Analysing the impact of renewable generation on the locational marginal price (LMP) forecast for California ISO

by

Chinmay Suhas Vad

A Thesis Presented in Partial Fulfilment
of the Requirements for the Degree
Master of Science

Approved April 2019 by the
Graduate Supervisory Committee:

Christiana B. Honsberg, Chair
Richard R. King
Sarah Kurtz

ARIZONA STATE UNIVERSITY

August 2019

ABSTRACT

Accurate forecasting of electricity prices has been a key factor for bidding strategies in the electricity markets. The increase in renewable generation due to large scale PV and wind deployment in California has led to an increase in day-ahead and real-time price volatility. This has also led to prices going negative due to the supply-demand imbalance caused by excess renewable generation during instances of low demand. This research focuses on applying machine learning models to analyse the impact of renewable generation on the hourly locational marginal prices (LMPs) for California Independent System Operator (CAISO). Historical data involving the load, renewable generation from solar and wind, fuel prices, aggregated generation outages is extracted and collected together in a dataset and used as features to train different machine learning models. Tree- based machine learning models such as Extra Trees, Gradient Boost, Extreme Gradient Boost (XGBoost) as well as models based on neural networks such as Long short term memory networks (LSTMs) are implemented for price forecasting. The focus is to capture the best relation between the features and the target LMP variable and determine the weight of every feature in determining the price.

The impact of renewable generation on LMP forecasting is determined for several different days in 2018. It is seen that the prices are impacted significantly by solar and wind generation and it ranks second in terms of impact after the electric load. The results of this research propose a method to evaluate the impact of several parameters on the day-ahead price forecast and would be useful for the grid operators to evaluate the parameters that could significantly impact the day-ahead price prediction and which parameters with low impact could be ignored to avoid an error in the forecast.

To my parents,

Mr. Suhas Madhukar Vad & Mrs. Seema Suhas Vad

and my grandparents,

Late Mr. Madhukar Raghunath Vad

&

Mrs. Sudha Madhukar Vad

ACKNOWLEDGMENTS

My sincere thanks to my advisor, Dr. Christiana Honsberg for believing in me from right when I took a course with her in Spring 2017. She gave me the chance to volunteer under her for a project which led to building up my interest in renewable energy. Getting an opportunity to work under her was the best thing happened to me during my Masters at Arizona State University.

I would also like to mention Dr. Sarah Kurtz, for being an excellent mentor and critic at times. I feel privileged to have you on my committee and would like to thank you for helping me learn the basics of writing a technical paper. You've been an excellent guide for our conference paper and it wouldn't have been possible to have a clear focus for the same without you.

My sincere thanks to Dr. Richard King who has been a pillar of support towards my thesis and the conference paper. Your inputs towards the conference paper were extremely valuable and helpful. A special mention to Dr. Stuart Bowden for taking out time to discuss his interesting projects and appreciating my inputs on the same.

Not many have thirteen priorities in their life and I am lucky to have those. The 'Wolfpack' made me the person I am today, and I feel extremely proud to have them in my life. Finally, I would like to thank my partner, Apoorva Joshi for her undying support and constant motivation. You believed in me more than I do in myself. And, my parents for their constant support and motivation.

Being more than 8000 miles away from home for around two years would not have been easy had it not been my roommates, Samir Chaudhari and Aditya Bheemavarapu. The constant support that you both provided is something I can be never thankful enough of.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

LIST OF SYMBOLS

ANN	Artificial Neural Networks
CAISO	California Independent System Operator
CSV	Comma-separated values
CV	Cross Validation
EIM	Energy Imbalance Market
GB	Gradient Boost
GW	Gigawatt
ISO	Independent System Operator
LMP	Locational Marginal Prices
LSTM	Long short term memory
MMBTU	One Million British Thermal Units
MISO	Midcontinent Independent System Operator
ML	Machine learning
MW	Megawatt
MWh	Megawatt-hour
NREL	National Renewable Energy Laboratory
OASIS	Open Access Same-Time Information System
PJM	Pennsylvania-New Jersey-Maryland Interconnection
PNode	Pricing Node
PTC	Production Tax Credit
PV	Photovoltaic
RF	Random Forest
RNN	Recurrent Neural Network

RTO	Regional Transmission Organization
RMSE	Root Mean Square Error
rRMSE	Relative Root Mean Square Error
XG	Extreme Gradient
XML	eXtensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 Problem Statement and Motivation

Accurate forecasting of electricity prices has been a key issue for competitive markets due to the increasing renewable penetration into the grid. Electricity markets followed the restructuring of the vertically integrated electric utility industry in the U.S. in the 1990's. While designing such competitive wholesale markets, the focus was on traditional forms of generation, including fossil, nuclear, and hydro. Since wind and solar had not flourished significantly then, little consideration was given to market design and operation under conditions of high penetrations of variable renewable resources [4]. Renewable energy sources have experienced tremendous investment and growth in the past decade in the United States due to technological advances and policy changes. Improved renewable generation and grid integration technology, combined with improved forecasting methods and changes to wholesale market rules and other subsidies and mandates have fostered tremendous growth in the amount of wind and solar power utilized in the U.S. [7]. Variable renewable generation presents a challenge to market design due to the intermittent nature and uncertainty associated with plant output. The integration of a significant amount of wind and solar power into a power system results in important operational challenges, which in turn originate alterations in electricity prices. The problem of forecasting locational marginal price (LMP)in a deregulated electricity market is important to both system operators and market participants. Accurate LMP forecasts produced in real-time are essential for demand response, revenue and risk management, and an efficient operation of a smart grid [2].

The thesis focuses on analysing the impact of renewable generation on day-ahead locational marginal price (LMP) forecast for California ISO. The electricity market for California ISO is split into three zones: NP-15, ZP-26, SP-15. All of these are trading zones which were created to facilitate bilateral transactions between energy buyers and sellers. Fig.1.1 shows the trading zones for California ISO.

Figure 1.1 Trading zones for California ISO [39]

1.2 Understanding the Supply-Demand Curve

The prices in the electricity market are determined on a day-ahead or real-time basis by looking at the demand and supply. For the day-ahead markets, generators offer certain amount of power for a price while the buyers lodge the bids to buy power at a certain price. This bidding process for the next day goes on till 12 noon every day. The bids placed for buying and selling power can trace two curves. The point of intersection

of those two curves determine the 'market clearing price' which is paid to every successful bid. Fig 1,2,1 shows an example of a supply-demand curve in Germany.

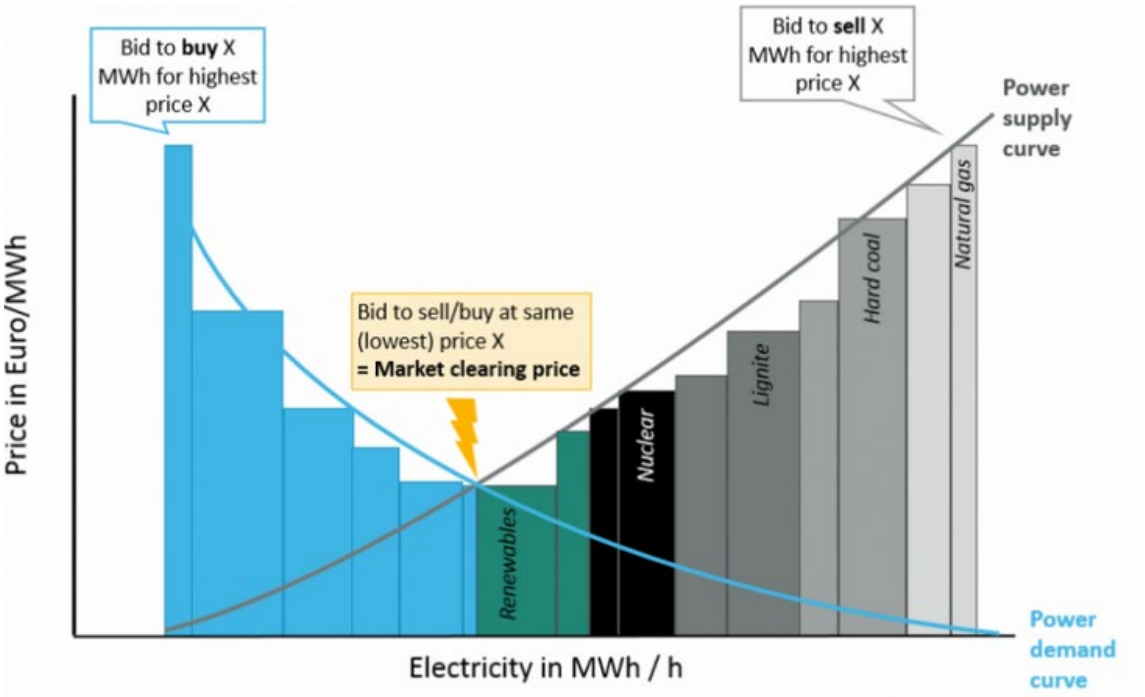

Figure 1.2.1 Example of a supply-demand curve based on a German day-ahead power market [34]

However, there might be instances when supply would exceed the demand. This occurs when there are instances with low demand and at the same time, the generation is inflexible to be ramped down. Renewable generation usually peaks in the middle of the day and if there's low demand observed during the same, there might be several instances of prices going below zero for even an hour or so. This would lead to setting the 'market clearing price' less than zero. Fig 1.2.2 shows the supply-demand curve for a scenario when supply exceeds the demand. Consider an example when a lignite plant with a low ramp rate was running at its full capacity to meet the power demand. At some point in the middle of the day, PV production increases and reaches its peak, creating an imbalance in the supply and demand. In such cases, grid operators can either curtail the PV or can curtail the generation of the lignite plant. Curtailing the generation from the lignite plant is difficult since there are policy issues and even risks

of damaging the plant. If the operator does not choose any of the above actions, there occurs an instance of negative price.

The other reason for a below-zero price is the tax credit that is offered to wind farms. Several wind farms receive a Production Tax Credit (PTC) for every MWh of energy they produce. This gives them an incentive to produce as much power as possible and in such cases, the plants can even accept a negative price. For example, if the PTC offered to a wind farm is $20/MWh and the price is -$5/MWh, the farm still earns a net revenue of $15/MWh. Hence, some wind generators submit their supply bids at a negative price in the day-ahead and real-time market but only when they have strong confidence in their offer being the cheapest. [35]

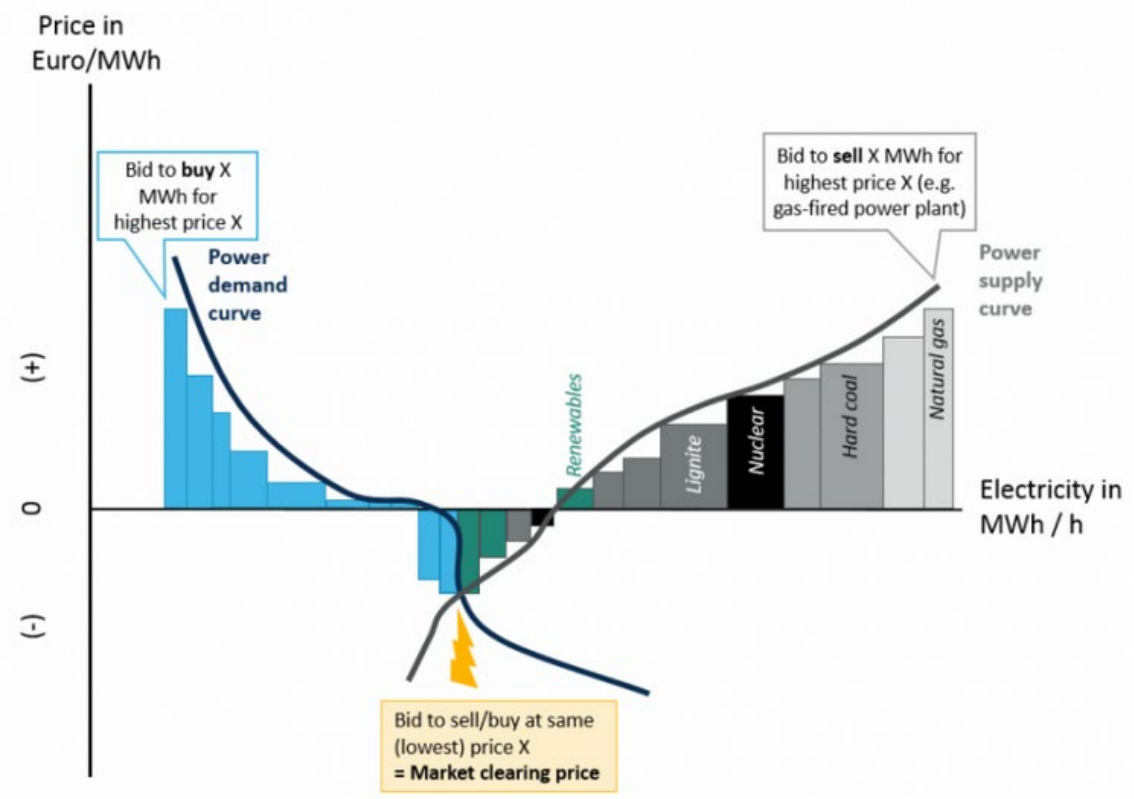

Figure 1.2.2 Example of a supply-demand curve for instances when supply exceeds the demand in a German day-ahead power market [34]

1.3 Locational Marginal Price (LMP) and Trading Zone Price Definition

Locational Marginal Pricing (LMP) represent the cost to buy and sell power at different locations within wholesale markets, usually called Regional Transmission Organizations (RTOs). There are seven RTOs in United States: CAISO (California Independent System Operator), ERCOT (Electric Reliability Council of Texas), PJM (Pennsylvania, Jersey and Maryland power pool), NEISO (New England ISO), Southwest Power Pool (SPP), NYISO (New York ISO) and MISO (Midcontinent ISO). Fig 1.3 shows the location of RTOs when United States.

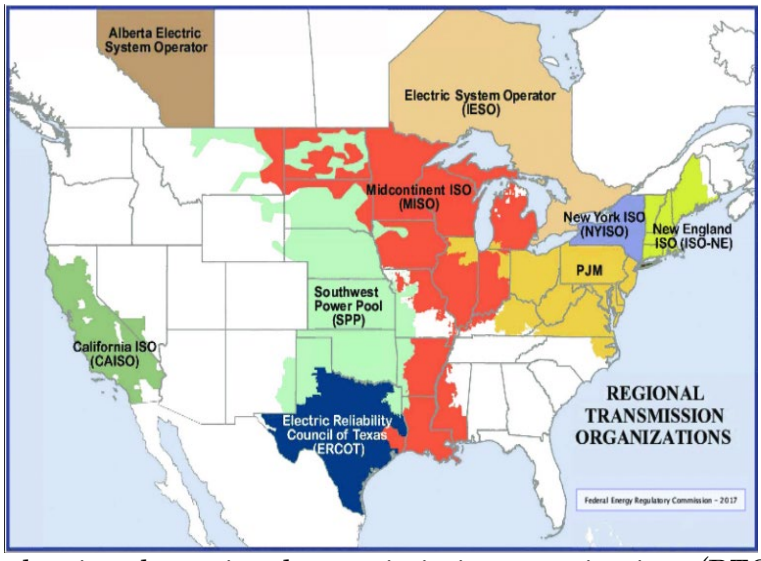

Figure 1.3.1 Map showing the regional tranmimission organizations (RTOs) within United States [38]

RTOs have Day Ahead and Real Time LMPs. Day-ahead LMPs represent prices in day-ahead markets which let market participants buy and sell wholesale electricity a day before the operating day to avoid volatility. Real-time LMPs represent prices in real time markets which let participants buy and sell power during the day of operation. For example, if an

area under an RTO expects a demand of 50 MW to occur tomorrow at 2 pm, they would buy 50 MW of electricity to be delivered at 2 pm tomorrow on the day-ahead market. However, at 2 pm on the next day, demand is actually 55 MW, then the additional 5 MW would be bought on the real-time market. Real-time market prices are generally more volatile than day-ahead market prices [19].

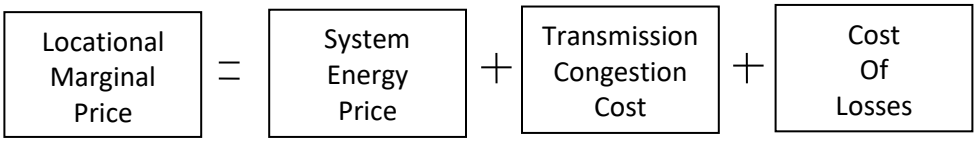

Figure 1.3: Components of LMP

Typically, LMP is the cost for the operator to deliver one additional MW to a bus/node in the network. LMP comprises of three components (see Fig.2.2).

1. Energy: The price of energy at the reference node. The energy component does not vary with location. The price at the reference bus is the load-weighted average of the system node prices.
2. Congestion: The congestion component reflects the marginal cost of congestion at a given node relative to the reference node. Congestion is location-sensitive and arises when there are binding constraints on the transmission system.
3. Losses: Represent the transmission losses and are locational sensitive. They reflect the cost of losses at that location relative to the load-weighted average of system node prices.

CAISO constitutes thousands of pricing nodes or PNodes under its area of operation. A pricing node or PNode is defined as a point where power generation or withdrawal is modelled within a system. LMP is calculated at such nodes and used for financial settlements [40]. Fig 1.3.3 shows the distribution of PNodes for CAISO in 2019.

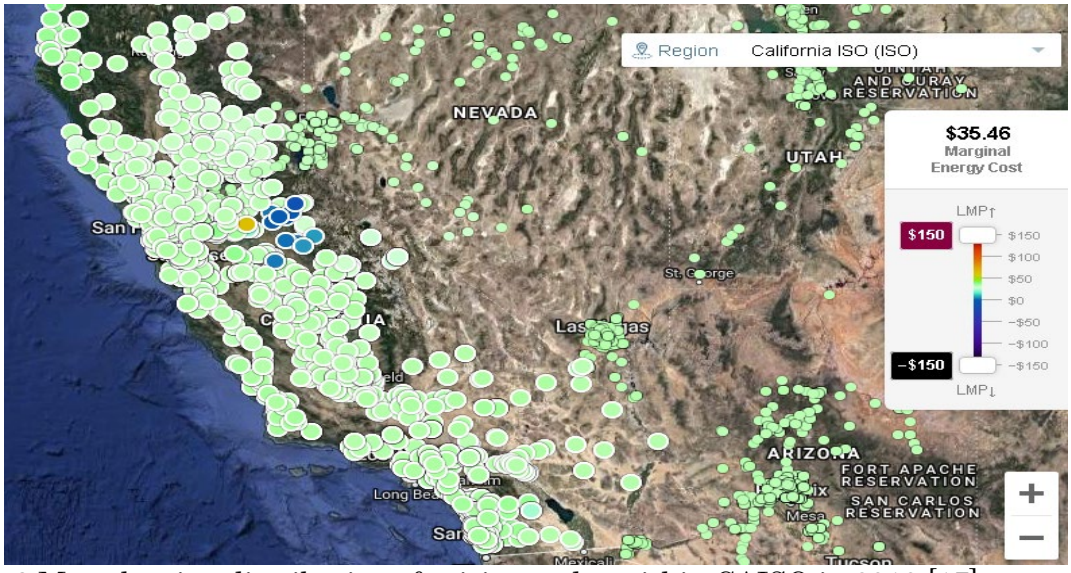

Fig 1.3.3 Map showing distribution of pricing nodes within CAISO in 2019 [17]

A trading zone would constitute several such PNodes and would have its own price called as trading zone LMP. For this research, the trading zone LMP is calculated by simply averaging LMPs of all nodes that are a part of the zone.

1.4 Thesis Outline

Chapter 1 identifies the problem and its importance, the history of electricity markets in United States, explaining the supply-demand curve to show how the prices are determined for the next day and defining how nodal and trading zone LMPs are calculated.

Chapter 2 focuses on the literature survey and provides the methods used for price forecasting in different electricity markets in the United States.

Chapter 3 gives a detailed procedure on how the dataset for the thesis was created and the methods used for cleaning and processing the data for further analysis.

Chapter 4 provides a detailed analysis of the dataset described in Chapter 3 showing the yearly variation along with the average yearly data. It also describes the correlation between prices and renewable generation through different plots.

Chapter 5 describes in detail the different machine learning (ML) algorithms used in this thesis and the terminologies used in machine learning. Also, it provides the forecasting procedure applied on the data. Chapter 5 also includes methods to tune the model for better accuracy along with a way to determine the weight or impact of every feature on the price prediction.

Chapter 6 shows the results which are the comparison between the forecasting methods described in Chapter 5. Also, Chapter 6 describes the weights of every feature in predicting the price for every machine learning method.

Chapter 7 is the final chapter which provides a conclusion and discusses the scope of future work on this topic.

CHAPTER 2

LITERATURE SURVEY

Day-ahead electricity price forecasting has been performed by several authors by methods such as time series models, recurrent neural networks (RNNs) and simulations tools. [2] provides a state-space approach to predict LMPs built by analysing the PJM real-time pricing model. The results are obtained using a Monte-Carlo simulation and accuracy is compared with the artificial neural network (ANN) forecasting results. The authors of [3] use random forest (RF), an ensemble learning model to predict day-ahead hourly LMPs for CAISO. The study performed in [3] focuses on the prices from January 2014 to February 2016 and is quite accurate. [33] includes applying deep learning to forecast extreme loads observed in PJM. The results show that their model performs better than the traditional Fourier series methods. The thesis extends the approach applied in [3] and [33] and compares different methods to analyse the impact of renewable generation on price forecasting.

[5] studies the impact of variable renewable generation on LMPs for several RTOs and creates a model to build scenarios which observe the impact with low and high solar and wind generation. [6] gives a short introduction to the demand-supply curve which shows how the price is determined and conducts a case study to demonstrate the effect of increasing renewable penetration on the LMPs. The thesis correlates with the work done by the authors of [7] but differs in terms of the dataset and methods applied for forecasting. The authors of [7] analyse the impact of increasing wind generation on Midcontinent ISO (MISO) pricing using regression-based techniques. [10] provides similar research as conducted in [7] but in more depth and including several regional transmission organizations (RTOs) in United States. The article mentioned in [11] tries to establish a correlation between the negative prices and solar generation by studying the pricing data

for several Regional Transmission Organizations (RTOs). [12] strongly relates to a section of this thesis which is to determine the correlation between the increasing solar generation with negative prices for CAISO.

[13]-[16] and [23]-[32] provide a clear description of the models used for price forecasting in the thesis. Also, they provide a good understanding on how to tune the model for higher accuracy and determining the impact of every quantity included the dataset on the price forecast.

The primary goal of this thesis is to compare several advanced machine learning algorithms such as the algorithm developed in [33] to determine the impact of renewable generation on electricity price forecasting. Chapter 5 would cover the algorithms in detail.

CHAPTER 3

DATA COLLECTION AND CLEANING

3.1 Data Collection

The thesis involves working with historical data for California ISO (CAISO). The ISO provides an Open Access Same-Time Information System (OASIS) which includes real-time data related to the ISO transmission systems and the electricity market for CAISO. [20]. OASIS allows the user to download data for a maximum duration of 30 days either through its interface or through querying the API. The data are available either in XML format or as a ZIP file which contains the .CSV document [20].

