

Capstone Project by Group F

Explaining XGBoost versus LSTM models in day-ahead electricity demand forecasting with XAI

Christian Behan (z3376823)

Mia Jensen (z3460140)

Daniel Karp (z5338909)

Kim Dien Nguyen (Kevin) (z3318038)

School of Mathematics and Statistics

UNSW Sydney

April 2023

Submitted in partial fulfilment of the requirements of the Capstone Course ZZSC9020

Abstract

Explainable artificial intelligence (XAI) is becoming part of the discourse in data science circles, and for good reason. Explaining the underlying contributions of individual data elements to how a model operates facilitates transparency, allows the detection of bias, improves insights and even model usability. When applied to near-term electricity demand forecasting it will not only allow for more rapid improvements in creating these models, but also rationalise feature selection and therefore code concision, data collection and storage requirements. There have been numerous studies on the best way to forecast electricity demand, but few have used model explainability to assist in improving the models. Here we explore the use of SHAP and Model Approximation as key tools to facilitate model explainability, applying them to two types of advanced modelling - Extreme Gradient Boosting (XGBoost) and Long Short-Term Memory (LSTM) network models. We show that both are effective forecasting tools but that they differ in how they utilise the data to arrive at their predictions. Whilst both are most effective, and heavily reliant on, utilising prior electricity demand, other exogenous features such as calendar effects, temperature, humidity, solar radiation, solar panel electricity output and wind speed also contribute, taking on varying levels of importance for each model. We show here that it therefore matters both which type of model is selected for electricity demand forecasting, and which exogenous features are consequently prioritised for it to learn from, particularly when prior electricity demand is not present as a predictor. XGBoost should first and foremost utilise average temperature, solar radiation, day of the week, month of the year and recommended retail price. Humidity, wind speed and solar panel electricity output are somewhat less important to this model. For the LSTM, calendar effects like month of the year and day of the week contribute the most, particularly weekends, followed by solar panel electricity output, recommended retail price and all weather related features. This demonstrates that before productionising a model, that the relative contributions of the features in the data that contribute to it should be explained so as to extract maximum benefit from their appropriate inclusion within the given model and simultaneously minimise maintenance costs accruing from code maintenance, data collection and data storage.

Table of Contents

[1 Introduction 4](#_Toc132471981)

[2 Literature Review 8](#_Toc132471982)

[2.1 Short-term demand 8](#_Toc132471983)

[2.2 Model Explainability 8](#_Toc132471984)

[2.3 Dual Model Approach 9](#_Toc132471985)

[2.4 Feature Selection 10](#_Toc132471986)

[2.5 Historical Data 10](#_Toc132471987)

[3 Material and Methods 11](#_Toc132471988)

[3.1 Software 11](#_Toc132471989)

[3.2 Description of the Data 11](#_Toc132471990)

[3.3 Pre-processing Steps 12](#_Toc132471991)

[3.4 Data Cleaning 13](#_Toc132471992)

[3.5 Assumptions 13](#_Toc132471993)

[3.6 Modelling Methods 13](#_Toc132471994)

[3.6.1 Extreme Gradient Boost for Time Series Analysis 13](#_Toc132471995)

[3.6.2 Long Short Term Memory (LSTM) Model for Time Series Analysis 15](#_Toc132471996)

[3.6.3 Shapley Additive exPLanations (SHAP) 17](#_Toc132471997)

[4 Exploratory Data Analysis 18](#_Toc132471998)

[4.1 Descriptive Statistics 18](#_Toc132471999)

[4.2 Time Series Plots 19](#_Toc132472000)

[4.3 Outlier Analysis 20](#_Toc132472001)

[4.4 Correlation Analysis 21](#_Toc132472002)

[4.4.1 Correlation Heatmap 21](#_Toc132472003)

[4.4.2 Pairplot 21](#_Toc132472004)

[5 Analysis and Results 21](#_Toc132472005)

[5.1 Prior Model 21](#_Toc132472006)

[5.2 XGBoost for Time Series Analysis 22](#_Toc132472007)

[5.2.1 Prediction Error 22](#_Toc132472008)

[5.2.2 Feature Importance / XAI (with prior total demand variables) 23](#_Toc132472009)

[5.2.3 Feature Importance / XAI (without prior total demand variables) 24](#_Toc132472010)

