Title

Time series prediction for Electric Vehicle Charging Load and Solar Power Generation in the context of Smart Grid

Author

Majidpour, Mostafa

Publication Date

2016

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Time series prediction for Electric Vehicle Charging Load and Solar Power Generation in the context of Smart Grid

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Electrical Engineering

by

Mostafa Majidpour

2016

ABSTRACT OF THE DISSERTATION

Time series prediction
for Electric Vehicle Charging Load and Solar Power Generation
in the context of Smart Grid

by

Mostafa Majidpour

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2016

Professor Rajit Gadh, Co-Chair

Professor Jason L. Speyer, Co-Chair

In view of the success of machine learning based prediction algorithms in the recent years, in this study, we have employed a selection of these algorithms on some time series prediction problems in the context of smart grid. We have used real world data from the UCLA campus solar PV panels and parking lots. In the process of applying these algorithms on the Electric Vehicle (EV) charging load prediction problem, two new prediction algorithms have been proposed, namely Modified Pattern Sequence Forecasting (MPSF) and Time Weighted Dot Product Nearest Neighbor (TWDP NN). One of the objectives when predicting the EV charging load is speed of prediction since it is intended to be used in a real time application (smartphone application for EV customers). Using our dataset, TWDP NN decreased the processing time by a third.

As missing data is a significant concern in real world data, the effect of missing values on the prediction quality has been investigated. Six different imputation methods have been applied to compensate for missing values in EV charging data. Based on non-parametric statistical tests, suitable (or unsuitable) imputation methods for each prediction algorithm are recommended.

Forecasting of the Electric Vehicle (EV) charging load can be done based on two different datasets: data from the customer profile (charging record) and data from outlet measurements (station record). We found that charging records provide relatively faster prediction while putting customer privacy at jeopardy. On the other hand, station records provide relatively slower prediction while respecting the customer privacy. In general, both datasets generate comparable prediction error.

Forecasting solar power generation with application on real-time control of energy system has also been investigated. Since predictions are made on every minute for one minute ahead values, the designed system has to be rapidly responsive. This has been pursued by: first, we have solely relied on past values of solar power data (rather than external data), hence lowering the volume of input data; second, the investigated algorithms are capable of generating predictions in less than a second. The results show that kNN and SVR show lower error.

The dissertation of Mostafa Majidpour is approved.

Alan J. Laub

Lieven Vandenberghe

Jason L. Speyer, Committee Co-Chair

Rajit Gadh, Committee Co-Chair

University of California, Los Angeles

2016

To My Wife, Fahimeh, For Her Unconditional Support and Love,

To My Parents, Ali and Sousan, For Everything They have Done to Get Me to This Day,

To My Sisters, Maryam, Mahdieh, and Soodeh, For All the Laughter, Happiness and Affection,

To My In-Laws, As They were Always there for Me,

To My Aunts, Uncles, Relatives and All My Friends…

And To My unborn bundle of joy! :*

Table of Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

I would like to acknowledge my beloved wife, Fahimeh Fakour, for her unrestricted support during this journey. Her help as an emotional support during the ups and downs of my PhD, and intellectual support in proof reading all my publications is undeniable to making this dissertation happen.

Next, I would like to acknowledge members of my committee: Prof. Rajit Gadh, Prof. Jason L. Speyer, Prof. Lieven Vandenberghe, and Prof. Alan J. Laub. This dissertation would not be possible without their time and attention. My special gratitude goes to Prof. Gadh and Prof. Speyer for their invaluable advice, support, and guidance during my time as a PhD student at the Electrical Engineering department at UCLA.

I also would like to acknowledge Prof. Hemanshu R. Pota for his scientific inputs and our insightful discussions. He was always ready to help when I needed him. I would like to thank my friends and colleagues at UCLA SMERC (Smart Grid Energy Research Center), Dr. Peter Chu, Charlie Qiu, Ching-Yen Chung, Yubo Wang, Bin Wang, Wenbo Shi, and Rui Huang. My sincere gratitude goes to Hamidreza Nazaripouya, not only for his great scientific contributions, but also for his time and dedication as a friend.

Last but not least, I would like to acknowledge all the UCLA staff, especially Ms. Deeona Columbia, Student Affairs Manager at Electrical Engineering department. There is no problem for which she cannot find a solution.

This work has been sponsored in part by grant from the LADWP/DOE fund 20699 & 20686, Smart Grid Regional Demonstration Project.

VITA

M.Sc. in Control Engineering	University of Tehran, Iran	2008
B.Sc. in Electrical Engineering	University of Tehran, Iran	2006

JOURNAL PUBLICATIONS

M. Majidpour, C. Qiu, P. Chu, R. Gadh, H. R. Pota, "Fast Prediction for Sparse Time Series: Demand Forecast of EV Charging Stations for Cell Phone Applications", *Industrial Informatics, IEEE Transactions on* 11, no. 1 (2015): 242-250.

M. Majidpour, C. Qiu, P. Chu, H. R. Pota, R. Gadh, "Forecasting the EV Charging Load based on Customer Profile or Station Measurement?", Elsevier *Applied Energy* 163 (2016): 134-141.

M. Majidpour, C. Qiu, P. Chu, H. R. Pota, R. Gadh, "Treatment of Missing Values in Electric Vehicle Charging Load Prediction", submitted to IEEE Transaction on Smart Grid.

M. Majidpour, H. Nazaripouya, P. Chu, H. R. Pota, R. Gadh, "Fast Univariate Time Series Prediction of Solar Power for Real-Time Control of Energy Storage System at UCLA Campus", to appear in Springer Technology and Economics of Smart Grids and Sustainable Energy.

J. Jetcheva, **M. Majidpour**, W-P. Chen, "Neural Network Model Ensembles for Building-Level Electricity Load Forecasts." Elsevier *Energy and Buildings*, *84*, pp.214-223.

CONFERENCE PUBLICATIONS

M. Majidpour, C. Qiu, C.-Y. Chung, P. Chu, R. Gadh, H. R. Pota, "Fast Demand Forecast of Electric Vehicle Charging Stations for Cell Phone Application", In *PES General Meeting| Conference & Exposition, 2014 IEEE*, pp. 1-5. IEEE, 2014.

M. Majidpour, C. Qiu, P. Chu, R. Gadh, H. R. Pota, "Modified Pattern Sequence-based Forecasting for Electric Vehicle Charging Stations", In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pp. 710-715. IEEE, 2014.

M. Majidpour, C. Qiu, P. Chu, R. Gadh, H. R. Pota, "A Novel Forecasting Algorithm for Electric Vehicle Charging Stations", In *Connected Vehicles and Expo (ICCVE), 2014 International Conference on*, pp. 1035-1040. IEEE, 2014.

M. Majidpour, P. Chu, R. Gadh, H. R. Pota, "Incomplete Data in Smart Grid: Treatment of Missing Values in Electric Vehicle Charging Data", In *Connected Vehicles and Expo (ICCVE), 2014 International Conference on*, pp. 1035-1040. IEEE, 2014.

M. Majidpour, W-P. Chen, "Grid and Customer Constrained Electric Vehicle Charging using Node Sensitivity Approach." In *Connected Vehicles and Expo (ICCVE), 2012 International Conference on*, pp. 304-310. IEEE, 2012.

PATENTS

W-P Chen, and **M. Majidpour**. "Intelligent electric vehicle recharging." U.S. Patent Application 13/436,608, filed March 30, 2012.

J. Jetcheva, and **M. Majidpour**. "Time series forecasting ensemble." U.S. Patent Application 14/050,038, filed October 9, 2013.

HONORS

The Henry Samueli Outstanding Teaching Award for the best Teaching Assistant 2013-2014
NITP Fellowship Award (NIH funded) Fall 2012-Summer 2013
UCLA Graduate Division Fellowship Award Fall 2011

1 Introduction

1.1 Electric Vehicles

Electric Vehicles (EVs) and Plug-in Hybrid Electric Vehicles (PHEVs) will be an important part of the Smart Grid. In this study, PHEVs are considered EVs as they influence the smart grid and power grid in a similar way. The U.S. market share of plug-in electric passenger cars increased from 0.14% in 2011, to 0.37% in 2012, 0.62% in 2013, and 0.75% in 2014 [1]. Although the market share is still less than 1%, it has experienced a rapid growth.

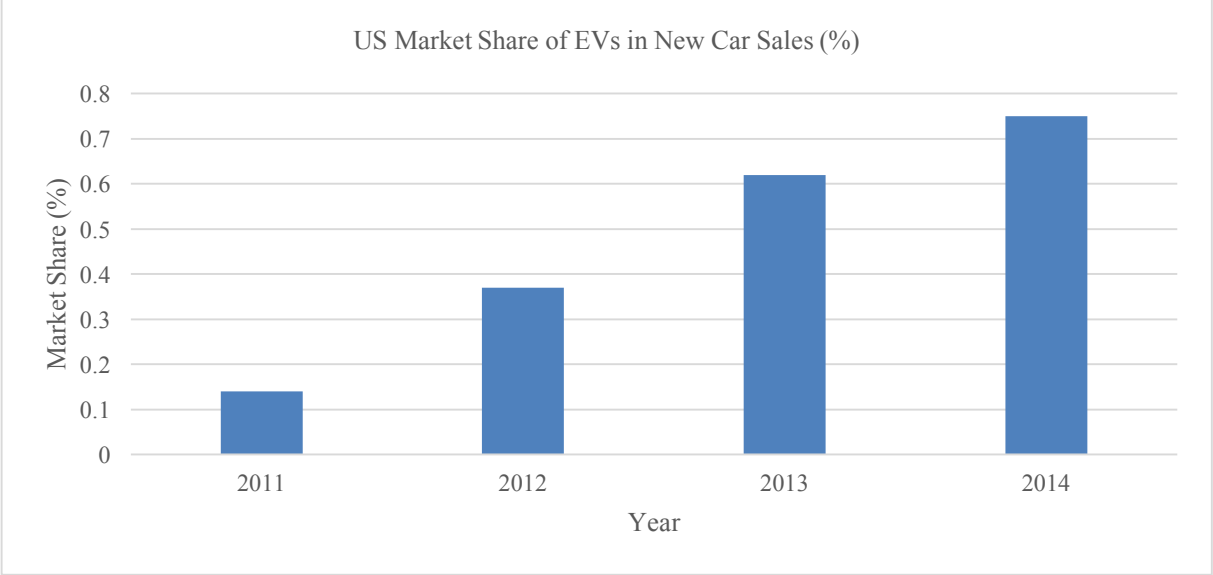


Figure 1-1 US Market Share of EVs in New Car Sales (%)

There are countries with more penetration of EVs in their new car market. For instance, Norway's auto market share for EVs in the similar years of 2011-2014, has been 1.6%, 3.1%, 5.6%, 13.8% [2].

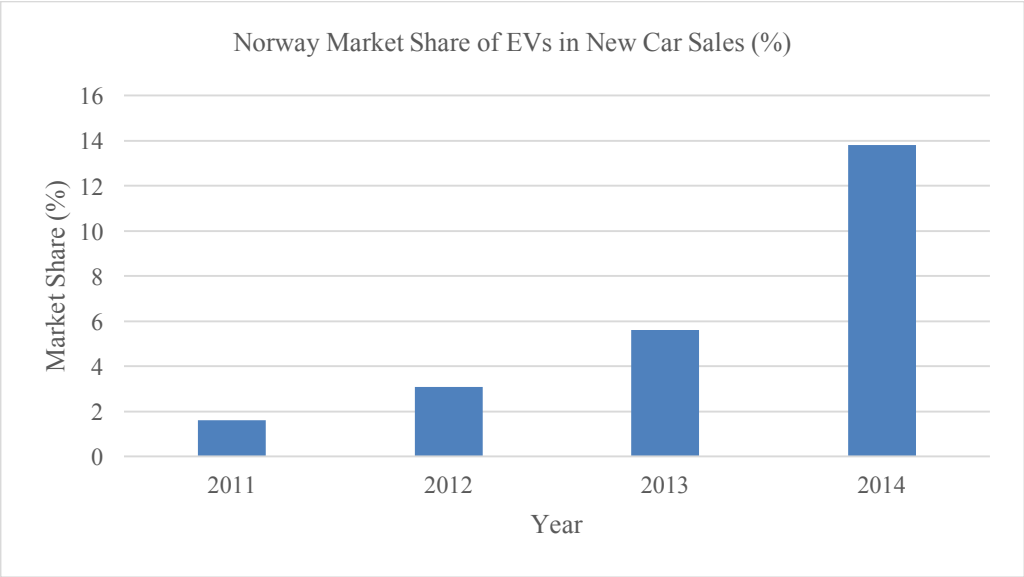


Figure 1-2 Norway's Market Share of EVs in New Car Sales (%)

The common fact in both graphs is that the number of EVs are growing in both countries and this growth is considerably rapid.

1.2 Charging Infrastructure

The big challenge for EVs is charging them within the existing distribution system infrastructure. According to the EV33 rule (33 miles driving range for a single charge [3]), the minimum battery size in EVs varies from 8.6 kWh to 15.2 kWh [4]. Charging batteries with the aforementioned size will take between four and eight hours in a Level 1 household charger (120V, 16 A). As an alternative, EV owners can charge their vehicle at their place of employment, provided that the employer has installed EV chargers in the parking lot. Other than workplace parking lots, cities and private operators have invested in charging stations for public use. As of April 2016, the number of these stations in the US has reached 13,431 stations (with 33,056 outlets) which is roughly an eighth of the number of gas stations in the US [5].

1.3 Forecasting and Smartphones

Electric power has to be consumed simultaneously upon generation. As a result, the EV charging station needs to have a good idea of how many vehicles will need charging at each moment. On the other hand, since a reasonable amount of charging will take tens of minutes, it would be useful for a customer to know when and how much energy is expected to be available at a given charging station in a given time window. Both the customer and moderator will substantially benefit from an algorithm that can predict the power consumption at charging stations.

Other than the availability of outlets in a charging station, there is another complexity: Each charging station has multiple outlets and in view of the upper limit on the available power, the charging is multiplexed among the outlets. Due to this multiplexing, the expected charging time for each outlet can vary depending on the number of EVs being charged simultaneously; therefore, by predicting the available power at each outlet, the charging time for such multiplexed outlets can be computed. In the situation pictured above, EV owners are most likely forced to wait for a considerable amount of time in charging stations to get their EV batteries charged. Therefore, an estimation of how long they have to wait in order to charge their EV can be very beneficial information. With accessibility to online data through smartphones, EV owners can look up the expected waiting time for different charging stations in the area and optimize their time spent at charging stations. Predicting the charging demand for charging stations could be useful for the EV charging station owners too, thus helping them to adjust their inventory in advance. Both of these valuable pieces of information, namely waiting time for charging the battery for EV owners and demand forecast for station owners, rely on forecasting the energy consumption at EV charging stations.

The next chapter discusses prediction algorithms that can run in less than a few seconds, so that EV users can query the system and get the results in a reasonable time on their smartphones. To our

knowledge, our work is the first research that discusses fast prediction of the available energy and/or expected charging finishing time at the charging outlet level for use in a smartphone application.

1.4 Missing Values

In the process of analyzing almost any data, there is the possibility of encountering corrupted data points. These corrupted data points might be missing their value or data might have been altered due to measurement errors and is therefore considered a meaningless outlier. We treat all these values that need to be repaired as missing values.

The process of missing value substitution is called imputation. In chapter four, we will go over some imputation methods and show how imputation affect the quality of prediction. The suggested imputation methods to pair with forecasting algorithms are also discussed.

1.5 Privacy Issues

The most distinct feature of the smart grid is its extensive use of information and communication technologies to improve the efficiency and reliability of the generation and distribution of electricity. A large volume of information is gathered from different meters that might be sufficient to reveal the behavior of different players such as suppliers and consumers. This calls for a privacy concern as pointed out in [6].

Electric Vehicle (EV) charging related data is no exception to privacy issues and has its own problems. One such problem is the large battery size in today's EVs, which may require a relatively large amount of charging time depending on the charging station capabilities. The long charging time may obligate EV owners to charge their EVs in places other than their household, including public charging stations or charging stations at their work place. This implies that not only utilities have access to charging data through household chargers, but also charging station administrators in work places and public stations have access to them. These data, when used for analysis in utility or public charging station

operation or planning, might expose information such as the pattern of entrance and exit times of the customers from charging lots or their home, hence risking their privacy.

One might claim since the charging records are anonymized, there is no threat to customer's privacy. However, anonymizing might not be enough, as in a famous incident, the medical record for the then governor of Massachusetts was easily extracted from anonymous medical records when combined with voter registration rolls [7]. The medical records were anonymous but they had sex, ZIP code, and birth-date of patients. This incident shows that even anonymity is not enough, and anonymous data might still be revealing when combined with other datasets.

In chapter five, we use two different datasets to forecast the energy consumption/availability at the EV charging station: charging record that comes from anonymous customer profiles and station record that comes from measurements (voltage, current, etc.). Either one of them can be used for building a load time series and hence forecasting at the outlet level; however, due to the fact that the charging record is driven from customer profiles (although anonymous), it is a potentially privacy-jeopardizing dataset. We compare the accuracy and speed of the forecasting process using these two types of records. To our knowledge, this type of comparison has not been done in this context.

1.6 Prediction of Solar power Generation for Real-time Control of Energy Storage

Chapter six investigates another application of forecasting in the smart grid context: forecasting solar power generation to control the energy storage system. This time, predictions are a minute ahead predictions performed at each minute; therefore, once again, the whole process has to be fast. The speed of the forecasting is improved in two different ways: only historical solar generation data has been used rather than other external sources such as weather prediction and hence the input size is reduced, also the investigated algorithms are capable of generating the forecasted output in less than a second.

1.7 Data Source

All the simulation and analysis in this dissertation uses real world data acquired from different measurement devices installed and monitored by the Smart Grid Energy Research Center (SMERC) at UCLA. This involves all the EV historical charging related data in the station record and charging record format, as well as solar generation power data.

EV charging stations are installed in UCLA Parking Structure 2, 3, 4, 6, 8, and 9. The solar data comes from solar PV panels located on the UCLA Ackerman Union building.

2 Instance-based Algorithms: Fast Prediction

2.1 Overview

In this chapter, we propose a new smartphone application algorithm which has been implemented for the prediction of energy consumption at Electric Vehicle (EV) Charging Stations at UCLA. For this interactive user application, the total time for accessing the database, processing the data and making the prediction needs to be within a few seconds so that it could be used by a smartphone user.

Other than the speed of generating output for a given input, the process should complete without any offline training, which is useful when the network administrator did not allow for an offline training process for the forecasting algorithm.

Consequently, the only family of Machine Learning algorithms that are relatively fast and postpone the training to when a query has been received (i.e. no offline training) are Lazy Learning algorithms which are also called Instance-based algorithms, or Memory-based algorithm [8].

We first analyze three relatively fast Machine Learning based time series prediction algorithms (from the Instance-based algorithms family) and find that the Nearest Neighbor (NN) algorithm (k Nearest Neighbor with k=1) yields better accuracy. Considering the sparseness of the time series from the charging records, we then discuss a new algorithm based on the proposed Time Weighted Dot Product (TWDP) dissimilarity measure to improve the accuracy and processing time. Two applications have been designed on top of the proposed prediction algorithm: one predicts the expected available energy at the outlet and the other one predicts the expected charging finishing time. The total time, including accessing the database, data processing, and prediction is approximately one second for both applications. The granularity of the prediction is one hour and the horizon is 24 hours.

The work described in this chapter differs from other previous works in the literature in that: 1) we have used just one type of recorded data, Charging Records, which only contains the start and end of the charging transaction and the total amount (a scalar value; not time dependent) of energy received in the charging transaction rather than any geographical or driving habit related data; 2) our predictions is at the charging outlet level (not parking lot or building level); 3) our method is online and fast with the whole process taking about a second. Indeed, the reason that we do not use other sources of data is our speed requirement and the fact that adding more data will slow the process. Other state-of-the-art methods which have been applied on this same data take substantially more time. Specifically, depending on the parameter selection process, Support Vector Regression and Random Forest take respectively 3900 seconds and 306 seconds to produce the results [9].

The rest of this chapter is organized as follows: Section 2.2 provides a brief review of existing literature, Section 2.3 formulates the problem, and Section 2.4 explains the methods that are evaluated in order to compare with the proposed method. Section 2.5 reports and analyzes the result of applying the algorithms on the University of California, Los Angeles (UCLA) parking structures' data. Section 2.6 explains the proposed algorithm, Section 2.7 talks about the implementation of the smartphone applications using the proposed algorithm, and Section 2.8 provides the summary of the chapter.

2.2 Literature Review

Time series prediction (forecasting) methods predict the future of a certain variable given its past history. There is a rich literature on different methods of time series prediction that has evolved from statistics, mathematics, computer science, economics, and engineering [10]-[12]. Probably, the most famous model in time series prediction is the ARIMA model with Box-Jenkins approach [11] which has been widely used in economics and statistics [12] and is considered to be the traditional approach in time

series prediction. Selecting the correct parameters for the ARIMA model is not a trivial task, and, depending on the approach, it might be time consuming.

Prediction algorithms are part of most modern smart grid technologies [13] such as Wind Turbines [14], Photovoltaic Systems [15], and Smart Buildings [16]. Reference [17] combines probability and fuzzy systems concepts to propose a method for predicting the wind speed. A similar problem involving wind power forecasting was addressed with implementing an Artificial Neural Network (ANN) based on the predictions of Global Forecast System. A new optimal type reduction for the interval type-2 Fuzzy logic systems along with ANN has been proposed for load forecasting in [18].

Extensive research has focused on EV charging algorithms and charging station infrastructures [19]-**[22**]; as a next step, EV related research has started to utilize forecasting algorithms [23]-[29]. Some studies apply forecasting algorithms to EV driving habits in order to predict the State of Charge (SOC) of a particular EV and when it needs to be charged [24][25]. Authors in [23] have applied ANN forecasting algorithms to predict the charging profile of the EV within the Building Energy Management System (BEMS) in order to improve the overall energy efficiency of the building. Reference [26] and [27] discuss the prediction of the EV charging profile while taking into account various sources of data, such as vehicle driving and usage data. Authors in [28] and [29] consider forecasting at the charging station level based on the EV user classification and Monte Carlo simulations method. Reference [30] discusses an energy management system for EVs that takes advantage of prediction in different levels through hierarchical Model Predictive Control. A more comprehensive scenario of combining prediction of solar energy (using Radial Basis Function Networks) and optimizing the cost of PHEVs based on factors such as daily distance driven, electricity market price, and load values (by using Genetic Algorithms), has been studied in [15]. Some researchers have facilitated prediction and access to EV related information by aggregating several sources of data [31].

As mentioned above, the aim of this chapter is to focus on relatively fast time series forecasting algorithms in which the whole process of prediction (including preprocessing) takes a reasonably short amount of time for a user that sends prediction-related queries from a smartphone. Machine Learning (ML) based heuristic methods have been shown to provide a good performance in forecasting [32]. Some of these methods such as k-Nearest Neighbor (kNN) and weighted kNN, depending on the number of neighbors, could be pretty fast. However, the selection of parameters for these ML methods still remains a challenge.

For our predictor application, we are interested in the ML based algorithms that have low computation requirements and are relatively faster. A brief introduction of these methods will follow in Section 2.4.

2.3 Problem Statement

The objective is to predict the available energy in the next 24 hours at each charging outlet with a minimum time for processing. Formally, we assume there is some function relating future available energy and the past consumed energy:

$$\hat{E}(t) = f(E(t-1), E(t-2), \dots), \qquad (2\text{-}1)$$

where $E(t)$ is the actual energy consumption at time *t*, $\hat{E}(t)$ is the prediction of the energy consumption at time *t*, and $(E(t-i))$ indicates the past energy consumption at time $(t-i)$.

As is usual in forecasting, we are interested to find an estimation of $E(t)$ according to a particular performance (or error) criteria. For the error measurement, we have chosen Symmetric Mean Absolute Percentage Error (SMAPE). For the day *i*, the SMAPE is defined as:

$$SAMPE(i) = \frac{1}{H} \sum_{t \in day\ i} \frac{\left|E(t) - \hat{E}(t)\right|}{E(t) + \hat{E}(t)} \times 100, \qquad (2\text{-}2)$$

where H is the horizon of prediction in a given day (H=24 in this chapter).

Since there is no access to future data in real life, the last portion of the data (last 10% in this dissertation) is set aside as the test set to evaluate the performance of the algorithm. Thus, our goal is to find an algorithm that minimizes the error between the actual value and its prediction in the test set. Note that the test set is not used in either the parameter selection or training phase.

We use the notation $\widehat{\boldsymbol{E}}(t)$ as the vector of prediction values for the next 24 hours ending at *t* (Fig. 2-1.a). In order to use Machine Learning algorithms, each sample (training, test or validation) is composed of input and output pairs. The output is the energy consumption for the next 24 hours, $\boldsymbol{y}(t) = \boldsymbol{E}(t)$ and the input is the concatenation of the consumption records for up to D prior days, $\boldsymbol{x}(t) = \{\boldsymbol{E}(t-24), \boldsymbol{E}(t-48), \dots, \boldsymbol{E}(t-24D)\}$ (Fig. 2-1.b). This concatenation repeats for all days: if there are N days in the dataset, we will need D days to make the first prediction; therefore, there will be $N - D + 1$ of these input and output $(\boldsymbol{y}(t), \boldsymbol{x}(t))$ pairs (Fig. 2-1.c). The total number of data points is $n = 24N$. Note that $S_{tr} = \{1,2, \dots, N_{tr}\}$ and $S_{ts} = \{N_{tr}+1, \dots, N\}$ are the set of indices for the training and test sets, respectively. Later, in the parameter selection phase, parts of the training set will be treated as the validation set. The different methods used to select the validation set are further explained in the parameter selection section of this chapter.

2.4 Applied Algorithms

Four prediction algorithms have been briefly described here. These algorithms were employed to compare and demonstrate the effectiveness of the proposed approach. A detailed description can be found in [33].

2.4.1 Historical Average

This algorithm is one of the simplest algorithms that is sometimes referred to as the naïve approach and is used only for comparison with other methods. According to this approach, the predicted charging energy in the future is the average of the charging energy consumption in the past. Formally:

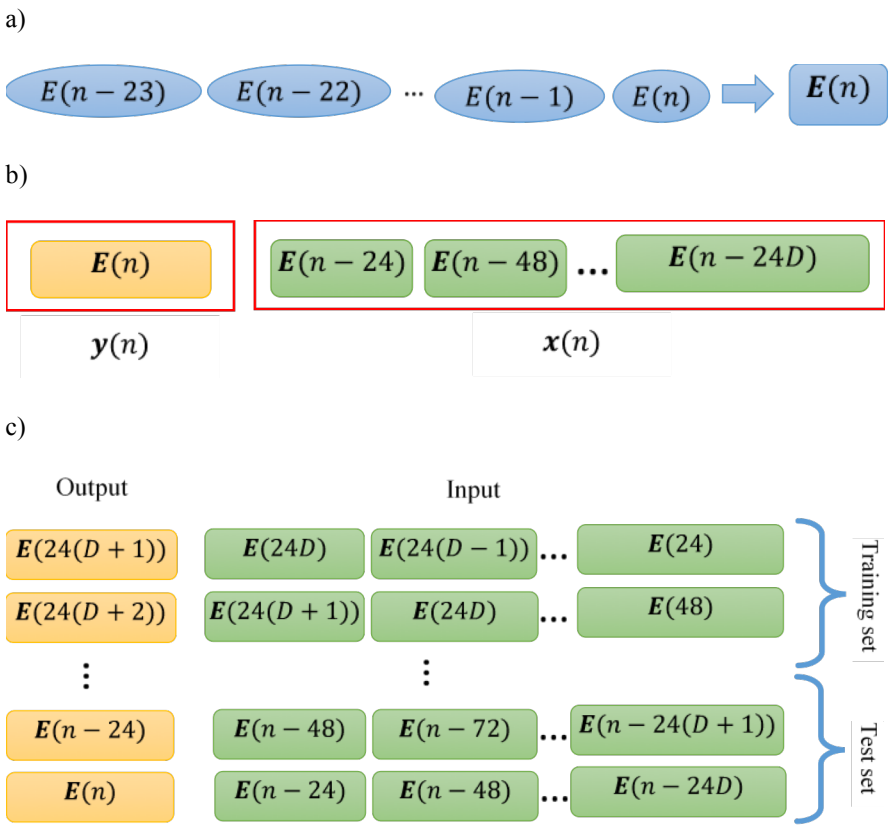


Figure 2-1 a) energy consumption vector (**E**) for 24 hours, b) labeling inputs as $\boldsymbol{x}$ and outputs as y, c) input-output pairs and division of data into training and test sets.