Data were collected for the following quantities:

1. Fuel Prices

   Hourly gas prices for each day in $/mmBtu by fuel region

2. Renewable Generation

   5-minute data for actual wind and solar generation in MW for every hour of the day aggregated by trading zones

3. System Load

   Load in 5-minute intervals for every hour of the day in MW aggregated by trading zones

4. Locational Marginal Prices (LMP)

   Hourly LMPs for all nodes in $/MWh

5. Aggregated Generation Outages

   Hourly data for each day which includes generator de-rates and outages in MW split by trading zones (SP-15, NP-15, ZP-26) and resource type (thermal, hydro, renewable).

The final dataset includes the above data over a period of four years (2015-2018) except for generation outages which has three-year data (2016-2018). This thesis involves analysing the data using Python and hence .csv files were preferred over .xml files.

3.2 Data Cleaning & Processing

The thesis involves performing analysis for the data over a year, hence, multiple .csv files need to be aggregated together to generate the yearly data. For the renewable generation and the load, the dataset involves values for every 5 minutes in an hour and is averaged to produce the hourly values. Also, there are several hours missing in most of the datasets. Around 80-100 hours were seen to be missing from every year. These missing values are filled by averaging the values for the missing hour for the previous and the next seven days.

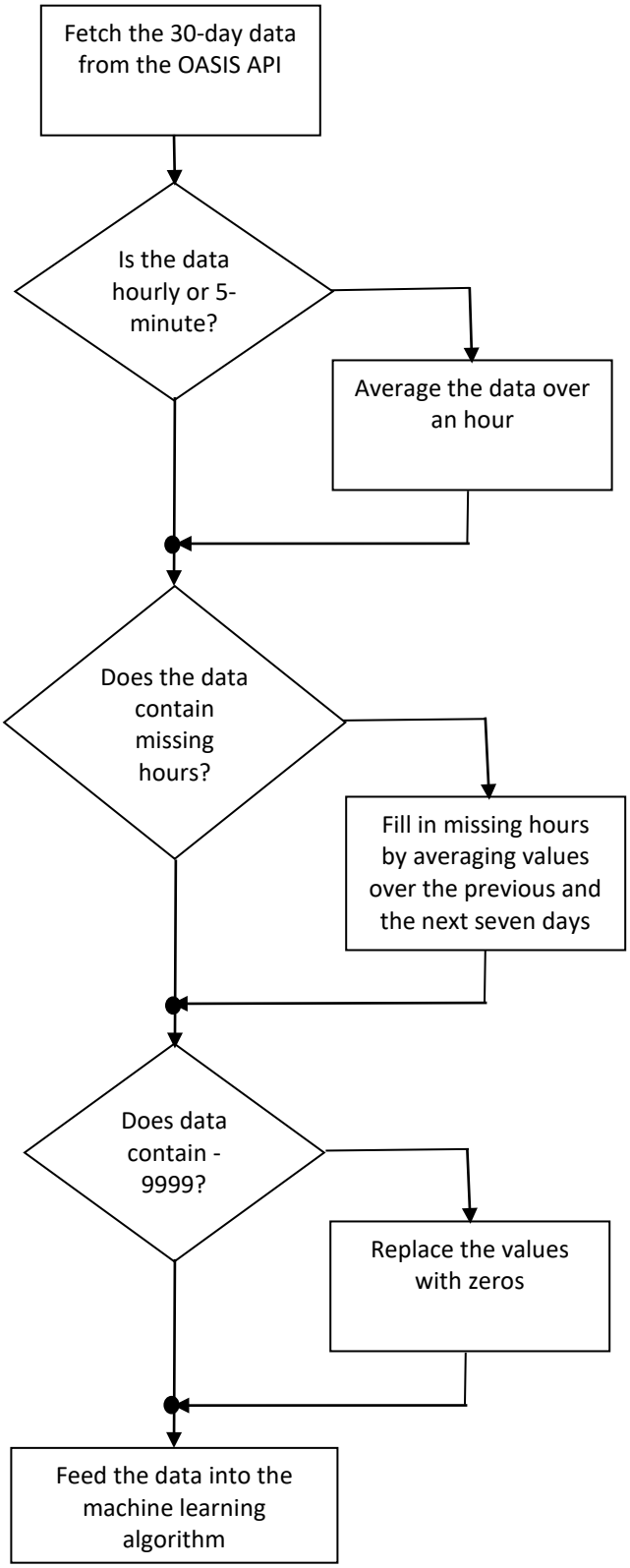

Figure 3.2. Flowchart describing the data cleaning and processing algorithm

CHAPTER 4

ANALYSING THE ISO DATA

4.1 Analysis of Fuel Prices and Electric Load

The variation in every quantity described in Chapter 4 gives a picture of the trend followed throughout the last few years. Fig 4.1.1 and 4.1.2 show fuel price variation and average yearly fuel price from 2015-2018. Average fuel prices have dropped compared to 2015, and 2018 experienced a third-quarter high gas price window due to high prices reported at the SoCal Citygate trading hub. [21]

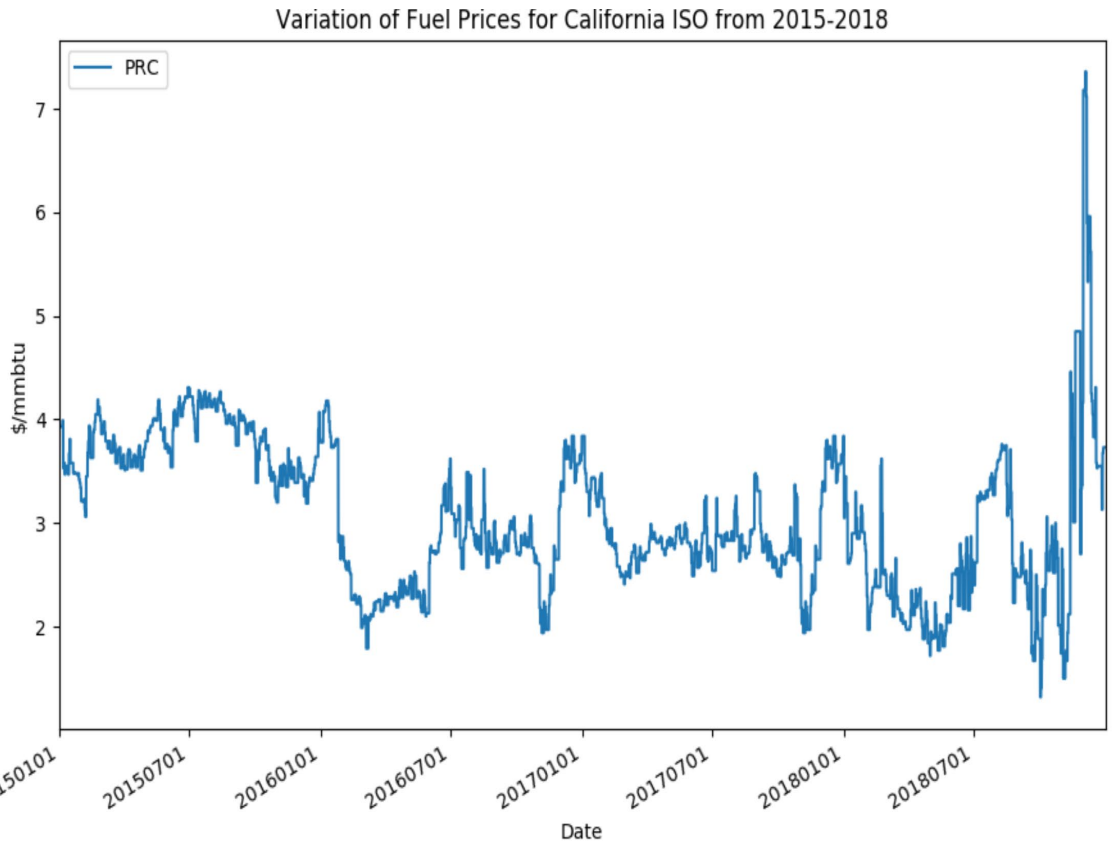

Figure 4.1.1 Plot showing the variation of fuel prices from 2015-2018

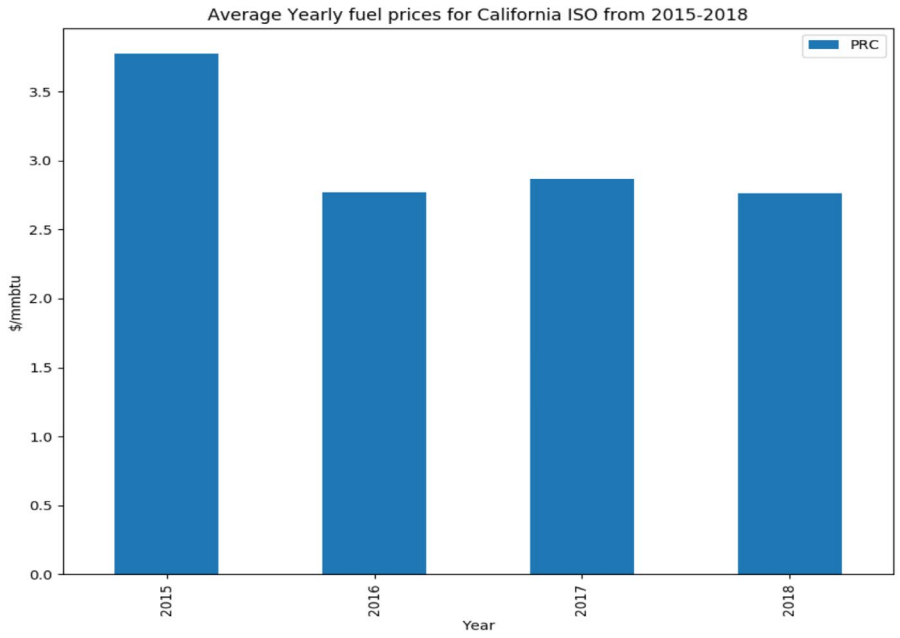

Figure 4.1.2 Average fuel prices by year from 2015-2018

Fig 4.1.3 shows the hourly electric load variation from 2015-2018. It is seen that the electric load profile looks quite similar throughout the last few years. The average load for the entire year has not changed significantly from 2015-2018 as shown in Fig 4.1.4

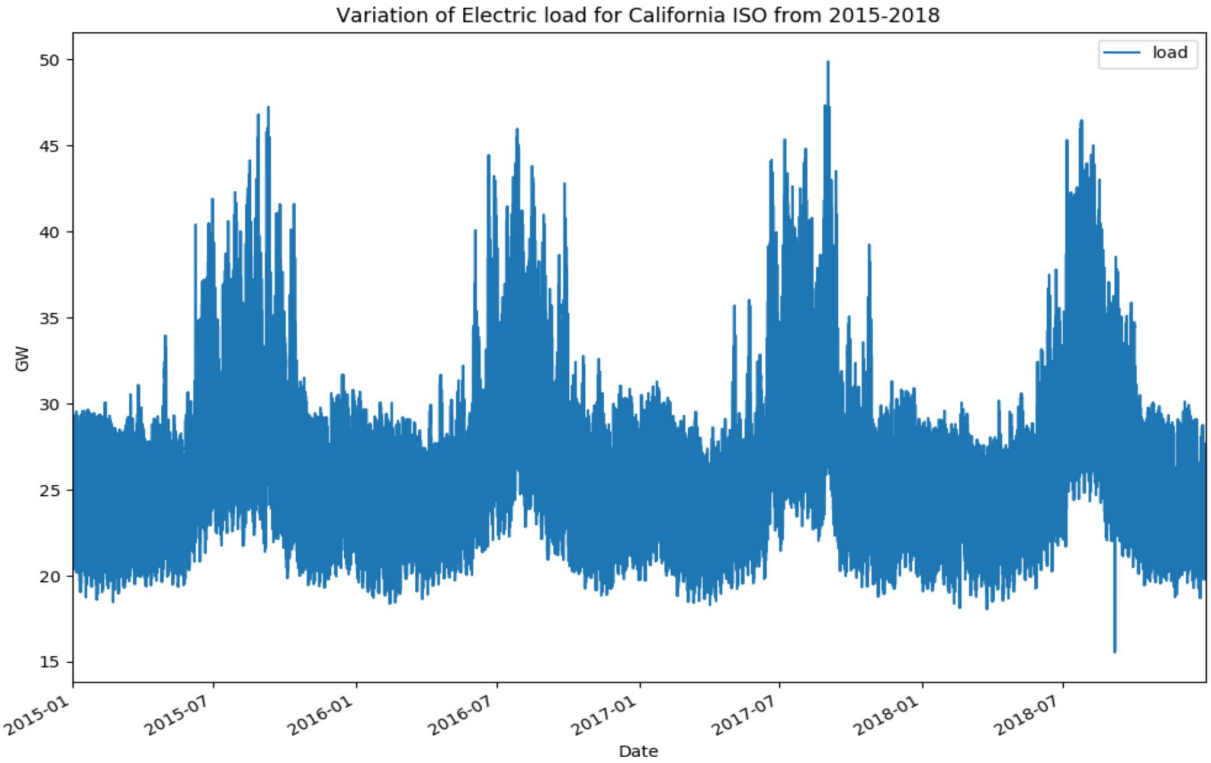

Figure 4.1.3 Electric Load Variation from 2015-2018

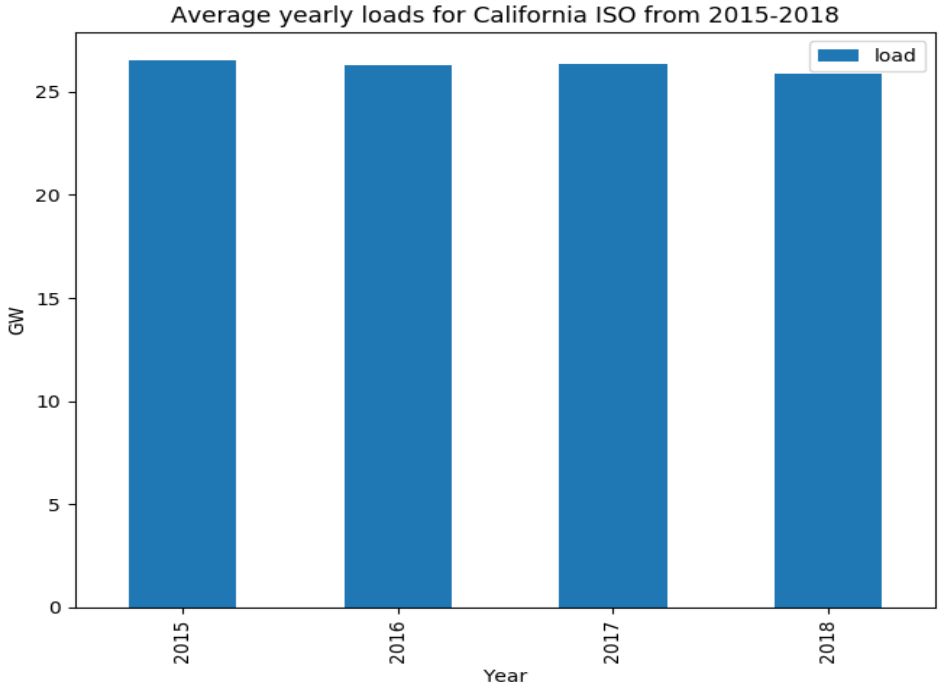

Figure 4.1.4 Average yearly load in GW from 2015-2018

4.2 Analysis of Renewable Generation and Generation Outages

The focus of this thesis, renewable generation, has increased significantly over the last three years. Fig 4.2.1 shows the same and Fig 4.2.2 clearly states the increase in PV and Wind deployment over the past few years.

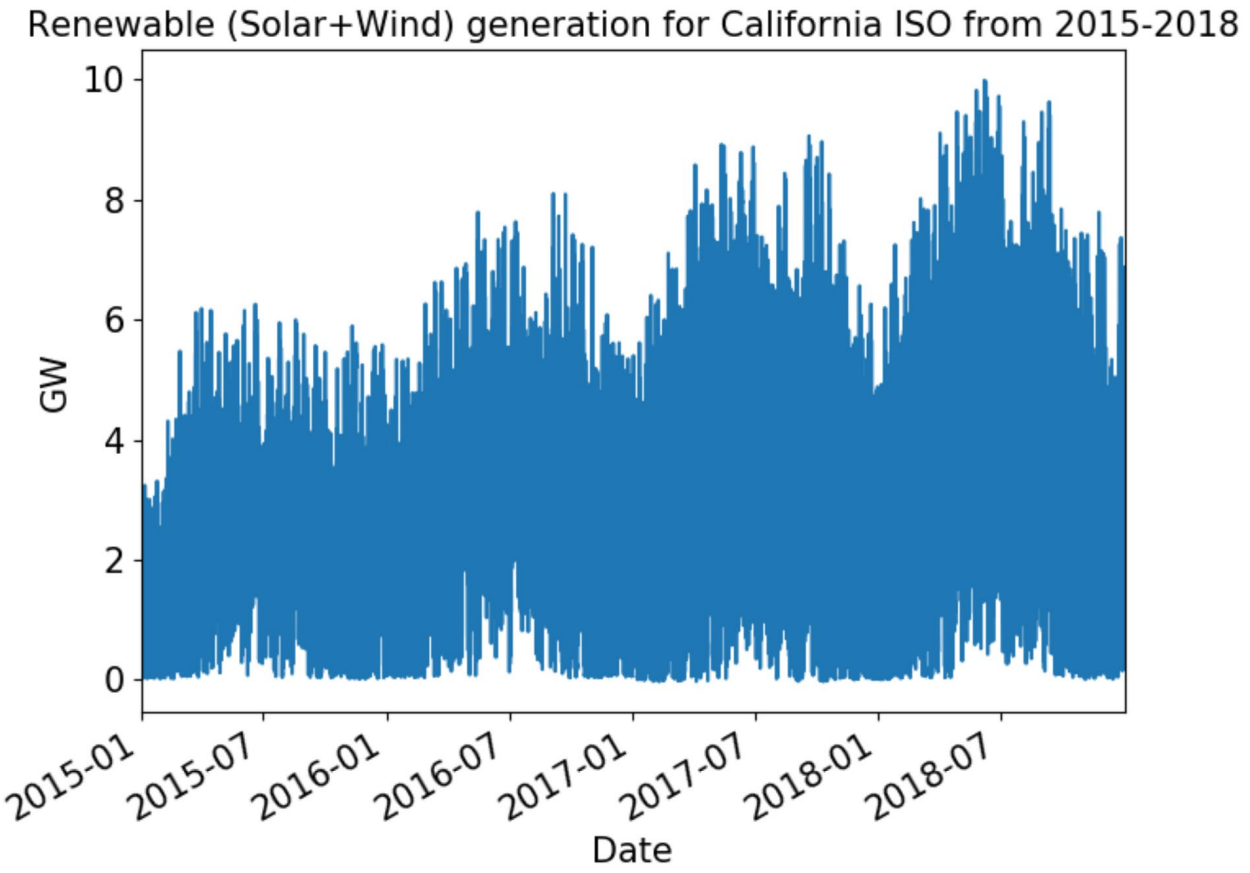

Figure 4.2.1 Renewable (Solar+Wind) Generation from 2015—2018

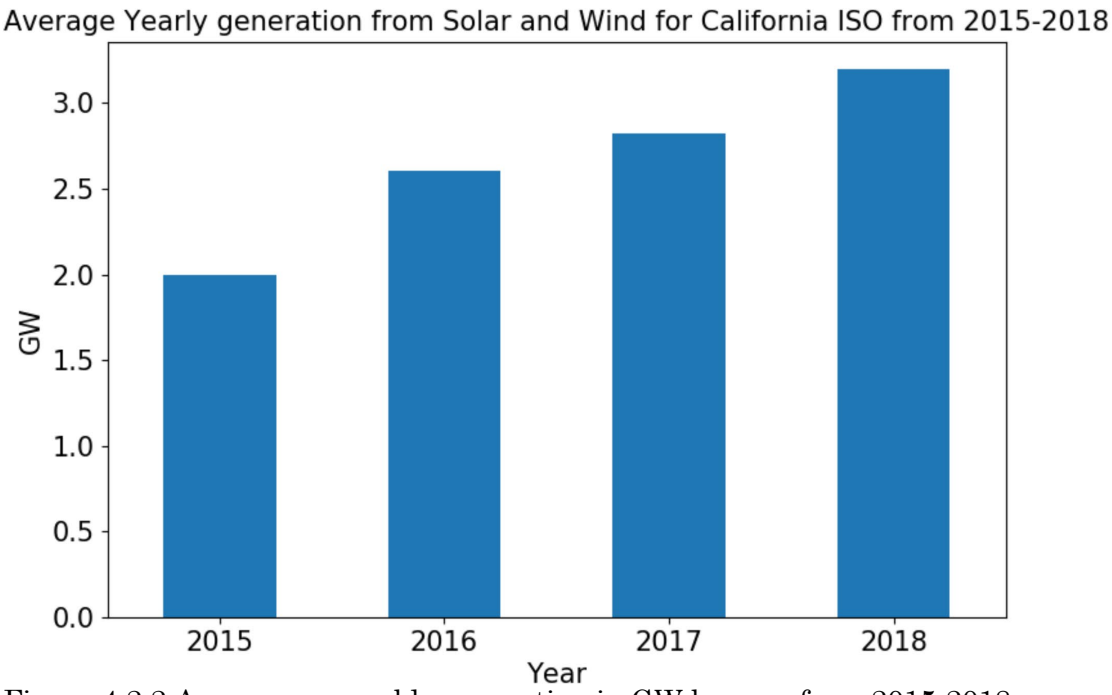

Figure 4.2.2 Average renewable generation in GW by year from 2015-2018

Generation Outages account for aggregated outages from sources such as thermal, hydroelectric and renewable power plants. Fig 4.2.3 represent the aggregated generation outages from 2016-2018. It is seen from Fig 4.2.4 that average generation outages have decreased in comparison to 2016.

