Wh
R2 1	0,891	0,814	0,747	0,838	0,912
R2 2	0,873	0,884	0,511	-0,238	-0,238

At this point the ensembles using a combination of the original models were also used, initially using equal weights. Using the radiation data from 3 hours ago also appears to improve

accuracy. The ensemble of the Decision Tree model and the SVR model greatly outperformed the rest of the models.

Using Radiation for the current hour, 1, 2, 3 and 4 hours before:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	9778 Wh	9619 Wh	20226 Wh	9040 Wh	6497 Wh
MAE	4293 Wh	4946 Wh	13642 Wh	5457 Wh	3253 Wh
R2 1	0,919	0,822	0,751	0,853	0,948
R2 2	0,884	0,888	0,503	-0,224	-0,238

Adding more of the previous hours' radiation as features further improves all models' accuracy except for Linear Regression.

Using Radiation for the current hour, 1, 2, 3, 4 and 5 hours before:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	8849 Wh	9640 Wh	20762 Wh	9008 Wh	6063 Wh
MAE	3920 Wh	4999 Wh	14441 Wh	5725 Wh	3144 Wh
R2 1	0,932	0,822	0,746	0,856	0,954
R2 2	0,905	0,887	0,476	-0,207	-0,238

Using Radiation for the current hour, as well as 1 to 6 hours before:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	9055 Wh	9641 Wh	21331 Wh	9197 Wh	6844 Wh
MAE	4021 Wh	4999 Wh	15110 Wh	6040 Wh	3428 Wh
R2 1	0,924	0,822	0,736	0,851	0,939
R2 2	0,9	0,887	0,447	-0,193	-0,238

Using the radiation value for 6 hours before as an input feature decreased the accuracy, unlike the hours after it. As such, only the first 5 hours before the current time were used for further tests.

Using Radiation for the current hour, as well as 1 to 4 hours before, as well as Hour of the day as a categorical feature:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	8792 Wh	11444 Wh	Error	11712 Wh	9486 Wh
MAE	3975 Wh	5947 Wh	Error	6429 Wh	4793 Wh
R2 1	0,929	0,873	0	0,870	0,914
R2 2	0,906	0,841	Error	Error	0,071

In this test, the Hour was added as a categorical feature, converted into a sparse matrix using OneHotEncoder. Linear regression displayed erroneous 12-digit numbers for results which were deemed not to have any meaning. The R2 2 value of the Linear & Decision Tree ensemble model had the same error. Compared to the model using the same data except the Hour, this model shows improved accuracy for the decision tree model but greatly reduced accuracy for the Ensemble, which greatly outperformed the Decision Tree in the previous test.

Using Radiation for the current hour, as well as 1 to 4 hours before, plus Hour of the day and Month of the year as categorical features:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	8795 Wh	11444 Wh	Error	11712 Wh	9489 Wh
MAE	3992 Wh	5947 Wh	Error	6429 Wh	4801 Wh
R2 1	0,929	0,873	0	0,870	0,914
R2 2	0,906	0,841	Error	Error	0,071

Adding the Month as a categorical feature changes the results negligibly compared with the previous test.

Using Radiation for the current hour, as well as 1 to 5 hours before, plus Hour of the day and Month of the year as categorical features:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	8504 Wh	11443 Wh	Error	11710 Wh	9321 Wh
MAE	3745 Wh	5946 Wh	Error	6429 Wh	4722 Wh
R2 1	0,932	0,873	0	0,870	0,916
R2 2	0,912	0,841	Error	Error	0,071

Consistently with the previous test, the addition of the Hour and Month improve the Decision Tree model but decrease the accuracy of the ensembles. Adding the data from 5 hours before improves only the Decision Tree model by a considerable amount.

Season was also tested as a feature. However, only the Decision tree displayed any results at all when both the Season and the Hour were used as categorical features.

Using Radiation for the current hour, as well as 1 to 4 hours before, plus Hour and Season as categorical features, declaring Season using string data:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	8366 Wh	Error	Error	Error	Error
MAE	3810 Wh	Error	Error	Error	Error
R2 1	0,925	Error	Error	Error	Error
R2 2	0,915	Error	Error	Error	Error

Using Radiation for the current hour, as well as 1 to 5 hours before plus Hour and Season as categorical features, declaring Season using string data:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	9102 Wh	11443 Wh	Error	Error	Error
MAE	4087 Wh	5946 Wh	Error	Error	Error
R2 1	0,917	0,873	0	Error	Error
R2 2	0,899	0,841	Error	Error	Error

Adding the fifth hour, unlike in the previous tests, decreased the model's accuracy. For that reason, it was not used in the next test.

Using Radiation for the current hour, as well as 1 to 5 hours before, plus Hour and Season as categorical features, declaring Season as an integer:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	8599 Wh	Error	Error	Error	Error
MAE	3894 Wh	Error	Error	Error	Error
R2 1	0,921	Error	Error	Error	Error
R2 2	0,910	Error	Error	Error	Error

The results also had decreased accuracy compared to when Season was declared as a string.

These tests were performed again, testing different hyperparameters for the models. The decision tree was set to have a max depth of 22 and min_leaf of 20, leaving other parameters the same. SVR was set with C=30. Using radiation at the current time and up to 2 hours before as input features, the results were the following:

	Decision Tree Method	**SVR**	**Linear Regression**	**SVR & Linear**	**Dec. Tree & SVR**
RMSE	9159 Wh	9783 Wh	20066 Wh	9516 Wh	6827 Wh
MAE	3981 Wh	5007 Wh	13338 Wh	5465 Wh	3401 Wh
R2 1	0,925	0,815	0,744	0,832	0,94
R2 2	0,898	0,883	0,510	-0,244	-0,260

It is observed that increasing C to 30 in SVR increases accuracy, while reducing it to 10 in an individual test reduced it. The Decision Tree model also increased in accuracy using the new parameters when compared to the test that used the same input features.

Using the new hyperparameters and, radiation for the current hour, as well as 1 to 3 hours before as input features:

	Decision Tree Method	**SVR**	**Linear Regression**	**SVR & Linear**	**Dec. Tree & SVR**
RMSE	8662 Wh	9602 Wh	20059 Wh	9303 Wh	6683 Wh
MAE	3856 Wh	4882 Wh	13331 Wh	5361 Wh	3359 Wh
R2 1	0,930	0,824	0,747	0,842	0,942
R2 2	0,909	0,888	0,511	-0,238	-0,260

The SVR and Ensemble models can be compared in the following graphs, with orange representing the real values and blue representing the predicted values.

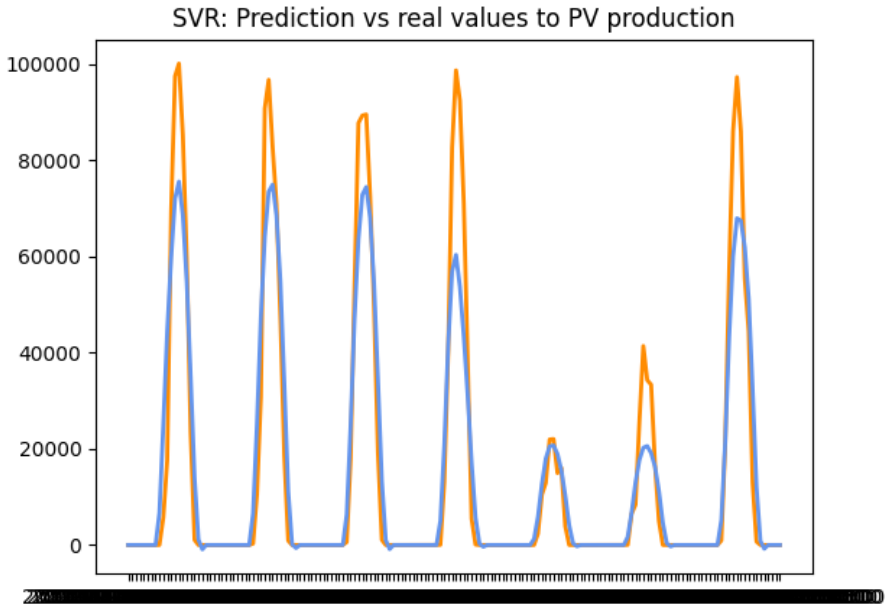

Figure 24 SVR Prediction vs Real values using radiation for the current hour, as well as 1 to 3 hours before as input features

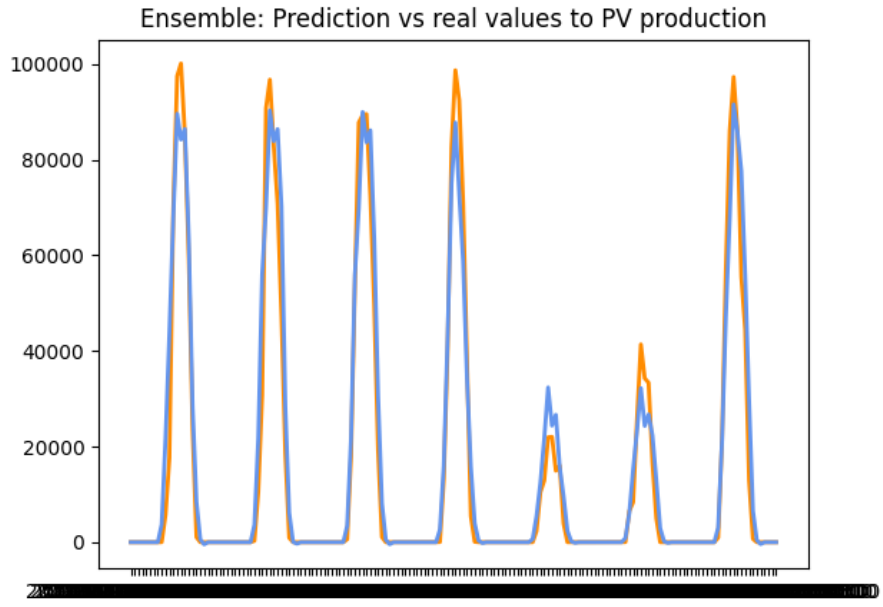

Figure 25 Decision Tree and SVR Ensemble Prediction vs Real values using radiation for the current hour, as well as 1 to 3 hours before as input features

Accuracy is also increased using this set of features.

Using radiation for the current hour and 1 to 4 hours before as input features:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	8072 Wh	9481 Wh	20226 Wh	8981 Wh	6788 Wh
MAE	3749 Wh	4864 Wh	13642 Wh	5423 Wh	3405 Wh
R2 1	0,936	0,83	0,751	0,856	0,938
R2 2	0,921	0,891	0,503	-0,224	-0,260

During this test, it was observed that while the individual models keep increasing in accuracy with the new parameters, the ensemble model of the Decision Tree and SVR, which outperforms the rest, actually reduces in accuracy, both compared to the test that used less features and the test that used the same features with different hyperparameters.

Using as input features: Radiation for the current hour and 1 to 3 hours before, Hour of the day as integer:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	9163 Wh	9602 Wh	20060 Wh	9303 Wh	6647 Wh
MAE	4178 Wh	4882 Wh	13331 Wh	5362 Wh	3324 Wh
R2 1	0,924	0,824	0,747	0,842	0,943
R2 2	0,898	0,888	0,511	-0,237	-0,260

Adding the hour of the day as an integer reduces the precision of the decision tree but marginally improves the ensemble.

Using as input features the Radiation for the current hour and 1 to 3 hours before, Hour and Month as integers:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	9098 Wh	9602 Wh	20031 Wh	9327 Wh	6941 Wh
MAE	3859 Wh	4882 Wh	13331 Wh	4859 Wh	3500 Wh
R2 1	0,922	0,824	0,748	0,841	0,937
R2 2	0,899	0,888	0,513	-0,192	-0,277

Inversely, adding the Month as an integer decreases the accuracy of the ensemble and improves the decision tree.

Using as input features the Radiation for the current hour, and 1 to 3 hours before and Month as categorical using a sparse matrix:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	8566 Wh	9583 Wh	20876 Wh	9244 Wh	6979 Wh
MAE	3898 Wh	4892 Wh	13331 Wh	5643 Wh	3626 Wh
R2 1	0,927	0,826	0,742	0,849	0,934
R2 2	0,911	0,888	0,471	-0,264	-0,265

Similarly to its use as an integer, Month marginally improves the singular models and reduces the accuracy of the ensemble.

Using as input features the Radiation for the current hour, and 1 to 3 hours before, and Hour as categorical:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	8449 Wh	11800 Wh	Error	11885 Wh	9422 Wh
MAE	3811 Wh	6079 Wh	Error	6497 Wh	4748 Wh
R2 1	0,933	0,868	0	0,867	0,915
R2 2	0,913	0,831	Error	Error	0,230

Inversely from when it was used as an integer, Hour marginally improves the decision tree and reduces the performance of the rest of the models. The same occurred with the previous hyperparameters.

Using as input features the Radiation for the current hour and 1 to 3 hours before, and Season as categorical:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	7442 Wh	9602 Wh	20543 Wh	9287 Wh	7006 Wh
MAE	3535 Wh	4882 Wh	13329 Wh	5783 Wh	3550 Wh
R2 1	0,941	0,824	0,744	0,845	0,931
R2 2	0,933	0,888	0,487	-0,247	-0,277

Season affects the models in a way similar to the hour.

Using as input features the Radiation for the current hour and 1 to 3 hours before, month and season as categorical:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	8030 Wh	9582 Wh	20857 Wh	9244 Wh	7363 Wh
MAE	3813 Wh	4892 Wh	13330 Wh	5643 Wh	3703 Wh
R2 1	0,931	0,826	0,742	0,849	0,924
R2 2	0,922	0,888	0,471	-0,263	-0,282

Using both month and season as inputs lowers the performance of all models, possibly because season is derived from month data.

Using as input features the Radiation for the current hour and 1 to 3 hours before, hour and season as categorical:

	Decision Tree Method	SVR	Linear Regression	SVR & Linear	Dec. Tree & SVR
RMSE	6667 Wh	N/A	N/A	N/A	N/A
MAE	3299 Wh	N/A	N/A	N/A	N/A
R2 1	0,952	N/A	N/A	N/A	N/A
R2 2	0,946	N/A	N/A	N/A	N/A

While during this test only the decision tree model displayed any results at all, it had the best performance for this particular model, which was also close to the best performance of all tests.