[5.3 LSTM for Time Series Analysis 26](#_Toc132472011)

[5.3.1 Modelling and optimisation 26](#_Toc132472012)

[5.3.2 Manual Model Training Approach 26](#_Toc132472013)

[5.3.3 Model Version 1: Manual Optimisation 26](#_Toc132472014)

[5.3.4 Model Version 2: GridSearch 27](#_Toc132472015)

[5.3.5 LSTM Model Version 3: Bayesian Optimisation 27](#_Toc132472016)

[5.3.6 Comparing the LSTM forecast to the University supplied forecasts 29](#_Toc132472017)

[5.3.8 Feature Importance / XAI (with prior total demand variables) 30](#_Toc132472018)

[5.3.8 Feature Importance / XAI (without prior total demand variables) 32](#_Toc132472019)

[6 Discussion 34](#_Toc132472020)

[6.1 Data Quality 34](#_Toc132472021)

[6.2 XGBoost 35](#_Toc132472022)

[6.3 LSTM 35](#_Toc132472023)

[6.4 Model Explainability / XAI 36](#_Toc132472024)

[6.4.1 XAI for XGBoost 37](#_Toc132472025)

[6.4.2 XAI for LSTM 37](#_Toc132472026)

[7 Conclusion and Further Issues 37](#_Toc132472027)

[Appendix 43](#_Toc132472028)

[Tables 43](#_Toc132472029)

[Graphs 49](#_Toc132472030)

# 1 Introduction

In the past decade or more, the energy industry has become front and centre in the Australian news media, with front-page mentions common. This constant focus on Australia’s energy security reflects the degree of concern, within residential, industrial and political circles around the most effective approaches to establishing a reliable and affordable energy supply whilst still giving appropriate consideration to environmental impacts.

Whilst the conversation is ongoing, the ultimate aim appears to be to provide just enough energy to supply demand. Generating no more than is necessary will limit resource wastage and therefore costs, as well as reducing unecessary environmental impacts. According to Rai et al (2019), who address the electricity market in particular, it is due to the increasing penetration of renewable electricity generation that intra-day demand profiles have dramatically changed, driving out those producers who have been unable to quickly respond.

This is in an environment of decreasing national electricity consumption which puts extra pressure on electricity producers to optimise their processes. According to data available on the statista.com website (Statista, 2023), there has been a gradual and nearly consistent decline in Australian national electricity consumption over the 11 years to 2021 (Figure 1) - noting that at the time the Statista report was written, data was only available up to February 2022. We have calculated the average magnitude of this decline at approximately 1.93 terawatt hours per year.

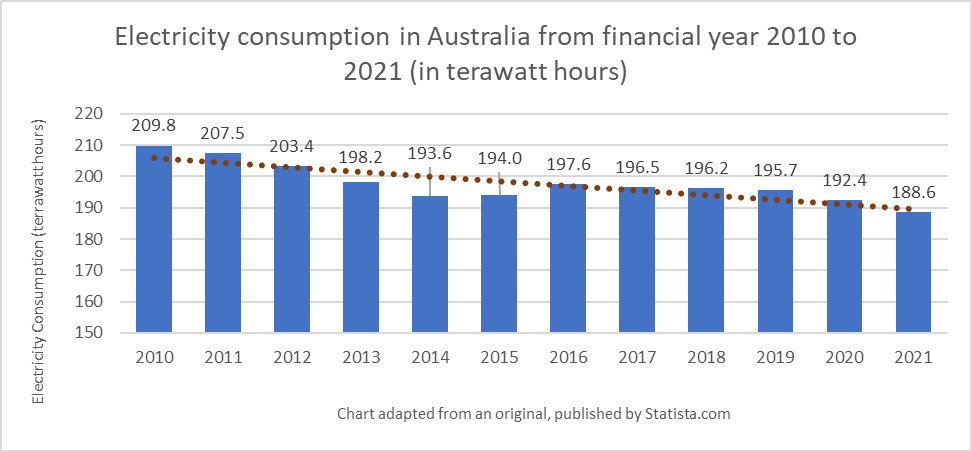


Figure 1: Electricity consumption in Australia from 2010 to 2021

This is not exactly replicated on a state basis when it comes to NSW, the subject area of the current study. As can be seen in the chart below derived from data on the energy.gov.au website (Energy Australia, 2023), NSW has had an overall electricity demand decline of only around 300 megawatt hours per year over the same time period. It can, however be seen that NSW electricity consumption appears to have declined dramatically for the first half of this period until 2015 where the decline was around 2.14 terawatt hours per year, but then it increased again by around 1.23 terawatt hours per year until 2020-21 (Figure 2).

