Comparison of Classical and Nonlinear Models for Short-Term Electricity Price Prediction

Elaheh Fata, Igor Kadota and Ian Schneider

***Abstract*—Electricity is bought and sold in wholesale markets at prices that fluctuate significantly. Short-term forecasting of electricity prices is an important endeavor because it helps electric utilities control risk and because it influences competitive strategy for generators. As the "smart grid" grows, short-term price forecasts are becoming an important input to bidding and control algorithms for battery operators and demand response aggregators. While the statistics and machine learning literature offers many proposed methods for electricity price prediction, there is no consensus supporting a single best approach. We test two contrasting methods for predicting electricity prices–regression decision trees and recurrent neural networks (RNNs)–and compare them to a more traditional ARIMA implementation. We conduct the analysis on a challenging dataset of electricity prices from ERCOT, in Texas, where price fluctuation is especially high. We find that regression decision trees in particular achieves high performance compared to the other methods, suggesting that regression trees should be more carefully considered for electricity price forecasting.**

***Index Terms*—Electricity Markets, Smart Grid, Machine Learning, Statistics, Decision Trees, Neural Networks.**

I. INTRODUCTION

THE cost of supplying electricity varies constantly according to factors like demand, fuel prices, and the availability of power plants and renewable energy. Demand is a significant driver of the electricity price which itself depends on a variety of factors such as time of the day, day of the week, and the weather. As renewable energy, e.g. wind and solar energy, is increasingly incorporated into the electricity grid, it adds new uncertainty to electricity provision. For instance, if wind blows in a given hour then the short-term price of supplying electricity decreases.

In recent years, electricity markets in different countries have been deregulated, allowing for the introduction of dynamic pricing. For most residential consumers in the United States, a local utility procures energy on their behalf through wholesale electric power markets with varying rates. However, in some markets, commercial and industrial customers are directly exposed to wholesale rates.

Long-term forecasting is used by power utilities for risk management and investment profitability analysis, while short-term forecasting has been used by consumers and producers to derive bidding strategies. Much attention has been paid to new "smart grid" technologies that could improve grid flexibility, like batteries for energy storage or technologies to engage customers in demand response. Operators of renewable energy generation might also use price prediction to determine their optimal energy offerings, given some uncertainty about the availability of energy in the near future [1] [2]. The success of these technologies depends, in part, on their ability to predict short-term electricity prices and optimize their operation in order to profit from fluctuations in energy prices. For this reason, the interest in the development of price forecasting tools has increased [3]–[17]. **Because of its growing importance for energy management and the "smart grid", the focus of this paper is on short-term forecasting**.

For planning purposes, most system operators run two sequential electricity markets: day-ahead and real-time. The day-ahead energy market allows participants to secure the price of electricity one day in advance and hedge against real-time price fluctuations through a forward contract. The real-time market allows participants to buy and sell electricity throughout the operation day. Figure 1 illustrates the evolution of the day-ahead and real-time prices over time in the PJM region, which covers much of the mid-Atlantic region in the United States.

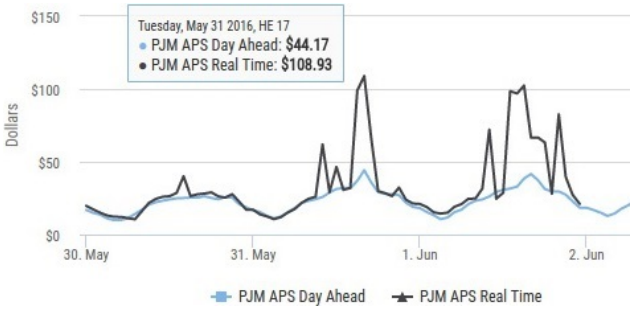

Fig. 1. Comparison of day-ahead and real-time prices from PJM in the period between 05/30/2016 and 06/02/2016. Figure from the website.

In most U.S. markets, the day-ahead market for electricity is cleared hourly, and the the real-time price for electricity varies every hour or every fifteen minutes. The real-time price can be significantly different from the day-ahead price due to unexpected fluctuations in the demand or because of a malfunctioning components of the grid (network outage). In general, electricity prices have a non-stationary mean and variance, which makes their prediction especially challenging [3]. **In this paper, we develop and compare algorithms that predict short-term real-time electricity prices** based on the available information, namely past prices, past demand, and current time/date information, wind predictions, and recent wind energy availability.

In the literature, three classes of techniques are used for predicting short-term electricity prices: classical statistical methods, data driven models and hybrid techniques. Classical statistical methods such as regression and Auto-Regressive

Location	Denmark	PJM	ERCOT
Mean	26.94	38.53	20.15
SD	13.08	44.71	**59.22**
Max	117.37	1722	**5040.40**

Fig. 2. Mean, Standard Deviation (SD), and Max Prices (all in USD) for electricity prices at single price nodes in three separate locations.

Integrated Moving Average (ARIMA) [4]–[9] are well-known and widely used to predict time-series data. The main benefit of such methods is their simplicity and possibility of further analysis due to their explicit prediction function. Data Driven models such as Neural Networks learn the structure of the problem, namely the relationship between input and output, from data. Some papers that utilize this technique are [10]–[13]. Hybrid techniques combine both approaches [3], [14], [15]. Literature surveys can be found in [16], [17]

This paper explores both classical statistical methods and data driven models. In particular, we develop prediction algorithms based on ARIMA, Regression Decision Trees and Neural Networks. We compare their performance in terms of prediction accuracy and then discuss their advantages and drawbacks with respect to feature engineering, computational complexity and suitability to the price prediction task.

The algorithms in this research are designed and evaluated based on datasets obtained from ERCOT, the operator for the electricity system in Texas. The datasets contain time-stamped information over a two-year period (2014 and 2015), including real-time prices, real-time demand, demand forecast, day-ahead prices set by the market and wind conditions. The ERCOT dataset is particularly challenging for electricity price analysis for two reasons. First, while its data is publicly available it is not simple to manage or use. We elaborate on this in Section II-B.