The optimal input features from the results of the manual tests were the following:

Model	Input Features
Decision Tree	Solar Radiation from current up to 3 hours before (hourly), hour of the day and season as categorical
SVR	Solar Radiation from current up to 4 hours before (hourly)
Linear Regression	Solar Radiation from current and previous hour
SVR & Linear	Solar Radiation from current up to 4 hours before (hourly)
SVR & Decision Tree	Solar Radiation from current up to 5 hours before (hourly)

It was observed from the tests that using different hyperparameters did not significantly affect the way the models reacted to different combinations of input features in most cases. There were exceptions however. The combination that gave the best performance using manual tests was found to be: 1) Radiation values from current time to 5 hours before, for the ensemble

model, and 2) Radiation value from current time to 3 hours before together with hour and season as categorical values, for the Decision tree.

5.3 Optimizing the forecasting models

GridSearchCV was used to find the optimal hyperparameters for each model. The model using the Decision tree was tested using the optimal input features set as described above. Different parameter ranges were tested in several tests instead of just one, in order to cut back on computational requirements.

5.3.1 Optimizing the Decision Tree model

The parameters tested were defined as following:

```
'random_state':[0],
'criterion':['squared_error'],
'splitter':['best','random'],
'max_depth': [ 50, 70, 75, 80, 100 ],
'min_samples_split': [ 8, 11, 12, 13, 16 ],
'min_samples_leaf': [20, 40],
```

GridSearchCV returned the optimal hyperparameters to be:

```
{'criterion': 'squared_error', 'max_depth': 50, 'min_samples_leaf': 20,
'min_samples_split': 8, 'random_state': 0, 'splitter': 'random'}
```

These parameters gave an R-squared score equal to 0,888. More hyperparameters were tested, specifically:

```
'random_state':[0],
'criterion':['squared_error', 'friedman_mse',
'absolute_error', 'poisson'],
'splitter':['best', 'random'],
'max_depth': [ 5, 10, 20, 22, 40, 50 ],
'min_samples_split': [ 2, 4, 8 ],
'min_samples_leaf': [1, 10, 20, 40],
```

Returned the best parameters as:

```
{'criterion': 'friedman_mse', 'max_depth': 40, 'min_samples_leaf': 20,
'min_samples_split': 2, 'random_state': 0, 'splitter': 'random'}
```

R-squared: 0,889

Testing the values:

```
'random_state':[0],
'criterion':['squared_error', 'friedman_mse'],
'splitter':['best', 'random'],
'max_depth': [ 40, 50, 100 ],
'min_samples_split': [ 2, 4, 8, 13 ],
'min_samples_leaf': [18, 20, 22]
```

Returned:

```
{'criterion': 'friedman_mse', 'max_depth': 40, 'min_samples_leaf': 20,
'min_samples_split': 2, 'random_state': 0, 'splitter': 'random'}
```

R-squared: 0,889

Testing the values:

```
'random_state':[0],
'criterion':['squared_error', 'friedman_mse'],
'splitter':['best', 'random'],
'max_depth': [ 38, 40, 42 ],
'min_samples_split': [ 2, 4, 6, 8 ],
'min_samples_leaf': [19, 20, 21],
```

Returned:

```
{'criterion': 'squared_error', 'max_depth': 38, 'min_samples_leaf': 21,
'min_samples_split': 2, 'random_state': 0, 'splitter': 'random'}
```

R-squared: 0,890

Testing the values:

```
'random_state':[0],
'criterion':['squared_error', 'friedman_mse'],
'splitter':['best', 'random'],
'max_depth': [
9,10,11,12,13,14,15,16,17,18,19,20,21,22, 24, 26, 28, 30, 32, 34, 35,
36, 37 ,38, 39 ],
'min_samples_split': [ 2 ],
'min_samples_leaf': [21]
```

Returned the final values of:

```
{'criterion': 'squared_error', 'max_depth': 15, 'min_samples_leaf': 21,
'min_samples_split': 2, 'random_state': 0, 'splitter': 'random'}
```

R-squared: 0,890

The last test was concluded to determine the best parameters for the decision tree model, as the values tested were continuous. For example, for maximum depth, the values of 14, 15 and 16 were all tested and the algorithm specifically chose 15; it did not select the upper limit of the range selected or a value of which the neighboring values were not tested. As such, for random state 0, this set of values were the optimal. It was observed that if the random state is not defined, the optimal values can vary.

5.3.2. Optimizing the SVR model

The best set of input features for SVR was found to be the Radiation values from current time up to 4 hours before. That set was used for the tests. In the case of SVR, the data had to be standardized, so in order to use GridSearchCV a pipeline integrating StandardScaler was used.

```
grid_svr =
Pipeline(steps=[('scaler',preprocessing.StandardScaler()),('svr',
svm.SVR())])
```

Testing the values:

```
              'svr__C': [1, 10, 30, 100],
              'svr__epsilon': [1, 0.2, 0.1],
              'svr__gamma':['scale', 'auto'],
              'svr__kernel': ['linear','poly', 'rbf', 'sigmoid',
'precomputed']
```

Returned the optimal values:

```
{'svr__C': 100, 'svr__epsilon': 1, 'svr__gamma': 'auto', 'svr__kernel':
'rbf'}
```

R-squared: 0,825

Since the optimal C and epsilon were the maximum values, a range of larger values were tested in the following test.

Testing the values:

```
              'svr__C': [90, 100,500,1000,10000,100000],
              'svr__epsilon': [2, 1, 0.2, 0.1],
              'svr__gamma':['auto'],
              'svr__kernel': ['linear', 'rbf']
```

Returned the optimal values:

```
{'svr__C': 100000, 'svr__epsilon': 0.2, 'svr__gamma': 'auto',
'svr__kernel': 'rbf'}
```

R-squared: 0,877

Testing the values:

```
              'svr__C': [90000,100000, 200000,1000000,2000000],
              'svr__epsilon': [1,0.8,0.4, 0.2],
              'svr__gamma':['auto'],
              'svr__kernel': ['linear', 'rbf']
```

Returned the optimal values:

```
{'svr__C': 200000, 'svr__epsilon': 0.2, 'svr__gamma': 'auto',
'svr__kernel': 'rbf'}
```
R-squared: 0,878

Notably, the computation for the last test lasted about 24 hours, unlike the previous tests.

Testing the values:

```
'svr__C': [180000, 200000,220000],
'svr__epsilon': [0.25, 0.2, 0.15],
'svr__gamma':['auto'],
'svr__kernel': ['rbf']
```

Returned the optimal values:

```
{'svr__C': 220000, 'svr__epsilon': 0.15, 'svr__gamma': 'auto',
'svr__kernel': 'rbf'}
```
R-squared: 0,878

Testing the values:

```
'svr__C': [210000,220000,230000,240000],
'svr__epsilon': [0.2],
'svr__gamma':['auto'],
'svr__kernel': ['rbf']
```

Returned the optimal values:

```
{'svr__C': 220000, 'svr__epsilon': 0.2, 'svr__gamma': 'auto',
'svr__kernel': 'rbf'}
```
R-squared: 0,878

Since the values tested were not at the end of the range, it was concluded that they are the optimal for this model.

5.3.3 Optimizing the Linear Regression model

The Linear regression model does not have any parameters to optimize in GridSearchCV. It also performed much worse compared to the other models. Its optimal input set was found to be the current and previous hour solar radiation values.

5.3.4 Optimizing the ensemble models

The optimization of the ensemble models included using the optimized models of the previous tests, trying to determine the optimal weights for each.

For the SVR-Linear regression model, tests were made using the optimal input features for SVR.

Testing the weights:

```
'weights': [(0.5, 0.5),(0.25, 0.75),(0.75, 0.25)]
```

Returned the results:

```
{'weights': (0.75, 0.25)}
```
R-squared: 0,874

Testing the weights:

```
'weights': [(0.625, 0.375),(0.75, 0.25),(0.875, 0.125)]
```

Returned the results:

```
{'weights': (0.875, 0.125)}
```
R-squared: 0,878

Testing the weights:

```
'weights': [(0.875, 0.125),(0.9, 0.1),(1, 0)]
```

Returned the results:

```
{'weights': (1, 0)}
```
R-squared: 0,878

It is concluded that the SVR-Linear Ensemble is inferior to the plain SVR model, because the Linear Regression model is much more inaccurate. The optimal setting is to ignore the Linear Regression model and give 100% weight on SVR.

The SVR-Decision Tree Ensemble was then tested, using the optimal input set for SVR.

Testing the weights:

```
'weights': [(0.5, 0.5),(0.25, 0.75),(0.75, 0.25)]
```

Returned the results:

```
{'weights': (0.75, 0.25)},
```
with the greater weight on the SVR model
R-squared: 0,880

Testing the weights:

```
'weights': [(0.625, 0.375),(0.75, 0.25),(0.875, 0.125),(1, 0)]
```

Returned the results:

```
{'weights': (0.75, 0.25)}
```

R-squared: 0,880

The optimal setting for the SVR-Decision Tree model was 0,75 weight on SVR and 0,25 on the Decision Tree.

5.4 Tests with a smaller train set

After determining the optimal parameters both manually and using GridSearchCV, tests were performed to assess the accuracy of the model when the train data set was comprised of the first 50% of the available data and the rest were used as the test set. The sets used were three of the manual tests for each model that gave the best results, plus the optimal set derived from GridSearchCV.

The sets are presented in descending order based on their accuracy. For the Decision Tree model, the sets used and the corresponding results were the following:

Using max_depth=22, min_samples=4, min_leaf=20,

1) Input set: Current and up to 3 hours before radiation values, hour and season as categorical:

RMSE	15054
MAE	7151
R2 1	0,879
R2 2	0,879

Using the previous data set, the model had less than half the RMSE (6666) and 0,946 R-squared. This is within expectations as this model tries to predict a much larger set of data and knows much less.

2) Input set: Current and up to 3 hours before radiation values and season as categorical:

RMSE	16209
MAE	7947
R2 1	0,858
R2 2	0,86

3) Input set: Current and up to 3 hours before radiation values, hour as integer and season as categorical:

RMSE	14616
MAE	6901
R2 1	0,886
R2 2	0,886

4) Using the GridSearchCV determined hyperparameters of:

```
{'criterion': 'squared_error', 'max_depth': 15, 'min_samples_leaf': 21,
'min_samples_split': 2, 'random_state': 0, 'splitter': 'random'}
```

Input set: Current and up to 3 hours before radiation values, hour and season as categorical:

RMSE	15224
MAE	7284
R2 1	0,873
R2 2	0,876

It is observed that the accuracy reduction varies compared to the models using the larger input dataset. Model 3, which in the previous case was the least accurate of the 3, is now the most accurate model. The model determined by GridSearchCV is comparatively more accurate than before, which is expected, as it was meant to optimize accuracy in reference to the input set, unlike the manual models, which were defined by the accuracy with which they predicted the last week of the dataset only.

For the SVR model:

The following hyperparameters were used in the manual tests:

```
{kernel='linear',C=30, epsilon=0.2,gamma='auto'}
```

1) Input set: Radiation from current to 4 hours before as input:

RMSE	20180
MAE	10736
R2 1	0,712
R2 2	0,782

2) Input set: Current and up to 3 hours before radiation values, month and season as categorical:

RMSE	20126
MAE	10741
R2 1	0,715
R2 2	0,783

3) Input set: Current and up to 3 hours before radiation values and month as categorical:

RMSE	20126
MAE	10741
R2 1	0,715
R2 2	0,783

4) Using the GridSearchCV determined hyperparameters of:

```
{'svr__C': 220000, 'svr__epsilon': 0.2, 'svr__gamma': 'auto',
'svr__kernel': 'rbf'}
```

And using the input feature set of 1), which had been determined to be the most accurate before, the results were:

RMSE	15160
MAE	7325
R2 1	0,870
R2 2	0,877

In this case, the GridSearchCV-determined model greatly outperformed all manually defined models, as the C parameter was selected too conservatively in the latter.

For the Linear Regression:

1) Input set: Current and previous hour radiation:

RMSE	20212
MAE	12138
R2 1	0,755
R2 2	0,782

2) Input set: Current and up to 3 hours before radiation values, month and hour as integers:

RMSE	20264
MAE	12072
R2 1	0,756
R2 2	0,780

3) Input set: Current and up to 3 hours before radiation values:

RMSE	20312
MAE	12120
R2 1	0,754
R2 2	0,779

The accuracy of the Linear Regression model is almost unaffected by the reduction of the Input data size. Because of the comparative decrease in accuracy of the other models, this model's results appear more useful than before, although it is still less accurate than the rest.

For the SVR-Linear Regression Ensemble, GridSearchCV determined that the best setting is to use SVR alone. Weights were equal for each model (0,5 and 0,5).

1) Using the SVR hyperparameters:

```
{kernel='linear',C=30, epsilon=0.2,gamma='auto'}
```

Input set: Radiation from current to 4 hours before:

RMSE	19915
MAE	11151
R2 1	0,734
R2 2	-0,363

2) The C parameter is set to 20, with an input set of: Radiation from current to 5 hours before as input:

RMSE	19742
MAE	11307
R2 1	0,736
R2 2	-0,35

3) The C parameter is set to 20, with an input set of: Radiation from current to 4 hours before as input:

RMSE	19952
MAE	11190
R2 1	0,73
R2 2	-0,363

4) The SVR model using the optimal parameters gave the following results:

RMSE	15160
MAE	7325
R2 1	0,870
R2 2	0,877

The Ensemble's accuracy has decreased proportionally in the same way as the SVR model.

