Forecast of Electricity Wholesale Price in Victoria, Australia

by

Haosheng Luo

Abstract

This portfolio contains three projects that utilize standard linear models and machine learning models for prediction.

Project 1 is to forecast the 12-hour electricity wholesale price in Victoria, Australia, based on the high-frequency wholesale price and total demand data. The data was provided by a proprietary trading firm.

Project 2 is to implement the Fast False Selection Rate algorithm, that controls variable selection without simulation, and thus significantly raises the efficiency and reduces MSE.

Project 3 is to recreate the boundaries of police precincts on the Manhattan Island based on the parking violation database with a size around 9 million rows. The data can be found on NYC Open Data project.

Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

Abbreviations

0.0.1 Project 1

AEMO	Australia Energy Market Operator, an organization that manages Australia's National Electricity Market.
RRP	The settlement price of electricity by 30 minutes.
LCRRP	The logarithmatic change of RRP.
TOTD	The total demand of electricity by 30 minutes.
LCTOTD	The logarithmatic change of TOTD.
OSL	Optimal step length.
SARIMA	Seasonal Autoregressive Integrated Moving Average
PNN	Probabilistic Neural Network

0.0.2 Project 2

FSR	False selection rate

Acknowledgements

I extend a sincere gratitude to the helpful professors at Duke University for the systematic training in statistics and econometrics.

1

Forecast of the Electricity Wholesale Price in Victoria, Australia

The goal of this project is to model and predict the 30-minute transaction series, including the electricity settlement price (RRP) and total demand in Victoria, Australia in 2014. The electricity settlement price series exhibits enormous volatility and the total demand has a strong daily and weeekly combined seasonal pattern. While traditionally people use the seasonal autoregressive integrated moving average (SARIMA) models to forecast time-series data, current players in the most of deregulated electricity markets are using machine learning models, to minimize prediction errors stemmed from the non-linear and volatile nature of electricity price and demand series. Inspired by Spurio (2014), the method adpoted by Australia Energy Market Operator (AEMO) that oversees the domestic deregulated retail market, Zhang (2003) and Mehdi Khashei and Ardali (2012), we use artificial neural networks in this project. The modeling process can be separated into two parts: forecasting total demand and modeling the association between settlement price and total demand. To forecast the total demand, an one hidden-layer neural network is applied;

however, to forecast the settlement price, we use a hybrid model of SARIMA and neural network due to the complicated mechanism of settlement price. We select the proposed models based on Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Akaike's Information Criterion (AIC) and cross-validation technique. The error analysis also involves comparison with a naive model, which uses the current value as a forecast of the next value. This project also involves a dataset that contains public holidays in Victoria we create independently, that can be served as an exogenous variable. Based on the selected model, we provide a 12-hour forecast of settlement prices from January 1, 2015 00:00 am to January 1, 2015 11:30 am. Finally, it follows the discussion about variables and improvement.

1.1 Data Series

The huge volatility of the electricity price is caused mainly by the fact that we can not store the electricity, or transport it to another region due to either no existing passage or limited transportation capacity. Therefore, it is highly reliant on the balance between changing demand and predetermined supply, and can easily become either exceptionally high or negative under certain circumstance. Compared with the price, the demand volume is relatively more stable, but it can still be easily affected by weather and facility failures due to human errors or natural disasters, as is also suggested in J.P.S. Catalão and Ferreira (2007).

In Fig. 1.1 and Fig. 1.2 we can discover a major price and demand series fluctuation from January 14th to 17th, due to the 4 consecutive days of "heatwave not seen for 100 years" (Caldwell (2014)), and several minor fluctuations in days such as Feb 02, due to the similar hot weather (AEMO (2014)).

Selecting data in April, where the price series exhibits mild fluctuation, we observe a strong daily seasonality and weak weekly seasonality in the total demand series

