University of South Australia

INFS 5099 Data Science Professional Development

SP5 2022

Electricity Market in South Australia

Project Report

**Version:** 1

**Status:** Approved by all project team members

**Date:** 6 November 2022

**Group members (In alphabetical order):**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | UniSA Email ID | UniSA Student ID |
| 1 | Mahmoud Ghazi | ghamy023 | 110330165 |
| 2 | Mark Jeremy | jermy006 | 110336884 |
| 3 | Pang Man Leung | leupy032 | 110351943 |
| 4 | Rajveet Kaur Gandhi | ganry014 | 110356368 |

**Document Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Changes |
| 10-Sep-2022 | V0.1 | Pang | Initialize the document layout |
| 17-Sep-2022 | V0.2 | All group members | Revised section topics and layout |
| 27-Sep-2022 | V0.3 | Pang | Update Literature Review and Data Summary used in this project |
| 29-Sep-2022 | V0.4 | Ravjeet | Update Abstract and Limitation |
| 15-Oct-2022 | V0.5 | Mahmoud and Mark | Provide draft models information, analysis and forecasting for models |
| 23-Oct-2022 | V0.6 | Pang | Update Prediction Model, Prediction Results and Comparison |
| 24-Oct-2022 | V0.7 | Ravjeet | Update Further Improvement and Conclusion |
| 25-Oct-2022 | V0.8 | Pang | Update with ARMA and Lucheroni model and forecasting result. |
| 28-Oct-2022 | V0.9 | Ravjeet | Update with abstract, limitation and conclusion. |
| 6-Nov-2022 | V1 | All group members | Finalize the entire document |

**Approved Version**

|  |  |  |
| --- | --- | --- |
| Date | Version | Description |
| 5-Oct-2022 | V0.2 | The draft version with report layout approved by all group members |
| 4-Nov-2022 | V1 | Finalized version for submission |

**Associated Document**

|  |  |
| --- | --- |
| Name | Description |
| Meeting\_&\_Progress.xlsx | This document records every weekly meeting detail and individual weekly update. The document with full records will submit along with the Final Report. |
| Input\_Data\_1\_w\_forecast.xlsx | Dataset used by ARMA and Lucheroni w/wo GARCH model and forecasting result |
| Input\_Data\_2.xlsx | Dataset used by LSTM & Linear Regression model |

Table of Contents

[1. Abstract 5](#_Toc118556317)

[2. Introduction to the electricity market in South Australia 5](#_Toc118556318)

[2.1 Background 5](#_Toc118556319)

[2.2 Goals 7](#_Toc118556320)

[2.3 Objectives 8](#_Toc118556321)

[3. Data summary 10](#_Toc118556322)

[3.1 Aggregated price and demand data 10](#_Toc118556323)

[3.2 Mean maximum and minimum temperature of Adelaide 10](#_Toc118556324)

[3.3 Data transforming specific for ARMA, Lucheroni with and without GARCH model 12](#_Toc118556325)

[4. Literature Review 13](#_Toc118556326)

[4.1 Statistical Models 13](#_Toc118556327)

[4.2 Machine Learning Technique 14](#_Toc118556328)

[4.3 Models used in this project 15](#_Toc118556329)

[5. Methods 16](#_Toc118556330)

[5.1 ARMA Model 16](#_Toc118556331)

[5.2 ARMA + GARCH Model 17](#_Toc118556332)

[5.3 Lucheroni Model 18](#_Toc118556333)

[5.4 Lucheroni + GARCH Model 19](#_Toc118556334)

[5.5 Linear Regression Model 20](#_Toc118556335)

[5.6 LSTM model 20](#_Toc118556336)

[5.7 Training Data Fitting Comparison 21](#_Toc118556337)

[5.8 Other Finding 22](#_Toc118556338)

[6. Summary of Results and Findings 23](#_Toc118556339)

[6.1 ARMA Model 23](#_Toc118556340)

[6.2 ARMA + GARCH Model 24](#_Toc118556341)

[6.3 Lucheroni Model 24](#_Toc118556342)

[6.4 Lucheroni + GARCH Model 25](#_Toc118556343)

[6.5 Linear Regression Model 25](#_Toc118556344)

[6.6 LSTM Model 26](#_Toc118556345)

[6.7 Models Prediction Results Comparison 26](#_Toc118556346)

[7. Limitation and Further Improvement 28](#_Toc118556347)

[8. Conclusion 28](#_Toc118556348)

[9. Reference 29](#_Toc118556349)

[Appendix A: LSTM model Python source code 30](#_Toc118556350)

[Appendix B: Linear Regression model Python source code 33](#_Toc118556351)

# 1. Abstract

Energy is an essential commodity and needs to be continuously generated in order to meet the demand. It’s crucial for countries to monitor the supply and demand in order to meet the need of end users which are industry and households. In Australia, AEMO (Australian Energy Market Operator) has the responsibility to manage the electricity market and ensure that fair price is charged to the consumer. In this study, we have used data from AEMO and of Adelaide from Bureau of Meteorology, Australian Government, containing electricity prices, demand, temperature, and other variables for the year 2019 to 2021 and build the model to predict the price of electricity accurately using Auto Regressive Moving Average (ARMA), Lucheroni and long short-term memory networks (LSTM) and incorporating Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) with them and linear regression. All the models are compared based on the Mean Bias Error (MBE), Normalized Mean Bias Error (NMBE) and Normalized Root Mean Square Error (NRMSE) for training and test data to confirm the best model for predicting electricity price to most accuracy.

# 2. Introduction to the electricity market in South Australia

## 2.1 Background

AEMO (Australian Energy Market Operator) was founded by the Council of Australian Governments (COAG) on 1 July 2009 to operate the National Electricity Market (NEM) in the eastern and south-eastern states and Australian gas markets. Although, later AEMO’s duties have progressively grown and has included adding a variety of gas market functions and developing the market and independent power system operator for Western Australia from 2015. The ownership was and even now is shared between government and industry, with members representative are the federal and state governments and industry. AEMO is responsible for generation and production, distribution, retail and resources businesses across Australia. AEMO have always operated on a user-pays cost recovery basis. It recovers all operating costs through fees paid by industry participants.

AEMO operates the systems that allow energy to be generated, transmitted and distributed, and manages the financial markets that allow energy to be sold and bought. From a physical operations perspective, AEMO operate the electricity systems in the NEM and south-west Western Australia. This includes monitoring supply and demand, voltage and frequency, and managing planned and unplanned outages, and emergencies. In short, we help ensure that Australian consumers, businesses and industry always have access to secure and reliable energy.

Just like the stock exchange, the markets permit energy or other energy related services to be purchased and sold in a competitive environment. For instance, these markets can be used to allow electricity generators to sell their electricity to retailers, or for gas suppliers and distributors to schedule their deliveries for the day. AEMO lists the energy at the lowest available prices, settle trades, and makes sures that data and information flows between members. While the market mechanisms encourage affordable and reliable energy, AEMO does not lists or regulate retail energy prices. Instead, retailers decide the energy prices for Australian consumers.

Australia’s energy landscape is undergoing rapid and unprecedented levels of change. This change is being driven by an evolving power supply mix, ageing infrastructure, weather, changing technologies, consumer preferences and increasing interdependencies between our gas and electricity markets. We’re actively driving and planning for Australia’s energy future, making sure that Australians will continue to have access to secure and reliable energy in the years to come.