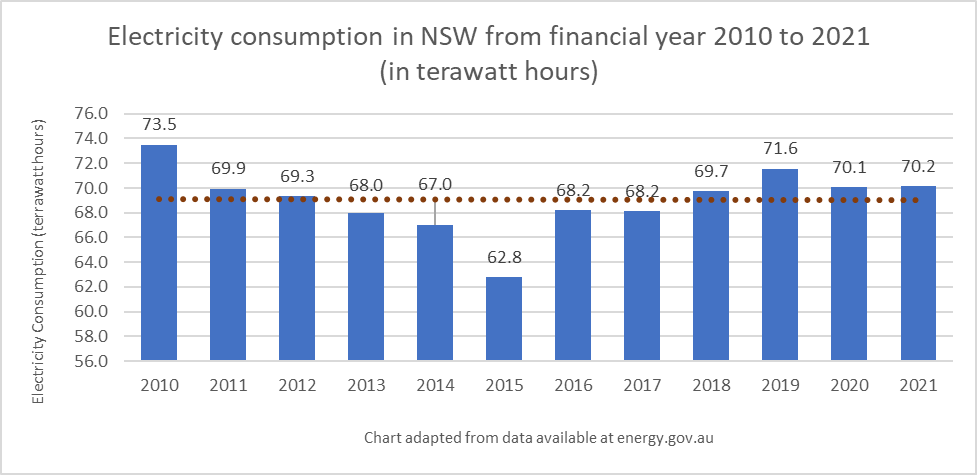


Figure 2: Electricity consumption in NSW during from 2010 to 2021

It seems that as consumption changes - whether increasing or decreasing, it is critical to ensure that there will always be a sufficient electricity supply to meet the anticipated demand. Given the competing goals between limiting resource wastage and cost on the one hand, and meeting electricity demand on the other, there clearly needs to be an ongoing balance established between them, and the more finely tuned the balance, the better the chance of meeting the expectations of government, consumers and producers alike.

Indeed, the current study recognises the importance of short-term electricity demand forecasting which addresses the need for finely balanced electricity production, by generating next-day electricity forecast models. It explores the relationships between the features seen as contributors to electricity demand within each of the chosen models, with a view to providing short-term accurate demand forecasts, and explaining the extent to which each feature contributes to the forecasts.

Boulaire et al (2014) assert that being able to adequately invest into and manage electricity assets requires an accurate estimate of future demand within “specified service areas”. With this in mind, the current study limits its focus to a regional level, and like the Boulaire et al study, is concerned with forecasting electricity demand for the Australian state of New South Wales. It utilises provided electricity demand data as well as data from a number of open sources including, daily temperature, humidity and solar radiation.

XAI refers to Explainable AI, which are a series of methods applied to artificial intelligence and machine learning models in an attempt to understand how they work. The model explainability aspect of the current study is concerned with comparing the XGBoost ensemble model with the LSTM recurrent neural network model, as they have both been highly thought of in the literature for demand forecasting tasks. What is largely missing from the literature is specifically exploring how different types of electricity demand forecast models differ in terms of which features are most important in contributing to their outputs. In this study we attempt to fill this gap by running a feature importance comparison between optimised XGBoost and LSTM demand forecast models.

The goals of this are not just to determine if there is value in including all the many features within the models in order to maintain accurate forecast results, but also to ensure there is not an ‘over-featurisation’ problem on a model-by-model basis. By this we mean that we want to explore which features are specifically useful for each model in order to possibly be able to reduce our data collection, maintain concise coding, and highlight unseen-before insights into the data.

Although it is billed as a model-agnostic method to explain the output of any machine learning model, we had some technical difficulty utilising the SHapely Additive exPlanations (SHAP) library with our LSTM model. It was, however, successfully applied to the XGBoost model, from which we were able to derive meaningful explanations. We attempted contact with Jakub Bialek of the Warsaw University of Technology, the author of a study of one of the main proponents we found of using SHAP with neural networks for forecasting (Bialek, 2022) to hopefully gain some advice. We also tried contacting Scott Lundberg, the author of the Python shap library at the University of Washington, for his advice. As of the date of publication of this report, we had not yet received a response from either of them.

