NC STATE UNIVERSITY

ECE 592 Project

Day-Ahead Locational Marginal Price Forecasting

Instructors: Dr. Wenyuan Tang — Dr.Arnie de Castro

Team Members:

Dwaraknaath Varadharajan

Shashank Govindarajan

Vishal Tyagi

Executive Summary

The goal of this project is to develop and test time series models to forecast the day ahead price of electricity for the Long Island zone in the NYISO market. The software tool used for forecasting is SAS 9.4.

The introduction section of this report briefly introduces the topic of price forecasting and its significance in the deregulated energy market. The problem statement and the key objectives of the project are then identified. The ARIMA, ARIMAX, LSTM and Linear Regression forecasting techniques that were implemented in this project are discussed in detail. The results and discussion section of this report provides a summary of the forecasting results obtained using the different forecasting techniques and compares their forecasting accuracy in order to present a final recommendation. Final thoughts and observations on the project are presented in the conclusion section of the report. The programming codes written using SAS 9.4 for this project are included as part of the Appendix section of this report. This report uses multiple references. We thank all authors and acknowledge their contribution in this report.

Contents

1 Introduction

Over the past decade, electricity price forecasts have become a critical input to the decision-making mechanisms of energy companies at the corporate level. Electricity is a very unique commodity as it cannot be stored in large quantities and requires a constant balance between production and consumption. Energy demand depends on multiple factors like weather (temperature, wind speed, precipitation, etc.), the intensity of business activities and consumer habits during peak and off-peak hours. These unique characteristics lead to price dynamics not observed in any other market. As a result of this unique behavior, electricity markets have become a central point of research in the energy sector and accurate electricity price forecasting has emerged as one of the biggest challenges faced by the different market entities. In energy markets, price forecasting is required by consumers and producers to maximize profits and minimize wastage of utilities. Day-ahead price forecasting is also very important for virtual bidding.

The volatile and uncertain nature of electricity markets and the fact that electricity is an essential commodity for consumers are two factors that further emphasize the importance of price forecasting. Consequently, different electricity forecasting tools and models have been proposed by researchers across the globe with varying degrees of accuracy. Our challenge in this project will be to identify the model that can capture the volatility of DA prices at long island with a better accuracy.

2 Description

The goal of this project is to accurately forecasting day ahead electricity prices for the Long Island zone of the NYISO market. The main objectives of the project are to:

- Review technical papers to find out and evaluate different price forecasting techniques and factors that greatly affect these forecasts.
- Develop price forecast models using NYISO data to predict the next day electricity price of the long island zone of NYISO
- Forecast day ahead prices using these developed models and evaluate their performance by comparing the results with actual values.
- Briefly summarize and present a recommendation based on the results observed.

To further improve our performance, we will deploy different state-of-the-art versions of a fully connected Long-Short Term Memory (LSTM) model of Recurrent Neural Network (RNN) class, train on the same dataset and compare their performance metrics with time-complexity in training the model and generating predictions

3 Techniques and Methods

3.1 Data Preparation

The hourly energy market historical data we require from NYISO is publicly available from their website. Price and load are said to have a good correlation, so we have downloaded hourly load forecast information as well. We will train the Linear regression model using these time dependent features and the results will be compared using ARIMA model and LSTM. Hyperparameter tuning for LSTM is performed using Google Colab which offers high computing power and GPU.

3.2 Multiple Linear Regression

Linear regression analysis is the most widely used statistical technique and is the study of linear, additive relationships between variables. In this type of analysis, there is the dependent variable that shall be forecasted and the independent variables that are used as inputs for the forecast. Linear regression analysis is widely used because of its simplicity and the fact that relationships between variables for most models are approximately linear over the range of values that are of interest. Even when that is not the case, the variables can often be transformed in such ways that linearize their relationships.

In regression, if the dependent variable is affected by only one independent variable, then the time series is a simple linear regression. If there are multiple independent variables involved, then the regression model is said to be multiple regression model. The general equation for a multiple linear regression model is shown below. For forecasting day ahead prices, multiple linear regression is used. The general equation for multiple linear regression is:

$$y_t = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_k x_k + e \tag{1}$$

Where,

y_t is the dependent variable or response

x_1 to x_k are the regressor or predictor variables

α_0 to α_k are the regression coefficients

The forecast for the value of y_t is, therefore, a straight-line function of all the x variables and the contribution of all x variables are additive. The intercept value of α_0 is the value that the model would make if all the independent variables were 0. The coefficients and the intercept value are calculated by least squares.

This approach models the relationship between two or more explanatory variables and a dependent variable by fitting a linear equation to the observed data. To estimate the parameters or regression coefficients the method of least square can be used. Once the coefficients are estimated, the new value of the dependent variable can be easily found. The main factors that determine the pattern of electricity market price may vary from one market to another. Apparently, it is quite difficult to have a universal pattern. However, all electricity markets may share one thing in common; the highly volatile nature of electricity prices. Due to the nature of the power market, the degree of volatility is worse on electricity markets than other markets and it is also more uncertain.

In SAS, the procedure PROC REG is used to find the linear regression model between two variables.

3.2.1 Model and Features

After reviewing the different data and price patterns of the NYISO market, we found out that the electricity market price for a given hour "K" of the following day (tomorrow's price) is highly correlated to the electricity market price of:

- Previous day (today's) K^{th} hour price
- K^{th} hour price of previous week of the same day

- $(K-1)^{th}$ hour of the same day

The correlation strength, however, may depend on the period of the year at which it is considered. The price at hour K is more closely correlated with the price of hour K-1 of the same day. However, as this data is not available prior to making the forecast, therefore, it is not considered for the 24-hour-ahead price forecast model. Also, it is not possible to make predictions for more than 24 hours into the future, as the previous day K^{th} hour price will not be available prior to prediction. It is also observed that electricity demand, and hence price, shows dependency on hour of the day, day of the week and season of the year. Therefore, it is assumed that selecting a specific hour of a day means somehow approximating the level of the demand at that hour. Following features were added in addition to the price features to increase the prediction accuracy.