Second, the ERCOT data has a much higher maximum price than other markets in the United States and worldwide, and, partially as a result, it features very high price variance compared to other regions. For our dataset, the average price is $20.15 with a standard deviation of $59.22 and the maximum price is $5040, 250 times the average price! In some parts of Texas, taking into account variation in demand, the top 2% of most expensive hours make up 20% of total annual electricity costs [18]. Figure 2 shows a comparison of mean, standard deviation, and max prices for three regions from 2014-2015. Price data from Denmark and from PJM (in the Mid-Atlantic United States) are often used for testing forecasting models, but they offer significantly lower standard deviation (especially proportional to their mean) and significantly lower peak prices compared to ERCOT. Thus, ERCOT represents a more challenging dataset for electricity price prediction that has not been tested previously in the literature.

We focus on prediction of a specific nodal price. Nodal prices are calculated at every major intersection of the transmission network in ERCOT, there are about 500 of them in ERCOT. Much existing research focuses instead on regional average prices, which display significantly lower variance since they are an average over multiple locations. Thus, the prediction of nodal prices poses a much more significant learning challenge. However, it is also much more relevant, since the nodal prices are ultimately what is used for the settlement of electricity consumption by generators at their particular location.

The remainder of the paper is outlined as follows. In Sec. II, we formalize the objective of the price prediction task and describe the dataset in detail. In Sections III, IV and V we develop prediction algorithms based on ARIMA, Regression Decision Trees and Neural Networks, respectively, and discuss each method. The paper is concluded in Sec.VI.

II. PRICE PREDICTION MODEL

In this section, we describe the prediction model in detail. First, we present the loss function to be minimized, and then we describe the datasets that are utilized to train and validate the prediction algorithms.

A. Accuracy Metrics

Different metrics can be utilized to evaluate the performance of prediction algorithms. Let P_t^{real} and $P_t^{\text{prediction}}$ be the actual and predicted electricity price at time t. Using this notation we describe some well-known accuracy metrics [16].

Two basic metrics are the Absolute Error and the Absolute Percentage Error. The Absolute Error at time t is defined as $|P_t^{\text{real}} - P_t^{\text{prediction}}|$ and the Absolute Percentage Error is

$$\frac{|P_t^{\text{real}} - P_t^{\text{prediction}}|}{|P_t^{real}|} . \tag{1}$$

Two metrics that consider the accuracy over a time-horizon with a total of N predictions are the Mean Absolute Error and the Root Mean Squared Error. The Mean Absolute Error is given by

$$MAE = \frac{1}{N}\sum_{t=1}^{N} |P_t^{\text{real}} - P_t^{\text{prediction}}| , \tag{2}$$

and the Root Mean Square Error metric is:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N} \left(P_t^{\text{real}} - P_t^{\text{prediction}}\right)^2} . \tag{3}$$

MAE is widely used in the literature because of the extremely high variance of electricity prices. However, its derivative is non-continuous, which introduces additional challenges for implementation. We use both MAE and RMSE in this research.

B. Dataset

Our dataset contains time-stamped information, including:

- real-time prices in intervals of 15 minutes;
- real-time demand in intervals of 15 minutes;
- demand forecast in hourly intervals;
- day-ahead prices set by the market in hourly intervals;
- wind conditions and predictions in hourly intervals.

In Figures 3 and 4 we compare the different information in the data over two representative periods[1].

[1]In Figures 3 and 4 the demand data was re-scaled (divided by 1000) for the sake of clarity.

The goal of the algorithms developed in this report is to predict real-time prices. To do so the algorithms use information on *previous* prices P_n^{real}, demand D_n^{real} and wind conditions W_n, for $n < t$; and information on *previous and future* demand forecast D_n^{forecast} and day-ahead prices DA_n, for any time n. Algorithms should not rely on the entire historical data to predict price at time t. It is evident that the relevance of information from one hour is higher than the information from one month ago. For this reason, we define a past time-window W_P and say that at time t past information is available only in the interval $n \in \{t - W_P, \cdots, t-1\}$. Future information is only available up to a certain horizon. In particular, day-ahead prices are available only 24 hours in advance. Hence, we also define a future time-window W_F and say that at time t future information is available only in the interval $n \in \{t, \cdots, t+W_F\}$.

Thus, the **features** utilized to obtain $P_t^{\text{prediction}}$ are the real-time price, the real-time demand and the wind data in the interval $n \in \{t - W_P, \cdots, t-1\}$, in addition to the demand forecast and the day-ahead price in the interval $n \in \{t - W_P, \cdots, t+W_F\}$. Notice that W_F and W_P control the dimension of the input vector and, as a result, they influence running time of algorithms. One example of interest is $W_P = 30$ hours and $W_F = 4$ hours, for which algorithms can see information from the previous day and take advantage of some "periodic" events such as: at the beginning of business hours, the demand grows. Figure 3 shows that P_t^{real} and P_{t-24}^{real} can be correlated. Moreover, Figure 3 demonstrates that the real-time price is correlated to the demand forecast and day-ahead prices. Hence, having $W_F = 4$ should benefit the real-time price prediction. However, Figure 4 illustrates a case in which data seems to be not correlated, thus **emphasizing the challenge of predicting short-term electricity prices**.

It is worth mentioning that the ERCOT dataset does not have complete data on every hour in 2014 and 2015 and some files have missing information. We chose to adjust the implementation of our data algorithms to ignore missing data, and in calculating our final loss values we also ignored periods with missing data (this always amounted to less than 1% of total data). Due to the missing data, and some instances of repeated data, it was a significant challenge to clean and align the data, requiring hours of manpower and computational resources. This might explain why we haven't found any mention of price forecasting for the ERCOT data, even though it is feature-rich and publicly available.