As a consequence of our inability to leverage SHAP for the LSTM model, it was decided to utilise a different, but literature-supported, approach to explaining it, called Model Approximation. This method also utilised by Bouktif et al (2018) and Nakagawa et al (2019) involves running a much simpler model over the results of the LSTM model, using the LSTM predicted electricity demand results as the dependent variable, and then explaining the resultant simpler model. Although by its very nature, a simpler model doesn’t have the depth of sophistication that was used by the neural network to generate the original result, by training the simpler model specifically on the LSTM results it can give a general idea of the feature importances as they would have operated within the neural network. Indeed the Approximation Model was then analysed using the Python shap library.

The study found two major outcomes. The first was that it was easier to generate more accurate results using the LSTM neural network model than it was when using the XGBoost ensemble model. This was likely due to the time-series nature of the data. LSTM was designed for processing this type of data, whereas the XGBoost model required extensive data preprocessing to create multiple overlapping records with additional time lags for selected variables before the forecasts could be generated. There was also the additional factor of the limitations of the available computing power. Whilst the XGBoost model was trained on a machine with a CPU, the LSTM model was able to be trained on a machine with a GPU. This meant it was more difficult to apply optimisation code to extract the most effective hyperparameters for the XGBoost model, than it was for the LSTM model.

The second finding was that whilst previous total electricity demand exerted the highest, outsized influence on both models, the subsequent ordering of the feature importances between the XGBoost and LSTM models were rather different. The next top 5 features for the LSTM model appeared to be rounded out by humidity, solar radiation amount, average temperature, recommended retail price and whether it was a Sunday. For the XGBoost model, on the other hand, these next 5 features were Fridays, 6-days prior total electricity demand, Sundays, Saturdays and 5-days prior total electricity demand. These differences are likely due to how the models work internally, especially since the process of weighting features is significantly different in each model. Also possibly a factor is the level of experience of the investigative team. We did have some difficulty applying moving windows to the XGBoost model, and needed to resort to engineered lagged features. We did use moving windows in the LSTM model, but the coding required for this was less sophisticated.

Prior to running the models much time had been devoted to collecting, cleaning and calculating the average daily temperature records. But it seems that in neither model that average daily temperature figures prominently. This is rather surprising in light of all we have read about the effect of temperature on electricity demand in the literature. It may mean that the model is preferentially relying on other correlated measures, or that the models are incorrectly configured in some way.

However, when reviewing the literature, it appears our results are consistent with at least one other study. The short to mid-term electricity demand forecasting study by Bouktif et al (2018) also found that temperature contributed to an insignificant degree when compared to previous electricity demand lags that had been engineered into the data. Though when reviewing their study, it appears that they may not have one-hot encoded their categorical variables in the same way we did and there is a chance that their approach was different to ours.

# 2 Literature Review

The ability to forecast electricity demand has been deemed extremely important to the research community due to the need to both mitigate the costs of producing excess energy, as well as to avoid risking an under-supply of critical energy requirements to a given region.

As Muratori, Schuelke-Leech and Rizzoni (2014) argue that the generation of electricity must be enough to match electricity demand at any point in time, and take into account both seasonal and instantaneous fluctuations. They say “one of the biggest drivers of costs and capacity requirements is the electricity demand that occurs during peak periods”. Engle, Mustafa and Rice (1992) further note that “forecasts of daily system peaks are extremely important for generator management, maintenance scheduling, and purchase and sales of electricity across utilities”.

## 2.1 Short-term demand

There have been many studies conducted over the last few decades that address the question of forecasting electricity demand in both the near and longer term. In the current study we have elected to focus on forecasting the next day electricity demand for the state of NSW in Australia, as a preview to possible subsequent studies that may be able to extend this out to longer time horizons, and possibly also focus on smaller intra-state regions. According to Son and Kim (2016) “short-term electricity demand forecasting plays a significant role in power system planning, including economic scheduling of generating capacity, scheduling of fuel purchases, and power system management”.