- Previous day (today's) K^{th} hour load
- K^{th} hour load of previous week of the same day
- K^{th} hour load of the same day
- Calendar data
  - Hour of the day, Day of the week, and Month of the year

After a careful study of the nature of the electricity market, the best approach that is believed to give a better forecast result was chosen and the forecasting model is developed. The simple linear multiple regression model that is developed using a series of price data from previous weeks and previous days, series of load data from previous weeks and previous days and calendar data to predict the next day market price has the following form.

$$P_k = \alpha_0 + \alpha_1 P_{pd} + \alpha_2 P_{pw} + \alpha_3 L_{pd} + \alpha_4 L_{pw} + \alpha_5 L + \alpha_6 C_h + \alpha_7 C_{wd} + \alpha_8 C_m \qquad (2)$$

Where,

P_k: K^{th} hour price of following day (price of tomorrow)

P_{pd} : Previous day (today's) K^{th} hour price

P_{pw} : K^{th} hour price of the same day of the previous week

L_{pd}: Previous day (today's) K^{th} hour load

L_{pw}: K^{th} hour load of the same day of the previous week

L: K^{th} hour load of the same day

C_h: Hour of the day [0,23]

C_{wd}: Day of the week [1,7]

C_m: Month of the year [1,12]

α_0 to α_8: Regression coefficients

3.2.2 Model Training and Testing

Finally, using the multiple linear regression model of the form given in above equation and calculating the coefficients by the method of least square (MLS), price forecasts were made. To evaluate the forecast performance, the popular Mean Absolute Percentage Error (MAPE) has been used. The MAPE is defined as the sum of the absolute value of the difference between actual and forecast values divided by the actual value. Mathematically it is represented as

$$\text{MAPE} = \frac{1}{T}\sum_{n=1}^{T} \frac{|ForecastedPrice - GroundTruth|}{GroundTruth} \times 100$$

Where T is forecast period

In order to train and test the model price and load data of NYISO's Long Island zone for the years 2016 through 2019 has been used. To analyze the accuracy of the model, 24-hour day-ahead price forecasts have been produced for 4 different days of the year. The 4 days are selected from 4 different seasons of the year. The model is trained using the data from January 1, 2016 till the day prior to the day of prediction, and then the 24-hour predictions for the selected day are done using the trained model.

Table 1: Test cases (season-wise)

Test Cases		
Season	**Prediction Day**	**Training Duration**
Winter (Dec to Feb)	Jan 22, 2019	Jan 01, 2016 to Jan 21, 2019
Spring (Mar to May)	May 20, 2019	Jan 01, 2016 to May 19, 2019
Summer (Jun to Aug)	Jun 02, 2019	Jan 01, 2016 to Jun 01, 2019
Fall (Sep to Nov)	Nov 11, 2019	Jan 01, 2016 to Nov 10, 2019

Table 2: MAPE for different seasons

Test Results		
Season	**Prediction**	**MAPE**
Winter (Dec to Feb)	Jan 22, 2019	8.30 %
Spring (Mar to May)	May 20, 2019	9.82 %
Summer (Jun to Aug)	Jun 02, 2019	11.27 %
Fall (Sep to Nov)	Nov 11, 2019	10.83 %