III. ARIMA

In this section, we develop an ARIMA estimator based on the technique described in [9], [19]. The basic form of an ARIMA estimator is given by

$$P_t^{\text{prediction}} = \sum_{i=1}^{k} \alpha_i P_{t-i}^{\text{real}} + \varepsilon_t + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} \,, \tag{4}$$

where α_i and θ_j are parameters associated with the Auto-Regressive and Moving Average parts, respectively, and ε_t are iid Gaussian random variables with zero mean and variance σ^2. Notice that ARIMA relies only on the past real-time prices in order to forecast $P_t^{\text{prediction}}$. ARIMA is a very simple model and it does not use any of the exogenous features in our dataset.

To develop an ARIMA estimator, we follow a sequence of steps: 0) time-series data preprocessing, in particular, first order differencing; 1) model selection based on (4); 2) Maximum Likelihood estimation of the parameters α_i, θ_j and σ^2 using the training set; 3) data forecast $P_t^{\text{prediction}}$ using the trained ARIMA model on the validation set; 4) computation of the prediction error. The steps are repeated if the prediction error is unsatisfactory.

Recall form Section II-B that P_t^{real} is correlated with recent prices and also with the price at the same time in the previous day, P_{t-24}^{real}. From this intuition, a suitable ARIMA model would be

$$\begin{aligned} P_t^{\text{prediction}} = &\sum_{i=1}^{2} \alpha_i P_{t-i/4}^{\text{real}} + \alpha_3 P_{t-24}^{real} + \\ &+ \varepsilon_t + \sum_{j=1}^{2} \theta_j \varepsilon_{t-j/4} + \theta_3 \varepsilon_{t-24} \,. \end{aligned} \tag{5}$$

To predict P_t^{real}, this model leverages information from the real-time prices in the past half-hour $\{t-1/4, t-1/2\}$ and in the previous day $\{t-24\}$.

To estimate the parameters α_i, θ_j and σ^2 and evaluate the accuracy of the trained ARIMA model, we use a cross-validation technique called *forecast evaluation with a rolling*

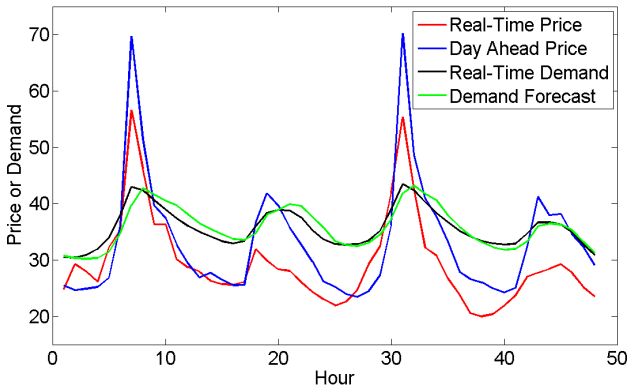

Fig. 3. Illustration of the ERCOT data with hourly information in the period between 01/15/2014 and 01/16/2014.

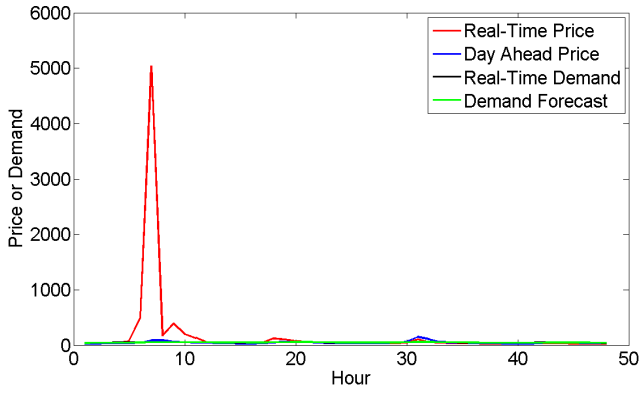

Fig. 4. Illustration of the ERCOT dataset with hourly information in the period between 01/06/2014 and 01/07/2014.

origin [20], see Figure 5. The idea is to train the parameters based on the ARIMA model in (5) and the values of P_t^{real} within the training window. Then, we use the trained ARIMA model to predict $P_t^{\text{prediction}}$ in the test window and compute the error $P_t^{\text{real}} - P_t^{\text{prediction}}$. Next, we slide both the training window and the test window by the length of the test window and repeat the process. When the windows reach the end of the dataset, we can compute the overall MAE (2) and RMSE (3) associated with the rolling Test Set. For the ERCOT dataset and the ARIMA model in (5) with $TrainingWindow = 20$ days and $TestWindow = 5$ days, we obtain $MAE = 5.09$ and $RMSE = 23.39$.

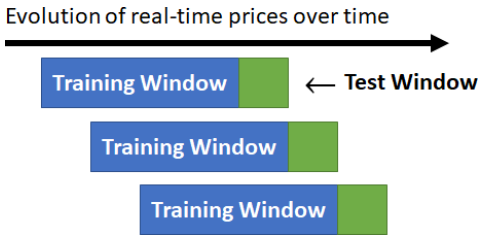

Fig. 5. Illustration of the cross-validation technique utilized for ARIMA.

Figure 6 displays the evolution of both P_t^{real} and $P_t^{\text{prediction}}$ over time. In general, the ARIMA model accurately predicts the behavior of the real-time price. However, the prediction is noticeably less accurate when there are price peaks.

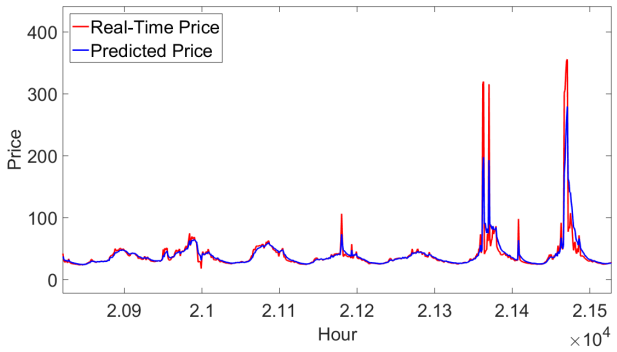

Fig. 6. Comparison of P_t^{real} and $P_t^{\text{prediction}}$ over time.