.

$$\hat{E}(t) = \frac{1}{D}\sum_{d=1}^{D} E(t - 24d), \qquad (2\text{-}3)$$

where *D* is the depth of the averaging. For instance, the predicted energy consumption for 3pm on the next day is equal to the average of the energy consumed at 3pm of today, yesterday…, and up to the past *D* days. *D* is a parameter that needs to be selected before the evaluation on the test set.

2.4.2 K-Nearest Neighbor

This algorithm is a well-known algorithm in the Machine Learning community [8]. Based on the k-Nearest Neighbor (kNN) algorithm, each sample (training, test or validation) is composed of input and output pairs. In our application, as seen in Fig. 2-1.b, the output is the predicted energy consumption for the next 24 hours:

$$\boldsymbol{y}(t) = \boldsymbol{E}(t), \qquad (2\text{-}4)$$

and the input is the concatenation of the consumption records for up to *D* previous days:

$$\boldsymbol{x}(t) = \{\boldsymbol{E}(t - 24), \boldsymbol{E}(t - 48), \dots, \boldsymbol{E}(t - 24D)\}. \qquad (2\text{-}5)$$

This concatenation repeats for all days: if there are *N* days in the dataset, there will be *N-D+1* of these input-output pairs (Fig. 2-1.c). The total number of data points is $n = 24N$. Now, in order to find an estimate for $\boldsymbol{y}(t_{s^*})$ where $t_{s^*} \in S_{ts}$ is an instance of test set indices, first, the dissimilarity between $\boldsymbol{x}(t_{s^*})$ and all other $\boldsymbol{x}(t_r)$ that belong to the training set is computed. We have used the Euclidian distance as the measure of dissimilarity here. After determining the k closest $\boldsymbol{x}(t_r)$ to $\boldsymbol{x}(t_{s^*})$, the average of their corresponding $\boldsymbol{y}(t_r)$ is generated as $\boldsymbol{y}(t_{s^*})$. In this algorithm, the parameter *k* needs to be determined. Fig. 2-2 illustrates the algorithm where $dis[j]$ refers to dissimilarity between $\boldsymbol{x}(t_{s^*})$ and $\boldsymbol{x}(j)$.

k-Nearest Neighbor Algorithm

Inputs: $\boldsymbol{x}(t_r), \boldsymbol{y}(t_r), \boldsymbol{x}(t_{s^*}), k$

Output: $\boldsymbol{y}(t_{s^*})$

1. **for** $j \in S_{tr}$
2. $dis[j] = \|\boldsymbol{x}(t_{s^*}) - \boldsymbol{x}(j)\|$
3. **for** $i \in \{1, \ldots, k\}$
4. $idx[i] = \text{index of } i^{th} \text{ smallest}(dis)$
5. $\boldsymbol{y}(t_{s^*}) = \frac{1}{k}\sum_{i\in\{1,\ldots,k\}} \boldsymbol{y}(idx[i])$

Figure 2-2 k-Nearest Neighbor Algorithm.

2.4.3 Weighted k-Nearest Neighbor

Weighted kNN is based on the idea that closer points to the query should contribute more to the output, and this increased contribution of closer points will improve the accuracy of the prediction compared to kNN [34].

This algorithm is very similar to the original kNN algorithm with the exception that instead of averaging the *k* closest training outputs (Fig. 2-2, step 5) their weighted average is used, where the weights are a function of the dissimilarity between input pairs. The weights are defined based on Dudani's weights [34] which give better results compared to other similar weight assigning methods according to both literature [35] and our simulations. Thus, the following two steps will substitute for Step 5 in Fig. 2-2:

$$w_p = \frac{dis[k+1] - dis[p]}{dis[k+1] - dis[1]} \quad p = 1, .., k, \qquad (2\text{-}6)$$

$$\boldsymbol{y}(t_{s^*}) = \frac{1}{\sum_{q \in idx} w_q} \sum_{p \in idx} w_p \boldsymbol{y}(p). \qquad (2\text{-}7)$$

2.4.4 Lazy Learning

Lazy Learning (LL) is a generic term which refers to algorithms that postpone the learning until a query is submitted to the system. In fact, weighted kNN and kNN are simple forms of LL. A version of LL from [36] has been implemented in this chapter. This version of the LL algorithm is similar to the kNN algorithm in principle, with the difference that for each query, the optimum number of neighbors (*k*) is not fixed and is estimated separately. The idea is that for some queries it seems better to look at more neighbors and for some others, fewer neighbors would be enough. For each query, kNN is performed k_{max} times for $k = \{1,2, \dots, k_{max}\}$. Then, based on Leave-One-Out (LOO) cross validation, the error for each *k* is estimated, and the output corresponding to the *k* with a lower LOO cross validation error is selected. We use the PRESS statistic [37] to estimate LOO for each *k*. For a fixed $k \in \{1,2, \dots, k_{max}\}$, suppose $j^* \in \{1,2, \dots, k\}$ indicates the index of the j^{th} closest neighbor to the query $\boldsymbol{x}(t_{s^*})$. For each j^*, we define an error term

$$\boldsymbol{e_k}(j^*) = k \frac{\boldsymbol{y}(j^*) - \boldsymbol{y}(i)}{k-1}, \qquad (2\text{-}8)$$

which gives the LOO cross validation error for the specific *k*:

$$\boldsymbol{e_{LOO}}(k) = \frac{1}{k} \sum_{j^*=1}^{k} (\boldsymbol{e_k}(j^*)^T * (\boldsymbol{e_k}(j^*)). \qquad (2\text{-}9)$$

The *k* with smallest $\boldsymbol{e_{LOO}}$ will be selected as the optimum *k*.

The steps are detailed in Fig. 2-3.

Lazy Learning Algorithm

Inputs: $\boldsymbol{x}(t_r), \boldsymbol{y}(t_r), \boldsymbol{x}(t_{s^*}), k_{max}$

Output: $\boldsymbol{y}(t_{s^*})$

1. **for** $j \in S_{tr}$

2. $dis[j] = \|\boldsymbol{x}(t_{s^*}) - \boldsymbol{x}(j)\|$

3. **for** $i \in \{1, \dots, k_{max}\}$

4. $idx[i] = \text{index of } i^{th} \text{ smallest}(dis)$

5. **for** $k \in \{2, \dots, k_{max}\}$

6. $\boldsymbol{y}(k) = \frac{1}{k}\sum_{i\in\{1,\dots,k\}} \boldsymbol{y}(idx[i])$

7. Calculate $\boldsymbol{e_{LOO}}(k)$ according to Eq. (2-9)

8. $k^* = \arg\left(\min_{k\in\{2,\dots,k_{max}\}} \boldsymbol{e_{LOO}}(k)\right)$

9. $\boldsymbol{y}(t_{s^*}) = \boldsymbol{y}(k^*)$

Figure 2-3 Lazy Learning Algorithm.

2.5 Simulation Setup and Preliminary Results

2.5.1 Data and Preprocessing

The algorithms described above are applied to charging stations located on the UCLA campus. The data used in this chapter were recorded from December 7, 2011 to February 28, 2014; however, for each day, not all outlets were in use. Among all the charging outlets at UCLA, 20 outlets have charging data for more than 60 effective days (days that some nonzero charging has been reported) which have been used in our implementation. The number of effective days for each outlet is reported in Table 2-I.

Data for each outlet comes in a format which is referred to as Charging Records. Each Charging Record contains the beginning and end of the charging time as well as the acquired energy.

The Charging Records are converted to time series by uniformly dividing the acquired energy to the charging interval. For example, if the charging interval is 3 hours and the acquired energy is 3kWh, it is assumed that the EV received 1kWh of energy in each hour. In preprocessing the data, if all the values in an input-output pair $(\boldsymbol{x}(i), \boldsymbol{y}(i))$ are zero or not reported, the pair was removed from the dataset.

There was no normalization or feature extraction from the data. The only implemented preprocessing was to force energy records that were mistakenly recorded as more than the physical maximum of the charging device (E_{max}) and less than zero to the interval of $[0, E_{max}]$.

2.5.2 Parameter Selection

The following parameters need to be determined for our algorithms: Depth, D, for all algorithms, which is the number of previous days considered in the input vector, and the number of neighbors, k, in kNN and weighted kNN.

There are different options for performing parameter selection through validation in time series [38]. One of the popular methods is the k-fold cross validation. In k-fold cross validation, for evaluating a certain set of candidate parameters, the training data is divided into k parts $(P_i, i = 1, \dots, k)$. The algorithm trains on k-1 parts $(P_{-i} = Training\ set - P_i)$ while the error is calculated on the remaining part (P_i), i.e., the validation set. This process iterates k times, and each time one of the parts will be the validation set and the other k-1 parts will make up the training set. The average error of the algorithm on these k iterations will determine the final error for the current selection of parameters. The whole process is repeated for different combinations of parameters and the combination that yields the least final error is selected as the algorithm parameters.

However, [39] discusses how k-fold cross validation in its original form might not be appropriate for time series since it does not respect the order. It proposes a "time series cross validation" in which, for cross validation, we are only allowed to use training data prior to the validation set, i.e., if validation set is P_i, then the training data would be $P_j, j = 1, \dots, i-1$ instead of P_{-i}. Another approach taken in [40] is to use the last portion of the training data for validation and parameter selection purposes. In this approach, the validation set is the last block of training data, P_k, and the data prior to it, $P_j, j = 1, \dots, k-1$, is used for training; hence, it is less computationally expensive since it evaluates parameters on just one block rather than k blocks.

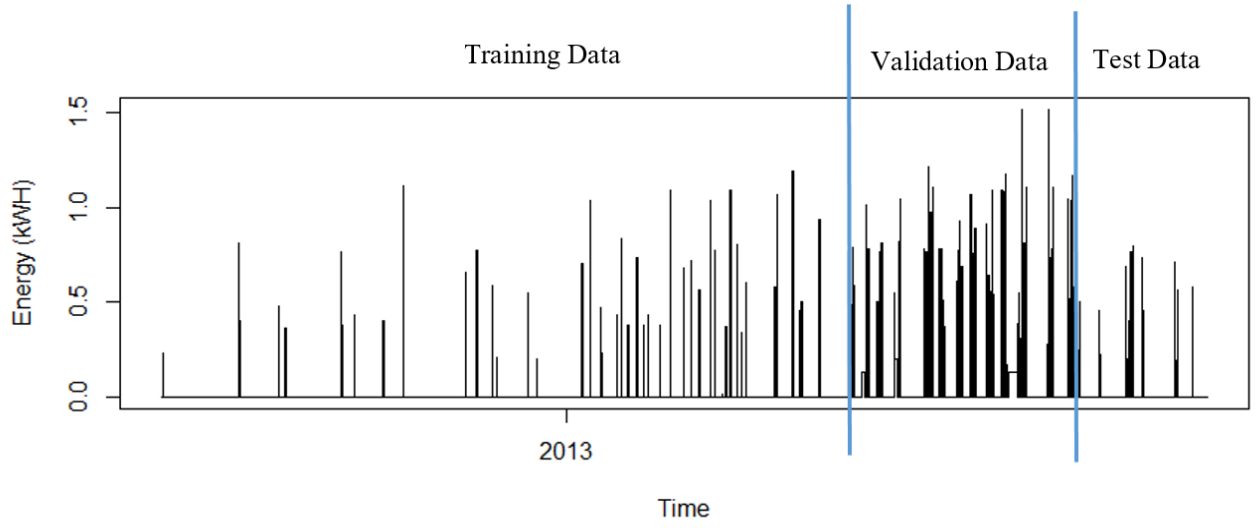


Figure 2-4 Separating validation data from training data in the last block validation method.

We have tried all of the three mentioned validation methods: k-fold cross validation, time series cross validation, and last block validation. We found that they lead to similar behavior in the final results; therefore, in this chapter, we are going to dedicate the last block of training data to validation data which is relatively less computationally expensive, and therefore more appropriate for our fast prediction

application. This method basically partitions the data to non-overlapping training, validation and test datasets as depicted in Fig. 2-4.

The length of the validation block is the next matter of importance. We must determine how much of the last portion of the available data is a good representative of the behavior of all the data for specific parameters. In order to answer this question, we picked different percentages of the last part of the training data for each outlet and each algorithm, and we compared it with the whole dataset in terms of similarity in response to change in parameters.

The smallest percentage of data that could represent the whole training dataset for all algorithms is selected as the final validation set for each outlet. Table 2-I shows the percentage of the data that is used as the validation set for each outlet. The candidate percentages were: 10%, 15%, 20%, 25%, and 30%.

For parameter selection, the depth parameter, D, is varied between 1 to 60 (equal to looking only at yesterday or up to the past two months); in order to make the process faster, a subset of 1 to 60, namely {1, 2, …, 9, 10, 15, 20, …, 60} is employed. The number of neighbors varies between 1 to 5 for kNN and 2 to 5 for weighted kNN (weighted kNN with k=1 is the same as kNN with k=1).

2.5.3 Preliminary Results

Table 2-II and 2-III show the depth (D) and number of neighbors (k) parameters, selected according to the previous section, for each site and each parameter.

After selecting the parameters, the union of the training data and validation data are treated as the new training dataset and used for predicting the first day in the test dataset. For predicting the second day in the test dataset, the union of training dataset, validation dataset, and the first day in the test data is used; similarly, for predicting the ith day in the test set, all the data prior to it (training dataset, validation dataset, and test dataset up to $(i-1)$th day) are employed.

TABLE 2-I Number Of Effective Days For Each Outlet And Percentage Of Data Used For Training, Validation, And Test Sets

No	Outlet	Effective Days	Training Set (%)	Validation Set (%)	Test Set (%)
1	PS2L201LIA4	125	75	15	10
2	PS3L401LIA3	95	75	15	10
3	PS8L201LIA1	105	75	15	10
4	PS8L201LIA3	116	75	15	10
5	PS8L201LIA4	192	80	10	10
6	PS8L202LIIA1	184	75	15	10
7	PS8L202LIIA2	189	80	10	10
8	PS8L202LIIA3	179	80	10	10
9	PS8L203LIA1	154	80	10	10
10	PS8L203LIA2	153	75	15	10
11	PS8L203LIA3	159	80	10	10
12	PS8L203LIA4	158	80	10	10
13	PS9L401LIA1	196	70	20	10
14	PS9L401LIA2	147	65	25	10
15	PS9L401LIA3	328	80	10	10
16	PS9L401LIA5	210	70	20	10
17	PS9L401LIA6	290	70	20	10
18	PS9L601LIA1	227	60	30	10
19	PS9L601LIA3	199	70	20	10
20	PS9L601LIA4	145	75	15	10

TABLE 2-II Selected Depth (D) And Number Of Neighbors (K) Parameter For Each Algorithm Based On Validation Results

No	Outlet	Historical Average	kNN		Weighted kNN		Lazy Learning
		D	***D***	***k***	***D***	***k***	***D***
1	PS2L201LIA4	1	60	1	6	2	15
2	PS3L401LIA3	1	60	1	60	2	60
3	PS8L201LIA1	1	55	1	8	2	60
4	PS8L201LIA3	1	50	1	60	2	60
5	PS8L201LIA4	2	55	1	4	2	20
6	PS8L202LIIA1	1	40	1	1	2	40
7	PS8L202LIIA2	1	60	1	1	2	35
8	PS8L202LIIA3	1	20	1	1	2	20
9	PS8L203LIA1	1	15	1	1	2	9
10	PS8L203LIA2	1	7	1	1	2	9
11	PS8L203LIA3	1	25	1	40	2	30
12	PS8L203LIA4	1	40	1	7	2	7
13	PS9L401LIA1	1	15	1	1	2	2
14	PS9L401LIA2	1	15	1	4	2	15
15	PS9L401LIA3	1	35	1	15	2	25
16	PS9L401LIA5	1	45	1	45	2	45
17	PS9L401LIA6	1	60	1	1	2	60
18	PS9L601LIA1	1	60	1	60	2	60
19	PS9L601LIA3	1	60	1	55	2	55
20	PS9L601LIA4	1	30	1	30	2	50

Interestingly, as can be seen in Table 2-III, for all outlets, the optimum number of neighbors is chosen by cross-validation to be equal to 1 for the kNN algorithm. This shows that, regardless of the optimum number of days prior to the forecast (the *D* parameter), it is always better to look at the most similar event in the past and copy its future energy consumption values as the prediction. Also, the optimum number of neighbors for weighted kNN in all outlets is 2; considering that *k* ranges from 2 to 5 and the kNN algorithm with k=1 shows better performance, one can conclude that k=1 is the optimum parameter.

Table 2-IV shows the average and standard deviation of SMAPE on test days for each algorithm and each outlet.

The historical average has the worst performance by far. Comparing LL results with kNN, it seems that LL was not successful in choosing the best number of neighbors (k) for each query; otherwise, it would have better results compared to kNN (which has a constant k). For better comparison, we have applied non-parametric statistical tests to compare the algorithms according to [38]. First, the null hypothesis of these four algorithms having the same error distribution is rejected by the Friedman test (p-value= 9.80×10^{-11}). Then the Friedman post-hoc test was performed with Hommel's procedure for adjusting p-values. Table 2-IV shows the z-values, as well as the unadjusted and adjusted p-values when considering the kNN algorithm as a control method. As can be seen, adjusted p-values are significantly less than α=0.05, thus kNN has significantly better performance than the other three algorithms.

All simulations were run with RStudio Version 0.97.551 on an Intel Core i-7 CPU at 3.40 GHz with 16 GB RAM.

TABLE 2-III Average And Standard Deviation (In Parentheses) Of SMAPE (%) On Test Days For Each Algorithm

No	Historical Average		kNN		Weighted kNN		Lazy Learning	
1	72.71	(12.50)	15.93	(18.64)	23.20	(19.86)	21.42	(20.12)
2	96.45	(1.89)	3.54	(6.60)	3.64	(6.45)	4.13	(7.20)
3	97.87	(1.39)	16.14	(33.90)	34.68	(45.24)	16.61	(29.41)
4	97.50	(1.11)	48.65	(30.95)	49.71	(33.00)	49.72	(32.47)
5	94.88	(3.97)	2.59	(2.81)	14.04	(17.34)	15.10	(11.85)
6	87.52	(8.43)	28.91	(19.47)	36.04	(23.43)	35.45	(15.06)
7	83.28	(10.87)	35.85	(11.87)	26.98	(15.49)	38.79	(13.16)
8	87.07	(11.61)	28.87	(16.99)	23.17	(15.65)	38.93	(14.04)
9	89.28	(6.84)	23.80	(13.37)	37.05	(17.25)	34.47	(17.48)
10	83.67	(6.75)	32.01	(16.30)	27.93	(15.69)	32.26	(18.30)
11	79.91	(13.64)	33.17	(16.87)	34.06	(16.30)	33.86	(17.95)
12	81.59	(7.72)	29.83	(16.41)	33.68	(16.56)	35.20	(17.45)
13	81.26	(13.72)	21.36	(27.17)	22.39	(21.16)	26.12	(26.89)
14	96.06	(2.44)	13.02	(8.04)	18.92	(8.33)	16.12	(10.20)
15	93.42	(9.83)	9.44	(14.30)	12.36	(18.08)	13.31	(17.56)
16	88.09	(11.33)	7.45	(11.49)	12.72	(12.72)	13.74	(13.61)
17	76.78	(12.14)	4.16	(13.22)	19.12	(16.96)	4.16	(13.33)
18	92.03	(4.80)	11.38	(9.42)	16.41	(10.65)	16.77	(9.89)
19	91.70	(4.52)	10.06	(11.85)	14.75	(12.16)	15.52	(12.04)
20	88.44	(4.96)	8.88	(12.09)	10.27	(14.58)	12.10	(15.60)
Mean	**87.68**	**(7.82)**	**19.08**	**(15.94)**	**23.24**	**(18.16)**	**23.29**	**(17.00)**

TABLE 2-IV Post-Hoc Freidman Test With Hommel's Method Of Adjusting p-values. kNN Is Considered The Control Method

Post-hoc Friedman Test	**Historical Average**	**Weighted kNN**	**Lazy Learning**
z-value	6.919809	2.510727	3.551760
Unadjusted p-value	4.522545×10^{-12}	0.012048	0.000382
Adjusted p-value (Hommel's method)	1.356764×10^{-11}	0.012048	0.000765

2.6 Proposed Algorithm: Time Weighted Dot Product based Nearest Neighbor

In the previous section, we found that the Nearest Neighbor (NN) algorithm has the best performance as the fast predictor algorithm on the examined data. However, there is still a room to improve the results since the results for some of the outlets is not yet satisfactory, e.g. outlet no 3 (Table 2-III).

Looking at the kNN algorithm in Fig. 2-2, it turns out that after selecting k (which, through parameter selection, is equal to 1), there is not much room for modifying the algorithm except step 2 which is the dissimilarity calculation step. The dissimilarity that has been used in this chapter is Euclidean distance. Thus, we are focusing on modifying the dissimilarity measure in step 2 of the kNN algorithm in order to improve the performance.

2.6.1 Dissimilarity measures

In applying the NN algorithm, we used Euclidian distance to measure the dissimilarity between two data points and determine the nearest neighbor of each input query. However, the EV charging data is sparse, meaning that in significant chunks of time, e.g. during the night in the government or official

parking spaces, there is no charging in progress and the consumed power is zero. When the data is sparse, it might be beneficial to employ dissimilarity measures other than Euclidian distance.

One way to define the dissimilarity measure is to use the inverse or negative of a similarity measure. A candidate for similarity measure is the dot product of two vectors since it is zero when the two vectors are orthogonal to each other and is maximum when they are equal. Specifically, the dot product of two signals that have non-zero elements in different indices is equal to zero. This fits well to sparse time series prediction applications, since the similarity between two pieces of signals with different indices of non-zero elements should be the least as less as possible.

2.6.2 Kernelized similarity

Upon using the dot product as similarity measure, one can use kernelized similarity measures to get more flexibility and to compute similarity in higher dimensions [41]. In particular, polynomial kernels are of interest for us since it is the natural extension of the dot product. A polynomial kernel for similarity between x and y is often defined as the following [33]:

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^t \boldsymbol{y} + c)^d, \qquad (2\text{-}10)$$

where $c \geq 0$ is a constant that is trading off the influence of higher-order terms versus lower order ones and d is the degree of the polynomial kernel. Now, we can define the dissimilarity measure based on the polynomial kernel:

$$dis\big(\boldsymbol{x}(t_1), \boldsymbol{x}(t_2)\big) = -K\big(\boldsymbol{x}(t_1), \boldsymbol{x}(t_2)\big). \qquad (2\text{-}11)$$

Another alternative for defining dissimilarity based on kernels is to find the distance of two inputs in the kernel space which can be obtained from the following equation:

$$dis\left(\varphi\big(\boldsymbol{x}(t_1)\big), \varphi\big(\boldsymbol{x}(t_2)\big)\right)^2 = K\big(\boldsymbol{x}(t_1), \boldsymbol{x}(t_1)\big) + K(\boldsymbol{x}(t_2), \boldsymbol{x}(t_2)) - 2K(\boldsymbol{x}(t_1), \boldsymbol{x}(t_2)). \qquad (2\text{-}12)$$

It, however, needs more computation because of self-mapping terms, and it does not improve our results in practice.

2.6.3 Time weighted dissimilarity

Another intuitive modification of the dissimilarity measures could be time weighting; for instance, outputs $\boldsymbol{y}(t_1)$ and $\boldsymbol{y}(t_2)$ are more similar if the recent values of their corresponding inputs $\boldsymbol{x}(t_1)$ and $\boldsymbol{x}(t_2)$ are more similar. In order to weight the recent values for an input that has been defined in (2-5), we have used linear time weighting:

$$\boldsymbol{TW} = [1 + D\ , \dots, 1 + 2\Delta, 1 + \Delta, 1],$$

where $\Delta = 1/(24D - 1)$ and D is the depth of input. Also, we have tried other weighting methods such as *exp(**TW**)* but we did not see improvement in the final results.

Combining all the modifications together, the dissimilarity measure used in kNN algorithm will be substituted with:

$$dis\big(\boldsymbol{x}(t_1), \boldsymbol{x}(t_2)\big) = -(\boldsymbol{x}(t_1)^t\ diag(\boldsymbol{TW})\ \boldsymbol{x}(t_2) + c)^d. \qquad (2\text{-}13)$$

We name this dissimilarity Time Weighted Dot Product (TWDP) dissimilarity. Our modified Nearest Neighbor algorithm which uses this dissimilarity measure would be Time Weighted Dot Product based Nearest Neighbor (TWDP NN). The algorithm is detailed below.

2.6.4 Results

We used the same settings as discussed in Section 2.5, and the only difference in this section is the modification of the dissimilarity measure. Note that this modification adds two more parameters *c* and *d* according to (2-10) which, like other parameters, need to be determined with cross validation. However, in different simulations we did not see much of a difference in the final results of the NN algorithm with the

change in c or d. Therefore, we set $c = 0$ and $d = 1$ and use dot product instead of higher order kernels. The conclusion is that the similarity in higher order terms are not necessary for finding the prediction related similarity in our application, and only the linear terms' contribution is enough for prediction. This is good news computational wise since there is no need to try higher order terms. Combining dot product and time weighting, our dissimilarity measure will be Time Weighted Dot Product (TWDP).

TWDP based Nearest Neighbor (TWDP NN) Algorithm

Inputs: $\boldsymbol{x}(t_r), \boldsymbol{y}(t_r), \boldsymbol{x}(t_{s^*})$

Output: $\boldsymbol{y}(t_{s^*})$

1. **for** $j \in S_{tr}$
2. $\boldsymbol{dis}[j] = -\boldsymbol{x}^t(t_{s^*}).diag(\boldsymbol{TW}).\boldsymbol{x}(j)$
3. $idx = \arg(\min_j(\boldsymbol{dis}[j]))$
4. $\boldsymbol{y}(t_{s^*}) = \boldsymbol{y}(idx)$

Figure 2-5 Nearest Neighbor Algorithm with TWDP similarity measure

Fig. 2-5 shows a sample test day from Outlet 1 and its prediction. The SMAPE for this specific day is 16.93%. The results for all outlets are presented in Table 2-V. According to the Table 2-V, the average SMAPE has been improved in all methods compared with the Euclidean distance case. It's notable that maximum SMAPE for all methods has been decreased to less than 35%.

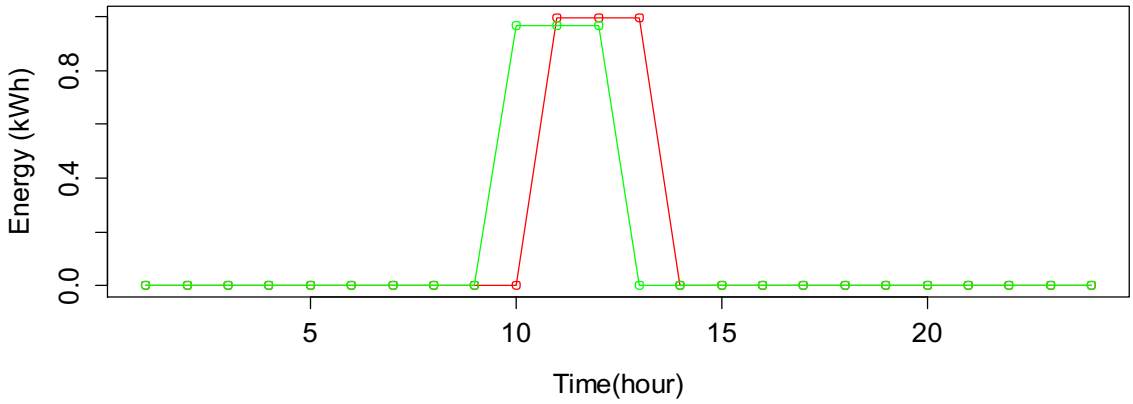


Figure 2-6 Actual enrgy consumption (green) and its prediction with TWDP based NN algorithm (red) for a sample test day in Outlet 1. The SMAPE for this day is 16.93%.