3.2.3 Plot of Ground Truth and forecasted values

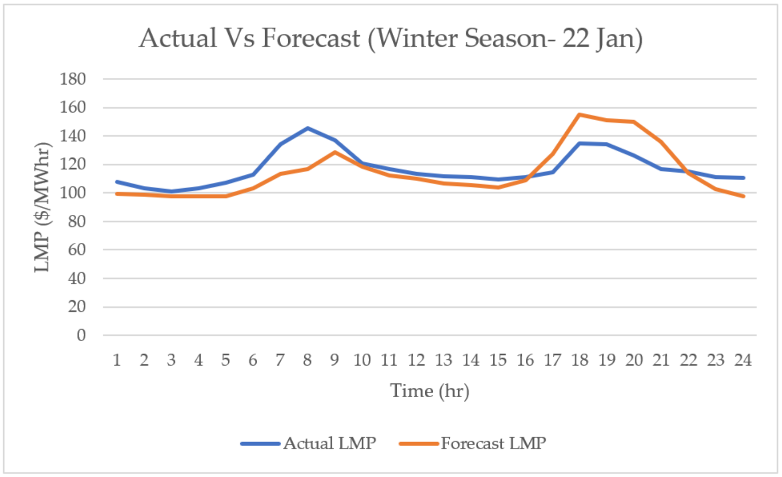


Figure 1: Actual Vs Predicted Price- Winter

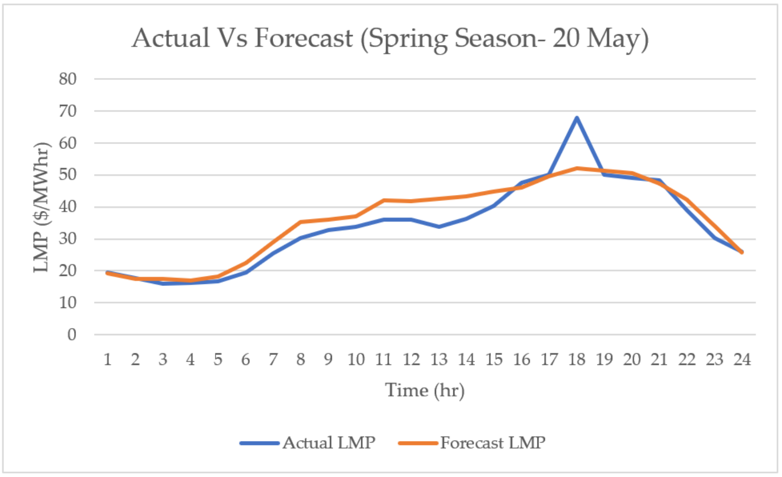


Figure 2: Actual Vs Predicted Price- Spring

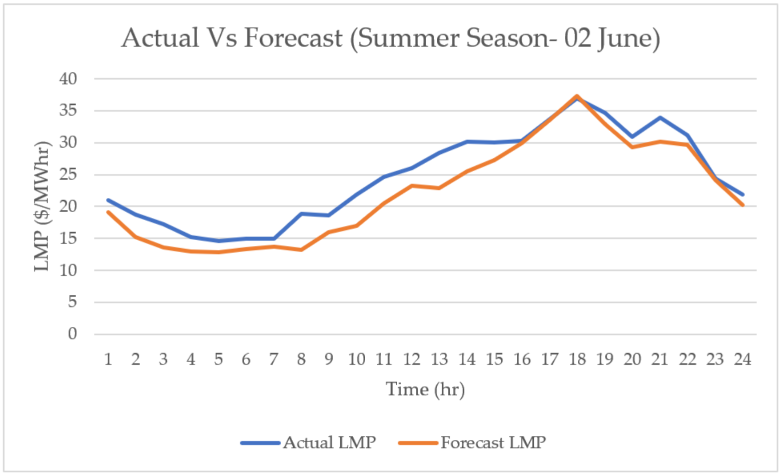


Figure 3: Actual Vs Predicted Price- Summer

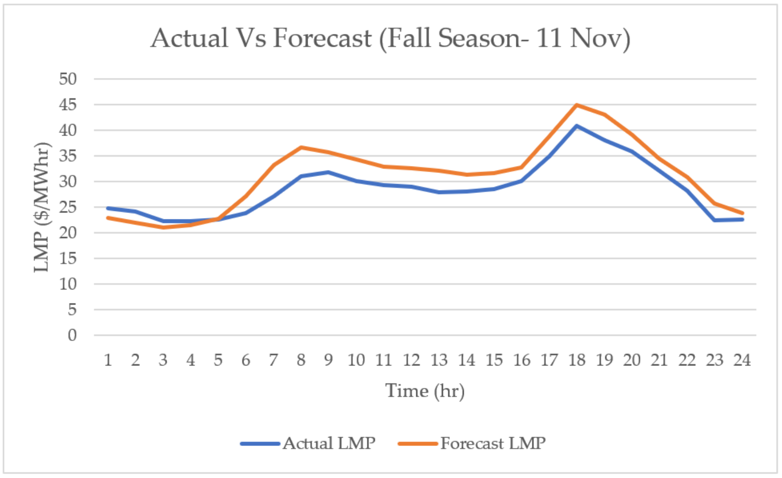


Figure 4: Actual Vs Predicted Price- Fall

It is observed that the MAPE varies for different days of different seasons, showing that the correlation between the LMP and used features depends on the period of the year. There can be multiple reasons for difference in the predicted and actual LMP values, ranging from accuracy of the load forecast to the behavior of the dominant player in the market.

3.3 Auto Regressive Integrated Moving Average (ARIMA)

An ARIMA model predicts a value in a response time series as a linear combination of its own past values, past errors (also called shocks or innovations), and current and past values of other time series. The ARIMA approach was first popularized by Box and Jenkins, and ARIMA models are often referred to as Box-Jenkins models. When an ARIMA model includes other time series as input variables, the model is sometimes referred to as an ARIMAX model or dynamic regression model. There are different variations of ARIMA models. Few variations are listed below:

- Seasonal
- Subset
- Factored
- Interrupted Time Series Models
- Multiple regression analysis with ARMA errors
- Rational transfer function models of any complexity.

ARIMA Modeling involves three stages:

1. Identification Stage
2. Estimation and Diagnostic Stage
3. Forecasting Stage

3.3.1 Model Identification and Time series analysis

In this stage, we analyze the time series data. We identify the suitable model parameters by plotting auto-correlation plot or correlogram, partial auto-correlation (PAC) and inverse auto-correlation and cross correlation. Cross correlation helps in the case of arimax where we use another time series data as input.

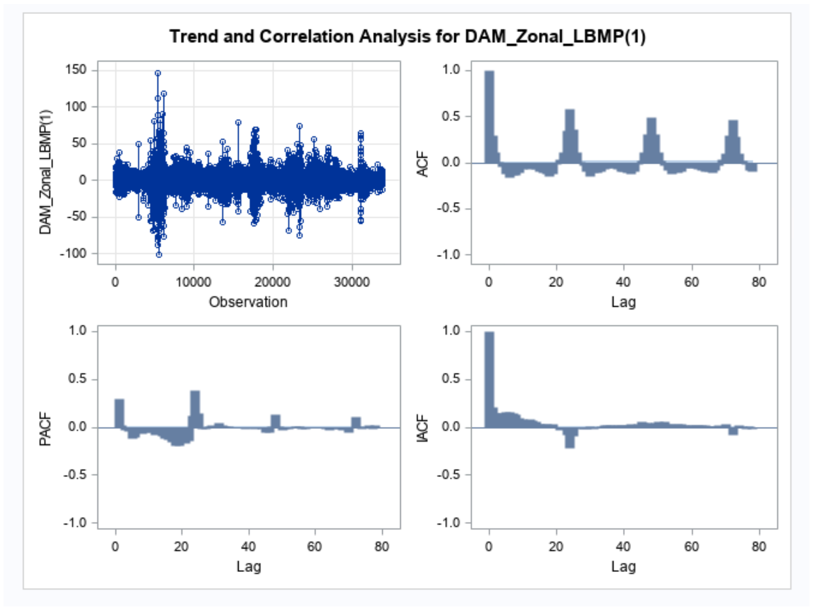


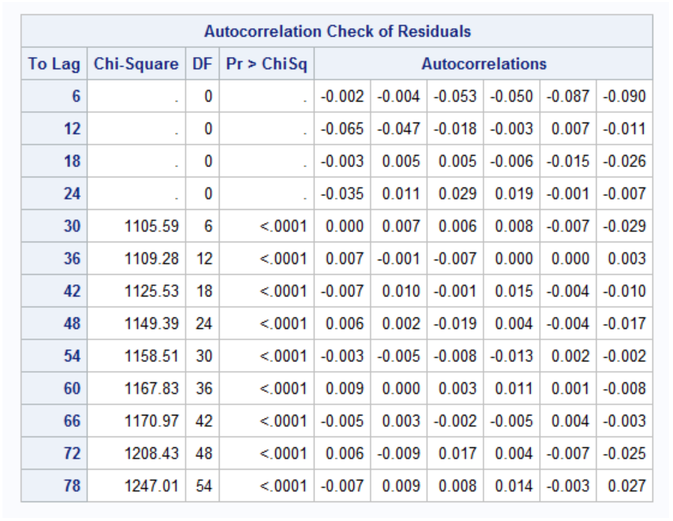
Figure 5: Correlation Analysis

Figure 5 shows the correlation analysis after differencing the input series. The differencing is mainly used to make the time series stationary. The Auto-Correlation plot dies down quite rapidly in just 2 or 3 lags, thus the time series is stationary. From Figure 5, we see that there is periodicity in the correlation. The Auto-Regressive terms can be found using partial Auto-Correlation plot. Lag 1, 2, 23, 24 are seen as significant terms.

3.3.2 Parameter Estimation and Diagnostics analysis

After considerable tuning of P and Q, we have come up with the following P and Q terms for our model which gives us very good MAPE. We have regressed price(LMP) on Load but with the error term of the regression model (called the noise series in ARIMA modeling terminology) assumed to be an ARMA(p) process. P = (1,2,23,24,25,26,27,46,47,48,49,70,71,72,73, 74,75,76,93,94,95,96,97,98)

For this model, the following figures explains about the residual after fitting the model. Figure 6. Auto-correlation of residuals tells us whether this model sufficiently explains the data properly or do we have to look for a more complex model. As we can see, the auto-correlations are almost close to 0. Figure 7 gives us a histogram of residuals and Q-Q plot to test its normality. We can clearly see that mean of residuals is 0 and uncorrelated with other lags. Therefore, this model very well explains the data and can be used to predict future values.