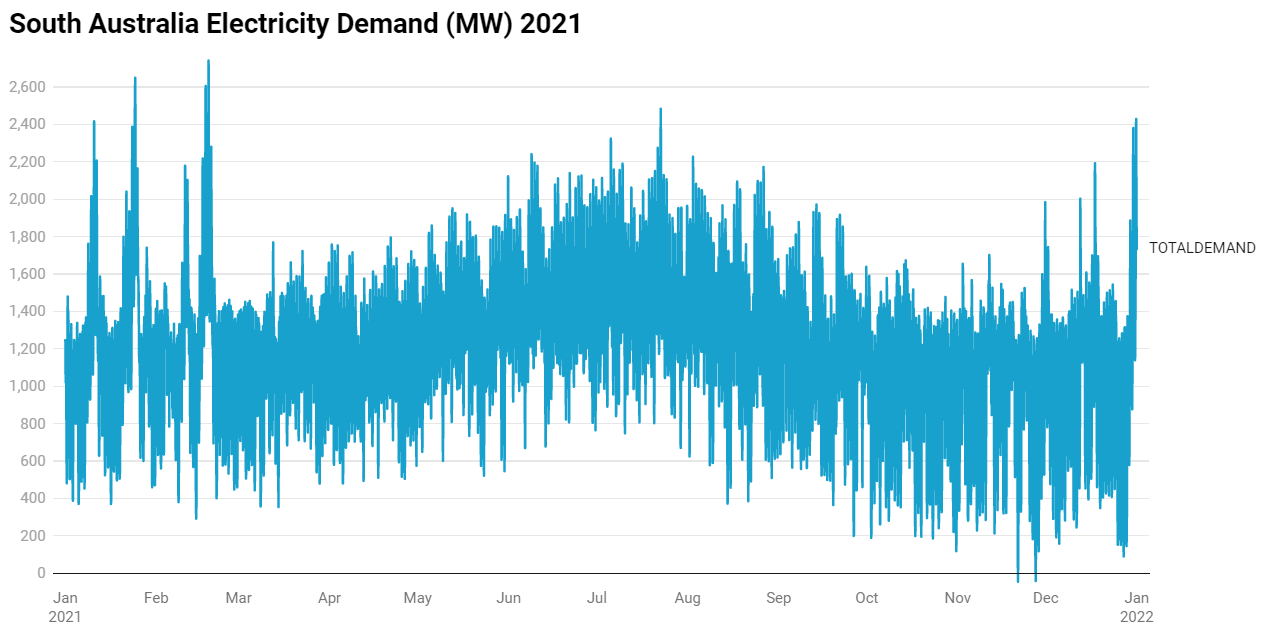
The Australian national electricity market comprises of five regions (South Australia, Melbourne, Tasmania, New South Wales, Queensland) with interconnectors (transmission links) which join the regions and regional reference nodes (RRNs) at the largest load centre of each region. The spot prices are calculated at each regional reference node, and it can be set by generation within the region or in another region.

The electricity moves around regions through interconnectors which connect neighbouring regions. Interconnectors deliver energy from lower price regions to higher price regions and would regularly equalise prices between regions. However, when interconnectors are operating at maximum capacity, electricity would still be transferred from a lower price region and sold in a higher price region, although spot prices would be different.

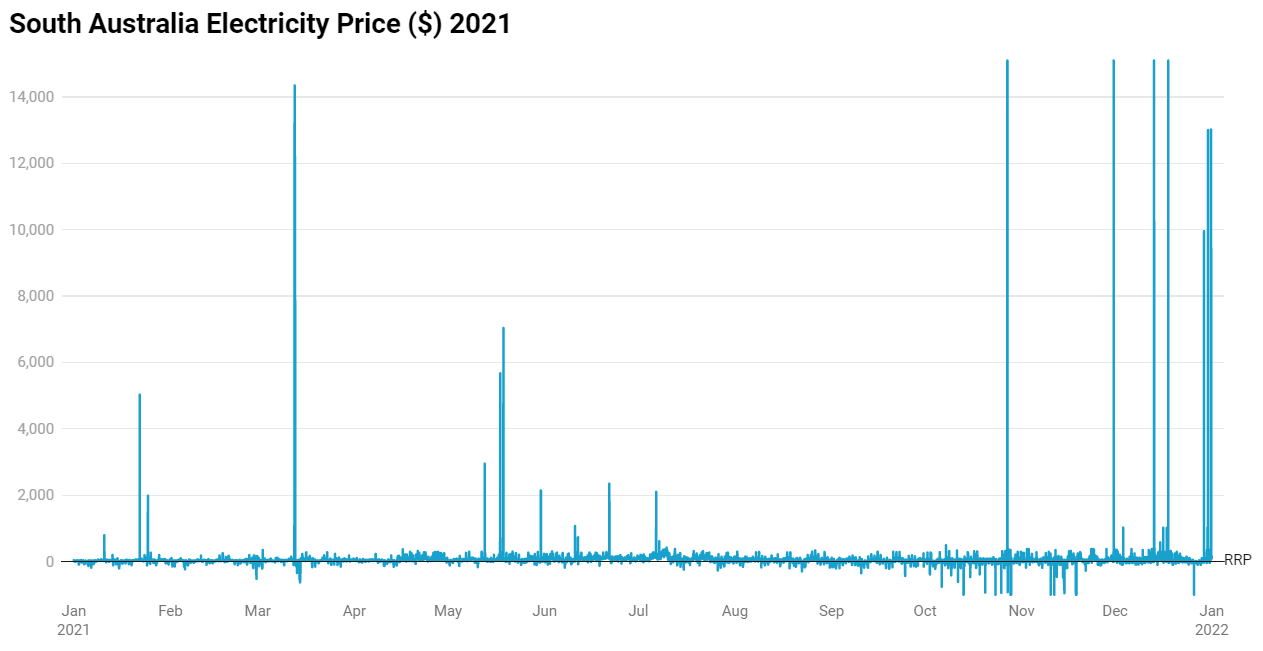
Interconnectors could be a partial alternative for local generation in a region to the point they can import electricity making them a substitute to increasing the capital stock of production within a region.

Earlier, large-scale electricity generation in South Australia was from non-renewable source like coal, diesel and gas. Recently, coal-fired power has completely exited the state, and 19 fresh wind farms and three new solar farms have entered, as well as two utility-scale batteries and two grid-connected emergency backup diesel generators.

The electricity demand in South Australia had high demand in February and March 2001 and the due to the summer months and the demand during the winter months were also high during July (figure 1). The electricity was also high price at multiple instances during the year 2021 (figure 2).



**Figure 1. South Australia Electricity demand (MW) in year 2021**



**Figure 2. South Australia Electricity Price in year 2021**

## 2.2 Goals

The major goals of this project are to better understanding the electricity market in South Australia, identifying how external factors affect the market price, classification of market status to catch for unusual opportunity and perform forecasting based on prediction models, so that to provide a mean to increase the profitability in the electricity market in South Australia.

The major improvement themes for this project listed as following:

**Major Improvement Themes**

* Discover knowledge about the electricity market
  + Improved understanding of the characteristics of electricity market in South Australia
  + Improved understanding of importance of external factors
* Save cost and increase profitability
* Capture unusual opportunity
  + Construct model that can be used to classify market status and identify abnormal situation ahead, to provides extra time for decision making, proper preparation to abnormal situation
* Better forecasting
  + Enable better resources planning and logistics management

By addressing the improvement themes listed above, can brings the following impact:

**Major Impact Themes**

* Better big picture of the market
* Better decision making based on knowledge acquired
* Better long-term planning
* Supports of decision making
* Quicker react to change of external factors
* Save cost by better resource and logistics management
* Save cost by avoiding vicious competition
* Increase profit by price optimalization

## 2.3 Objectives

This section listed the objectives of this project in order to address goals listed in section 2.2, where details of individual objectives in order to guiding the planning of this project.