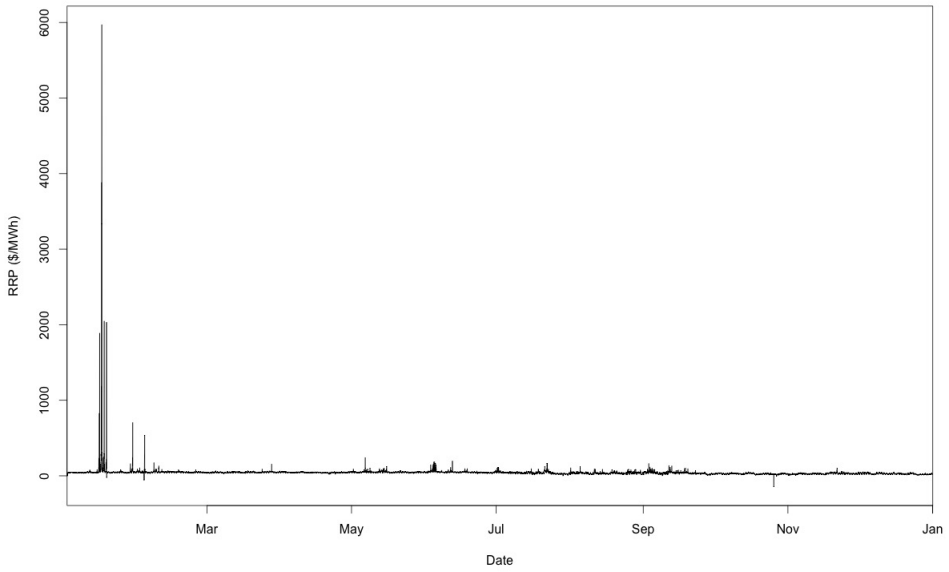

FIGURE 1.1: Electricity Settlement Price in 2014

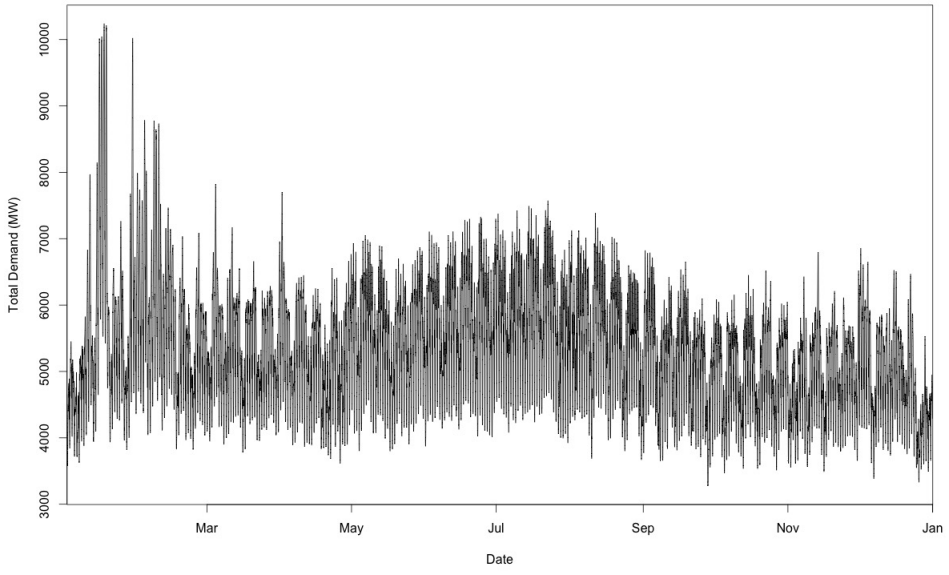

FIGURE 1.2: Electricity Total Demand in 2014

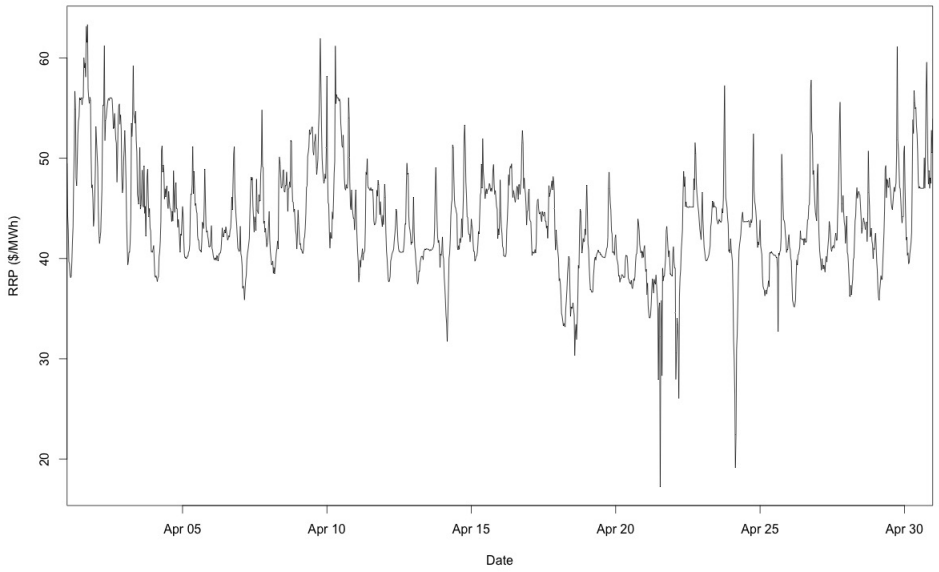

FIGURE 1.3: Electricity Settlement Price in April, 2014

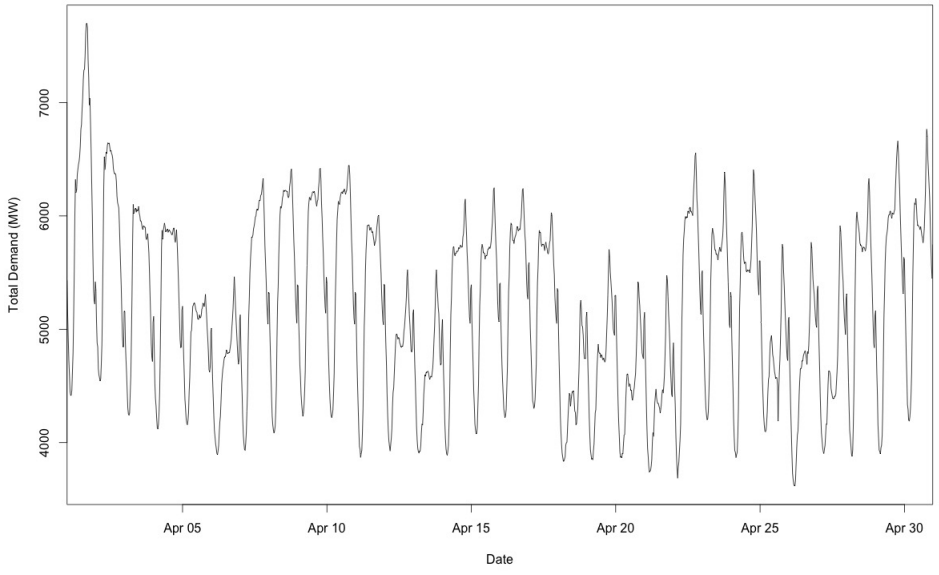

FIGURE 1.4: Electricity Total Demand in April, 2014

Table 1.1: 2014 Public Holidays in Victoria, Australia

Day	Date	Holiday	Key
Wednesday	January 01	New Years Day	National
Monday	January 27	Australia Day	National
Monday	March 10	Labor Day	Regional
Friday	April 18	Good Friday	National
Monday	April 21	Easter Monday	National
Friday	April 25	ANZAC Day	National
Monday	June 09	Queens Birthday	Regional
Tuesday	November 04	Melbourne Cup Day	Regional
Thursday	December 25	Christmas Day	National
Friday	December 26	Boxing Day	National

(Fig. 1.4). Both of them are less obvious in the price series (Fig. 1.3). We incorporate the weekly effect in the model by adding a dummy variable indicating weekday, and assume that the daily seasonality in the price series can be sufficiently captured by the daily seasonality in the demand series. Also, the holiday effect is modeled as a potential variables, as suggested in J.P.S. Catalão and Ferreira (2007). A public holiday database is generated in Table 1.1[1].

To address the problem of model instability caused by the huge volatility in the price series, we smooth the data of the major volatility event during Tuesday, January 14 to Friday, January 17, by replacing the total demand and price series with the average of demand and price at each half hour on weekdays. However, we choose to truncate the price spikes in the other minor events at 80 dollars, and try to model the occasional yet relatively milder price volatility through SARIMA and neural network hybrid models. Similar to most researches, we transform the price and demand series to the logarithmic scale and then difference it. Also, we replace the negative price with 0.01, whose log value can still be sufficiently small. In summary, we model the log change of price and demand in 2014.

[1] See http://www.officeholidays.com/countries/australia/victoria/2014.php

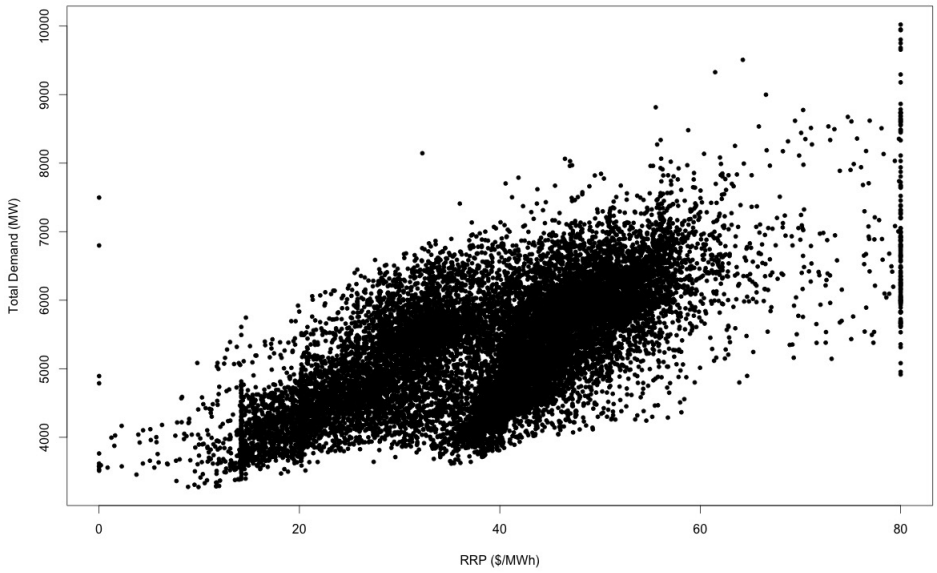

FIGURE 1.5: Correlation Between Settlement Price and Total Demand after Cleaning

It is worth noting that data smoothing and truncating will distort the pattern that is deeply seated in the series, which inevitably leads to a failure to forecast rare and extreme events. Therefore, it is crucial for traders and quants to closely monitor the market transaction and collect more market information. We will discuss the additional variables that are necessary in the last section of this project.

Fig. 1.5 shows a positive correlation between the total demand for electricity and settlement price after cleaning. This pattern indicates that we can use total demand as an exogenous variable in the SARIMA model to predict the settlement price.

1.2 Model Settings

1.2.1 Seasonal Autoregressive Integrated Moving Average Model

In the past several decades, autoregressive moving average models play as a dominant tool to forecast time-series data. A SARIMA model assumes that the generation

process of a stationary process can be formulated as:

$$\Phi(B^S)\phi(B)\Delta_S^D\Delta^d y_t = \Theta(B^S)\theta(B)w_t, \tag{1.1}$$

where

1. B is the lag operator
2. S is the seasonal period
3. y_t is the actual observation at t
4. $\Delta^d y_t$ is the dth order differenced series of y_t. If $d = 1$, then $\{\Delta^d y_t\} = \{y_t - y_{t-1}\}$.
5. $\Delta_S^D y_t$ is the Dth order differenced series of y_t by the period of S. If $D = 1$, then $\{\Delta_S^D y_t\} = \{y_t - y_{t-S}\}$
6. w_t is the random error at t, having a white noise process with a constant variance σ^2
7. The non-seasonal components are
   (a) AR: $\phi(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$
   (b) MA: $\theta(B) = 1 + \theta_1 B + \cdots + \theta_q B^q$
8. The seasonal components are
   (a) AR: $\Phi(B^S) = 1 - \Phi_1 B^S - \cdots - \Phi_P B^{PS}$
   (b) MA: $\Theta(B) = 1 + \Theta_1 B^S + \cdots + \Theta_Q B^{QS}$

and we write $y_t \sim ARIMA(p, d, q) \times (P, D, Q)_S$. In this project, we also add a group of exogenous variables in the SARIMA model for settlement price.

The prevalence of ARIMA model is largely due to its simple interpretation and implementation. Box-Jenkins (1976) proposes a three-step iterative method for the

modeling procedure: model identification, parameter estimation and result diagnose. The order of lag in the AR and MA component can be determined by the partial auto-correlation function (PACF) and auto-correlation function (ACF) respectively. Information criteria such as Akaike's information criterion (AIC) and the minimum description length can also be applied as complementary methods. In this project, we propose a group of models based on ACF and PACF plots and select the best one based on RMSE, MAE and AIC.

1.2.2 Artificial Neural Network

An artificial Neural network is a machine learning model that can be applied to both classification and regression problems. It demonstrates great flexibility and success to capture the non-linear pattern deeply seated in the data. A neural network does not need any specification for the form of the estimated functions, and therefore it can be regarded as a non-parametric modeling framework that mostly relies on the data characteristics. With the rapid technological improvement in computational capability and data storage, neural networks have become one of the dominant models in many areas of application.

Artificial neural network works like a human brain. When an event is recognized by different sensors, the signals generated by sensors will be propagated along the synaptic connections to the following layers of neurons, with different weights depending on the importance or strength of the connection. Finally they reach the output neuron. In the mathematical framework, the signals can be regarded as the variables of an observation, activation in each neuron can be quantified with activation functions and weights are tuned based on experience, or in other words, training data set. A typical structure of artificial neural networks with single hidden-layer is shown in Fig. 1.6[2].