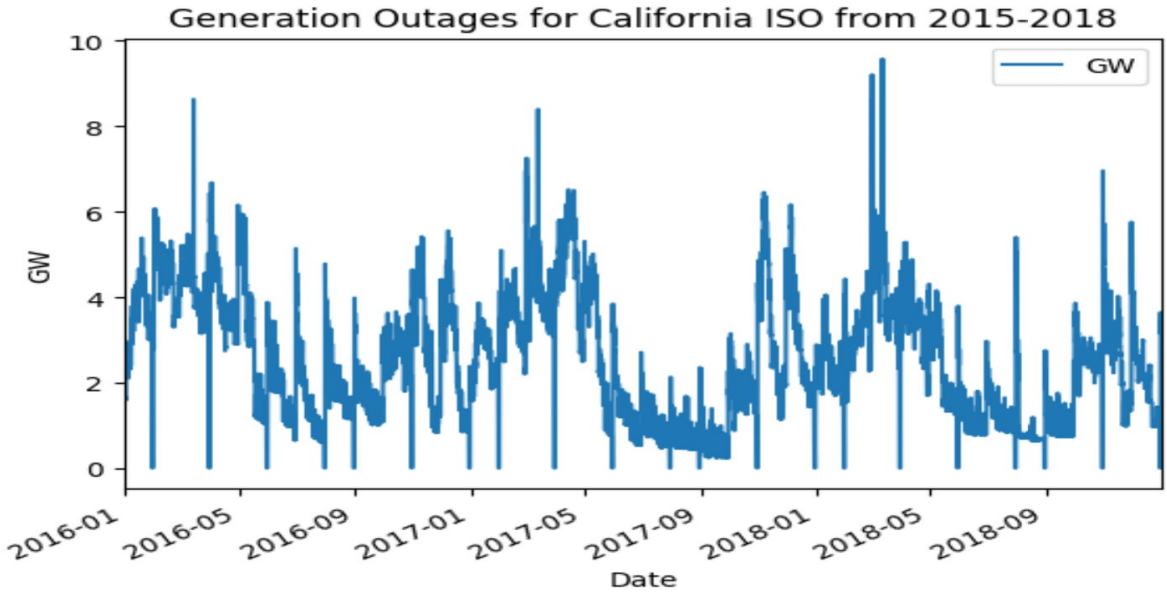

Figure 4.2.3 Aggregated Generation Outages in GW from 2016-2018

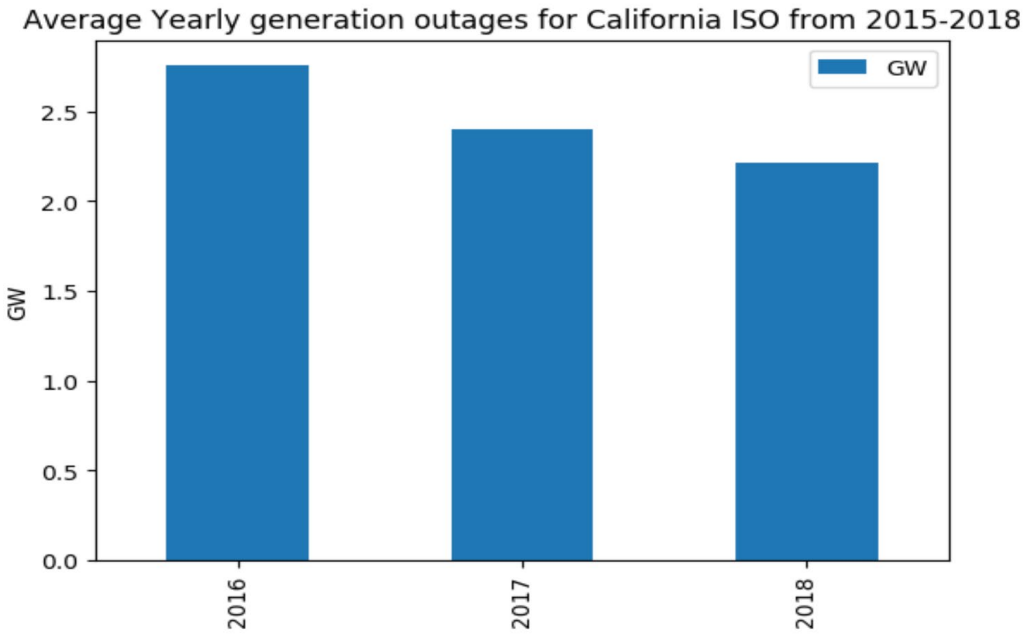

Figure 4.2.4 Average generation outages in GW by year from 2016-2018

4.3 Variation of Hourly LMPs

Fig 4.3.1 and Fig 4.3.2 show the variation of hourly LMPs for SP-15 hub from 2015-2018. The average price has increased by around $4/MWh in comparison to 2015.

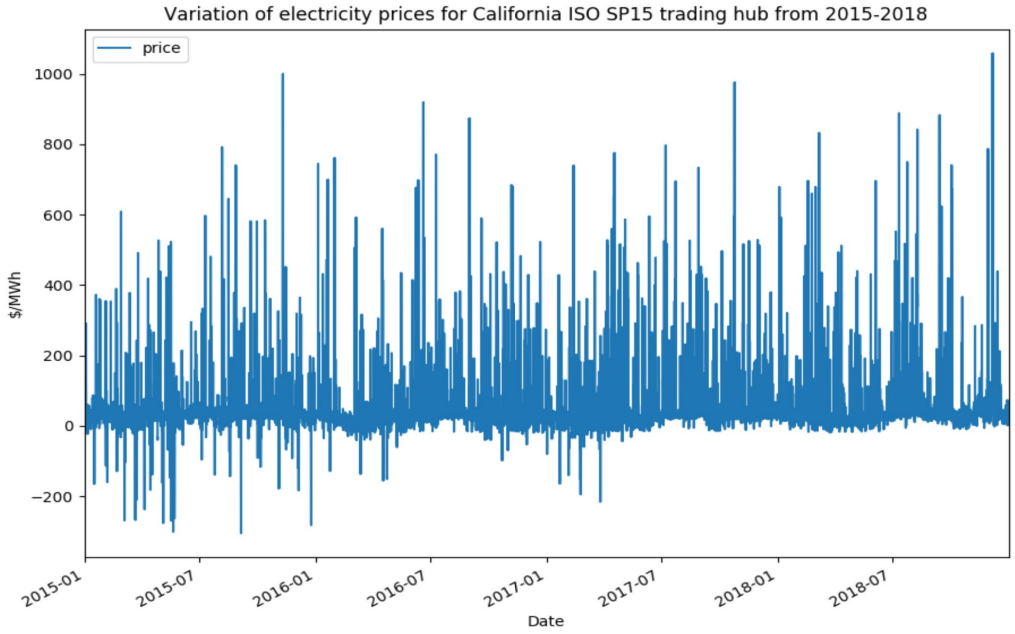

Figure 4.3.1 Plot showing hourly LMPs for CAISO SP-15 from 2015-2018

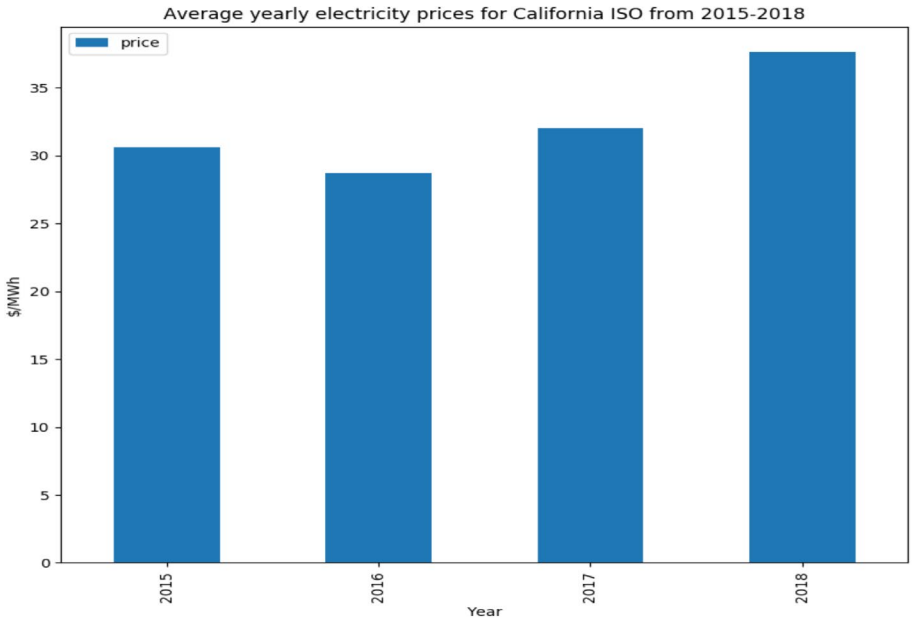

Figure 4.3.2 Average LMP by year for CAISO SP-15 from 2015-2018

4.4 Impact of Renewable Generation on Hourly LMPs

Figures 4.4.1 to 4.4.3 clearly show that renewable generation cause a dip in the hourly LMPs. The prices are compared with the renewable generation by solar and wind for the first week of April in 2016, 2017 and 2018. A clear reduction in prices is seen when renewable generation is close to its peak even forcing the prices to go negative. Also, another observation is the sharp increase in prices for most of the cases when there are instances of minimum renewable generation.

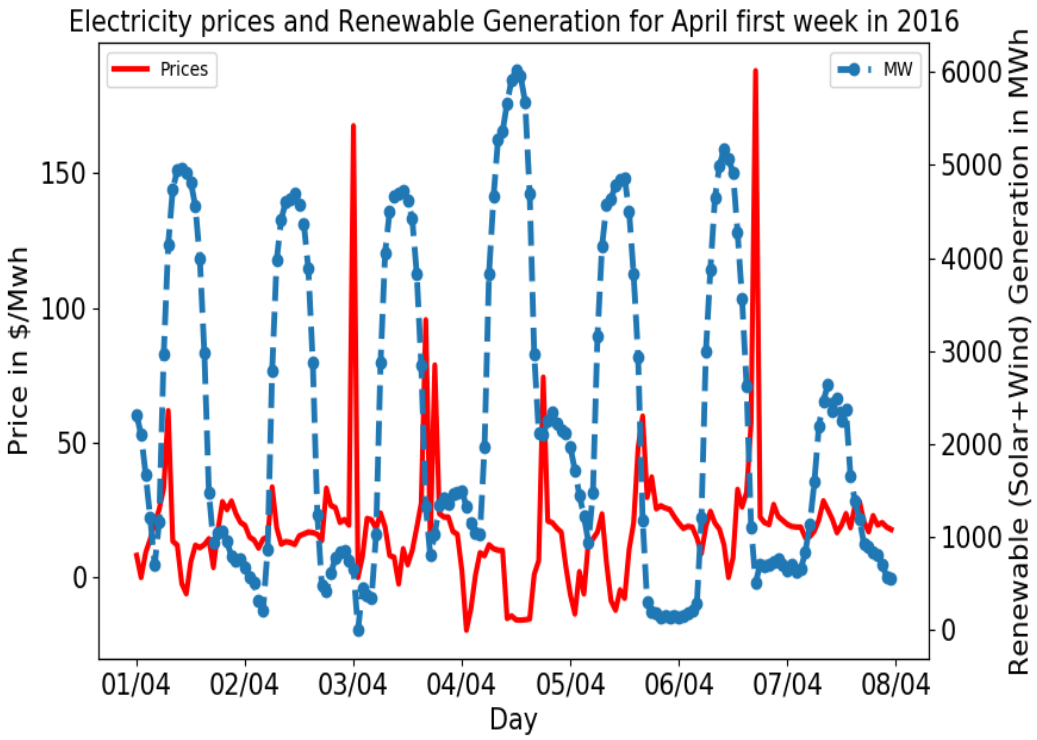

Figure 4.4.1 Variation of hourly LMPs and renewable (Solar+Wind) generation for April first week in 2016

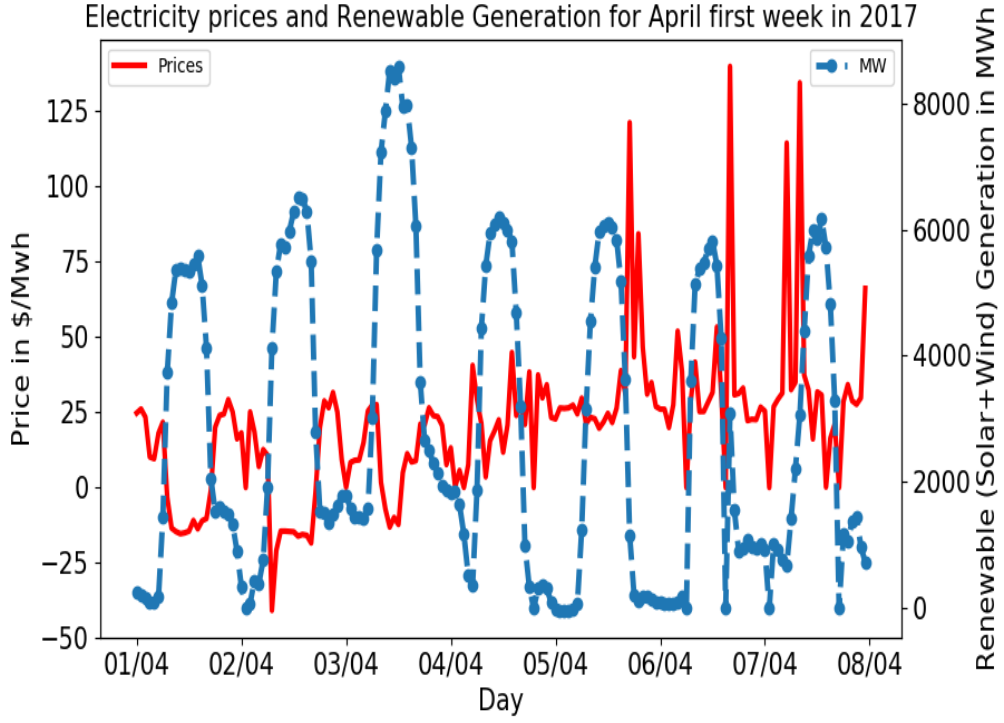

Figure 4.4.2 Variation of hourly LMPs and renewable (Solar+Wind) generation for April first week in 2017

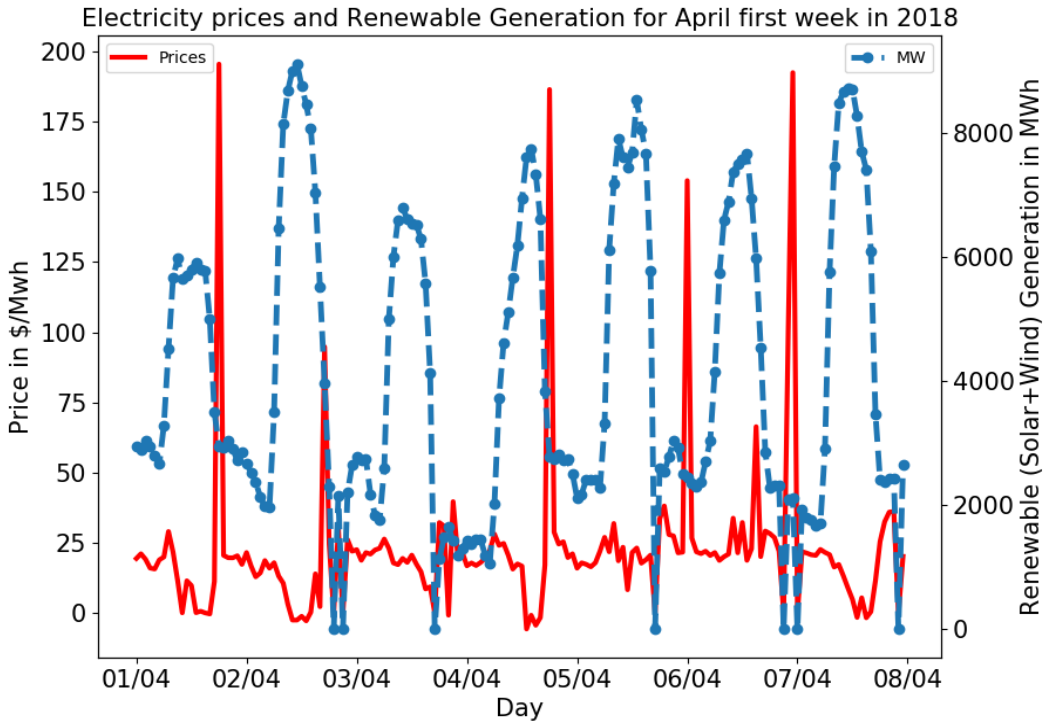

Figure 4.4.3 Variation of hourly LMPs and renewable generation in MWh for the first week of April in 2018

The plots in Fig 4.4.4 show the negative price share for every month of the year from 2015-2018. Negative pricing instances in 2016 and 2017 show significant increase of around 10% for March and April in comparison to 2015. March and April are also the two months when renewable generation due to PV is maximum in comparison to the rest of the year. A significant decrease is seen in the negative pricing share for 2018 as CAISO has done well in reducing the instances by measures such as load shifting, storage and trading on the Energy Imbalance Market (EIM).

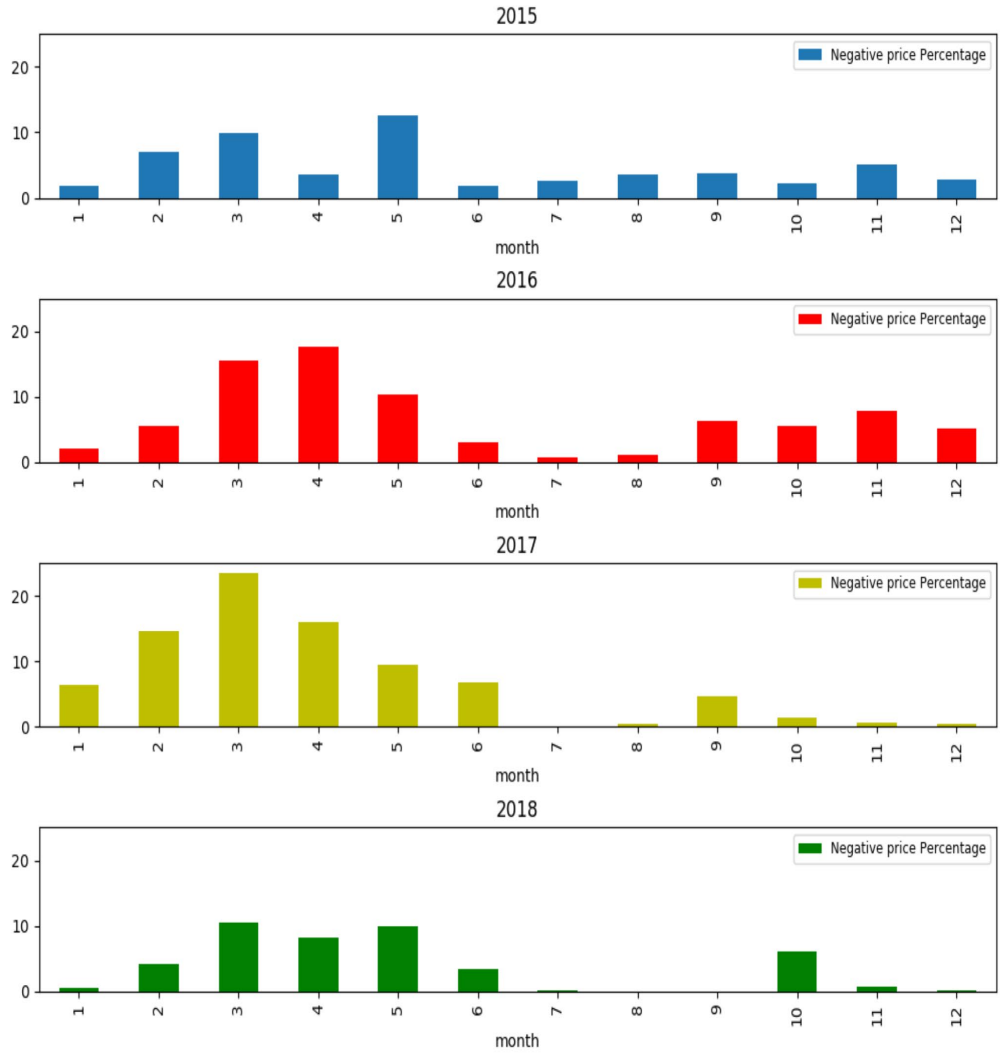

Figure 4.4.4 Plot showing the negative price share for every month of a year from 2015-2018

Fig 4.4.5 shows the negative price share for every hour of the day from 2015-2018. The negative price share for 2016 and 2017 is found to be maximum around noon when usually PV generation is maximum. 2018 reports a significant decrease in negative price share as reported earlier which might be due to the counter-measures implemented by CAISO.

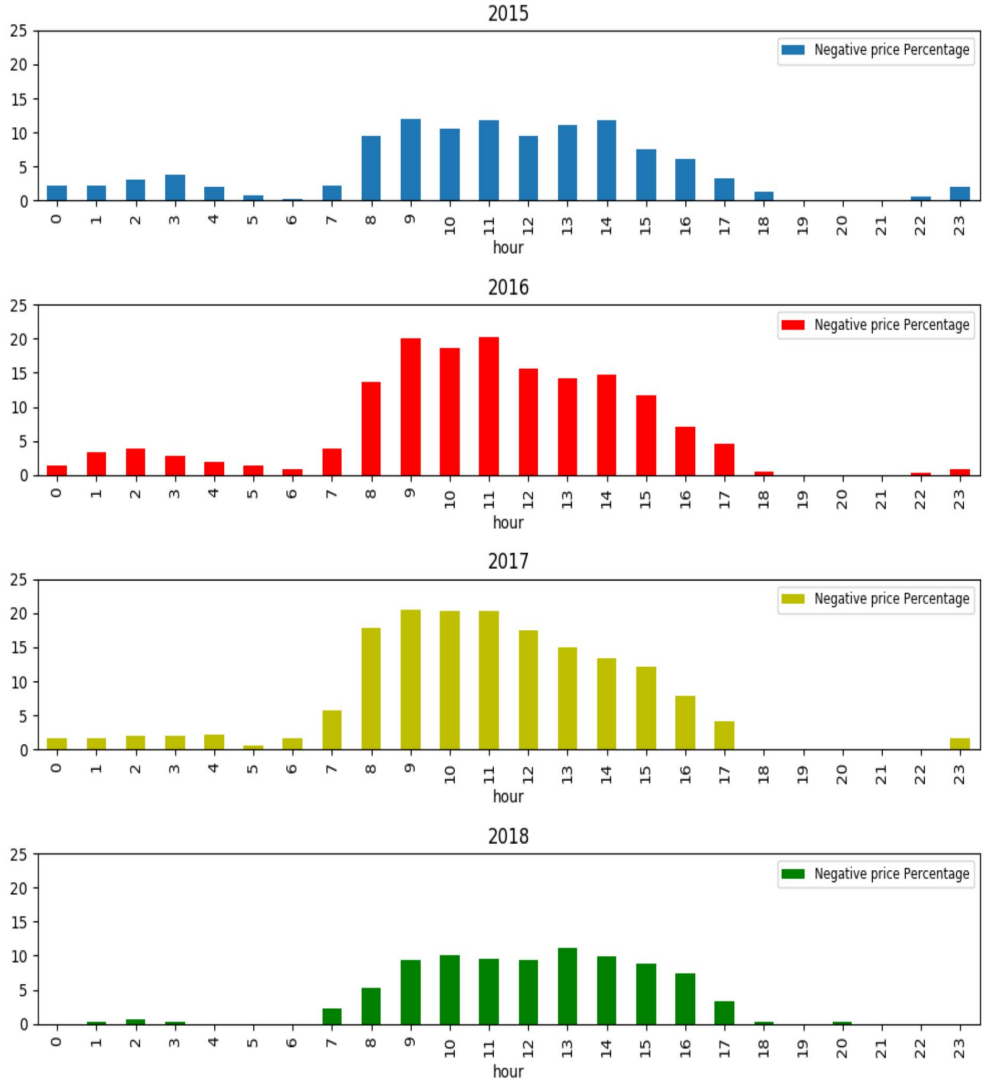

Figure 4.4.5 Plot showing negative price share for every hour of the day from 2015-2018

Figure 4.4.6 tries to establish a correlation between LMPs and renewable generation from 2015-2018. From the plot, a slight negative correlation is seen between the two as the band of prices reduces with the increasing renewable generation. The band of prices gets concentrated around zero as the renewable generation by solar and wind increases. Another observation is that the volatility of the prices decreases as the combined solar and wind generation goes beyond 5 gigawatts.