In order to test for statistical significance, we use the same method as in Section 2.5.3: Friedman test to reject the null hypothesis that all these algorithms have similar power, and if the null hypothesis is rejected, then we apply the post-hoc procedures to compare the statistical significance of the power of these algorithms [38]. The p-value is 0.000118 for the Friedman test; therefore, there is a statistical significance between the methods, and we can apply the post-hoc Friedman test and consequently adjust the p-values. The statistics have been displayed in Table 2-VI. Accordingly, the adjusted p-values are less than α=0.05 which shows the statistical significance of the proposed TWDP NN method over weighted kNN and Lazy Learning.

The interesting difference in the pattern of SMAPE errors between results from two different dissimilarity measures has been depicted for NN case in Fig. 2-7. As this figure shows, for outlets that the Euclidean dissimilarity has relatively high errors such as outlet no. 4, the TWDP dissimilarity has relatively low errors and vice versa. The fact that SMAPE error in outlet no. 4 has decreased from 48.65% (NN with Euclidean distance) to 1.04% (NN with dot product dissimilarity) illustrates that TWDP was extremely successful in finding similar points in the time vectors and making the prediction based on that. This phenomenon shows that, depending on the characteristic of the time series in hand, we might need to

change our point of view (from measuring Euclidean dissimilarity to measuring the dissimilarity with TWDP) to be able to see the similar points in the training data to our test query.

TABLE 2-V Average And Standard Deviation (In Parantheses) Of SMAPE (%) On Test Days For Each Algorithm With TWDP Dissimilarity Measure

No	Outlet	kNN (k=1)		Weighted kNN (k=2)		Lazy Learning	
1	PS2L201LIA4	13.60	(16.55)	18.15	(16.58)	19.41	(14.63)
2	PS3L401LIA3	3.37	(6.95)	3.28	(6.04)	4.12	(6.13)
3	PS8L201LIA1	0.62	(2.27)	0.62	(2.39)	0.90	(2.20)
4	PS8L201LIA3	1.04	(3.90)	1.85	(4.43)	1.85	(5.06)
5	PS8L201LIA4	9.24	(9.87)	19.07	(13.06)	19.07	(12.48)
6	PS8L202LIIA1	20.23	(18.92)	23.43	(20.84)	22.00	(17.59)
7	PS8L202LIIA2	30.07	(16.41)	35.00	(9.26)	33.79	(10.05)
8	PS8L202LIIA3	23.83	(18.97)	26.04	(18.83)	26.59	(20.23)
9	PS8L203LIA1	22.03	(13.37)	24.78	(22.20)	25.24	(15.03)
10	PS8L203LIA2	20.84	(14.90)	24.32	(16.81)	23.92	(16.86)
11	PS8L203LIA3	31.20	(11.54)	36.57	(17.52)	35.54	(16.90)
12	PS8L203LIA4	26.53	(15.27)	24.69	(13.35)	26.56	(12.34)
13	PS9L401LIA1	22.85	(21.03)	32.94	(28.15)	31.91	(27.97)
14	PS9L401LIA2	11.93	(7.62)	13.56	(8.61)	13.88	(10.04)
15	PS9L401LIA3	6.40	(13.14)	6.40	(13.35)	6.40	(13.89)
16	PS9L401LIA5	14.49	(12.88)	16.17	(13.84)	15.72	(13.62)
17	PS9L401LIA6	19.09	(17.69)	14.23	(21.82)	14.23	(21.96)
18	PS9L601LIA1	13.69	(12.87)	16.62	(14.67)	16.42	(14.03)
19	PS9L601LIA3	7.58	(11.72)	8.15	(10.89)	8.84	(10.71)
20	PS9L601LIA4	6.44	(11.43)	6.61	(11.16)	6.84	(11.43)
Mean		**15.27**	**(13.24)**	**17.52**	**(14.56)**	**17.59**	**(14.01)**

TABLE 2-VI Post-Hoc Friedman Test With Hommel's Method Of Adjusting p-values. TWDP NN Is Considered The Control Method.

Post-hoc **Friedman Test**	**Weighted kNN** **(k=2)**	**Lazy Learning**
z-value	3.083221	3.794733
Unadjusted p-value	0.002047	0.000147
Adjusted p-value (Hommel's method)	0.002047	0.000295

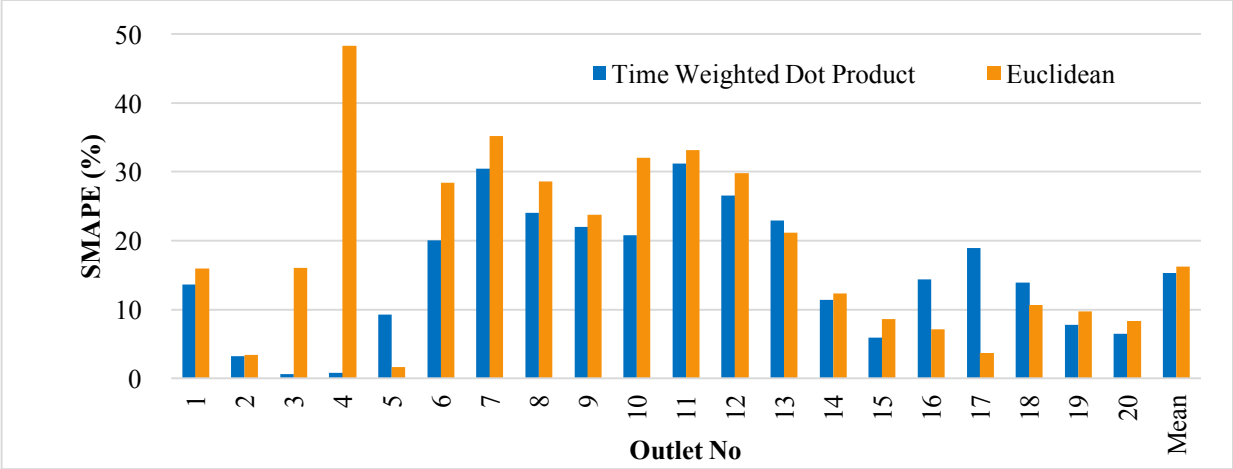


Figure 2-7 Comparing the accuracy of TWDP and Euclidean based dissimilarities in NN.

In fact, the effective characteristic here seems to be the sparseness of the time series. Fig. 2-8 shows the percentage of the sparseness of the time series (number of zero entries divided by the total number of entries in the times series) calculated with the optimum depth for each outlet. Comparing the orange and blue bars shows an interesting relationship: The time series in which TWDP dissimilarity does better are mostly sparser ones.

In order to take advantage of both dissimilarity measures, we implement the best method for each outlet with the best dissimilarity measure. For example, from Fig. 2-8, for outlet no. 4 we use the NN with the TWDP dissimilarity while for outlet no. 17, we use the NN with Euclidean dissimilarity.

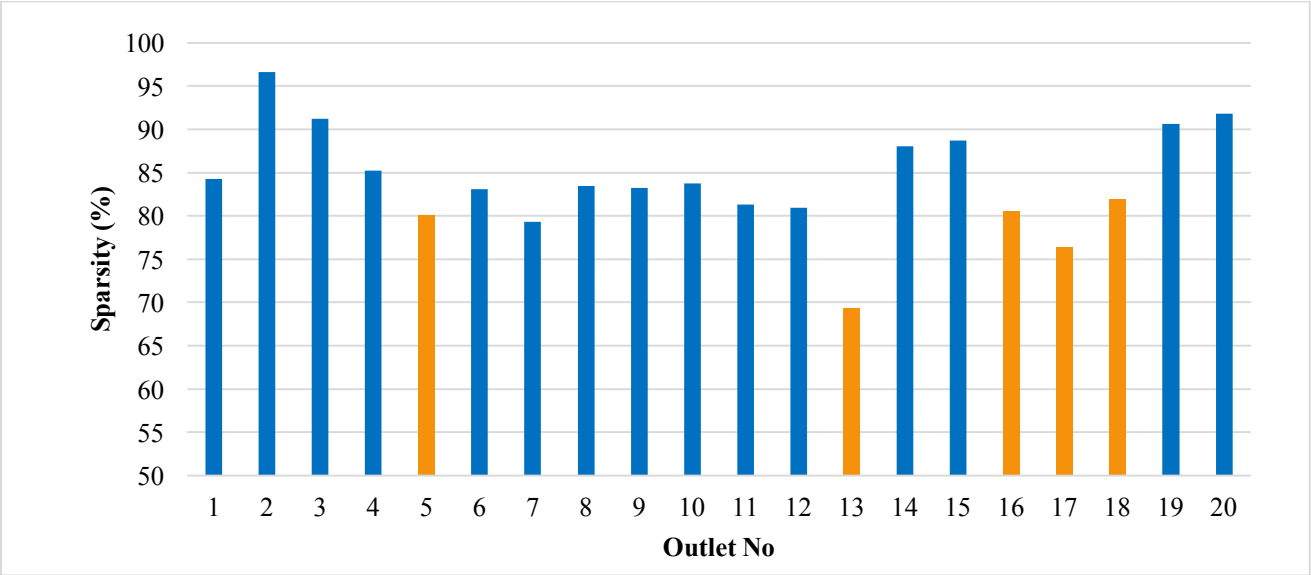


Figure 2-8 Sparseness of the time series of each outlet calculated at the optimum depth. Orange and blue bars are for outlets that Euclidean and TWDP dissimilarity has better accuracy, respectively.

2.7 Smartphone Applications Based on Proposed Algorithm

Based on the results, NN with the TWDP dissimilarity measure has higher accuracy on most of the outlets with sparser time series and NN with the original Euclidean distance dissimilarity performs better on outlets with less sparse time series. Since the number of outlets that have better accuracy with TWDP is in majority, we select it to be implemented in the smartphone applications. We implemented two applications on top of the prediction algorithm:

One application takes the outlet name, required energy (in kWh) needed to charge the vehicle, and the start of charging time as input. The output is the predicted end time of charging.

Another application takes the outlet name, starting time and ending time for the charging as input. The output is the predicted amount of available energy in kWh.

The total time for the algorithm to run is about a second (less than a second for less crowded outlets), which is composed of the time to (a) run in C# under Microsoft Visual Studio 2012, (b) access the database through Microsoft SQL Server 2012, and (c) generate the output. This is well within the

acceptable time for a mobile application response time. The algorithm is now running on the mobile application and is available to UCLA EV owners.

2.8 Summary

In this chapter, we have developed an approach for fast demand prediction of the sparse time series in general, and specifically EV charging outlets.

Data has been recorded from EV owners that charge their vehicle at UCLA campus parking lots. Any EV can use any parking lot on the campus and each outlet might be used by different EVs during the day. The entrance and exit time of each EV to the parking lot and the total acquired energy from each outlet have been recorded as a Charging Record datum. Also, there is a multiplexing of current at each station (station is a group of outlets sharing one source of current); therefore, not all the outlets can be supplied simultaneously and their consumption is dependent on each other.

Since the algorithms have been applied at the outlet level, the complexity of the algorithm is not the function of the total number of EVs or the total number of outlets in the system. For each outlet, the complexity of the algorithm is a function of the size of Charging Record data per day which is a bounded number (it take a few hours to charge an EV, thus the number of EVs getting charged per day per outlet is bounded).

An accurate prediction will enable station owners to provide a realistic time to charge which is essential to customer satisfaction. The more accurate algorithm will help the EV charging outlet owner to predict the energy consumption of his/her facility in the future, thus the station owner can utilize all the capacity of charging stations and therefore obtain more profit. Depending on which factor is more important for a certain EV station owner, s/he can penalize the over prediction of consumption (which translates to empty stations in some hours of the day) or under prediction of the consumption (which translates to a disappointed EV owner whose car was not charged in the predicted time frame) in an

appropriate way. Here, the SMAPE accuracy measure introduced in (2-2) is a symmetric one and does not penalize either over prediction or under prediction. However, algorithms can be easily modified for asymmetric error measurement criteria.

We found that, in general, Nearest Neighbor based predictions generate better predictions than kNN and weighted kNN. We modified the dissimilarity measure in NN from standard Euclidean distance to TWDP (Time Weighted Dot Product) in two ways: 1) changing Euclidean distance to (negative) dot product and 2) adding time weightings to the dissimilarity measure so that recent similar indices in time series get more weight than older ones. Each of these modifications improves the accuracy by itself and their combination improves the results more.

We have implemented this method in the smartphone application system that is used by UCLA EV owners.

3 Non Instance-based Algorithms

3.1 Overview

One assumption that we made in the last chapter was that the system administrator does not allow for using offline computation or storing any training related data in the server, which leads to the selection of Instance-based, or Memory-based algorithms. In this chapter, we want to investigate whether removing this constraint will affect the forecasting performance, i.e. if we can use prediction algorithms other than Instance-based learning methods, what would be the change in the forecasting accuracy?

To remind the reader, the work presented here is different from the other studies in the literature, in the sense that we use just one type of recorded data, called Charging Records, which contains only the start and end time of the charging transaction and the total amount of energy received in the charging transaction (a scalar value; not time dependent). Geographical or driving habit related data was not used in our prediction. Our predictions are at the charging outlet level (not charging station, parking lot or city level) which makes it a more difficult problem as it does not have the aggregated behavior of charging stations, parking lots, or cities.

The rest of this chapter is organized as follows: Section 3.2 reviews the literature, Section 3.3 describes the problem, and Section 3.4 briefly explains the Auto Regressive Integrated Moving Average (ARIMA), Pattern Sequence-based Forecasting (PSF), Support Vector Regression (SVR), and Random Forest (RF) methods. Section 3.5 reports and analyzes the result of applying these algorithms on the University of California, Los Angeles (UCLA) parking structures' EV charging data. Section 3.6 proposes a new algorithm, Modified PSF (MPSF), and analyzes its application to our data, and Section 3.7 summarizes this chapter.

3.2 Literature Review

As mentioned in the Introduction, forecasting EV loads is of recent interest to researchers. In [42], authors have proposed a method to forecast the EV charging load in China based on the Monte Carlo simulation. Reference [43] discusses three daily-load forecasting methods, namely BP and RBF Neural Networks, and GM(1,1) from the Gray model families on one charging station and concluded that each method is more accurate for some specific times during the day. Another method has been proposed in [44] based on Support Vector Regression for forecasting EV charging loads at the city level. Four forecasting methods including Decision Tables, Decision Trees, MPL Neural Networks, and Support Vector Machines (SVM) have been compared in [45] on the US aggregated residential data; however, no conclusion was made regarding the preference of each method over the other ones.

3.3 Problem Statement

The Problem Statement remains the same as the previous chapter. The objective continues to be the prediction of the available energy in the next 24 hours at each charging outlet. However, in this chapter, we look into whether removing the constraint on offline computation will affect the forecasting accuracy.

3.4 Applied Algorithms

The four prediction algorithms used in this chapter have been described here briefly [33].

3.4.1 Support Vector Regression (SVR)

SVR is an expansion of the Support Vector Machines (SVMs) idea, thus moving from classification in SVMs to regression in SVR [46]. The idea is that for classification, there is no need to use all training samples to construct the decision boundaries, rather a few samples, called Support Vectors, are important.

The ε-SV regression algorithm is one of the extensions of SVMs to regression problems. For our problem here, the ε-SV formulation can be defined as following:

$$\widehat{\boldsymbol{E}}(n) = f\big(\boldsymbol{x}(n)\big) = < \boldsymbol{x}(n), \boldsymbol{\omega} > + b, \qquad (3\text{-}1)$$

where $\boldsymbol{x}(n)$ is the input vector (as shown in Fig. 2-1. b). $\boldsymbol{\omega} \in \mathbb{R}^{24D}$ and $b \in \mathcal{R}$ are the solutions to the following optimization problem:

$$\min_{\boldsymbol{\omega}} \quad \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{N_{tr}} \sigma_i \qquad (3\text{-}2)$$

$$subject\ to \quad \begin{cases} |< \boldsymbol{x}(n), \boldsymbol{\omega} > + b - E(i)| \le \varepsilon + \sigma_i \\ \sigma_i \ge 0 \end{cases}.$$

It is not always possible to find $\boldsymbol{\omega}, b$ such that all the $E(i)$ and $\hat{E}(i)$ lie in ε distance of each other. By adding slack variables σ_i and the coefficient C, ω will be as follows [46]:

$$\boldsymbol{\omega} = \sum_{i=1}^{N_{tr}} (\alpha_i - \alpha_i^*)\, \boldsymbol{x}(n), \qquad (3\text{-}3)$$

where α_i, α_i^* are Lagrange multipliers. Note that α_i, α_i^* are nonzero only for training samples that violate the ε proximity constraint. Therefore, $\boldsymbol{\omega}$ only depends on a few training samples that are in fact support vectors.

Thus far, f was a linear function of the training samples, but it can be extended to include nonlinear functions of the training samples through kernel trick [46]:

$$\widehat{\boldsymbol{E}}(n) = f\big(\boldsymbol{x}(n)\big) = \sum_{m=1}^{N_{tr}} (\alpha_i - \alpha_i^*) G(\,\boldsymbol{x}(n), \boldsymbol{x}(i)) + b, \qquad (3\text{-}4)$$

where $G\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big)$ is a kernel function. Examples of popular kernels are polynomial, $G\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big) = \big(< \boldsymbol{x}_i, \boldsymbol{x}_j > + c\big)^p$, hyperbolic tangent, $G\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big) = \tanh\big(a < \boldsymbol{x}_i, \boldsymbol{x}_j > + c\big)$ (for some positive a), and Gaussian radial basis function, $G\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big) = \exp(-\gamma\big\|\boldsymbol{x}_i - \boldsymbol{x}_j\big\|^2)$ for $\gamma > 0$.

In this chapter, we used function 'svm' in the package 'e1071' of the R programming language.

3.4.2 Random Forest (RF)

The Random Forest algorithm is an aggregated version of decision trees [47]. A decision tree tries to learn a set of rules to predict the output value for an unseen input. To this end, at each node, a decision criteria is defined. For instance, in our application, the first rule (corresponding to the root node of the tree) could be that if current consumption is less than 1 kW, the hour ahead consumption will be 1.2 kW. Most probably this rule is not enough for precise prediction, so another rule (corresponding to another node) will be added which creates the new rule: if current consumption is less than 1 kW and last hour's consumption is more than 0.75 kW, then the hour ahead consumption is 1.4 kW. This process will keep going until there is no undecided input variable, the desired precision has been met, or other stopping criteria (such as maximum number of terminal nodes for the tree) has been observed. At each node, selecting the input variable to split is usually done to optimize some criteria. Two common criteria are entropy and Gini impurity [48]. For instance, if we use the entropy criteria, at each node, we will split an input variable that reduces the uncertainty as much as possible. In other words, we will pick a variable to split that gives us the most certain decision.

Decision trees suffer from overfitting; that is, they are high variance models for data. This problem leads to decision trees having poor generalization and hence poor prediction accuracy. One way to solve this problem is using the Random Forest (RF) algorithm. RFs do not use just one tree but many decision trees where the training set of each tree is made of N_{tr} training data points, sampled with replacement from the original training set of size N_r. Also, when finding a variable to split at each node of each tree, instead of all input variables, we look at a random subset of variables. Consequently, Random Forest is a randomized and forested version of the decision trees.

An interesting property of RF is its ability to assign "importance" to each input variable. The idea is based on the average difference in the performance of the RF when some input variable has been selected

to split; intuitively, if selecting that variable to split improves the results further that variable is more important [47]. RF has been shown to be very powerful in classification and regression problems [49], and we include it in this chapter as one of the state of the art methods.

In this chapter, we used function 'randomForest' in the package 'randomForest' of the R programming language.

3.4.3 Auto Regressive Integrated Moving Average (ARIMA)

This model for time series behavior, which is also called Box-Jenkins models [11] (because of Box and Jenkins' fundamental work in this area), models the behavior of future variables as a linear combination of the past values and noise terms. The Auto Regressive (AR) portion models the contribution of the past values of the variable, while the Moving Average (MA) portion models the contribution of noise terms. The Integrated (I) portion models the number of differences needed in order to transform the time series to a stationary time series [50]. The ARIMA model is often specified by ARIMA(p,d,q); p, d and q are the order of the AR, I, and MA terms respectively. Mathematically ARIMA(p,d,q) for variable $X(t)$ can be written as:

$$\left(1 - \sum_{i=1}^{p} \varphi_i L^i\right)(1 - L)^d X(t) = \left(1 - \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon(t), \qquad (3\text{-}5)$$

where L is the lag operator such that $LX(t) = X(t-1)$, $\varepsilon(t)$ is a representative of the noise (or shock or error) contribution, and φ, θ are the coefficients of the model that need to be determined. For our problem, the formula can be rewritten as follows:

$$\hat{E}(t) = (1 - L)^{-d}\left(1 - \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon(t) + \left(\sum_{i=1}^{p} \varphi_i L^i\right) E(t). \qquad (3\text{-}6)$$

Notice that instead of forecasting for the next 24 hours all at once, each hour is forecast based on the previously forecast hour(s) and the past actual values. This process iterates 24 times to complete the prediction of the next 24 hours.

Estimation of φs and θs is usually less challenging and is done by some sort of fitting method like Maximum Likelihood (ML) estimation once the order of the model (i.e., p, d, and q) is determined. Selecting the proper order for the model is usually difficult, and there is no unique method for it. One approach is to use the correlation analysis of the time series and error terms through Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF). There are suggested tips for determining p and q based on ACF and PACF plots, but it does not always give the best model [50]. After selecting the model and estimating the parameters, the fitness of the model to data is examined with criteria such as Akaike Information Criterion (AIC), or Bayesian Information Criterion (BIC). It is worth mentioning that better AIC or BIC does not mean that model has the least SMAPE. Therefore, we have used cross validation to select the best model parameters.

In this study, we used the 'auto.arima' function of the 'forecast' toolbox in the R programming language to select the model and estimate the parameters [51]. Cross validation was used to determine the (p,d,q) triple that reduces the SMAPE the most. The φ and θ parameters were estimated on the training data for the optimum selected model to forecast on the test dataset.

3.4.4 Pattern Sequence-based Forecasting (PSF)

This method was first introduced in [53] and an improved version was published later in [54]. The idea is based on assigning each 24 hour set, i.e., a day, to a cluster and then the forecast is based on the cluster labels rather than actual values in each day. By clustering, the dimension of each day reduces to one

(label of the day) instead of 24 (values for 24 hours). It also adds the robustness by substituting real values (e.g., power consumption) with integer numbers (cluster labels).

The first step in applying this algorithm is to find the clustering method and the optimum number of clusters. In this study, we use the k-means clustering algorithm similar to the one in [53]. In order to determine the optimum number of clusters, k-means clustering is performed on the training dataset for a range of k and the one with the higher validity index is selected. The value of k varies from 2 to the number of unique days in the dataset. The validity index is a clustering statistic that helps to find the optimum number of clusters. The silhouette index, SI, is the validity index used in [53]. If the total number of data points is N_{tr}, for each data point $\boldsymbol{E}(i)$ (i.e., a day in the training dataset) which belongs to cluster C with n_c members, the silhouette value is defined as follows:

$$sil(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \qquad (3\text{-}7)$$

where

$$a(i) = \frac{1}{n_c - 1} \sum_{j \neq i, j \in C} d(i,j), \qquad b(i) = \frac{1}{N_{tr} - n_c} \sum_{j \notin C} d(i,j), \qquad (3\text{-}8)$$

and $d(i,j)$ denotes the dissimilarity or distance between $\boldsymbol{E}(i)$ and $\boldsymbol{E}(j)$. The value of $sil(i)$ can range between -1 and +1 where a negative value ($a(i) > b(i)$) indicates a poor clustering for data point $\boldsymbol{E}(i)$ (the average distance of $\boldsymbol{E}(i)$ from other data points in the same cluster is more than the average distance of $\boldsymbol{E}(i)$ from all other data points belonging to other clusters), and similarly, a positive value indicates a suitable clustering for $\boldsymbol{E}(i)$. The silhouette index is then calculated as the average of $sil(i)$ on all the data points. Evidently, the number of clusters that maximizes the silhouette index is selected as optimum.

In order to improve the finding of the true optimum number of clusters, it has been proposed in [54] to use three different validity indices for clustering, namely silhouette, Dunn, and Davies-Bouldin. The optimum number of clusters is then picked based on a voting mechanism which is very similar to majority voting. However, in our simulations, these three indices rarely agreed (even on the top five number of clusters), so we decided to only use silhouette index as in [53].

After clustering, a 24 hour vector of each day is substituted by a cluster number; therefore, the real valued time series of $\{\boldsymbol{E}(24), \boldsymbol{E}(48), \dots, \boldsymbol{E}(n-24), \boldsymbol{E}(n)\}$ is replaced with an integer valued time series of $\{c(1), c(2), \dots, c(\frac{n-24}{24}), c\left(\frac{n}{24}\right)\}$ where $c\left(\frac{i}{24}\right)$ indicates the cluster label for data point $\boldsymbol{E}(i)$. Now, in order to find $\widehat{\boldsymbol{E}}(t_{s^*})$, where $t_{s^*} \in t_s$ is an instance of the test set indices, a template of cluster labels for the previous days $\{c(t_{s^*}-D), \dots, c(t_{s^*}-2), c(t_{s^*}-1)\}$ is created where similar to kNN, (explained in Section 2.4.2), D is the depth of comparison (which is called window length in [54]). Then, the time series of all the preceding days of t_{s^*}, i.e. $\{c(1), c(2), \dots, c(t_{s^*}-2), c(t_{s^*}-1)\}$, is matched against the aforementioned template. $\widehat{\boldsymbol{E}}(t_{s^*})$ is then equal to the corresponding cluster center of the day immediately following the most recent matched template index. If there is no match for the template with the length of D, the template length is shortened to $D-1$ and algorithm iterates until there is some match in the time series. Similar to kNN, parameter D is determined through cross validation.

The steps of PSF are detailed in Fig. 3-1.

3.5 Simulation Setup and Preliminary Results

3.5.1 Data and Preprocessing

The data and preprocessing for this chapter is similar to the previous chapter. The data used in this chapter were recorded from December 7, 2011 to October 16, 2013; however, not all outlets were in use all days. Among charging outlets at UCLA, 15 outlets have charging data for more than 60 effective days

(days that some nonzero charging has been reported); these outlets have been used in our implementation. The number of effective days for each outlet is reported in Table 3-II.

Pattern Sequence-based Forecasting (PSF) Algorithm

Inputs: $\{\boldsymbol{E}(t_r)\} = \{\boldsymbol{E}(24), \boldsymbol{E}(48), \dots, \boldsymbol{E}(n-24), \boldsymbol{E}(n)\}, D$

Output: $\widehat{\boldsymbol{E}}(t_{s^*})$

1. **for** $k \in \{2,3, \dots, |unique\ \{\boldsymbol{E}(t_r)\}|\}$

2. Perform k-means clustering with $k \rightarrow C_k$ with cluster centers $\boldsymbol{CC}_1, \boldsymbol{CC}_2, \dots, \boldsymbol{CC}_k$

3. Calculate $SI[k]$

4. $k^* = \arg\left(\min_k SI[k]\right)$

5. Replace $\{\boldsymbol{E}(t_r)\}$ with $\{c(\frac{t_r}{24})\}$ according to C_{k^*}

6. **while** $idx = \emptyset$

7. $\boldsymbol{temp}(t_{s^*}) = \{c(t_{s^*} - D), \dots, c(t_{s^*} - 2), c(t_{s^*} - 1)\}$

8. $idx = find(\underset{i \in t_r}{\boldsymbol{temp}}(i) == \boldsymbol{temp}(t_{s^*}))$

9. $D = D - 1$

10. $\widehat{\boldsymbol{E}}(t_{s^*}) = \frac{1}{size(idx)} \sum_{i \in idx} \boldsymbol{CC}_{c(idx[i])}$

Figure 3-1 PSF Algorithm according to [53].