[2] From the R-blogger site: Visualizing neural networks from the nnet package, by beckmw on

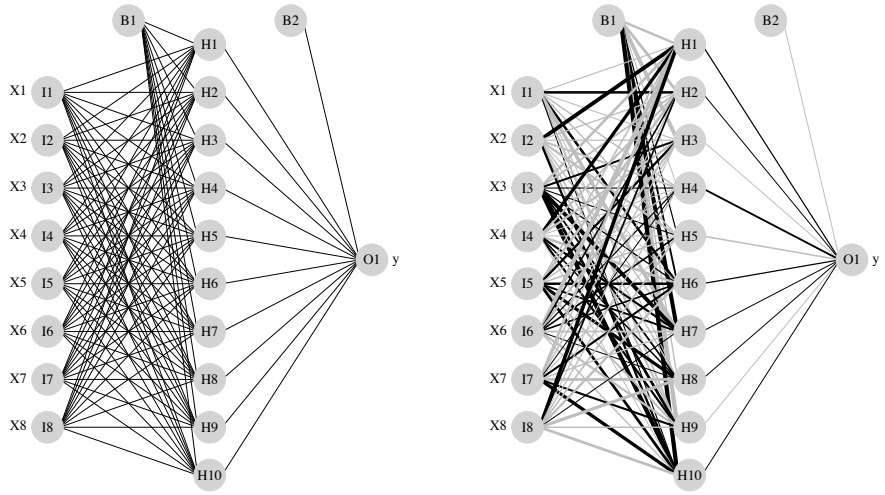

FIGURE 1.6: Structure of Neural Networks. The left image is a standard illustration of a neural network model and the right image is the same model illustrated as an interpretation diagram. The black lines represent positive weights and the grey lines represent negative weights.

A wide variety of researches have shown that a single hidden-layer neural network is sufficient for time-series forecasting. In Fig. 1.6, the left layer of neurons is called the input layer, the middle is the hidden layer and the right is the output layer. In this model, the output y_t and inputs $x_{t,k}$ have the mathematical relationship with the form of:

$$y_t = \alpha_0 + \sum_{j=1}^{q} \alpha_j g(\beta_{0j} + \sum_{i=1}^{p} \beta_{ij} x_{t,k}) + \varepsilon_t, \tag{1.2}$$

where p and q are the number of input and hidden nodes (neurons) respectively; $\alpha_j (j = 0, 1, 2 \cdots q)$ and $\beta_{ij} (i = 0, 1, 2 \cdots p, j = 0, 1, 2 \cdots q)$ are the weights of connection from the hidden layer to the output layer and connection from the input layer to the hidden layer respectively; and ε_t is the prediction error. For an artificial neural

March 4, 2013. http://www.r-bloggers.com/visualizing-neural-networks-from-the-nnet-package/

network, a logistic function is chosen as the activation function in the hidden layer:

$$g(x) = \frac{1}{1 + exp(-x)}. \tag{1.3}$$

The neural networks for classification differ from those for regression in the output layer. We assign one output node to each class and the neural networks compute an output value for each class. Without any prior information about the feature in each group, we select the class with the largest output value as the fitted or predicted result. Similar to standard statistical models, neural networks can also have over-fitting problem when the number of neurons in the hidden layer is too large. In practice, people set the number of hidden neurons around 2/3 of the total number of input variables. We also apply this rule in this project. A more general way to determine the number of hidden nodes can be cross-validation.

1.2.3 Formulation of the Hybrid Model for the Settlement Price

Zhang (2003) suggests that, "the approximation of ARIMA models to complicated non-linear series is not satisfactory, while neural networks also yield mixed results to linear problems, depending on factors including the noise level and sample size". Therefore, a hybrid model is used to overcome the deficiencies of using a single model.

In this project, we use the SARIMA model to fit the settlement price series and a neural network to detect the step shift of the price series by classifying the direction of residuals of SARIMA models. Then an optimal step length (OSL) is calculated that minimizes the additional variation the hybrid model fails to capture. Finally we apply OSL to the fitted or predicted values of SARIMA as our estimation. A more detailed procedure is described as follows:

1. Selecting and fitting the SARIMA model. In this step, the Box-Jenkins' method is applied for model selection. We choose the model based on RMSE, MAE

and AIC. Though better metrics of model validity can be out-of-sample MSE or MAE, cross-validation for time-series is not implemented in this project due to the computational concerns. The residuals are calculated as $\epsilon_t = y_t - \hat{y}_t$.

2. Classifying the residuals of SARIMA model. The desired level of error (DLE) is predetermined as the threshold to decide the direction of step shift. Given the huge volatility in the price series, we set DLE = 10%× MAE. MAE can be obtained in Step 1.

   (a) Price data with a residual that is greater than the desired level of error, $\{y_t|\epsilon_t > DLE\}$ are classified into category one, labeled with Direction = +1.

   (b) Price data with a residual that is smaller than the desired level of error yet larger than the negative desired level of error, $\{y_t||\epsilon_t| < DLE\}$ are classified into category two, labeled with Direction = 0.

   (c) Price data with a residual that is smaller than the negative desired level of error, $\{y_t|\epsilon_t < -DLE\}$ are classified into category three, labeled with Direction = -1.

3. Training a neural network classification model. Effective variables of a data point in this step can be:

   (a) Lags of the time-series data of price y_t

   (b) Lags of the residuals in the SARIMA model ϵ_t

   (c) The Fitted value and its lags in the SARIMA model $\hat{y}_t$

   (d) Exogenous variables at time t $x_{t,i}$

   Denote the output of neural network classifier as Tar_t, the fitted or predicted class of y_t.

4. Calculating the optimal step length (OSL). OSL is chosen such that the loss function

$$Z = \sum_{t=1}^{n} g(y_t - \hat{y}_t - Tar_t \times OSL) \tag{1.4}$$

is minimized. In this project, $g(x) = |x|$ is chosen, because it is robust to severe outliers and volatility, that the price series exhibits.

5. Applying the adjustment to the SARIMA predicted values. The fitted value of the hybrid model $\tilde{y}$ is

$$\tilde{y}_t = \hat{y}_t + Tar_t \times OSL. \tag{1.5}$$

1.3 Model Fitting and Prediction of the Total Demand

AEMO utilizes a single hidden-layer of neural network to fit the log change in total demand series. Given the relatively simpler pattern in the total demand, we apply AEMO's method in this project. Fixed inputs are selected according to Spurio (2014), including:

1. Four lags of the logarithmic changes in demand immediately prior to the period being predicted $\{y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}\}$.

2. Five lags of the logarithmic changes in demand prior to and including the period one week before the time being predicted $\{y_{t-336}, y_{t-337}, y_{t-338}, y_{t-339}, y_{t-340}\}$. [3].

We also test whether holiday and weekend have effects on the total demand of electricity. Set:

$$w_t = \begin{cases} 0.1 & \text{if it is a weekday} \\ 0 & \text{otherwise} \end{cases} \tag{1.6}$$

[3] The fixed inputs are similar to the seasonal AR model $(4,0) \times (1,0)_{366}$, where we set the seasonal period as one week.

Table 1.2: Error Analysis of the Neural Networks for Total Demand on the Original Scale

Model	RMSE	MAPE
M1	144.2	0.0216
M2	143.9	0.0216
M3	144.2	0.0216
M4	144.0	0.0216
M5 (Naive)	159.0	0.0233

$$h_t = \begin{cases} 0.1 & \text{if it is a public holiday} \\ 0 & \text{otherwise} \end{cases} \tag{1.7}$$

The dummy value 0.1 is chosen due to the computational concern in the neural network models.

Therefore, five models are compared in this part as follows:

1. **M1**: $y_t \sim y_{t-1}+y_{t-2}+y_{t-3}+y_{t-4}+y_{t-336}+y_{t-337}+y_{t-338}+y_{t-339}+y_{t-340}+w_t+h_t$
2. **M2**: $y_t \sim y_{t-1}+y_{t-2}+y_{t-3}+y_{t-4}+y_{t-336}+y_{t-337}+y_{t-338}+y_{t-339}+y_{t-340}+w_t$
3. **M3**: $y_t \sim y_{t-1}+y_{t-2}+y_{t-3}+y_{t-4}+y_{t-336}+y_{t-337}+y_{t-338}+y_{t-339}+y_{t-340}+h_t$
4. **M4**: $y_t \sim y_{t-1}+y_{t-2}+y_{t-3}+y_{t-4}+y_{t-336}+y_{t-337}+y_{t-338}+y_{t-339}+y_{t-340}$
5. **M5**: A naive model, that uses y_{t-1} as the predictor of y_t

Table 1.2 is the error analysis of the five models on the original data scale. It indicates that our neural network models uniformly have a better performance than the naive model does. However, there is no significant improvement by considering the weekend and holiday effects. Therefore, we favor the smaller model, M4, for prediction.

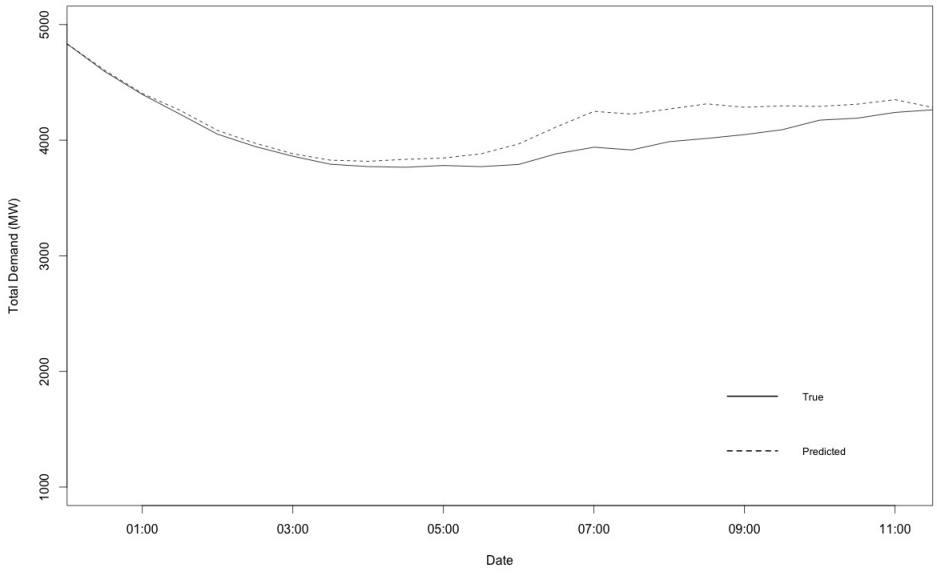

FIGURE 1.7: Comparison Between Predicted and True Total Demand Series

Fig. 1.7[4] demonstrates the predicted and true total demand series from January 1, 2015 00:00:00 am to January 1, 2015 11:30:00 am.