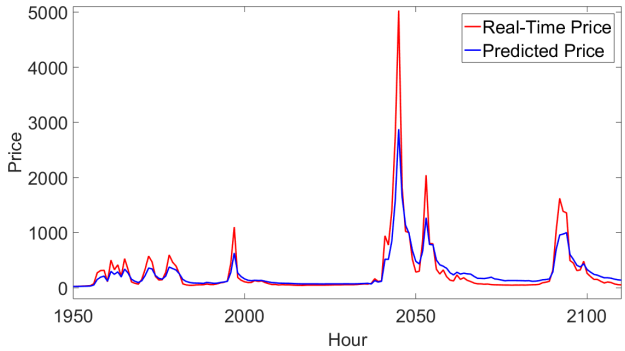

Fig. 7. Comparison of P_t^{real} and $P_t^{\text{prediction}}$ over time. The peak in this picture is the same as the one in Fig.4

The main advantage of ARIMA is that it is based on a simple linear function (4) that can be analyzed and can provide intuition into the prediction model. On the other hand, this simplicity is also its main disadvantage. In general, real-time prices are nonlinear in their features [16] which cannot be represented by the linear ARIMA model. Hence, nonlinear models such as Regression Decision Trees and Neural Networks might achieve better accuracies. Next, we discuss those two nonlinear models in depth.

IV. Regression Decision Trees

We now turn our focus to different variations of regression trees and how they can be used for real-time price forecasting. The problem with naively implementing a regression model for electricity price prediction is that there are important underlying patterns in the data which cannot be explored by simple regression. For instance, electricity consumption during weekdays in July should have similar patterns, while the consumption during a weekend in November should have a different pattern, mainly due to the weather conditions and business intensity. One way to explore those patterns is to utilize an unsupervised learning technique, such as decision trees, to separate the data into subclasses with similar electricity prices and then regress on each subclass. These methods can be seen in as similar to recent literature in electricity price prediction that combines SVM for classification with linear techniques for price prediction within each class [21]. However, the decision tree implementation has distinct advantages because it does not require the designer to specify the features that divide the data; instead it (ideally) chooses the features and classes that provide the most help for reducing the loss function.

Binary regression decision trees, bagged regression trees and boosted regression trees represent three widely used nonparametric approaches to predict demand and price [22]–[24]. The features we use in our model to predict the real-time price at time t are date/time information associated with t (i.e. year, month, day of the year, day of the week, hour, etc.), day-ahead prices starting from the last eight hours to four hours ahead of the prediction time t and real-time prices, wind, real-time demand and demand forecast information from the last eight hours. These particular time windows, namely $W_P = 8$ hours and $W_F = 4$ hours, were chosen by trying different time windows and observing the features that played important roles in the structure of decision trees.

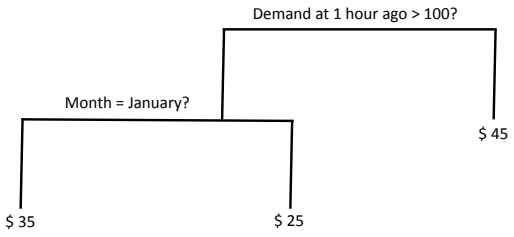

Fig. 8. Example of a binary decision tree.

Next, we briefly introduce regression trees, bagged regression trees and boosted regression trees. We refer the reader to [25] for additional information. A simple binary decision tree for price prediction is illustrated in Figure 8. We start at

the top of the tree and ask whether the electricity demand at the last hour was more then 100 units. If the answer is yes, we move to the right branch of the tree and predict the price of $45 for the current time; otherwise, we move to the left branch and ask whether the prediction is done for the month of January. If the answer is yes, we predict the price of $25, if the answer is no, our forecast is $35. As seen in the example, one main benefit of decision trees is their ease of interpretation.

A well-known problem associated with decision trees is over-fitting the training data by growing large and deep trees. We address this challenge in three different ways. In Section IV-A, we set bounds on model parameters such as minimum number of data elements in each leaf and maximum number of splits in the tree, in order to control the growth of the regression tree. The value of these bounds are found using cross-validation. In Section IV-B, we avoid over-fitting and reduce variance by implementing bagged regression trees. Bagging is an ensemble method in which the Training Set is divided into b sets using *sampling with replacement* and then a single regression tree is built for each of these b sets. Once all these trees are built, the prediction of an unseen data point is given by the average of the predictions of each of the b trees. This method is known to reduce prediction variance and to avoid over-fitting. Finally, in Section IV-C, we implement boosted regression trees and compare the result with both previous approaches. Boosted regression trees are an ensemble of weak prediction models that are iteratively designed to make a single strong prediction model.

A. Single Regression Decision Trees

In this section, we develop a single regression tree on the entire training set and optimize the tree's parameters using cross-validation. First, we randomly choose 70% of the ERCOT dataset as the training set and the remaining as the test set. Then, we use k-fold cross-validation to choose the best parameters for the regression tree. In the k-fold cross-validation, the training set is divided into k subsets. One of the k subsets is selected as the validation set and training is performed on the remaining $k-1$ sets. This process is repeated for each of the k subsets. The final validation result is obtained by averaging the k validations.