Autocorrelation Check of Residuals

To Lag	Chi-Square	DF	Pr > ChiSq	Autocorrelations					
6	.	0	.	-0.002	-0.004	-0.053	-0.050	-0.087	-0.090
12	.	0	.	-0.065	-0.047	-0.018	-0.003	0.007	-0.011
18	.	0	.	-0.003	0.005	0.005	-0.006	-0.015	-0.026
24	.	0	.	-0.035	0.011	0.029	0.019	-0.001	-0.007
30	1105.59	6	<.0001	0.000	0.007	0.006	0.008	-0.007	-0.029
36	1109.28	12	<.0001	0.007	-0.001	-0.007	0.000	0.000	0.003
42	1125.53	18	<.0001	-0.007	0.010	-0.001	0.015	-0.004	-0.010
48	1149.39	24	<.0001	0.006	0.002	-0.019	0.004	-0.004	-0.017
54	1158.51	30	<.0001	-0.003	-0.005	-0.008	-0.013	0.002	-0.002
60	1167.83	36	<.0001	0.009	0.000	0.003	0.011	0.001	-0.008
66	1170.97	42	<.0001	-0.005	0.003	-0.002	-0.005	0.004	-0.003
72	1208.43	48	<.0001	0.006	-0.009	0.017	0.004	-0.007	-0.025
78	1247.01	54	<.0001	-0.007	0.009	0.008	0.014	-0.003	0.027

Figure 6: Auto-correlation Check for residuals

3.3.3 Forecasting

Forecast for the next 24 hours is should in Figure 9. We can also find the 95% confidence interval of the forecast prices. One important thing to notice is that the width of the confidence interval is narrowed in the early morning and the width expands towards the end of the day. Therefore, this forecast method is not suitable when the forecasting period is more than 24 hours.