1.4 Model Fitting and Prediction of the Settlement Price

1.4.1 Fitting an SARIMA model

Not surprisingly, Fig. 1.8 and Fig. 1.9 show that the log change in settlement price still exhibits seasonality. Therefore we difference the log change of price series (LCRRP) by a day.

Their ACF and PACF shown in Fig. 1.10 and Fig. 1.11 suggest that a possible SARIMA model for the settlement price data is $ARIMA(0,0,4) \times (0,1,1)_{48}$, where 48 is the number of observations in a day. We also propose some variation on it. 7 models are fitted and tested as follows:

1. M1: ARIMA$(0,0,4) \times (0,1,1)_{48}$

[4] True data can be found in http://www.aemo.com.au/Electricity/Data/Price-and-Demand/Aggregated-Price-and-Demand-Data-Files/Aggregated-Price-and-Demand-2011-to-2016

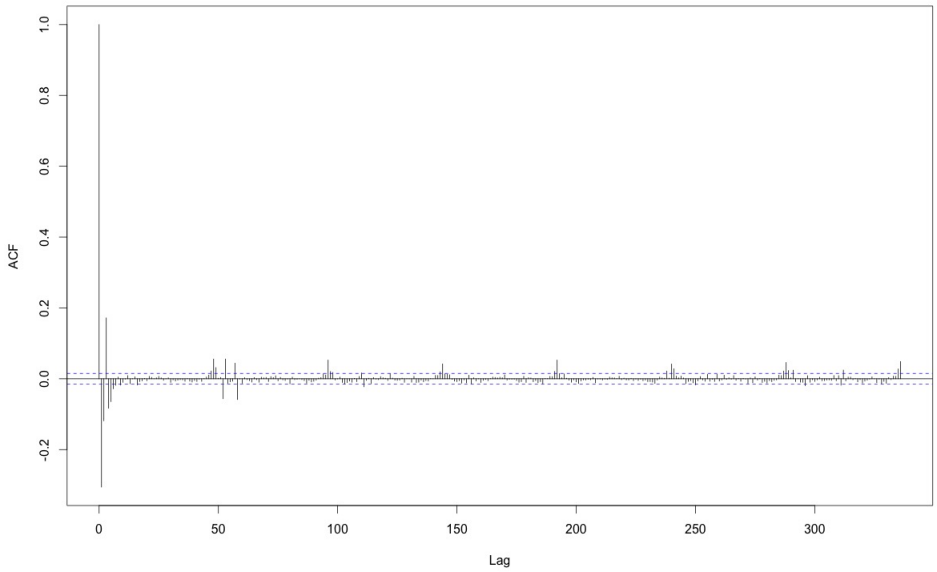

FIGURE 1.8: ACF of Log Change in Settlement Price

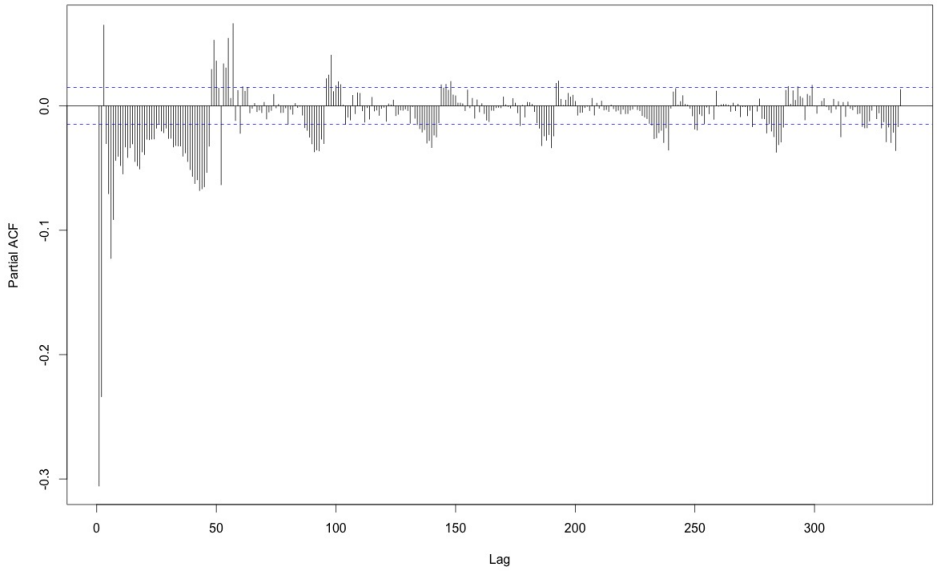

FIGURE 1.9: PACF of Log Change in Settlement Price

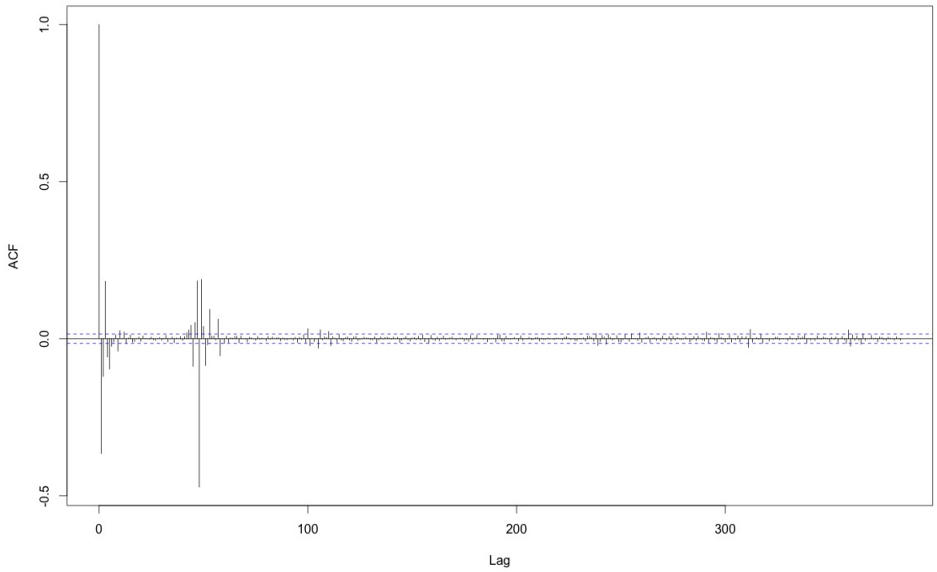

FIGURE 1.10: ACF of Differenced Log Change in Settlement Price

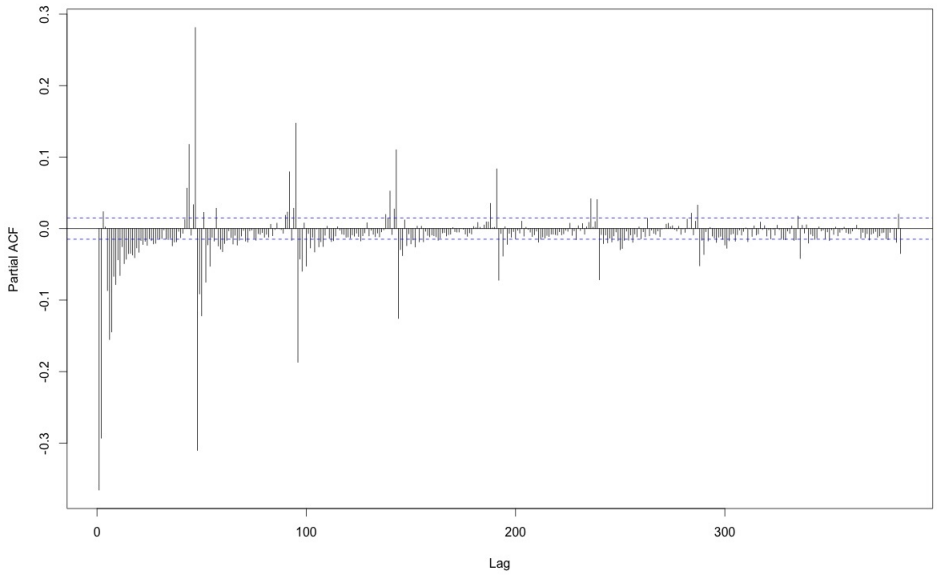

FIGURE 1.11: PACF of Differenced Log Change in Settlement Price

Table 1.3: Error Analysis of the SARIMA models for Settlement Price on the Logarithmic Scale

Model	RMSE	MAE	AIC
M1	0.1968	0.0729	-7036.93
M2	0.1931	0.0699	-7684.90
M3	0.1931	0.0699	-7684.53
M4	0.1930	0.0698	-7687.93
M5	0.1910	0.0695	-8051.48
M6	0.1910	0.0695	-8060.84
M7	0.1991	0.0679	-6616.04
M8 (Naive)	0.3662	0.0949	n/a

2. M2: ARIMA$(0,0,4) \times (0,1,1)_{48}$ with exogenous variable: LCTOTD

3. M3: ARIMA$(0,0,4) \times (0,1,1)_{48}$ with exogenous variables: LCTOTD, w_t and h_t

4. M4: ARIMA$(0,0,4) \times (0,1,2)_{48}$ with exogenous variable: LCTOTD

5. M5: ARIMA$(0,0,5) \times (0,1,1)_{48}$ with exogenous variable: LCTOTD

6. M6: ARIMA$(0,0,5) \times (0,1,2)_{48}$ with exogenous variable: LCTOTD

7. M7: ARIMA$(0,0,3) \times (0,1,2)_{48}$ with exogenous variable: LCTOTD

Table 1.3 is the error analysis of our proposed models, and one naive model as the benchmark.