For the Decision Tree-SVR Ensemble:

1) Using the hyperparameters:

```
svm.SVR(kernel='linear',C=20, epsilon=0.2,gamma='auto')
```

```
DecisionTreeRegressor(criterion='squared_error', random_state = 0,
max_depth = 15, min_samples_split=4,
min_samples_leaf=1,splitter='random')
```

Weights were equal for each model (0,5 and 0,5).
Input set: Radiation from current to 5 hours before:

RMSE	17100
MAE	8866
R2 1	0,813
R2 2	-0,383

2) Using the same hyperparameters as in 1).

Input set: Radiation from current to 4 hours before:

RMSE	17518
MAE	9025
R2 1	0,803
R2 2	-0,388

3) Using the hyperparameters:

```
svm.SVR(kernel='linear',C=30, epsilon=0.2,gamma='auto')
```

```
DecisionTreeRegressor(criterion='squared_error', random_state = 0,
max_depth = 22, min_samples_split=4,
min_samples_leaf=20,splitter='random')
```

Weights were also equal for each model (0,5 and 0,5).

Input set: Current and up to 3 hours before radiation values, and hour as integer:

RMSE	16075
MAE	8357
R2 1	0,833
R2 2	-0,389

4) Using the GridSearchCV determined hyperparameters of:

```
{'svr__C': 220000, 'svr__epsilon': 0.2, 'svr__gamma': 'auto',
'svr__kernel': 'rbf'}
{'criterion': 'squared_error', 'max_depth': 15, 'min_samples_leaf': 21,
'min_samples_split': 2, 'random_state': 0, 'splitter': 'random'}
```

Weights were set to 0,75 for SVR and 0,25 for the Decision Tree.

Input set: Radiation from current to 4 hours before:

RMSE	15107
MAE	7403
R2 1	0,87
R2 2	-0,398

The GridSearch-defined Ensemble model is comparatively more accurate than the manually optimized models, unlike when the test set was smaller. However, it is observed that the single Decision Tree model outperforms the Ensemble when the test set is bigger.

5.5 Results

The optimized models were found to be the following:

1) Decision Tree:

Input Features Solar Radiation from current up to 3 hours before (hourly), hour of the day and season as categorical

Hyperparameters **Large Input Dataset**

```
{'criterion': 'squared_error', 'max_depth': 22,
'min_samples_leaf': 20, 'min_samples_split': 4,
'random_state': 0, 'splitter': 'random'}
```

Small Input Dataset

```
{'criterion': 'squared_error', 'max_depth': 15,
'min_samples_leaf': 21, 'min_samples_split': 2,
'random_state': 0, 'splitter': 'random'}
```

Results

	Large Input Dataset		**Small Input Dataset**
RMSE	6666 Wh	**RMSE**	14616 Wh
MAE	3299 Wh	**MAE**	6901 Wh
R2 1	0,952	**R2 1**	0,886
R2 2	0,946	**R2 2**	0,886

2) SVM:

Input Features Solar Radiation from current up to 4 hours before (hourly)

Hyperparameters **Large Input Dataset**

```
{'svr__C': 30, 'svr__epsilon': 0.2, 'svr__gamma':
'auto', 'svr__kernel': 'linear'}
```

Small Input Dataset

```
{'svr__C': 220000, 'svr__epsilon': 0.2, 'svr__gamma':
'auto', 'svr__kernel': 'rbf'}
```

Results

Large Input Dataset		Small Input Dataset	
RMSE	9481 Wh	**RMSE**	15160 Wh
MAE	4864 Wh	**MAE**	7325 Wh
R2 1	0,830	**R2 1**	0,870
R2 2	0,890	**R2 2**	0,877

3) Linear Regression:

Input Features Solar Radiation from current and previous hour

Hyperparameters N/A

Results

Large Input Dataset		Small Input Dataset	
RMSE	19925 Wh	**RMSE**	20212 Wh
MAE	13331 Wh	**MAE**	12138 Wh
R2 1	0,749	**R2 1**	0,755
R2 2	0,517	**R2 2**	0,782

4) SVR & Linear:

Input Features Solar Radiation from current up to 4 hours before (hourly)

Hyperparameters **Large Input Dataset:**

```
{'svr__C': 30, 'svr__epsilon': 0.2, 'svr__gamma':
'auto', 'svr__kernel': 'linear'}
{'weight': 0.5, 'weight': 0.5}
```

Small Input Dataset:

```
{'svr__C': 220000, 'svr__epsilon': 0.2, 'svr__gamma':
'auto', 'svr__kernel': 'rbf'}
{'weight': 1.00, 'weight': 0.00}
```

Results

Large Input Dataset		Small Input Dataset	
RMSE	8981 Wh	**RMSE**	15160 Wh
MAE	5423 Wh	**MAE**	7325 Wh
R2 1	0,856	**R2 1**	0,870
R2 2	-0,224	**R2 2**	0,877

5) SVR & Decision Tree:

Input Features Solar Radiation from current up to 5 hours before (hourly)

Hyperparameters **Large Input Dataset:**

```
{'criterion': 'squared_error', 'max_depth': 15,
'min_samples_leaf': 1, 'min_samples_split': 4,
'random_state': 0, 'splitter': 'random'}
{'svr__C': 20, 'svr__epsilon': 0.2, 'svr__gamma':
'auto', 'svr__kernel': 'linear'}
{'weight': 0.5, 'weight': 0.5}
```

Small Input Dataset:

```
{'criterion': 'squared_error', 'max_depth': 15,
'min_samples_leaf': 21, 'min_samples_split': 2,
'random_state': 0, 'splitter': 'random'}
{'svr__C': 220000, 'svr__epsilon': 0.2, 'svr__gamma':
'auto', 'svr__kernel': 'rbf'}
{'weight': 0.75, 'weight': 0.25}
```

Results

Large Input Dataset		Small Input Dataset	
RMSE	6063 Wh	**RMSE**	15107 Wh
MAE	3144 Wh	**MAE**	7403 Wh
R2 1	0,954	**R2 1**	0,870
R2 2	-0,238	**R2 2**	-0,398

The models' relative performance can be compared in the following table:

Large Input Dataset

	Decision Tree Method	SVR	Linear Regression	Dec. Tree & Linear	Dec. Tree & SVR
RMSE	6666 Wh	9481 Wh	19925 Wh	8981	6063 Wh
MAE	3299 Wh	4864 Wh	13331 Wh	5423 Wh	3144 Wh
R2 1	0,952	0,83	0,749	0,856	0,954
R2 2	0,946	0,89	0,517	-0,224	-0,238

Small Input Dataset

RMSE	14616 Wh	15160 Wh	20212 Wh	15160 Wh	15107 Wh
MAE	6901 Wh	7325 Wh	12138 Wh	7325 Wh	7403 Wh
R2 1	0,886	0,87	0,755	0,87	0,87
R2 2	0,886	0,877	0,782	0,877	-0,398

The average production value in the dataset was 58973 Wh. When making forecasts for the final week only, the average value of the test data was 38953 Wh. As such, nRMSE is calculated as follows:

	Decision Tree Method	SVR	Linear Regression	Dec. Tree & Linear	Dec. Tree & SVR
	Large Input Dataset				
nRMSE	17,11%	24,34%	51,15%	23,06%	15,56%
	Small Input Dataset				
nRMSE	24,78%	25,71%	34,27%	25,71%	25,62%

Chapter 6: Conclusions and Future work

6.1 Conclusions

The weather data aside from Solar Radiation were not found to be useful for making forecasts for this particular case study. The use Humidity, temperature and wind speed as input reduced the accuracy of the models. The literature confirms that the utility of different weather parameters varies on each individual case, depending on various circumstances. The data that contributed to the forecasting were the radiation values of the previous hours and the values of hour and month.

For this particular dataset, Linear Regression was clearly inferior to the rest of the models, indicating that the problem that was being examined was non-linear. SVR tended to underestimate the data in size, predicting values that were much smaller than the real value. The ensemble using both Linear Regression and SVR slightly improved the accuracy compared to standalone SVR when making predictions using the larger dataset, but SVR outperformed it when the smaller dataset was used.

The Decision Tree model has been observed to be the most accurate when making predictions using a smaller dataset, while the SVR-Decision Tree Ensemble was the most accurate when making forecasts using the larger dataset. When making use of the smaller dataset, the variation of accuracy between the different models becomes smaller. The Linear Regression model has nearly the same accuracy for both dataset sizes, which is within expectations as it is a linear model.

The optimized Decision Tree had a MAE of 6901 Wh when 50% of the dataset was used as input, indicating a 11,7 % normalized Mean Average Error rate and a 24,78% nRMSE. When using the larger dataset, the MAE of the SVR-Decision Tree Ensemble was 3144, giving a normalized Mean Average Error rate of 8,3% and an nRMSE of 15,56%.

The performance of the models can be deemed to be satisfactory compared to the results found in literature, as results up to 41,20% nRMSE were published.

This thesis provides an insight on the efficiency of Decision Tree and Linear Regression as forecasting models compared to the frequently used SVR. Such an insight was not commonly found in literature concerning PV forecasting. The results are noteworthy as in this case study the Decision Tree and the Decision Tree ensemble outperformed the standalone SVR model, providing the notion that Decision Tree may be useful in PV forecasting.

6.2 Future Work

For the particular case study, more extensive tests can be made, utilizing an even smaller dataset, in order to test the models' reliability when less data are available. Forecasts using ensemble models utilizing different input parameters than the SVR-optimized may also be made to compare their performance.

Different models can also be useful in further testing. The dataset can produce accurate forecasts using different models than the ones selected. This particularly includes tests with Deep Learning models, which are the most popular group of methods in literature for PV forecasts.

Different datasets can also be used to test the models to assess how they perform in other case studies.

The software developed can also be deployed either as a web application or offline software. An interface can be developed to provide access to the functions of the software to someone who is not familiar with Python.