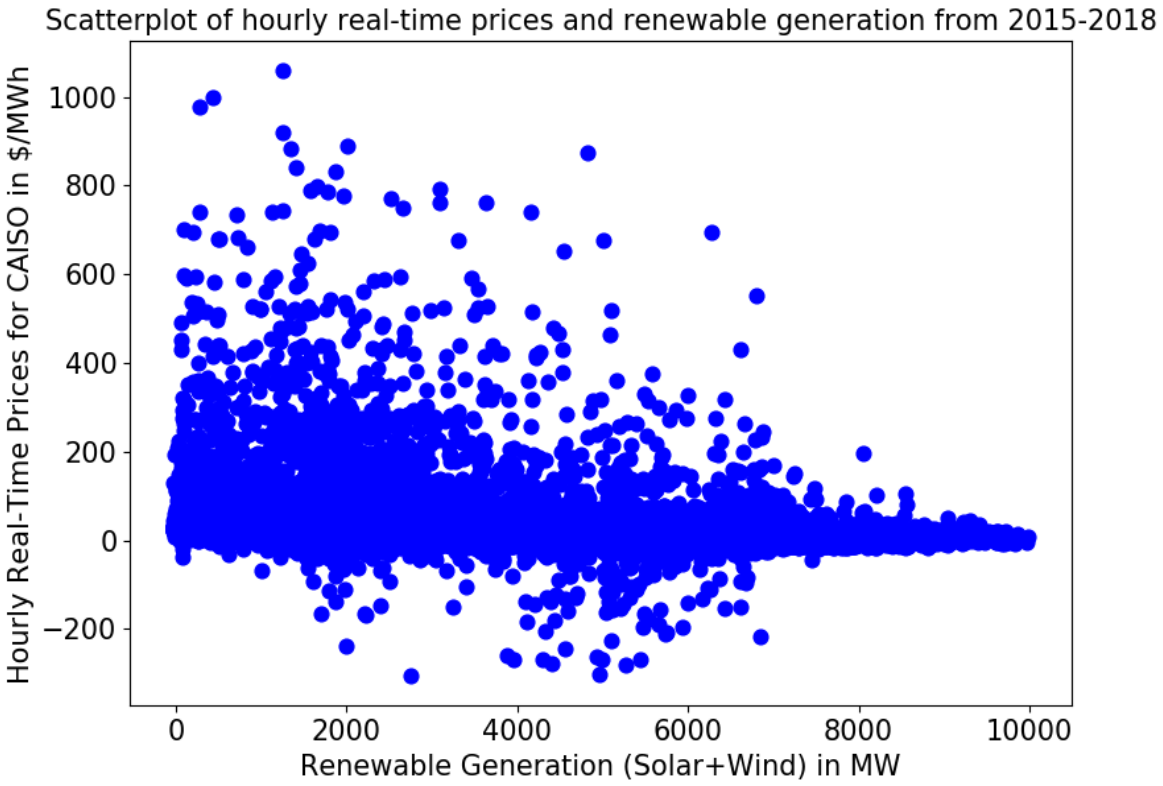

Figure 4.4.6 Scatterplot showing the variation of hourly LMPs with increasing renewable generation in MWh from 2015-2018

CHAPTER 5

Machine Learning (ML) forecasting algorithms for trading zone prices

5,1 Detailed Description of Machine Learning Models

Machine learning (ML) makes it easier to understand the data accurately and in less time. As defined in 90s by Arthur Samuel, it is a "field of study that gives computer the ability to learn without being explicitly programmed" [41]. Multiple ML algorithms are implemented to forecast electricity prices with a goal to evaluate the effect of every feature on the price forecast and compare the percentage of error in the forecast for each model. The features are load, renewable generation, fuel prices and aggregated generation outages. Supervised learning algorithms are used since they provide more accurate predictions over the clustering algorithms. Supervised learning is when you have the given data set and the result, all you have to do is concoct a relationship between the input (given data set) and the output (the given result). Since the output is price in $/MWh which is a continuous variable, regression-based learning approaches are used in forecasting. Regression involves predicting real-valued numbers by determining the statistical relationship between input and output quantities. Ensemble- based learning methods have been implemented in this thesis for forecasting since they provide much accurate results compared to linear models. Ensemble models combine outputs from several models by using techniques such as averaging, bagging etc. to give a better prediction of the output variable. Fig 5.1.1 shows the ensemble learning process.

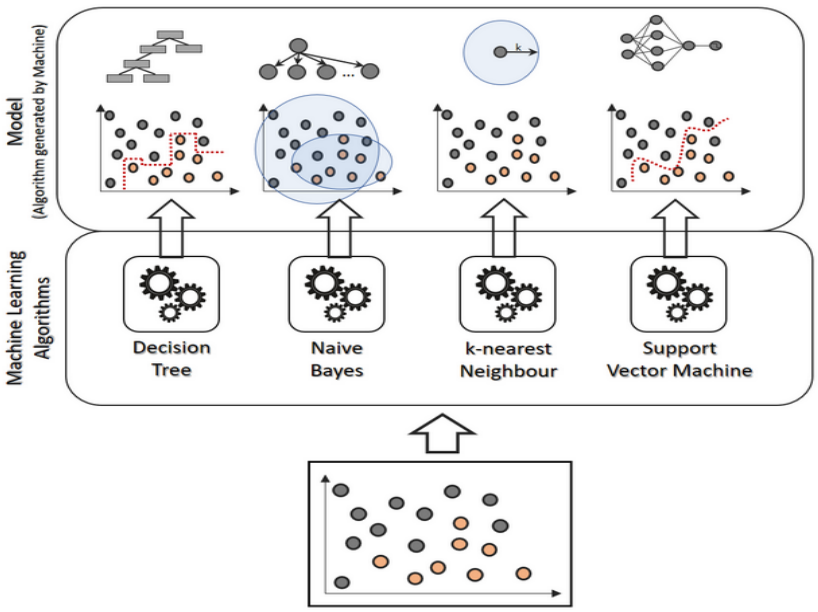

Figure 5.1.1 Ensemble learning process [22]

The forecasting algorithms utilize eight features to model the problem. The features are as follows:

- Load in MW
- Fuel Prices in $/mmBtu
- Generation Outages in MW
- Renewable (Solar+Wind) generation in MW
- Year
- Month of the year
- Day of the month
- Hour of the day

Three of these features (Day, Month and Year) were found to have little or no significant impact on the price forecast as compared to the other features. Hence, they were not accounted for while building the prediction model and the model is built on five features which are:

- Load in MW
- Fuel Prices in $/mmBtu
- Generation Outages in MW
- Renewable (Solar+Wind) generation in MW
- Hour of the day

Four ML algorithms have been implemented and the accuracy of every algorithm is assessed along with the computational time. The scikit learn library in Python has been utilized to implement the first three algorithms while the fourth has been implemented by using the keras and Tensorflow library developed by Google. The algorithms are:

- Extra Trees Regression
- Gradient Boost Regression
- XGBoost Regression
- Long short term memory networks (LSTM) using Tensorflow

Extra Trees or Extremely Randomized Trees is a supervised learning approach with the goal to optimize Random Forest further in terms of reducing the variance in the output. It differs from random forest learning method in terms of node split where the cut points are chosen randomly instead of optimally finding the cut-point based on features. Also, it uses the entire learning sample rather than using bootstrap copies of the same [23]. Random Forest is a supervised learning algorithm which creates multiple decision trees and merges the results to provide an accurate prediction. Fig 5.1.2 shows the random forest tree structure. It involves splitting the dataset into several subsamples and then building a decision tree for regression on every sample. The sampling of the data is done with

replacement till an optimal cut-point is found. The final or the predicted value is the mean value of all output variables in the leaf node. [25]

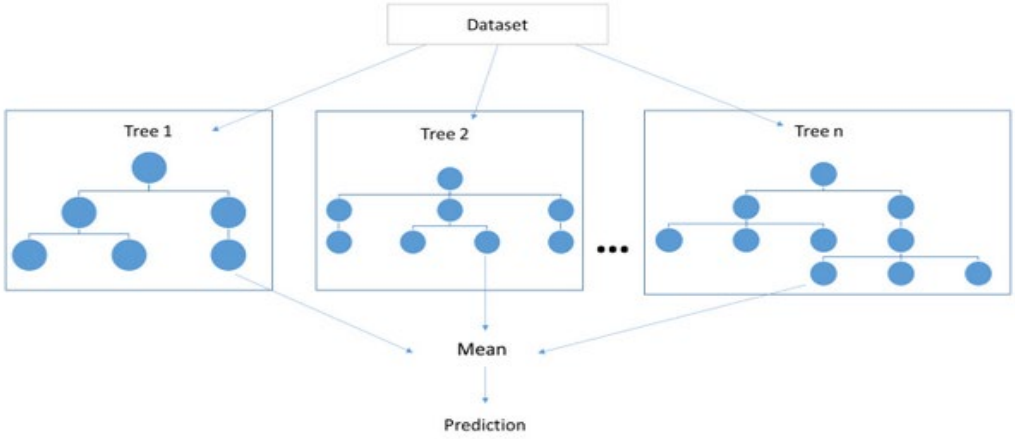

Figure 5.1.2 Random Forest Tree structure [24]

Gradient Boost (GB) is one more example of an optimized RF algorithm. GB works on creating a sequence of trees with each successive tree built on the residuals of the prediction of the previous tree. [26] Boosting is a way to transform weak learners into strong learners. The learners mentioned previously are basically decision trees with assigned weights. Fig 5.1.3 shows a boosting trees structure. Gradient Boosting involves a gradual and sequential approach in which the weak learners are identified using gradient descent in the loss function. The definition of a loss function is measuring how good are the model coefficients at fitting the data. The algorithm tries to minimize the loss function by adding weak learners at each step to build a strong learner. Hence, the magnitude of the loss function is reduced at every step leading to less error between the actual and the predicted value. [27].

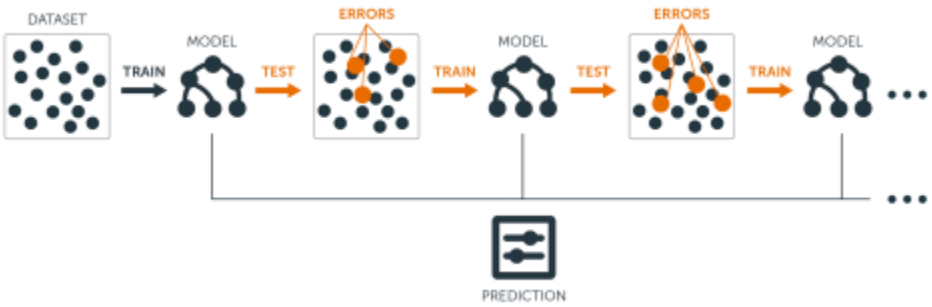

Figure 5.1.3 – Boosting trees algorithm structure [28]

XGBoost or Extreme Gradient Boosting is one of the best ensemble-based learning methods used in data science. XGBoost is similar to Gradient Boosting but has certain features which makes it more efficient. It uses Newton Boosting which implements Newton-Raphson method of approximations to provide a faster solution than gradient descent. Also, it adds a randomization parameter to reduce the correlation between individual trees leading to a better model. Also, XGBoost requires non-continuous memory access and ensures the maximum utilization of available disk space leading to faster and efficient computing. [29]

Long short term memory (LSTM) networks are examples of recurrent neural networks (RNNs) which are used to identify patterns in a dataset. RNNs can be visualized as duplications of the same network, with each copy passing its output to the next. Fig 5.1.4 shows a basic structure of a recurrent neural network.

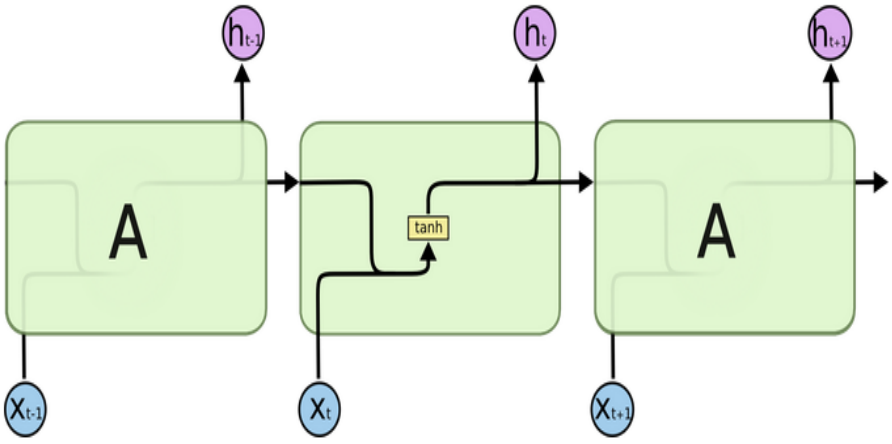

Figure 5.1.4 Structure of a repeating module in RNN containing a single layer [30]

LSTMs are basically RNNs which can learn long-term dependencies. They have the ability to preserve the error which could be fed back to the previous layers. In this way, RNNs could learn over a longer span of time and give better outputs since the error is maintained as a constant value. Fig 5.1.5 shows the structure for a LSTM network.

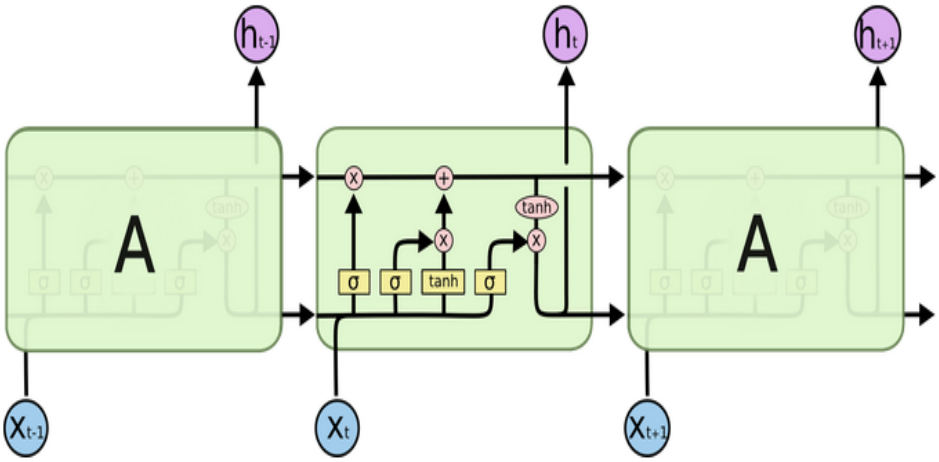

Figure 5.1.5 Structure of a repeating module in LSTM containing four layers [30]

The horizontal lines running through the model are called as cell states through which the information flows. The cell state can be loaded with information through structures called as gates. These gates are a combination of a sigmoid neural net layer and a pointwise multiplier. The output from the sigmoid layer is binary where zero means it acts as blocked while one opens the gate. The following steps are implemented in LSTM:

1. LSTM decides through a sigmoid layer called as "forget gate layer" which data should be passed to the cell state. The layer outputs 0 and 1 for every value within the cell.
2. The information to be stored in the cell is decided through a sigmoid and a hyperbolic tan (tanh) layer. The sigmoid layer decides the values to be stored while the tanh layer creates a candidate vector to add to the state.
3. The cell state is updated using Step.2

4. Output is determined based on the cell state. It is determined by applying a sigmoid layer through the cell state to determine what to output. Then, the cell state is normalized by a tanh function to limit the values between (-1,1) and is multiplied by the sigmoid layer output to get the final prediction.

5.2 FORECASTING PROCEDURE

Before applying the ML algorithms on the dataset, the dataset is split into 'training' and 'test' data. Supervised learning methods are used to train the model on the 'training' dataset. This helps the model to build relationship between the input variables or features and the target variable. The model is evaluated on the 'test' dataset which is kept aside and unknown to the model. The models built by the algorithms specified in Section 6.1 need to be tuned to achieve better accuracy. However, the test data cannot be used to check the prediction accuracy since in the real world, the data would be completely new Also, tuning the model using the 'test' data would flaw the generalization measure and lead to inaccurate results. Hence, the model is tuned using a chunk of the 'training' data called as 'validation' data. Every ML model has a set of 'hyperparameters' which need to be set manually and adjusted. These set of values describe the higher-level properties of the model such as the learning rate, complexity etc. They cannot be learned from the data and need to be pre-defined before starting the training process. The tuning process of the model on the 'validation' data has the goal to find the optimal 'hyperparameters' which would result in the best degree of accuracy.

With the goal to tune the model for best accuracy, a method called as cross-validation (CV) is used to select the 'validation' dataset. Cross-validation tests the model using the training data to make sure it does not overfit or underfit resulting in a greater degree of error. The

ML algorithms described in Section 5.1 are tuned using K-Fold CV which involves splitting the data into folds and using each fold as a testing set at some point [31]. The method splits the dataset into K parts or folds. Consider K=3 as an example. For the first iteration, the first fold would be used as 'test' data while the rest are used as 'training' data. The second iteration would involve using the second fold as 'test' data and the rest serving as 'training' data. The process would repeat until every fold is utilized as 'test' data. Fig 5.2.1 shows the K- Fold CV algorithm for K=3.

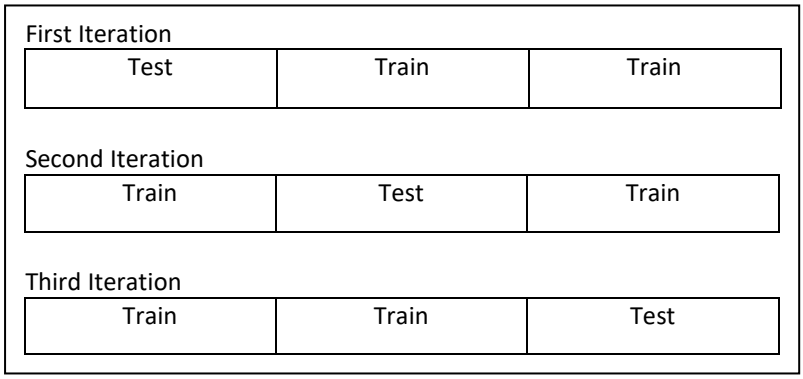

Figure 5.2.1 3-Fold Cross Validation

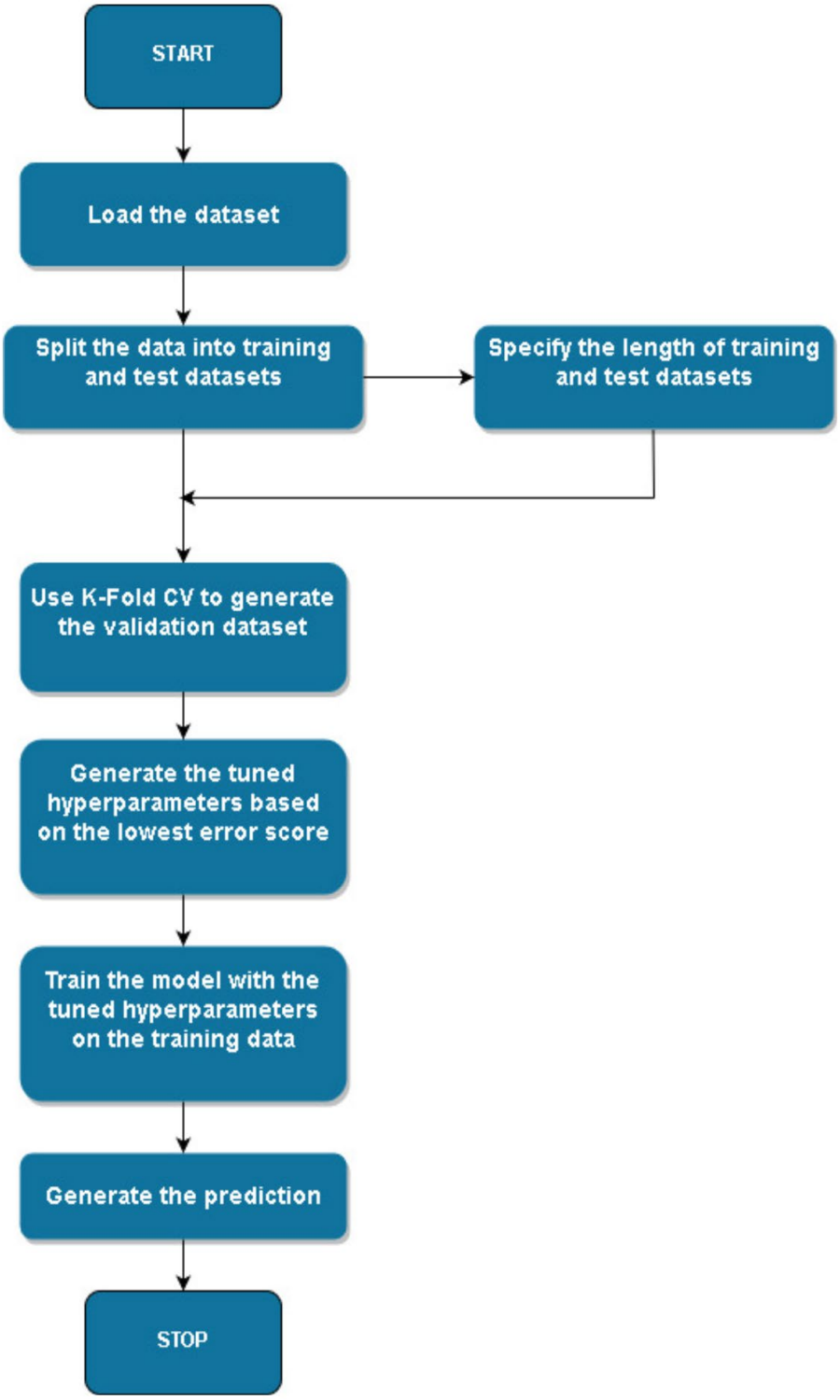

Figure 5.2.2 Flowchart describing the prediction process by machine learning (ML)

CHAPTER 6

RESULTS

6.1 Forecasting Results

The ML algorithms discussed in Chapter 5 are applied on the dataset. The goal is to compare the results of the three models and analyse how much renewable generation impacts the price forecast. A common metric used to verify the accuracy of every ML model is root mean squared error (RMSE). RMSE is widely used in forecasting and is defined as the standard deviation of the errors in prediction. The prediction errors are often called as residuals and they indicate how spread out the points are from the regression curve [35]. RMSE is defined as follows:

$$RMSE = \sqrt{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2 / N}$$

Here,

Predicted – Predicted values by the model

Actual - Actual values (Test data)

N - Number of actual values

For this model, we use relative RMSE (rRMSE) as an accuracy metric. rRMSE is expressed as a percentage and the better the value, the more accurate is the model. rRMSE is defined as follows:

$$rRMSE = \sqrt{(1/N) * \sum_{i=1}^{N} (Predicted_i - Actual_i)^2 / (\sum_{i=1}^{N} (Predicted_i)^2)}$$

Here,

$Predicted_i$ – Predicted value by the model

$Actual_i$ - Actual value

N - Number of actual values

Three different days are chosen in 2018 to plot the predicted prices versus the actual prices. Fig 6.1.1 – 6.1.3 shows the forecast and actual values for January 21, 2018 for all four ML algorithms: Extra Trees Regression, Gradient Boosting Regression and LSTMs using tensorflow.