A recent single meta-analysis by Verwiebe et al. (2021) reviewing the 6-year period between 2015 and 2020 yielded 348 papers dealing with this topic alone. Interestingly, while more than half of the papers covered all consumption sectors, the majority (99) of the remaining studies were focused on the residential sector, with the commercial and industry sectors following distantly behind with 51 and 27 papers respectively.

It has been surmised (Lisi and Shah, 2019) that prediction of energy demand is more accurate than prediction of energy prices partly due to more regular behaviour, which displays fewer spikes. This suggests that forecasting electricity demand is therefore more likely to provide the insights needed by energy retailers to also decide on their pricing strategies, implying that by forecasting demand, both pricing and supply requirements may be catered to.

## 2.2 Model Explainability

The current study is borne of an analysis of this information rich area of study, where it was discovered that while there are many approaches to the problem of forecasting electricity demand, not many of these studies incorporate any sort of explainability methodology to highlight the relative contribution of the data features to the results of the adopted model. A paper by Białek et al. (2022) is one of the few which actually does use an explanatory approach to understanding the models involved in an energy demand forecast study. That this instance of utilising an explanatory methodology was found in so recent a publication, is reflective of how new and under-developed this form of analysis is within the data science community.

Indeed, model explainability is emerging as one of the most important trends in data science today. Belle and Papantonis (2021) claim “business stakeholders in the very least, have growing concerns about the drawbacks of models, data-specific biases and so on”, and assert that data scientists are often unaware about the latest approaches to this field. They highlight several advantages of explaining models, including being confident in their correctness and robustness, understanding their biases, highlighting improvements that can be made to them, facilitating their transferability across application domains, and perhaps most importantly, improving human comprehensibility. Included in their toolbox are the use of SHAP values (in addition to other methods) which is leveraged in the current study. Białek et al. (2022) also highlight the use of SHAP values in their 2022 study emphasising explainability of their neural network forecast demand model.

According to Bialek et al. (2022) Shapley Additive exPlanations (SHAP) compares the mean difference of running a model multiple times with a particular feature to running it multiple times without this feature. This difference represents the attribution of the feature to the model. Instead of attribution reflecting model performance, it is relative to the model prediction, which makes it much easier to interpret than, say, Permutation Importance (which measures the effect of the presence of a feature on model error). SHAP is also able to cope with models that train on many features, irrespective of type, or whether or not they are correlated - such as in the case of the present study which deals with time-delayed temperature, humidity and solar radiation features (for example). It can also cope with time-related features such as day and season. The main drawback of the general SHAP implementation is that the computation time grows exponentially with the number of features. However, in the context of the present study where we are modelling using a limited number of variables, using SHAP values appears to be an appropriate choice.

Due to SHAP’s inability to work directly with LSTM’s, the current study was required to research other methods of explainability. Indeed, the Model Approximation method employed by Bouktif et al (2018) and Nakagawa et al (2019), leveraged the use of this method which involves running a simpler, but explainable model over the results of the more sophisticated model. In the case of Boutkif et al, they ran a regression decision tree model. Nakagawa et al preferred to use a linear regression model. The Bouktif et al study has more applicability to the current study, as they ran their approximation model against the output of their short to mid-term electricity demand forecast LSTM model. In this way the predicted demand values became the independent variable for the regression model, against which feature importance was then extracted. Interestingly, the Bouktif et al (2018) study found an outsized effect of prior electricity demand on the model's effectiveness, with the other features such as temperature and humidity relegated far down the importance list.

## 2.3 Dual Model Approach

As the current study aims to highlight possible differences between how much different models rely on each of the data features used to train them, we elected to select two dissimilar types of machine learning model. The decision to focus on the XGBoost and LSTM models in particular for creating the forecasts in the current study, is due to the popularity of both models in the literature, as well as the very different ways in which they each operate.

Although the problem of electricity demand forecasting considers time-series data, and XGBoost is a multiple regression algorithm, we have been able to utilise it for forecasting by engineering the data into multiple, overlapping time windows, within each of which are contained aggregated representations of the data for the specific window. This approach is echoed by Zhao et al. (2022) in their study of Singaporean electricity demand, and others (see Lu et al., 2020 and Divina et al., 2019).