Data for each outlet is in the format that is called Charging Records. Each charging record contains the beginning and end of the charging time as well as the acquired energy. The Charging Records are converted to time series by uniformly dividing the acquired energy to the charging interval; e.g., if

charging interval is 3 hours and the acquired energy is 3kWh, it is assumed that the EV received 1kWh of energy in each hour.

There was no normalization or feature extracting from the data. The only implemented preprocessing was to force energy records that were mistakenly recorded as more than the physical maximum of the charging device (E_{max}) and less than zero to the interval of $[0, E_{max}]$.

3.5.2 Parameter Selection

The following combinatorial parameters need to be determined for our algorithms via cross validation: Depth (D) for SVR, RF and PSF algorithms, and (p,d,q) for ARIMA. Minimum number of terminal nodes, number of trees, and number of variables randomly sampled at each split for the RF algorithm need to be determined as well. Also for the ε-SV algorithm, tradeoff coefficient C, desired ε, type of the kernel, and its corresponding parameters need to be determined via cross validation.

There are some challenges with cross validation when applying machine learning methods to time series forecasting problems [55],[56]. On one hand, in the machine learning community, methods such as k-fold cross validation are popular. In k-fold cross validation, for evaluating a certain set of candidate parameters, the training data is randomly divided into k blocks. Then the algorithm trains on k-1 blocks while the error is calculated on the remaining part, i.e., the validation set. This process iterates for total of k times, where each time one of the blocks will be the validation set and the other k-1 blocks will make up the training set. The average error of the algorithm on these k iterations will determine the final error for the current selection of parameters. The whole process is repeated for different combinations of parameters and the combination that yields the least final error is selected as the algorithm parameters. For k-fold cross validation, 5 or 10 is a typical choice for k [57].

On the other hand, in time series forecasting literature, in order to recognize the temporal order of the data, methods similar to last block validation are more popular. In this method, only the last block of the training data is considered as validation data and the performance of different parameters are reported on it. It is similar to k-fold cross validation, except that the data is not selected randomly and only the last folder is being treated as validation set.

The advantage of k-fold cross-validation is to use all the training data for validation whereas the advantage of last block validation is respecting the temporal order. However, it has been shown that last block validation has a poor estimation of the error on the test set [55]. In order to take advantage of these two methods, we are using a modified form of blocked cross validation method. Blocked cross validation [55] is similar to k-fold cross validation, except that the order of the samples in each block has been preserved. Now, since the first block does not have any preceding values, the modification (similar to [39]) is to select cross validation blocks after a minimum training data (which is used for training the first block). Fig. 3-2 illustrates the modified blocked cross validation with five validation blocks. First, the algorithm is trained on {T1, T2} blocks and validated on V1 block, then it is trained on {T1, T2, V1} blocks and validated on V2 block, and so on. This cross validation method uses the maximum possible data (in comparison with last block validation) while respecting the temporal order of time series data.

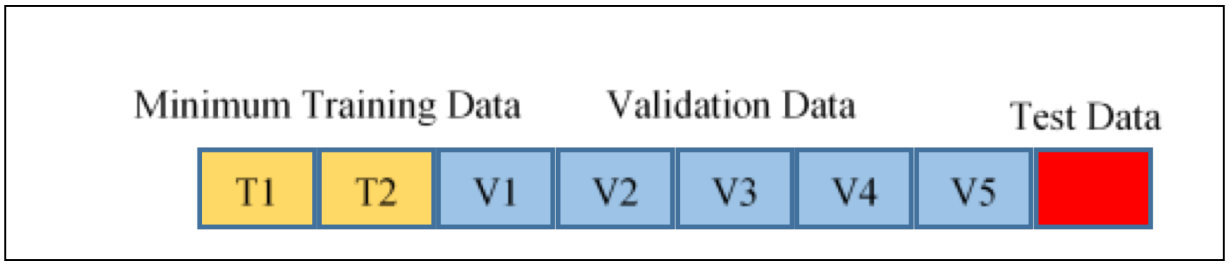


Figure 3-2 Modified blocked cross vailadtion. Training data is divided to minimum training data {T1,T2} and validation data {V1,…V5}. Model is first trained on minimum training data {T1,T2} and evaluated on V1, then it is trained on {T1,T2,V1} and evaluated on V2, up until training on {T1,T2,V1,…,V4} and evaluating on V5.

In cross validation, the depth parameter (D) ranges from 1 to 60 (equal to looking only at yesterday up to the past two months). The kernel type for SVR was selected from linear, radial basis, sigmoid, and polynomial. Other parameters for SVR and their ranges are (the bold parameter is the default in the relevant R package): $\varepsilon \in \{0.01, \mathbf{0.1}\}$ and $C = \{0.1, \mathbf{1}\}$. Similarly for RF, the parameters are the number of trees, $nt \in \{200, \mathbf{500}\}$, number of variables to consider for splitting at each node, $m \in \left\{\frac{1}{6}, \frac{\mathbf{1}}{\mathbf{3}}, \frac{2}{3}\right\} \times D$, and minimum size of terminal nodes, $ns \in \{\mathbf{5}, 10\}$. There are lots of other parameters for SVR and RF for which we used their default value in the relevant R package. Also, in the auto.arima function, the maximum of p and q was set to 5 and 8 respectively. Parameter d was picked by the auto.arima function based on the KPSS test [51].

3.5.3 Preliminary Results

The training set in our simulation was the first 90% of the data which makes the test set the last 10% of the data. The minimum training data was 30% of the training data (30% of 90%=27% of the whole data) and the validation data consists of 70% of the training data. The results were not that sensitive to less or more minimum training data. We used five blocks in cross validation.

Table 3-I and 3-II show different parameters selected through cross validation, for each site and each algorithm.

Table 3-III shows the average and standard deviation of SMAPE on test days for each algorithm and each outlet.

As it was reported in our earlier work [52], the optimum k (number of neighbors) is chosen to be equal to 1 for the kNN algorithm in all outlets. This shows that for our data, regardless of the optimum number of previous days to forecast, when dealing with algorithms based on nearest neighbors, it is always better

to look at the most similar event in the past and copy its future energy consumption values as the prediction.

TABLE 3-I Selected Parameters For Each Algorithm Based On Cross Validation

No	Outlet	SVR (D, Kernel, ε, C)				RF (D, n, m, ns)				ARIMA (p,d,q)	PSF (Depth)	MPSF (Depth)
1	PS3L401LIA3	1	Radial	0.1	1	1	200	2/3	5	5,1,5	20	1
2	PS8L201LIA1	10	Polynomial	0.1	0.1	10	500	2/3	5	1,1,8	53	1
3	PS8L201LIA3	4	Radial	0.1	0.1	3	200	2/3	5	0,1,8	60	1
4	PS8L201LIA4	2	Radial	0.1	0.1	3	200	2/3	5	3,1,8	49	1
5	PS8L202LIIA1	10	Radial	0.1	0.1	10	500	1/6	5	2,1,6	59	1
6	PS8L202LIIA2	15	Radial	0.1	1	10	200	1/3	10	5,1,7	7	1
7	PS8L202LIIA3	10	Radial	0.1	1	10	500	1/3	10	5,0,8	59	1
8	PS9L401LIA1	30	Linear	0.01	1	25	200	1/6	5	4,1,6	5	1
9	PS9L401LIA2	1	Polynomial	0.01	1	1	200	1/6	5	5,1,5	12	1
10	PS9L401LIA3	25	Polynomial	0.01	1	20	500	2/3	5	1,1,2	60	1
11	PS9L401LIA5	5	Radial	0.01	0.1	5	200	2/3	5	4,1,5	33	1
12	PS9L401LIA6	20	Radial	0.1	0.1	15	500	2/3	5	5,1,6	11	2
13	PS9L601LIA1	3	Polynomial	0.1	0.1	1	200	1/6	5	4,1,8	14	1
14	PS9L601LIA3	1	Linear	0.01	1	2	200	1/3	10	4,1,5	17	1
15	PS9L601LIA4	1	Radial	0.1	1	1	500	1/6	5	5,1,8	25	1

TABLE 3-II Selected Number Of Clusters For PSF And MPSF Based On Cross Validation Results And Number Of Effective Days For Each Charging Outlet

No	**PSF** (Clusters)	**MPSF** (Clusters)	**Effective Days**
1	29	28	95
2	20	33	84
3	38	47	97
4	29	24	171
5	98	89	163
6	101	101	168
7	2	90	151
8	44	65	178
9	71	88	126
10	2	130	307
11	86	86	189
12	104	127	269
13	80	74	206
14	65	54	178
15	53	28	124

Another interesting observation is the poor average performance of the ARIMA model compared with NN (Table 3-III, last row). We speculate the reason for this is lots of irregularities in the data along with sparseness (some outlet plugs are not used for a few days and these days do not occur periodically). Consequently, ARIMA (which looks for periodic behaviors) will fail in this situation compared to NN (looks for local similarities and is able to work around this situation).

TABLE 3-III Average SMAPE (%) On Test Days For Each Algorithm

No	SVR	RF	ARIMA	PSF	MPSF
1	23.72	14.76	9.71	42.50	6.30
2	21.22	6.25	5.30	8.35	0.90
3	1.34	0.98	1.13	70.01	0.25
4	66.18	43.96	26.17	15.85	10.72
5	88.24	70.07	39.42	38.01	31.99
6	81.25	74.41	40.39	37.04	26.67
7	89.02	59.32	84.18	81.81	23.33
8	78.23	66.80	40.63	27.82	18.38
9	78.13	79.42	46.65	22.00	13.98
10	97.03	25.00	11.81	96.87	7.76
11	65.34	27.20	8.05	49.07	8.40
12	89.88	35.75	9.34	49.63	23.25
13	91.04	55.36	27.84	35.85	15.51
14	37.95	31.95	15.18	39.50	10.72
15	20.18	19.06	8.57	35.08	8.63
Mean	**61.92**	**40.69**	**24.96**	**43.29**	**13.78**

All simulations were run with RStudio Version 0.98.507 on an Intel Core i-7 CPU at 3.40 GHz with 16 GB RAM.

3.6 Proposed Algorithm: Modified PSF

Based on the results analyzed in the previous section, the Nearest Neighbor (NN) algorithm has higher accuracy on most of the charging outlets and outperforms PSF. However, noticing the way PSF works, it is

very similar to kNN, with the difference that the number of nearest neighbors (k) is not specified beforehand; rather it is equal to the number of template matching incidents. On the other hand, from the results in Table 3-II, it is evident that kNN has the best performance when only one nearest neighbor is considered. Therefore, we propose a modified PSF (MPSF), such that the PSF considers only the most recent match in the previous data and returns the corresponding cluster center as output.

Also, the optimum cluster size seems to be ill conditioned for some of outlets, e.g. outlet 7 and 10. As an example, the silhouette index (as a function of the number of clusters) is depicted in Fig. 3-3 for outlet 10. This ill conditioned optimum number of clusters is expected since the data is sparse; therefore, the number of clusters that maximizes the silhouette index might be only two (one cluster for near zero days and one cluster for non-zero days). Two clusters essentially map the time series to a binary one, and it is not distinguishing enough for forecasting purposes since it cannot code different possible scenarios. We decided to start k (number of clusters in k-means) equal to 10% of distinct days to avoid degenerate clustering.

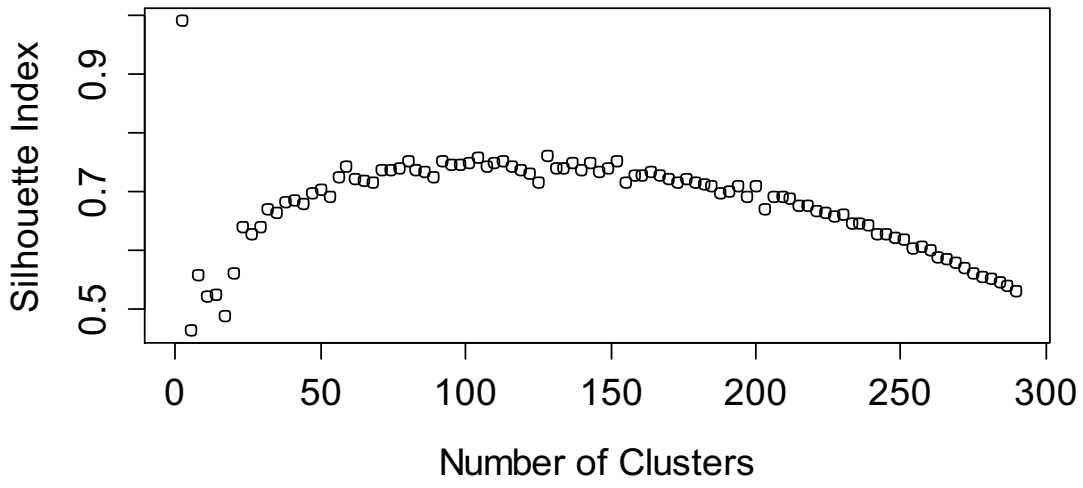


Figure 3-3 Silhouette index for clustering data at outlet 10. At 2 (selected by PSF), there is a cosmetic maximum for the index while 130 (selected by MPSF) seems to be a better maximum; clustering and prediction error wise.

An issue in the original PSF algorithm is that sometimes there is no matched sequence in the past, even when the depth of the template is 1. This means that the last day in the test template, $c(t_{s^*}-1)$, is a member of a cluster with only one member. The algorithm fails in this case. In our modification, we set the output in such a case to be the center of the most common cluster. Our modified algorithm is depicted in Fig. 3-4.

Modified Pattern Sequence-based Forecasting (MPSF) Algorithm

Inputs: $\{\boldsymbol{E}(t_r)\} = \{\boldsymbol{E}(24), \boldsymbol{E}(48), \dots, \boldsymbol{E}(n-24), \boldsymbol{E}(n)\}, D$

Output: $\widehat{\boldsymbol{E}}(t_{s^*})$

1. **for** $k \in \{0.1\times|unique\{\boldsymbol{E}(t_r)\}|, \dots, |unique\{\boldsymbol{E}(t_r)\}|\}$

2. Perform k-means clustering with $k \rightarrow C_k$ with cluster centers $\boldsymbol{CC}_1, \boldsymbol{CC}_2, \dots, \boldsymbol{CC}_k$

3. Calculate $SI[k]$

4. $k^* = \arg\left(\min_k SI[k]\right)$

5. Replace $\{\boldsymbol{E}(t_r)\}$ with $\{c(\frac{t_r}{24})\}$ according to C_{k^*}

6. **while** $(idx = \emptyset \;\&\; D > 0)$

7. $\boldsymbol{temp}(t_{s^*}) = \{c(t_{s^*}-D), \dots, c(t_{s^*}-2), c(t_{s^*}-1)\}$

8. $idx = \max_i(find(\boldsymbol{temp}(i) == \boldsymbol{temp}(t_{s^*})))$

9. $D = D - 1$

10. **if** $D = 0$

11. $idx = \arg\left(\max_i\left(\operatorname{count}_{i\in t_r} c(i)\right)\right)$

12. $\widehat{\boldsymbol{E}}(t_{s^*}) = \boldsymbol{CC}_{c(idx)}$

Figure 3-4 Modified PSF (MPSF) Algorithm.

Fig. 3-5 shows the "importance" of the input variables (discussed in Section 3.4.2) computed through construction of the RF model for outlet 13. According to this figure, for hour ahead forecasting, the current hour and the previous hour are the most important input variables. This trend was also true for all other outlets despite different selected depth: the most recent values are the important ones. This also justifies our selection of the most recent template in our MPSF algorithm.

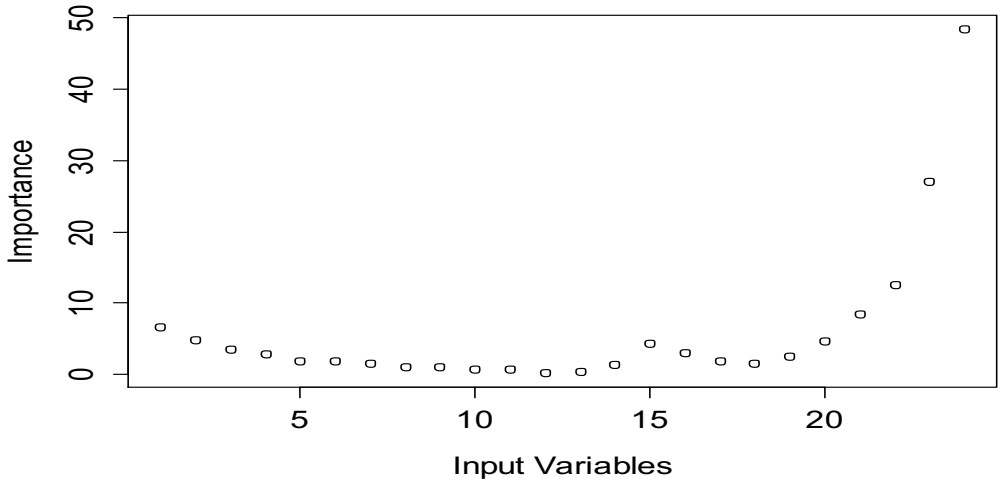


Figure 3-5 Importance of input variables of outlet 13 computed from RF, illustrating that recent values have higher importance. Similarly, in all other outlets the most recent values were selected as important ones.

3.6.1 Statistical Significance of the MPSF results

Now that we have introduced the MPSF algorithm, the next step would be to compare the results from a statistical point of view and see whether there is statistical significance in the performance of MPSF. As pointed out in 2.6.4, we first need to reject the null hypothesis. The null hypothesis shows that, in this case, none of the algorithms is statistically significantly different from the other ones. Using the Friedman non-parametric test, the p-value is equal to 3.033×10^{-8} which is well below the significance level of $\alpha = 0.05$. Thus, there is a statistically significant difference between the algorithms.

In the second step, by selecting MPSF as the control algorithm, we want to check whether the post-hoc p-values are also below the 0.05 significance level.

TABLE 3-IV Post-Hoc Friedman Test With Hommel's Method Of Adjusting p-Values. MPSF Is Considered The Control Method.

Post-hoc Friedman Test	**SVR**	**RF**	**ARIMA**	**PSF**
z-value	6.0044	3.6950	1.9629	3.9259
Unadjusted p-value	1.9199-09	0.0002	0.0496	8.6376e-05
Adjusted p-value (Hommel's method)	7.6796e-09	0.0004	0.0496	0.0002

As Table 3-IV shows, all adjusted p-values are less than the 0.05 significance level; therefore, MPSF not only generates lower errors, but also it does so consistently and with statistically significantly lower error than other algorithms.

3.7 Summary

In this chapter, we analyzed four algorithms namely SVR, RF, ARIMA, and PSF, for the prediction of energy consumption in EV charging stations at the outlet level. We then propose an improved algorithm. Considering the advantage of PSF which is its increased robustness to noise in comparison with NN (kNN with k=1), and the advantage of NN which has the least SMAPE for our data, we proposed Modified PSF (MPSF) where only the most recent match in the previous data is used to generate the prediction. Results show that the modification has improved the performance.

Although MPSF has better prediction accuracy than NN, since NN is faster than MPSF, it would be the best choice for the smartphone application. It is important to note, however, that NN has lower error

than the other four offline algorithms (SVR, RF, ARIMA, and PSF) and is still a fairly good choice. Therefore, we have not lost a great deal by excluding non-Instance-based algorithms.

4 Missing value and imputations

4.1 Overview

Many public places are being equipped with sensors and meters to monitor and record data throughout the day. This huge amount of recorded data calls for a special analysis known as Big Data. Among the issues with analyzing Big Data, such as the computational complexity of the algorithms and scaling the well behaved algorithms for a larger number of data points, there is the problem of missing values in the data.

Missing values might have different causes. Sometimes, data is not reported due to sensor outage. At other times, the value that is reported is far outside the expected range, therefore rendering the reported value unsuitable. In these scenarios, the value is considered a missing value.

There are a variety of methods for compensating the missing data values, commonly referred to as imputation methods [58]. One might wonder which imputation method is more suitable for a specific application or essentially whether there is a practical difference between imputation methods. Another factor in deciding on an imputation method is the relative amount of missing values in the data. In this chapter, we go over several imputation methods and their effect on a number of prediction algorithms. It is important to note that, in the case of time series, some imputation methods such as “case deletion” cannot be applied since they will change the relative order of events and make the time series lose its ordinal properties such as periodicity.

One common approach when comparing different algorithms is to look at the performance of the algorithms on one specific problem (e.g. prediction performance of k-NN and Random Forest algorithms on an electricity load time series) and claiming that the algorithm with higher performance is superior. Another popular approach is to look at the average performance of algorithms on a couple of problems

(e.g. prediction performance of k-NN method and Random Forest on an electricity load time series and weather time series) and claiming that the one with the higher average performance is superior. When there is no analytical proof for the ranking of the algorithms, statistical tests should be used in order to claim a more general superiority of an algorithm. Also, a demonstration of the algorithm's performance for a variety of prediction problems (i.e. weather prediction, load prediction, etc.) will result in stronger claims [38]. In this chapter, we use statistical tests on 13 performance samples (for 13 outlet time series) to compare our algorithm and imputations pairs and derive conclusions. We analyze the effect of missing values on the prediction of the Electric Vehicle (EV) loads in the University of California, Los Angeles (UCLA) parking structures as a sample problem. Forecasting algorithms that we have applied in this chapter are kNN, TWDP NN, MPSF, SVR, and RF that all have been discussed in Sections 2.4 or 3.4.

The work described in this chapter differs from other previous works in the literature in that: 1) most of the other works do not consider or report their missing value compensation mechanism; 2) our conclusions are based on non-parametric statistical tests on several similar problems rather than just looking at the average performance of the methods/algorithms; 3) our predictions are at the charging outlet level (not parking lot or building level).

The rest of this chapter is organized as follows: Section 4.2 provides a brief review of the existing literature, while Section 4.3 presents a statement and description of the problem we seek to investigate. Section 4.4 explains the imputation methods applied on time series data to compensate for missing values. Section 4.5 reports the results of applying the imputation methods and prediction algorithms on the UCLA parking structure data. Section 4.6 analyzes the results with non-parametric statistical tests to investigate any statistically meaningful difference between imputation methods and prediction algorithm results. Finally, Section 4.7 provides the summary.

4.2 Literature Review

Although missing data and imputation methods have been discussed extensively in statistics [59] and subsequently in social and human sciences [60], it has not been broadly used for engineering applications. This is understandable when considering that data processing was not essential to most engineering disciplines until the recent wave of Big Data and the increasing amount of meter and sensor measurements.

However, there are a few studies discussing the missing data treatment issue in the smart grid context. In [61], Chen, et al. use B-Spline smoothing and Kernel smoothing to impute the missing or corrupted data with suitable values on electricity load data. Authors in [62] have proposed a linearized load flow model based on smart meter data with incomplete or inaccurate values. Then, they used this model to obtain voltage sensitivities of the grid and showed that these sensitivities are close to accurate voltage sensitivities with complete data. Data acquired from a laboratory grid was used in this study. In another study, [63], a weighted average combination of the available data was used to impute the missing data for better operation of the grid. The weighting mechanism is a simple one which gives more weight to the samples that are closer to the missing value. The imputation method along with other operational mechanisms has been applied on the Georgia Tech distribution system. However, the performance of the imputation method has not been reported separately. A rather complicated imputation method, namely Distributed Principal Components Pursuit method, has been developed in [64], where authors have taken advantage of the low dimensionality of spatiotemporal load profiles and the sparsity of missing or corrupted data. The algorithm has been evaluated on the load curve of five buildings [64].

Time series prediction has been used in the smart grid context in the form of load forecasting, price forecasting, wind generation forecasting, solar energy forecasting, etc. Saez, et al. compare fuzzy prediction interval models for forecasting load, solar and wind generation [65]. They compare their proposed algorithm with linear regression models only in the load forecasting case. Wan, et al. have used a

two stage algorithm to perform probabilistic forecasting of energy prices [66]. In the first stage, extreme learning machine (ELM) is applied to estimate the point forecasts. In the second stage, they use the maximum likelihood method to estimate the noise variance, thus obtaining a probabilistic forecast. However, this proposed algorithm has been validated on only one region in the Australia Electricity Market. A forecasting algorithm based on an ensemble of Gaussian Processes and Neural Networks has been proposed in [67] for short-term wind power forecasting. The average error of this forecasting algorithm has been reported on seven wind farms. Another forecasting model for electricity load has been developed in [68]. It is a semi-parametric additive model and results have been reported on more than 2200 substations in the French distribution network; however, the comparison between results is in the format of looking at the relative errors without a rigorous statistical analysis. Another research work is [69] where authors have combined Empirical Mode Decomposition and Support Vector Regression (SVR) to perform long-term load forecast on a publicly available competition dataset and an office building dataset. The means of comparison was by simply comparing the errors. In [70], Motamedi, et al. have proposed a forecasting engine that combines both price and demand prediction with Data Association Mining algorithms. The simulation results are based on the electricity market data of Australia and New England where Mean Absolute Percentage Error was used as a comparison measurement. Amajdy, et al. [71] propose a two stage algorithm based on feature extraction (with Neural Networks) and forecasting engine (with evolutionary algorithm) that has been tested on one dataset from the University of Calgary campus. The measure of the performance is the weekly mean error.

All the works reported above have validated their algorithms by applying it on one problem (dataset) or have used simple comparison methods such as average performance on different problems (datasets) to show the improved performance with their approach. In this chapter, we apply statistical analysis on several datasets to make conclusions on the superiority of a given method.

4.3 Problem Statement

The Problem Statement remains the same as the previous chapters. The objective continues to be the prediction of the available energy in the next 24 hours at each charging outlet. However, in this chapter, we look into how missing data affects the accuracy of the predictions in the previous chapters, and whether the choice of treatment (or substitution) of missing values can significantly influence our predictions.

Similar to previous chapters, we need to select a means of comparing performance of predictions. However, instead of SMAPE, we have chosen Mean Absolute Error (MAE) because of its simplicity and the fact that we are interested in the method with lowest absolute error (not percentage error as in SMAPE) which allows for a more straightforward comparison. For day *i*, the MAE is defined as:

$$MAE(i) = \frac{1}{H} \sum_{t \in day\ i} \left| E(t) - \hat{E}(t) \right|, \qquad (4\text{-}1)$$

where H is the horizon of prediction in a given day (H=24 in this paper), $E(t)$ is the actual energy consumption at time *t*, and $\hat{E}(t)$ is the prediction of the energy consumption at time *t*.

4.4 Imputation Methods

The process of providing the best guess for a missing value is called imputation. Multivariate imputation methods such as "Maximum Likelihood" are applicable to datasets with more than one variable observed at a time (multivariate). We will be using four imputation methods which are applicable to both multivariate and univariate time series and two imputation methods that are only applicable to multivariate time series.

Our imputation implementations are briefly explained below. A more detailed discussion can be found in [58].

4.4.1 Constant (Zero) Imputation

In this imputation, a single constant value is substituted for all missing values. While this number is arbitrary, we have chosen it to be zero in order to preserve the sparsity of our time series. In this method, missing power data, due to the missing voltage, current, or power factor, is substituted with zero.

4.4.2 Mean Imputation

This imputation is an extension of Constant imputation where the constant number is the mean of the available data points. While an advantage of inserting the mean value is that it does not change the mean value of the data, one disadvantage of it is that the imputed value (mean) might not be any of the observed values; for instance, in our case, the received power by the EV can be zero, the maximum power, or half of it, etc. depending on the number of EVs being charged at that specific station. Thus, because of the discrete nature of the values, the mean value might not exist as an observed value.

4.4.3 Median Imputation

Like Mean imputation, this imputation is an extension of Constant imputation; the constant number that is used in this imputation is the median of the available data points. The advantage of this imputation is that median is always one of the observed values of the data, making median imputation a good candidate for our study.