**Discover knowledge specific to the electricity market in South Australia**

Study background knowledge of the electricity market in order to provide domain knowledge on the selected topic and solid foundation going through the project

**Explore existing models that applied for electricity market forecasting**

Study on existing models to identify strength and weakness of them,

**Explore existing forecasting models and identify its gaps**

Analyse existing models to identify gaps (problems or things that can be improved) or useful methods utilized for forecasting, as support to our model design. Explain the gaps, incidents and events to understanding the fluctuations in electricity price and demand.

**Develop models for classification and forecasting**

Find all the necessary inputs including the electricity prices, demand, temperature, and other variables that could affect the electricity price.

Models for classification and forecasting will be developed separately instead of integrated into all-in-one model. However, the classification results will affect the decision of choosing which model to be applied for forecasting.

* Spike classification: Study spikes to identify causing factors which influencing on price. Build a classification model to predict the chance of a spike at given conditions.
* Price forecasting: Develop Statistical, LSTM model to identify long-term trends and forecasting.

**Apply developed models for classification and forecasting**

Apply developed models to perform classification and forecasting using test dataset

**Evaluate new models and provide suggestion for future enhancement**

Explain results of our analysis and models. Compare existing models to our new models for performance comparison, identify problems and provide suggestion for further improve

# 3. Data summary

In this project, there are three datasets used which obtained from different data source. One is the aggregated price and demand data from Australian Energy Market Operator, another two are mean maximum and minimum temperature of Adelaide from Bureau of Meteorology, Australian Government.

These datasets were pre-processed and aggregated into the final dataset used to build our prediction models. Details of these datasets and pre-processing done will be covered in this section.

## 3.1 Aggregated price and demand data

The aggregated price and demand data (Australian Energy Market Operator 2022) for South Australia for the period of 1 January 2018 to 24 August 2022 were used in this project. A web scraper is employed to download monthly data from the Australian Energy Market Operator. The data interval for this dataset is 30 minutes for those data before September 2021, 5 minutes afterward.

**Data fields used**

There are 3 data fields in the raw data of this dataset used for the project as listed below:

**SETTLEMENTDATE** : Datetime label for the entry

**TOTALDEMAND** : Real value of demand in MWh

**RRP** : Real value of recommended retail price

**Data pre-processing**

Unwanted data fields were removed from the dataset. Hourly total demand and recommended retail price were calculated by averaging values within hourly interval.

## 3.2 Mean maximum and minimum temperature of Adelaide

The mean maximum and minimum temperature come from two separated dataset (Bureau of Meteorology, Australian Government 2022). Daily temperature data from the North Adelaide weather station for the period 1 January 2018 to 24 August 2022.

**Data fields used**

There are 4 data fields in the raw data of Mean maximum temperature dataset used for this project as listed below:

**Year** : Integer Year label of the data entry

**Month**  : Integer Month label for the data entry

**Day** : Integer Day label for the data entry

**Maximum temperature (Degree C)** : Real vale for maximum temperature

There are 4 data fields in the raw data of Mean minimum temperature dataset used for this project as listed below:

**Year** : Integer Year label of the data entry

**Month**  : Integer Month label for the data entry

**Day** : Integer Day label for the data entry

**Minimum temperature (Degree C)** : Real vale for maximum temperature

**Data pre-processing and estimation**

Two datasets were first merged together using Year, Month and Day as index, and then Year, Month and Day merge together to form a new temporary working dataset with 3 data fields: Date, Max Temp, Min Temp.

Because the desire interval for temperature data is 1 hour, but only the daily maximum and minimum value available from the data source, we applied some technique to obtained the estimated hourly value.

A Fourier series model built by Professor John Boland, up to hourly precision, based on detailed temperature data was used incorporate with the maximum and minimum temperature from our data source. Below are the equations used for hourly values estimation at timestamp t.

**Raw daily average temperature** = (Max Temperature + Min Temperature) / 2

**Model daily average temperature** = Sum of all model hourly temperature / 24

**Raw to model ratio** = Model daily average temperature(t) / Raw day average temperature(t)

**Estimated hourly temperature** = Model hourly temperature(t) X Raw to model ratio(t)

The Estimated hourly temperature values were used in training the Long-short term memory (LSTM) model.

## 3.3 Data transforming specific for ARMA, Lucheroni with and without GARCH model

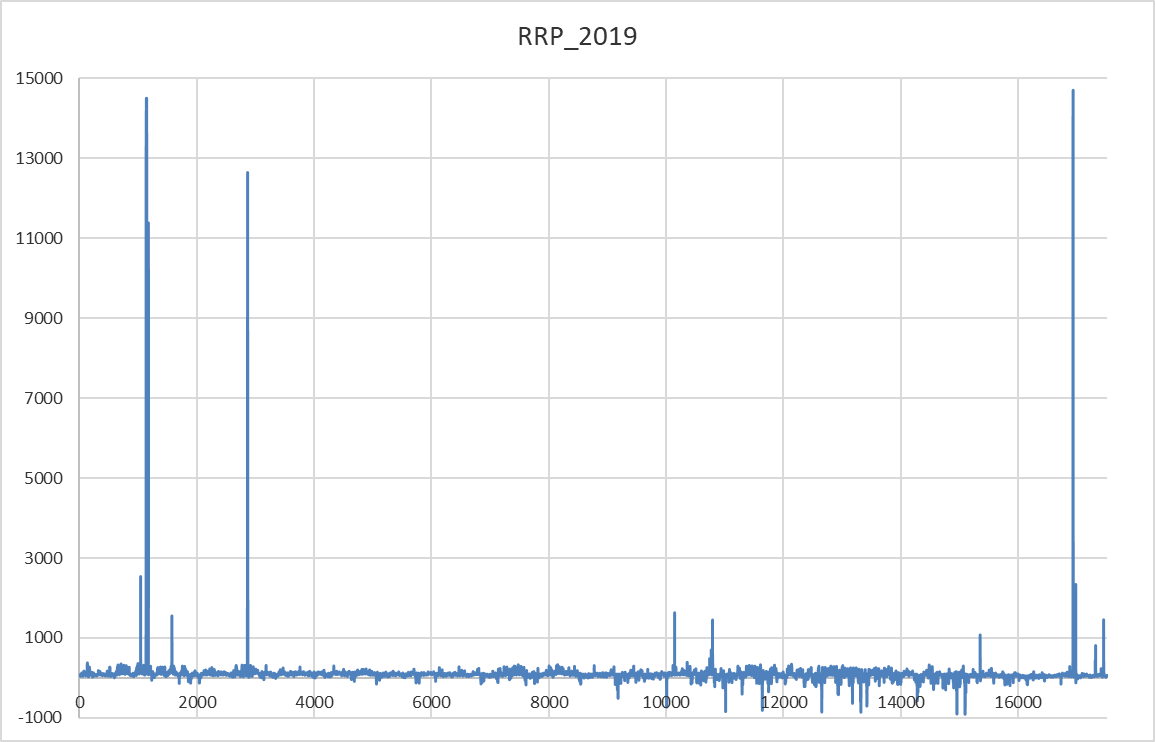
**Shift Operation**

After inspecting the aggregated price data, we discovered that there are negative values exists, the minimum value for year 2019 is -$907. We would like to have all positive values to train the model, therefore, we perform shift operation to all prices data by adding $2000 to it.