Appendix A – Code used for statistics analysis and machine learning

```
import pandas as pd
import datetime
import time
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import statistics as stat
import xlsxwriter
import math
import numpy as np
from matplotlib import interactive
interactive(True)
import statsmodels as stm
import statsmodels.tsa.seasonal
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score,
mean_absolute_error
from sklearn import svm
from sklearn import preprocessing
from sklearn.pipeline import make_pipeline
from sklearn.pipeline import Pipeline
import sklearn.linear_model
import os
import pickle
from sklearn.ensemble import VotingRegressor
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import GridSearchCV

try: # For offline execution
    data =
pd.read_excel(r'C:\python\CorrectedDataNew.xlsx',sheet_name="Data")
    buildingdata = data.values.tolist()
except: # For Google Colab
    data = pd.read_excel(r'/content/drive/MyDrive/Colab
Notebooks/CorrectedDataNew.xlsx',sheet_name="Data")
    buildingdata = data.values.tolist()

#Fix outliers
def fixoutliers(buildingdata):
    bq99 = []
    for i in range(len(buildingdata)):
        bq99.append(buildingdata[i][6])

    q99 = np.quantile(bq99,0.99)
    for i in range(len(buildingdata)):
        if buildingdata[i][6] > q99:
            buildingdata[i][6] = q99
    return buildingdata

def datetotimestamp(buildingdata):# Convert date columns into timestamp
    btime = [] # List of timestamps
    btimestring = ""
    for i in range(len(buildingdata)):
```

```
        #btimestring =(str(buildingdata[i][2]) + "/" +
str(buildingdata[i][1]) + "/" + str(buildingdata[i][0]) + " " +
str(buildingdata[i][3]).split("+",1)[0] )
        btimestring =(str(buildingdata[i][2]) + "/" +
str(buildingdata[i][1]) + "/" + str(buildingdata[i][0]) + " " +
str(buildingdata[i][3]).split("+",1)[0] + "+0" +
str(buildingdata[i][3]).split("+",1)[1] + ":00" )
        btime.append(datetime.datetime.strptime(btimestring,"%d/%m/%Y
%H:%M:%S%z").timetuple())
    return btime

def wdatetotimestamp(buildingdata):# Convert date column into timestamp
    btime = [] # List of timestamps
    btimestring = ""
    for i in range(len(buildingdata)):
        btimestring = buildingdata[i][1] + "+01:00"
        btime.append(datetime.datetime.strptime(btimestring,"%Y-%m-%d
%H:%M:%S%z").timetuple())
    return btime

def testmissingdata(bhour): #Print all values which do not follow
hourly sequence - Should only print DST changes in March if all is OK
    bhournew = []
    for i in range(1,len(bhour)):
        if datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) -
datetime.datetime.fromtimestamp(time.mktime(bhour[i-1][0])) !=
datetime.timedelta(hours=1):
            print(datetime.datetime.fromtimestamp(time.mktime(bhour[i-
1][0])))

print(datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])))


def bplot(bhour,title="empty"): #Plot the data
    bhourly1 = []
    for i in range(len(bhour)):
        bhourly1.append(time.mktime(bhour[i][0])) #conversion to epoch
time

    plotpvx = []
    plotpvy = []
    for i in range(len(bhour)):

plotpvx.append(mdates.date2num(datetime.datetime.utcfromtimestamp(bhour
ly1[i])))# Convert timestamp to a format matplotlib can handle
        plotpvy.append(bhour[i][1])
    plt.title(title)
    plt.plot(plotpvy)

def hourlymeans(bhourly, season='Yearly'): # Calculate hourly means
    hmeans = []
    hcount = []
    hourly = []

    for j in range(24):
        hmeans.append(0)
        hcount.append(0)
```

```
        for i in range(len(bhourly)):
            if season == "Yearly":
                if bhourly[i][0].tm_hour==j:
                    hmeans[j] += bhourly[i][1]
                    hcount[j] += 1
            elif season == "Summer":
                if bhourly[i][0].tm_hour==j and (bhourly[i][0].tm_mon >
3 and bhourly[i][0].tm_mon < 10):
                    hmeans[j] += bhourly[i][1]
                    hcount[j] += 1
            elif season == "Winter":
                if bhourly[i][0].tm_hour==j and (bhourly[i][0].tm_mon
<= 3 or bhourly[i][0].tm_mon >= 10):
                    hmeans[j] += bhourly[i][1]
                    hcount[j] += 1
            else:
                print('Error: Season not defined')
    for j in range(24):
        if hcount[j] != 0:
            hmeans[j] = hmeans[j]/hcount[j] # Divide the sum of
elements by the number of elements in each
        else:
            hmeans[j] = 0
    return hmeans

def standarddeviation(bhourly, season='Yearly'): # Calculate hourly
standard deviation
    listbyhour = []
    std = []
    for j in range(24):
        listbyhour.append([])
        for i in range(len(bhourly)):
            if season == "Yearly":
                if bhourly[i][0].tm_hour==j:
                    listbyhour[j].append(bhourly[i][1])
            elif season == "Summer":
                if bhourly[i][0].tm_hour==j and (bhourly[i][0].tm_mon >
3 and bhourly[i][0].tm_mon < 10):
                    listbyhour[j].append(bhourly[i][1])
            elif season == "Winter":
                if bhourly[i][0].tm_hour==j and (bhourly[i][0].tm_mon
<= 3 or bhourly[i][0].tm_mon >= 10):
                    listbyhour[j].append(bhourly[i][1])
            else:
                print('Error: Season not defined')

    for j in range(24):
        if len(listbyhour[j]) > 1:
            std.append(stat.stdev(listbyhour[j]))
        else:
            std.append(0)

    return std
```

```
def display_ml_error_indicators(test_sety,predicted_y):
    print("RMSE", format(mean_squared_error(predicted_y, test_sety,
squared=False), ".0f"))
    print("MAE", format(mean_absolute_error(predicted_y, test_sety),
".0f"))
    print("r_square", format(r2_score(predicted_y, test_sety), ".3f"))

#Trim production values over 99% of the rest of the data - not used
#buildingdata = fixoutliers(buildingdata)

pvhourly = []
weatherdata = []
btime = wdatetotimestamp(buildingdata)

for i in range(len(buildingdata)): # Create a list containting the
timestamp and production for each hour
    pvhourly.append([btime[i],buildingdata[i][6]])

# Hourly statistics for PV production
hourlystatslabel = 'Hourly Statistics'
pvhmeansyearly = hourlymeans(pvhourly) # Hourly mean production for the
whole year
pvhstdevyearly = standarddeviation(pvhourly) # Hourly deviation of
production for the whole year

pvhmeanssummer = hourlymeans(pvhourly, 'Summer') # Hourly mean
production for summer months
pvhstdevsummer = standarddeviation(pvhourly, 'Summer') # Hourly
deviation of production for summer months

pvhmeanswinter = hourlymeans(pvhourly, 'Winter') # Hourly mean
production for winter months
pvhstdevwinter = standarddeviation(pvhourly, 'Winter') # Hourly
deviation of production for winter months

# Prepeare legend table
blegend = (hourlystatslabel,
           'pvhmeansyearly: Hourly mean production for the whole year',
           'pvhstdevyearly: Hourly deviation of production for the
whole year',
           'pvhmeanssummer: Hourly mean production for summer months',
           'pvhstdevsummer: Hourly deviation of production for summer
months',
           'pvhmeanswinter: Hourly mean production for winter months',
           'pvhstdevwinter: Hourly deviation of production for winter
months')


datesinstring = []
day = []
month = []
season = []
year = []
hour = []
weekday = []
pvvalues = []
wvalues = []
```

```
wvalues1temp = []
wvalues2humid = []
wvalues3wind = []
wvalues4cloud = []
wvalues5rad = []
wvalues5rad_previoushour = []
wvalues5rad_prepreviousshour = []
wvalues5rad_3hoursbefore = []
wvalues5rad_4hoursbefore = []
wvalues5rad_5hoursbefore = []
wvalues5rad_6hoursbefore = []
for i in range(len(pvhourly)): # Check if dates match and prepare the
lists for each value
    try:
        wvalues2humid.append(buildingdata[i][8])
        wvalues3wind.append(buildingdata[i][9])
        wvalues4cloud.append(buildingdata[i][10])
        wvalues5rad.append(buildingdata[i][11])
        if i > 6:
            wvalues5rad_previoushour.append(wvalues5rad[i-1])
            wvalues5rad_prepreviousshour.append(wvalues5rad[i-2])
            wvalues5rad_3hoursbefore.append(wvalues5rad[i-3])
            wvalues5rad_4hoursbefore.append(wvalues5rad[i-4])
            wvalues5rad_5hoursbefore.append(wvalues5rad[i-5])
            wvalues5rad_6hoursbefore.append(wvalues5rad[i-6])
        else:
            wvalues5rad_previoushour.append(wvalues5rad[i])
            wvalues5rad_prepreviousshour.append(wvalues5rad[i])
            wvalues5rad_3hoursbefore.append(wvalues5rad[i])
            wvalues5rad_4hoursbefore.append(wvalues5rad[i])
            wvalues5rad_5hoursbefore.append(wvalues5rad[i])
            wvalues5rad_6hoursbefore.append(wvalues5rad[i])
        datesinstring.append(time.strftime('%Y-%m-%d %H:%M:%S',
pvhourly[i][0]))
        day.append(time.strftime('%d', pvhourly[i][0]))
        month.append(time.strftime('%m', pvhourly[i][0]))

        currentmonth = int(time.strftime('%m', pvhourly[i][0]))
        # Season
        if 3 < currentmonth and currentmonth <= 6:
            season.append('Spring') #Spring
        elif 6 < currentmonth and currentmonth <= 9:
            season.append('Summer') #Summer
        elif 9 < currentmonth and currentmonth <= 12:
            season.append('Autumn') #Autumn
        elif 0 < currentmonth and currentmonth <= 3:
            season.append('Winter') #Winter
        else:
            print("error")

        year.append(time.strftime('%Y', pvhourly[i][0]))
        hour.append(time.strftime('%H', pvhourly[i][0]))
        weekday.append(time.strftime('%w', pvhourly[i][0]))
        pvvalues.append(pvhourly[i][1])

        wvalues1temp.append(buildingdata[i][7])
```

```
    except:
        print('error')

encmonth = OneHotEncoder(categories='auto',handle_unknown='ignore')
enchour = OneHotEncoder(categories='auto',handle_unknown='ignore')
encseason = OneHotEncoder(categories='auto',handle_unknown='ignore')

month_cat = encmonth.fit_transform(np.array(month).reshape(-1, 1))
month_cat = month_cat.toarray()
hour_cat = enchour.fit_transform(np.array(hour).reshape(-1, 1))
hour_cat = hour_cat.toarray()
season_cat = encseason.fit_transform(np.array(season).reshape(-1, 1))
season_cat = season_cat.toarray()

s1 = pd.Series(datesinstring, name='Date/Time')
s2 = pd.Series(day, name='Day')
s3 = pd.Series(month, name='Month')
s4 = pd.Series(year, name='Year')
s5 = pd.Series(hour, name='Hour')
s7 = pd.Series(pvvalues, name='pvvalues')
s8 = pd.Series(wvalues1temp, name='air temperature [B°C]')
s9 = pd.Series(wvalues2humid, name='relative humidity [%]')
s10 = pd.Series(wvalues3wind, name='wind speed[m/s]')
s11 = pd.Series(wvalues4cloud, name='cloudcover [%]')
s12 = pd.Series(wvalues5rad, name='global radiation [W/m^2]')
s12_1 = pd.Series(wvalues5rad_previoushour, name='rvalues2')
s12_2 = pd.Series(wvalues5rad_preprevioushour, name='rvalues3')
s12_3 = pd.Series(wvalues5rad_3hoursbefore, name='rvalues4')
s13 = pd.Series(season, name='Season')
#Different input sets of numerical features - the sets not used in the
current test are commented out
'''df = pd.DataFrame({

'rvalues2':wvalues5rad_previoushour,'rvalues3':wvalues5rad_prepreviousho
our,

'rvalues4':wvalues5rad_3hoursbefore,'h':hour,'globalradiation':wvalues5
rad
                    }
                )'''
df = pd.DataFrame({

'rvalues2':wvalues5rad_previoushour,'rvalues3':wvalues5rad_prepreviousho
our,

'rvalues4':wvalues5rad_3hoursbefore,'rvalues5':wvalues5rad_4hoursbefore
,'globalradiation':wvalues5rad
                    }
                )
'''df = pd.DataFrame({

'rvalues2':wvalues5rad_previoushour,'rvalues3':wvalues5rad_prepreviousho
our,

'rvalues4':wvalues5rad_3hoursbefore,'globalradiation':wvalues5rad
                    }
                )'''
```

```
'''df = pd.DataFrame({

'rvalues2':wvalues5rad_previoushour,'rvalues3':wvalues5rad_prepreviousho
our,

'rvalues4':wvalues5rad_3hoursbefore,'globalradiation':wvalues5rad,

'rvalues5':wvalues5rad_4hoursbefore,'rvalues6':wvalues5rad_5hoursbefore
                        }
                )'''

# Categorical input features - the variables not used in the current
test are commented out
#df[encmonth.categories_[0]]= month_cat # Use month as a categorical
feature
#df[enchour.categories_[0]]= hour_cat # Use hour as a categorical
feature
#df[encseason.categories_[0]]= season_cat # Use season as a categorical
feature

df2 = pd.DataFrame(pvvalues)
df_time = pd.DataFrame(datesinstring)

test_set_index = int(len(wvalues1temp)*0.5) # The size of the test set
- set to 50% of the dataset

train_setx = df.iloc[:len(wvalues1temp)-test_set_index,:]
test_setx = df.iloc[len(wvalues1temp)-test_set_index:,:]
train_sety = df2.iloc[:len(wvalues1temp)-test_set_index,:]
test_sety = df2.iloc[len(wvalues1temp)-test_set_index:,:]
train_time = df_time.iloc[:len(wvalues1temp)-test_set_index,:]
test_time = df_time.iloc[len(wvalues1temp)-test_set_index:,:]

train_setx= train_setx.to_numpy()
test_setx = test_setx.to_numpy()

#Plots for statistics - not used for machine learning tests
plt.figure(2)
bplot(pvhourly,'production') # Plot historical energy production
plt.show

# Plot autocorrelation

plt.figure(3)
plt.title("Autocorrelation of PV energy production")
plt.acorr(pvvalues,maxlags=10)


# Time Series Decomposition to identify trend and seasonal components
try:
    seasonalpv = stm.tsa.seasonal.seasonal_decompose(pvvalues,
period=24*30) # Monthly intervals
    statsmodels.tsa.seasonal.DecomposeResult.plot(seasonalpv,
observed=True, seasonal=True, trend=True, resid=True, weights=False)
except: # Normal syntax is not compatible with colab, an exception is
raised and I resolve it by writing the colab comatible syntax
```

```
    seasonalpv = stm.tsa.seasonal.seasonal_decompose(pvvalues,
freq=24*30) # Monthly intervals
    statsmodels.tsa.seasonal.DecomposeResult.plot(seasonalpv)

weather_regr = []

# Prepare the lists of the output data and timestamps for machine
learning
train_sety = []
test_sety = []
train_time = []
test_time = []
for i in range(len(wvalues1temp)):
    if i < len(wvalues1temp)-test_set_index:
        train_sety.append(pvvalues[i])
        train_time.append(datesinstring[i])
    else:
        test_sety.append(pvvalues[i])
        test_time.append(datesinstring[i])

sc_x = preprocessing.StandardScaler()
train_setx_norm = sc_x.fit_transform(train_setx) #Normalized input data
test_setx_norm = sc_x.fit_transform(test_setx)

# Decision tree Regressor - Predict values
#regr_1 = DecisionTreeRegressor(criterion='squared_error', random_state
= 0, max_depth = 22, min_samples_split=4, min_samples_leaf=20) # If
max_depth is too low it won't take into account all the input features
regr_1 = make_pipeline(preprocessing.StandardScaler(),
DecisionTreeRegressor(criterion='squared_error', random_state = 0,
max_depth = 15, min_samples_split=2,
min_samples_leaf=21,splitter='random'))

regr_1.fit(train_setx, train_sety)

y_1 = regr_1.predict(test_setx)

plt.figure()
plt.title("Decision Tree: Prediction vs real values to PV production")
plt.plot(test_time, test_sety, color="darkorange", label="real values",
linewidth=2)
plt.plot(test_time, y_1, color="cornflowerblue", label="max_depth=2",
linewidth=2)

print("Decision Tree Method:")
display_ml_error_indicators(test_sety,y_1)
rscore = regr_1.score(test_setx, test_sety)
print("R-squared:", format(rscore, ".3f")) # R-Squared

# SVM - Predict values

regr_2 = make_pipeline(preprocessing.StandardScaler(),
svm.SVR(kernel='rbf',C=220000, epsilon=0.2,gamma='auto'))
#regr_2 = make_pipeline(preprocessing.StandardScaler(),
svm.SVR(kernel='linear',C=30, epsilon=0.2,gamma='auto'))

regr_2.fit(train_setx, train_sety)
```

```
y_2 =  regr_2.predict(test_setx)

print("SVR:")
display_ml_error_indicators(test_sety,y_2)
rscore = regr_2.score(test_setx, test_sety)
print("R-squared:", format(rscore, ".3f")) # R-Squared

plt.figure()
plt.title("SVR: Prediction vs real values to PV production")
plt.plot(test_time, test_sety, color="darkorange", label="real values",
linewidth=2)
plt.plot(test_time, y_2, color="cornflowerblue", label="predicted",
linewidth=2)

# Linear Regression - Predict values

regr_3 = sklearn.linear_model.LinearRegression().fit(train_setx_norm,
train_sety)
y_3 =  regr_3.predict(test_setx_norm)


print("Linear Regression:")
display_ml_error_indicators(test_sety,y_3)
test_setx_norm = np.array(test_setx_norm, dtype=float) # Need to
explicitly convert to numeric or the score function gives a warning
test_sety = np.array(test_sety, dtype=float)
rscore = regr_3.score(test_setx_norm, test_sety)
print("R-squared:", format(rscore, ".3f")) # R-Squared


plt.figure()
plt.title("Linear Regression: Prediction vs real values to PV
production")
plt.plot(test_time, test_sety, color="darkorange", label="real values",
linewidth=2)
plt.plot(test_time, y_3, color="cornflowerblue", label="predicted",
linewidth=2)

# Ensemble of Linear regression and SVR
print("Ensemble of Linear regression and SVR:")
weights = [0.5, 0.5]
models = []
models.append(('r1',regr_2))
models.append(('r2',regr_3))
ensemble = VotingRegressor(estimators=models, weights=weights)
ensemble.fit(train_setx, train_sety)

y_4 = ensemble.predict(test_setx)
display_ml_error_indicators(test_sety,y_4)

print("R-squared:", format(ensemble.score(test_setx_norm, test_sety),
".3f"))

print("Ensemble of Decision tree and SVR:")
#weights = [0.5, 0.5]
weights = [0.75, 0.25]
models = []
```

```
models.append(('r1',regr_2))
models.append(('r2',regr_1))
ensemble = VotingRegressor(estimators=models, weights=weights)
ensemble.fit(train_setx, train_sety)

y_5 = ensemble.predict(test_setx)
display_ml_error_indicators(test_sety,y_5)

print("R-squared:", format(ensemble.score(test_setx_norm, test_sety),
".3f"))

plt.figure()
plt.title("Ensemble: Prediction vs real values to PV production")
plt.plot(test_time, test_sety, color="darkorange", label="real values",
linewidth=2)
plt.plot(test_time, y_5, color="cornflowerblue", label="predicted",
linewidth=2)

#Grid search

# SVR Hyperparameter Tests
param_grid_svr = {'svr__C': [210000,220000,230000,240000],
              'svr__epsilon': [0.2],
              'svr__gamma':['auto'],
              'svr__kernel': ['rbf']}

grid_svr =
Pipeline(steps=[('scaler',preprocessing.StandardScaler()),('svr',
svm.SVR())])

#Decision Tree Regressor hyperparameter tests
param_grid_tree = {'tree__criterion':['squared_error', 'friedman_mse',
'absolute_error', 'poisson'],
                   'tree__splitter':['best', 'random'],
                   'tree__max_depth': [ 5, 10, 20, 22, 40, 50 ],
                   'tree__min_samples_split': [ 2, 4, 8 ],
                   'tree__min_samples_leaf': [1, 10, 20, 40],

                   }
param_grid_tree1 = {'criterion':['squared_error', 'friedman_mse',
'absolute_error', 'poisson'],
                   'splitter':['best', 'random'],
                   'max_depth': [ 5, 10, 20, 22, 40, 50 ],
                   'min_samples_split': [ 2, 4, 8 ],
                   'min_samples_leaf': [1, 10, 20, 40],

                   }
param_grid_tree2 = {'tree__criterion':['squared_error'],
                   'tree__splitter':['best'],
                    'tree__random_state':[0],
                   'tree__max_depth': [ 8, 9, 10,11,12,13, 22, 40, 80,
120 ],
                   'tree__min_samples_split': [ 4 ],
                   'tree__min_samples_leaf': [20]

                   }
param_grid_tree3 = {'random_state':[0],
```

```
                    'criterion':['squared_error', 'friedman_mse'],
                   'splitter':['best', 'random'],
                   'max_depth': [ 9 ],
                   'min_samples_split': [ 2 ],
                   'min_samples_leaf': [21],

                   }
extra_tree_features = {
                       'tree__max_features':['None','auto', 'sqrt',
'log2'],
                   #'tree__min_impurity_decrease':[0,0.1,0.2,1],
                   #'tree__ccp_alpha':[0,0.1,0.5,1]
                       }
grid_params_voting = { #Tests for Ensemble Models
               'weights': [(0.625, 0.375),(0.75, 0.25),(0.875,
0.125),(1, 0)]}

grid_tree = DecisionTreeRegressor()
#grid_tree =
Pipeline(steps=[('scaler',preprocessing.StandardScaler()),('tree',
DecisionTreeRegressor())])

#The selection of the above parameters can be typed here to perform the
corresponding test
grid = GridSearchCV(VotingRegressor(estimators=[('regr2',
regr_2),('regr1', regr_1)]), param_grid=grid_params_voting, refit =
True, verbose = 3,n_jobs=-1)
# Fitting the model for grid search
grid.fit(train_setx, train_sety)

# Print best parameter after tuning
print(grid.best_params_)
grid_predictions = grid.predict(test_setx)

# print the Score of the best parameters
print("R-squared:", format(grid.best_score_, ".3f"))

#End of Gridsearch

# Prediction 1 Model Saving
filename = 'adegapalmela_temp_solar.sav'
pickle.dump(regr_1, open(filename, 'wb'))

# Export Correlation Data

s1 = pd.Series(np.corrcoef(wvalues1temp,pvvalues)[0,
1],name='Temperature-PV Production Correlation')
s2 = pd.Series(np.corrcoef(wvalues2humid,pvvalues)[0,
1],name='Humidity-PV Production Correlation')
s3 = pd.Series(np.corrcoef(wvalues3wind,pvvalues)[0, 1],name='Wind
Speed-PV Production Correlation')
s4 = pd.Series(np.corrcoef(wvalues4cloud,pvvalues)[0, 1],name='Cloud
Cover-PV Production Correlation')
s5 = pd.Series(np.corrcoef(wvalues5rad,pvvalues)[0, 1],name='Solar
radiation-PV Production Correlation')

exportcorrdata = pd.concat([s1,s2,s3,s4,s5], axis=1)
```

```
# Export data to Excel
exporthourlystats = pd.DataFrame({

'pvhmeansyearly':pvhmeansyearly,'pvhstdevyearly':pvhstdevyearly,

'pvhmeanssummer':pvhmeanssummer,'pvhstdevsummer':pvhstdevsummer,

'pvhmeanswinter':pvhmeanswinter,'pvhstdevwinter':pvhstdevwinter
                                        }) # Hourly statistics

exportdatalegend = pd.DataFrame({'Legend':blegend}) # Legend explaining
the statistics page

# Export Arranged Data

exportbdata = pd.concat([s1,s2,s3,s4,s5,s7,s8,s9,s10,s11,s12,s13],
axis=1) # Must be written like this to allow different size lists in
the same worksheet

# Create a Pandas Excel writer using XlsxWriter as the engine.
try:
    bwriter = pd.ExcelWriter('Exported_Data.xlsx', engine='xlsxwriter')
except:
    # Colab version
    bwriter = pd.ExcelWriter('/content/drive/MyDrive/Colab
Notebooks/ExportedData.xlsx', engine='xlsxwriter')

# Saving the Excel file
exportbdata.to_excel(bwriter, sheet_name='Data')

exportcorrdata.to_excel(bwriter, sheet_name='Correlation Data')
exporthourlystats.to_excel(bwriter, sheet_name='Hourly statistics')
exportdatalegend.to_excel(bwriter, sheet_name='Legend')

bwriter.save()
```