The error analysis indicates that the SARIMA models uniformly have a better performance than the naive model does. Also, there is no significant improvement by considering the weekend and holiday effects, but log change of total demand does improve the model performance. Among our proposed models, M6: ARIMA$(0,0,5) \times (0,1,2)_{48}$ + LCTOTD has the smallest RMSE, MAE and AIC. The residual analysis, Fig. 1.12 and Fig. 1.13 show that the residual series remains seasonality, though it

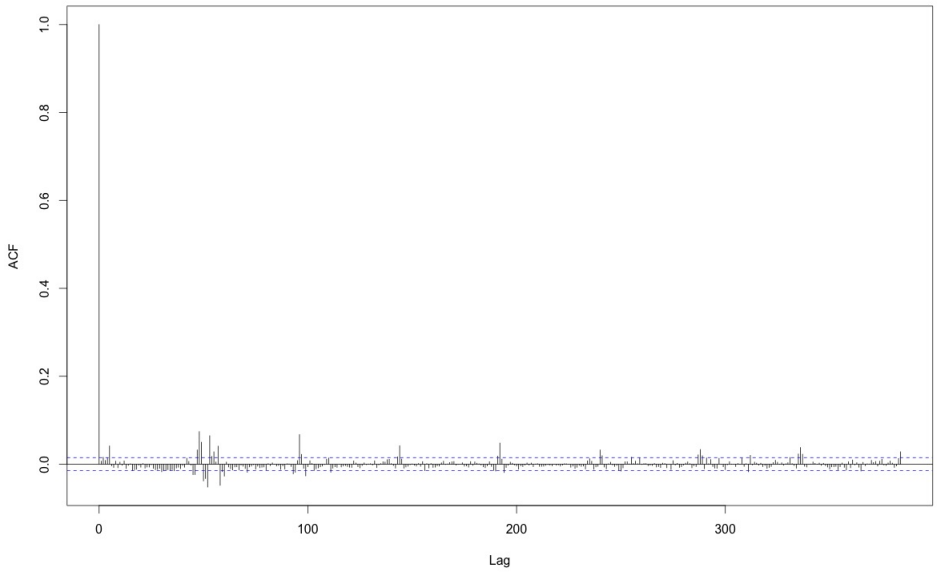

FIGURE 1.12: ACF of M6 Residuals

is much weaker. To minimize the prediction error and seasonality, our next step is to classify the residuals and fit them with an artificial neural network.

1.4.2 Fitting the SARIMA Residuals With an Artificial Neural Network

We first classify the direction of step shift in each residual with DLE, and then fit the step shift direction series with an artificial neural network. After multiple trials, we select an input set as follows:

1. Lags of LCRRP series in time t: $(y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-336}, y_{t-337}, y_{t-338}, y_{t-339}, y_{t-340})$, which are similar with the variables in the neural network model for total demand series.

2. LCTOTD, the log change of total demand in time t

3. Lags of SARIMA fitted series in time t: $(\hat{y}_t, \hat{y}_{t-1}, \hat{y}_{t-336}, \hat{y}_{t-337})$

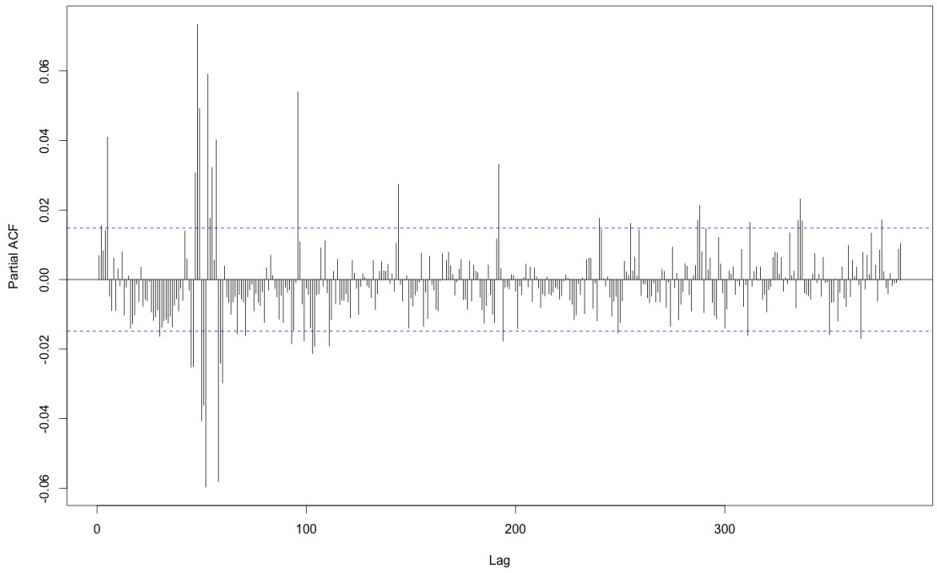

FIGURE 1.13: PACF of M6 Residuals

The mis-classification rate of the neural network is 24.19%. While it is not a decent rate, the hybrid model still perform much better from using a single model of either SARIMA or neural network. We will discuss the mis-classification problem in the conclusion of this project.

In step 4, OSL for LCRRP is computed: $OSL = 0.0272$. We adjust our SARIMA estimation with OSL and the fitted direction in the third step. The error comparison between SARIMA M6 and the hybrid model can be found in Table 1.4. We can see a significant improvement in the estimation error before and after the adjustment. Fig. 1.14 and Fig. 1.15 demonstrate the ACF and PACF of the residuals after adjustment, where a huge mitigation of seasonality in the residuals can be found. In other words, the residual series of our hybrid model is smaller, and is closer to a white noise series than that of an SARIMA.