The LSTM (Long Short-Term Memory) model was chosen as it has also proven both popular and effective, with Fatema, Kong and Fang (2021) choosing to use it after claiming “deep learning algorithms provide a strong way to extract key attributes from complex variable historical electricity data to predict the demand/price efficiently. As a result, applying deep learning algorithms to process time series electricity data yields better results”. They continue by saying “Researchers have been applying deep-learning methods, such as Artificial Neural Network[s] (ANN[s]), that can produce excellent results in critical issues, reduce errors and improve the models’ prediction accuracy”.

Others, such as Wu et al. (2022) have also seen fit to utilise LSTM models for electricity demand modelling.

Indeed, Nia, Awasthi and Bhuiyan (2021) commented about neural networks (of which LSTM is a prime example) that “NNs are most distinguished for being appropriate to predict the future values of nonlinear data sets…”. Le et al. (2019) highlight that LSTM modelling is a popular and specialised version of artificial recurrent neural networks (RNN’s) that are especially capable of modelling sequential or temporal data. They also note that LSTM’s are capable of handling changes within extremely long sequential data, and has therefore been used widely including in time-series analysis. As electric energy demand forecasting is a multivariate time series problem, LSTM’s seem to be an appropriate modelling tool to use to solve it.

The literature supports the assertion that nonparametric functional models give better results than parametric models (Lisi and Shah, 2019). XGBoost is a non-parametric model, as it does not make strong assumptions about the underlying data distribution or the functional form of the relationships between the input features and the target variables. It is able to learn the data environment and thereby make reasonable inferences. LSTMs are considered parametric models because they have a fixed number of parameters once their architecture is determined, but the fact that the parameters are learned from the data together with the configuration of their gates is what makes them highly effective. The current study uses a Bayesian Optimisation method to automatically select the best combination of hyperparameters for the LSTM forecasting model, and a grid-search method to automatically select the optimum combination of hyperparameters for the XGBoost model.

## 2.4 Feature Selection

When considering which features to use in our models, the current paper relies heavily on the literature. Tribble (2003) highlighted that “weather has a significant impact on load demand and load forecasting”. Her PhD thesis utilised temperature, apparent temperature, relative humidity, solar radiation and wind speed. Additionally, she included the non-weather features of weekday, weekends and holidays.

Son and Kim (2017) utilised 20 “influential variables”. These included the following weather-related variables - mean temperature, maximum temperature, minimum temperature, relative humidity, wind speed, rainfall, daylight time, global solar radiation, cooling degree days, heating degree days, degree days, vapour pressure, enthalpy latent days; and the following social variables - real gross domestic product, industrial production index, population, consumer price index and real electricity price.

In terms of the relative importance of temperature, Manderson and Considine (2021) find “monthly electricity consumption responds to temperature in a non-linear manner for the residential and commercial sectors, with elevated consumption for days at the two extremes of the temperature distribution”.

Maia-Silva, Kumar and Nateghi (2020) claim “projections based on air temperature alone underestimates cooling demand by as much as 10–15% under both present and future climate scenarios”, asserting that humidity is highly important in bridging this prediction gap. Schoen (2005) highlights the importance of humidity, when he discusses the apparent temperature as being a joint function of temperature and humidity in terms of the widely used temperature-humidity index.

Shah, Iftikhar and Ali (2022) proposes a number of features should be included in order to efficiently forecast electricity demand and prices. They include the yearly, seasonal and weekly electricity demand trends, calendar effects and lagged ‘exogenous information’.

## 2.5 Historical Data

Due to the changing nature of electricity demand over time, we have used 5 years of historical data to generate the models (four for training and one for testing) despite the availability of around 10 years of data. This is in line with Fatema et al. (2021) who only used 3 years of data to train their model. Because of the apparent reduction over time in electricity demand, we have seen in the data, we have added another year to help account for this.

# 3 Material and Methods

## 3.1 Software

We used Microsoft Teams for group discussions, instant messaging and conference calls. Project planning was managed through ClickUp, which allowed all members to collaboratively manage lists, kanban boards, checklists and gantt charts in the cloud. Additionally, all information was stored and shared via a GitHub repository. This includes all data, code, and links to ClickUp and Google documents. Storing all information via GitHub was an effective method of collaboration. We found that using cloud based tools was the best option considering group members are in different states and countries.

Several of our group members were more confident with Python than R, hence we opted to use Python to perform computational and visualisation tasks. The code was written and run in Jupyter Notebooks as it’s a good resource for compiling both code and text in a markdown type environment, resulting in an aesthetically pleasing format.