Appendix B – Code for Pre-processing data

```
import pandas as pd
import datetime
import time
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import statistics as stat
import xlsxwriter
import math
import numpy as np

try: # Normal offline run
    wdata = pd.read_csv (r'BD4NRG.csv', delimiter=';')
    weatherdata = wdata.values.tolist()
    data =
pd.read_excel(r'BD4NRG_Pilot3_ASM_Headquarters_Power_Profiles.xlsx',she
et_name="Building")
    buildingdata = data.values.tolist()
    data =
pd.read_excel(r'BD4NRG_Pilot3_ASM_Headquarters_Power_Profiles.xlsx',she
et_name="Photovoltaic")
    pvdata = data.values.tolist()
except: # For Colab
    wdata = pd.read_csv (r'/content/drive/MyDrive/Colab
Notebooks/BD4NRG.csv', delimiter=';')
    weatherdata = wdata.values.tolist()
    data = pd.read_excel(r'/content/drive/MyDrive/Colab
Notebooks/BD4NRG_Pilot3_ASM_Headquarters_Power_Profiles.xlsx',sheet_nam
e="Building")
    buildingdata = data.values.tolist()
    data = pd.read_excel(r'/content/drive/MyDrive/Colab
Notebooks/BD4NRG_Pilot3_ASM_Headquarters_Power_Profiles.xlsx',sheet_nam
e="Photovoltaic")
    pvdata = data.values.tolist()

def datetotimestamp(buildingdata):# Convert date columns into timestamp
    btime = [] # List of timestamps
    btimestring = ""
    for i in range(len(buildingdata)):
        #btimestring =(str(buildingdata[i][2]) + "/" +
str(buildingdata[i][1]) + "/" + str(buildingdata[i][0]) + " " +
str(buildingdata[i][3]).split("+",1)[0] )
        btimestring =(str(buildingdata[i][2]) + "/" +
str(buildingdata[i][1]) + "/" + str(buildingdata[i][0]) + " " +
str(buildingdata[i][3]).split("+",1)[0] + "+0" +
str(buildingdata[i][3]).split("+",1)[1] + ":00" )
        btime.append(datetime.datetime.strptime(btimestring,"%d/%m/%Y
%H:%M:%S%z").timetuple())
    return btime

def wdatetotimestamp(weatdata):# Convert date columns into timestamp
from weather data and fix NaN values

    wtimestring = ""
    for i in range(len(weatdata)):
```

```
        wtimestring = str(int(weatdata[i][0])) + "+01:00"
        weatdata[i][0] =
datetime.datetime.strptime(wtimestring,"%Y%m%d%H%M%z").timetuple()

        for j in range(1,len(weatdata[i])):
            if np.isnan(weatdata[i][j]):
                weatdata[i][j] = weatdata[i-1][j]

    return weatdata

def pvcleanup(pv,pvtime): # Clean up power data
    for i in range(len(pv)):
        if pv[i][4] < 0:
            pv[i][4] = -pv[i][4] # Retrieve absolute power values to
correct reverse power flow values
        if pv[i][4] < 100 and ( pvtime[i].tm_hour > 17 or
pvtime[i].tm_hour < 7 ):
            pv[i][4] = 0 # Eliminates noise - Power produced after 5pm
and before 7am less than 100 Watt is set to zero
    return pv

def add10mintohourly(buildingdata, btime): # Convert 10-minute values
to hourly Wh values
    bhourly = [] #List Containing the timestamp and the energy consumed
    hourcounter = 0
    hourlyproduction = 0
    currenthour = 0
    for i in range(len(btime)-1): # Omit the last element of the list
to solve the out of index on btime[i+1] error
        hourlyproduction += buildingdata[i][4]
        hourcounter += 1 # Some elements are missing, not always 6 per
hour, so they are counted to calculate Wh
        if btime[i].tm_hour != btime[i+1].tm_hour:
            currenthour = list(btime[i])
            currenthour[4] = 0 # Set the timestamp minutes to zero
            currenthour[5] = 0 # Set the timestamp seconds to zero

bhourly.append([time.struct_time(tuple(currenthour)),hourlyproduction/h
ourcounter])
            hourcounter = 0
            hourlyproduction = 0

    return bhourly

def keepmatchingdatesonly(bhour,pvhour,wdata): # Keep only the data for
each set corresponding to the same dates and times

    pvcommondata = []
    bcommondata = []
    wcommondata = []
    lateststart = 0
    earliestfinish = 0
    lateststart =
max(time.mktime(bhour[1][0]),time.mktime(pvhour[1][0]),time.mktime(wdat
a[1][0]))
    earliestfinish = min(time.mktime(bhour[-1][0]),time.mktime(pvhour[-
1][0]),time.mktime(wdata[-1][0]))
```

```
    for i in range(len(bhour)):
        if earliestfinish >= time.mktime(bhour[i][0]) >= lateststart:
            bcommondata.append(bhour[i])
    for i in range(len(pvhour)):
        if earliestfinish >= time.mktime(pvhour[i][0]) >= lateststart:
            pvcommondata.append(pvhour[i])
    for i in range(len(wdata)):
        if earliestfinish >= time.mktime(wdata[i][0]) >= lateststart:
            wcommondata.append(wdata[i])
    return bcommondata, pvcommondata, wcommondata

def fixtimedata(bhour,correctingpv = 0):
    bhournew = []
    fillervalue = []
    meanval = []
    for i in range(1,len(bhour)-1):
        if (datetime.datetime.fromtimestamp(time.mktime(bhour[i+1][0]))
- datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) !=
datetime.timedelta(hours=1)):
            if bhour[i][0] == bhour[i-1][0] and
((datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) -
datetime.datetime.fromtimestamp(time.mktime(bhour[i+1][0]))) ==
datetime.timedelta(hours=2)):
                # If the time is the same as the previous cell and two
hours later in the one after that then it is just DST so add the value
normally

bhournew.append([(datetime.datetime.fromtimestamp(time.mktime(bhour[i][
0])) + datetime.timedelta(hours=1)).timetuple(),bhour[i][1]])
            if bhour[i][0] != bhour[i-1][0] and
((datetime.datetime.fromtimestamp(time.mktime(bhour[i+1][0])) -
datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])))  ==
datetime.timedelta(hours=2)):
                #print(bhour[i][0])
                fillervalue =
[(datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) +
datetime.timedelta(hours=1)).timetuple(),bhour[i-24][1]]
                ''' For using weekly averages to replace the value -
not working
                for j in range(-7,7):
                    meanval.append(bhour[i + j*24][1])
                print(meanval)
                meanval = stat.mean(meanval)
                fillervalue =
[(datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) +
datetime.timedelta(hours=1)).timetuple(),meanval]
                print(fillervalue)
                '''
                bhournew.append(bhour[i])
                bhournew.append(fillervalue) # Fill in the value of the
previous day for the missing data
            if
((datetime.datetime.fromtimestamp(time.mktime(bhour[i+1][0])) -
datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])))  >
datetime.timedelta(hours=2)):
            # Fill missing values if the hours missing are over 2 but
less than 24
```

```
                hours_missing =
datetime.datetime.fromtimestamp(time.mktime(bhour[i+1][0])) -
datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) -
datetime.timedelta(hours=1)
                counter1 = 0
                bhournew.append(bhour[i])
                while hours_missing > datetime.timedelta(hours=0):
                    fillervalue =
[(datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) +
datetime.timedelta(hours=1+counter1)).timetuple(),(bhour[i-
24+counter1][1]+bhour[i+24+counter1][1])/2]
                    bhournew.append(fillervalue) # Fill in the value of
the previous day for the missing data

                    counter1 += 1
                    hours_missing -= datetime.timedelta(hours=1)
            if
((datetime.datetime.fromtimestamp(time.mktime(bhour[i+1][0])) -
datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])))  >
datetime.timedelta(days=1)):
                print('Error, missing values!', bhour[i][0])
                print(bhour[i][0], 'is where the value is missing')
                sys.exit("Error message")

        # If the timedifference is over 24 hours, the value is not
added and an error is displayed
        # If the time is the same and the time difference with the next
is not 2 hours, the value is also not added
        else:
            if datetime.datetime.fromisoformat('2019-09-09 00:00:00')
>= datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) >=
datetime.datetime.fromisoformat('2019-08-23 00:00:00') and correctingpv
== 1:
            # If specific dates with known missing values match, use
the average data from the previous and next month

bhournew.append([(datetime.datetime.fromtimestamp(time.mktime(bhour[i][
0]))).timetuple(),(bhour[i-30*24][1]+bhour[i+30*24][1])/2])
            elif datetime.datetime.fromisoformat('2018-08-27 00:00:00')
>= datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) >=
datetime.datetime.fromisoformat('2018-08-22 00:00:00') and correctingpv
== 1:
            # If specific dates with known missing values match, use
the average data from the previous and next week

bhournew.append([(datetime.datetime.fromtimestamp(time.mktime(bhour[i][
0]))).timetuple(),(bhour[i-7*24][1]+bhour[i+7*24][1])/2])
            elif datetime.datetime.fromisoformat('2021-05-09 00:00:00')
>= datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) >=
datetime.datetime.fromisoformat('2021-05-05 00:00:00') and correctingpv
== 1:
            # If specific dates with known missing values match, use
the average data from the previous and next week

bhournew.append([(datetime.datetime.fromtimestamp(time.mktime(bhour[i][
0]))).timetuple(),(bhour[i-7*24][1]+bhour[i+7*24][1])/2])
```

```
            else: # In any other case, add the value normally
                bhournew.append(bhour[i])
    return bhournew


def testmissingdata(bhour): #Print all values which do not follow
hourly sequence - Should only print DST changes in March if all is OK
    bhournew = []
    for i in range(1,len(bhour)):
        if datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])) -
datetime.datetime.fromtimestamp(time.mktime(bhour[i-1][0])) !=
datetime.timedelta(hours=1):
            print(datetime.datetime.fromtimestamp(time.mktime(bhour[i-
1][0])))

print(datetime.datetime.fromtimestamp(time.mktime(bhour[i][0])))

# Get weather timestamps
weatherdata = wdatetotimestamp(weatherdata)

# Data for building consumption
hourlystatslabel = 'Hourly Statistics'
btime = datetotimestamp(buildingdata)
bhourly = add10mintohourly(buildingdata, btime)

# Data for PV
pvtime = datetotimestamp(pvdata)
pvcleanup(pvdata,pvtime)
pvhourly = add10mintohourly(pvdata, pvtime)

# Keep only the common dates among all data sets
bhourly, pvhourly, weatherdata = keepmatchingdatesonly(bhourly,
pvhourly, weatherdata)

# Fill in blank hours
weatherdata = fixtimedata(weatherdata)
bhourly = fixtimedata(bhourly)
pvhourly = fixtimedata(pvhourly)

datesinstring = []
day = []
month = []
season = []
year = []
hour = []
weekday = []
bvalues = []
pvvalues = []
wvalues = []
wvalues1temp = []
wvalues2humid = []
wvalues3wind = []
wvalues4cloud = []
wvalues5rad = []
```

```
for i in range(len(bhourly)): # Check if dates match and prepare the
lists for each value
    if bhourly[i][0] == pvhourly[i][0] == weatherdata[i][0]:
        try: # Will raise an error when trying the time when DST
starts, ignore that value -will omit a (erroneous) value of the
original index
            wvalues2humid.append(weatherdata[i][2])
            wvalues3wind.append(weatherdata[i][3])
            wvalues4cloud.append(weatherdata[i][4])
            wvalues5rad.append(weatherdata[i][5])

            datesinstring.append(time.strftime('%Y-%m-%d %H:%M:%S',
bhourly[i][0]))
            day.append(time.strftime('%d', bhourly[i][0]))
            month.append(time.strftime('%m', bhourly[i][0]))
            # Season
            if 3 > int(time.strftime('%m', bhourly[i][0])) >= 6:
                season.append(2) #Spring
            elif 6 > int(time.strftime('%m', bhourly[i][0])) >= 9:
                season.append(1) #Summer
            elif 9 > int(time.strftime('%m', bhourly[i][0])) >= 12:
                season.append(3) #Autumn
            else:
                season.append(4) #Winter
            year.append(time.strftime('%Y', bhourly[i][0]))
            hour.append(time.strftime('%H', bhourly[i][0]))
            weekday.append(time.strftime('%w', bhourly[i][0])) #
Weekday needs to be taken into account for consumption - it is not
exported to excel
            bvalues.append(bhourly[i][1])
            pvvalues.append(pvhourly[i][1])

            wvalues1temp.append(weatherdata[i][1])

        except:
            pass

    else:
        print("Error: Date mismatch")

# Export Arranged Data

s1 = pd.Series(datesinstring, name='Date/Time')
s2 = pd.Series(day, name='Day')
s3 = pd.Series(month, name='Month')
s4 = pd.Series(year, name='Year')
s5 = pd.Series(hour, name='Hour')
s6 = pd.Series(bvalues, name='bvalues')
s7 = pd.Series(pvvalues, name='pvvalues')
s8 = pd.Series(wvalues1temp, name='air temperature [B°C]')
s9 = pd.Series(wvalues2humid, name='relative humidity [%]')
s10 = pd.Series(wvalues3wind, name='wind speed[m/s]')
s11 = pd.Series(wvalues4cloud, name='cloudcover [%]')
s12 = pd.Series(wvalues5rad, name='global radiation [W/m^2]')
```

```
exportbdata = pd.concat([s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12],
axis=1) # Must be written like this to allow different size lists in
the same worksheet


# Create a Pandas Excel writer using XlsxWriter as the engine.
try:
    bwriter = pd.ExcelWriter('Exported_Data.xlsx', engine='xlsxwriter')
except:
    # Colab version
    bwriter = pd.ExcelWriter('/content/ExportedData.xlsx',
engine='xlsxwriter')

# Saving the Excel file
exportbdata.to_excel(bwriter, sheet_name='Data')

bwriter.save()
```