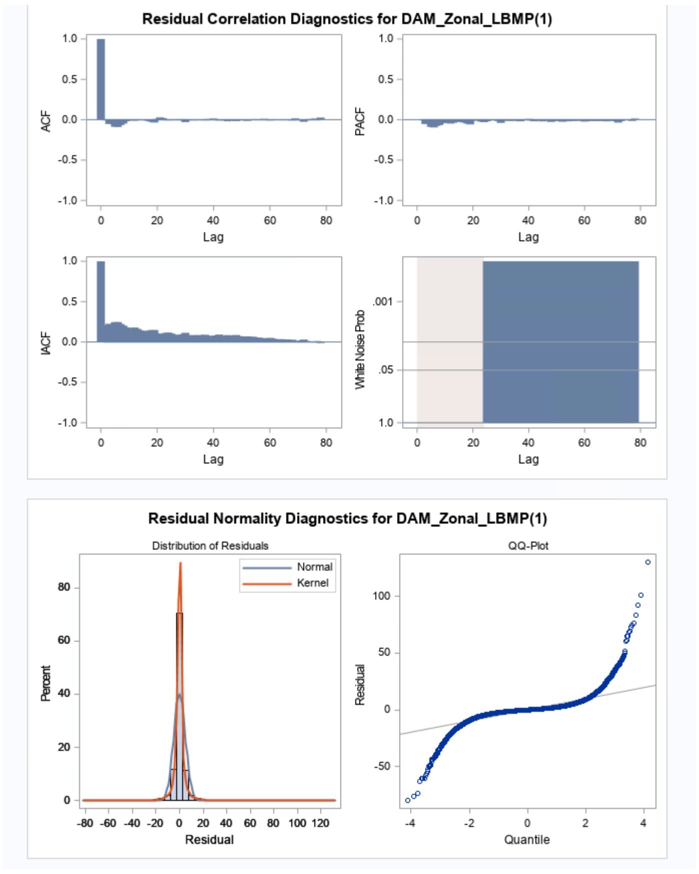


Figure 7: Residual Diagnostics

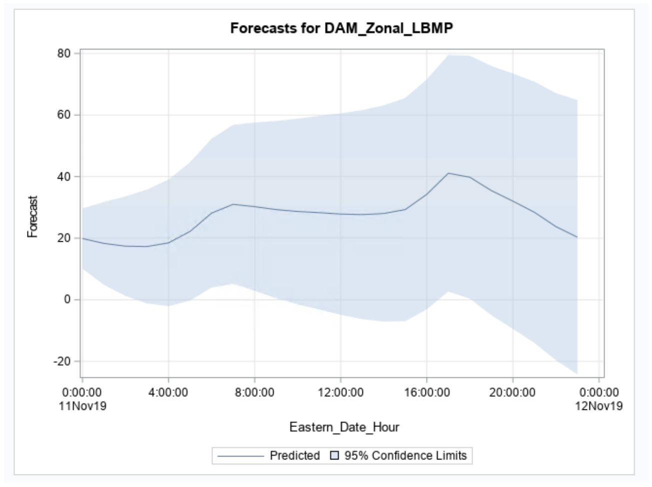


Figure 8: Forecast for the next 24 hours

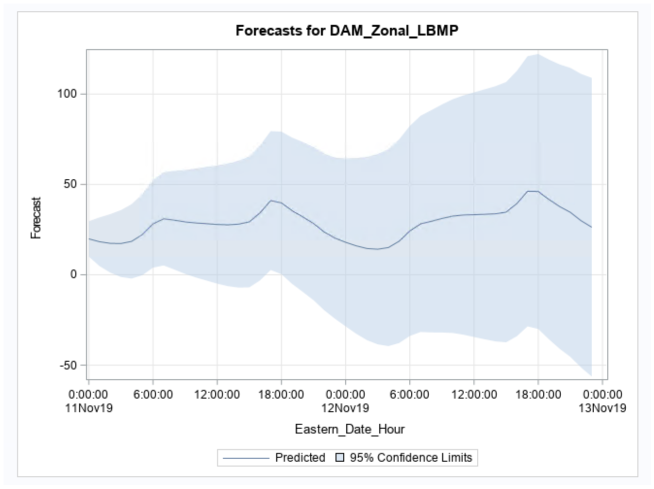
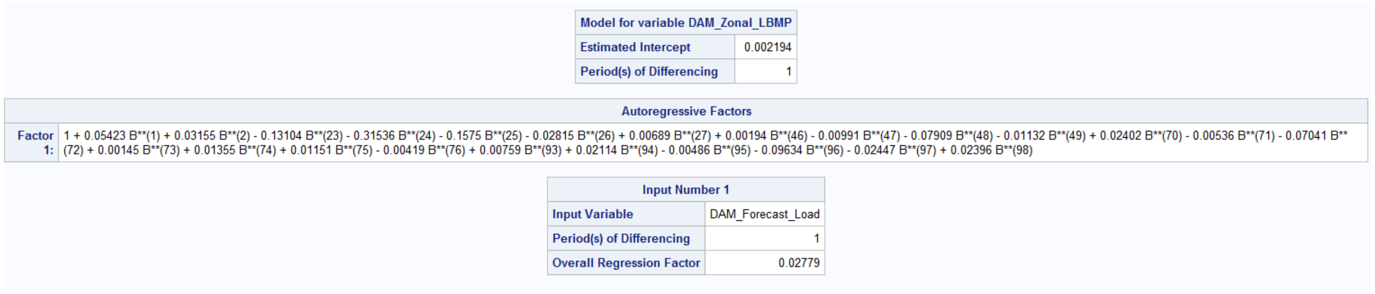


Figure 9: Forecast for the next 48 hours



Model for variable DAM_Zonal_LBMP	
Estimated Intercept	0.002194
Period(s) of Differencing	1

Autoregressive Factors	
Factor 1:	1 + 0.05423 B**(1) + 0.03155 B**(2) - 0.13104 B**(23) - 0.31536 B**(24) - 0.1575 B**(25) - 0.02815 B**(26) + 0.00689 B**(27) + 0.00194 B**(46) - 0.00991 B**(47) - 0.07909 B**(48) - 0.01132 B**(49) + 0.02402 B**(70) - 0.00536 B**(71) - 0.07041 B**(72) + 0.00145 B**(73) + 0.01355 B**(74) + 0.01151 B**(75) - 0.00419 B**(76) + 0.00759 B**(93) + 0.02114 B**(94) - 0.00486 B**(95) - 0.09634 B**(96) - 0.02447 B**(97) + 0.02396 B**(98)

Input Number 1	
Input Variable	DAM_Forecast_Load
Period(s) of Differencing	1
Overall Regression Factor	0.02779

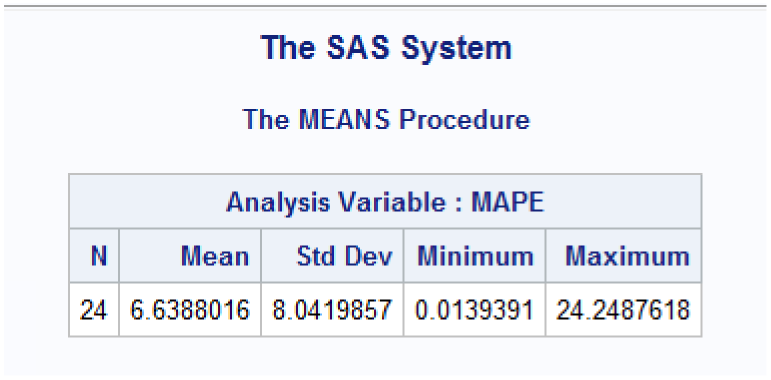
Figure 10: Model Equation

3.3.4 Model Validation

The mean shown in the below Figure 11 indicates the MAPE for this model.

$$\text{MAPE} = \frac{1}{T}\sum_{n=1}^{T}\frac{|ForecastedPrice - GroundTruth|}{GroundTruth} \times 100$$

Where T is forecast period



The SAS System

The MEANS Procedure

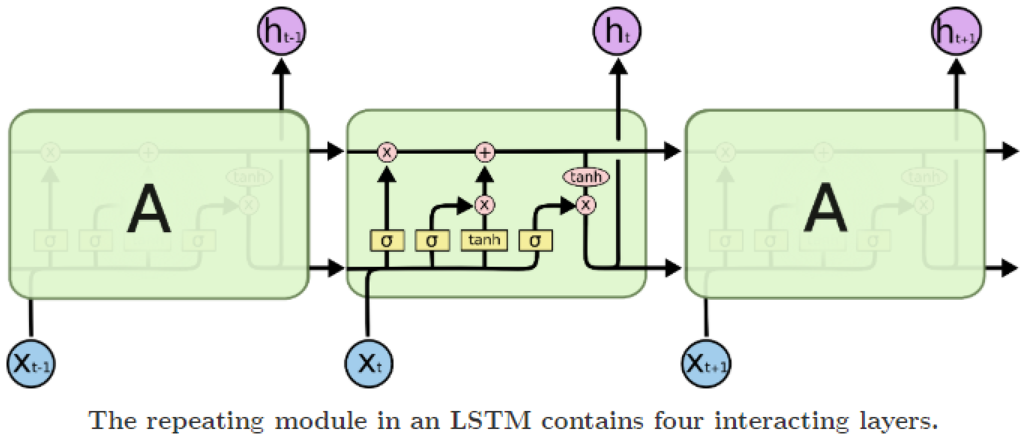
Analysis Variable : MAPE				
N	Mean	Std Dev	Minimum	Maximum
24	6.6388016	8.0419857	0.0139391	24.2487618

Figure 11: MAPE for ARIMA(p)

3.4 Long Short Term Memory (LSTM)

3.4.1 LSTM Model Parameters

Using state of the art machine learning libraries for deep learning such as Tensorflow with a Keras backend in Python, several types of LSTM models were constructed. The following table summarizes model architecture for the different LSTM models used in the study, with their optimal parameters and hyper-parameters. Based on LSTM model architecture, one can expect that it captures long-term as well as short-term dependencies in time-series forecasting data with better accuracy than ARIMA.



h_{t-1}
h_t
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
X_{t-1}
X_t
X_{t+1}
The repeating module in an LSTM contains four interacting layers.