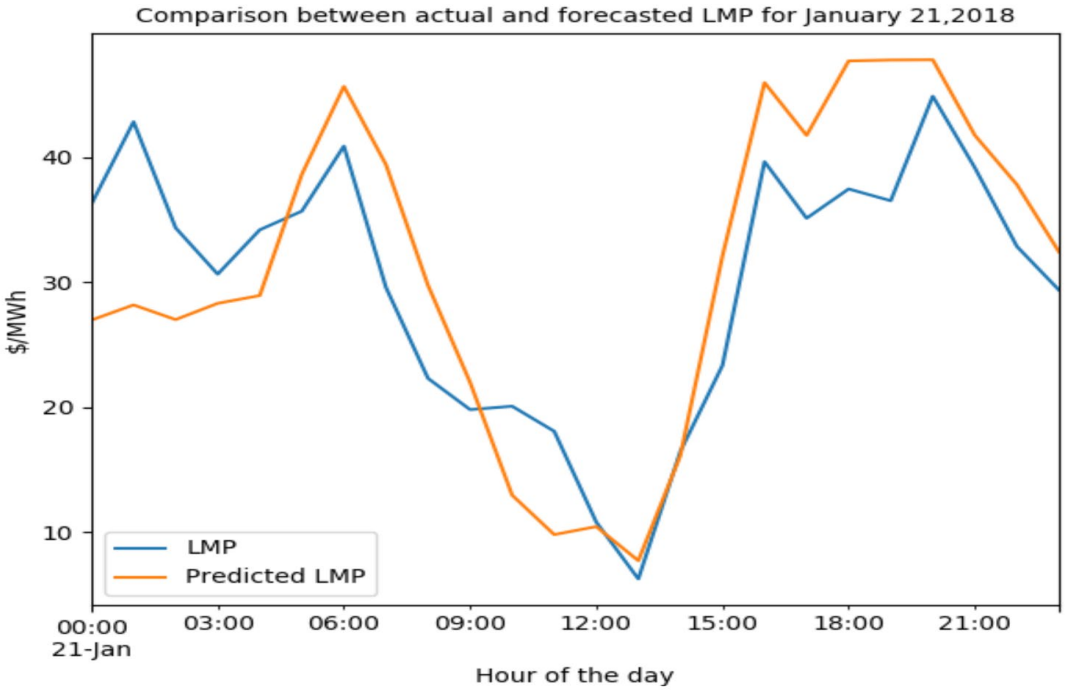

Figure 6.1.1 Plot showing the actual and forecasted LMPs for January 21,2018 using Extra Trees Regression

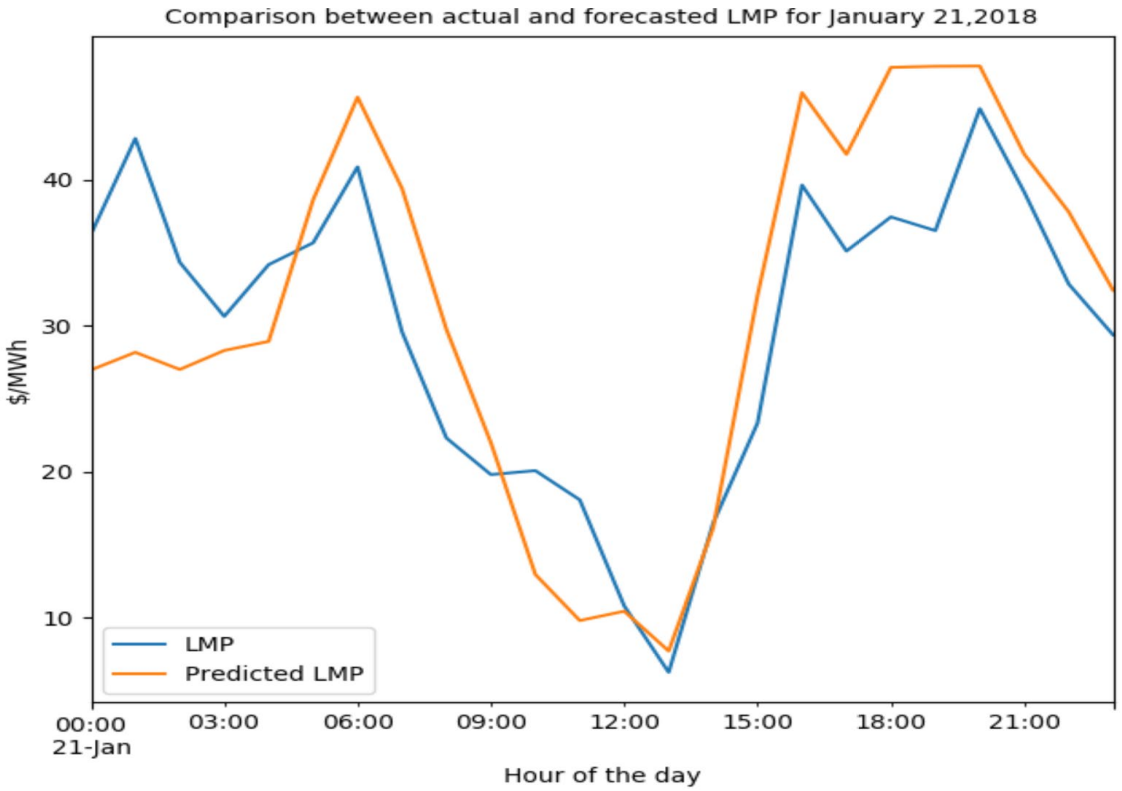

Figure 6.1.2 Plot showing the actual and forecasted LMPs for January 21,2018 using Gradient Boost Regression

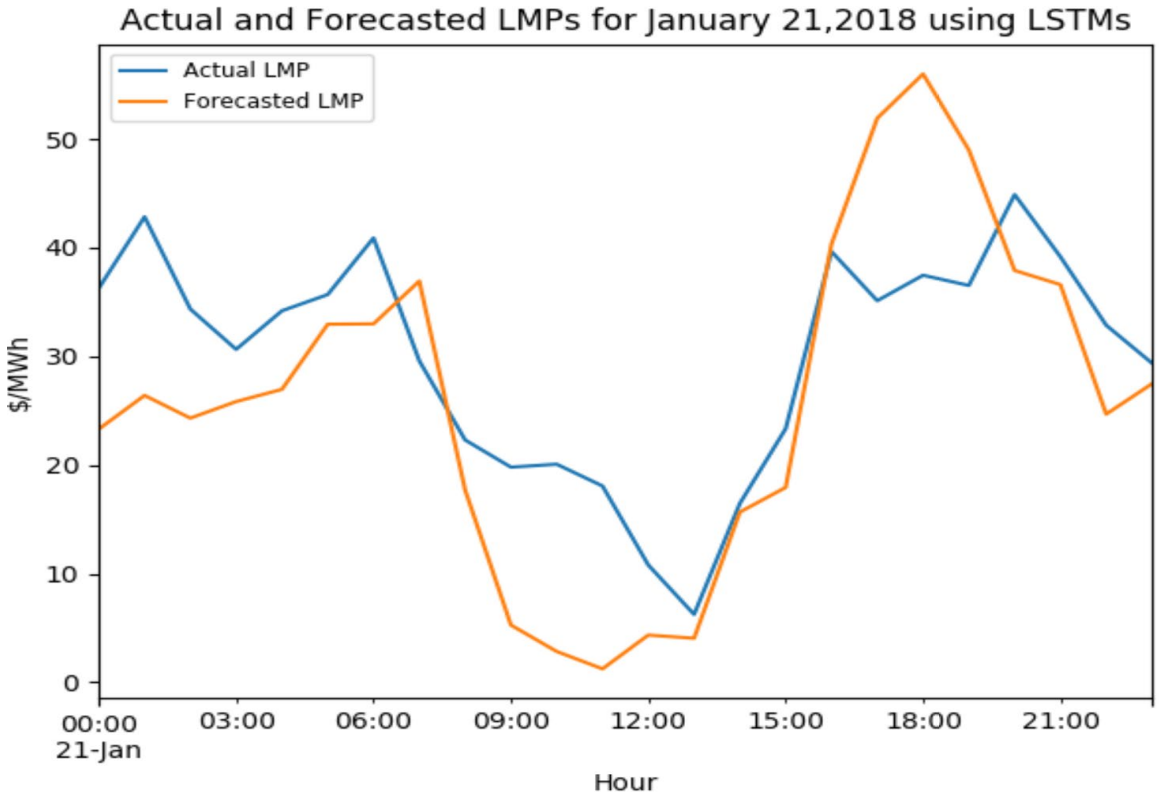

Figure 6.1.3 Plot showing the actual and forecasted LMPs for January 21,2018 using Long short term memory networks (LSTMs)

To ensure that the prediction covers a wide range of dates, the ML algorithms are tested for two more days which are randomly picked. Fig 6.1.4 – 6.1.6 shows the forecast and actual values for June 13, 2018 for all four ML algorithms: Extra Trees Regression, Gradient Boosting Regression and LSTMs using tensorflow.

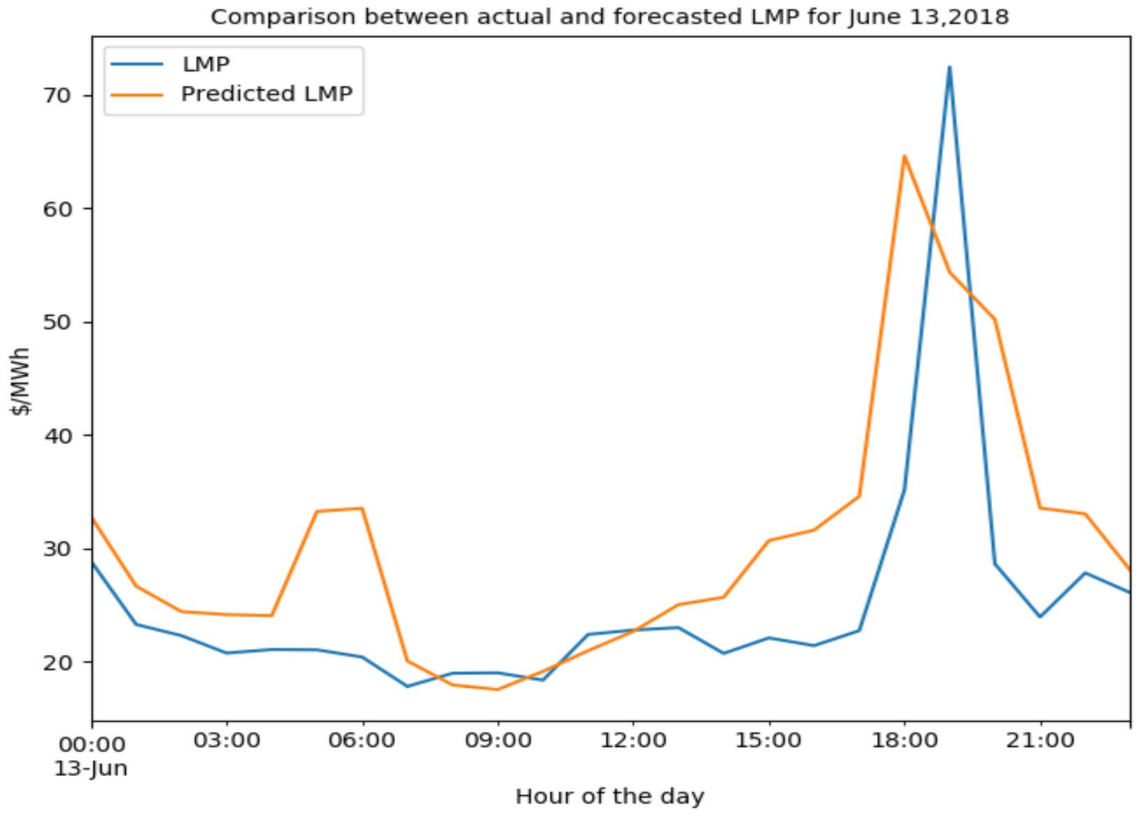

Figure 6.1.4 Plot showing the actual and forecasted LMPs for June 13,,2018 using Extra Trees Regression

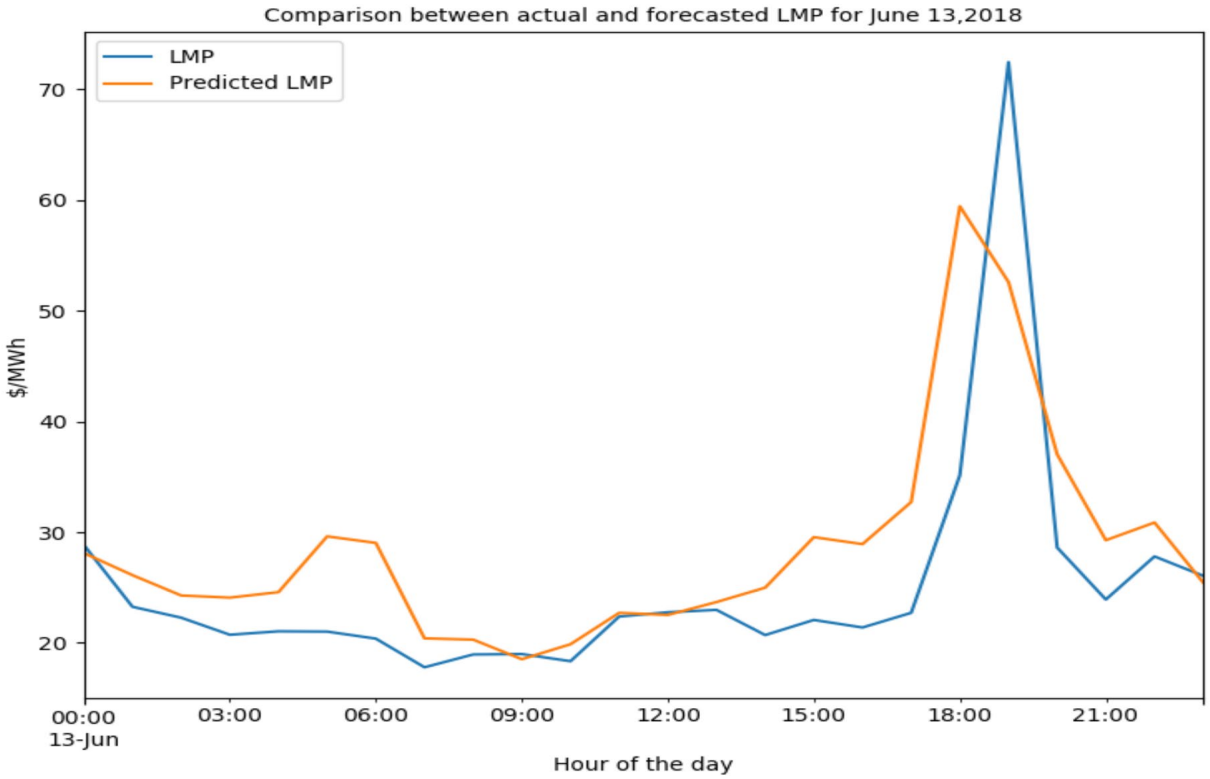

Figure 6.1.5 Plot showing the actual and forecasted LMPs for June 13,2018 using Gradient Boost Regression

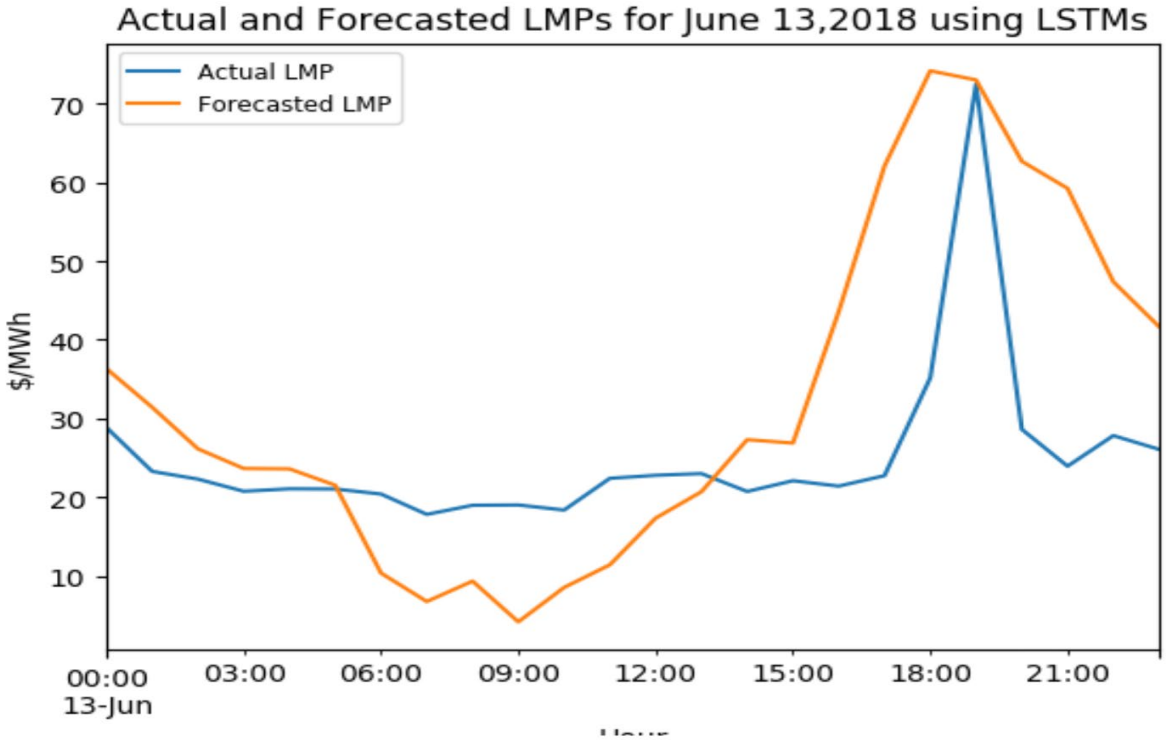

Figure 6.1.6 Plot showing the actual and forecasted LMPs for June 13,2018 using Long short term memory networks (LSTMs)

Fig 6.1.7 – 6.1.9 shows the forecast and actual values for June 13, 2018 for all four ML algorithms: Extra Trees Regression, Gradient Boosting Regression and LSTMs using tensorflow.

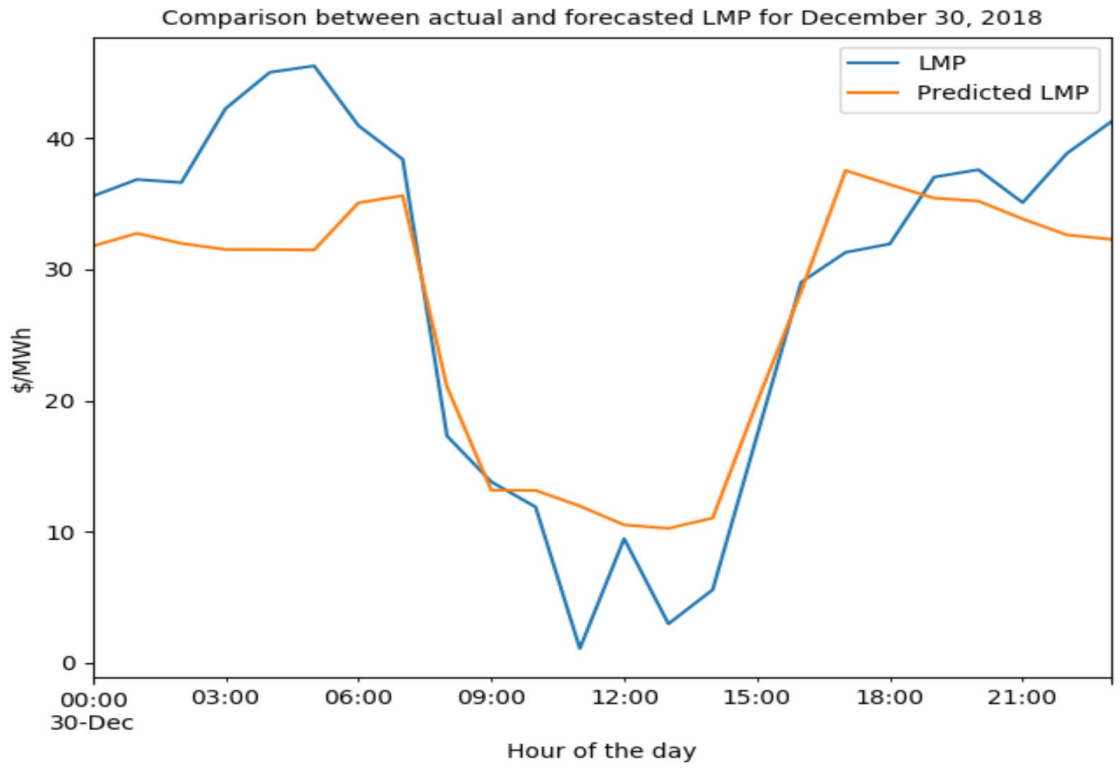

Figure 6.1.7 Plot showing the actual and forecasted LMPs for December 30,2018 using Extra Trees Regression

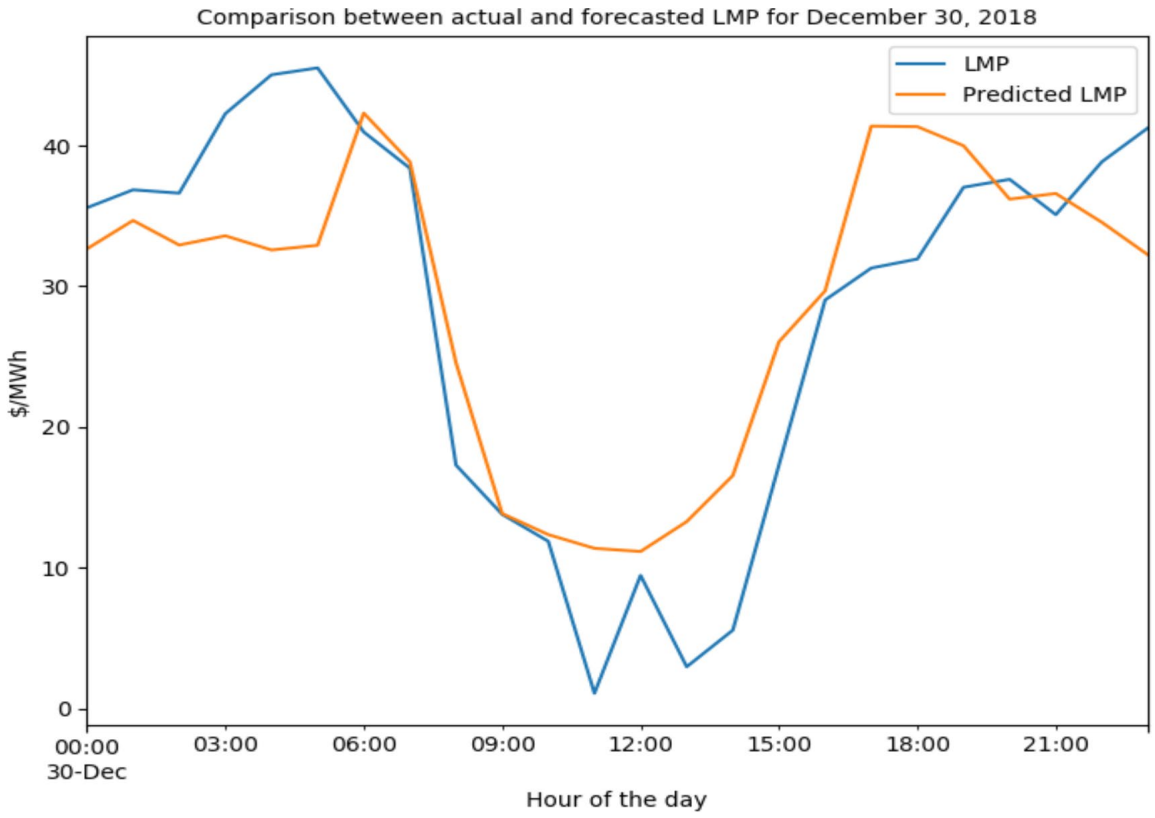

Figure 6.1.8 Plot showing the actual and forecasted LMPs for December 30,2018 using Gradient Boosting Regression