Bibliography

1. **The importance and source of solar forecasting. [Online] https://www.solaranywhere.com/2017/accurate-solar-forecasting-value/.**

2. *Solar generation prediction using the ARMA model in a laboratory-level micro-grid in IEEE proceedings of the third international conference on smart grid communications (SmartGridComm) p. 528–33.* Huang R, Huang T, Gadh R, Li N. s.l. : IEEE, 2012.

3. *Forecasting of photovoltaic power generation and model optimization: A review.* Utpal Kumar Das, University of Malaya. Kuala Lumpur : Elsevier.

4. D. S. Pillai, N. Rajasekar, J. P. Ram, and V. K. Chinnaiyan. Design and testing of two phase array reconfiguration procedure for maximizing power in solar PV systems under partial shade conditions (PSC). *Energy Conv. Manag.* 2018, Vol. 178.

5. Dalton G, Lockington D, Baldock T. Feasibility analysis of renewable energy supply options for a grid-connected large hotel. *Renewable Energy.* 2009, Vol. 34.

6. Mitsuru Kudo, Akira Takeuchi,Yousuke Nozaki,Hisahito Endo,Jiro Sumita. Forecasting electric power generation in a photovoltaic power system for an energy network. *Electrical Engineering in Japan.* 2009, Vol. 167.

7. Muhammad Naveed Akhter, Saad Mekhilef, Hazlie Mokhlis, Noraisyah Mohamed Shah. Review on forecasting of photovoltaic power generation based on machine learning and metaheuristic techniques. *IET Renewable Power generation.* 2019, Vol. 13, 7.

8. *Comparative study of power forecasting methods for PV stations.* Huang Y, Lu J, Liu C, Xu X, Wang W, Zhou X. s.l. : Proceedings of international conference on power system technology (POWERCON). IEEE; 2010. p. 1–6..

9. *Forecasting power output of photovoltaic systems based on weather classification and support vector machines.* Shi J, Lee W-J, Liu Y, Yang Y, Wang P. s.l. : IEEE Trans Ind Appl 2012. 48:1064–9.

10. *Validation of short and medium term operational solar radiation forecasts in the US. Sol Energy.* Perez R, Kivalov S, Schlemmer J, Hemker K, Renné D, Hoff TE. 2010. 84:2161–72..

11. *Wind power forecasting and error analysis using the autoregressive moving average modeling. Power & Energy Society General Meeting.* Rajagopalan S, Santoso S. 2009.

12. J., Boland. Time series modelling of solar radiation. *Modeling Solar Radiation at the Earth's Surface pp 283-312.* s.l. : Springer, 2008.

13. *Arima model and exponential smoothing method: a comparison. AIP Conference Proceedings p. 1312–21.* Wan Ahmad WKA, Ahmad S, Ishak A, Hashim I, Ismail ES, Nazar R. s.l. : AIP, 2013.

14. *Review of solar irradiance forecasting methods and a proposition for small-scale insular grids. Renewable and Sustainable Energy Reviews.* Diagne M, David M, Lauret P, Boland J, Schmutz N. 2013, Vol. 27.

15. ***Short term photovoltaic power generation forecasting using neural network. In: Proceedings of the 11th international conference on environment and electrical engineering p. 706–11.*** **Oudjana SH, Hellal A, Mahamed IH. s.l. : IEEE, 2021.**

16. Hossain MS, Chao OZ, Ismail Z, Noroozi S, Yee KS. Artificial neural networks for vibration based inverse parametric identifications: A review. ***Applied Soft Computing.*** **2017, Vol. Elsevier, 52.**

17. Mellit A, Kalogirou SA. Artificial intelligence techniques for photovoltaic applications: A review. ***Progress in Energy and Combustion Science.*** **2008, Vol. 34, 5.**

18. Zhang Y, Chen G, Malik O, Hope G. An artificial neural network based adaptive power system stabilizer. ***IEEE Trans Energy Convers.*** **1993.**

19. ***Weather forecasting using photovoltaic system and neural network. Computational Intelligence, Communication Systems and Networks (CICSyN) s International Conference on: IEEE p. 96–100.*** **Isa IS, Omar S, Saad Z, Noor NM, Osman MK. 2010.**

20. ***Application of neural network to one-day-ahead 24h generating power forecasting for photo-voltaic system. In: Proceedings of international conference on intelligent systems applications to power systems, ISAP.*** **Yona A, Senjyu T, Saber AY, Funabashi T, Sekine H, Kim C-H. s.l. : IEEE, 2007.**

21. Malki HA, Karayiannis NB, Balasubramanian M. Short-term electric power load forecasting using feedforward neural networks. ***Expert Syst.*** **2004.**

22. ***Application of recurrent neural network to short-term-ahead generating power forecasting for photovoltaic system. 2007 IEEE Power Engineering Society General Meeting.*** **Yona A, Senjyu T, Funabashi T.**

23. Sözen A, Arcaklioğlu E, Özalp M, Kanit EG. Use of artificial neural networks for mapping of solar potential in Turkey. ***Applied Energy.*** **2004, Vol. 77, 3.**

24. ***Traffic flow forecasting with particle swarm optimization and support vector regression. In: Proceedings of the IEEE 17th international conference on intelligent transportation systems (ITSC).*** **Hu J, Gao P, Yao Y, Xie X. 2014.**

25. Zhang F, Deb C, Lee SE, Yang J, Shah KW. Time series forecasting for building energy consumption using weighted Support Vector Regression with differential evolution optimization technique. ***Energy and Buildings.*** **2016, Vol. 126, Pages 94-103.**

26. Yona A, Senjyu T, Funabashi T, Kim C-H. Determination Method of Insolation Prediction With Fuzzy and Applying Neural Network for Long-Term Ahead PV Power Output Correction. ***IEEE Transactions on Sustainable Energy.*** **Vol. 4, 2.**

27. Shyi-Ming Chen, Yu-Chuan Chang, Jeng-Shyang Pan. Fuzzy Rules Interpolation for Sparse Fuzzy Rule-Based Systems Based on Interval Type-2 Gaussian Fuzzy Sets and Genetic Algorithms. ***IEEE Transactions on Fuzzy Systems.*** **2013, Vol. 21, 3.**

28. ***Automatic sunspot classification for real-time forecasting of solar activities. In: Proceedings of the 3rd international conference on recent advances in space technologies 2007 RAST'07.*** **Colak T, Qahwaji R. s.l. : IEEE, 2007.**

29. Martin János Mayer, Gyula Gróf. Extensive comparison of physical models for photovoltaic power forecasting. ***Applied Energy.*** **2021, Vol. 283.**

30. Spyros Theocharides, Marios Theristis, George Makridesm, Marios Kynigos, Chrysovalantis Spanias, George E. Georghiou. Comparative Analysis of Machine Learning Models for Day-Ahead Photovoltaic Power Production Forecasting. ***Energies.*** **2021, Vol. 14.**

31. Alfredo Nespoli, Emanuele Ogliari,Sonia Leva,Alessandro Massi Pavan,Adel Mellit,Vanni Lughi, Alberto Dolara. Day-Ahead Photovoltaic Forecasting: A Comparison of the Most Effective Techniques. ***Energies.*** **2019, Vol. 12.**

32. ***Solar Irradiance Forecasting Using Triple Exponential Smoothing in 2018 International Conference on Smart Energy Systems and Technologies (SEST).*** **Dev, Soumyabrata, et al. 2018.**

33. ***Machine learning algorithms for photovoltaic system power output prediction in 2018 IEEE International Energy Conference (ENERGYCON).*** **Theocharides, Spyros, et al. 2018.**