Figure 12: Structure of LSTM

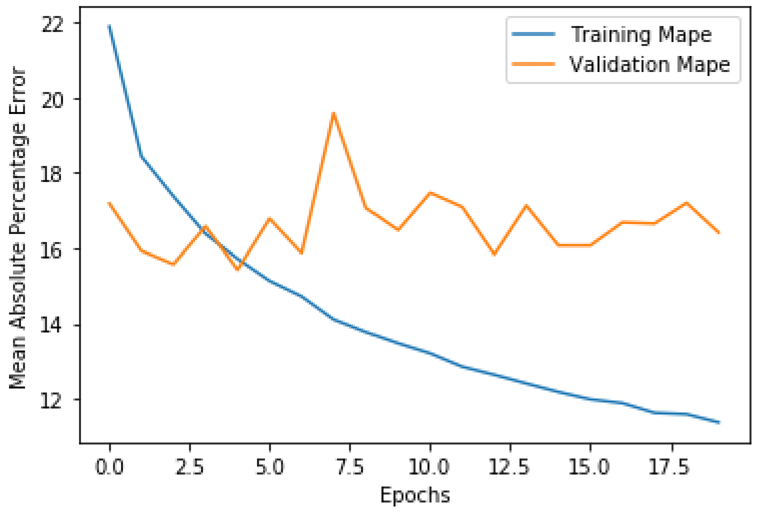


Figure 13: Learning Curve

4 Results and Discussions

In order to compare the performance of the multiple linear regression model with other models (ARIMA and LSTM), certain testing scenarios have been identified. The results for different test cases are captured in the tables below.

Table 3: MAPE in % for different models

Last training date : 11/09/2019			
Forecast Period	**Linear Regression**	**ARIMAX**	**LSTM**
48 Hour	—	5.8	20.64
24 Hour	12.95	4.52	6.55
12 Hour	12.86	2.85	3.24
4 Hour	12.66	2.21	1.35

Table 4: MAPE in % for different models

Last training date : 11/10/2019		
Forecast Period	**Linear Regression**	**ARIMAX**
48 Hour	—	10.55
24 Hour	10.83	7.24
12 Hour	10.44	10.95
4 Hour	5.37	21.86

Table 5: MAPE for different models

Last training date : 11/11/2019		
Forecast Period	**Linear Regression**	**ARIMAX**
48 Hour	—	20.17
24 Hour	6.17	6.62
12 Hour	7.38	9.12
4 Hour	6.28	12.49

Table 6: MAPE for different models

Average MAPE		
Forecast Period	**Linear Regression**	**ARIMAX**
48 Hour	—	12.53
24 Hour	9.88	6.15
12 Hour	10.22	7.63
4 Hour	8.10	12.31

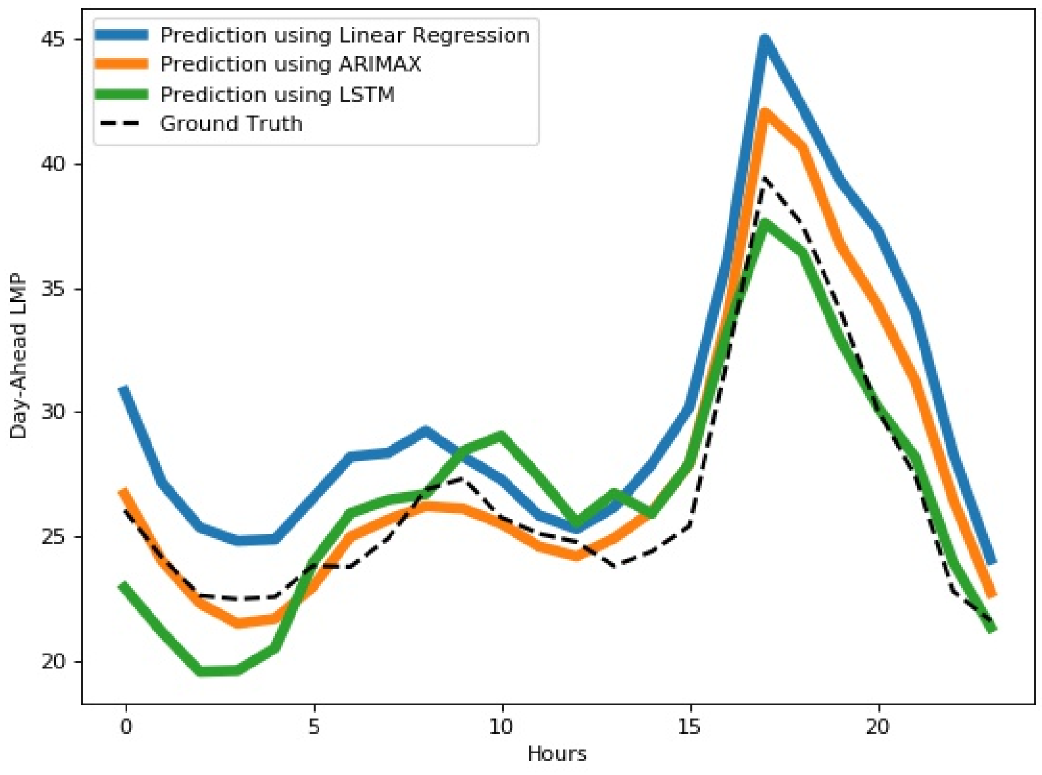


Figure 14: LMP ($\$/MWh$) Forecast for 11/10/2019

5 Summary and Conclusions

From our results, we can see that on average ARIMAX gives us a better prediction than LSTM or Linear Regression. For a shorter forecast period, we see that LSTM performs much better than ARIMA or Linear regression. In this project, we have desgined our model such that it works better for a 24 hour forecast period. We believe LSTM could perform better than ARIMA in case where we have large volume of data. This is because LSTM has lot of parameters to learn which is difficult to learn with less data. Using our linear regression model, we will not be able to predict more than 24 hours into the future. This is a drawback on our linear regression model. In our models, we have used forecast load as one of our input feature. As part of the future directions, we will be including solar and wind generation information as input feature to predict the 24 hour prices. ARIMAX is a powerful tool because it uses the inherent properties of time series to predict the prices, leading to a more concise model. This is possible because the model makes strong assumptions about the data, such as the true order of the AR process. These models, however, do not scale well for a large volume of training data, particularly if there are long-range dependencies or complex interactions. This is where LSTM can be used to learn from huge volume of data.

6 References

References

[1] Conejo, A., Plazas, M., Espinola, R., & Molina, A. (2005). Day-Ahead Electricity Price Forecasting Using the Wavelet Transform and ARIMA Models. IEEE Transactions on Power Systems, 20(2), 1035–1042. doi: 10.1109/tpwrs.2005.846054.

[2] D. H. Mazengia and Le Anh Tuan, ”Forecasting spot electricity market prices using time series models,” 2008 IEEE International Conference on Sustainable Energy Technologies, Singapore, 2008, pp. 1256-1261.doi: 10.1109/ICSET.2008.4747199.