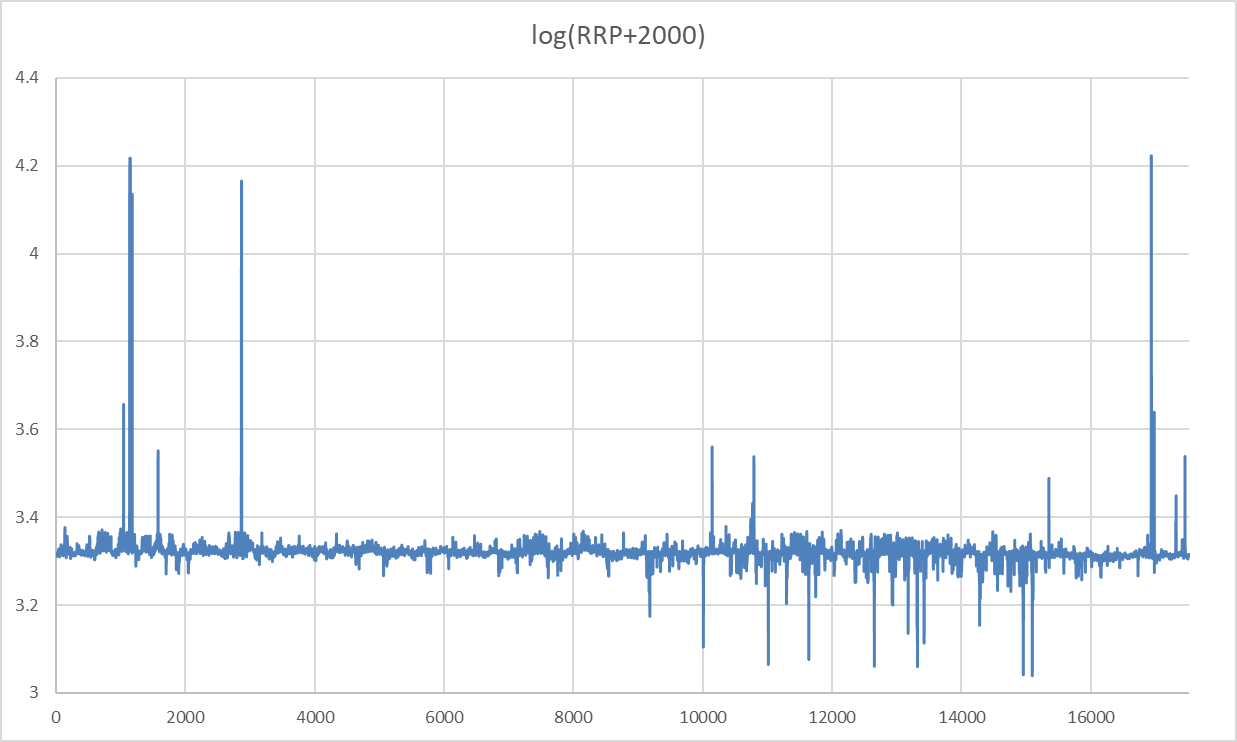
**Log Transformation**

Since the shifted price data have significant spikes in it, therefore, log transformation on price were done before it was used to train the ARMA, ARMA + GARCH, Lucheroni and Lucheroni + GARCH model.

Figure 3 and 4. shows the comparison of price data before and after shift operation and log transformation.



**Figure 3. Before shift operation and log transformation**



**Figure 4. After shift operation and log transformation**

# 4. Literature Review

Electricity price prediction always been a hot research area, many different methods and models have been designed and proposed to predict electricity price, from traditional statistical model to recently development of artificial intelligence technique, like artificial neural network. Some models from both traditional and machine learning approach will be introduced.

## 4.1 Statistical Models

Statistical forecasting models make use of statistics knowledge and apply to historical data in order to predict what could happen in the future. Many statistical forecasting models have been developed, some of them identified as useful for this project, a brief description for each will be provided in this section.

**ARMA model**

Autoregressive moving average (ARMA) models are frequently used to modelling time series data in different areas. It combined the autoregressive (AR) and moving average (MA) model together. In ARMA model, previous lags and the residuals are taken into consideration for predicting the future values of the time series. It has proven to be a model can predict very well in most situations. ARMA have been successfully applied to different problems with good results, like predicting financial time series, GDP, housing demand, electric loads (Petridis, V & Kehagias, A 1998) and so many others.

**Lucheroni model**

Lucheroni model has been applied to the electricity market and delivered very well performance (Lucheroni, C. 2007), (Lucheroni, C. 2009). It will be included in this project and compares to other models.

**GARCH model**

Generalized autoregressive conditional heteroskedasticity (GARCH) model was developed by Dr. Tim Bollerslev in 1986, which is a generalization of the ARCH process where the variance equation is replaced (Brockwell, PJ & Davis, RA 2016). GARCH model can be used to analysis variance in time series data to predict the volatility. It assumes the error term follows an autoregressive moving process. It is very useful when the variance of error is not constant in the time series. From the preliminary analysis of the electricity market data in South Australia, high volatility and variance of error have been spotted, the project team believed introducing the GARCH model might improve the accuracy of price forecasting.

## 4.2 Machine Learning Technique

**Recurrent Neural Network (RNN)**

RNN is a generalized version of the basic feedforward neural network, with the memory. As the term recurrent suggested, every output of the RNN will send back as input to the RNN for computation. Unlike other form of neural network that inputs are independent to each other, inputs are related to each other in the RNN.

RNN can be very efficient and effective when modelling data in sequence and where each data is dependent on the previous one data, and it exactly matches the nature of our problem. However, gradient will vanish in RNN and it may case some serious issues.

**Long Short-Term Memory (LSTM)**

LSTM is a modified version of the RNN, it allows to store past data in the memory to address the problem of vanishing gradient issues happen in RNN. In LSTM, an input gate is used to determine what input value should be used to adjust values in memory. Forget gate used to determine which part of the memory will be discard and the output gate will determine the output value by using values from input as well as memory. Due to the memory feature of LSTM, make it easier to train comparing to RNN model.

**Ensemble model**

In real-life situation, there are always challenges when selecting the appropriate model to do forecasting. One model may be very sensitive to volatility in the input data, while some might predict well for low variance, or some might rely heavily on limited features in the dataset. Because of such unique characteristics, they might perform good in some situation, but not always.

An ensemble model is designed to address these issues by combines two or more individual models into one. Prediction was made by individual models, and each individual prediction results aggregated into a single result (Woodward, WA et al., 2022, pp. 487).

## 4.3 Models used in this project

From the background information given in the previous sections, shows that different methods have its own advantages and disadvantages, and there are numbers of available methods, from both traditional methods as well as contemporary machine learning techniques been considered for this project.

From the traditional approach, linear regression method is the simplest method we choose, in order to provide a baseline reference to other method. ARMA model combining the advantages from both AR and MA model, which has been widely used in different areas and always provides good result, that why it has been selected. The electricity price in the South Australia is dynamic, inference by different factors, including supply, demand, weather, pricing rule and strategy adopted by different electricity providers, electricity price could run into extreme in some certain situation. By comparing to ARMA and some other models, Lucheroni model is much more sensitive to spikes, and it is a good idea to included it as well.

The GARCH model also supplemented to ARMA and Lucheroni, hopefully it can model the variance in the dataset, and improve the accuracy.