To avoid over-fitting we restrict two parameters of the decision tree: minimum number of data elements in each leaf and maximum number of splits in the tree. Notice that if the maximum number of splits is not bounded, the tree can potentially split until there is only one data element in leaf. Our approach is to first train an unrestricted tree on the training set during the 10-fold cross-validation process. We set the limit on the maximum number of splits as half the average number of splits in trained trees, i.e. 50. We observe that limiting the number of splits increases the cross-validation mean square error by 5.6%. For the minimum leaf size, we test different limits from 1 to 50 in the 10-fold cross-validation and observe that a bound of 6 provides good performance. Figure 9 shows the impact of minimum leaf size on mean absolute error (MAE). Observe that minimum leaf size of 6 provides the best cross-validation error. Furthermore, despite the fact that we train only a single tree, this method achieves very accurate test errors (MAE less than $1).

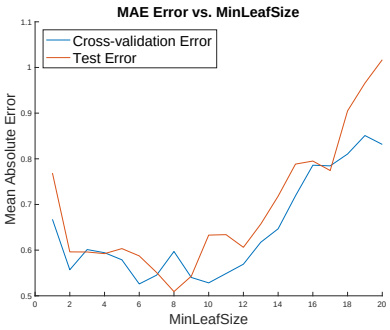

Fig. 9. Cross-validation and test MAEs for different bounds on min leaf size.

Lastly, we analyze the regression trees and verify which features were used more frequently in the decision process. Surprisingly, we observe that almost all branching was done on one feature: the price at two hours prior to the time of interest. Occasionally, other features such as time of the day and demand forecasts were used for branching but their role was not significant. This indicates that the two hours prior price feature is crucial in increasing information gain.

B. Bagged Regression Trees

Next we implement bagged regression trees, where data is divided into subsets and a regression tree is trained on each data subset independently; the output is the average of outputs of all these trees. Since bagging reduces variance to avoid over-fitting, we do not bound tree parameters and instead focus on finding the suitable number of trees to grow on the training set. For bagged regression trees it is conventional to use out-of-bag (OOB) analysis for cross-validation. Each tree in the bagged regression is trained on a subset of the training set, hence the remaining of the dataset is unknown to that tree and can be used as validation set to it. In OOB analysis, each training data point is tested on all the trees that did not have that point in their training set [26]. To find the optimal number of trees we find the mean absolute error of all training data points for bagged regression trees with size between 1 and 100 and pick the size associated to the smallest mean absolute OOB error, see Figure 10. It can be seen that bagged regression trees of size 60 has a reasonable mean absolute OOB error and adding more trees to the bag does not improve the model much, hence we bag 60 trees in our regression. The mean absolute OOB and test errors associated to the bagged regression trees are $0.89 and $1.03, respectively. Observe that the test error for the bagged regression trees compared to a single regression tree in Section IV-A is slightly larger; however, the difference is not substantial, and the bagged regression model could generalize better to new datasets.

Figure 11 shows the relative usage of features for branching in our bagged regression trees with 60 trees. It can be seen that one feature is used dominantly for branching and a few other features are used about one fifth as much and the rest of features are minimally used. The top five features in the order of significance are price at fifteen minutes ago, day of the year,

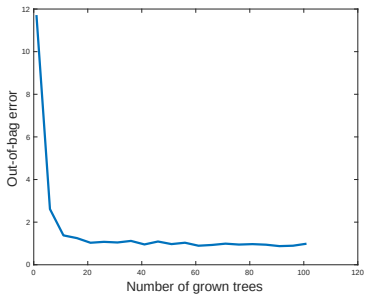

Fig. 10. Out-of-bag MAE for different number of trees.

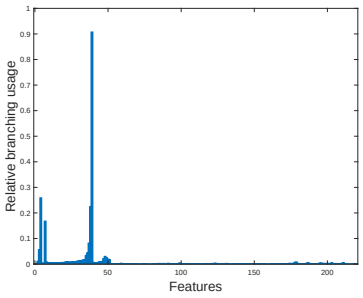

Fig. 11. Relative usage of feats. for branching in bagged regression trees with 60 trees.

price half an hour ago, time and price at forty five minutes ago. In comparison with a single regression tree, bagged regression trees use more features for branching, which suggests that it is a richer model and can be used when there are more complex features available in the data.

C. Boosted Regression Trees

In this section, we discuss boosted regression trees. Recall that boosted regression trees are an ensemble of weak prediction models that are iteratively designed to make a single strong prediction model which minimizes a loss function. In particular, we use *Least Square boosting* (LSboost), where the loss function is the squared error and the weak estimators are trees with bounded number of splits.

Let x and y denote the input feature and the correct prediction, respectively. At each stage i of boosting, LSboost improves the previous imperfect model F_{i-1} by adding a weak estimator h to it, i.e. $F_i(x) = F_{i-1}(x) + h(x)$. The process is as follows: initially there is no prediction model and the best prediction is the mean values of the data points, i.e., $F_0 = \bar{y}$. Then, at each subsequent stage, the goal is to find the estimator h that provides the best prediction of $y - F_{i-1}(x)$. Notice that $F_i(x) = F_{i-1}(x) + h(x) = y$ or, equivalently, $h(x) = y - F_{i-1}(x)$. Moreover, observe that $y - F_{i-1}(x)$ is the gradient of least square error $\frac{1}{2}(F(x) - y)^2$ with respect to $F(x)$. Hence, LSboost improves the model by minimizing the square error [27].

A modification to LSboost that yields remarkable improvement is to utilize shrinkage or learning rate. This modification changes the model into $F_i(x) = F_{i-1}(x) + \nu h(x)$, where $0 < \nu \leq 1$. It is known that small learning rates improve the performance of the model dramatically at the cost of slower training [27]. Next, we employ four different learning rates: 0.1, 0.25, 0.5 and 1.

As discussed earlier, the weak learners are trees with bounded number of splits. Since our weak learners are binary trees, the maximum number of splits in a tree is one less than the number of data points. Therefore, we run LSboost for different bounds on the maximum number of splits, namely between 1 and a quarter of the training data, to avoid substantial over-fitting. Moreover, to reduce over-fitting we perform 3-fold cross-validation on the data.