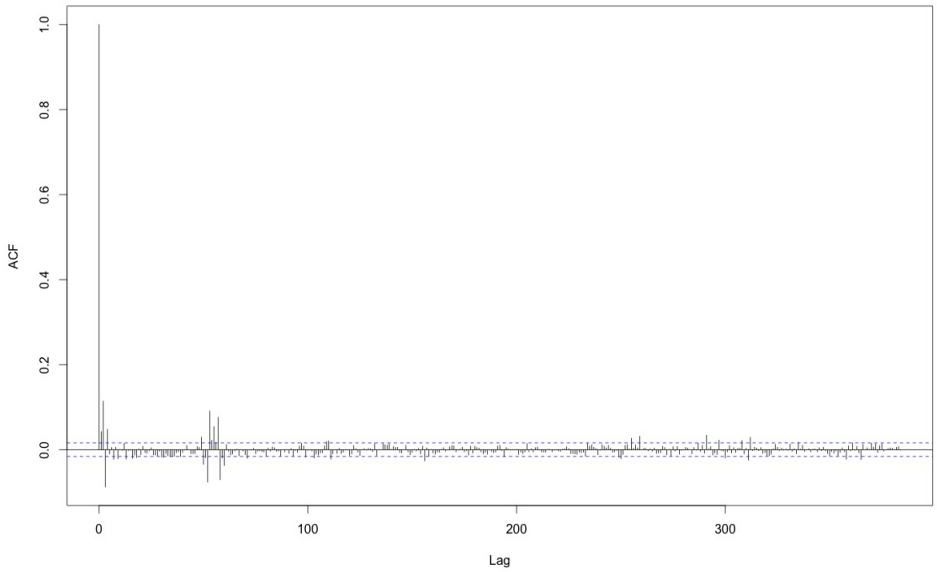

FIGURE 1.14: ACF of Hybrid's Residuals

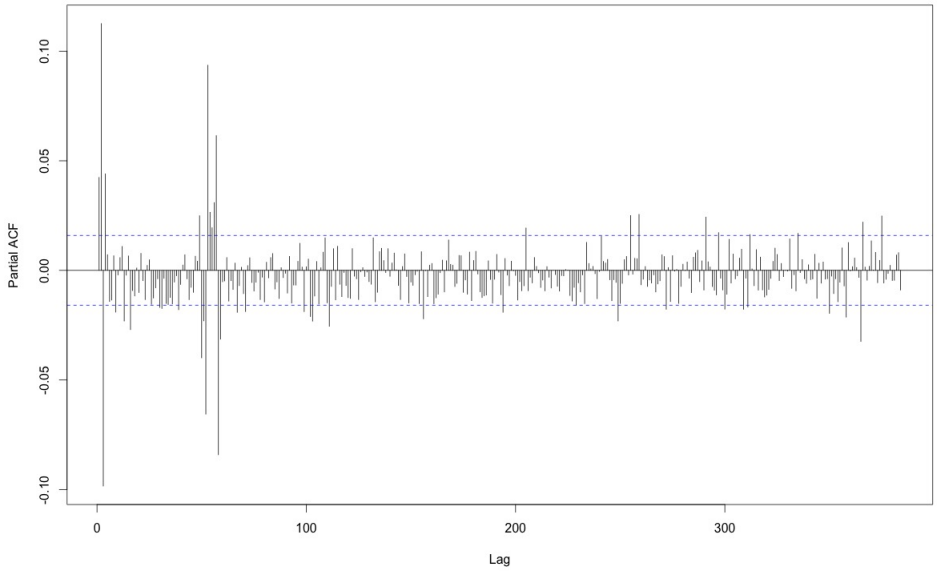

FIGURE 1.15: PACF of Hybrid's Residuals

Table 1.4: Error Comparison Between SARIMA Model and Hybrid Model

Model	RMSE	MAE
SARIMA M6	0.1910	0.0695
Hybrid	0.1607	0.0610

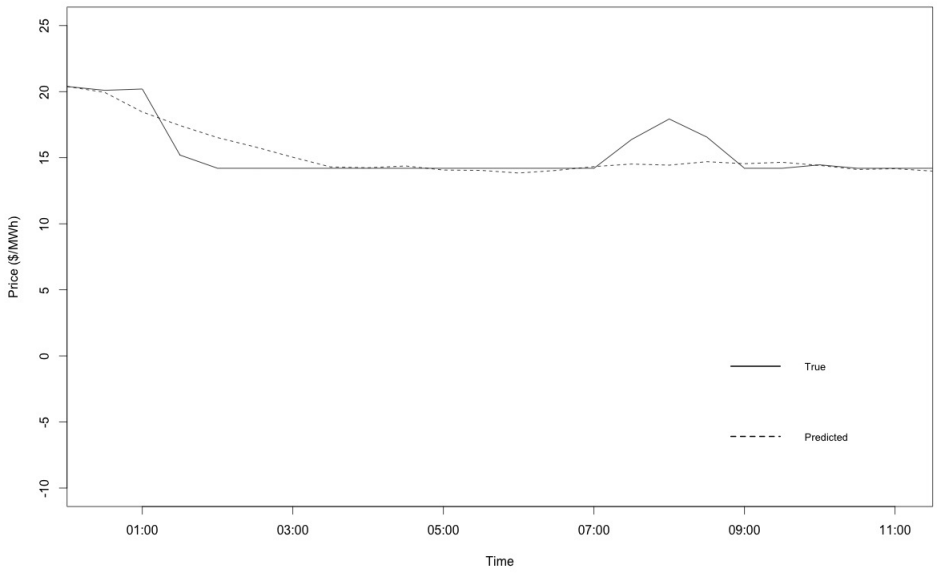

FIGURE 1.16: Comparison Between Predicted and True Settlement Price Series

1.4.3 Predicting the Settlement Price in the Next 12 Hours

Utilizing the predicted values of total demand with the neural network discussed in Section 1.3, we are able to calculate the predicted values of settlement price with an SARIMA model, the direction of the step shift with the neural network and ultimately the adjusted estimation of settlement price, through an iterative procedure. Fig. 1.16[5] is the prediction and true value of the settlement price from January 1, 2015 00:00 am to January 1, 2015 11:30 am.

To calculate the confidence interval for the predicted values, we can estimate the

[5] True data can be found in http://www.aemo.com.au/Electricity/Data/Price-and-Demand/Aggregated-Price-and-Demand-Data-Files/Aggregated-Price-and-Demand-2011-to-2016

distribution of the prediction residuals parametrically or non-parametrically with n-fold cross-validation.

1.5 Discussion about Model Improvement

Our hybrid model exhibits superior strength to predict the high frequency electricity price series. However, the model does not perform well when the price and demand fluctuate dramatically. In fact, there is a couple of methods to improve the goodness of fit:

1. Obtain more variables. As we've discussed above, weather related variables and power outage can be important factors that drive the total demand and price to fluctuate. Weather variables include continuous variables such as temperature, precipitation, wind-speed, humidity, and discrete variables such as bushfire, flood or storm. On the supply side, power import and export should be considered. Models with these variables are expected to capture the extremely volatile pattern we see in the price and demand plot.

2. Adopt a Probabilistic Neural Network (PNN) for direction classification. PNN is more powerful in the area of classification because it can approximate the true probability density distribution for each category with flexible kernel functions, and extenuate the effect of outliers by incorporating Bayesian method. For example, Mehdi Khashei and Ardali (2012) uses a PNN in the hybrid model. PNN is not implemented in this project due to the computational resource needed for the huge data set we have.

3. Use the Artificial Neural Network with a logistic activation function to fit the residuals unexplained by SARIMA. We can predict the residuals in the SARIMA models and add it back to the SARIMA fitted values directly, instead

of using ANN to classify the direction of residuals (which we treat as step shifts).

4. Calculate the residuals of the hybrid model and repeat Step 2 - Step 5 until a certain condition is met. That is, we fit the residuals unexplained by the hybrid model with a neural network, calculate a 2nd-order OSL, adjust the original fitted values by the hybrid model with the 2nd-order OSL, and so forth. Convergence of the residual sum of squares may cause the over-fitting problem. Therefore we can set a threshold that stops the iteration through cross-validation.

2

Fast False Selection Rate Variable Selection

In this project, we will implement the Fast False Selection Rate (FSR) Variable Selection algorithm in Python. The algorithm is developed by Dennis D. Boos, Leonard A. Stefanski and Yujun Wu in their article "Fast FSR Variable Selection with Applications to Clinical Trials" (Dennis D. Boos and Wu (2009)).

2.1 Algorithm Introduction

Many variable selection procedures have been developed in the literature for linear regression models. This paper proposes an updated version of FSR method to control variable selection without simulation. By adding a number of phony variables to the real set of data and monitoring the proportion of the phony variables falsely selected as a function of the tuning parameter, like α-to-enter of forward selection, the FSR method developed in Yujun Wu and Stefanski (2007) is able to estimate the appropriate tuning parameter and control the model false selection rate, selecting informative variables and preventing uninformative ones from being selected. Fast FSR introduced in Dennis D. Boos and Wu (2009) allows researchers to estimate the tuning parameter from the summary table of the forward selection variable sequence.

In other words, Fast FSR improve FSR algorithm by skipping the step of generating phony variables.

2.1.1 The False Selection Rate Method

The false selection rate is defined as

$$\gamma = E\left\{\frac{U(Y,X)}{1+S(Y,X)}\right\}, \tag{2.1}$$

where $1 + S(Y, X)$ and $U(Y, X)$ are the total number of variables (including the intercept) and number of uninformative variables selected in the model. In a model selection method that depends on a tuning parameter α, $U(Y, X)$ and $S(Y, X)$ are the functions of α. In the α-to-enter forward selection method, on which the Fast FSR method is built in Dennis D. Boos and Wu (2009), α is the significance level for entry. As $U(Y, X)$, or equivalently $U(\alpha)$, is not known, we estimate $U(\alpha)$ as:

$$\hat{U}(\alpha) = \hat{\theta}(\alpha)[k_T - S(\alpha)], \tag{2.2}$$

where $\hat{\theta}(\alpha)$ is an estimate of the rate at which uninformative variables enter the selected model at the significance level of α, and k_T is the total number of variables, including both real and phony variables. The method randomly attaches k_p phony variables to the training data set, fits the model, calculates the number of falsely selected variables $U(\alpha)$ that is associated with the model selection method with a tuning parameter α and repeats the process for a large number of times, and $\hat{\theta}(\alpha)$ is the average of $\frac{U(\alpha)}{k_p}$ in each sampling iteration. In short, the FSR method selects the tuning parameter α that the following condition is satisfied:

$$\begin{aligned} \hat{\alpha} &= sup_{\alpha \leqslant \alpha_{max}}\{\alpha : \hat{\gamma}(\alpha) \leqslant \gamma_0\} \\ \hat{\gamma}(\alpha) &= \frac{\hat{\theta}(\alpha)[k_T - S(\alpha)]}{1+S(\alpha)}, \end{aligned} \tag{2.3}$$

where γ_0 is the target FSR we want to control. Typically we set $\gamma_0 = 0.05$.

Theoretically, $k_T - S(\alpha)$ is neither the number of uninformative variables nor the number of informative variables in the data set. In fact, this quantity tends to overestimate the true number of uninformative variables in the data set when α is small, and underestimate it when α is big. However, evidence in Dennis D. Boos and Wu (2009) shows that it is a proper estimate in relatively sparse models, which occurs more often in certain fields of study, such as clinical research.

2.1.2 The Fast False Selection Rate Method

The Fast FSR method is an application of the FSR method in Yujun Wu and Stefanski (2007) to forward selection. The forward selection method selects a variable with the smallest p-value out of the variables not in the model in each step. However, the full sequence of k_T p-values is not necessarily monotonically increasing. The fast FSR method carries the largest p-value forward if the p-value in the next step is smaller, and obtains a monotonic p-value sequence $\tilde{p}_1 \leqslant \tilde{p}_2 \leqslant \cdots \leqslant \tilde{p}_k$.

The major improvement of the Fast FSR algorithm is that it skips the step of sampling phony variables by replacing $\hat{\theta}(\alpha)$ with α. The justification of this approach is that in the Gaussian linear model with a known error σ^2 and uncorrelated variables, the p-value of an uninformative variable is uniformly distributed over $(0, 1)$ and the p-values of all the variables should be independent with each other. As a consequence, the number of uninformative variables selected through forward selection with the tuning parameter α should follow the binomial distribution $B(k_U, \alpha)$, where k_U is the number of uninformative variables in the data set, and the expected number $E\{U(\alpha)\} = k_U\alpha$. While σ^2 is estimated in each step of forward selection, which slightly alters the independence and uniformity, and in general the uncorrelation condition of variables is rarely satisfied, the simulation studies in Dennis D. Boos and Wu (2009) show that α in the Fast FSR method and $\hat{\theta}(\alpha)$ in the regular FSR method

are very close, even in the cases where variables are highly correlated ($\rho = 0.7$).