4.4.4 Last Observation Carried Forward (LOCF)

This imputation is one of the so called “hot-deck” imputations. In hot-deck imputation, a missing value is copied from another record. Hot-deck imputation methods were popular in the past and are still practiced [58]. A proper hot-deck imputation method for time series is the LOCF method since, out of all

possible values in the available data, it replaces the missing value with the last observed one. For instance, if the voltage is missing, the missing value is replaced with the last observed voltage.

4.4.5 Maximum Likelihood Imputation

The idea behind Maximum Likelihood (ML) is to impute the unobserved values by considering the other variables at that moment. For instance, the voltage value might be missing but the current value is available. Then, the missing value of the voltage can be replaced by considering the corresponding voltage for similar current values. This idea is usually implemented through the Expectation Maximization (EM) algorithm.

4.4.6 Multiple Imputation

A disadvantage of the aforementioned methods is that they underestimate the error by adding more data points without adding more information. Consider the standard error in the average of data points (sample mean): by adding more data points, the sample mean error (which is inversely proportional to the square root of number of data points) reduces, without adding any new information to the available dataset.

The Multiple imputation method is similar to ML with the difference that each missing value is imputed by adding an error term so that it estimates the actual values more accurately. This process iterates a few times (usually five) and the final imputed value is the average of these iterations [58].

4.5 Simulations

4.5.1 Data and Preprocessing

The algorithms described above are applied to charging stations located on the UCLA campus. The data used in this chapter were recorded from December 7, 2011 to February 28, 2014; however, not all

outlets were in use on all days. Among charging outlets at UCLA, 20 outlets have charging data for more than 60 effective days (days that some nonzero charging has been reported); among these 20 outlets, 13 of them have missing data which makes them suitable for the current study. By missing data, as pointed out earlier, we mean either unreported data or the data that is not in the nominal expected range and hence non usable.

Data for each outlet is in a format referred to as Station Records. Each station record contains data including voltage, current, and power factor of the charging outlet in three to five minute intervals. In order to obtain the real power at each hour to form the time series, we multiply the voltage, current, and power factor. Since we are interested in forecasting on an hourly basis, we up-sample the time series to form a power time series of one hour granularity for each outlet. For the sake of investigating the effect of the missing values, they have not been removed in the up-sampling process.

Before applying the imputation methods, missing values need to be identified. For voltage, current, and power factor, in addition to unreported and negative reported values, the reported values that were more than voltage maximum, current maximum and maximum power factor (one) were identified as missing values. The total percentage of missing values in the hourly time series for each outlet has been depicted in the Fig. 4-1.

4.5.2 Parameter Selection

The following discrete parameters need to be determined for our algorithms via cross validation. Depth (D) must be determined for all algorithms. For the RF algorithm, we determine the minimum number of terminal nodes, number of trees, and number of variables randomly sampled at each split. Also for the ε-SV algorithm, the tradeoff coefficient C, desired ε, kernel type, and its corresponding parameters need to be determined via cross validation.

There are some challenges with cross validation when applying machine learning methods to time series forecast problems which have been pointed out in [72], and a modified form of the blocked cross-validation has been chosen since it seems to possess advantages mentioned in both machine learning and time series forecasting literature.

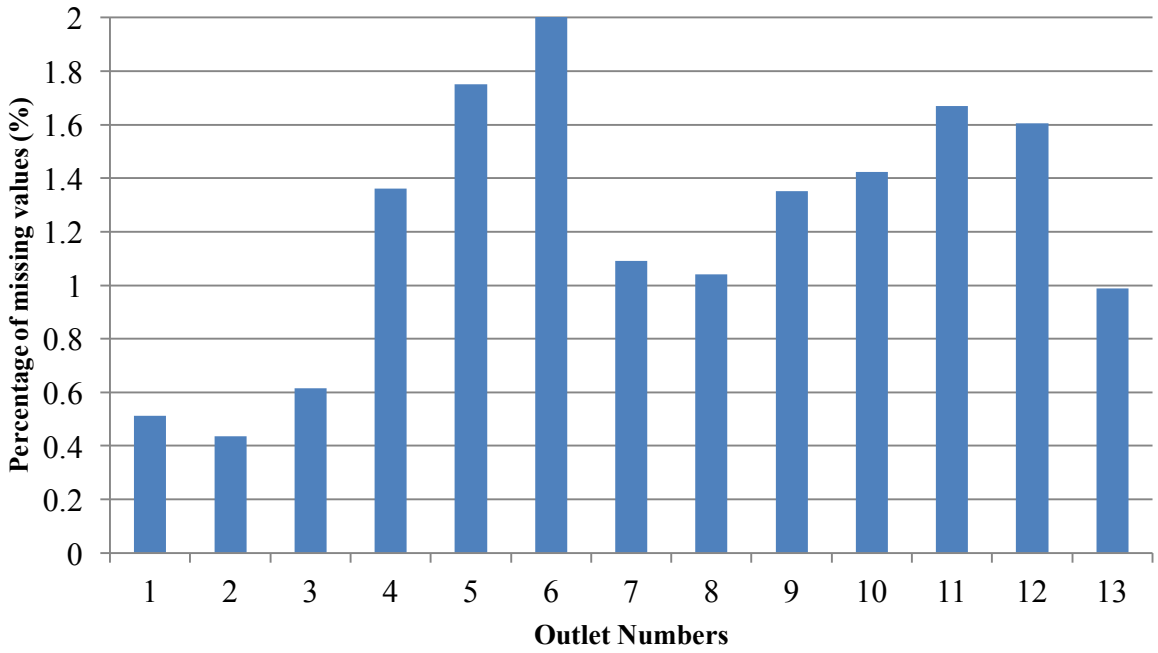


Figure 4-1 Percentage of missing values in the hourly time sereis for each outlet

4.5.3 Simulation Results

All simulations were run with RStudio version 0.98.1091 on an Intel Core i-7 CPU at 3.40 GHz with 16 GB RAM. RStudio is running under R version 3.1.2.

The training set in our simulations was the first 90% of the data which makes the test set the last 10% of the data. We used five blocks in the cross validation procedure. Also, half of the training data was treated as minimum training data.

After selecting the parameters, the union of the training data and validation data are treated as the new training dataset and used for predicting the first day in the test dataset. For predicting the second day in the test dataset, the union of training dataset, validation dataset, and the first day in the test data is used; similarly, for predicting the *i*th day in the test set, all the data prior to it (training dataset, validation dataset, and test dataset up to *(i-1)*th day) is employed.

Fig. 4-2 shows part of a day (from 11:00 to 17:00) in the test set with a missing value in voltage at 14:00 and different imputation methods applied to it. The Zero imputation replaces the voltage value with zero and is not visible in the figure (green triangle line).

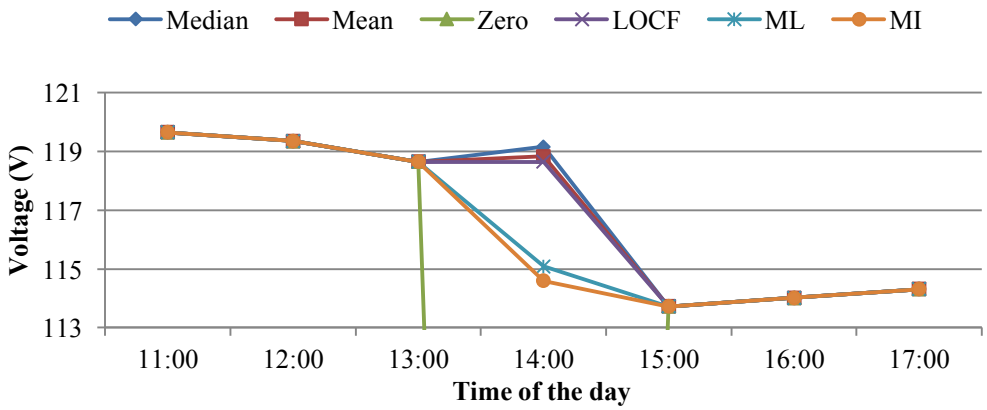
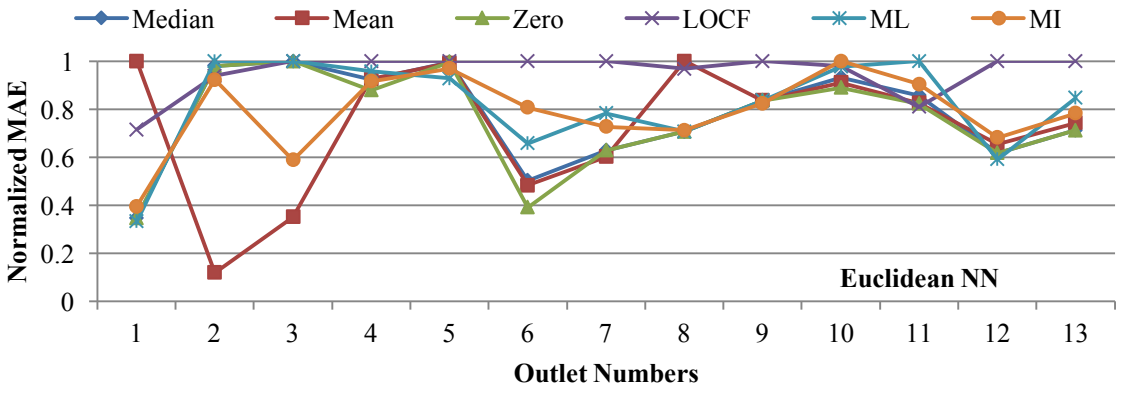
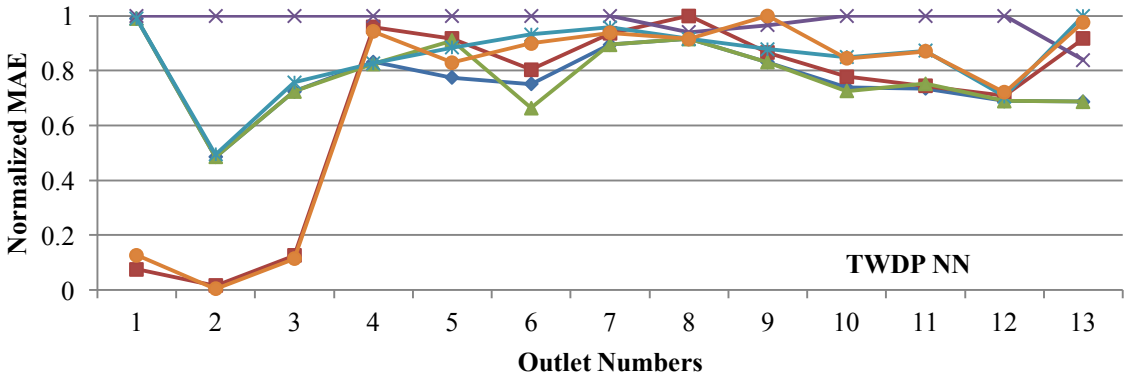
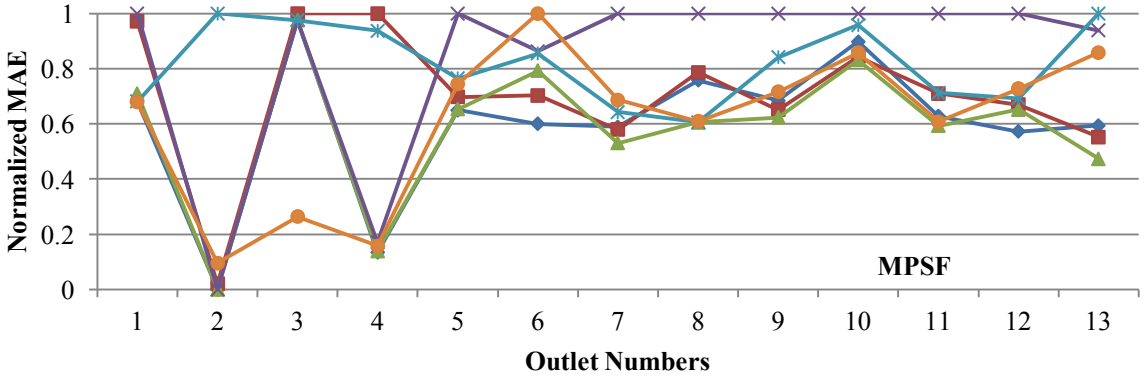


Figure 4-2 Illustrating different imputation methods. Each imputation method substitutes a different value for missing value of voltage at 14:00.

According to the results, the performance of the imputation methods is different for each prediction algorithm. In order to conveniently compare the errors between imputation methods, normalized MAEs have been employed. Normalized MAE values for each specific outlet are acquired by dividing MAEs by the maximum MAE for that outlet. Fig. 4-3 shows the average normalized MAE of each algorithm on test days for each imputation method and outlet.


Median
Mean
Zero
LOCF
ML
MI
Normalized MAE
Euclidean NN
Outlet Numbers


Normalized MAE
TWDP NN
Outlet Numbers


Normalized MAE
MPSF
Outlet Numbers

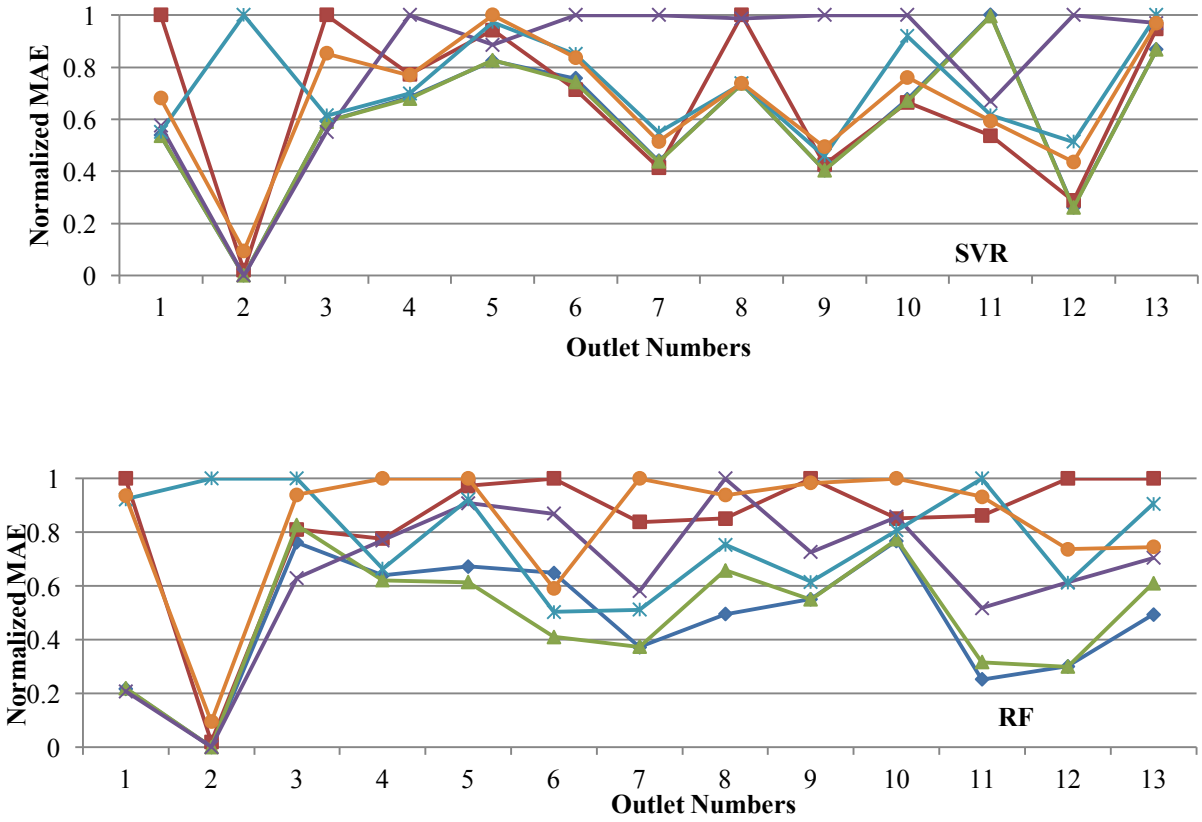


Figure 4-3 Average normalized Mean Absolute Error (MAE) on test days for each imputation method and outlet for Euclidean NN, TWDP NN, MPSF, SVR, and RF algorithms.

4.6 Analysis

In this section, we want to identify the best imputation method for each algorithm (or for all algorithms) that yields a minimum MAE for the forecasting algorithm. In this section, we investigate whether we can recommend an imputation method to pair along with each algorithm. In order to make such suggestions, it is not enough to compare the error average for each combination of imputation method and forecasting algorithm, rather it needs to be based on statistical guarantees that also provide some quantitative measure of confidence.

Our approach here is inspired from [38] and has three steps of non-parametric statistical tests. First, we need to reject the null hypothesis, which is to make sure there is a meaningful difference between results. Second, we will perform a comparison for each pair of imputation methods per forecasting

algorithm to find potential statistically significantly different imputation methods. Third, we will check if the candidate methods are indeed statistically significantly different by choosing them as control methods against other imputation methods.

In the last subsection, we will investigate whether the imputation results are independent from algorithms, i.e., there are some imputation methods that are recommended or should be avoided regardless of the applied prediction algorithm.

4.6.1 Rejecting the Null Hypothesis

In order to see whether there is a statistically meaningful difference between imputation results for each prediction algorithm, we apply the Friedman test to reject the null hypothesis. Here the null hypothesis is that there is no difference between imputation methods for any algorithms. Table 4-I shows the p-values from the Friedman test for each algorithm for the results depicted in Fig. 4-3.

TABLE 4-I p-Values Of Friedman Test For Each Algorithm

NN	**TWDP NN**	**MPSF**	**SVR**	**RF**
0.008029	1.063e-05	5.942e-05	0.0001538	2.312e-07

All the p-values are significant (less than 0.1) which shows that there is a statistically significant difference in the results of each forecasting algorithm due to the imputation method used for the treatment of missing values.

4.6.2 All Pair Comparison

Now that we know the null hypothesis is rejected for all algorithms, we want to know the result of which imputation methods are (statistically) significantly different from each other. Hence, to find our potential control imputation method for each algorithm we perform the Holm post-hoc procedure on p-values obtained from the Friedman test. The adjusted p-values have been reported in Table 4-II.

The significant values ($p < 0.1$) have been displayed in bold in Table 4-II. For instance, in the SVR algorithm, Zero imputation has a significant difference with LOCF, ML, and MI imputations with 0.0015, .0051, and 0.0070 p-values respectively. This makes Zero imputation a candidate for our control method. Also, LOCF imputation has a significant difference with Zero and Median imputations (0.0015 and 0.0557 respectively) which makes LOCF imputation another candidate for control method.

Looking at Table 4-II, the results of forecasting with NN and TWDP NN are significantly different when applying LOCF imputation instead of Zero or Mean imputations (0.0066, 0.0551 for NN and 7.16e-05, 2.16e-02 for TWDP NN). On the other hand, MPSF and SVR show significant statistical difference when applying Zero and LOCF imputations (for instance look at the Median vs. LOCF row for these algorithms with 0.0034 and 0.0557). RF shows significant difference with both Zero and Median imputations (Median vs. all other methods except Zero have values less than the threshold of 0.1).

4.6.3 Control Methods

In this subsection, we pick control methods for each algorithm based on the candidate methods derived in the previous subsection and make sure each control method for each algorithm produces significantly different results than all other methods. To this end, we perform Friedman post-hoc test with Hommel's procedure [38] for adjusting p-values. Each row in Tables 4-III and 4-VI shows the adjusted p-

values for the post-hoc procedure for each forecasting algorithm when one of the imputation methods is selected as control.

TABLE 4-II Friedman Test With Post-Hoc Holm's Method Of Adjusting p-Values For Multiple Comparison Per Algorithm

Hypothesis	**NN**	**TWDP NN**	**MPSF**	**SVR**	**RF**
Median vs. Mean	1.0000	0.7710	1.0000	1.0000	**2.24-04**
Median vs. Zero	1.0000	1.0000	1.0000	1.0000	1.0000
Median vs. LOCF	0.1328	**4.63e-05**	**0.0034**	**0.0557**	0.1898
Median vs. ML	1.0000	0.1124	0.1651	0.1306	**3.66e-02**
Median vs. MI	1.0000	0.5326	1.0000	0.1590	**9.83e-05**
Mean vs. Zero	1.0000	0.7850	0.2663	0.2493	**3.30e-04**
Mean vs. LOCF	**0.0551**	**2.16e-02**	0.2663	0.7479	0.2783
Mean vs. ML	1.0000	1.0000	1.0000	1.0000	0.6918
Mean vs. MI	1.0000	1.0000	1.0000	1.0000	1.0000
Zero vs. LOCF	**0.0066**	**7.16e-05**	**1.25e-04**	**0.0015**	0.2216
Zero vs. ML	0.5770	0.1376	**0.0150**	**0.0051**	**4.64e-02**
Zero vs. MI	1.0000	0.5858	0.2663	**0.0070**	**1.49e-04**
LOCF vs. ML	1.0000	0.3242	1.0000	1.0000	1.0000
LOCF vs. MI	0.4912	**5.57-02**	0.2663	1.0000	0.2216
ML vs. MI	1.0000	1.0000	1.0000	1.0000	0.5792

Table 4-III shows that when applying the NN algorithm, the LOCF method is statistically different from all other methods, except ML, since the adjusted p-values are less than 0.1. Even for ML, the p-value

is close to 0.1 (0.1158). However, on average LOCF results in a higher MAE for the NN algorithm. Similar results can be found in Tables 4-V and 4-VI for MPSF and SVR. Hence, we can claim: when applying the NN, MPSF, or SVR prediction algorithm on the current type of energy consumption data, using the LOCF imputation method for treating the missing data, will most likely generate worse results than other imputation methods (except maybe ML) and should be avoided.

The previous statement can be claimed even more strongly for TWDP NN since, according to Table 4-IV, all p-values are substantially smaller than 0.1.

According to Tables 4-V and 4-VI, the Zero imputation method is significantly different from other imputation methods (except Median), thus we can claim: when applying MPSF, or SVR prediction algorithms on the current type of energy consumption data, using the Zero imputation method for treating the missing data will most likely result in lower MAEs than other (except Median) imputation methods. The practical suggestion is that along with MPSF or SVR algorithms, it is recommended to try Zero (or Median) imputation. Also since our results for Median imputation is not (statistically) significantly different from Zero imputation, Median imputation should be tried as an alternative.

TABLE 4-III Post-Hoc Friedman Test With Hommel's Method Of Adjusting p-Values With Control Test For NN Algorithm

Control	**Median**	**Mean**	**Zero**	**LOCF**	**ML**	**MI**
LOCF	0.0306	0.0157	0.0022	--	0.1158	0.0818

As Table 4-VII indicates, the Zero and Median imputations are both (statistically) significantly different from other methods but not different from each other for the RF algorithm. Therefore, it is recommended to try both these imputation methods and avoid others when applying RF as the prediction algorithm on a similar type of energy consumption data.

TABLE 4-IV Post-Hoc Friedman Test With Hommel's Method Of Adjusting p-Values With Control Test For TWDP NN Algorithm

Control	Median	Mean	Zero	LOCF	ML	MI
LOCF	1.28e-05	0.0049	2.05E-05	--	0.0360	0.0092

TABLE 4-V Post-Hoc Friedman Test With Hommel's Method Of Adjusting p-Values With Control Test For MPSF Algorithm

Control	Median	Mean	Zero	LOCF	ML	MI
LOCF	0.00097	0.0484	4.19e-05	--	0.2278	0.0484
Zero	0.4317	0.0554	--	4.19e-05	0.0046	0.0554

TABLE 4-VI Post-Hoc Friedman Test With Hommel's Method Of Adjusting p-Values With Control Test For SVR Algorithm

Control	Median	Mean	Zero	LOCF	ML	MI
LOCF	0.0185	0.2804	0.0005	--	0.7531	0.7531
Zero	0.2945	0.0554	--	0.0005	0.0010	0.0016

TABLE 4-VII Post-Hoc Friedman Test With Hommel's Method Of Adjusting p-Values With Control Test For RF Algorithm

Control	Median	Mean	Zero	LOCF	ML	MI
Median	--	6.89e-05	0.9165	0.0422	0.0100	3.28e-05
Zero	0.9165	0.0001	--	0.0554	0.0139	5.34e-05

4.6.4 Dependence of Imputation Methods on Prediction Algorithms

Here, we are interested to see whether, independent of the algorithm, an imputation method could be recommended. Tables 4-III through 4-VII introduce different control methods for each algorithm, and Table 4-II shows that the pairwise comparison significance patterns are different for each algorithm. As a result, we can conclude that there is no universal best imputation method, and it is dependent on the prediction algorithm applied.

One other possibility is that for each specific outlet data there is an imputation method which minimizes the prediction error regardless of the prediction algorithm. However, since the number of prediction algorithms (five) is less than the number of imputation methods (six), there are not enough samples in order to perform statistical analysis between six methods.

4.7 Summary

In this chapter, we analyzed the effect of imputation methods (methods of replacing missing values) on prediction algorithms. The investigated data is EV charging data, as an example of energy consumption data. The most important finding in this work is that each prediction algorithm works better when paired with certain imputation methods; therefore, instead of cleaning the data and then separately working on the prediction algorithm, our findings show that the missing value correction method and prediction algorithm should be selected together.

According to our results, the LOCF imputation method should be avoided with NN, TWDP NN, MPSF, or SVR algorithms. Zero imputation method results in significantly lower average MAE when paired with MPSF, SVR, or RF prediction algorithms. With the RF prediction algorithm, however, Median imputation method is also recommended due to its significantly lower MAE.

Overall, Zero and Median imputation methods seem to be better choices. The reason that Zero and Median have similar results could be explained by the sparsity of the data. For sparse time series (time series with lots of zeros), the median is equal or close to zero, hence giving us the close performance of Zero and Median imputations.

LOCF, which replaces the missing data with the last observed value, seems to be the worst imputation method for most of the algorithms. For sparse data, the likelihood that a missing data point is zero is more than the last observed value and this might be a reason of poor performance of the LOCF method.

Another interesting observation is that NN and TWDP NN are not that sensitive to the imputation method. Other than LOCF, that is not recommended, there is no preference among the imputation methods, which makes these two nearest neighbor methods more robust with respect to the imputed values. In the smartphone application that has been developed for predicting of the EV charging consumption data at UCLA parking lots [73], the Median imputation method has been used.

Note that these conclusions are specific to these datasets.

5 Privacy and data source

5.1 Overview

In this chapter, our focus is on privacy issues and data sources when forecasting (predicting) the EV charging load based on historical charging data. We have two available datasets: the charging record that comes from anonymous customer profiles and the station records that come from measurements (voltage, current, etc.). Either one of them can be used for building a load time series and hence future load forecasting at the outlet level.

One might wonder that, since the charging records are anonymized, there is no threat to customer privacy. However, anonymizing might not be enough, as in a famous incident, the medical record for the then governor of Massachusetts was extracted easily from anonymous medical records when combined with voter registration rolls [7]. The medical records were anonymous but they had sex, ZIP code, and birth-date of patients. This incident shows that even anonymity is not enough and anonymous data might still be revealing when combined with other datasets.

Therefore, we deal with two datasets: The charging record comes from customer profiles, and, as pointed out earlier, it is prone to privacy issues. On the other hand, the station record does not have any information about specific customers and hence protects customer privacy. We compare the accuracy and speed of the prediction process using these two types of records. Specifically, for EV charging data, we investigate the potential increase in prediction accuracy and speed, as a tradeoff of endangering customer privacy. Interestingly, we found that prediction accuracy is not significantly increased while using the privacy-jeopardizing dataset (charging records). To our knowledge, this type of comparison has not been done in this context. Forecasting algorithms that we have applied in this chapter are kNN, MPSF, SVR, and RF that have all been discussed in Sections 2.4 or 3.4.