On the other hand, the advantage of LSTM comparing to traditional neural network architecture like feedforward neural network and convolutional neural network is past memory of useful information in the neural network will not be discarded immediately, by retained in the network and gradually fades out as time passed. This is a very good feature when considering time-series forecasting, as we want the past to be taken into consideration and fades gradually. Although ensemble model may harness advantages from different models and more likely to deliver better prediction over single model, however, it will far beyond the capacity of this project group, therefore, only a LSTM model will be implemented.

Summary in short, the project team decided to adopt methods from both traditional and machine learning approach. They are Linear regression, ARMA, ARMA + GARCH, Lucherni, Lucheroni + GARCH and LSTM.

# 5. Methods

This section provides the details on how all models were built and trained.

## 5.1 ARMA Model

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The equation for an ARMA(p,q) process (Brockwell, PJ & Davis, RA 2016, pp. 73) shown below, has been used to calculate forecasting value.  The coefficients estimated by determine using autocorrelation function (ACF) and partial autocorrelation function (PACF), and obtained the results listed in Figure 5. | |  |  | | --- | --- | | **Type** | **Coefficient** | | AR 1 | 0.82854 | | MA 1 | -0.04749 | | Constant | 0.569241 |   **Figure 5. Coefficient for ARMA model** |

Figure 6. shows the ARMA model fitting with year 2019 training data. Where Figure 7 shows the error measures.

**Figure 6. 2019 Actual RRP vs AMRA Model RRP (log scale)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MBE** | **NMBE** | **NRMSE** |
| ARMA | 3.8585 | 0.0390 | 2.3446 |

**Figure 7. Error measures for ARMA model fitting with training data**

## 5.2 ARMA + GARCH Model

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| On top of the ARMA model with same coefficient specify in section 5.1, a GARCH model has been applied. The equation for an GARCH model (Brockwell, PJ & Davis, RA 2016, pp. 196) shown below.  with | |  |  | | --- | --- | | **Type** | **Coefficient** | | AR 1 | 0.8242 | | AR 2 | -0.0678 | | MA 1 | 0.5593 | | Constant | 0.000074 |   **Figure 8. Coefficient for GARCH model** |

The coefficients estimated by determine using autocorrelation function (ACF) and partial autocorrelation function (PACF), and obtained the results listed in Figure 8. Squared residuals of the ARMA then taken as input for the GARCH model.

Figure 9. shows the ARMA model fitting with year 2019 training data. Where Figure 10 shows the error measures.

**Figure 9. 2019 Actual RRP vs AMRA + GARCH Model RRP (log scale)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MBE** | **NMBE** | **NRMSE** |
| ARMA + GARCH | 1.6286 | 0.0165 | 2.2813 |

**Figure 10. Error measures for ARMA + GARCH model fitting with training data**

## 5.3 Lucheroni Model

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The equation for an Lucheroni model shown below, has been used to calculate forecasting values  where and are noise terms, is the time step.  (Lucheroni, C. 2007, pp. 6), (Lucheroni, C. 2009, pp. 2), (Lucheroni, C 2010, pp. 83)  Estimated coefficients for the Lucheroni model listed in Figure 11. | |  |  | | --- | --- | | **Type** | **Coefficient** | | Delta | 1 | | Kappa | 0.000959306 | | Lambda | 0.000477681 | | Gamma | -0.00430681 | | a | 0 | | b | 0 | | j | 0 | | Epsilon | 0.15 |   **Figure 11. Coefficient for Lucheroni model** |

Figure 12. shows the ARMA model fitting with year 2019 training data. Where Figure 13 shows the error measures.

**Figure 12. 2019 Actual RRP vs Lucheroni Model RRP (log scale)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MBE** | **NMBE** | **NRMSE** |
| Lucheroni | 2.7571 | 0.0279 | 2.2190 |

**Figure 13. Error measures for Lucheroni model fitting with training data**

## 5.4 Lucheroni + GARCH Model

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| On top of the Lucheroni model with same coefficient specify in section 5.3, a GARCH model has been applied. The equation for an GARCH model same as shown in section 5.2 with estimated coefficients as listed in Figure 14. | |  |  | | --- | --- | | **Type** | **Coefficient** | | AR 1 | 0.7883 | | AR 2 | -0.0551 | | MA 1 | 0.5581 |   **Figure 14. Coefficient for GARCH model** |

Figure 15. shows the ARMA model fitting with year 2019 training data. Where Figure 16 shows the error measures. Squared residual of the Lucheroni then taken as input for the GARCH model.

**Figure 15. 2019 Actual RRP vs Lucheroni + GARCH Model RRP (log scale)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MBE** | **NMBE** | **NRMSE** |
| Lucheroni + GARCH | 1.5443 | 0.0156 | 2.1981 |

**Figure 16. Error measures for Lucheroni + GARCH model fitting with training data**

## 5.5 Linear Regression Model

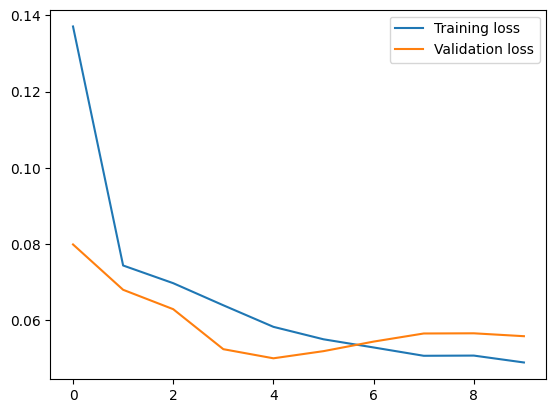
A simple linear regression model has been built using Python also. The simple linear regression model expected to have a characteristic to generate some of mean values and do not expect to perform very well as we know there are huge spikes in the training data. However, the team decide that having the linear regression for reference will still to be a good idea. Error measures given in Figure 18.

## 5.6 LSTM model

**Building and training the model**

Data have been normalized to become the final input for the LSTM model. Three variables: Total Demand, RRP and Temperature were used during training.

The model was tuned to use 14 days of past data to predict future values.



**Figure 17. LSTM model training and validation loss**

Figure 17. shows the training loss and validation loss for the LSTM model during training process. As you can see that the training and validation loss decreased and approaching to zero from cycle to cycle, all both training and validation loss converge to each other, which is a good sign that shows the model getting better in each cycle. Error measures for LSTM fitting training data given in Figure 18.

Complete Phyton source code used to build, train and perform forecasting of the model, please reference to Appendix A.

## 5.7 Training Data Fitting Comparison

This section compares the error measures of 6 different models with training data, in order to find the ranking among them in terms of accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MBE** | **NMBE** | **NRMSE** |
| ARMA | 3.8585 | 0.0390 | 2.3446 |
| ARMA + GARCH | 1.6286 | 0.0165 | 2.2813 |
| Lucheroni | 2.7571 | 0.0279 | 2.2190 |
| Lucheroni + GARCH | 1.5443 | 0.0156 | 2.1981 |
| LSTM | -6.24 | -0.078 | 0.181 |
| Linear Regression | 6.81 | 0.068 | 0.72 |

**Figure 18. Error measures of 6 models**

Figure 18. shows error measures for 6 different models. According to their MBE, their accuracy or performance can be ranked as below:

1. Lucheroni + GARCH

2. ARMA + GARCH

3. Lucheroni

4. ARMA

5. LSTM and Linear Regression

From the Figure 6, 9, 12 and 15, we can see that, all ARMA, Lucheroni with and without GARCH fitting to training data well when price was in normal range and when spikes happen.