34. ***Hour-ahead solar PV power forecasting using SVR based approach in: 2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT).*** **Alfadda, Abdullah, et al. 2017.**

35. Guido Cervone, LauraClemente-Harding, Stefano Alessandrini, Luca Delle Monache. Short-term photovoltaic power forecasting using Artificial Neural Networks and an Analog Ensemble. ***Renewable Energy.*** **2017, Vol. 108.**

36. Francesco Grimaccia, Sonia Leva, Marco Mussetta, Emanuele Ogliari. ANN Sizing Procedure for the Day-Ahead Output Power Forecast of a PV Plant. ***Appl. Sci. .*** **2017, Vol. 7.**

37. Utpal Kumar Das, Kok Soon Tey,Mehdi Seyedmahmoudian,Mohd Yamani Idna Idris,Saad Mekhilef,Ben Horan, Alex Stojcevski. SVR-Based Model to Forecast PV Power Generation under Different Weather Conditions. ***Energies.*** **2017, Vol. 10.**

38. ***PV PRODUCTION FORECASTING MODEL BASED ON ARTIFICIAL NEURAL NETWORKS (ANN).*** **Spyros Theocharides, George Makrides, Venizelos Venizelou, Paris Kaimakis, Andreas Kyprianou, George E. Georghiou. s.l. : FOSS Research Centre for Sustainable Energy, 2017.**

39. ***Short-term PV power forecasting using Support Vector Regression and local monitoring data in 2016 International Renewable and Sustainable Energy Conference (IRSEC).*** **Fentis, Ayoub, et al. 2016.**

40. ***Comparison of SARIMAX, SARIMA, modified SARIMA and ANN-based models for short-term PV generation forecasting In: 2016 IEEE International Energy Conference (ENERGYCON).*** **Stylianos I. Vagropoulos, G. I. Chouliaras, E. G. Kardakos, C. K. Simoglou, A. G. Bakirtzis. s.l. : IEEE, 2016.**

41. Björn Wolff, Jan Kühnert, Elke Lorenz, Oliver Kramer, Detlev Heinemann. Comparing support vector regression for PV power forecasting to a physical modeling approach using measurement, numerical weather prediction, and cloud motion data. *Solar Energy.* 2016, Vol. 135.

42. Zhaoxuan Li, SM Mahbobur Rahman,Rolando Vega, Bing Dong. A Hierarchical Approach Using Machine Learning Methods in Solar Photovoltaic Energy Production Forecasting. *Energies.* 2016, Vol. 55.

43. Almeida MP, Perpiñán O, Narvarte L. PV power forecast using a nonparametric PV model. *Solar Energy.* 2015, Vol. 115.

44. Chu Y, Urquhart B, Gohari SM, Pedro HT, Kleissl J, Coimbra CF. Short-term reforecasting of power output from a 48 MWe solar PV plant. *Solar Energy.* 2015, Vol. 112.

45. R. De Leone, M. Pietrini & A. Giovannelli. Photovoltaic energy production forecast using support vector regression. *Neural Computing and Applications.* 2015, Vol. 26.

46. Dolara A, Leva S, Manzolini G. Comparison of different physical models for PV power output prediction. *Solar Energy.* 2015, Vol. 119.

47. Alberto Dolara, Francesco Grimaccia, Sonia Leva, Marco Mussetta, Emanuele Ogliari. A Physical Hybrid Artificial Neural Network for Short Term Forecasting of PV Plant Power Output. *Energies.* 2015, Vol. 8.

48. Leva S, Dolara A, Grimaccia F, Mussetta M, Ogliari E. Analysis and validation of 24 hours ahead neural network forecasting of photovoltaic output power. *Mathematics and Computers in Simulation.* 2017, Vol. 131.

49. Liu J, Fang W, Zhang X, Yang C. An improved photovoltaic power forecasting model with the assistance of aerosol index data. *IEEE Trans Sustain Energy.* 2015, Vol. 6.

50. Ramsami P, Oree V. A hybrid method for forecasting the energy output of photovoltaic systems. *Energy Conversion and Management.* 2015, Vol. 95.

51. *Forecasting solar power generated by grid connected PV systems using ensembles of neural networks. In: Proceedings of international joint conference on neural networks (IJCNN) 2015.* Rana M, Koprinska I, Agelidis VG. s.l. : IEEE.

52. Yang C, Thatte AA, Xie L. Multitime-Scale Data-Driven Spatio-Temporal Forecast of Photovoltaic Generation. *IEEE Transactions on Sustainable Energy.* 2015, Vol. 6.

53. Dazhi Yang, Vishal Sharma, Zhen Ye, Lihong Idris Lim, Lu Zhao, Aloysius W.Aryaputera. Forecasting of global horizontal irradiance by exponential smoothing, using decompositions. *Energy.* 2015, Vol. 81.

54. Zhu H, Li X, Sun Q, Nie L, Yao J, Zhao G. A Power Prediction Method for Photovoltaic Power Plant Based on Wavelet Decomposition and Artificial Neural Networks. *Energies .* 2016, Vol. 9.

55. ***Hybrid Model Analysis and Validation for PV energy production forecasting, 2014 International Joint Conference on Neural Networks (IJCNN).*** **A. Gandelli, F. Grimaccia, S. Leva, M. Mussetta, E. Ogliari. 2014.**

56. De Giorgi M, Congedo P, Malvoni M. Photovoltaic power forecasting using statistical methods: impact of weather data. ***IET Sci Meas Technol 2014.*** **Vol. 8.**

57. Junior JGdSF, Oozeki T, Ohtake H, Shimose K-i, Takashima T, Ogimoto K. Forecasting Regional Photovoltaic Power Generation - A Comparison of Strategies to Obtain One-Day-Ahead Data. ***Energy Procedia.*** **57, 2014.**

58. ***Distributed PV. power forecasting using genetic algorithm based neural network approach. In: Proceedings of international conference on advanced mechatronic systems (ICAMechS) 2014.*** **Tao Y, Chen Y. s.l. : IEEE, 2014.**

59. Yang H-T, Huang C-M, Huang Y-C, Pai Y-S. A Weather-Based Hybrid Method for 1-Day Ahead Hourly Forecasting of PV Power Output. ***IEEE Transactions on Sustainable Energy.*** **2014, Vol. 5, 3.**

60. Yanting Li, Yan Su, Lianjie Shu. An ARMAX model for forecasting the power output of a grid connected photovoltaic system. ***Renewable Energy.*** **2014, Vol. 66.**

61. Zibo Dong, Dazhi Yang, Thomas Reindl, Wilfred M.Walsh. Short-term solar irradiance forecasting using exponential smoothing state space model. ***Energy.*** **2013, Vol. 55.**

62. ***Solar PV. power generation forecast using a hybrid intelligent approach. Power and Energy Society General Meeting (PES) IEEE.*** **Haque AU, Nehrir MH, Mandal P. 2013.**

63. ***Evaluation of a Kalman predictor approach in forecasting PV solar power generation. In: Proceedings of the 4th IEEE international symposium on power electronics for distributed generation systems (PEDG).*** **Tuyishimire B, McCann R, Bute J. 2013.**

64. Yona A, Senjyu T, Funabashi T, Kim C-H. Determination Method of Insolation Prediction With Fuzzy and Applying Neural Network for Long-Term Ahead PV Power Output Correction. ***IEEE Transactions on Sustainable Energy.*** **2013, Vol. 4.**

65. Mandal P, Madhira STS, Meng J, Pineda RL. Forecasting Power Output of Solar Photovoltaic System Using Wavelet Transform and Artificial Intelligence Techniques. ***Procedia Computer Science.*** **2012, Vol. 12.**

66. ***Development of GRBFN with global structure for PV generation output forecasting. Power and energy society general meeting 2012 IEEE.*** **Mori H, Takahashi M. s.l. : IEEE, 2012.**

67. Pedro HT, Coimbra CF. Assessment of forecasting techniques for solar power production with no exogenous inputs. ***Solar Energy.*** **2012, Vol. 86.**

68. Shi J, Lee W-J, Liu Y, Yang Y, Wang P. Forecasting power output of photovoltaic systems based on weather classification and support vector machines. ***IEEE Trans Ind Appl 2012.*** **Vol. 48.**

69. ***Short-term photovoltaic power forecasting with weighted support vector machine. In: Proceedings of IEEE international conference on automation and logistics (ICAL).*** **Xu R, Chen H, Sun X. 2012.**

70. Chen C, Duan S, Cai T, Liu B. Online 24-h solar power forecasting based on weather type classification using artificial neural network. ***Solar Energy.*** **2011, Vol. 85.**

71. Chupong C, Plangklang B. Forecasting power output of PV grid connected system in Thailand without using solar radiation measurement. ***Energy Procedia.*** **2011, Vol. 9.**

72. ***24-hour-ahead forecasting of energy production in solar PV systems. In: Proceedings of the 11th international conference on intelligent systems design and applications (ISDA) 2011.*** **Cococcioni M, D'Andrea E, Lazzerini B. s.l. : IEEE.**

73. Ding M, Wang L, Bi R. An ANN-based Approach for Forecasting the Power Output of Photovoltaic System. ***Procedia Environmental Sciences.*** **2011, Vol. 11.**

74. ***Development of algorithm for day ahead PV generation forecasting using data mining method. In: IEEE Proceedings of the 54th international midwest symposium on circuits and systems (MWSCAS) 2011.*** **Kang M-C, Sohn J-M, Park J-y, Lee S-K, Yoon Y-T.**

75. ***Detection and prediction of faults in photovoltaic arrays: A review in: 2018 IEEE 12th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG 2018).*** **AbdulMawjood, Kais, Refaat, Shady S. and Morsi, Walid G. s.l. : IEEE, 2018.**

76. ***Development of a Visual Inspection Checklist for Evaluation of Fielded PV Module Condition in PV Module Reliability Workshop, NREL.*** **Packard C. E., Wohlgemuth1, J. H., Kurtz, S. R. 2012.**

77. Siva Ramakrishna Madeti, S.N. Singh. A comprehensive study on different types of faults and detection techniques for solar photovoltaic system. ***Solar Energy.*** **2017, Vol. 158.**

78. Bosman Lisa B., Leon-Salas Walter D., Hutzel William and Soto Esteban A. PV System Predictive Maintenance: Challenges, Current Approaches, and Opportunities. ***Energies.*** **2020, Vol. 13.**

79. ***Shading analysis & improvement for distributed residential grid-connected photovoltaics systems. In Proceedings of the 52nd Annual Conference of the Australian Solar Council.*** **Bulanyi, P. and Zhang, R. Melbourne, Australia : s.n., 2014.**

80. Dhanup S. Pillai, N. Rajasekar. A comprehensive review on protection challenges and fault diagnosis in PV systems. ***Renewable and Sustainable Energy Reviews.*** **2018, Vol. 91.**

81. DS. Pillai, Frede Blaabjerg, Natarajan Rajasekar. A Comparative Evaluation of Advanced Fault Detection Approaches for PV Systems. ***IEEE JOURNAL OF PHOTOVOLTAICS.*** **2019, Vol. 9.**

82. ***Development of a model for photovoltaic arrays suitable for use in simulation studies of solar energy conversion systems.*** **Gow JA, Manning CD.**

83. ***Modelling, simulation and performance analysis of a PV array in an embedded environment. In: Proceedings of 42nd international universities power engineering conference, UPEC.*** **Chowdhury S, Taylor GA, Chowdhury SP, Saha AK, Song YH. s.l. : IEEE, 2007.**

84. ***Development of a two diode model for photovoltaic modules suitable for use in simulation studies. In: Proceedinga of power and energy engineering conference (APPEEC), Asia-Pacific.*** **Gupta S, Tiwari H, Fozdar M, Chandna V. s.l. : IEEE, 2012.**

85. Ram JP, Babu TS, Dragicevic T, Rajasekar N. A new hybrid bee pollinator flower pollination algorithm for solar PV parameter estimation. ***Energy Conversion and Management.*** **2017, Vol. 135.**

86. N.Rajasekar, Neeraja Krishna Kumar, Rini Venugopalan. Bacterial Foraging Algorithm based solar PV parameter estimation. ***Solar Energy.*** **2013, Vol. 97.**

87. Jordehi, A. Rezaee. Parameter estimation of solar photovoltaic (PV) cells: A review. ***Renewable and Sustainable Energy Reviews.*** **2016, Vol. 61.**

88. Chine W, Mellit A, Pavan AM, Kalogirou SA. Fault detection method for grid-connected photovoltaic plants. ***Renewable Energy.*** **2016, Vol. 66.**

89. Kymakis E, Kalykakis S, Papazoglou TM. Performance analysis of a grid connected photovoltaic park on the island of Crete. ***Energy Conversion and Management.*** **2009, Vol. 50.**

90. Silvestre S, Chouder A, Karatepe E. Automatic fault detection in grid connected PV systems. ***Solar Energy.*** **2013, Vol. 94.**

91. ***New method for fault detection of PV panels in domestic applications. In: Proceedings of the 3rd international conference on systems and control (ICSC).*** **Davarifar M, Rabhi A, El Hajjaji A, Dahmane M. s.l. : IEEE, 2013.**

92. ***Development of fault detection system in PV system. In: Proceedings of 33rd international telecommunications energy conference (INTELEC).*** **Shimakage T, Nishioka K, Yamane H, Nagura M, Kudo M. s.l. : IEEE, 2011.**

93. ***Evaluation of output performance of various photovoltaic systems in the hokuto mega-solar project. In: Proceedings of 32nd international telecommunications energy conference (INTELEC).*** **Nishioka K, Shimakage T, Yamane H, Kudo M, Ueda Y. s.l. : IEEE, 2010.**

94. ***Estimating effects of individual PV panel failures on PV array output. In: Proceedings of the 16th international conference on environment and electrical engineering (EEEIC).*** **M, Orkisz. s.l. : IEEE, 2016.**

95. ***Method for diagnosing photovoltaic array fault in solar photovoltaic system. In: Proceedings of Asia-Pacific power and energy engineering conference (APPEEC).*** **Xu X, Wang H, Zuo Y. s.l. : IEEE, 2011.**