The rest of this chapter is organized as follows: Section 5.2 provides a brief review of the existing literature, and Section 5.3 briefly discusses the problem we aim to investigate. Section 5.4 presents the structure of each dataset from the charging stations at the University of California, Los Angeles (UCLA) parking lots and the preprocessing stages to convert each of them into a time series. Section 5.5 reports the result of applying the prediction algorithms on each of the time series and then analyses the results with non-parametric statistical tests to investigate statistically meaningful differences. Section 5.6 summarizes the chapter.

5.2 Literature Review

Privacy has been an important issue in smart grids and it is one of the factors holding people back from participating in the use of these new technologies [74].

There are various levels of invasion of privacy in the smart grid context. For example, at the smart household level, the different ways that household privacy might be invaded are: access to power consumptions records, presence of different players with potential access to data such as service provider and distribution operator in the economic smart grid, using wireless communication technology between devices that might make communications vulnerable, accessing energy devices from the internet, and third parties that are not involved in any part of power generation and distribution but monitor the customer usage with customers' approval [75].

According to [76], the current resolution of smart meter data (usually between 15 minutes to 1 hour) invades customer privacy and the data might not be necessary for most of the smart grid planning and distribution functions. Some other research still relies on anonymous data to protect the privacy where the pattern of the EV customer driving times is used for designing an optimal charging algorithm [77].

There have been various suggestions on how to preserve privacy. Some of them are based on the idea of aggregating the data instead of using individual data. For example, in [78], building energy usage is

investigated and only the aggregated data, instead of individual data, is used for analysis. Similarly, instead of individual EV charging data, the aggregated data of EV loads has been used for coordinating the EV charging operation in [79].

On the other hand, centralized and distributed algorithms for the routing of the information flow, which preserve the privacy based on cryptographic methods, is proposed in [80]. Cryptographic methods are also used in [81] to perform privacy preserving bill calculations. All these methods fall under Secure Signal Processing (SSP) methods which protect the sensitive data by encryption and provide tools to analyze the data under the applied encryption [82].

According to [83], privacy is exposed even when relatively infrequent measurements are acquired, and on the other hand, an energy management system using batteries can protect customer privacy. The role of the battery and its use in more effectively protecting privacy is also discussed in [84].

Lastly, the above articles focus mostly on protecting privacy at the measurement level. Another way to protect the privacy is at the data mining algorithm level. Privacy preserving machine learning algorithms started to become more important in early 2000s [85],[86]. In subsequent years, privacy preserving versions of different machine learning algorithms such as nearest neighbor [87], Bayes classifiers [88], Support Vector Machines [89], and logistic regression [90] were introduced in literature to address privacy preservation at the algorithmic level.

There is rich literature for time series forecasting in various disciplines. Reference [91] provides a comprehensive review of different models. Machine Learning algorithms have also been successfully employed in the forecasting realm [32]. In the current work, we compare four machine learning based prediction algorithms on two time series built from different measurements of one phenomenon, one of which contains privacy-jeopardizing information whereas the other does not. We show that the dataset

without privacy-sensitive information allows us to make equally accurate charging load prediction compared to the other dataset, thus precluding the need for privacy-preserving data mining techniques.

5.3 Problem Statement

The problem statement is similar to previous chapters. The objective continues to be the prediction of the available energy in the next 24 hours at each charging outlet. However, in this chapter, we focus on the two different data sources, one of which is more prone to jeopardizing privacy. We seek to determine if the use of one data source gives significantly different prediction performance than the other.

5.4 Data and Preprocessing

The prediction algorithms described in the next section are applied to the charging stations located on the UCLA campus. The data used in this chapter were recorded from December 7, 2011 to February 28, 2014; however, not all outlets were installed at the same time or were in use on all days. Among the charging outlets at UCLA, 28 outlets have charging data for more than 30 effective days (days that nonzero charging has been reported); the data from these outlets have been used in this chapter.

Data for each outlet comes in two formats: Charging Records and Station Records. We are interested in analyzing the difference in using prediction algorithms on each of these formats. These formats have been explained below, as well as the timing and stages of preprocessing for each format in order to acquire the time series.

5.4.1 Station Records

This dataset comes directly from measurements at the outlets. Thus, when accessing this dataset from the server, it is not part of the customer's profile; rather it is the recorded quantities at the outlet. Each station record contains measurements, such as voltage, current, and power factor of the charging outlet, in

three to five minute intervals. In order to obtain the real power time series, we multiply the voltage, current, and power factor.

However, not all of these quantities are available or have been reported correctly at all instances. In this chapter, we classify data which is either unreported or the data that is not in the nominal expected range (and hence non usable) as missing data.

The process of providing the best guess for the missing values is called imputation [58]. Some imputation methods involving deletion, such as “case deletion” where the incomplete instance of data is removed from the dataset, are not suitable for time series since they will change the relative order of events and make the time series lose its ordinal properties such as periodicity. A more elaborate discussion on imputation methods and their application on energy time series have been discussed in [112]. The article argues that each prediction algorithm goes along well with a certain imputation method and care should be taken in selecting an imputation method for each prediction algorithm. Based on this reference, we chose zero imputation for SVR and MPSF prediction algorithms and median imputation for RF and NN prediction algorithms.

In the case of zero imputation, the value of zero is substituted for all missing values. In this method, missing voltage, current, or power factor, is substituted with zero. This imputation preserves the sparsity of the time series. In median imputation, on the other hand, missing values of each quantity are being replaced with the median of that quantity for that specific outlet. The advantage of the median imputation method is that the imputed value is always one of the actual values of the data.

Before applying the imputation methods, missing values need to be identified. For voltage, current, and power factor, in addition to unreported and negative reported values, the reported values that were more than the maximum voltage, maximum current or maximum power factor (one) were identified as missing values.

Since we are interested in comparing the prediction results with that of the charging records dataset and hence forecast on hourly basis, we up-sample the time series to form a power time series of one hour granularity for each outlet.

5.4.2 Charging Records

This dataset comes from anonymous user profiles. Every time that an EV uses a charging facility, a charging record is added to the EV profile. Each charging record contains the charging time (beginning and end) and the acquired energy in kWh. In order to make the times series with one hour granularity, the Charging Records are converted to time series by uniformly dividing the acquired energy to the (rounded) charging interval; e.g., if the charging interval is 3.2 hours and the acquired energy is 3kWh, it is assumed that the EV received 1kWh of energy in each hour.

Charging records are different from station records in that missing data cannot be easily identified. This is because charging records are event triggered measurements. In the case of station records, when we do not receive a value (i.e. for voltage) in a five minute period, we know that the value is missing; however, if we do not receive any charging record in a time period, we conveniently assume no charging event has occurred. Therefore, there is no need for missing value imputation in charging record dataset.

5.4.3 Comparing Two Datasets

As explained earlier, the main difference between the two datasets is that charging records are derived from user profile data. These records include the entrance and exit time of each EV and hence are prone to jeopardizing the user's privacy and could lead to misuse. Station records, on the other hand, are direct measurements of quantitates at the outlet and are independent from customer or particular EV information.

The figure below shows constructed time series with one hour granularity (as explained above) from both dataset formats for a sample day (August 13, 2013, to be specific) for one of the outlets.

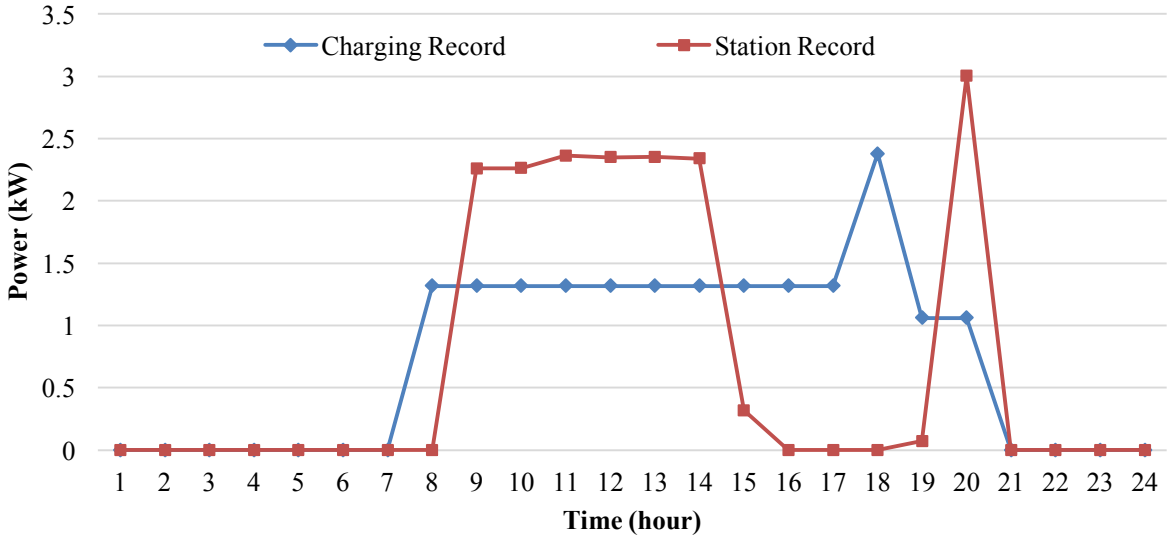


Figure 5-1 Time series constructed from station record and charging record formats.

According to Fig. 5-1, since the charging record (the diamond blue curve) indicates the beginning and ending of the EV presence at the outlet, we can speculate that one EV has been present at the outlet from 8 to 18 while another EV has been present from 18 to 20. The station record (the square orange curve), however, shows the time when the EV is actually receiving power from the outlet, which for the first EV is from 9 to 15 and for the second one only one hour at 20. So these two datasets are describing the same phenomenon with different accuracy. It is important to note that the areas under both curves are equal to each other, meaning that they both report the same amount of energy being consumed. If the ultimate goal is predicting the available energy at each outlet as in [52], the predictions based on either one are expected to behave competitively.

Another difference between these two types of datasets is their preprocessing time. Each charging record is made of three values (beginning and end of charging and acquired energy), and the time required for preprocessing is divided between accessing the database and creating the time series from those three

values. However, for preparing the time series from station records, one needs to access the database, identify missing values and impute them and, finally, up-sample the time series to achieve the time series with one hour granularity. Therefore, it should not be surprising that preprocessing time for station records is more than charging records.

Fig. 5-2 shows the preprocessing time for charging record dataset and station record dataset per each outlet.

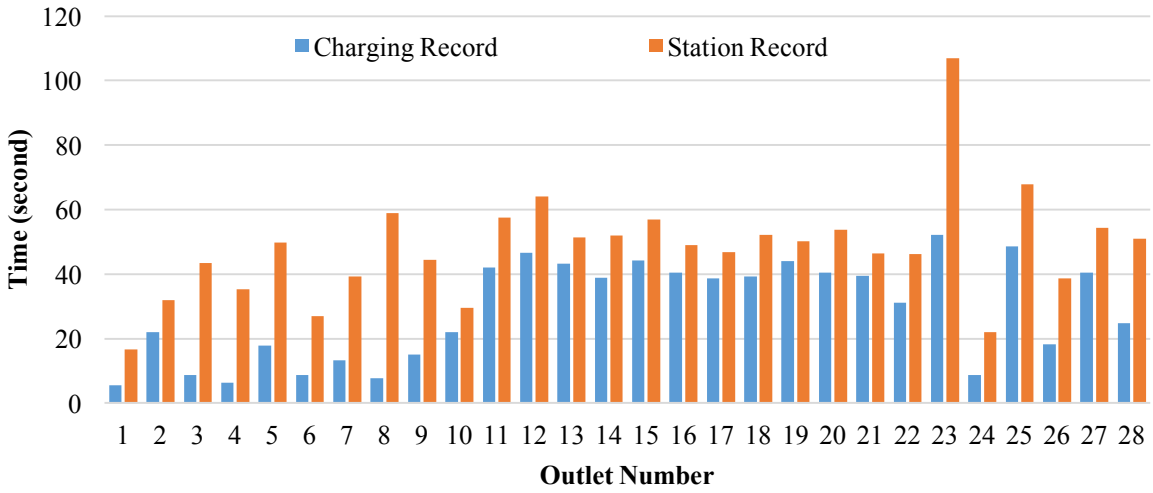


Figure 5-2 Preprocessing times for preparing times series from both formats of data per outlet

Depending on the number of charging records for each outlet, the preprocessing time will be different. Note that for a certain outlet, the preprocessing time for station records dataset will not change by the rate of usage of the outlet, while charging records dataset is a direct function of the number of charging events for the outlet. If there is no charging event for an outlet, then there will be no charging record and the time series is readily available, but for a station record all the steps of identifying missing values and possibly imputation as well as up-sampling should be performed.

Thus, we see that one difference between these two datasets is that it always takes longer to preprocess (and eventually predict) station records compared with charging records.

5.5 Simulations and Analysis

In this section we first describe our parameter selection procedure, followed by reporting the results and analyzing them.

5.5.1 Parameter Selection

We use Blocked Cross-Validation as explained in section 3.4.2. The range for each of the parameters of each algorithm has been discussed in the following lines.

In cross validation, the depth parameter (D) varies between 1 and 30 (equal to looking only at yesterday and up to the past month). The kernel type for SVR was selected from linear, radial basis, sigmoid, and polynomial kernels. Other parameters for SVR and their ranges are (the bold parameter is the default in the relevant R package): $\varepsilon \in \{0.01, \mathbf{0.1}\}$ and $C = \{0.1, \mathbf{1}\}$. Similarly for RF, the parameters are the number of trees, $nt \in \{200, \mathbf{500}\}$, number of variables to consider for splitting at each node, $m \in \left\{\frac{1}{6}, \mathbf{\frac{1}{3}}, \frac{2}{3}\right\} \times D$, and minimum size of terminal nodes, $ns \in \{\mathbf{5}, 10\}$. There are lots of other parameters for SVR and RF for which we used their default value in the relevant R package.

5.5.2 Results

The training set in our simulations was the first 90% of the data which makes the test set the last 10% of the data. We used five blocks in the cross validation procedure.

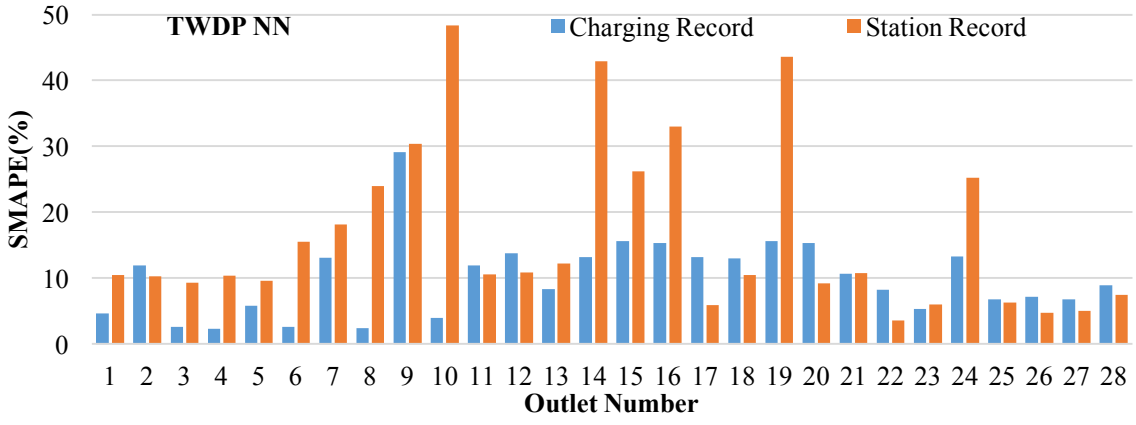
Table 5-I below shows the average prediction SMAPE on all outlets for each algorithm when using either charging record or station record time series.

TABLE 5-I Average Of SMAPE (%) On Test Days For Charging Record And Station Record Based Time Series For Each Algorithm

Time series origin	NN-TWDP	MPSF	SVR	RF
Charging Record	10.02	7.85	19.79	20.82
Station Record	16.45	6.28	20.68	20.09

Care should be taken before deciding which source of data leads to a better prediction accuracy based on just the average accuracy over all the outlets, due to varied performance of each algorithm on each individual outlet. In making such a decision, statistical analysis needs to be considered.

Fig. 5-3 shows the average SMAPE on test days based on both datasets per prediction algorithm and outlet. In the analysis section we will investigate whether there is a statistically significant difference in the accuracy of the algorithms when using either of the datasets.



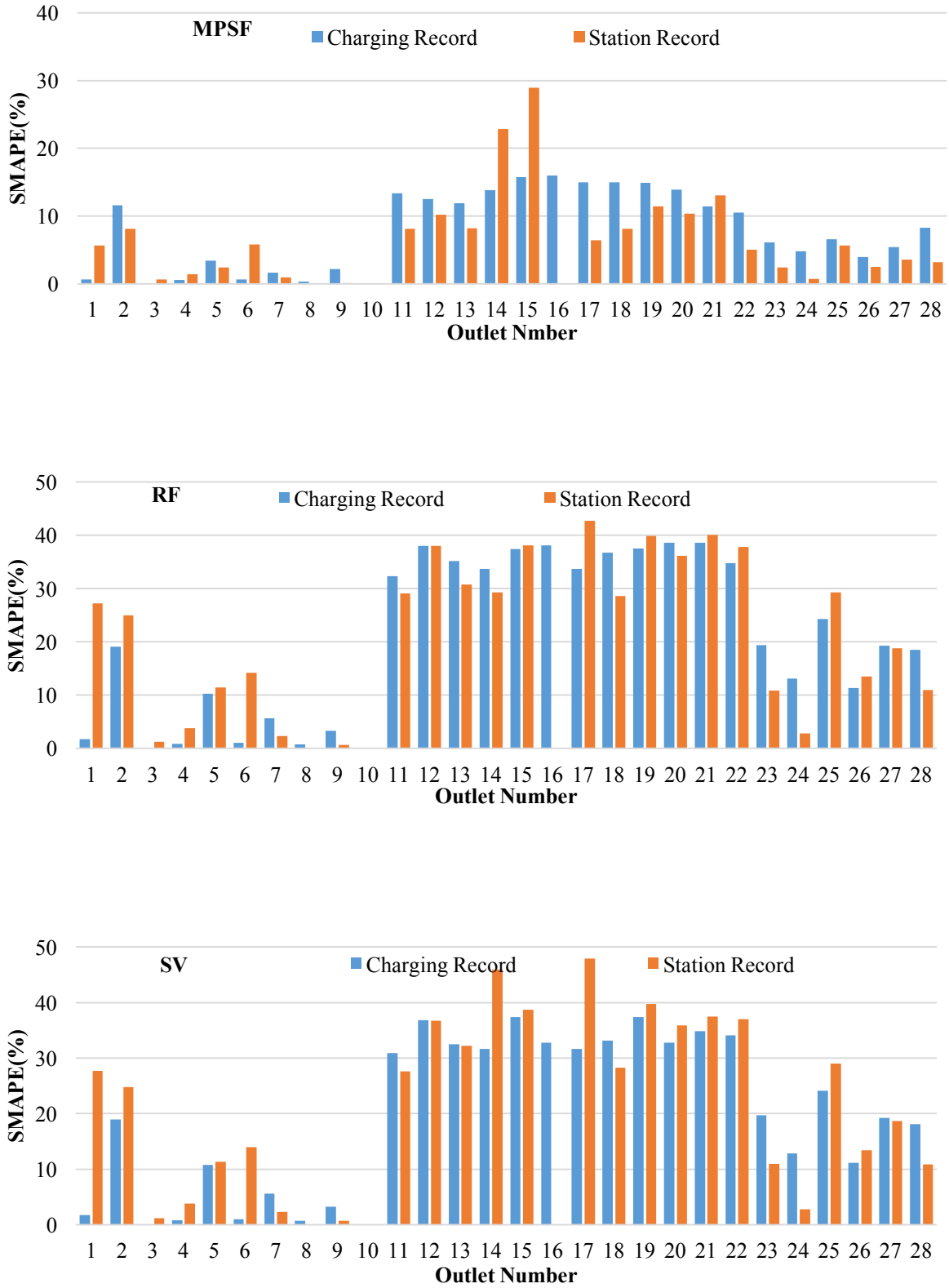


Figure 5-3 Average Symmetric Mean Absolute Percentage Error (SMAPE) on test days based on both datasets for each outlet for TWDP NN, MPSF, SVR, and RF algorithms.

Note that since the time series attained from either of the datasets have the same granularity, the processing time for a certain prediction algorithm and certain outlet on either of the time series would be the same.

The analysis has been performed with RStudio version 0.98.1091 and Microsoft SQL Server Management Studio on an Intel Core i-7 CPU at 3.40 GHz with 16 GB RAM. RStudio is running under R version 3.1.2.

5.5.3 Analysis

There are various criteria to judge whether a set of results is better than another. In general, the criterion depends on the application. For example in a business model, if the penalty depends on average SMAPE, the algorithm with less average SMAPE will be selected. However, if we want to know how often it is probable that two set of results significantly vary from each other, we need to use statistical tests.

In this section, we want to see whether there is a statistically significant difference between the accuracy of the predictions based on either of the datasets. To this end, we use the Wilcoxon signed rank test [38]. In our application, this test compares the SMAPE resulted from two different datasets at each outlet. Depending on the number of positive and negative differences and their absolute value, the test gives a probability (p-value) of how likely these two SMAPE results come from the same distribution. A small p-value means that it is unlikely that the two SMAPE population come from the same distribution, and hence there is a statistically significant difference between the results.

It is common to pick a threshold for the p-value which is called significance level and denoted by α. It is customary to pick α equal to 0.01, 0.05 or 0.1 [38]. We pick $\alpha = 0.05$. The Table 5-II shows applying the Wilcoxon test on each of the prediction algorithm results depicted in Fig 5-3.

TABLE 5-II p-Values Of Wilcoxon Signed Rank Test On Charging Record And Station Record Based Time Series For Each Algorithm

	NN-TWDP	MPSF	SVR	RF
p-value	0.02983	0.04235	0.4785	0.7639

By selecting the significance level (α) of 0.05 for p-value, there is no statistically significant difference between using charging record and station record based time series for SVR and RF algorithms (corresponding p-values are greater than 0.05). However, since the p-value is less than 0.05 for NN-TWDP and MPSF algorithms, there is a statistically significant difference between prediction results when using charging record or station record time series.

For NN-TWDP and MPSF, where the statistically significant difference between results has been observed, the dataset with greater occurrence of lower SMAPE is considered preferable. For NN-TWDP, most of the outlets have lower SMAPE when using the charging record; while for MPSF, most of the outlets have lower SMAPE when using the charging record (Fig. 5-3). Therefore, statistically speaking, the charging record based time series gives higher accuracy when NN-TWDP is the prediction algorithm; however, the station record based time series gives higher accuracy when MPSF is the prediction algorithm. There is no difference between using either of the datasets for RF and SV algorithms.

Next, we investigate whether there is a statistically significant difference between the prediction results of the two datasets, regardless of the algorithm. In order to make an overall conclusion on the effect of these two datasets on the prediction accuracy, one approach is to perform the Wilcoxon signed rank test on all the results together. This way instead of 28 samples (number of outlets) per algorithm, we have 112 samples (28 outlet and four algorithms) for all four algorithms. In this case, the p-value from the Wilcoxon test is 0.6136 which shows that, overall, there is no statistically significant difference in using either of the

charging record or station record based time series. This conclusion is not unexpected since the charging records are essentially zeroth-order approximation of the station records dataset, and the station records themselves are usually constant on pretty large intervals (e.g. Fig. 5-2). We speculate if the station records were a time series with lots of fluctuations, charging records, as it is zeroth-order approximation, would result in significantly worse prediction accuracy.

5.6 Summary

In this chapter, we investigated the difference between predictions based on time series obtained from charging records and station records. A charging record contains three values for each charging event (beginning and end of charging and the acquired energy) which comes from customer profiles. A station record, on the other hand, comes from station measurements and is a five-minute log of voltage, current, and power factor and its length depends on the length of the charging event.

Because of the greater volume of data per charging event for station records, preparing the time series from the station record dataset takes longer than the charging record dataset; hence, for fast prediction applications charging record is more suitable. On the other hand, charging records are part of the customer profile (although anonymous) and yields privacy concerns while station records come from station measurements without any access to customer behavior.

Perhaps the more important question is the difference in the prediction error when using these two datasets. In general, there is no statistically significant difference between prediction errors, although looking at the results for each algorithm demonstrates that for NN and MPSF, the charging records and station records create less error respectively.

Table 5-III summarizes the difference between the two datasets when used for prediction. The table can guide an application designer to choose the right dataset depending on the speed and privacy concerns as well as prediction algorithm used for the application in hand.

TABLE 5-III Summary Of Differences Between Prediction With Charging Record And Station Record Based Time Series

Time series origin	**Speed**	**Prediction Error (SMAPE)**	**Privacy Preserving**
Charging Record	Preprocessing on average twice as fast	lower SMAPE for NN-TWDP (Generally no statistically significant difference)	No (comes from customer profile)
Station Record	Preprocessing on average twice as slow	lower SMAPE for MPSF (Generally no statistically significant difference)	Yes (independent from each particular customer)

6 Solar power

6.1 Overview

In this chapter, we are applying forecasting algorithms to solve another problem in the context of smart grid that needs fast forecasting. This time, it is prediction of solar power generation from solar PV panels located on the UCLA Ackerman Union. The prediction is needed for real-time control and management of the energy storage system; if we know the generated power from PV panels, we can control the battery to have a constant current from the combination of the battery and PV unit.

The forecasting in this problem would be a minute ahead forecasting that is performed each minute and hence has to be fast. In this case, however, unlike Chapter 2, we have access to offline computation in the system and are able to train the model offline. Thus, there is no limit on training time; however, when querying the system, the output should be served in a few seconds so that, considering network overhead and communication time with battery, the one minute goal is achieved.

The rest of this chapter is organized as follows: Section 6.2 provides a literature review, Section 6.3 states the problem, and Section 6.4 reviews the prediction algorithms applied on solar power generation time series. Section 6.5 discusses the data, preprocessing of them, and the simulation setup. Section 6.6 reports the result of applying the prediction algorithms and then analyzes the results. Section 6.7 provides the summary of the chapter.

6.2 Literature Review

The increasing propagation of large-scale intermittent renewable energy resources to address the electricity power demand and the environmental concerns adds more challenges to power system operation and deregulated energy markets due to the stochastic nature of these resources [92]. Although it has been recognized for a while that renewable energy is critical to meeting our energy needs, its utilization has been hampered by its intermittency and the difficulty of predicting its availability. Thus, employing renewable energy as part of our energy supply requires reliable prediction of its availability for power generation. Embedding renewable energy prediction techniques in the grid operation procedure facilitates the massive integration of renewables. Real time control of renewables and compensating devices would be more efficient if prediction is available. Reliable prediction methods may also improve the power quality and reliability of the power grid by enabling prompt compensation of negative consequences of renewable dynamics and fluctuations [93].

The necessity of renewable energy prediction and its complexity have motivated many researchers to develop an efficient and practical solution. This chapter only focuses on solar power prediction. The solar power prediction methods can be categorized in two main groups based on the variety of parameters employed for prediction: 1) multivariate model based methods and; 2) univariate model based methods. The multivariate methods usually estimate the solar power based on multi-input parameters such as solar irradiance, cloudiness and clearness indices, temperature, wind speed, relative humidity, etc., which are mostly influential physical and atmospheric parameters on the solar generation. On the other hand, univariate methods only rely on the current or past values of the solar power time series. Evidently, the later approach is

relatively cheaper as it doesn't require acquiring and maintaining a weather station or other types of relevant measurement tools. In addition, for high speed dynamic control which requires short-term solar power prediction, univariate methods are more effective as they do not rely on prolonged data acquisition processes. Although univariate methods only look at previous recorded data, there is usually a tradeoff between accuracy, cost, and speed of the prediction methods. The focus of this chapter is on developing relatively fast forecasting method based on univariate data to serve as part of real time dynamic control system.

Although multivariate solar prediction methods have been already investigated significantly in the literature [94]-[100], univariate solar prediction methods are studied in a few publications.