More specific, the Lucheroni model especially with the GARCH variance added has a better/lower error measure compared to the other models and also fit the spikes to a higher precision. Also, the Lucheroni+ GARCH model fits the data very closely despite the drastic spikes when tried to fit the model in both yearly and quarterly data. The quarterly data has varying amounts of spikes across each quarter. Even in the quarter with no spikes the L+G model has a good fit.

LSTM and Linear Regression model seems to be outperformed by other 4 models.

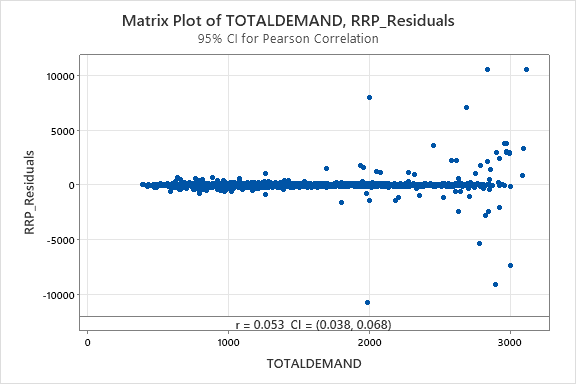
However, the prediction power of ARMA, ARMA + GARCH, Lucheroni and Lucheroni + GARCH are limited to only a few time steps ahead. Beyond, approximately 5 time-steps, the forecasts tend to flatten over the mean value of the model.

## 5.8 Other Finding

**Correlation with Total Demand**

After fitting the Lucheroni + GARCH model, we compare the residuals with the Total Demand variable from the dataset to check if there is any influence of demand on the electricity pricing.

Figure 19. is the correlation plot of Total Demand and RRP Residuals, it shows a mostly flat line, implying no correlation with an r value of 0.053 at 95% confidence.



**Figure 19. Matrix plot of Total Demand and RRP Residuals**

# 6. Summary of Results and Findings

After built and trained 6 models individually, they were used to perform one year prediction. The prediction results then compared with year 2021 actual data which are unseen to the model, to obtain error measure for model comparison and evaluation.

49 time-steps prediction covers the period 29-30 September 2021 done for ARMA, ARMA + GARCH, Lucheroni and Lucheroni + GARCH models, linear regression model and the LSTM model.

The Mean Bias Error (MBE), Normalized Mean Bias Error (NMBE) and Normalized Root Mean Square Error (NRMSE) have been used to measure the error. Please note that the error terms were calculated using log value of actual and predicted values. Where having a smaller error value represents higher accuracy, and vice versa.

Section 6.1 to 6.6 shows models forecasted RRP fits with actual RRP, as well as error measures for all models we developed.

## 6.1 ARMA Model

**Figure 20. 49 Time steps forecasting by ARMA model**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MBE** | **NMBE** | **NRMSE** |
| ARMA | -4.832 | -0.0895 | 3.7012 |

**Figure 21. Error measures for ARMA model prediction**

## 6.2 ARMA + GARCH Model

**Figure 22. 49 Time steps forecasting by ARMA + GARCH model**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MBE** | **NMBE** | **NRMSE** |
| ARMA + GARCH | -7.0189 | -0.1299 | 3.9436 |

**Figure 23. Error measures for ARMA + GARCH model prediction**

## 6.3 Lucheroni Model

**Figure 24. 49 Time steps forecasting by Lucheroni model**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MBE** | **NMBE** | **NRMSE** |
| Lucheroni | -1.8844 | -0.0349 | 3.9377 |

**Figure 25. Error measures for Lucheroni model prediction**

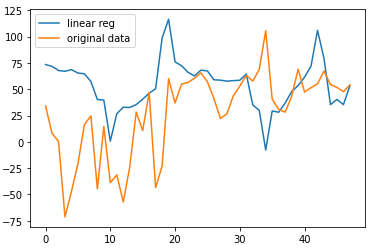
## 6.4 Lucheroni + GARCH Model

**Figure 26. 49 Time steps forecasting by Lucheroni + GARCH model**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MBE** | **NMBE** | **NRMSE** |
| Lucheroni + GARCH | -8.0415 | -0.1489 | 5.6708 |

**Figure 27. Error measures for Lucheroni + GARCH model prediction**

## 6.5 Linear Regression Model

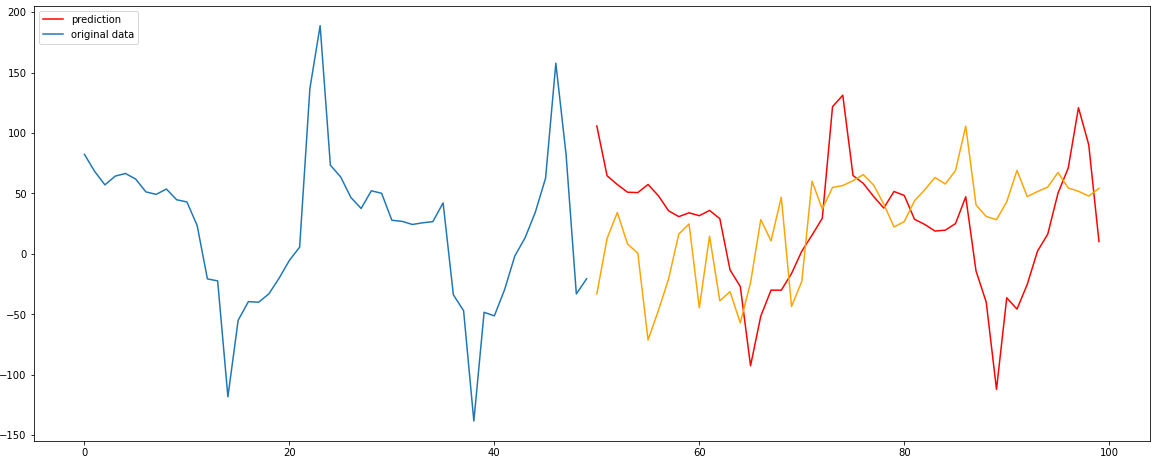


**Figure 28. Forecasting by linear regression model**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MBE** | **NMBE** | **NRMSE** |
| Linear regression | -37.74 | -2.286 | 3.22 |

**Figure 29. Error measures for linear regression model prediction**

## 6.6 LSTM Model



**Figure 30. LSTM model forecasting**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MBE** | **NMBE** | **NRMSE** |
| LSTM | 5.0003 | 0.201 | 2.64 |

**Figure 31. Error measures for LSTM model prediction**

## 6.7 Models Prediction Results Comparison

In this section, we will discuss the prediction results for all models and compares their performance using error metrics.

**Models’ Error Metrics**

This section compares the error measures of five different models, in order to find the ranking among them in terms of accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MBE** | **NMBE** | **NRMSE** |
| ARMA | -4.832 | -0.0895 | 3.7012 |
| ARMA + GARCH | -7.0189 | -0.1299 | 3.9436 |
| Lucheroni | -1.8844 | -0.0349 | 3.9377 |
| Lucheroni + GARCH | -8.0415 | -0.1489 | 5.6708 |
| LSTM | 5.0003 | 0.201 | 2.64 |
| Linear Regression | -37.74 | -2.286 | 3.22 |

**Figure 32. Error measures of 5 models**

Figure 32. shows error measures for 6 different models. According to their NMBE, their accuracy or performance can be ranked as below:

1. Lucheroni

2. ARMA

3. ARMA + GARCH

4. Lucheroni + GARCH

5. LSTM

6. Linear Regression

**Discussion On Results**

Figure 20 and 22 shows the fitting of ARMA and ARMA + GARCH models. We can see from that the prediction for both models begins to flatten over time. Error of ARAM + GARCH model is larger than ARMA only. The ARMA and ARMA + GARCH model fitting the training data very well, but the prediction did not catch up too much volatility in the actual data.