Coding tasks were performed on four computers, where the specifications are outlined in [Table 1](#Table1) in the Appendix, along with python packages utilized.

## 3.2 Description of the Data

All data used in this study were stored in a project-dedicated GitHub repository located at <https://github.com/ChrisB-UNSW/GroupF_Project>, within the data folder. Exploratory, data engineering and modelling code are also stored in the repository, within the code folder. For our analysis, several datasets have been used. These datasets are stored in various formats such as flat files, zip files, or folders containing several flat files that require merging.

Sourced from the Market Management System database (MMS), the dataset Total Electricity Demand contains data regarding total energy demand for NSW. This data is stored as a CSV file, contained within a zip file, and is approximately 44 MB in size. The dataset consists of 1,323,398 rows and 3 variables. An in-depth description of these variables is available in [Table 2](#Table2) in the Appendix.

Sourced from the MMS database, the ‘Forecast Demand’ dataset contains forecast energy demand data for NSW. This data is stored as a CSV file, contained within a two-part zip file, and approximately 770 MB in size. The dataset consists of 11,619,503 rows and 6 variables. An in-depth description of these variables is available in [Table 3](#Table3) in the Appendix.

Sourced from the Australian Data Archive for Meteorology (ADAM), the ‘Air Temperature’ dataset contains data regarding air temperature in NSW, measured from the Bankstown Airport weather station. This data is stored as a CSV file, contained within a zip file, and is approximately 8 MB in size. The dataset consists of 247,646 rows and 3 variables. An in-depth description of these variables is available in [Table 4](#Table4) in the Appendix.

Sourced from the Australian Energy Market Operator (AEMO) website, the ‘Price and Demand’ dataset contains energy price data and demand data for New South Wales. This data is stored as 152 CSV files, contained within a folder, with each file approximately 70 KB in size. The dataset consists of 302,448 rows and 5 variables. An in-depth description of these variables is available in [Table 5](#Table5) in the Appendix.

Sourced from the Time and Date website, the ‘Weather in Bankstown’ dataset contains humidity and wind speed data, measured from Sydney Airport. Using Python libraries BeautifulSoup and Selenium, web scraping techniques were employed to gather this data. The web scrape ultimately generated 7 separate CSV files, one file per year. Each file ranges from 1 MB to 7 MB in size, increasing in size each year. Once all files have been consolidated, the total dataset consists of 371,126 rows and 8 variables. An in-depth description of these variables is available in [Table 6](#Table6) in the Appendix.

Sourced from the Australian Government site, Clean Energy Regulator, the ‘Solar Panel Installations’ dataset contains data regarding small-scale solar panels that have been installed per postcode. This data includes a monthly count of installations and total power output in kilowatts. This data is stored as 13 CSV files, with each file approximately 650 KB in size. Each file contains postcode After consolidating the files and pivoting the month columns, the final dataset consists of 302 rows and 4 variables. An in-depth description of these variables is available in [Table 7](#Table7) in the Appendix.

Sourced from the Bureau of Meteorology, the ‘Solar Exposure‘ dataset contains daily records regarding solar exposure per postcode. This data is stored as a CSV file and approximately 411 KB in size. The dataset consists of 4,816 rows and 6 variables. An in-depth description of these variables is available in [Table 8](#Table8) in the Appendix.

Sourced from the Bureau of Meteorology, the ‘Min Temperature’ dataset contains daily records regarding minimum temperature per postcode. This data is stored as a CSV file and approximately 760 KB in size. The dataset consists of 4,830 rows and 8 variables. An in-depth description of these variables is available in [Table 9](#Table9) in the Appendix.

Sourced from the Bureau of Meteorology, the ‘Max Temperature’ dataset contains daily records regarding maximum temperature per postcode. This data is stored as a CSV file and approximately 766 KB in size. The dataset consists of 4,816 rows and 8 variables. An in-depth description of these variables is available in [Table 10](#Table10) in Appendix.

Sourced from the Australian Government site, Data, the ‘Public Holidays’ dataset contains data regarding public holidays per state. This data is stored as 7 CSV files, with each file ranging between 9 KB and 58 KB in size. After consolidating the files, the final dataset consists of 823 rows and 5 variables. An in-depth description of these variables is available in [Table 11](#Table11) in the Appendix.