The modification in the fast FSR leads to the rule of model selection as follows:

$$\begin{aligned} \hat{\alpha} &= sup_{\alpha \leqslant \alpha_{max}} \{\alpha : \hat{\gamma}(\alpha) \leqslant \gamma_0\} \\ \hat{\gamma}(\alpha) &= \frac{\alpha[k_T - S(\alpha)]}{1 + S(\alpha)}, \end{aligned} \tag{2.4}$$

and therefore the model size, $k(\gamma_0)$, is:

$$k(\gamma_0) = \max \left\{ i : \tilde{p}_i \leqslant \frac{\gamma_0[1 + S(\tilde{p}_i)]}{k_T - S(\tilde{p}_i)} \text{ and } \tilde{p}_i \leqslant \alpha_{max} \right\} \tag{2.5}$$

2.2 Pseudocode

1. Step 1: Use forward selection to generate the sequence of variables and the associated p-values.

$$p_1, p_2, \cdots, p_k \tag{2.6}$$

2. Step 2: Monotonize the p-value of the original sequence by carrying the larger p-value forward until the next p-value is even larger.

$$\tilde{p}_1 \leqslant \tilde{p}_2 \leqslant \cdots \leqslant \tilde{p}_k \tag{2.7}$$

3. Step 3: For the ith step in the forward selection sequence, calculate

$$\hat{\alpha}_i = \frac{\gamma_0(1 + S_i)}{k_T - S_i}, \tag{2.8}$$

4. Step 4: Compare $\tilde{p}_i$ and $\hat{\alpha}_i$. Select the model of size j, where $j = max\{i : \tilde{p}_i \leqslant \hat{\alpha}_i \text{ and } \tilde{p}_i \leqslant \alpha_{max}\}$. Also, return the corresponding $\hat{\alpha}_i$.

Dep. Variable:	y	**R-squared:**	0.811
Model:	OLS	**Adj. R-squared:**	0.800
Method:	Least Squares	**F-statistic:**	75.50
Date:	Sat, 19 Mar 2016	**Prob (F-statistic):**	2.49e-30
Time:	23:11:09	**Log-Likelihood:**	-315.88
No. Observations:	94	**AIC:**	643.8
Df Residuals:	88	**BIC:**	659.0
Df Model:	5		

FIGURE 2.1: Fast FSR Fitting Results 1

2.3 Unit Test

Unit Tests are conducted with data in William D. Mangold and Adams (2003): "The Impact of Intercollegiate Athletics on Graduation Rates Among Major NCAA Division I Universities."[1]. Fig. 2.1 and Fig. 2.2 are the model fitting results. They show that our algorithm obtains the same result as it is in Dennis D. Boos and Wu (2009)[2], and therefore it passes the unit test.

2.4 Comparison between Fast FSR and LASSO

In this section, we compare the performance of LASSO and Fast FSR in term of Model Error, False Selection Rate with the simulated data and running time with the real data. Model Error is defined as Mean Squared Error:

$$\frac{1}{n} \times \sum_{i=1}^{n} (\hat{f}(x_i) - y_i)^2, \tag{2.9}$$

[1] The database is saved in http://www4.stat.ncsu.edu/ boos/var.select/ncaa.data2.txt

[2] Their result is save in http://www4.stat.ncsu.edu/~boos/var.select/fsr.fast.ncaa.ex.txt

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
const	-42.1069	8.990	-4.684	0.000	-59.972 -24.242
x2	3.4714	0.467	7.428	0.000	2.543 4.400
x3	0.2391	0.076	3.163	0.002	0.089 0.389
x5	0.2787	0.078	3.582	0.001	0.124 0.433
x4	0.6770	0.195	3.475	0.001	0.290 1.064
x7	-2.5913	0.832	-3.115	0.002	-4.245 -0.938

Omnibus:	5.624	Durbin-Watson:	1.718
Prob(Omnibus):	0.060	Jarque-Bera (JB):	3.905
Skew:	0.351	Prob(JB):	0.142
Kurtosis:	2.290	Cond. No.	620.

FIGURE 2.2: Fast FSR Fitting Results 2

where $\hat{f}(x_i)$ is the fitted value and y_i is the dependent variable.

We create four data sets for testing. In each data set, we simulate 1500 data points with 21 variables. Next we take the square of the 21 variables and concatenate the squared data with the original one. Therefore there are 42 independent variables x_{ij} in each data set ($i = 1, 2, \cdots, 1500; j = 1, 2, \cdots, 42$). Finally, we generate the dependent variables y_i by linearly combining a subset of the original 21 variables with an array of weights that are determined arbitrarily. As a result, only the selected subset of variables are the informative factors in each test, and the model coefficients are expected close to the weights, should the model identify the informative factors. Four sets of influential factors and their weights are summarized as follows:

1. H1: y_i are generated independently from any variables x_{ij}

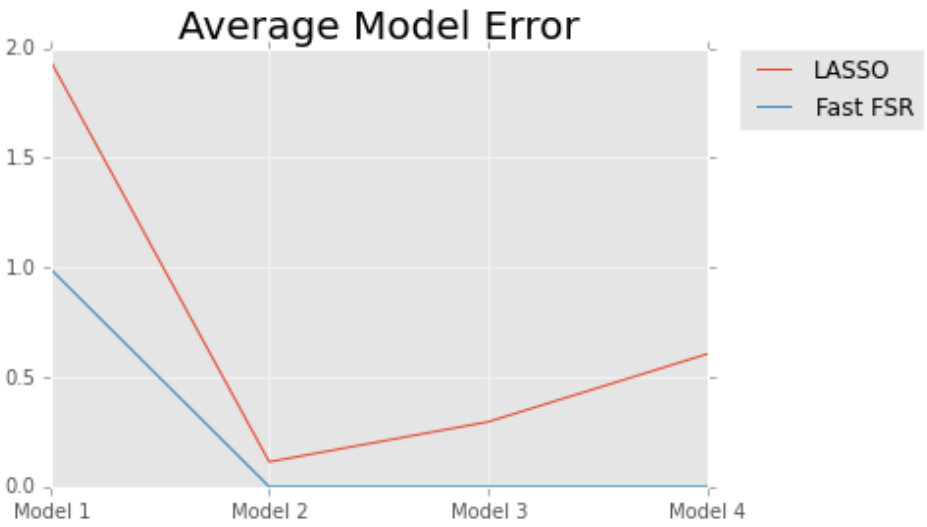

FIGURE 2.3: Average MSE of LASSO and Fast FSR

2. H2: $y_i = [x_{i6}, x_{i7}, x_{i8}, x_{i13}, x_{i14}, x_{i15}] \times [9, 4, 1, 9, 4, 1]^T$

3. H3: $y_i = [x_{i5}, x_{i6}, x_{i7}, x_{i8}, x_{i9}, x_{i12}, x_{i13}, x_{i14}, x_{i15}, x_{i16}] \times [25, 16, 9, 4, 1, 25, 16, 9, 4, 1]^T$

4. H4: $y_i = [x_{i4}, x_{i5}, x_{i6}, x_{i7}, x_{i8}, x_{i9}, x_{i10}, x_{i11}, x_{i12}, x_{i13}, x_{i14}, x_{i15}, x_{i16}, x_{i17}] \times$
   $[49, 36, 25, 16, 9, 4, 1, 49, 36, 25, 16, 9, 4, 1]^T$

The whole process is repeated for 50 times. Average MSE and average FSR are calculated and compared between the Fast FSR method and the built-in LASSO function. To accelerate the speed, we apply parallel programming that splits the job into two pieces.

Fig. 2.3 and Fig. 2.4 show that the Fast FSR method not only has a better goodness of fit than LASSO, but also a more powerful capability to identify informative factors from uninformative ones, which is crucial if the research target is interpretation. Moreover, we can notice a consistent outperformance of the Fast FSR in both sparse and less sparse models. Therefore, we conclude that Fast FSR has more merit than standard LASSO in model fitting.