[3] https://www.tutorialspoint.com/sas/sas_linear_regression.htm

[4] https://support.sas.com/documentation/onlinedoc/ets/132/arima.pdf

[5] https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/

[6] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

7 Appendix A: SAS Code

7.1 Linear Regression Code

```
    data /*price_load*/mylib.price_load_new;
set /*price_load*/mylib.price_load_new;
Hour=hour(Eastern_Date_Hour);
Date=datepart(Eastern_Date_Hour);
format date date9.;
Weekday=weekday(date);
Month=month(date);
/*datetime=Eastern_Date_Hour;*/
Ppd=lag24(DAM_Zonal_LBMP);
if _n_>=1 and _n_<=24 then Ppd=DAM_Zonal_LBMP;
Ppw=lag168(DAM_Zonal_LBMP);
if _n_>=1 and _n_<=168 then Ppw=DAM_Zonal_LBMP;
LMP=DAM_Zonal_LBMP;
Load=DAM_Forecast_Load;
Lpd=lag24(Load);
if _n_>=1 and _n_<=24 then Lpd=Load;
Lpw=lag168(Load);
if _n_>=1 and _n_<=168 then Lpw=Load;
run;

data /*price_load*/mylib.price_load_new;
set /*price_load*/mylib.price_load_new;
if hour=0 then hr0=1; else hr0=0;
if hour=1 then hr1=1; else hr1=0;
if hour=2 then hr2=1; else hr2=0;

if hour=3 then hr3=1; else hr3=0;

if hour=4 then hr4=1; else hr4=0;

if hour=5 then hr5=1; else hr5=0;

if hour=6 then hr6=1; else hr6=0;

if hour=7 then hr7=1; else hr7=0;

if hour=8 then hr8=1; else hr8=0;

if hour=9 then hr9=1; else hr9=0;

if hour=10 then hr10=1; else hr10=0;

if hour=11 then hr11=1; else hr11=0;

if hour=12 then hr12=1; else hr12=0;

if hour=13 then hr13=1; else hr13=0;

if hour=14 then hr14=1; else hr14=0;

if hour=15 then hr15=1; else hr15=0;

if hour=16 then hr16=1; else hr16=0;

if hour=17 then hr17=1; else hr17=0;

if hour=18 then hr18=1; else hr18=0;

if hour=19 then hr19=1; else hr19=0;

if hour=20 then hr20=1; else hr20=0;

if hour=21 then hr21=1; else hr21=0;

if hour=22 then hr22=1; else hr22=0;


if weekday=1 then wd1=1; else wd1=0;

if weekday=2 then wd2=1; else wd2=0;

if weekday=3 then wd3=1; else wd3=0;

if weekday=4 then wd4=1; else wd4=0;

if weekday=5 then wd5=1; else wd5=0;
if weekday=6 then wd6=1; else wd6=0;

if month=1 then m1=1; else m1=0;
if month=2 then m2=1; else m2=0;
if month=3 then m3=1; else m3=0;
if month=4 then m4=1; else m4=0;
if month=5 then m5=1; else m5=0;
if month=6 then m6=1; else m6=0;
if month=7 then m7=1; else m7=0;
if month=8 then m8=1; else m8=0;
if month=9 then m9=1; else m9=0;
if month=10 then m10=1; else m10=0;
if month=11 then m11=1; else m11=0;

run;



data train_val_loil;
set /*price_load*/mylib.price_load_new;
if (date>="01JAN2016"d and date<="09NOV2019"d);
run;

/*data test_loil;
set mylib.price_load_new;
if (date>="11NOV201900:"d and date<="11NOV2019"d);
run;*/
```

```
data test_loil;
set /*price_load*/mylib.price_load_new;
if (Eastern_Date_Hour>="10NOV19:00:00:00"dt and Eastern_Date_Hour<="10NOV19:23:00:00"dt)
run;

ods graphics on;
proc reg data=train_val_loil outest=model_base plots(maxpoints=none);
model_base: model LMP = Ppw Ppd Load Lpw Lpd hr0 hr1 hr2 hr3 hr4 hr5 hr6 hr7 hr8 hr9 hr
hr15 hr16 hr17 hr18 hr19 hr20 hr21 hr22 wd1 wd2 wd3 wd4 wd5 wd6 m1 m2 m3 m4 m5 m6 m7 m8
title'Multiple Linear Regression Model';
run;

proc score data=test_loil score=model_base type=parms predict out=load_pred_test;
   var Ppw Ppd Load Lpw Lpd hr0 hr1 hr2 hr3 hr4 hr5 hr6 hr7 hr8 hr9 hr10 hr11 hr12 hr13
hr15 hr16 hr17 hr18 hr19 hr20 hr21 hr22 wd1 wd2 wd3 wd4 wd5 wd6 m1 m2 m3 m4 m5 m6 m7 m8
run;

data mape_val;
set load_pred_test;
mape_val = 100*abs(model_base - LMP)/LMP;
run;

proc means data=mape_val mean;
  var mape_val;
   output out=mape_val;
   title 'Mape Value for Model-After Testing';
run;
```

```
proc print data=load_pred_test; var Eastern_Date_Hour LMP model_base;
run;
```

7.2 ARIMAX SAS CODE

```
PROC IMPORT OUT= WORK.daprice
            DATAFILE= "C:\Users\dvaradh\Desktop\Project\daprice_11_10.csv"
            DBMS=CSV REPLACE;
     GETNAMES=YES;
     DATAROW=2;
RUN;

PROC IMPORT OUT= WORK.VALIDATION
            DATAFILE= "C:\Users\dvaradh\Desktop\Project\OASIS_Day-Ahead_Market_Zonal_LBM
            DBMS=CSV REPLACE;
     GETNAMES=YES;
     DATAROW=2;
RUN;


DATA TRAINING;
SET WORK.DAPRICE;
WHERE EASTERN_DATE_HOUR >= '01JAN16:00:00'DT AND EASTERN_DATE_HOUR <= '12NOV19:23:00'DT
RUN;
```

```
DATA VALIDATION;

SET WORK.VALIDATION;

WHERE EASTERN_DATE_HOUR >= '11NOV19:00:00'DT AND EASTERN_DATE_HOUR <= '12NOV19:23:00'DT

RUN;


PROC ARIMA DATA = WORK.TRAINING;

IDENTIFY VAR = DAM_ZONAL_LBMP(1) CROSSCORR = DAM_FORECAST_LOAD(1) NLAG = 78;

ESTIMATE P=(1,2,23,24,25,26,27,46,47,48,49,70,71,72,73,74,75,94,96,97,118,120,121,123,1

 *P = (1,2,23,24,166,167,168,169,334,335,336,337) Q = 24 INPUT = DAM_FORECAST_LOAD(1);

*P=(1,2,23,24,25,26,27,46,47,48,49,70,71,72,73,74,75,76,93,94,95,96,97,98,118,120,121,1

forecast lead=24 interval=HOUR id=EASTERN_DATE_HOUR out=results;

RUN;


DATA ERROR;

SET WORK.RESULTS(where=(EASTERN_DATE_HOUR >='11NOV19:00:00:00'DT AND EASTERN_DATE_HOUR

SET WORK.VALIDATION(where=(EASTERN_DATE_HOUR >='11NOV19:00:00'DT AND EASTERN_DATE_HOUR

MAPE = ABS(FORECAST-DAM_ZONAL_LBMP)*100/DAM_ZONAL_LBMP;

RUN;


PROC CONTENTS DATA = WORK.ERROR;

RUN;


PROC MEANS DATA = WORK.ERROR;

VAR MAPE;

RUN;
```