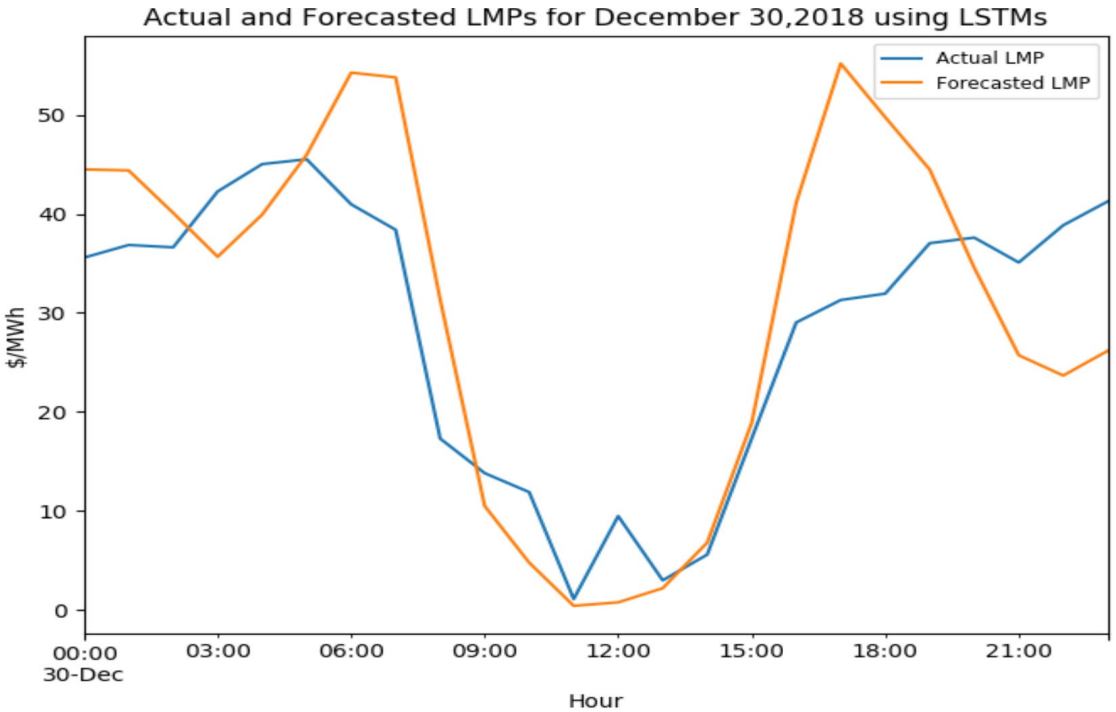

Figure 6.1.9 Plot showing the actual and forecasted LMPs for December 30,2018 using Long short term memory networks (LSTMs)

The relative RMSE (rRMSE) of all machine learning (ML) forecasts for all the three days has been tabulated into Table 6.1.1. Two rRMSEs values are calculated for every ML algorithm which are for the 'training' and the 'test' data. rRMSE for 'training' data indicates whether the model is overfitted or underfitted. Overfitting occurs when a model learns too much from the 'training' data. This leads to a negative impact on the model's performance when it encounters new data. Overfitting usually occurs in non-linear models which are more flexible [36]. An underfitted model is unable to model the 'training' data and cannot make a good prediction for the new data. Underfitting denotes poor performance of the ML forecasts. An ideal ML model tends to find a good value between overfitting and underfitting. This is done by the hyperparameter tuning process discussed in Chapter 5.

ML algorithm	January 21		June 13		December 30	
Data	Training rRMSE (%)	Test rRMSE (%)	Training rRMSE (%)	Test rRMSE (%)	Training rRMSE (%)	Test rRMSE (%)
Extra Trees	10.22	24.34	12.35	30.21	8.67	28.56
Gradient Boost	8.91	21.62	11.22	29.12	7.41	28.20
Long short term memory networks (LSTMs)	12.41	27.49	10.56	34.67	11.39	33.44

Table 6.1.1 Tabular form showing the relative RMSE (rRMSE) of the predictions from ML algorithms

The table shows that the model fits reasonably well for all three days after tuning the hyperparameters. However, the testing rRMSE is high for all three cases. This might be occurring since the thesis does not involve solving the economic dispatch problem that is

used for LMP formulation. The goal is to analyse the impact of every feature included in the dataset on the prices. Also, other factors such as binding constraints on the system, transmission outages aren't included in the dataset since the data for the same was not readily available on the OASIS portal for CAISO. These factors may cause a significant impact of LMP and might result in a better degree of prediction of prices.

6.2 Impact of Every Feature on the Locational Marginal Price (LMP)

Every feature in the 'training' dataset for a machine learning (ML) algorithm has an impact on the target variable or the predicted value. Knowing the weights of every feature in determining the target variable makes the model interpretable. These weights are called as feature importances. Feature importances help the user know which features are most important in predicting the output. They also help in detecting features which have very little or negligible impact on the forecast. The user can go ahead and discard those features if not important to get a better accurate prediction. Tree-based algorithms usually have a number of decision trees which have multiple nodes. Every node is an if/else condition on a single feature which is designed to split the data into two. In this way, similar samples end up in same sets. This optimal condition is chosen based on a measure which is called as variance for regression-based trees. Thus, during the training process, it can be computed how much each feature decreases the variance in a tree. For structures such as random forest, the decrease in variance from each feature can be averaged and features can be ranked according to this measure. [37]

Feature importances can be calculated using a built-in function in the Python scikit learning library for forest-based structures. The output is an array of weights for every

feature present in the 'training' dataset. Fig 6.2.1 to 6.2.6 show a bar plot of how much each feature in the dataset affects the price prediction.

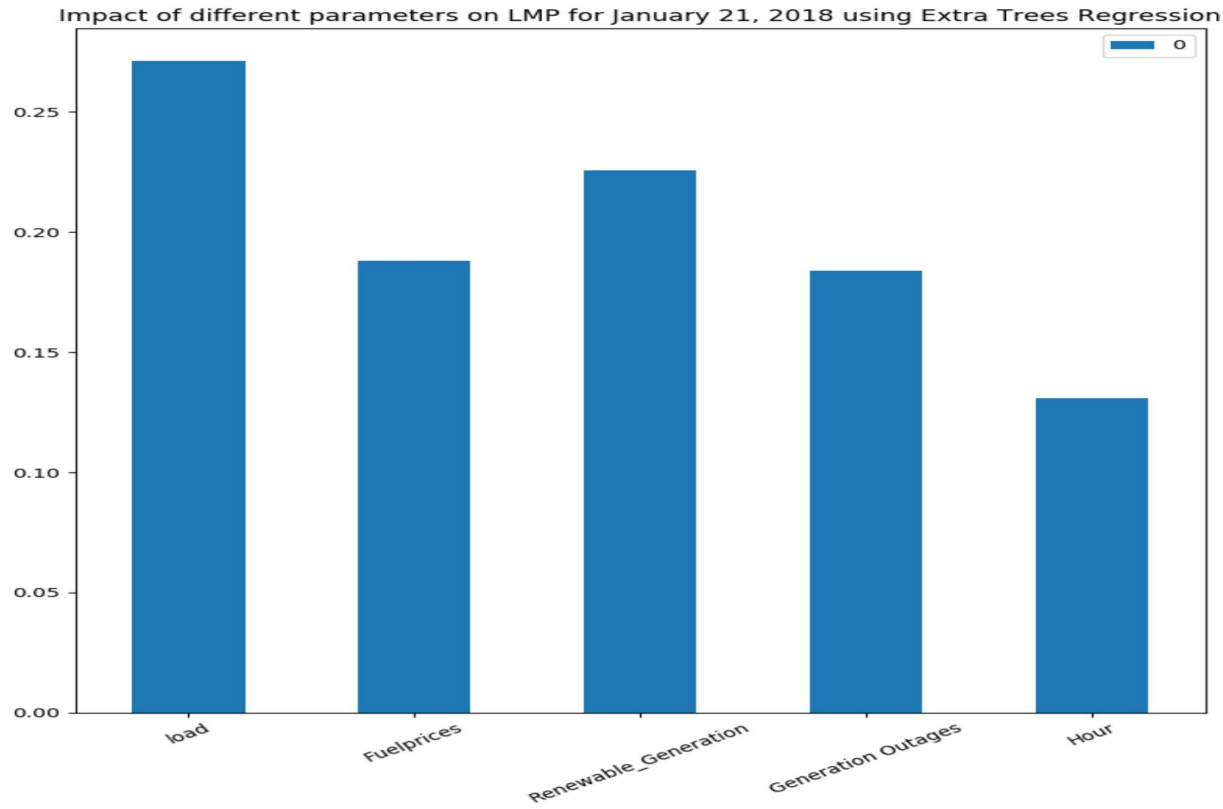

Figure 6.2.1 Plot showing the impact of features in the dataset on the price forecast for January 21, 2018 using Extra Trees Regression

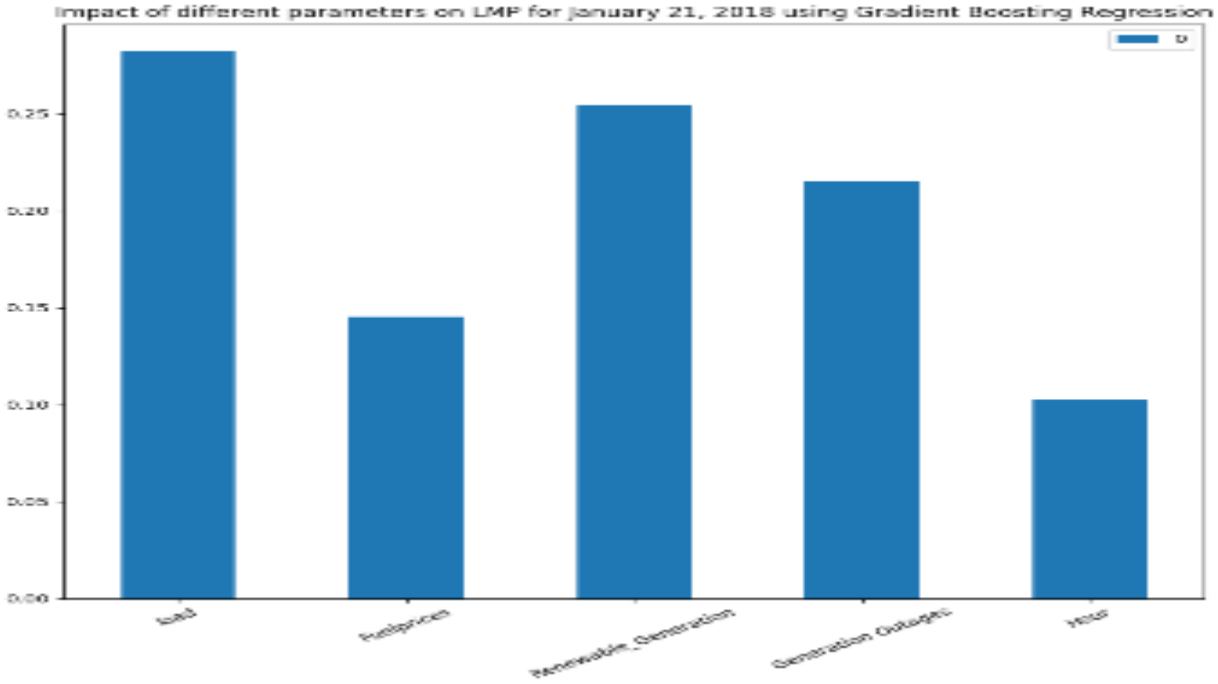

Figure 6.2.2 Plot showing the impact of features in the dataset on the price forecast for January 21, 2018 using Gradient Boosting Regression

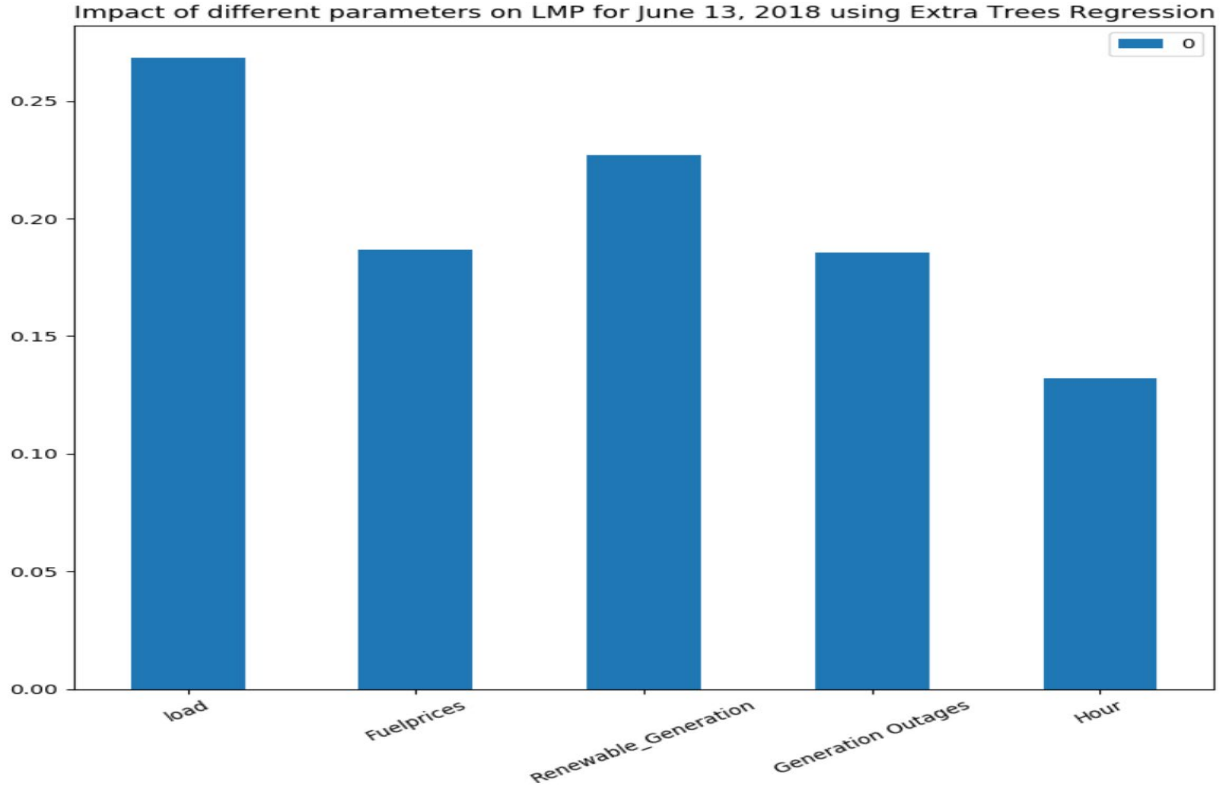

Figure 6.2.3 Plot showing the impact of features in the dataset on the price forecast for June 13, 2018 using Extra Trees Regression

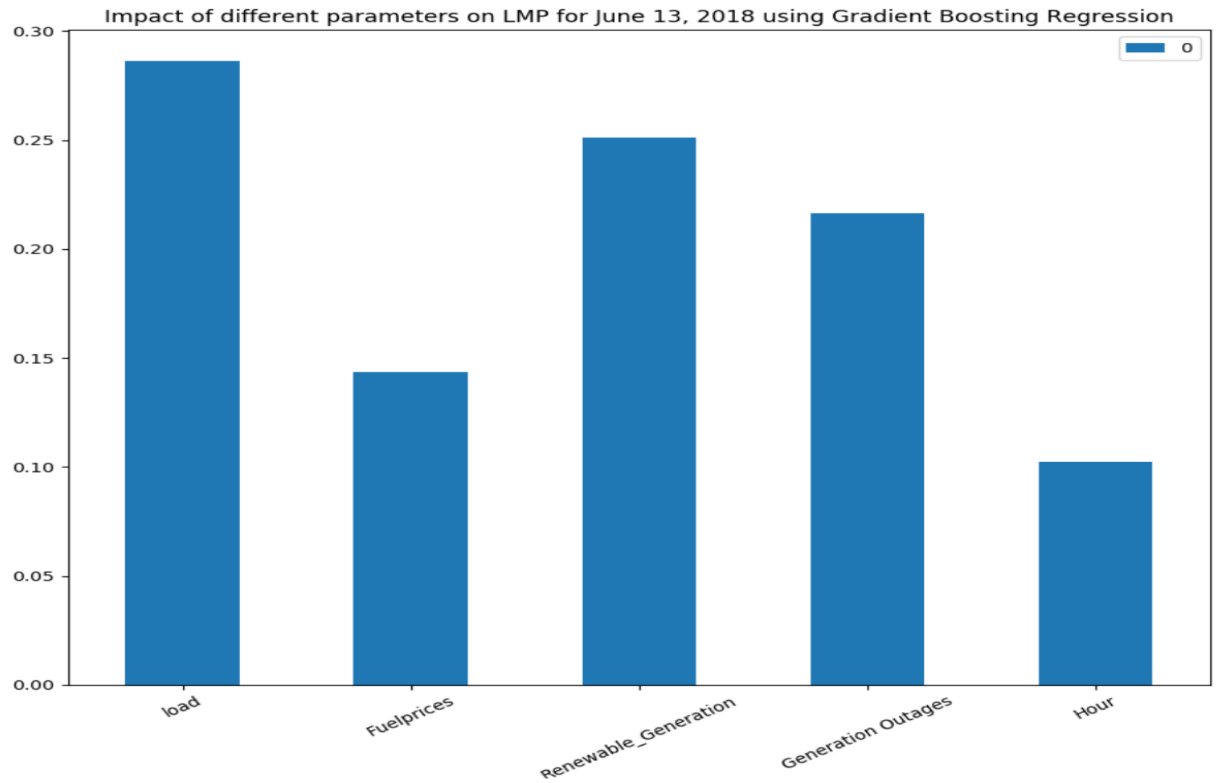

Figure 6.2.4 Plot showing the impact of features in the dataset on the price forecast for June 13, 2018 using Gradient Boosting Regression

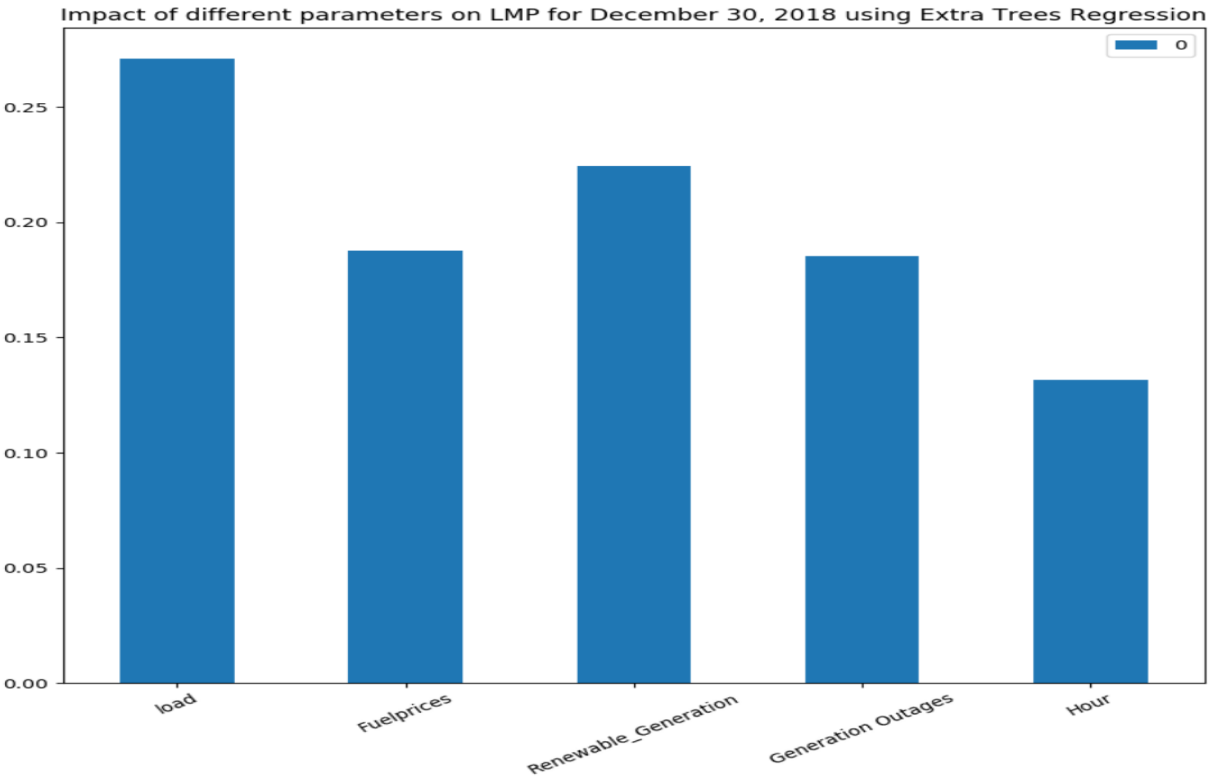

Figure 6.2.5 Plot showing the impact of features in the dataset on the price forecast for December 30, 2018 using Extra Trees Regression

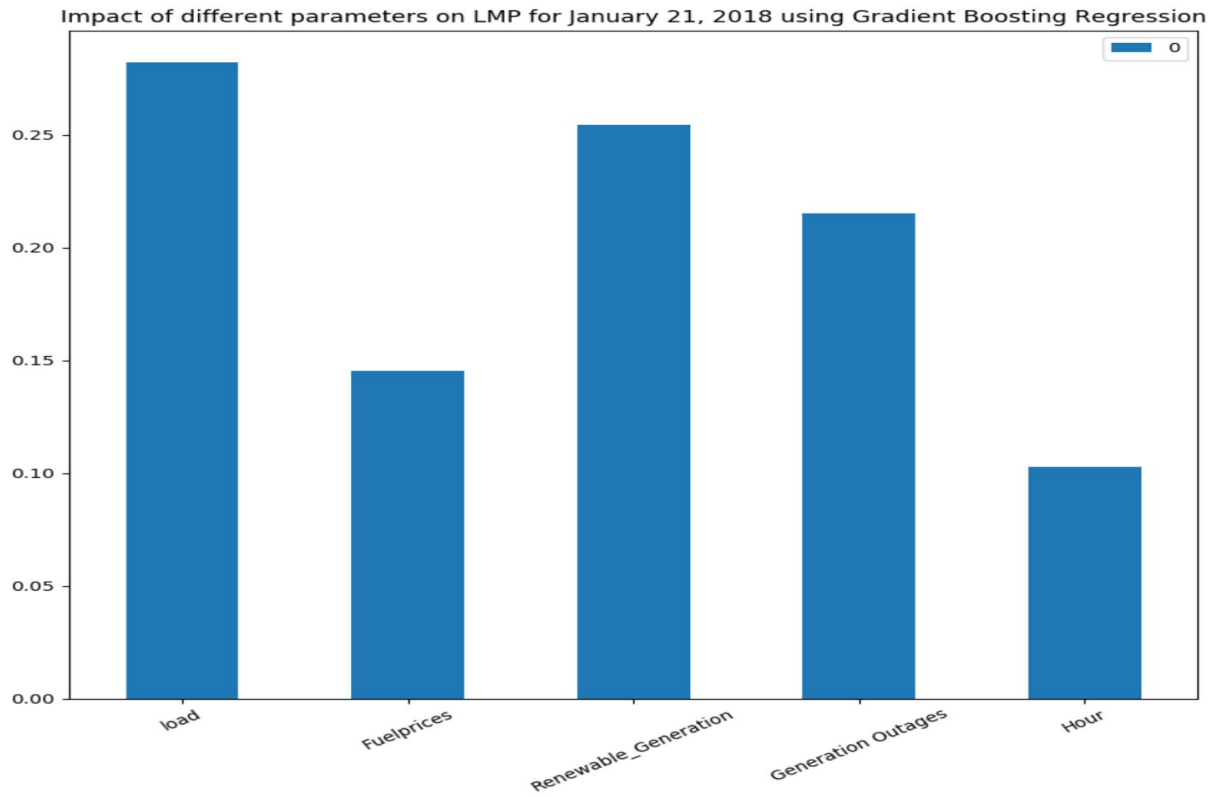

Figure 6.2.6 Plot showing the impact of features in the dataset on the price forecast for December 30, 2018 using Gradient Boosting Regression

From the plots in Fig 6.2.1 – 6.2.6, it is seen that the feature importances vary for every prediction. This is because the training data varies for each one of these predictions. Also, one other reason would be that the features are randomly permuted at each split in a tree. The best-found split might vary even if the training data remains the same.