The prediction result for Lucheroni and Lucheroni + GARCH shows in Figure 24 and 26. Both predictions show there are two huge spikes coming, but there are no such huge spikes in the real data.

From the error measures shown in Figures 21, 23, 25 and 27, we can see that both ARMA and Lucheroni perform better than the version incorporated with GARCH model.

By comparing the NMBE for different models, Lucheroni has the best accuracy, ARMA comes to second place.

Another thing to note is, the ARAM + GARCH and Lucheroni + GARCH perform better for the training data set over ARMA and Lucheroni alone, as shown in Figures 7, 10, 13 and 16. However, for the forecasting result, models without GARCH perform better than models with GARCH. It may be due to the actual data for the forecasting period being quite flat with no significant spike in it, on the other hand, GARCH models fit better for spikes as shown in Figures 6, 9, 12, and 15 for training data, that’s why models with GARCH have a larger error in the given in the forecasting results. Models with GARCH might provide a better result when spikes are in real data, but further testing is required to confirm this hypothesis.

LSTM and Linear Regression models perform the least accuracy in both training and forecasting compared to the other 4 models.

# 7. Limitation and Further Improvement

* The study was limited to the South Australia Electricity Market and the for the period of 2019 to 2021
* The study considered only few input parameters for building the model. However, there are other factors that affect the demand and price.
* The models could be further enhanced by use of more input variable.
* The training dataset contained more spikes compared to test dataset and some models were good at spike prediction, and the models performed differently.
* All models were tested on same time frame to understand the performance. However, different models perform differently in predicting spikes and further study is required to confirm it.

# 8. Conclusion

Overall, in this study we used both traditional and machine learning models to predict the electricity price of south Australia. Models performed differently on the test and training datasets. The models that performed well on the training dataset could not perform well on the test dataset for example the Lucheroni + GARCH model performed well during the training and had the least error 1.5443. However, the Lucheroni + GARCH model could not perform well on the test dataset and had -8.0415 MBE. It’s observed that the GARCH models fit better for spikes, but the test data didn’t have many spikes compared to the training dataset.

On the other hand, we saw that for the training dataset the model Lucheroni outperformed the rest of the models with MBE of -1.8844, Also, it fits well with higher precision for 49-time steps ahead and has the best performance in predicting the price based on demand.

It was also observed that model LSTM and Linear Regression could not perform well with during training and testing compared to the other models and had the least accuracy.

# 9. Reference

Ampatishan Sivalingma 2020, *Why do we need LSTM*, Towards Data Science 2018, viewed 1 October 2022, <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

Australian Energy Market Operator 2022, *Aggregated Price and Demand Data*, Aggregated price and demand data, AEMO Aggregated price and demand data, viewed 1 September 2022, <https://aemo.com.au/en/energy-systems/electricity/national-electricity-market-nem/data-nem/aggregated-data>

Bureau of Meteorology, Australian Government 2022, *Climate Data Online,* Climate Data Online – Map Search, viewed 1 September 2022, <http://www.bom.gov.au/climate/data/>

Brockwell, PJ & Davis, RA 2016, *Introduction to Time Series and Forecasting*, Springer International Publishing AG, Cham.

Devopedia. 2019, *Long Short-Term Memory*, viewed 3 October 2022, <https://devopedia.org/long-short-term-memory>

Huang, J & Boland, J 2018, 'Performance Analysis for One-Step-Ahead Forecasting of Hybrid Solar and Wind Energy on Short Time Scales'*,* *Energies*,vol*.* 11, 05/02, p. 1119.

Lucheroni, C. 2007, *Resonating models for the electric power market*, Physical Review E, vol.76, no.5, p. 9831.

Lucheroni, C. 2009, *A Resonating Model for the Power Market and its Calibration*, SSRN 2011, viewed 1 October 2022, <http://ssrn.com/abstract=1850469>

Lucheroni, C 2010, ‘Stochastic models of resonating markets’, *Journal of Economic Interaction and Coordination*, vol. 5, no. 1, pp. 77–88.

Michael Phi 2018, *Illustrated Guide to LSTM’s and GRU’s: A step by step explanation*, Towards Data Science 2018, viewed 1 October 2022, <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

Petridis, V & Kehagias, A 1998, *Predictive modular neural networks : applications to time series / Vassilios Petridis, Athanasios Kehagias*., 1st ed. 1998., Springer : Kluwer Academic Publishers, New York, New York.

University of South Australia (UniSA) 2022, Advanced Analytic Techniques 1 MATH 5045, *University of South Australia*, viewed 1 Oct 2022, <https://lo.unisa.edu.au/mod/resource/view.php?id=2924419>

Woodward, WA, Sadler, BP & Robertson, S 2022, *Time Series for Data Science : Analysis and Forecasting / Wayne A. Woodward, Bivin P. Sadler and Stephen D. Robertson*., First edition., CRC Press LLC, Boca Raton, Florida ; Milton.

# Appendix A: LSTM model Python source code

The complete Python source code used to build and train the LSTM model, as well as prediction shown below.