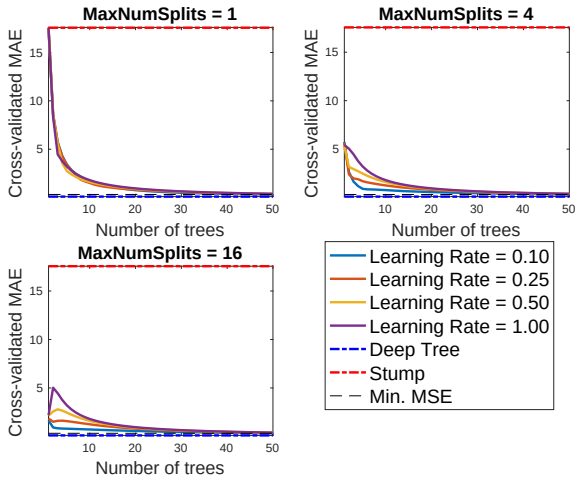

Fig. 12. Cross-validated MAEs for different learning rates and three different bounds on max number of splits of weak learners.

Figure 12 illustrates cross-validated mean absolute errors (MAE) for different number of boosting iteration using weak learners with maximum number of splits of 1, 4 and 16 and the four learning rates discussed above. Moreover, to better depict the performance of these boosted regression trees we plotted the MAE associated to deep trees (i.e., trees that exhaust the training data) and stumps (i.e., trees with only one splits). It can be seen that as the number of boosting iteration, i.e., number of trees, increases MAE improves. Moreover, using more sophisticated weak learners, that is, weak learners with higher bound on the number splits, results in smaller MAE, as expected. Deep trees will frequently over fit the data, while stumps do not have much power for accurate nonlinear prediction, as can be observed by their performance on our data. Lastly, we performed these boosted regression trees on the test data which provided similar results and accuracy.

Moreover, Figure 13 depicts the relative usage of features for branching in LSboost algorithms with 50 trees. Again there is one dominant feature in that respect and other features are not used as much. The five top features in the order of usage are last fifteen minutes price, last half an hour price down to last seventy five minutes ago.

Discussion: All the three variation of regression trees we tried provided excellent accuracy. In particular boosted trees obtained higher accuracy which might be due their iterative improvement of the model. In particular boosted regression trees with numerous weak learners have low test errors and using them we achieved MAE of $ 0.3 when 1024 weak learn-

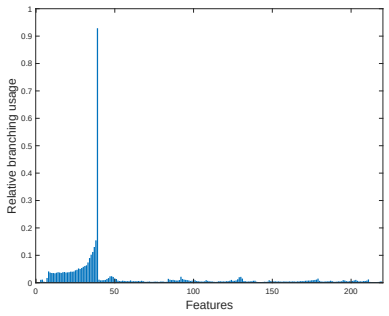

Fig. 13. Relative usage of feats. for LSboosting with 50 trees

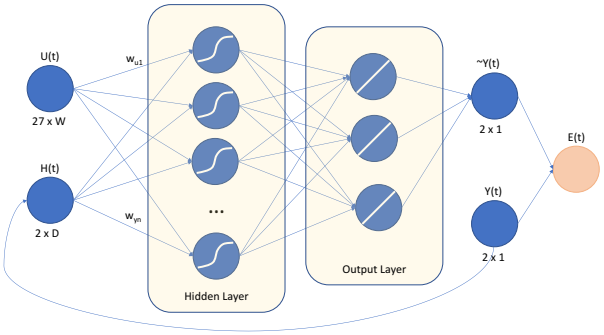

Fig. 14. Implemented RNN for electricity price prediction.

ers were used for LSboosting. Moreover, it was interesting that all these methods used recent prices as their branching features and the remaining features did not play a significant role in the structure of trees.

V. NEURAL NETWORKS

Variations of the Artificial Neural Networks (ANN) have been occasionally used in the literature for electricity price predictions due to their relatively high accuracy and capability of learning nonlinear relationships that are difficult to model with other methods. While "neural networks learns training data well, it may encounter large prediction errors in the test phase due to the time dependence of electricity prices " [3]. We decided to design and implement a Recurrent Neural Networks (RNN), these have been occasionally but not frequently used in the literature [17].

One of the main advantages of the RNN is that it tracks historical data, which allows it to automatically consider past data and predictions in its analysis. Thus it can provide nonlinear predictions related to both auto-regressive (AR) and moving average (MA) time-series predictions. By using a combination of historical data and exogenous inputs, the RNN is able to both consider current data and past prices when predicting current prices. However, in implementing the ARIMA and decision tree methods above, we use prepossessing to explicitly include previous values of certain features in the feature-list. There are two main disadvantages of RNNs for the purposes of short-term electricity price prediction. First, since it builds implicit functions, "further analysis on the function forms such as sensitivity analysis is difficult" [5]. This is true for all neural networks, not just RNNs. Second, RNNs suffer from the vanishing gradient problem, whereby it can be challenging to train their reaction to long-histories of data because the gradient will tend to vanish as you consider the effect of weights that operate on data farther back in the history.

We implemented a multi-layer RNN with exogenous inputs, see Figure 14. We started with the base implementation offered by MATLAB and created and adjusted our code to focus the implementation for short-term price forecasting, including by the use of MAE. The exogenous input $U(t)$ at each time step contains 27 features, which includes the current time and date, day-ahead prices, demand forecast information, and wind forecast information, i.e, all relevant data except for real-time price and real-time demand. The variable W refers to the width (in terms of time) of exogenous inputs to include, so the prediction for price at time t would use the exogenous inputs within times $(t-W,...,t)$. Unless otherwise noted, W is fixed as 16. The output $Y(t)$ contains real-time prices. Note that $Y(t)$ represents a $1\times(N+1)$ vector of the prices at N locations and system demand at time t. Typically, N is set to one, when are only trying to predict the price at a single location. The history $H(t)$ contains the prices and actual demand for the previous D periods, unless otherwise specified, $D=16$. The history is updated such that $H(t+1)$ is the concatenation of $Y(t+1)$ and all but the oldest value of $H(t)$.