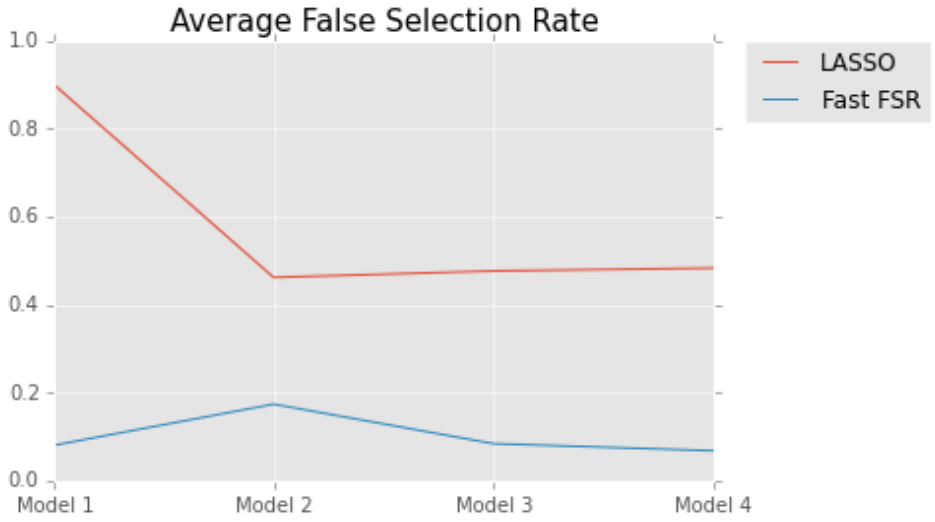

FIGURE 2.4: Average False Selection Rate of LASSO and Fast FSR

```
%timeit -r1 -n1 fsr_fast(data_x,data_y,.05)
%timeit -r1 -n1 lasso = Lasso(0.05, tol=0.001).fit(data_x, data_y).coef_
```

```
1 loops, best of 1: 13.5 ms per loop
1 loops, best of 1: 749 µs per loop
```

FIGURE 2.5: Running Time of LASSO and Fast FSR

2.5 Profiling

Not surprisingly, Fig. 2.5 suggests that our program for the Fast FSR is slower than the built-in LASSO function. Fig. 2.6 shows the duration of function calls in our program of the Fast FSR method.

The top 2 function calls that cost the most time are related to the package `rpy2`. To implement the forward selection at the beginning of Fast FSR, we import the R packages `leaps` and `stats` from `rpy2`. Unfortunately, however, there is no built-in functions for forward selection in Python's statistical models. Therefore, there is no method to make more improvement for the first two calls.

Similarly, the third call in the profiling table is to create a function in Python. This call is inevitable as long as we apply functional programming. Therefore, no

```
         4700 function calls (4616 primitive calls) in 0.023 seconds

   Ordered by: internal time
   List reduced from 358 to 15 due to restriction <15>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
       21    0.003    0.000    0.003    0.000 {method 'rcall' of 'rpy2.rinterface.SexpClosure' objects}
       40    0.002    0.000    0.002    0.000 numpy2ri.py:42(numpy2ri)
        1    0.002    0.002    0.011    0.011 functions.py:95(__call__)
        1    0.001    0.001    0.001    0.001 _abcoll.py:545(update)
       19    0.001    0.000    0.005    0.000 _distn_infrastructure.py:1610(cdf)
  398/395    0.001    0.000    0.001    0.000 {numpy.core.multiarray.array}
        1    0.001    0.001    0.023    0.023 <ipython-input-6-630cca1fee41>:1(fsr_fast)
       19    0.001    0.000    0.002    0.000 _distn_infrastructure.py:547(argsreduce)
       58    0.000    0.000    0.000    0.000 {method 'take' of 'numpy.ndarray' objects}
       48    0.000    0.000    0.000    0.000 {method 'reduce' of 'numpy.ufunc' objects}
       20    0.000    0.000    0.004    0.000 pandas2ri.py:49(py2ri_pandasseries)
        1    0.000    0.000    0.002    0.002 indexing.py:1386(_getitem_axis)
       19    0.000    0.000    0.000    0.000 _continuous_distns.py:1180(_cdf)
        2    0.000    0.000    0.001    0.000 decomp_svd.py:15(svd)
       24    0.000    0.000    0.000    0.000 vectors.py:230(__init__)
```

FIGURE 2.6: Profiling of Fast FSR

improvement for this algorithm is made at the moment.

3

Recreation of the Police Precinct Boundaries on the Manhattan Island

The target of this project is to reconstruct the boundaries of the 22 police precincts on the Manhattan Island. New York City have made an enormous amount of data public on the NYC Open Data website[1], content covering business, government, education, transportation, etc. And we focus on the parking violation data, because the size of database is big, and each data entry has information we need. It's worth noting that a considerable number of violations are labeled with a wrong police precinct number due to human error or omissions. Therefore we utilize a support vector machine to estimate which precinct each violation belongs to most likely, and therefore relabel the outliers with estimated police precinct number.

Databases we use include:

1. `NYParkingViolations`: that contains all the parking violation data between August 2013 and June 2014

2. `pluto`: that contains shapefiles which merges tax lot data with tax lot features

[1] https://nycopendata.socrata.com/

from the Department of Finance's Digital Tax Map. We can find the address and its geographic coordinates at the tax lot level.

We also independently build a library on the alternative names for address terms in NYC, such as `S.` vs `South`, that helps us to standardize the address.

3.1 Cleaning and Geocoding

From the parking violation database, we extract the variables of our interest and change the format of the date. Next, we remove the invalid violation records whose precinct numbers are beyond the scope. We extract and capitalize the address, and replace the address with the alternative terms in the library we build, that facilitate the process to match these information in the database of `pluto`.

A brief browse in the joint database of `NYParkingViolations` and `pluto` tells us that there are a lot of duplicated addresses and coordinates. However, almost all the addresses where multiple violations take place are labeled with more than one precinct number in different violations, which will cause fatal results in the analysis later. To figure out the true precinct of each address, we apply the pivot table. The underlying assumption is that, the most records of an address should have the true precincts. As a result, the precinct mean of the duplicated records for each address should be close to the true one. The procedure is as follows. First, we calculate the mean of the precinct numbers for each address. Second, we calculate the difference between the precinct numbers and its mean of the same address for each record. Thirds, we set the threshold = 0.5. If the difference is larger than the threshold, we say that the precinct of the address in that record is wrong, and thus we remove this record. Later, we remove the duplicated records and address, since they are no longer necessary in the following analysis. Finally we remove 1% of the total violations in each precinct whose distance from the center of that precinct is the largest, viewing

them as potential outliers.

Fig. 3.1 demonstrates all the violations after geocoding and cleaning. We can find a clear area in the Central Park where few violations happen. However, the Central Park is covered by the 22nd Precinct. Therefore, we add a bunch of phony violation data points in the Central Park so as to generate the boundary of the 22nd Precinct.

3.2 Support Vector Machine and Visualization

We train a support vector machine for multiclass classification with the cleaned data. Also, by using the R package `raster`, we create a grid over the Manhattan Island and then estimate which precinct each cell most likely belongs to. Combining the cells by their estimated police precinct numbers and converting them into spatial polygon data frame, we generate a polygon database for each precinct. We also plot the precinct polygons as in Fig. 3.2. Our last job is creating a Json file from the spatial polygon data frame.

FIGURE 3.1: Violations on the Manhattan Island

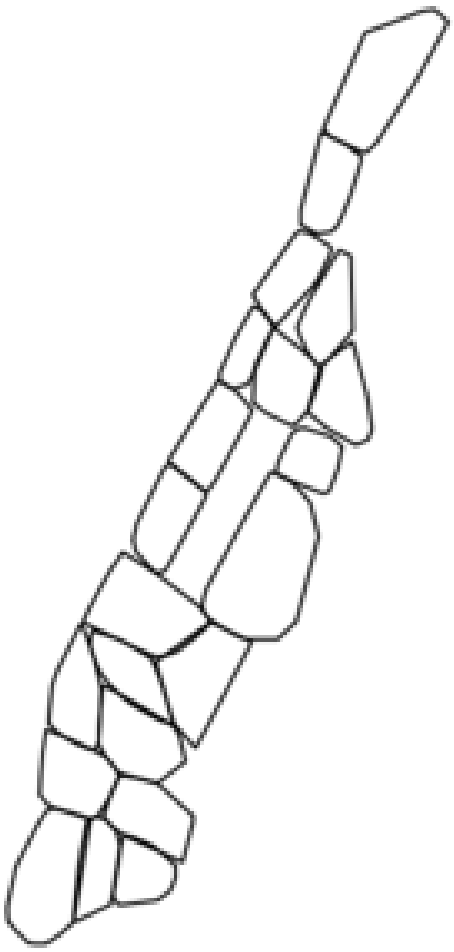

FIGURE 3.2: Police Precinct Boundaries on the Manhattan Island

Bibliography

AEMO (January 31, 2014), "High Demand Event Report - 28 Jan 2014 (Victoria and South Australia)," *AEMO*.

Caldwell, A. (January 17, 2014), "Melbourne Records Heatwave Not Seen for 100 Years," .

Dennis D. Boos, L. A. S. and Wu, Y. (2009), "Fast FSR Variable Selection with Applications to Clinical Trials," *Biometrics*, 65, 692–700.

J.P.S. Catalão, S.J.P.S. Mariano, V. M. and Ferreira, L. (2007), "Short-term Electricity Prices Forecasting in a Competitive Market: A Neural Network Approach," *Electric Power Systems Research*, 77, 1297–1304.

Mehdi Khashei, M. B. and Ardali, G. A. R. (2012), "Hybridization of Autoregressive Integrated Moving Average (ARIMA) with Probabilistic Neural Networks (PNNs)," *Computers & Industrial Engineering*, 63, 37–45.

Spurio, J. (2014), "Five Minute Electricity Demand Forecasting: Neural Network Model Documentation V2," *Australian Energy Market Operator Forecasting Document*.

William D. Mangold, L. B. and Adams, D. (2003), "The Impact of Intercollegiate Athletics on Graduation Rates among Major NCAA Division I Universities: Implications for College Persistence Theory and Practice," *The Journal of Higher Education*, 74, 540–562.

Yujun Wu, D. D. B. and Stefanski, L. A. (2007), "Controlling Variable Selection by the Addition of Pseudovariables," *Journal of the American Statistical Association*, 102, 235–243.

Zhang, G. P. (2003), "Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model," *Neurocomputing*, 50, 159–175.