Fig 6.2.1- 6.2.6 clearly show that the highest impact on the LMP is of the load or demand for that hour. It is seen that for most of the above cases, renewable generation has also had a significant impact on the price forecast. The impact is even greater than the impact caused by aggregated generation outages which usually impact the LMP by increasing the congestion component.

This shows that increasing renewable generation in California can significantly impact the real-time locational marginal prices (LMPs) for every node within California ISO. There might be hours with low demand during the day. If in such cases, renewable generation is at

its peak, for example, solar during the middle of the day, prices might go negative even for several hours. Such negative prices would be difficult to forecast in case they do not follow a consistent pattern.

Chapter 7

CONCLUSION

7.1 Conclusion

The goal of this research was to concentrate on how the load, fuel prices, renewable generation and generation outages affect the forecast of locational marginal prices (LMPs) for California ISO. This goal was achieved by comparing several machine learning (ML) methods for price forecasting and analysing their results. The research was aimed specifically to answer the question of "How would increasing renewable generation for California impact the day-ahead and the real-time LMP forecast?". The results show that solar and wind generation do cause a significant impact on the price forecast.

A detailed discussion on the problem statement, definitions for nodal and trading zone LMPs and the supply -demand curve for price determination was presented in this research. The methods described for data cleaning and extraction would prove to be useful since the OASIS portal for CAISO can provide data for a maximum of 30 days at a single time. Also, the dataset had to be cleaned and processed before the analysis. A single dataset with all the historical data for load, fuel prices, renewable generation which was generated, would be a great place for anyone to start their analysis.

The trend observed by the load, renewable (solar+wind) generation, fuel prices, LMPs provide a picture of the current scenario for California ISO and indicate the variation since the last few years. Moreover, the plots with renewable generation and LMPs state that there is a significant dip in the prices when renewable generation starts to increase and reaches its peak during the day. Also, a spike in LMPs is seen when the renewable generation falls to a minimum value.

The ML algorithms described in Chapter 5 provide a reasonable forecast of LMPs for specific days. The impacts of every feature obtained from the forecast describe how much the

forecasted price would be affected if there is a deviation in the dataset. It was expected that the load would have the maximum impact on the day-ahead price forecast and is verified by observing the feature importance plots. An interesting finding from this research is that the renewable generation ranks second in terms of impact which is unusual as generation outages were also included in the dataset. Hence, there could be price fluctuations in case of an intermittent renewable generation scenario.

The results show that machine learning could be applied effectively in terms of determining the impacts of several parameters on the day-ahead and real-time price forecast for California. This would help grid operators in determining which parameters would impact the price forecast significantly and which parameters could be ignored due to have little or less significant impact.

7.2 Future Work

As discussed earlier, the goal of this research was to analyse the impact of renewable generation on the locational marginal price (LMP) forecast. Features such as load, renewable generation, fuel prices, generation outages were considered for the prediction. The prediction did not involve factors such as transmission outages, binding constraints, etc. Hence, there is scope for improving the accuracy of the forecast. Some of the possibilities are:

- Solving the economic dispatch problem at every node to forecast the LMP
- Passing features such as transmission outages and constraints on the system to determine their impact on LMP
- Involving features such as weather forecast, temperature etc. to get an accurate forecast
- Using deep-learning algorithms to improve the forecasting accuracy

REFERENCES

[1] Juan M. Morales, Student Member, IEEE, Antonio J. Conejo, Fellow, IEEE, and Juan Pérez-Ruiz, Member, IEEE, 'Simulating the Impact of Wind Production on Locational Marginal Prices

[2] Yuting Ji, Jinsub Kim, Robert J. Thomas and Lang Tong, 'Forecasting Real-Time Locational Marginal Price: A State Space Approach, June 25, 2016

[3] Ashkan Sadeghi-Mobarakeh, Mahdi Kohansal, Evangelos E. Papalexakis, and Hamed Mohsenian-Rad, 'Data Mining based on Random Forest Model to Predict the California ISO Day-ahead Market Prices, April 26, 2017

[4] J.C. Smith, Stephen Beuning, Henry Durrwachter, Erik Ela, David Hawkins, Brendan Kirby, Warren Lasher, Jonathan Lowell, Kevin Porter, Ken Schuyler, Paul Sotkiewicz,' Impact of Variable Renewable Energy on US Electricity Markets, March 5, 2010

[5] Ryan Wiser, Andrew Mills, Joachim Seel, Todd Levin, Audum Botterud, 'Impact of Variable Renewable Energy on Bulk Power System Assets, Pricing, and Costs, November 2017

[6] Joachim Seel, Andrew Mills, Ryan Wiser, 'Impact of High Variable Renewable Energy (VRE) Futures on Wholesale Electricity Prices, and on Electric-Sector Decision Making, May 16, 2018

[7] Dov Quint, Steve Dahlke, 'The Impact of Wind Generation on Wholesale Electricity Prices in the Midcontinent Market, June 5, 2017

[8] J. Zarnikau, C.K. Woo, R. Baldick, 'Did the introduction of a nodal market structure impact wholesale electricity prices in the Texas (ERCOT) market?, October 10,2018

[9] Michael E. Birk," Impact of Distributed Energy Resources on Locational Marginal Prices and Electricity Networks, June 2016

[10] Philipp Brown, "U.S. Renewable Electricity: How Does Wind Generation Impact Competitive Power Markets?, November 7, 2012

[11] Lucas Davis, "Is Solar Really the Reason for Negative Electricity Prices?, https://energyathaas.wordpress.com/2017/08/28/is-solar-really-the-reason-for-negative-electricity-prices/

[12] Chris Namovicz," Rising solar generation in California coincides with negative wholesale electricity prices, EIA, TODAY IN ENERGY, April 7. 2017

[13] Will Koehrsen, "Hyperparameter Tuning the Random Forest in Python, https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74 , Jan 9, 2018

[14] Mohtadi Ben Fraj, "In Depth: Parameter tuning for Gradient Boosting, https://medium.com/all-things-ai/in-depth-parameter-tuning-for-gradient-boosting-3363992e9bae , Dec 24, 2017

[15] Eryk Lewinson, "Explaining Feature Importance by example of a Random Forest, https://towardsdatascience.com/explaining-feature-importance-by-example-of-a-random-forest-d9166011959e , Feb 11,2017

[16] Aarshay Jain, "Complete Guide to Parameter Tuning in XGBoost (with codes in Python), https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/ , March 1, 2016

[17] Today's Outlook, Prices, California ISO " http://www.caiso.com/TodaysOutlook/Pages/Prices.aspx

[18] California ISO, "Market Redesign & Technology Upgrade (MRTU) Project CAISO Proposal, October 26, 2005

[19] AEP Energy, "Real-Time vs Day-Ahead Pricing, January 5, 2018, https://www.aepenergy.com/2018/01/05/december-2017-edition/

[20] California ISO, "Interface specification for OASIS, Fall 2018 Release, Version 5.1.4, July 25, 2018, http://www.caiso.com/Documents/OASIS-InterfaceSpecification_v5_1_4Clean_Fall2018Release.pdf

[21] California ISO, "Q3 2018 Report on Market Issues and Performance, November 1, 2018, http://www.caiso.com/Documents/2018ThirdQuarterReportonMarketIssuesandPerformance.pdf

[22] Benjamin Aunkofer, "Ensemble Learning, December 3, 2017, https://data-science-blog.com/blog/2017/12/03/ensemble-learning/

[23] Zhi-Hua Zhou, "Ensemble Learning, https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/springerEBR09.pdf

[24] Pierre Geurts, Damien Ernst, Louis Wehenkel, “Extremely randomized trees, March 2, 2006

[25] Vaibhav Kumar, “ Random forests and decision trees from scratch in python, October 23, 2018, https://towardsdatascience.com/random-forests-and-decision-trees-from-scratch-in-python-3e4fa5ae4249

[26] Krishni Hewa, “ A Beginners guide to Random Forest Regression, November 26, 2018, https://medium.com/datadriveninvestor/random-forest-regression-9871bc9a25eb

[27] Jerome H. Friedman, “Stochastic Gradient Boosting, March 26, 1999

[28] Harshdeep Singh, “Understanding Gradient Boosting Machines, November 3, 2018, https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab

[29] Maria Jesus, “Introduction to Boosted Trees, March 14, 2017, https://blog.bigml.com/2017/03/14/introduction-to-boosted-trees/

[30] Tianqi Chen, Carlos Guestrin, “XGBoost: A Scalable Tree Boosting System, 2016

[31] Krishni Hewa, “ K-Fold Cross Validation, December 16, 2018, https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833

[32] Michael Polson, Vadim Sokolov, “Deep Learning for Energy Markets, April 2019

[33] Soren Amelang, Kerstine Appunn, “ The causes and effects of negative power prices, January 5, 2018, https://www.cleanenergywire.org/factsheets/why-power-prices-turn-negative

[34] Penn State College of Earth and Mineral Sciences, EBF 483: Introduction to Electricity Markets (12,3), “The Curious Case of the Negative Prices, https://www.e-education.psu.edu/ebf483/node/717

[35] Raunak Goswami, “Root-Mean-Square Error (RMSE) | Machine Learning, https://www.includehelp.com/ml-ai/root-mean-square%20error-rmse.aspx

[36] Model Evaluation – Regression, https://www.saedsayad.com/model_evaluation_r.htm

[37] Sven Stringer, "Feature importance – what's in a name?, https://medium.com/bigdatarepublic/feature-importance-whats-in-a-name-79532e59eea3

[38] "Regional Transmission Organizations (RTO)/Independent System Operators (ISO)", https://www.ferc.gov/industries/electric/indus-act/rto.asp

[39] "California ISO Open Access Same-time Information System (OASIS)", http://oasis.caiso.com/mrioasis/logon.do

[40] California ISO, "Market Redesign & Technology Upgrade (MRTU) Project CAISO Proposal, October 2005, https://www.caiso.com/Documents/N-5-AppendixA-October26_2005-October19_2005TradingHubWhitePaper.pdf

[41] "Machine Learning for Beginners, https://towardsdatascience.com/machine-learning-for-beginners-d247a9420dab

APPENDIX A

PYTHON CODES

A. PYTHON CODES

LOAD ANALYSIS

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 19 13:40:31 2019

@author: chinm
"""

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.dates as mdates

# Read the data from 2015-2018
demand_2015= pd.read_csv(r'CAISO_hourlydemand_2015.csv')
demand_2016 = pd.read_csv(r'CAISO_hourlydemand_2016.csv')
demand_2017 = pd.read_csv(r'CAISO_hourlydemand_2017.csv')
demand_2018 = pd.read_csv(r'CAISO_hourlydemand_2018.csv')

demand_2015['Date'] = demand_2015.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
demand_2015['Date'] = pd.to_datetime(demand_2015['Date'])
demand_2016['Date'] = demand_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
demand_2016['Date'] = pd.to_datetime(demand_2016['Date'])
demand_2017['Date'] = demand_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%m/%d/%Y %H:%M'))
demand_2017['Date'] = pd.to_datetime(demand_2017['Date'])
demand_2018['Date'] = demand_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
demand_2018['Date'] = pd.to_datetime(demand_2018['Date'])

#Join dataframes
demand = [demand_2015,demand_2016,demand_2017, demand_2018]
demand = pd.concat(demand)
demand['load']= demand['load']/1000
demand = demand.set_index('Date')
demand = demand.drop(['zone'],axis=1)

#Plot the data
ax1 = demand.plot(legend=False)
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
ax1.set_title('Variation of Electric load for California ISO from 2015-2018',fontsize=15)
ax1.set_xlabel('Date',fontsize=15)
ax1.set_ylabel('GW',fontsize=15)
```

```
ax1.tick_params(axis='both', which='both', labelsize=15)

#Plot the averages
g = demand.groupby((demand.index.year)).mean()
g.plot(kind='bar',legend=False,rot=0)
plt.xlabel('Year',fontsize=15)
plt.ylabel('GW',fontsize=15)
plt.title('Average yearly loads for California ISO from 2015-2018',fontsize=15)
plt.tick_params(axis='both', which='both', labelsize=15)
```

FUEL PRICES ANALYSIS

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 19 13:40:31 2019

@author: chinm
"""

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import matplotlib.dates as mdates


# Read the data from 2015-2018
fuelprices_2015 = pd.read_csv(r'CAISO_fuelprices2015.csv')
fuelprices_2016 = pd.read_csv(r'CAISO_fuelprices2016.csv')
fuelprices_2017 = pd.read_csv(r'CAISO_fuelprices2017.csv')
fuelprices_2018 = pd.read_csv(r'CAISO_fuelprices2018.csv')


fuelprices_2015['Date'] = fuelprices_2015.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
fuelprices_2015['Date'] = pd.to_datetime(fuelprices_2015['Date'])
fuelprices_2016['Date'] = fuelprices_2016.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
fuelprices_2016['Date'] = pd.to_datetime(fuelprices_2016['Date'])
fuelprices_2017['Date'] = fuelprices_2017.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
fuelprices_2017['Date'] = pd.to_datetime(fuelprices_2017['Date'])
fuelprices_2018['Date'] = fuelprices_2018.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
fuelprices_2018['Date'] = pd.to_datetime(fuelprices_2018['Date'])
#


#Join dataframes
```

```
fuelprices = [fuelprices_2015,fuelprices_2016,fuelprices_2017,fuelprices_2018]
fuelprices = pd.concat(fuelprices)
fuelprices = fuelprices.set_index('Date')
#Plot the data
ax = fuelprices.plot(legend=False)
ax.set_title('Variation of Fuel Prices for California ISO from 2015-2018',fontsize=15)
ax.set_xlabel('Date',fontsize=15)
ax.set_ylabel('$/mmbtu',fontsize=15)
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
ax.tick_params(axis='both', which='both', labelsize=15)
#Plot the yearly averages
g = fuelprices.groupby(fuelprices.index.year).mean()
g.plot(kind='bar',legend=False,rot=0)
plt.title('Average Yearly fuel prices for California ISO from 2015-2018',fontsize=15)
plt.xlabel('Year',fontsize=15)
plt.ylabel('$/mmbtu',fontsize=15)
plt.tick_params(axis='both', which='both', labelsize=15)
```

RENEWABLE GENERATION ANALYSIS

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 19 13:40:31 2019

@author: chinm
"""

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.dates as mdates

#Load the data
rengen_2015 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2015.csv')
rengen_2016 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2016.csv')
rengen_2017 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2017.csv')
rengen_2018 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2018.csv')


rengen_2015['Date'] = rengen_2015.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2015['Date'] = pd.to_datetime(rengen_2015['Date'])
rengen_2016['Date'] = rengen_2016.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2016['Date'] = pd.to_datetime(rengen_2016['Date'])
rengen_2017['Date'] = rengen_2017.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2017['Date'] = pd.to_datetime(rengen_2017['Date'])
rengen_2018['Date'] = rengen_2018.Date.apply(
```

```
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2018['Date'] = pd.to_datetime(rengen_2018['Date'])
#Join dataframes

rengen = [rengen_2015,rengen_2016,rengen_2017,rengen_2018]
rengen = pd.concat(rengen)
rengen = rengen.set_index('Date')
rengen = rengen/1000
#Plot the data
ax1 = rengen.plot(legend=False)
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
ax1.set_title('Renewable (Solar+Wind) generation for California ISO from 2015-
2018',fontsize=15)
ax1.set_xlabel('Date',fontsize=15)
ax1.set_ylabel('GW',fontsize=15)
ax1.tick_params(axis='both', which='both', labelsize=15)

#Plot the yearly averages
g = rengen.groupby(rengen.index.year).mean()
e,kind='bar',rot=0)
plt.title('Average Yearly generation from Solar and Wind for California ISO from 2015-
2018',fontsize=15)
plt.xlabel('Year',fontsize=15)
plt.ylabel('GW',fontsize=15)
plt.tick_params(axis='both', which='both', labelsize=15)
```

GENERATION OUTAGES ANALYSIS

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 19 13:40:31 2019

@author: chinm
"""

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.dates as mdates

#Load data from 2016-2018
genoutages_2016 = pd.read_csv(r'CAISO_genoutages_hourly_2016.csv')
genoutages_2017 = pd.read_csv(r'CAISO_genoutages_hourly_2017.csv')
genoutages_2018 = pd.read_csv(r'CAISO_genoutages_hourly_2018.csv')


genoutages_2016['Date'] = genoutages_2016.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
genoutages_2016['Date'] = pd.to_datetime(genoutages_2016['Date'])
genoutages_2017['Date'] = genoutages_2017.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
genoutages_2017['Date'] = pd.to_datetime(genoutages_2017['Date'])
genoutages_2018['Date'] = genoutages_2018.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
genoutages_2018['Date'] = pd.to_datetime(genoutages_2018['Date'])

#Join dataframes
genoutages = [genoutages_2016,genoutages_2017,genoutages_2018]
genoutages = pd.concat(genoutages)
genoutages['GW'] = genoutages['MW']/1000
genoutages = genoutages.drop(['MW'],axis=1)
genoutages = genoutages.set_index('Date')

#Plot the data
ax1 = genoutages.plot(legend=False)
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
ax1.set_title('Generation Outages for California ISO from 2016-2018',fontsize=15)
ax1.set_xlabel('Date',fontsize=15)
ax1.set_ylabel('GW',fontsize=15)
ax1.tick_params(axis='both', which='both', labelsize=15)

#Plot the yearly averages
g = genoutages.groupby(genoutages.index.year).mean()
g.plot(legend=False,kind='bar',rot=0)
plt.title('Average Yearly generation outages for California ISO from 2016-2018')
plt.xlabel('Year',fontsize=15)
plt.ylabel('GW',fontsize=15)
plt.tick_params(axis='both', which='both', labelsize=15)
```

LMP ANALYSIS

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 19 13:40:31 2019

@author: chinm
"""

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.dates as mdates

#Load the data from 2015-2018
prices_2015 = pd.read_csv(r'CAISO_hourlypricing2015.csv')
prices_2016 = pd.read_csv(r'CAISO_hourlypricing2016.csv')
prices_2017 = pd.read_csv(r'CAISO_hourlypricing2017.csv')
prices_2018 = pd.read_csv(r'CAISO_hourlypricing2018.csv')

prices_2015['Date'] = prices_2015.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2015['Date'] = pd.to_datetime(prices_2015['Date'])
prices_2016['Date'] = prices_2016.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2016['Date'] = pd.to_datetime(prices_2016['Date'])
prices_2017['Date'] = prices_2017.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2017['Date'] = pd.to_datetime(prices_2017['Date'])
prices_2018['Date'] = prices_2018.Date.apply(
    lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2018['Date'] = pd.to_datetime(prices_2018['Date'])
#

#Join dataframes
prices = [prices_2015,prices_2016,prices_2017,prices_2018]
prices = pd.concat(prices)
prices = prices.set_index('Date')

#Plot the data
ax1 = prices.plot(legend=False)
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
ax1.set_title('Variation of electricity prices for California ISO SP15 trading hub from 2015-
2018',fontsize=15)
ax1.set_xlabel('Date',fontsize=15)
ax1.set_ylabel('$/MWh',fontsize=15)
ax1.tick_params(axis='both', which='both', labelsize=15)