In our implementation, the hidden nodes use the Sigmoid activation functions. Unless otherwise noted, there are 10 hidden units per layer, and a single hidden layer. We experimented with multiple hidden layers, and with varying numbers of hidden units per layer and did not see significant improvement. The output layer features one node with linear activation functions. Moreover, we use an early stopping technique based on the validation error. For a fixed p, if validation error increases for p successive time steps we stop the training process. This can serve as a form of regularization that improves generalization and performance for insufficiently regularized implementations.

In short, the RNN predictor described above is given by:

$$\hat{Y}(t) = f(U(t-W),...U(t),H(t);w) \tag{6}$$

where $f(x;w)$ is the function applied by the RNN as a result of the network structure described in Figure 14 for weights w. Thus the natural loss function of mean absolute error is

$$l(w) = \sum_{t=1}^{T} |\hat{Y}_i(t) - Y_i(t)|. \tag{7}$$

The results of our RNN are described in Figure 15. We initially utilized backpropagation to learn network parameters. This worked fine for our initial testing, but it proved to be very slow when early stopping was added to the model. As such, we utilized MATLAB's implementation of scaled conjugate gradient backprop [28] in order to speed up the results. This produced some of our best results when combined with three hidden layers and fairly relaxed early stopping criteria. Moeover, our early stopping technique successfully improved generalization.

Loss Function	Training Method	Other Features	RSE_{train}	$RMSE_{test}$	MAE_{train}	MAE_{test}
MSE (inc. demand)	Levenberg-Marquardt backpropagation	-	25.74	38.85	9.37	10.27
MSE	Levenberg-Marquardt backpropagation	-	31.38	41.87	8.14	8.80
MAE	backprop	p = 10	67.63	58.79	16.67	16.96
MAE	scaled conjugate gradient backprop	p = 50	53.13	78.79	6.78	7.42
MAE	scaled conjugate gradient backprop	p = 100	45.97	74.64	5.39	6.20
MAE	scaled conjugate gradient backprop	p = 200 M = 2	63.50	40.45	6.24	5.74
MAE	scaled conjugate gradient backprop	p = 100 M = 3	50.45	57.81	**4.99**	**5.38**

Fig. 15. Results for the RNN. Unless otherwise specified, the number of hidden layers $M = 1$, and early stopping regularization are unused. For the first line, we also include the demand error in the loss function, and the RNN seeks to predict both price and demand simultaneously.

In our best implementation (both in terms of training and test accuracy for the MAE loss function), the test set loss was only about 7% higher than the training set loss on the training. We want to add that we would have preferred to use deeper nets with less stringent stopping regularization, given that our test set loss did not start to diverge from the training set loss for any of our tested methods. However, in trying to expand to a deeper network, we encountered problems with insufficient memory, and reducing the early stopping criteria resulted in high increase in the running time without noticeable improvement. One effort to get around these challenges was to utilize a more efficient backpropogation algorithm, called Levenberg-Marquardt backpropogation [29]. Unfortunately, this algorithm turned out to be of limited use for us, because its MATLAB implementation does not allow for the use of loss functions besides MSE. When training with this algorithm, we successfully reduced the MSE of the training set well below what was achieved with other algorithms, see Figure 15. However, we also saw evidence of over-fitting as the test set error was 33% higher than training set error. Most importantly, as Levenberg-Marquardt algorithm does not optimize MAE, its mean absolute errors were significantly higher for both the training and test sets compared to our best implementations.

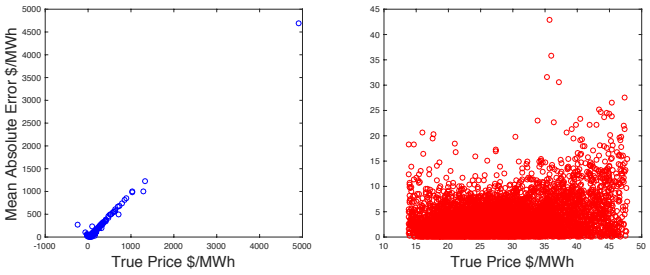

Fig. 16. The figure on the left shows true price versus mean absolute error for the test set for the best performing RNN implementation. The figure on the right excludes the 10% of data points that were the farthest outliers.

When analyzing how our model learns, we find that it learns well for most prices but struggles to predict price spikes in the network. Figure 16 shows the true price versus the mean absolute error for the best performing RNN implementation; we are interested in learning whether the RNN performs especially poorly when the true price is very low or very high. The graph on the left shows all data points, while the graph on the right excludes the 10% of the data points that were the farthest outliers. A predictor that performs equally well on all data points will have similar errors regardless of the underlying value. Taken together, these graphs suggest that the predictor accurately handles normal variation for most of the data points, but is unable to recognize or predict extremely high price periods.

Overall, as is typical with neural networks, we found that training parameters and initialization had important effects on performance. While the results for neural networks were not as impressive as we achieved using variations of regression decision tree models, this method shows promise for high-accuracy predictions, and could be improved further by continued adjustments and improvements to the implementation.

VI. CONCLUDING REMARKS

We implemented three different methods for electricity price prediction, ARIMA, regression decision trees, and recurrent neural networks. We trained and tested our implementations on a challenging set of electricity price data from ERCOT, Texas, that featured high variance data as compared to electricity prices in other areas. Moreover, there were some missing and repeated data points in this dataset. We were able to develop RNN implementations that performed about as well as the standard ARIMA model, which is very popular for electricity price prediction. However, the nonlinear time-series model offered by the RNN did not substantially improve performance in our implementations, serving as a reminder that tuning and training neural networks can prove to be difficult or fruitless. On the other hand, the regression tree models performed extremely well, predicting typical prices and outliers with higher success than most comparative models. They achieved a 90% reduction in test set error compared to the traditional ARIMA implementation, and they are comparable with the best methods in the electricity forecasting literature. While a direct comparison can not be made across datasets, our results show that regression tree implementations show high promise for accurate electricity price forecasting.