|  |
| --- |
| import numpy as np  from tensorflow.keras.models import Sequential, load\_model  from tensorflow.keras.layers import LSTM  from tensorflow.keras.layers import Dense, Dropout  import pandas as pd  from matplotlib import pyplot as plt  from sklearn.preprocessing import StandardScaler  import seaborn as sns  *#from datetime import datetime*  *#Read the csv file*  df = pd.read\_csv("price\_temp\_total.csv")  print(df.head())  train\_dates = pd.to\_datetime(df['SETTLEMENTDATE'])  *#Variables for training*  cols = list(df)[2:5]  *#Date is not used in training.*  print(cols)  *#New dataframe with only training data - 3 columns*  df\_for\_training = df[cols].astype(float)  print(df\_for\_training.head)  *#LSTM uses sigmoid and tanh that are sensitive to magnitude so values need to be normalized*  *# normalize the dataset*  scaler = StandardScaler()  scaler = scaler.fit(df\_for\_training)  df\_for\_training\_scaled = scaler.transform(df\_for\_training)  print(df\_for\_training.shape)  *# print(df\_for\_training\_scaled.shape)*  *#As required for LSTM networks, we require to reshape an input data into n\_samples x timesteps x n\_features.*  *#In this example, the n\_features is 5. We will make timesteps = 14 (past days data used for training).*  trainX = []  trainY = []  n\_future = 1 *# Number of days we want to look into the future based on the past days.*  n\_past = 14 *# Number of past days we want to use to predict the future.*  *#Reformat input data into a shape: (n\_samples x timesteps x n\_features)*  for i in range(n\_past, len(df\_for\_training\_scaled) - n\_future +1):  trainX.append(df\_for\_training\_scaled[i - n\_past:i, 0:df\_for\_training.shape[1]])  trainY.append(df\_for\_training\_scaled[i + n\_future - 1:i + n\_future, 1]) # 1 mean column of RRP    *#In my case, trainX1 has a shape (40690 , 14, 3).*  *#40690 because we are looking back 14 days (40704 - 14 = 40690).*  *#Remember that we cannot look back 14 days until we get to the 15th day.*  *#To predict more days in future, we need all the 3 variables which we do not have.*  *#We need to predict all variables if we want to do that.*  trainX, trainY = np.array(trainX), np.array(trainY)  print('trainX shape == {}.'.format(trainX.shape))  print('trainY shape == {}.'.format(trainY.shape))  *# define the model*  model = Sequential()  model.add(LSTM(64, activation='relu', input\_shape=(trainX.shape[1], trainX.shape[2]), return\_sequences=True))  model.add(LSTM(32, activation='relu', return\_sequences=False))  model.add(Dropout(0.2))  model.add(Dense(trainY.shape[1]))  model.compile(optimizer='adam', loss='mse')  model.summary()  *# fit the model*  history = model.fit(trainX, trainY, epochs=10, batch\_size=16, validation\_split=0.1, verbose=1)  plt.plot(history.history['loss'], label='Training loss')  plt.plot(history.history['val\_loss'], label='Validation loss')  plt.legend()  model.save('model\_2')  model = load\_model('model')  *#Predicting seen data*  *#Remember that we can only predict one day in future as our model needs 5 variables*  n\_past = 16  n\_days\_for\_prediction=100 #let us predict past 15 days  *# predict\_period\_dates = pd.date\_range(list(train\_dates)[-n\_past], periods=n\_days\_for\_prediction, freq=us\_bd).tolist()*  *# print(predict\_period\_dates)*  *#Make prediction*  prediction = model.predict(trainX[:100]) #shape = (n, n\_past\_from\_training, n\_of\_columns) where n is the n\_days\_for\_prediction  print(prediction.shape)  *#Since we used 5 variables for transform, the inverse expects same dimensions*  *#Therefore, let us copy our values 3 times to make shape similar to original data to inverse transform.*  *#Perform inverse transformation to rescale back to original range*  prediction\_copies = np.repeat(prediction, df\_for\_training.shape[1], axis=-1)  y\_pred = scaler.inverse\_transform(prediction\_copies)[:,1] # RRP  prediction\_copies = np.repeat(trainY[:100], df\_for\_training.shape[1], axis=-1)  y\_inverse = scaler.inverse\_transform(prediction\_copies)[:,1]# RRP  plt.plot(y\_pred,c='red',label="Predict")  plt.plot(y\_inverse,label="Price")  plt.legend(loc="upper left")  plt.xlabel("days")  plt.ylabel("Price")  plt.show()  Error = y\_pred - y\_inverse  Err\_2 = np.power(Error, 2)  MBE = np.mean(Error)  print(f'MBE is {MBE}')  NMBE = MBE/ np.mean(y\_inverse)  print(f'NMBE is {NMBE}')  Err\_2 = np.power(Error, 2)  print(f'R-square {sum(Err\_2)}')  NRMSE = np.sqrt(np.mean(Err\_2))/np.mean(y\_inverse)  print(f'NRMSE is {NRMSE}')  *#Predicting Unseen data*  *#Remember that we can only predict one day in future as our model needs 5 variables*  n\_past = 16  n\_days\_for\_prediction=100 #let us predict past 15 days  *# predict\_period\_dates = pd.date\_range(list(train\_dates)[-n\_past], periods=n\_days\_for\_prediction, freq=us\_bd).tolist()*  *# print(predict\_period\_dates)*  *# testX, testY = np.array(trainX[-100:]), np.array(trainY[-100:])*  *#Make prediction*  prediction = model.predict(trainX[-100:-50]) #shape = (n, n\_past\_from\_training, n\_of\_columns) where n is the n\_days\_for\_prediction  print(prediction.shape)  prediction\_copies = np.repeat(prediction, df\_for\_training.shape[1], axis=-1)  prediction\_copies = scaler.inverse\_transform(prediction\_copies)[:,1]  *# unseen\_data = df\_for\_training.iloc[-100:-50,0].values*  *# unseen\_data = df\_for\_training.iloc[-200:-100,0].values*  *# plt.plot(y\_pred\_future,c='red')*  *# plt.plot(unseen\_data)*  *# plt.show()*  prediction\_copies.shape  *# prediction\_copies*  predict = pd.DataFrame(data=prediction\_copies, columns=['pred'])  original = df[['SETTLEMENTDATE', 'RRP']]  original['SETTLEMENTDATE']=pd.to\_datetime(original['SETTLEMENTDATE'])  predict["date"] = original.iloc[-50:,]['SETTLEMENTDATE'].values  original\_2 = original.iloc[-200:-50,]  *# original\_2*  sns.lineplot(original\_2,x=range(0,150), y='RRP')  sns.lineplot(predict,x=range(150,200), y='pred') |

# Appendix B: Linear Regression model Python source code

The complete Python source code used to build and train the Linear Regression model, as well as prediction shown below.

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn import preprocessing;  from sklearn.model\_selection import train\_test\_split;  from sklearn import linear\_model;  import matplotlib.pyplot as plt  from sklearn.preprocessing import PolynomialFeatures  from sklearn.preprocessing import StandardScaler  prices\_df = pd.read\_csv('price\_temp\_total.csv')  price = pd.DataFrame(prices\_df.RRP)  # price.plot()  # price  test = price[-100:]  # test  forecast\_out = 100  test\_size = 0.2;  df = price[:-100]  label = df.shift(-forecast\_out);  X = np.array(df);  # X = preprocessing.scale(X)  scaler = StandardScaler()  scaler = scaler.fit(X)  X = scaler.transform(X)  # label = preprocessing.scale(label)  #plt.plot(X)  X\_lately = X[-forecast\_out:]  X = X[:-forecast\_out]  label.dropna(inplace=True)  y = np.array(label)  X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, y, test\_size=test\_size)  plt.scatter(X\_train, Y\_train)  plt.show()  # Linear regression  learner = linear\_model.LinearRegression()  learner.fit(X\_train,Y\_train);  score=learner.score(X\_test,Y\_test)  forecast = learner.predict(X\_lately)  response = {}  response['test\_score'] = score  response['forecast\_set'] = forecast  # print(response);  # forecast.shape  # X\_first = scaler.inverse\_transform(X\_lately)  plt.plot(forecast, label='linear reg')  plt.plot(test.values, label='original data')  plt.legend()  # test  # Linear regression ploynomial  #################  poly = PolynomialFeatures(degree=4)  X\_poly = poly.fit\_transform(X\_train)  poly.fit(X\_poly, Y\_train)  linear2= linear\_model.LinearRegression()  linear2.fit(X\_poly,Y\_train)  X\_lately\_ = poly.fit\_transform(X\_lately)  forecast\_ = linear2.predict(X\_lately\_)  # forecast.shape  # X\_first = scaler.inverse\_transform(X\_lately)  plt.plot(forecast\_, label='linear reg')  plt.plot(test.values, label='original data')  plt.legend()  # test  X\_first = scaler.inverse\_transform(X\_lately)  # X\_La  Error = X\_first - forecast\_  Err\_2 = np.power(Error, 2)  print(f'R-square {sum(Err\_2)}')  MBE = np.mean(Error)  print(f'MBE is {MBE}')  NMBE = MBE/ np.mean(X\_first)  print(f'NMBE is {NMBE}')  Err\_2 = np.power(Error, 2)  print(f'R-square {sum(Err\_2)}')  NRMSE = np.sqrt(np.mean(Err\_2))/np.mean(X\_first)  print(f'NRMSE is {NRMSE}') |