7.3 LSTM Python Code

```
    import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
import keras
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential, load_model
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import GRU
from keras.layers import TimeDistributed
from keras.layers import Bidirectional
from keras.layers import Dropout
from keras.utils import to_categorical
import random

# Read price data and keep only required columns
def daprice_dwarak():
  # price_2 = pd.read_csv('/content/gdrive/My Drive/Final Project/Data Source/Price data
  # price_1 = pd.read_csv('/content/gdrive/My Drive/Topics in Data Science/Final Project

  price_2 = pd.read_csv('/content/gdrive/My Drive/Final Project/Data Source/Price data/2
  price_1 = pd.read_csv('/content/gdrive/My Drive/ECE 592 Topics in Data Science/Final P
  price_1.rename(columns = {'Eastern Date Hour':'Datetime', 'DAM Zonal LBMP': 'LBMP', 'I
  p1 = price_1[['Datetime', 'LBMP', 'Losses', 'Congestion']].copy()

  price_2.rename(columns = {'Eastern Date Hour':'Datetime', 'DAM Zonal LBMP': 'LBMP', 'I
  p2 = price_2[['Datetime', 'LBMP', 'Losses', 'Congestion']].copy()
  price_data = pd.concat([p1, p2], ignore_index= True)
  price_data



  load_2 = pd.read_csv('/content/gdrive/My Drive/Final Project/Data Source/Load data/20
  load_1 = pd.read_csv('/content/gdrive/My Drive/Final Project/Data Source/Load data/20



  load_1 = load_1[load_1['Zone Name'] == 'LONGIL'].copy()
  load_1.rename( columns= {'Eastern Date Hour':'Datetime', 'DAM Forecast Load':'Forecast
  l1 = load_1[['Datetime', 'Forecasted_load']]

  load_2 = load_2[load_2['Zone Name'] == 'LONGIL'].copy()
  load_2.rename( columns= {'Eastern Date Hour':'Datetime', 'DAM Forecast Load':'Forecast
  l2 = load_2[['Datetime', 'Forecasted_load']]

  load_data = pd.concat([l1, l2], ignore_index= True)
  load_data
  return price_data,load_data

 # merging data based on end-index date
price_data,load_data = daprice_dwarak()
```

```
end_index = load_data.index[load_data.Datetime == price_data.Datetime.iloc[-1]].tolist(
data = pd.merge(price_data , load_data[:end_index[0]+1], how = 'left', on = 'Datetime')


def create_model(X):

  # define model
  model = Sequential()
  model.add(Bidirectional(LSTM(100, return_sequences=True, input_shape=(X.shape[0], X.s
  #model.add(GRU(256, return_sequences=True, input_shape=(X.shape[1], X.shape[2])))

  #model.add(Bidirectional(LSTM(128)))
  model.add(TimeDistributed(Dense(128, activation='relu'), input_shape=(None, X.shape[1
 # model.add(TimeDistributed(Dense(64, activation='relu')))
 # model.add(TimeDistributed(Dense(32, activation='relu')))
  #model.add(TimeDistributed(Dense(5, activation = 'relu')))
  #model.add(Dense(64, activation='relu'))
  model.add(Flatten())
  model.add(Dense(512, activation='relu'))
  model.add(Dropout(0.3))
  #model.add(Dense(256, activation='relu'))
  #model.add(Dropout(0.4))
  #model.add(Dense(128, activation='relu'))
  #model.add(Dropout(0.4))
  model.add(Dense(12, activation='relu'))
  model.compile(optimizer='adam', loss=['mape'],  metrics=['mse', 'mape'])
  return model



# split a multivariate sequence into samples
def split_sequence_multi(sequence, n_steps):
  lbmp = sequence.LBMP.values
  load = sequence.Forecasted_load.values
  X, y = list(), list()
  for i in range(124,len(sequence)):
   end_ix = i + n_steps
   if end_ix+ n_steps > len(sequence) :
     break
   T1 = [1,2,23,24,169,170,191,192,365,366,368,376,377,378,379]
   T = [1,2,23,24,25,26,27,47,48,49,70,71,72,73,74,75,76,93,94,95,96,97,120,121,122,123]
   q = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26]
   #q = [1,2,3,4,5,6,7,8,20,21,22,23,24,25,26]
   #
   seq_x1 = [[lbmp[i-t]] for t in T]
   seq_x2 = [[load[i-t]] for t in T]
   seq_x3 = [[lbmp[i]-lbmp[i-t]] for t in T] # P term
   seq_x4 = [[lbmp[i-t]-lbmp[i-t-1]] for t in q]
   seq_x5 = [[load[i-t]-load[i-t-1]] for t in q]
   seq_y = lbmp[i:end_ix]
   X.append([seq_x3,seq_x4,seq_x5])
   y.append(seq_y)
  return (X), (y)
```

```
 from numpy import array
from keras.layers.normalization import BatchNormalization
from keras.layers import Bidirectional, Flatten
from keras.layers import TimeDistributed



n_steps = 12
X, y = split_sequence_multi(data, n_steps)

X = np.asarray(X)
y = np.asarray(y)
print(X.shape)
print(y.shape)
# # reshape from [samples, timesteps] into [samples, timesteps, features]
n_features = 3
X = X.reshape((X.shape[0], X.shape[2],n_features))



#from sklearn.model_selection import train_test_split
#X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.01, random_stat

model = create_model(X)

# fit model
history = model.fit(X, y, epochs=20, validation_split= 0.25, batch_size = 16)
```