ACKNOWLEDGMENT

Igor Kadota led the ARIMA implementation, Elaheh Fata led the regression tree implementation, and Ian Schneider led the RNN implementation. MATLAB was used for all implementations. Thank you to Hoon Cho for his help and comments on the research.

REFERENCES

[1] I. Schneider and M. Roozbehani, "Endogenous error pricing for energy imbalance settlements," in *Proceedings of the American Control Conference*, vol. 2016-July, 2016.

[2] ——, "Energy Market Design for Renewable Resources: Imbalance Settlements and Efficiency-Robustness Tradeoffs," *IEEE Transactions on Power Systems*, 2017.

[3] F. Feijoo, W. Silva, and T. K. Das, "A computationally efficient electricity price forecasting model for real time energy markets," *Energy Conversion and Management*, vol. 113, no. Supplement C, pp. 27 – 35, 2016.

[4] H. Zareipour, C. A. Canizares, K. Bhattacharya, and J. Thomson, "Application of public-domain market information to forecast ontario's wholesale electricity prices," *IEEE Transactions on Power Systems*, vol. 21, no. 4, pp. 1707–1717, Nov 2006.

[5] H. Liu and J. Shi, "Applying arma-garch approaches to forecasting short-term electricity prices," *Energy Economics*, vol. 37, no. Supplement C, pp. 152 – 166, 2013.

[6] R. Weron and A. Misiorek, "Forecasting spot electricity prices: A comparison of parametric and semiparametric time series models," *International Journal of Forecasting*, vol. 24, no. 4, pp. 744 – 763, 2008, energy Forecasting.

[7] A. Escribano, J. I. Pena, and P. Villaplana, "Modelling electricity prices: International evidence," *Oxford Bulletin of Economics and Statistics*, vol. 73, no. 5, pp. 622–650, 2011.

[8] C. R. Knittel and M. R. Roberts, "An empirical examination of restructured electricity prices," *Energy Economics*, vol. 27, no. 5, pp. 791 – 817, 2005.

[9] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "Arima models to predict next-day electricity prices," *IEEE Power Engineering Review*, vol. 22, no. 9, pp. 57–57, Sept 2002.

[10] B. R. Szkuta, L. A. Sanabria, and T. S. Dillon, "Electricity price short-term forecasting using artificial neural networks," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 851–857, Aug 1999.

[11] H. Yamin, S. Shahidehpour, and Z. Li, "Adaptive short-term electricity price forecasting using artificial neural networks in the restructured power markets," *International Journal of Electrical Power & Energy Systems*, vol. 26, no. 8, pp. 571 – 581, 2004.

[12] J. Catalao, S. Mariano, V. Mendes, and L. Ferreira, "Short-term electricity prices forecasting in a competitive market: A neural network approach," *Electric Power Systems Research*, vol. 77, no. 10, pp. 1297 – 1304, 2007.

[13] W.-M. Lin, H.-J. Gow, and M.-T. Tsai, "Electricity price forecasting using enhanced probability neural network," *Energy Conversion and Management*, vol. 51, no. 12, pp. 2707 – 2714, 2010.

[14] J. Catalao, H. Pousinho, and V. Mendes, "Short-term electricity prices forecasting in a competitive market by a hybrid intelligent approach," *Energy Conversion and Management*, vol. 52, no. 2, pp. 1061 – 1065, 2011.

[15] G. Osorio, J. Matias, and J. Catalao, "Electricity prices forecasting by a hybrid evolutionary-adaptive methodology," *Energy Conversion and Management*, vol. 80, no. Supplement C, pp. 363 – 373, 2014.

[16] I. Vardakas, John S.and Zenginis, *A Survey on Short-Term Electricity Price Prediction Models for Smart Grid Applications*. Cham: Springer International Publishing, 2015, pp. 60–69.

[17] R. Weron, "Electricity price forecasting: A review of the state-of-the-art with a look into the future," *International Journal of Forecasting*, vol. 30, no. 4, pp. 1030 – 1081, 2014.

[18] I. Schneider and C. R. Sunstein, "Behavioral considerations for effective time-varying electricity prices," *Behavioural Public Policy*, vol. 1, no. 2, p. 219251, 2017.

[19] M. T. Hagan and S. M. Behr, "The time series approach to short term load forecasting," *IEEE Transactions on Power Systems*, vol. 2, no. 3, pp. 785–791, Aug 1987.

[20] R. J. Hyndman, "Measuring forecast accuracy," *Citeseer*, 2014.

[21] J. Zhang, J. Han, R. Wang, and G. Hou, "Day-ahead electricity price forecasting based on rolling time series and least square-support vector machine model," in *Control and Decision Conference (CCDC), 2011 Chinese*. IEEE, 2011, pp. 1065–1070.

[22] J. C. Reston Filho, A. Tiwari, and C. Dwivedi, "Understanding the drivers of negative electricity price using decision tree," in *Green Technologies Conference (GreenTech), 2017 Ninth Annual IEEE*. IEEE, 2017, pp. 151–156.

[23] Z. Yu, F. Haghighat, B. C. Fung, and H. Yoshino, "A decision tree method for building energy demand modeling," *Energy and Buildings*, vol. 42, no. 10, pp. 1637–1646, 2010.

[24] G. K. Tso and K. K. Yau, "Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks," *Energy*, vol. 32, no. 9, pp. 1761–1768, 2007.

[25] C. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[26] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[27] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[28] M. F. Mller, "A scaled conjugate gradient algorithm for fast supervised learning," *NEURAL NETWORKS*, vol. 6, no. 4, pp. 525–533, 1993.

[29] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.