Univariate model based methods can be divided into linear based models, mainly including autoregressive and autoregressive moving average [101], and nonlinear based models such as artificial neural networks [102], support vector machine with kernel trick [103], fuzzy model [104], wavelet-based methods [105], echo state network [106], and k-nearest neighbors (kNN) [107]. Although nonlinear models (compared with linear models) seem to be more accurate in terms of capturing the nonlinear characteristics and time varying behavior of solar power generation, these methods generally take a longer time for training/tuning parameters and easily fall into local optimum.

On the other hand, the univariate prediction methods can be categorized based on the prediction horizon ranging from super-short-term (about a minute ahead) to super-long-term prediction (more than a year prediction horizon). Although a few studies are available for the super-short-term prediction timeframe, this type of prediction is useful for real time control of renewables, regulation actions and power quality enhancement. While the short term prediction

methods are suitable for economic load dispatch planning or load increment/decrement decisions, and long-term prediction is normally valuable for unit commitment decisions, reserve requirement decisions, and maintenance scheduling to obtain optimal operating cost [108]. In [101], the short term univariate prediction method based on auto-regressive moving average (ARMA) model is used to predict the future solar generation according to the historical solar radiation data. Authors in [109] employ a hybrid solar power prediction method for super-short-term prediction. The objective of the latter study is to predict one step-ahead solar power generation (minutely) based only on historical solar power time series data. The long–term solar prediction is investigated in [110] while it develops a one day-ahead forecasting model based on an artificial neural network with tapped delay lines. In [111], super-long-term solar prediction is discussed while it is targeting the seasonality variations of solar potential for generation of electric and thermal powers. It also presents time series models for the analysis of insolation using daily data, transformed into monthly averages, covering a period between January 1961 and December 2008 to confirm a prediction for 2012, and it compares the results with the data measured in the 2008-2011 period.

6.3 Problem Statement

The objective is to predict the solar power generation for the next minute ahead based on historical solar power generation recorded data. Formally, we assume there is a function relating the predicted power and the past power:

$$\hat{p}(t) = f(p(t-1), p(t-2), \dots), \qquad (6\text{-}1)$$

where $p(t)$ is the actual power generated by the solar panel at time t, $\hat{p}(t)$ is the prediction of the generated power by the solar panel at time t, and $(p(t-i))$ indicates the generated power in the

past at time $(t-i)$. The main constraint for this application is that the whole process of measurement, communication, forecasting, and control action should take less than a minute to be useful. By assuming that whole process except forecasting takes about 30 seconds, the forecasting part should take well below 30 seconds to guarantee enough time for measurement, communication and control.

As is the usual practice in forecasting, we are interested in finding an estimation of $p(t)$ that optimizes a particular performance (or error) criterion. There are variety of different definitions of forecasting error in the literature [91]. To this end, two of the most common error definitions are selected and results are reported in both: Symmetric Mean Absolute Percentage Error (SMAPE) and Mean Absolute Error (MAE). The SMAPE and MAE are defined as:

$$SMAPE = \frac{1}{N_{ts}} \sum_{t \in S_{ts}} \frac{|p(t) - \hat{p}(t)|}{p(t) + \hat{p}(t)} \times 100,$$
$$MAE = \frac{1}{N_{ts}} \sum_{t \in S_{ts}} |p(t) - \hat{p}(t)| \times 100, \quad (6\text{-}2)$$

where N_{ts} is the number of data points in the test set (defined below).

Let $S_{tr} = \{1,2,\dots,N_{tr}\}$ and $S_{ts} = \{N_{tr}+1,\dots,N\}$ be two sets of indices for the training and test sets, respectively where N is the total number of data points, and N_{tr} is the number of data points in training set which makes $N_{ts} = N - N_{tr}$. Later on, in the parameter selection phase, parts of the training set will be treated as the validation set. The different methods used to select the validation set are further explained in the parameter selection section. In this chapter, the most recent 10% of the data is used to evaluate the performance of the algorithm (test set). Note that the test dataset is not used in either the parameter selection or training phase.

6.4 Applied Algorithms

The applied algorithms are ARIMA, kNN, SVR, and RF that have been introduced in 2.4 and 3.3.

6.5 Simulation Setup

6.5.1 Data and Preprocessing

The prediction algorithms described in the previous section are applied to the recorded solar power from solar PV panels located on the UCLA Ackerman union. The data used in this chapter have one minute granularity and were recorded from November 18, 2014 to April 24, 2015; however, not all the time the measurement was being recorded due to communication issues.

Missing values and outliers have also been identified and treated. If there is a missing value, the Last Observation Carried Forward (LOCF) imputation is used to substitute the value. On the other hand, if there are more than one measurement in a given minute, the median of them has been used as the power value at that minute. As power generation is a positive value, negative values are considered outliers and substituted with zero. There was no normalization or feature extracting from the data.

6.5.2 Parameter Selection

In cross validation, the depth parameter (D) varies between 2 and 60 (equal to looking only at the last two minutes and up to the past hour). We have looked at more than 60 minutes depth up to the last 24 hours as well; however, the results were not improved. This is not unexpected given that we are forecasting the next minute value, and more recent values will have more impact on

the forecasting compared with the older data in time. Also, the two periodicities imaginable for solar radiation in one location are daily and annually periodicities. Since our available data is less than one year, only daily periodicity is noteworthy here. However, given the fact that the amount of daylight is variable in the year, there is no fix daily period for solar radiations, and hence more recent values should be considered as input to the algorithms (rather than something like the last 24 hour values), where it is the case in this study.

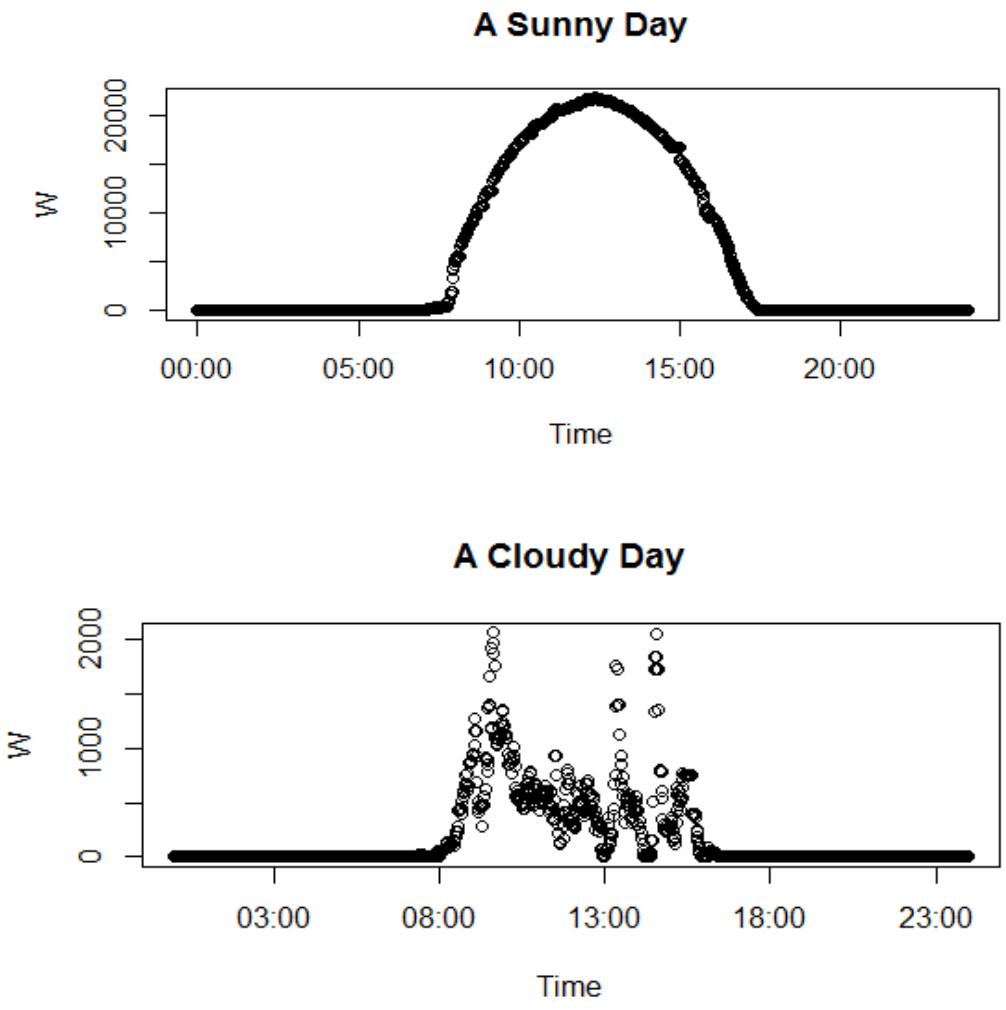


Figure 6-1 Sample recorded solar power data for a sunny day (Feb 12, 2015) and a cloudy day (Dec 2, 2014) .

The Blocked CV has been used as discussed in 3.4.2. The number of neighbors (k) ranges from 1 to 10 for kNN. Also, in the auto.arima function, maximum of p and q was set to 5 and 8 respectively. Parameter d was picked by the auto.arima function based on the KPSS test [51]. The kernel type for SVR was selected from linear, radial basis, sigmoid, and polynomial kernels. Other parameters for SVR and their ranges are (the bold parameter is the default in the related R package): $\varepsilon \in \{0.01, \mathbf{0.1}\}$ and $C = \{0.1, \mathbf{1}\}$. Similarly for RF, the parameters are the number of trees, $nt \in \{200, \mathbf{500}\}$, number of variables to consider for splitting at each node, $m \in \left\{\frac{1}{2}, \frac{\mathbf{1}}{\mathbf{3}}, \frac{1}{5}\right\} \times D$, and minimum size of terminal nodes, $ns \in \{\mathbf{5}, 10\}$. There are lots of other parameters for SVR and RF for which we used their default value in the relevant R package.

6.6 Results and Analysis

6.6.1 Results

The training set in our simulations was the first 90% of the data which makes the test set the last 10% of the data. We used five blocks in the cross validation procedure.

Fig. 6-2 shows the SMAPE and MAE for each algorithm while Table 6-I shows the optimum selected parameter for each algorithm.

According to Fig. 6-2, kNN has the best overall performance with respect to both MAE and SMAPE. Its relative error (SMAPE=1.67%) is significantly better than the other algorithms and its absolute error (MAE=237) is comparable with SVR's MAE at 217. Although all three kNN, SVR, and RF algorithms have comparable absolute errors (MAE), their relative error (SMAPE) is very different. This phenomenon has been discussed deeper in the Analysis subsection of this

chapter. ARIMA fails to accurately predict according to both criteria which is not unexpected due to ARIMA's heavy reliance on periodic behavior in the time series.

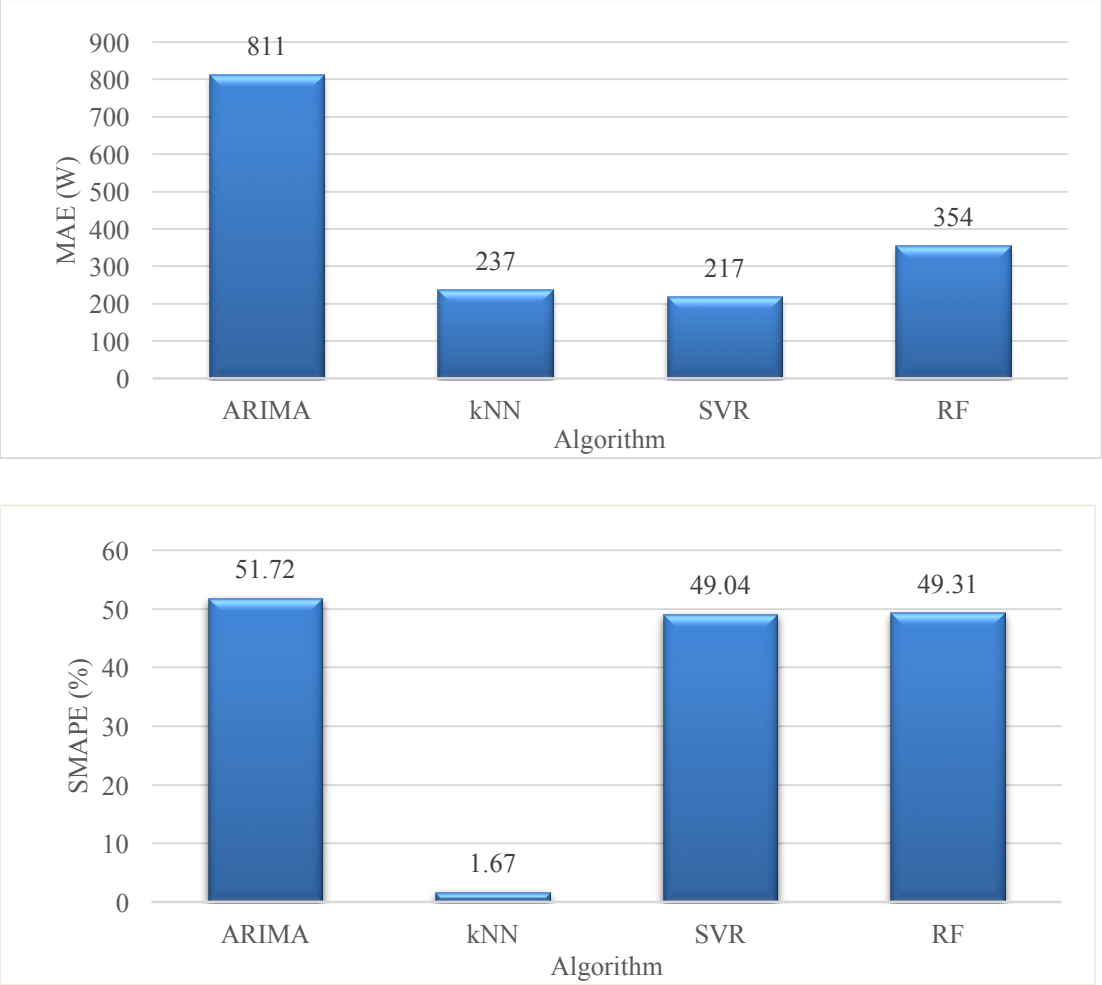


Figure 6-2 Symmetric Mean Absolute Percentage Error (Smape) and Mean Absolute Error (MAE) Averaged on test days for each algorithm

TABLE 6-I Optimum Selected Paramerters For Each Algorithm

Parameter	ARIMA	kNN	SVR	RF
Depth (D)	--	5	5	10
Neighbor (k)	--	1	--	--
Order (p,d,q)	(5,0,7)	--	--	--
Kernel	--	--	Polynomial	--
ε	--	--	0.01	--
Cost (C)	--	--	1	--
Number of trees (nt)	--	--	--	200
Splitting leaves at each node (m)	--	--	--	$\frac{1}{2}D = 5$
Minimum of terminal nodes (ns)	--	--	--	5

As Table 6-I shows, the selected depth for all the algorithms is rather short, i.e. in kNN and SVR the prediction is made by looking at the last five minutes of values and in RF by looking at the last 10 minutes of values. Even p in ARIMA (the order of Auto-Regressive model) has been selected as five, which is equivalent to considering the last five minutes of values. Hence, through optimum parameter selection, all the models rely on the most recent data to make the prediction. This, once again, emphasizes the importance of local patterns in the times series prediction rather than global patterns.

As mentioned earlier, the timing is also important, and making the prediction should not take more than a few seconds so that the control process can finish in one minute. Fig. 6-3 shows the execution time for each algorithm once they are provided with the query. Clearly, all algorithms

are able to respond to the query in less than a fraction of a second which is well below a few seconds limit. It is noteworthy to mention the higher response time is for kNN, as it is considered a Lazy Learning (or Instance-based) algorithm, such that no *learning* has been done unless a query is received. Therefore, it is not surprising if it takes longer, as the other algorithms are trained offline but kNN is not. Also, keep in mind that the training time for other algorithms (and the parameter selection time for kNN), which could take a couple of hours, is not factored in here. However these training/parameter selection can be done offline and periodically (it is done every 15 days in this chapter) so it should not interfere with the querying part. Depending on the computation cost, running the training/parameter selection more often will generate the same or better accuracies. Note that for each query, kNN searches the whole training dataset and as the data grows, the response time will increase too. The 0.1 seconds is for searching in about half a year of data (our currently available data), so it can increase to about 0.2 seconds when querying against a whole year of data.

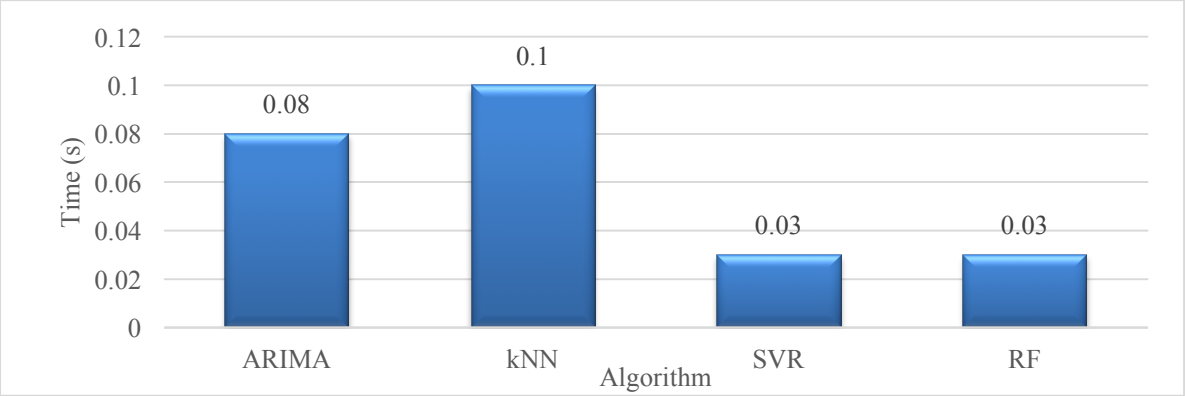


Figure 6-3 The average time (in seconds) needed for each algorithm to make a one minute ahead prediction.

The simulations has been performed with RStudio version 0.98.1091 on an Intel Core i-7 CPU at 3.40 GHz with 16 GB RAM. RStudio is running under R version 3.1.2.

6.6.2 Analysis

The results reveal interesting characteristics of each of the applied algorithms. For example the low SMAPE of the kNN algorithm can be justified as follows: according to (6-2), SMAPE is 100% when either of the predicted or actual value (and not both of them) is equal to zero. Considering that in kNN the prediction is always an instance of the past data and the fact that there are lots of zeros in a 24 hour period (especially at night, refer to Fig. 6-1), there will be lots of instances that actual power and the prediction are zero, hence yielding a SMAPE equal to zero. However, in other algorithms, as they have more arithmetic involved, chances are that their prediction for night time is a very small number but not exactly zero, which makes the SMAPE of that time equal to 100%. However, this type of error in prediction has a lot less of effect on MAE; therefore, MAE for other algorithms is much better compared to MAE for kNN.

One way to test if this discrepancy in error is occurring for the aforementioned reason, is to set the smaller predicted values of SVM, RF, and ARIMA equal to zero. Since the peak of the values is in the order of 35kW and the first value after night hours is in the order of 50 W, we modify these algorithms to output zero when predicted value is less than 10 W. With this modification, the results are depicted in the fig. 6-4.

The fact that thresholding the predicted values does not change the MAE considerably, but changes the SMAPE (from 49.04% to 1.53% for SVR and from 49.31% to 1.68% for RF) shows that the reasoning behind the source of difference in error measurements was right and there are lots of near zero predictions that make the SMAPE large for these algorithms. Please note this is not the case for ARIMA as the minimum of predicted values is around 500W and thresholding with 10W would not change the predicted values and error measurements.

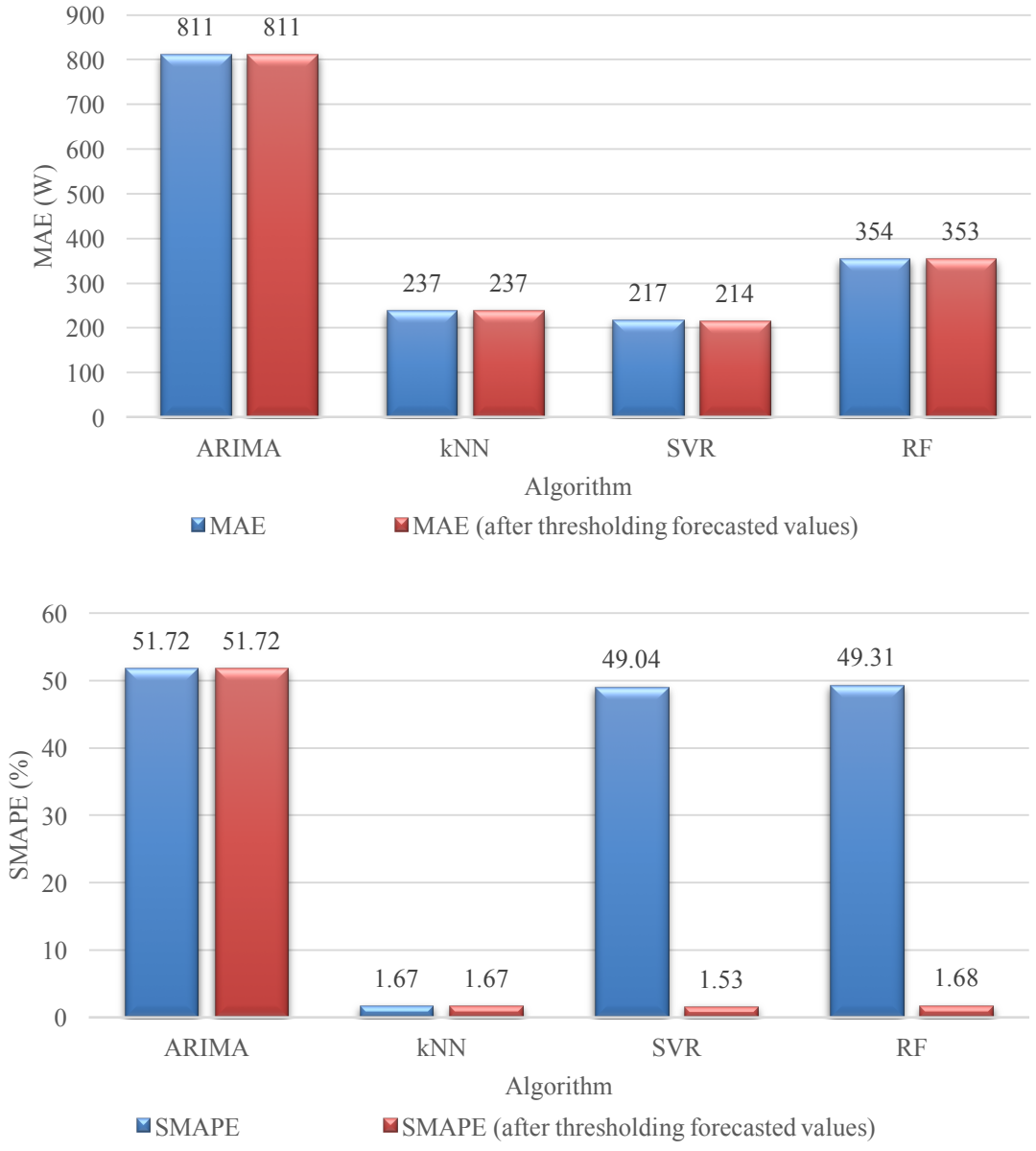


Figure 6-4 Symmetric Mean Absolute Percentage Error (SMAPE) and Mean Absolute Error (MAE) Averaged on test days for each algorithm when including the thresholding effect: the output values of ARIMA, SVR, and RF algorithms that are less than 10W is rounded to zero

When considering the thresholding procedure, SVR generates the best results with respect to both MAE and SMAPE measurements; however, kNN follows very closely in both measures.

Considering parameter selection for kNN will be relatively faster (according to Table 6-I, SVR and RF both have four parameters to select while kNN only has two), it could be a suitable substitute for SVR, when simplicity of the algorithm is considered.

6.7 Summary

In this chapter, we investigated four different algorithm for predicting the power generated by solar radiation. This is a challenging task as cloudiness of the weather can affect the quality of prediction. In our case, we want to predict generated power a minute ahead for dynamic control of energy storage. Therefore, it makes sense to focus on the most recent values of the time series and compare local temporal patterns to make the prediction. This hypothesis is verified in the parameter selection phase where, for all the algorithms, the optimum history of the data (the one that results in a lower MAE) used for predicting the minute ahead value is at most ten minutes.

One other observation is that, depending on the error measurement, naturally, different algorithms are deemed optimum. For example, according to Fig. 6-2, kNN is the best algorithm when considering the SMAPE while SVR is the best one when considering the MAE. It is important for a system designer to pick an error measurement that models their concerns/costs of the problem better and also to understand the difference of these error measurement criteria to be able to tell why one error is high while the other is low. We speculate in this chapter that the mismatch between results of both errors is because of near zero situations where SMAPE is 100% if actual value is zero and the predicted value is not zero regardless of the magnitude of the predicted value and the magnitude of its difference with the actual value. Based on this speculation, we set a threshold to round near-zero predicted values to zero, thus changing the SMAPE of that point from 100% to 0%.

SVR and kNN outperformed the other algorithms; kNN needs less parameters to tune and hence will result in a simpler system. In general, it seems that machine learning based algorithms (SVR, RF, and kNN) outperform the traditional ARIMA algorithm drastically.

7 Conclusion and Future Work

Throughout this research, we focus on forecasting algorithms with an emphasis on Machine Learning algorithms in the smart grid context. We believe that this research will be useful in this new era of smart grid growth. We have proposed two new algorithms, a cross validation method, as well as took the approach of using statistical methods to evaluate the performance and draw conclusions on the effectiveness of the applied algorithms which is not commonly used in the engineering field.

In this section, we provide our conclusions and suggest future research directions:

7.1 Fast Prediction for a Smartphone Application

We look at Instance-based prediction algorithms for their speed and their potential for being appropriate for our smartphone application. Among all Instance-based algorithms, kNN was faster and more accurate for almost all of the outlets.

One interesting finding is that for all outlets, $k = 1$ is the optimum number of neighbors in the kNN algorithm. This means it is optimal to look at the most similar pattern in the past, to predict the future instead of aggregating more than one similar patterns. Note that the depth parameter (D) is different from number of neighbors (k) and might be different for each outlet, i.e. for one outlet it might be optimum to compare 5 day patterns with the training set while in the other one it might be 10 days.

We improved the forecasting accuracy and speed by substituting the Euclidean distance in kNN with Time Weighted Dot Product (TWDP) dissimilarity. The TWDP dissimilarity helped speed by reducing the time to compute the dissimilarity from O(3n) to O(2n), and it improved

accuracy by having it time weighted. Also, the dot product tends to detect the events (non-zero activities) in sparse time series (where there is lots of zero values), and hence, it is appropriate for comparing sparse times series. We used it to implement the forecasting engine on the smartphone application for EV drivers at UCLA campus.

Looking into other dissimilarity definitions for further reducing computational cost could be a future direction. It would be interesting to map different dissimilarities to different time series by some statistical attribute of the time series.

7.2 Non-Instance based Algorithms

We explore the potential of including offline computation in the accuracy of our predictions. This was an important step in our study, as we needed to make an educated decision on any potential trade-offs in accuracy when using Instance-based algorithms for our smartphone application. After comparing the performance of different non-Instance based algorithms, we proposed the Modified Pattern Sequence-based Forecasting (MPSF) algorithm which is an improvement of the Pattern Sequence-based Forecasting (PSF) algorithm that has been successful in price forecasting applications.

The challenge with most of these algorithms is number of their "tuning knobs" (parameters, kernel function, etc.). Finding the optimal combination for each algorithm is a significant task to be investigated in future work.

7.3 Blocked Cross-Validation

We have introduced Blocked Cross-Validation as the fittest Cross-Validation (CV) method for time series. We discussed how k-fold CV which comes from the Machine Learning realm does not respect the order of the samples since data samples are shuffled randomly. On the other hand,

last block CV (from the world of time series analysis) does not provide more than one error measure per parameter combination and hence is not able to provide a confidence interval for the error measure. Our proposal takes advantage of both approaches: it respects the order while providing more samples per parameter combination.