#Plot the yearly averages
```

```
g = prices.groupby(prices.index.year).mean()
g.plot(kind='bar',legend=False,rot=0)
plt.xlabel('Year',fontsize=15)
plt.ylabel('$/MWh',fontsize=15)
plt.title('Average yearly electricity prices for California ISO from 2015-2018',fontsize=15)
plt.tick_params(axis='both', which='both', labelsize=15)
```

PRICING AND RENEWABLE GENERATION

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 19 13:40:31 2019

@author: chinm
"""

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.dates as mdates

#Load the data
prices_2015 = pd.read_csv(r'CAISO_hourlypricing2015.csv')
prices_2016 = pd.read_csv(r'CAISO_hourlypricing2016.csv')
prices_2017 = pd.read_csv(r'CAISO_hourlypricing2017.csv')
prices_2018 = pd.read_csv(r'CAISO_hourlypricing2018.csv')

rengen_2015 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2015.csv')
rengen_2016 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2016.csv')
rengen_2017 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2017.csv')
rengen_2018 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2018.csv')

rengen_2015['Date'] = rengen_2015.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2015['Date'] = pd.to_datetime(rengen_2015['Date'])
rengen_2016['Date'] = rengen_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2016['Date'] = pd.to_datetime(rengen_2016['Date'])
rengen_2017['Date'] = rengen_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2017['Date'] = pd.to_datetime(rengen_2017['Date'])
rengen_2018['Date'] = rengen_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2018['Date'] = pd.to_datetime(rengen_2018['Date'])

prices_2015['Date'] = prices_2015.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2015['Date'] = pd.to_datetime(prices_2015['Date'])
prices_2016['Date'] = prices_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2016['Date'] = pd.to_datetime(prices_2016['Date'])
```

```
prices_2017['Date'] = prices_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2017['Date'] = pd.to_datetime(prices_2017['Date'])
prices_2018['Date'] = prices_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2018['Date'] = pd.to_datetime(prices_2018['Date'])
#

rengen_2015 = rengen_2015.set_index('Date')
rengen_2016 = rengen_2016.set_index('Date')
rengen_2017 = rengen_2017.set_index('Date')
rengen_2018 = rengen_2018.set_index('Date')

prices_2015 = prices_2015.set_index('Date')
prices_2016 = prices_2016.set_index('Date')
prices_2017 = prices_2017.set_index('Date')
prices_2018 = prices_2018.set_index('Date')

neg_2015=prices_2015[prices_2015['price']<0]
neg_2016=prices_2016[prices_2016['price']<0]
neg_2017=prices_2017[prices_2017['price']<0]
neg_2018=prices_2018[prices_2018['price']<0]
##
prices_2015['MW']= rengen_2015['MW']
prices_2016['MW']= rengen_2016['MW']
prices_2017['MW']= rengen_2017['MW']
prices_2018['MW']= rengen_2018['MW']
xticks = np.arange(0,24,1)

#Plot the data
day_2015 = prices_2015['20151225':'20151225']
day_2015= day_2015.reset_index()
day_2015 = day_2015.drop(['Date'],axis=1)
fig, axes = plt.subplots(nrows=2, ncols=1)
xticks = np.arange(0,25,1)
ax1 =
day_2015[['price']].plot(kind='bar',legend=False,ax=axes[0],xticks=xticks,xlim=(0,24),color='r')
ax2 =
day_2015[['MW']].plot(kind='bar',legend=False,ax=axes[1],xticks=xticks,xlim=(0,24),color='g')
ax1.tick_params(axis='both', which='both', labelsize=15)
ax2.tick_params(axis='both', which='both', labelsize=15)
ax1.set_xlabel('',fontsize=15)
ax2.set_xlabel('',fontsize=15)
ax1.set_ylabel('$/MWh',fontsize=15)
ax2.set_ylabel('MW',fontsize=15)

#Plot the data
day_2016 = prices_2016['20160429':'20160429']
day_2016= day_2016.reset_index()
day_2016 = day_2016.drop(['Date'],axis=1)
fig, axes = plt.subplots(nrows=2, ncols=1)
```

```
ax1 =
day_2016[['price']].plot(kind='bar',legend=False,ax=axes[0],xticks=xticks,xlim=(0,24),color='r')
ax2 =
day_2016[['MW']].plot(kind='bar',legend=False,ax=axes[1],xticks=xticks,xlim=(0,24),color='g')
ax1.tick_params(axis='both', which='both', labelsize=15)
ax2.tick_params(axis='both', which='both', labelsize=15)
ax1.tick_params(axis='both', which='both', labelsize=15)
ax2.tick_params(axis='both', which='both', labelsize=15)
ax1.set_xlabel('',fontsize=15)
ax2.set_xlabel('',fontsize=15)
ax1.set_ylabel('$/MWh',fontsize=15)
ax2.set_ylabel('MW',fontsize=15)
#
#Plot the data
day_2017 = prices_2017['20170420':'20170420']
day_2017= day_2017.reset_index()
day_2017 = day_2017.drop(['Date'],axis=1)
fig, axes = plt.subplots(nrows=2, ncols=1)
ax1 =
day_2017[['price']].plot(kind='bar',legend=False,ax=axes[0],xticks=xticks,xlim=(0,24),color='r')
ax2 =
day_2017[['MW']].plot(kind='bar',legend=False,ax=axes[1],xticks=xticks,xlim=(0,24),color='g')
ax1.tick_params(axis='both', which='both', labelsize=15)
ax2.tick_params(axis='both', which='both', labelsize=15)
ax1.tick_params(axis='both', which='both', labelsize=15)
ax2.tick_params(axis='both', which='both', labelsize=15)
ax1.set_xlabel('',fontsize=15)
ax2.set_xlabel('',fontsize=15)
ax1.set_ylabel('$/MWh',fontsize=15)
ax2.set_ylabel('MW',fontsize=15)
#
#Plot the data
day_2018 = prices_2018['20181111':'20181111']
day_2018= day_2018.reset_index()
day_2018 = day_2018.drop(['Date'],axis=1)
fig, axes = plt.subplots(nrows=2, ncols=1)
ax1 =
day_2018[['price']].plot(kind='bar',legend=False,ax=axes[0],xticks=xticks,xlim=(0,24),color='r')
ax2 =
day_2018[['MW']].plot(kind='bar',legend=False,ax=axes[1],xticks=xticks,xlim=(0,24),color='g')
ax1.tick_params(axis='both', which='both', labelsize=15)
ax2.tick_params(axis='both', which='both', labelsize=15)
ax1.tick_params(axis='both', which='both', labelsize=15)
ax2.tick_params(axis='both', which='both', labelsize=15)
ax1.set_xlabel('',fontsize=15)
ax2.set_xlabel('',fontsize=15)
ax1.set_ylabel('$/MWh',fontsize=15)
ax2.set_ylabel('MW',fontsize=15)

#Comparison of negative prices and renewable generation
#plt.suptitle('Renewable Generation and Trading Hub Pricing for a day in 2015')
#prices_2016[prices_2016['price']>200]=0
```

```
#week_2018 = prices_2016['20160401':'20160407']
#fig, ax = plt.subplots()
#plt.plot(week_2018['price'], 'r', label='Prices',linewidth=3)
##plt.ylim(25,40)
#plt.legend(loc=2)
#ax.tick_params(axis='both', which='both', labelsize=15)
#ax.xaxis.set_major_formatter(mdates.DateFormatter('%d/%m'))
## Get second axis
#ax2 = ax.twinx()
#plt.plot(week_2018['MW'], marker='o',linestyle='--',linewidth=3.5)
##plt.ylim(0,10)
#plt.legend()
#ax2.tick_params(axis='both', which='both', labelsize=15)
#ax.set_ylabel('Price in $/Mwh',fontsize=15)
#ax2.set_ylabel('Renewable (Solar+Wind) Generation in MWh',fontsize=15)
#ax.set_xlabel('Day',fontsize=15)
#ax.set_title('Electricity prices and Renewable Generation for April first week in
2016',fontsize=15)
```

FINAL CODE

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 19 13:40:31 2019

@author: chinm
"""

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np


#Load the data
prices_2016 = pd.read_csv(r'CAISO_hourlypricing2016.csv')
prices_2017 = pd.read_csv(r'CAISO_hourlypricing2017.csv')
prices_2018 = pd.read_csv(r'CAISO_hourlypricing2018.csv')

fuelprices_2016 = pd.read_csv(r'CAISO_fuelprices2016.csv')
fuelprices_2017 = pd.read_csv(r'CAISO_fuelprices2017.csv')
fuelprices_2018 = pd.read_csv(r'CAISO_fuelprices2018.csv')

demand_2016 = pd.read_csv(r'CAISO_hourlydemand_2016.csv')
demand_2017 = pd.read_csv(r'CAISO_hourlydemand_2017.csv')
demand_2018 = pd.read_csv(r'CAISO_hourlydemand_2018.csv')

rengen_2016 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2016.csv')
rengen_2017 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2017.csv')
rengen_2018 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2018.csv')

genoutages_2016 = pd.read_csv(r'CAISO_genoutages_hourly_2016.csv')
```

```
genoutages_2017 = pd.read_csv(r'CAISO_genoutages_hourly_2017.csv')
genoutages_2018 = pd.read_csv(r'CAISO_genoutages_hourly_2018.csv')

#Convert the timestamp into a datetime object
fuelprices_2016['Date'] = fuelprices_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
fuelprices_2016['Date'] = pd.to_datetime(fuelprices_2016['Date'])
fuelprices_2017['Date'] = fuelprices_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
fuelprices_2017['Date'] = pd.to_datetime(fuelprices_2017['Date'])
fuelprices_2018['Date'] = fuelprices_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
fuelprices_2018['Date'] = pd.to_datetime(fuelprices_2018['Date'])
#

prices_2016['Date'] = prices_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2016['Date'] = pd.to_datetime(prices_2016['Date'])
prices_2017['Date'] = prices_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2017['Date'] = pd.to_datetime(prices_2017['Date'])
prices_2018['Date'] = prices_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2018['Date'] = pd.to_datetime(prices_2018['Date'])
#
demand_2016['Date'] = demand_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
demand_2016['Date'] = pd.to_datetime(demand_2016['Date'])
demand_2017['Date'] = demand_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%m/%d/%Y %H:%M'))
demand_2017['Date'] = pd.to_datetime(demand_2017['Date'])
demand_2018['Date'] = demand_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
demand_2018['Date'] = pd.to_datetime(demand_2018['Date'])

rengen_2016['Date'] = rengen_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2016['Date'] = pd.to_datetime(rengen_2016['Date'])
rengen_2017['Date'] = rengen_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2017['Date'] = pd.to_datetime(rengen_2017['Date'])
rengen_2018['Date'] = rengen_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2018['Date'] = pd.to_datetime(rengen_2018['Date'])
#

genoutages_2016['Date'] = genoutages_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
genoutages_2016['Date'] = pd.to_datetime(genoutages_2016['Date'])
genoutages_2017['Date'] = genoutages_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
genoutages_2017['Date'] = pd.to_datetime(genoutages_2017['Date'])
```

```
genoutages_2018['Date'] = genoutages_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
genoutages_2018['Date'] = pd.to_datetime(genoutages_2018['Date'])

#Join dataframes
fuelprices = [fuelprices_2016,fuelprices_2017,fuelprices_2018]
fuelprices = pd.concat(fuelprices)
prices = [prices_2016,prices_2017,prices_2018]
prices = pd.concat(prices)
rengen = [rengen_2016,rengen_2017,rengen_2018]
rengen = pd.concat(rengen)
demand = [demand_2016,demand_2017, demand_2018]
demand = pd.concat(demand)
genoutages = [genoutages_2016,genoutages_2017,genoutages_2018]
genoutages = pd.concat(genoutages)

#Create a single dataframe with all the data
demand['LMP']= prices['price']
demand['Fuelprices']= fuelprices['PRC']
demand['Renewable_Generation']= rengen['MW']
demand['Generation Outages']= genoutages['MW']
demand['Year']= demand['Date'].dt.year
demand['Month']= demand['Date'].dt.month
demand['Day']= demand['Date'].dt.day
demand['Hour']= demand['Date'].dt.hour
demand['LMP']= demand['LMP'].astype(float)
demand['load']= demand['load'].astype(float)
demand['Fuelprices']= demand['Fuelprices'].astype(float)
demand['Renewable_Generation']= demand['Renewable_Generation'].astype(float)
demand['Generation Outages']= demand['Generation Outages'].astype(float)
demand['load']= demand['load']/1000
demand['Generation Outages']= demand['Generation Outages']/1000
demand['Renewable_Generation']= demand['Renewable_Generation']/1000
demand = demand.reset_index()
demand_new = demand
demand_new = demand_new.drop(['index','zone','load','Fuelprices',
                        'Renewable_Generation','Generation Outages',
                         'Hour','Year','Month','Day'],axis=1)
demand = demand.drop(['Date','index','zone'],axis=1)
demand = demand.drop(['Year','Month','Day'],axis=1)

#Split the data into test and training
len_data = 26280
test_lowerlim = 18000
test_upperlim = test_lowerlim + 24
demand_new = demand_new[test_lowerlim:test_upperlim]
X_train1= demand.iloc[0:test_lowerlim,demand.columns!='LMP'].values
X_train2 = demand.iloc[test_upperlim:len_data,demand.columns!='LMP'].values
X_train = np.vstack((X_train1,X_train2))
X_test = demand.iloc[test_lowerlim:test_upperlim,demand.columns!='LMP'].values
y_train1= demand.iloc[0:test_lowerlim,1].values
y_train2 = demand.iloc[test_upperlim:len_data,1].values
```

```
y_train = np.concatenate([y_train1,y_train2])
y_test = demand.iloc[test_lowerlim:test_upperlim,1].values

###   Import libraries from sklearn
from sklearn import ensemble
from sklearn.metrics import mean_squared_error
import xgboost as xg
### Store the different type of regressors used in a list.Every regressor in the
### list is tried and tested for the score. Apparently, the Gradient Boosting
### regressor gives the maximum score

regressors = [
        ensemble.ExtraTreesRegressor(n_estimators
=400,random_state=0,max_depth=24,max_features='sqrt'),
        ensemble.RandomForestRegressor(random_state=0,n_estimators=100),
        ensemble.GradientBoostingRegressor(max_depth=12, max_features='log2',
                                   n_estimators= 400, subsample= 0.95,learning_rate=0.01),
        xg.XGBRegressor(n_estimators=1400,
learning_rate=0.01,max_depth=17,gamma=5,subsample=0.85,
                    colsample_bytree = 0.85,min_child_weight=1)
        ]
clf = regressors[0]
y_test1 = np.zeros(len(y_test))
y_train1 = np.zeros(len(y_train))
#### Train model
model = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_trainpred = clf.predict(X_train)
print(clf.score(X_test,y_test))
print('The rmse of test prediction is:', (mean_squared_error(y_test, y_pred)
/mean_squared_error(y_test, y_test1))** 0.5)
print('The rmse of train prediction is:', (mean_squared_error(y_train, y_trainpred)
/mean_squared_error(y_train, y_train1))** 0.5)

#Plot the actual and forecasted LMPs
demand_new['Predicted LMP']= y_pred
demand_new = demand_new.set_index('Date')
demand_new.plot(legend=True)
plt.xlabel('Hour of the day',fontsize='x-large')
plt.ylabel('$/MWh',fontsize='x-large')
plt.title('Comparison between actual and forecasted LMP for January 21,2018',fontsize='x-large')
plt.tick_params(axis='both', which='minor', labelsize=15)
plt.tick_params(axis='both', which='major', labelsize=15)

#### Calculate feature importances
importances = model.feature_importances_
importances = pd.DataFrame(importances)
importances = importances.T
labels = list(demand)
importances.columns = ['load',
 'Fuelprices',
 'Renewable_Generation',
```

```
 'Generation Outages',
 'Hour']
importances = importances.T
importances.plot(kind='bar',rot=10,legend=False)
#plt.title('Impact of different parameters on LMP for  January 21,2018 using Extra Trees Regression',fontsize=15)
#plt.title('Impact of different parameters on LMP for June 13,2018 using XGBoost Regression',fontsize=15)
#plt.title('Impact of different parameters on LMP for January 21,2018 using Gradient Boost Regression',fontsize=15)
plt.tick_params(axis='both', which='minor', labelsize=15)
plt.tick_params(axis='both', which='major', labelsize=15)
```

FINAL CODE LSTM NETWORKS

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 19 13:40:31 2019

@author: chinm
"""

import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,LSTM
import matplotlib.pyplot as plt
seed = 7
np.random.seed(seed)

#Load the data
prices_2016 = pd.read_csv(r'CAISO_hourlypricing2016.csv')
prices_2017 = pd.read_csv(r'CAISO_hourlypricing2017.csv')
prices_2018 = pd.read_csv(r'CAISO_hourlypricing2018.csv')

fuelprices_2016 = pd.read_csv(r'CAISO_fuelprices2016.csv')
fuelprices_2017 = pd.read_csv(r'CAISO_fuelprices2017.csv')
fuelprices_2018 = pd.read_csv(r'CAISO_fuelprices2018.csv')

demand_2016 = pd.read_csv(r'CAISO_hourlydemand_2016.csv')
demand_2017 = pd.read_csv(r'CAISO_hourlydemand_2017.csv')
demand_2018 = pd.read_csv(r'CAISO_hourlydemand_2018.csv')

rengen_2016 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2016.csv')
rengen_2017 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2017.csv')
rengen_2018 = pd.read_csv(r'CAISO_hourlyrenewablegeneration_SP15_2018.csv')

genoutages_2016 = pd.read_csv(r'CAISO_genoutages_hourly_2016.csv')
genoutages_2017 = pd.read_csv(r'CAISO_genoutages_hourly_2017.csv')
genoutages_2018 = pd.read_csv(r'CAISO_genoutages_hourly_2018.csv')
```

```
fuelprices_2016['Date'] = fuelprices_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
fuelprices_2016['Date'] = pd.to_datetime(fuelprices_2016['Date'])
fuelprices_2017['Date'] = fuelprices_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
fuelprices_2017['Date'] = pd.to_datetime(fuelprices_2017['Date'])
fuelprices_2018['Date'] = fuelprices_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
fuelprices_2018['Date'] = pd.to_datetime(fuelprices_2018['Date'])
#

prices_2016['Date'] = prices_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2016['Date'] = pd.to_datetime(prices_2016['Date'])
prices_2017['Date'] = prices_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2017['Date'] = pd.to_datetime(prices_2017['Date'])
prices_2018['Date'] = prices_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
prices_2018['Date'] = pd.to_datetime(prices_2018['Date'])
#
demand_2016['Date'] = demand_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
demand_2016['Date'] = pd.to_datetime(demand_2016['Date'])
demand_2017['Date'] = demand_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%m/%d/%Y %H:%M'))
demand_2017['Date'] = pd.to_datetime(demand_2017['Date'])
demand_2018['Date'] = demand_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
demand_2018['Date'] = pd.to_datetime(demand_2018['Date'])

rengen_2016['Date'] = rengen_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2016['Date'] = pd.to_datetime(rengen_2016['Date'])
rengen_2017['Date'] = rengen_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2017['Date'] = pd.to_datetime(rengen_2017['Date'])
rengen_2018['Date'] = rengen_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
rengen_2018['Date'] = pd.to_datetime(rengen_2018['Date'])
#

genoutages_2016['Date'] = genoutages_2016.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
genoutages_2016['Date'] = pd.to_datetime(genoutages_2016['Date'])
genoutages_2017['Date'] = genoutages_2017.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
genoutages_2017['Date'] = pd.to_datetime(genoutages_2017['Date'])
genoutages_2018['Date'] = genoutages_2018.Date.apply(
   lambda x: pd.to_datetime(x).strftime('%d/%m/%Y %H:%M'))
genoutages_2018['Date'] = pd.to_datetime(genoutages_2018['Date'])
```

```
#Join dataframes
fuelprices = [fuelprices_2016,fuelprices_2017,fuelprices_2018]
fuelprices = pd.concat(fuelprices)
prices = [prices_2016,prices_2017,prices_2018]
prices = pd.concat(prices)
rengen = [rengen_2016,rengen_2017,rengen_2018]
rengen = pd.concat(rengen)
demand = [demand_2016,demand_2017, demand_2018]
demand = pd.concat(demand)
genoutages = [genoutages_2016,genoutages_2017,genoutages_2018]
genoutages = pd.concat(genoutages)
demand['LMP']= prices['price']
demand['Fuelprices']= fuelprices['PRC']
demand['Renewable_Generation']= rengen['MW']
demand['Generation Outages']= genoutages['MW']

demand['Year']= demand['Date'].dt.year
demand['Month']= demand['Date'].dt.month
demand['Day']= demand['Date'].dt.day
demand['Hour']= demand['Date'].dt.hour
demand['LMP']= demand['LMP'].astype(float)
demand['load']= demand['load'].astype(float)
demand['Fuelprices']= demand['Fuelprices'].astype(float)
demand['Renewable_Generation']= demand['Renewable_Generation'].astype(float)
demand['Generation Outages']= demand['Generation Outages'].astype(float)
demand['load']= demand['load']/1000
demand['Generation Outages']= demand['Generation Outages']/1000
demand = demand.reset_index()
demand = demand.drop(['Date','index','zone'],axis=1)
demand = demand[['LMP','load','Fuelprices','Renewable_Generation','Generation
Outages','Year','Month','Day','Hour']]
values = demand.values
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled = scaler.fit_transform(values)

#convert data to supervised form

def to_supervised(data,dropNa = True,lag = 1):
    df = pd.DataFrame(data)
    column = []
    column.append(df)
    for i in range(1,lag+1):
        column.append(df.shift(-i))
    df = pd.concat(column,axis=1)
    df.dropna(inplace = True)
    features = data.shape[1]
    df = df.values
    supervised_data = df[:,:features*lag]
    supervised_data = np.column_stack( [supervised_data, df[:,features*lag]])
    return supervised_data
```

```
timeSteps = 2
supervised = to_supervised(scaled,lag=timeSteps)
#print(pd.DataFrame(supervised).head())


# splitting the data
# training on only first year data
features = demand.shape[1]
train_hours = 26250
X = supervised[:,:features*timeSteps]
y = supervised[:,features*timeSteps]

x_train = X[:train_hours,:]
x_test = X[train_hours:,:]
y_train = y[:train_hours]
y_test = y[train_hours:]

print (x_train.shape,x_test.shape,y_train.shape,y_test.shape)

#convert data to fit for lstm
#dimensions = (sample, timeSteps here it is 1, features )

x_train = x_train.reshape(x_train.shape[0], timeSteps, features)
x_test = x_test.reshape(x_test.shape[0], timeSteps, features)

print(x_train.shape,x_test.shape)

#define the model

model = Sequential()
model.add( LSTM( 50, input_shape = ( timeSteps,x_train.shape[2]) ) )
model.add( Dense(1, activation='tanh') )
model.compile(optimizer='adam',loss='mae')

history =  model.fit( x_train,y_train, validation_data = (x_test,y_test), epochs = 1000, batch_size
= 32, verbose = 0, shuffle = False)
#
#plt.plot(history.history['loss'], label='train')
#plt.plot(history.history['val_loss'], label='test')
#plt.legend()
#plt.yticks([])
#plt.xticks([])
#plt.title("loss during training")
#plt.show()

#scale back the prediction to orginal scale
y_pred = model.predict(x_test)
x_test = x_test.reshape(x_test.shape[0],x_test.shape[2]*x_test.shape[1])
inv_new = np.concatenate( (y_pred, x_test[:,-(features-1):] ) , axis =1)
inv_new= scaler.inverse_transform(inv_new)
final_pred = inv_new[:,0]
```

```
y_test = y_test.reshape( len(y_test), 1)
inv_new = np.concatenate( (y_test, x_test[:,-(features-1):] ) ,axis = 1)
inv_new = scaler.inverse_transform(inv_new)
actual_pred = inv_new[:,0]

#plot the prediction with actual data

plt.plot(final_pred, label = "prediction",c = "b")
plt.plot(actual_pred,label = "actual data",c="r")
#plt.xlim(0, 100)
#plt.ylim(0, 300)
#plt.yticks([])
#plt.xticks([])
plt.title("comparison between prediction and actual data")
plt.legend()


from sklearn.metrics import mean_squared_error
#
print('The rmse of prediction is:', mean_squared_error(final_pred,actual_pred) ** 0.5)
from sklearn.metrics import r2_score
# Compute error between our test predictions and the actual values.
                print(r2_score(actual_pred,final_pred,multioutput='variance_weighted'))
```