96. ***A density peak-based clustering approach for fault diagnosis of photovoltaic arrays.*** **Lin, P., Lin, Y., Chen, Z., Wu, L., Chen, L., Cheng, S. s.l. : Int. J. Photo Energy, 2017.**

97. Takashima, T., Yamaguchi, J., Otani, K., Oozeki, T., Kato, K., Ishida, M. Experimental studies of fault location. ***PV module strings. Sol. Cells 93.*** **2009.**

98. ***Diagnosis of a power output lowering of PV array with a (−dI/dV)-V characteristic in: Photovoltaic Energy Conversion, Conference Record of the IEEE 4th World Conference, vol. 2, pp. 442–2445.*** **Miwa, M., Yamanaka, S., Kawamura, H., Ohno, H. 2006.**

99. Kaplanis, S., Kaplani, E. Energy performance and degradation over 20 years performance of BP c-Si PV modules. ***Simulation Modelling Practice and Theory.*** **2011, Vol. 19, 4.**

100. ***Zhao Y, Lehman B, Ball R, Mosesian J, de Palma JF Zhao Y, Lehman B, Ball R, Mosesian J, de Palma JF applied power electronics conference and exposition (APEC).*** **Zhao Y, Lehman B, Ball R, Mosesian J, de Palma JF. s.l. : IEEE, 2013.**

101. ***Mining imperfect data: Dealing with contamination and incomplete records.*** **Pearson, RK. s.l. : SIAM, 2005.**

102. Zhao, Ye, et al. Graph-Based Semi-supervised Learning for Fault Detection and Classification in Solar Photovoltaic Arrays. ***Ye Zhao; Roy Ball; Jerry Mosesian; Jean-François de Palma; Brad Lehman.*** **2015, Vol. 30.**

103. ***Fault experiments in a commercial-scale PV laboratory and fault detection using local outlier factor. In: Proceedings of 40th photovoltaic specialist conference (PVSC).*** **Zhao Y, Balboni F, Arnaud T, Mosesian J, Ball R, Lehman B. s.l. : IEEE, 2014.**

104. Chouder, A., Silvestre, S. Automatic supervision and fault detection of PV systems based on power losses analysis. ***Energy Convers. Manage.*** **2010, Vol. 51.**

105. Madeti, S.R., Singh, S.N. Online fault detection and the economic analysis of grid-connected photovoltaic systems. ***Energy.*** **2017, Vol. 134.**

106. L. Schirone, F. P. Califano,M. Pastena. Fault detection in a photovoltaic plant by time domain reflectometry. ***Prog. Photovoltaics Res. Appl. 2 (1), 35–44.*** **1994.**

107. ***Experimental studies of failure detection methods in PV module strings", in Photovoltaic Energy Conversion, Conference Record of the IEEE 4th World Conference, vol. 2.*** **Takashima, T., Yamaguchi, J., Otani, K., Kato, K., Ishida, M. 2006.**

108. Vergura S, Accaciani G, Amoruso V, Patrono G, Vacca F. Descriptive and inferential statistics for supervising and monitoring the operation of PV plants. ***IEEE Trans Ind Electron.*** **2009, Vol. 56.**

109. Gallardo-Saavedra, S., Hernández-Callejo, L. and Duque-Perez, O. Technological review of the instrumentation used in aerial thermographic inspection of photovoltaic plants. ***Renew. Sustain. Energy Rev.*** **2018, Vol. 93.**

110. Quater, P.B., et al. Light Unmanned Aerial Vehicles (UAVs) for cooperative inspection of PV plants. ***IEEE J. Photovolt.*** **2014, Vol. 4.**

111. ***Outdoor electroluminescence imaging of crystalline photovoltaic modules: Comparative study between manual ground-level inspections and drone-based aerial surveys. In***

Proceedings of the 32nd European Photovoltaic Solar Energy Conference and Exhibition. **Koch, S., et al. 2016.**

112. ***IR real-time analyses for PV system monitoring by digital image processing techniques. In Proceedings of the 2015 International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP).*** **Aghaei, M., et al. 2015.**

113. ***Inspecting PV-plants using aerial, drone-mounted infrared thermography system. In Proceedings of the 3rd Southern African Solar Energy Conference.*** **Buerhop-Lutz, C. and Scheuerpflug, H. 2015.**

114. ***Processing infrared image of PV modules for defects classification in: IEEE Proceedings of the International Conference on Renewable Energy Research and Applications (ICRERA '15), pp. 1337–1341.*** **Vergura, S., Marino, F., Carpentieri, M. 2015.**

115. ***Health diagnostics of PV panels by means of single cell analysis of thermographic images in: Proceedings of the IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC '16).*** **Guerriero, P., Cuozzo, G., Daliento, S. 2016.**

116. Hund, T.D., King, D.L. ***Analysis techniques used on field degraded photovoltaic modules. NASA STI/Recon Technical Report N, p.96.*** **1995.**

117. Thomas Kirchartz, Anke Helbig,Wilfried Reetz,Michael Reuter,Jürgen H. Werner,Uwe Rau. Reciprocity between electroluminescence and quantum efficiency used for the characterization of silicon solar cells. ***Progress in Photovoltaics.*** **2009, Vol. 17, 6.**

118. M. Kasemann, D. Grote,B. Walter,W. Kwapil,T. Trupke,Y. Augarten,R.A. Bardos,E. Pink,M.D. Abbott,W. Warta. Luminescence imaging for the detection of shunts on silicon solar cells. ***Progress in Photovoltaics.*** **2008, Vol. 16, 4.**

119. Breitenstein, O., Bauer, J., Trupke, T., Bardos, R.A. On the detection of shunts in silicon solar cells by photo-and electroluminescence imaging. ***Prog. Photovoltaics Res. Appl.*** **2008, Vol. 16, 4.**

120. Berghout, T., et al. A deep supervised learning approach for condition-based maintenance of naval propulsion systems. ***Ocean Eng.*** **2021, Vol. 221.**

121. Schmidhuber, J. Deep learning in neural networks: An overview. ***Neural Netw.*** **2015, Vol. 61.**

122. Rao, B.K.N. The Role of Artificial Intelligence (AI) in Condition Monitoring and Diagnostic Engineering Management (COMAThe Role of Artificial Intelligence (AI) in Condition Monitoring and Diagnostic Engineering Management (COMADEM): A Literature Survey. ***Am. J. Artif. Intell.*** **2021, Vol. 5, 17.**

123. Garoudja, E., et al. An enhanced machine learning based approach for failures detection and diagnosis of PV systems. ***Energy Convers. Manag.*** **2017, Vol. 151.**

124. Ali, M.U., et al. A machine learning framework to identify the hotspot in photovoltaic module using infrared thermography. ***Sol. Energy.*** **2020, Vol. 208.**

125. Mansouri M, Trabelsi M, Nounou H, Nounou M. Deep Learning-Based Fault Diagnosis ofDeep Learning-Based Fault Diagnosis of Photovoltaic Systems: A Comprehensive Review and Enhancement Prospects. ***IEEE Access.*** **2021, Vol. 9.**

126. Haghighat, M. S. Mirnaghi and F. Fault detection and diagnosis of large scale HVAC systems in buildings using data-driven methods: A comprehensive review. ***Energy Buildings.*** **2020, Vol. 229.**

127. X. Wang, Y. Zhao, and F. Pourpanah. ***Recent advances in deep learning.*** **2020.**

128. F. Aziz, A. U. Haq, S. Ahmad, Y. Mahmoud, M. Jalal, U. Ali. A novel convolutional neural network-based approach for fault classification in photovoltaic arrays. ***IEEE Access.*** **2020, Vol. 8.**

129. A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi. A survey of the recent architectures of deep convolutional neural networks. ***Artif. Intell. Rev.*** **2020, Vol. 53.**

130. A. I. Khalyasmaa, S. A. Eroshenko, V. A. Tashchilin, H. Ramachandran,T. Piepur Chakravarthi, and D. N. Butusov. Industry experience of developing day-ahead photovoltaic plant forecasting system based on machine learning. ***Remote Sens.*** **2020, Vol. 12, 20.**

131. G. Kovács, L. Tóth, D. Van Compernolle, and S. Ganapathy. Increasing the robustness of CNN acoustic models using autoregressive moving average spectrogram features and channel dropout. ***Pattern Recognit.Lett.*** **2017, Vol. 100.**

132. Z. Lin, K. Ji, X. Leng, and G. Kuang. Squeeze and excitation rank faster R-CNN for ship detection in SAR images. ***IEEE Geosci. Remote Sens.Lett.*** **2018, Vol. 16, 5.**

133. M. Canizo, I. Triguero, A. Conde, and E. Onieva. Multi-head CNN-RNN for multi-time series anomaly detection: An industrial case study. ***Neurocomputing.*** **2019, Vol. 363.**

134. B. Lin, S. Deng, H. Gao, and J. Yin. A multi-scale activity transition network for data translation in EEG signals decoding. ***IEEE/ACM Trans. Comput. Biol. Bioinf., early access.*** **2020.**

135. M. Xia, X. Zheng, M. Imran, and M. Shoaib. Data-driven prognosis method using hybrid deep recurrent neural network. ***Appl. Soft Comput. Article 106351.*** **2020, Vol. 93.**

136. C. K. M. Lee, K. K. H. Ng, C.-H. Chen, H. C. W. Lau, S. Y. Chung and T. Tsoi. American sign language recognition and training method with recurrent neural network. ***Expert Syst. Appl.*** **2021, Vol. 167.**

137. X. Li, X. Ma, F. Xiao, F.Wang, and S. Zh. Application of gated recurrent unit (GRU) neural network for smart batch production prediction. ***Energies.*** **2020, Vol. 13, 22.**

138. R. Zhu, X. Tu, and J. X. Huang. Deep learning on information retrieval and its applications. ***Deep Learning for Data Analytics.*** **2020.**

139. P. Park, P. D. Marco, H. Shin, and J. Bang. Fault detection and diagnosis using combined autoencoder and long short-term memory network. ***Sensors.*** **2019, Vol. 19, 21.**

140. Z. Yang, D. Gjorgjevikj, J. Long, Y. Zi, S. Zhang, and C. Li. Sparse autoencoder-based multi-head deep neural networks for machinery fault diagnostics with detection of novelties. ***Chin. J. Mech. Eng.*** **2021, Vol. 34.**

141. Chang, C.-H. Deep and shallow architecture of multilayer neural networks. ***IEEE Trans. Neural Netw. Learn. Syst.*** **2015, Vol. 26, 10.**

142. R. Souriau, J. Lerbet, H. Chen, and V. Vigneron. A review on generative Boltzmann networks applied to dynamic systems. ***Mech. Syst. Signal Process.*** **2021, Vol. 147.**

143. Guerriero, P., et al. Monitoring and diagnostics of PV plants by a wireless self-powered sensor for individual panels. ***IEEE J. Photovolt.*** **2015, Vol. 6.**

144. ***A low-cost photovoltaic (PV) array monitoring system. In Proceedings of the 2013 IEEE Conference on Clean Energy and Technology (CEAT), Lankgkawi, Malaysia.*** **Rivai, A. and Rahim, N.A. 2013.**

145. Chouder, A., et al. Monitoring, modelling and simulation of PV systems using LabVIEW. ***Sol. Energy.*** **2013, Vol. 91.**

146. ***Design and analysis of a low cost PV analyzer using Arduino UNO. In Proceedings of the 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES).*** **Anand, R., et al. 2016.**

147. Prieto, M.J., et al. Development of a wireless sensor network for individual monitoring of panels in a photovoltaic plant. ***Sensors.*** **2014, Vol. 14.**

148. Chine W, Mellit A, Lughi V, Malek A, Sulligoi G, Pavan AM. A novel fault diagnosis technique for photovoltaic systems based on artificial neural networks. ***Renewable Energy.*** **2016, Vol. 90.**

149. Hu Y, Gao B, Song X, Tian GY, Li K, He X. Facile preparation of artemisia argyi oil-loaded antibacterial microcapsules by hydroxyapatite-stabilized Pickering emulsion templating. ***Colloids and Surfaces B: Biointerfaces.*** **2013, Vol. 112.**

150. ***Fault detection of bypass circuit of PV module—Detection technology of open circuit fault location. In: Proceedings of the 19th international conference on electrical machines and systems (ICEMS).*** **Kase R, Nishikawa S. s.l. : IEEE, 2016.**

151. Yi Z, Etemadi AH. Line-to-line fault detection for photovoltaic arrays based on multiresolution signal decomposition and two-stage support vector machine. ***IEEE Transactions on Industrial Electronics.*** **2017.**

152. Scikit-learn.org. ***Decision Trees.*** **[Online] https://scikit-learn.org/stable/modules/tree.html#tree.**

153. MinXu, PakornWatanachaturaporn, Pramod K.Varshney, Manoj K.Arora. data, Decision tree regression for soft classification of remote sensing. ***Remote Sensing of Environment.*** **2005, Vol. 97, 3.**

154. ***A comparative analysis on linear regression and support vector regression, 2016 Online International Conference on Green Engineering and Technologies (IC-GET).*** **s.l. : IEEE.**

155. Fan Zhang, Lauren J.O'Donnell. Chapter 7 - Support vector regression. ***Machine Learning - Methods and Applications to Brain Disorders.***

156. scikit-learn.org. ***Support Vector Machines.*** **[Online] https://scikit-learn.org/stable/modules/svm.html#svm.**

157. scikit-learn.org. ***Linear Models.*** **[Online] https://scikit-learn.org/stable/modules/linear_model.html#linear-model.**

158. IBM.com. ***IBM Linear Regression.*** **[Online] https://www.ibm.com/topics/linear-regression.**

159. Catalao J, Pousinho H, Mendes V. Hybrid wavelet-PSO-ANFIS approach for short-term wind power forecasting in Portugal. ***IEEE Trans Sustain Energy 2011.***

160. Silvestre S, Chouder A, Karatepe E. Automatic fault detection in grid connected PV systems. ***Sol. Energy.*** **2013, Vol. 94.**