Future work would be to investigate the application of Blocked Cross Validation method in other algorithms and time series prediction methods.

7.4 Comparing Algorithms using Statistical Methods

We used statistical tests to claim whether an algorithm has consistently lower error than another one, rather than merely looking at their average behavior. This is very important and not well practiced in the engineering world, but with abundance of data and data driven approaches in this era of big data, well-grounded statistical tests should be taken seriously. While the average error of some algorithm on several benchmarks (e.g. time series in our case) could be lower than another, it does not necessarily mean that algorithm should be preferred all the time. Rather, the distribution of the number of times that the algorithm generates the lower error determines whether or not the algorithm is consistently better than others. Hence, making decisions by just looking at average behavior might be misleading.

7.5 Imputation Methods and their Influence on Prediction Results

We looked into another challenge while dealing with data: missing or corrupted values. We looked at different ways to find the best guess for the missing value; this process is called imputation. We found that the accuracy of different forecasting algorithms might change based on the imputation method that has been used to remedy the missing data. Therefore the imputation method and forecasting algorithm should be treated as one package and not separately. We

proceed to determine a suggested set of imputation methods to avoid or pair along for each forecasting algorithm.

A significant research contribution would be to investigate the (statistical) properties that make forecasting algorithms and imputation methods combinations fit or unfit.

7.6 Privacy Preservation Concerns in Data

When dealing with data, there is always a concern with privacy. We had access to two sets of data that were describing a (somewhat) similar EV charging phenomenon. However, one of them is prone to jeopardizing privacy (by coming from an anonymous customer profile database) while the other dataset is not. We compared them for speed and accuracy and found that the one that comes from the anonymous customer profiles is faster to pre-process due to a reduced number of data points per charging event (three); on the other hand, the other dataset collects measurements each five minutes, so depending on the length of the charging event, it might have a relatively large number of data points. As for accuracy, we did not find any statistically significant difference which should not be surprising as they report the same phenomenon.

It would be interesting to see the privacy risks that these datasets can actually pose. More specifically, one could investigate whether customer EV information, such as vehicle make, model, etc., can be extracted from the charging patterns of different EVs.

7.7 Solar Power Forecasting for Energy Storage Management

As another forecasting problem in the context of smart grid, we investigated forecasting solar power in order to compensate the overall current with a battery management system. If we know the solar power in the future, we can adjust the battery charging or discharging so that the overall current from the solar panels and battery is constant. We investigated several forecasting

algorithms and conclude that kNN and SVR perform the best. Although kNN, because of its simple structure and less number of parameters, is recommended.

Future work would be to implement the algorithm into the battery management system and test it in a real world setting.

References

[1] Jeff Cobb, "December 2011 Dashboard", "December 2012 Dashboard", "December 2013 Dashboard", "December 2014 Dashboard", HybridCars.com and Baum & Associates.

[2] Jeff Cobb, "Top 6 Plug-In Vehicle Adopting Countries – 2014". HybridCars.com.

[3] M. Kintner-Meyer, K. Schneider, R. Pratt, "Impacts Assesment of Plug-in Hybrid Vehicles on Electric Utilities and Regional U.S. Power Grids Part 1: Technial Analysis", Pacific Northwest National Laboratory, 2007 [Online]. Avilable: http://energytech.pnnl.gov/publications/pdf/phev_feasibility_analysis_part1.pdf

[4] M. Majidpour, W.P. Chen, "Grid and Schedule Constrained Electric Vehicle Charging Algorithm Using Node Sensitivity Approach", *Proc. 2012 Intl. Conf. Connected Vehicles and Expo (ICCVE),* pp. 304-310.

[5] Alternative Fuels Data Center, U.S. Department of Energy, [Online]. Available http://www.afdc.energy.gov/fuels/stations_counts.html

[6] P. McDaniel and S. McLaughlin, "Security and Privacy Challenges in the Smart Grid," *IEEE Secur. Priv.*, vol. 7, no. 3, pp. 75–77, May 2009.

[7] L. Sweeney, "Weaving Technology and Policy Together to Maintain Confidentiality," *J. Law. Med. Ethics*, vol. 25, no. 2–3, pp. 98–110, Jun. 1997.

[8] N. S. Altman, "An introduction to kernel and nearest-neighbor non-parametric regression". *The American Statistician* 46 (3): pp. 175–185, 1992.

[9] M. Majidpour, C. Qiu, P. Chu, R. Gadh, H. Pota, "A Novel Forecasting Algorithm for Electric Vehicle Charging Stations", in *Proc. of International Conference on Connected Vehicles & Expo* (ICCVE-2014), Vienna, Austria.

[10] T. G. Dietterich, "Machine Learning for Sequential Data: A Review", *Springer Lecture Notes in Computer Science*, vol.2396, 2002, pp.15-30.

[11] GEP Box, GM Jenkins, GC Reinsel, *Time series analysis: forecasting and control*, Wiley Series in Probability and Statistics, 2013 (4th ed.)

[12] J. D. Hamilton, *Time Series Analysis*, Princeton University Press, 1994.

[13] C. E. Borges , Y. K. Penya and I. Fernandez "Evaluating combined load forecasting in large power systems and smart grids", *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp.1570 - 1577, 2013.

[14] E. Terciyanlı, T. Demirci, D. Küçük, M. Saraç, Işık Çadırcı, M. Ermi " Enhanced Nationwide Wind-Electric Power Monitoring and Forecast System ", *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp.1171-1184, 2014.

[15] C. Chen, S. Duan "Optimal Integration of Plug-In Hybrid Electric Vehicles in Microgrids", *IEEE Trans. Ind. Informat.*, vol. 99, pp.1-11, 2014.

[16] M. Sechilariu , B. Wang and F. Locment "Building integrated photovoltaic system with energy storage and smart grid communication", *IEEE Trans. Ind. Electron.*, vol. 60, no. 4, pp.1607 -1618, 2013.

[17] G. Zhang, H.-X. Li, and M. Gan " Design a Wind Speed Prediction Model Using Probabilistic Fuzzy System", *IEEE Ind. Informat.*, vol. 8, no. 4, pp. 819 - 827, 2012.

[18] A. Khosravi, S. Nahavandi, " Load Forecasting Using Interval Type-2 Fuzzy Logic Systems: Optimal Type Reduction ", *IEEE Trans. Ind. Informat.*, vol. 10, no.2, pp. 1055 - 1063, 2014.

[19] C. Chung, A. Shepelev, C. Qiu, C. Chu, R. Gadh, "Design of RFID Mesh Network for Electric Vehicle Smart Charging Infrastructure", *IEEE RFID TA 2013*, Johor Bahru, Malaysia, 4-5 September, 2013.

[20] S. Mal, A. Chattopadhyay, A. Yang, R. Gadh, "Electric vehicle smart charging and vehicle-to-grid operation", *Intl Journal of Parallel, Emergent and Distributed Systems*, vol. 27, no. 3, pp. 249-265, March 2012.

[21] C.Chung, E. Youn, J. Chynoweth, C. Qiu, C. Chu, R. Gadh, "Safety Design for Smart Electric Vehicle Charging with Current and Multiplexing Control", *2013 IEEE International Conference on Smart Grid Communications,* Vancouver, Canada, 21-24 October, 2013.

[22] C. Chung, J. Chynoweth, C. Qiu, C. Chu, R. Gadh, "Design of Fast Response Smart Electric Vehicle Charging Infrastructure", *IEEE Green Energy and Systems Conf.*, Long Beach, CA, Nov 25, 2013.

[23] K.N. Kumar, P.H. Cheah, B. Sivaneasan, P.L. So, D.Z.W. Wang, "Electric vehicle charging profile prediction for efficient energy management in buildings", in *Proc. 2012 IEEE Conference on Power & Energy,* pp. 480 – 485.

[24] A. Aabrandt , P. B. Andersen , A. B. Pedersen , S. You , B. Poulsen , N. O'Connell and J. Ostergaard "Prediction and optimization methods for electric vehicle charging schedules in the EDISON project", *IEEE Power Energy Soc. Innovative Smart Grid Tech. Conf.*, pp.1-7, 2012.

[25] R. Shankar, J. Marco, "Method for estimating the energy consumption of electric vehicles and plug-in hybrid electric vehicles under real-world driving conditions" *IEEE Intelligent Transport Systems*, vol. 7, pp. 138 – 150, March 2013

[26] A. Ashtari , E. Bibeau , S. Shahidinejad and T. Molinski "PEV charging profile prediction and analysis based on vehicle usage data", *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp.341 -350, 2012

[27] M. Alizadeh, A. Scaglione, J. Davies, K. S. Kurani, "A Scalable Stochastic Model for the Electricity Demand of Electric and Plug-In Hybrid Vehicles", *IEEE Trans. Smart Grid*, no. 99, pp.1-13, Sept 2013

[28] Y. Q. Li, Z. H. Jia, F. L. Wang, Y. Zhao, "Demand Forecast of Electric Vehicle Charging Stations Based on User Classification", *Applied Mechanics and Materials*, pp. 291-294, 2013

[29] J. Wang, K. H. Wu, F. Wang, Z. H. Li, Q. S. Niu, Z. Z. Liu, "Electric Vehicle Charging Station Load Forecasting and Impact of the Load Curve", 2012, *Applied Mechanics and Materials*, 229-231, 853.

[30] F. Kennel , D. Gorges and S. Liu "Energy management for smart grids with electric vehicles based on hierarchical MPC", *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp.1528 - 1537, 2013.

[31] J. C. Ferreira, V. Monteiro, J. L. Afonso "Vehicle-to-Anything Application (V2Anything App) for Electric Vehicles", *IEEE Trans. Ind. Informat.*, vol. 99, pp.1-11, 2013.

[32] N.K. Ahmed, A.F. Atiya, N. El Gayar, H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting", *Econometric Reviews*, 29 (2010), pp. 594–621.

[33] C. M. Bishop, N. M. Nasrabadi, *Pattern recognition and machine learning*. Vol. 1. New York: springer, 2006.

[34] S. A. Dudani, "The Distance-weighted k-Nearest Neighbor Rule," *IEEE Transactions on System, Man, and Cybernetics*, Vol. SMC-6, pp. 325-327, 1976.

[35] J. Zavrel, "An empirical re-examination of weighted voting for K-NN," In: Daelemans W, Flach P, van den Bosch A (eds) *Proceedings of the 7th Belgian-Dutch Conference on Machine Learning*, Tilburg, pp 139-148, 1997.

[36] G. Bontempi, S. Ben Taieb, and Y. Le Borgne. "Machine Learning Strategies for Time Series Forecasting." In *Business Intelligence*, pp. 62-77. Springer Berlin Heidelberg, 2013.

[37] D. M. Allen, "The relationship between variable selection and data augmentation and a method for prediction." *Technometrics* 16, no. 1 (1974): 125-127.

[38] J Derrac, S García, D Molina, F Herrera. "A practical tutorial on the use of non-parametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms." *Swarm and Evolutionary Computation* 1, no. 1 (2011): 3-18.

[39] R. Hyndman. (2010, October 4). [Online]. Available: http://robjhyndman.com/hyndsight/crossvalidation/

[40] Ben Taieb, S., Bontempi, G., Atiya, A.F., Sorjamaa, A. "A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition", (2012) *Expert Systems with Applications*, 39 (8), pp. 7067-7083.

[41] M. Aizerman, E. Braverman, L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning", *Automation and Remote Control* 25: 821–837, 1964.

[42] Luo, Zhuowei, Yonghua Song, Zechun Hu, Zhiwei Xu, Xia Yang, and Kaiqiao Zhan. "Forecasting charging load of plug-in electric vehicles in China." In Power and Energy Society General Meeting, 2011 IEEE, pp. 1-8. IEEE, 2011.

[43] Feixiang, Xie, Huang Mei, Zhang Weige, and Li Juan. "Research on electric vehicle charging station load forecasting." In Advanced Power System Automation and Protection (APAP), 2011 International Conference on, vol. 3, pp. 2055-2060. IEEE, 2011.

[44] Chen, Wenying, Xingying Chen, Yingchen Liao, Gang Wang, Jianguo Yao, and Kai Chen. "Short-term load forecasting based on time series reconstruction and support vector regression." In *TENCON 2013-2013 IEEE Region 10 Conference (31194)*, pp. 1-4. IEEE, 2013.

[45] Xydas, E. S., C. E. Marmaras, L. M. Cipcigan, A. S. Hassan, and N. Jenkins. "Electric Vehicle Load Forecasting using Data Mining Methods."

[46] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2004.

[47] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[48] L. Breiman, "Technical note: Some properties of splitting criteria," *Mach. Learn.*, vol. 24, no. 1, pp. 41–47, 1996.

[49] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 161–168.

[50] Weisang, G., & Awazu, Y. (2008). Vagaries of the Euro: an Introduction to ARIMA Modeling. Case Studies in Business, Industry, and Government Statistics, 2, 45-55.

[51] Hyndman, R.J. and Khandakar, Y. (2008) "Automatic time series forecasting: The forecast package for R", *Journal of Statistical Software*, **26**(3).

[52] M. Majidpour, C. Qiu, C.-Y. Chung, P. Chu, R. Gadh, and H. R. Pota, “Fast demand forecast of Electric Vehicle Charging Stations for cell phone application,” in *2014 IEEE PES General Meeting | Conference Exposition*, 2014, pp. 1–5.

[53] F. Martínez-Álvarez, A. Troncoso, J. C. Riquelme, and J. S. Aguilar-Ruiz, “LBF: A labeled-based forecasting algorithm and its application to electricity price time series,” in *Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on*, 2008, pp. 453–461.

[54] F. Martinez Alvarez, A. Troncoso, J. C. Riquelme, and J. S. Aguilar Ruiz, “Energy time series forecasting based on pattern sequence similarity,” *Knowl. Data Eng. IEEE Trans. On*, vol. 23, no. 8, pp. 1230–1243, 2011.

[55] C. Bergmeir and J. M. Benítez, “On the use of cross-validation for time series predictor evaluation,” *Inf. Sci.*, vol. 191, pp. 192–213, 2012.

[56] J. Opsomer, Y. Wang, and Y. Yang, “Non-parametric regression with correlated errors,” *Stat. Sci.*, pp. 134–153, 2001.

[57] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection". *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence* 2 (12): pp.1137–1143, 1995.

[58] P. D. Allison, *Missing Data*. SAGE Publications, 2001.

[59] J. K. Kim and J. Shao, Statistical Methods for Handling Incomplete Data. CRC Press, 2013.

[60] P. D. Allison and T. Oaks, “Missing data: Quantitative applications in the social sciences,” Br. J. Math. Stat. Psychol., vol. 55, no. 1, pp. 193–196, 2002.

[61] J. Chen, W. Li, A. Lau, J. Cao, and K. Wang, “Automated Load Curve Data Cleansing in Power Systems,” IEEE Trans. Smart Grid, vol. 1, no. 2, pp. 213–221, Sep. 2010.

[62] S. Weckx, R. D’Hulst, and J. Driesen, “Voltage Sensitivity Analysis of a Laboratory Distribution Grid With Incomplete Data,” IEEE Trans. Smart Grid, vol. PP, no. 99, pp. 1–1, 2015.

[63] J. Peppanen, M. J. Reno, M. Thakkar, S. Grijalva, and R. G. Harley, "Leveraging AMI Data for Distribution System Model Calibration and Situational Awareness," IEEE Trans. Smart Grid, vol. PP, no. 99, pp. 1–1, 2015.

[64] G. Mateos and G. B. Giannakis, "Load Curve Data Cleansing and Imputation Via Sparsity and Low Rank," IEEE Trans. Smart Grid, vol. 4, no. 4, pp. 2347–2355, Dec. 2013.

[65] D. Saez, F. Avila, D. Olivares, C. Canizares, and L. Marin, "Fuzzy Prediction Interval Models for Forecasting Renewable Resources and Loads in Microgrids," Smart Grid IEEE Trans. On, vol. 6, no. 2, pp. 548–556, Mar. 2015.

[66] C. Wan, Z. Xu, Y. Wang, Z. Y. Dong, and K. P. Wong, "A Hybrid Approach for Probabilistic Forecasting of Electricity Price," IEEE Trans. Smart Grid, vol. 5, no. 1, pp. 463–470, Jan. 2014.

[67] D. Lee and R. Baldick, "Short-Term Wind Power Ensemble Prediction Based on Gaussian Processes and Neural Networks," IEEE Trans. Smart Grid, vol. 5, no. 1, pp. 501–510, Jan. 2014.

[68] Y. Goude, R. Nedellec, and N. Kong, "Local Short and Middle Term Electricity Load Forecasting With Semi-Parametric Additive Models," IEEE Trans. Smart Grid, vol. 5, no. 1, pp. 440–446, Jan. 2014.

[69] L. Ghelardoni, A. Ghio, and D. Anguita, "Energy Load Forecasting Using Empirical Mode Decomposition and Support Vector Regression," IEEE Trans. Smart Grid, vol. 4, no. 1, pp. 549–556, Mar. 2013.

[70] Motamedi, H. Zareipour, and W. D. Rosehart, "Electricity Price and Demand Forecasting in Smart Grids," IEEE Trans. Smart Grid, vol. 3, no. 2, pp. 664–674, Jun. 2012.

[71] N. Amjady, F. Keynia, and H. Zareipour, "Short-Term Load Forecast of Microgrids by a New Bilevel Prediction Strategy," IEEE Trans. Smart Grid, vol. 1, no. 3, pp. 286–294, Dec. 2010.

[72] M. Majidpour, C. Qiu, P. Chu, R. Gadh, and H. R. Pota, "Modified pattern sequence-based forecasting for electric vehicle charging stations," in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2014, pp. 710–715.

[73] M. Majidpour, C. Qiu, P. Chu, R. Gadh, and H. R. Pota, "Fast Prediction for Sparse Time Series: Demand Forecast of EV Charging Stations for Cell Phone Applications," *IEEE Trans. Ind. Inform.*, vol. 11, no. 1, pp. 242–250, Feb. 2015.

[74] T. Krishnamurti, D. Schwartz, A. Davis, B. Fischhoff, W. B. de Bruin, L. Lave, and J. Wang, "Preparing for smart grid technologies: A behavioral decision research approach to understanding consumer expectations about smart meters," *Energy Policy*, vol. 41, pp. 790–797, Feb. 2012.

[75] M. Bae, H. Kim, E. Kim, A. Y. Chung, H. Kim, and J. H. Roh, "Toward electricity retail competition: Survey and case study on technical infrastructure for advanced electricity market system," *Appl. Energy*, vol. 133, pp. 252–273, Nov. 2014.

[76] E. McKenna, I. Richardson, and M. Thomson, "Smart meter data: Balancing consumer privacy concerns with legitimate applications," *Energy Policy*, vol. 41, pp. 807–814, Feb. 2012.

[77] E. B. Iversen, J. M. Morales, and H. Madsen, "Optimal charging of an electric vehicle using a Markov decision process," *Appl. Energy*, vol. 123, pp. 1–12, Jun. 2014.

[78] P. A. Mathew, L. N. Dunn, M. D. Sohn, A. Mercado, C. Custudio, and T. Walter, "Big-data for building energy performance: Lessons from assembling a very large national database of building energy use," *Appl. Energy*, vol. 140, pp. 85–93, Feb. 2015.

[79] Z. Xu, Z. Hu, Y. Song, W. Zhao, and Y. Zhang, "Coordination of PEVs charging across multiple aggregators," *Appl. Energy*, vol. 136, pp. 582–589, Dec. 2014.

[80] C. Rottondi, G. Verticale, and C. Krauss, "Distributed Privacy-Preserving Aggregation of Metering Data in Smart Grids," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 7, pp. 1342–1354, Jul. 2013.

[81] A. Rial and G. Danezis, "Privacy-preserving Smart Metering," in *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, New York, NY, USA, 2011, pp. 49–60.

[82] Z. Erkin, J. R. Troncoso-Pastoriza, R. L. Lagendijk, and F. Perez-Gonzalez, "Privacy-preserving data aggregation in smart metering systems: an overview," *IEEE Signal Process. Mag.*, vol. 30, no. 2, pp. 75–86, Mar. 2013.

[83] G. Kalogridis and S. Z. Denic, “Data Mining and Privacy of Personal Behaviour Types in Smart Grid,” in *2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW)*, 2011, pp. 636–642.

[84] G. Kalogridis, R. Cepeda, S. Z. Denic, T. Lewis, and C. Efthymiou, “ElecPrivacy: Evaluating the Privacy Protection of Electricity Management Algorithms,” *IEEE Trans. Smart Grid*, vol. 2, no. 4, pp. 750–758, Dec. 2011.

[85] Y. Lindell and B. Pinkas, “Privacy Preserving Data Mining,” in *Advances in Cryptology — CRYPTO 2000*, M. Bellare, Ed. Springer Berlin Heidelberg, 2000, pp. 36–54.

[86] R. Agrawal and R. Srikant, “Privacy-preserving Data Mining,” in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2000, pp. 439–450.

[87] M. Shaneck, Y. Kim, and V. Kumar, “Privacy Preserving Nearest Neighbor Search,” in *Machine Learning in Cyber Trust*, Springer US, 2009, pp. 247–276.

[88] Z. Yang, S. Zhong, and R. N. Wright, “Privacy-Preserving Classification of Customer Data without Loss of Accuracy,” in *In SIAM SDM*, 2005, pp. 21–23.

[89] H. Yu, X. Jiang, and J. Vaidya, “Privacy-preserving SVM Using Nonlinear Kernels on Horizontally Partitioned Data,” in *Proceedings of the 2006 ACM Symposium on Applied Computing*, New York, NY, USA, 2006, pp. 603–610.

[90] K. Chaudhuri and C. Monteleoni, “Privacy-preserving logistic regression,” in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 289–296.

[91] J. G. De Gooijer and R. J. Hyndman, “25 years of time series forecasting,” *Int. J. Forecast.*, vol. 22, no. 3, pp. 443–473, 2006.

[92] Mahmoudi, M.; Dong, J.; Tomsovic, K.; Djouadi, S., "Application of distributed control to mitigate disturbance propagations in large power networks," in North American Power Symposium (NAPS), 2015 , vol., no., pp.1-6, 4-6 Oct. 2015

[93] Nazaripouya H.; Wang Y.; Chu P.; Pota H.R.; and Gadh R., "Optimal Sizing and Placement of Battery Energy Storage in Distribution System Based on Solar Size for Voltage Regulation", 2015 IEEE PES General Meeting, Denver, Colorado, 26-30 July 2015.

[94] Ghanbarzadeh, A.; Noghrehabadi, A.R.; Assareh, E.; Behrang, M.A., "Solar radiation forecasting based on meteorological data using artificial neural networks," in Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conf. on Industrial Informatics, vol., no., pp.227-231, 23-26 June 2009.

[95] Srivastava, S.; Bhardwaj, S.; Sastri, O.S., "A novel hybrid model for solar radiation prediction," Emerging Trends in Electrical Engineering and Energy Management (ICETEEEM), 2012 International Conference on , vol., no., pp.243,248, 13-15 Dec. 2012.

[96] Assi, A.; M. Al-Shamisi, and M. Jama. "Prediction of monthly average daily global solar radiation in Al Ain City–UAE using artificial neural networks." In Proceedings of the 25th European photovoltaic solar energy conference, pp. 508-512. 2010.

[97] Liu, J.; Fang, W.; Zhang, X.; Yang, C., "An Improved Photovoltaic Power Forecasting Model With the Assistance of Aerosol Index Data," in Sustainable Energy, IEEE Transactions on , vol.6, no.2, pp.434-442, April 2015.

[98] 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," in Sustainable Energy, IEEE Transactions on , vol.5, no.3, pp.917-926, July 2014.

[99] Sharma, N.; Sharma, P.; Irwin, D.; Shenoy, P., "Predicting solar generation from weather forecasts using machine learning," in Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on , vol., no., pp.528-533, 17-20 Oct. 2011.

[100] Cao, S., "Total Daily Solar Irradiance Prediction using Recurrent Neural Networks with Determinants," in Power and Energy Engineering Conference (APPEEC), 2010 Asia-Pacific , vol., no., pp.1-4, 28-31 March 2010.

[101] Huang, R.; Huang, T.; Gadh, R.; Li, N., "Solar generation prediction using the ARMA model in a laboratory-level micro-grid," in Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on , vol., no., pp.528-533, 5-8 Nov. 2012.

[102] Zhang, N.; Behera, P.K.; Williams, C., "Solar radiation prediction based on particle swarm optimization and evolutionary algorithm using recurrent neural networks," in Systems Conference (SysCon), 2013 IEEE International , vol., no., pp.280-286, 15-18 April 2013.

[103] Yang, X.; Jiang, F.; Liu H., "Short-term solar radiation prediction based on SVM with similar data," in Renewable Power Generation Conference (RPG 2013), 2nd IET , vol., no., pp.1-4, 9-11 Sept. 2013.

[104] Iqdour, R.; Zeroual, A., "A rule based fuzzy model for the prediction of solar radiation." Revue des energies renouvelables 9, no. 2, 2006.

[105] Capizzi, G.; Bonanno, F.; Napoli, C., "A wavelet based prediction of wind and solar energy for Long-Term simulation of integrated generation systems," in Power Electronics Electrical Drives Automation and Motion (SPEEDAM), 2010 International Symposium on , vol., no., pp.586-592, 14-16 June 2010.

[106] Ruffing, S.M.; Venayagamoorthy, G.K., "Short to Medium Range Time Series Prediction of Solar Irradiance Using an Echo State Network," Intelligent System Applications to Power Systems, 2009. ISAP '09. 15th International Conference on, vol., no., pp.1,6, 8-12 Nov. 2009.

[107] Ferrari, S.; Lazzaroni, M.; Piuri, V.; Cristaldi, L.; Faifer, M., "Statistical models approach for solar radiation prediction," in Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE International , vol., no., pp.1734-1739, 6-9 May 2013.

[108] Soman, S.S.; Zareipour, H.; Malik, O.; Mandal, P., "A review of wind power and wind speed forecasting methods with different time horizons," in North American Power Symposium (NAPS), 2010 , vol., no., pp.1-8, 26-28 Sept. 2010.

[109] Nazaripouya H.; Wang B.; Wang Y.; Chu P.; Pota H.R.; and Gadh R., "Univariate Time Series Prediction of Solar Power Using a Hybrid Wavelet-ARMA-NARX Prediction Method", accepted in 2016 IEEE PES T&D, Dallas, Texas, 2-5 May 2016.

[110] Cococcioni, M.; D'Andrea, E.; Lazzerini, B., "24-hour-ahead forecasting of energy production in solar PV systems," in Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on , vol., no., pp.1276-1281, 22-24 Nov. 2011.

[111] Marafiga, E.B.; Farret, F.A.; Peixoto, N.H., "Effects of the seasonal sunlight variation on predictions of the solar-aeolic potential for power generation," in European Energy Market

(EEM), 2015 12th International Conference on the European Energy Market (EEM), vol., no., pp.1-5, 19-22 May 2015.

[112] M. Majidpour, C. Qiu, P. Chu, R. Gadh, H. Pota, "Treatment of Missing Values in Electric Vehicle Charging Load Prediction", submitted to *IEEE Trans. on Smart Grid*.

[113] M. Majidpour, C. Qiu, P. Chu, H. R. Pota, R. Gadh, "Forecasting the EV Charging Load based on Customer Profile or Station Measurement?", Elsevier *Applied Energy* 163 (2016): 134-141.

[114] M. Majidpour, H. Nazaripouya, P. Chu, H. R. Pota, R. Gadh, "Fast Univariate Time Series Prediction of Solar Power for Real-Time Control of Energy Storage System at UCLA Campus", to appear in Springer Technology and Economics of Smart Grids and Sustainable Energy.

[115] M. Majidpour, P. Chu, R. Gadh, H. R. Pota, "Incomplete Data in Smart Grid: Treatment of Missing Values in Electric Vehicle Charging Data", In *Connected Vehicles and Expo (ICCVE), 2014 International Conference on*, pp. 1035-1040. IEEE, 2014.