## 3.3 Pre-processing Steps

The Electricity Forecast Demand dataset was provided in a multi-part compressed file. The file archiving application 7-Zip was used to merge these parts and extract the dataset as a single entity. As the dataset was very large after merging, it was then stored in a single compressed file within the GitHub repository.

Many of the other datasets were stored within compressed files. For each case, the “zipfile” Python library was used to extract the contents of the compressed file. The contents, which consisted of one or more CSV files, were then stored in a temporary folder in the current working directory to use for data cleaning and analysis. A temporary folder was used due to storage constraints. If the contents contained more than one file, these files were merged and then converted into a dataframe object using the “Pandas” Python library. If it contained only one file, this file was simply converted into a dataframe object using the same library.

Web scraping techniques were employed to gather weather data from the Time and Date website. The webpage was accessed by sending HTTPS requests and the webpage’s underlying HTML was parsed to extract the data. The Python package “BeautifulSoup” was used to access and parse the webpage and the “Selenium” Python library was used to communicate with the web browser. The website did not offer a data export option, so this technique was used as an alternative solution. The web scraping resulted in a CSV file for each year of scraped data. These files were then merged and converted into a dataframe object using the “Pandas” library.

For each dataframe, the data types were updated. Year, month, and day integers were then extracted from all date variables. Redundant columns were removed, such as columns containing only one value, identification values, or information not of interest. Each data frame was then aggregated by date, taking the average of all exogenous variables. Finally, all datasets were merged on the year, month and day columns.

## 3.4 Data Cleaning

Outlier analysis was performed using box plots. Outliers were defined as values that fell outside of the upper and lower quartiles. For most variables, outliers were removed as they were few in number but could still have had a negative impact on model performance. Contrastingly, the outliers contained in the variable OUTPUT were across an entire month, so we opted to keep these outliers within the dataset. Similarly, because there were few missing variables in the dataset, these were removed prior to analysis.

One-hot encoding was applied during the modelling process on the weekday variable. This ensures that values of different magnitudes do not take on disproportionate importance. Standardisation was also applied to convert all variables to a common scale, also to reduce the effect of differing variable magnitudes affecting the model outcome.

## 3.5 Assumptions

This study assumes that the provided actual and forecast demand data provided by the University of New South Wales are accurate. It also assumes that the exogenous data used in the study are correct as collected from the various online data providers. It also assumes that the weather data being utilised is representative of the entire NSW region, despite being collected only from Bankstown airport and Sydney airport. The study also assumes that all the timestamps are correct and representative of the same times across all data sets, in order to facilitate data consolidation across them.

## 3.6 Modelling Methods

### 3.6.1 Extreme Gradient Boost for Time Series Analysis

According to the official online documentation (XGBoost Documentation, *Introduction to Boosted Trees*, 2022), Extreme Gradient Boost (XGBoost) is an ensemble machine learning algorithm that is built from a gradient boosting framework, which can be used for either classification or regression problems. XGBoost iteratively trains an ensemble of weak decision trees which ultimately produces stronger forecasts. Error correction is performed at each iteration, using loss functions with penalty terms (regularisation) that are determined based on the performance results from the previous iteration. At each iteration, a similarity score is calculated to minimise the gradient of the loss function, which is known as “gradient boosting”. This makes it a more robust model than standard gradient boosting methods.

The loss function is used to quantify a tree’s prediction quality by comparing the actual values and predicted values. The official online documentation for XGBoost (XGBoost Documentation, *Introduction to Boosted Trees*, 2022) highlights that decision trees are built by minimising this equation called an objective function, which consists of a loss function and regularisation term.

Where n is the number of observations, are actual values, are predicted values, is the loss function, is a scaling term, O is the optimal value, and is the regularization term.

The documentation continues by saying that to optimise the output from the previous decision tree at each iteration, the prediction value must be replaced by the previous prediction value plus the output value.

To solve for the output value:

XGBoost calculates a similarity score which is used to update the values in the next decision tree to minimise the loss function:

According to the literature (see literature review section), XGBoost can be used for time series analysis by employing lagged variables. Alternatively, more experienced programmers could consider another approach such as the sliding window method (Zhao et al, 2022). But in this report, due to both time constraints and lack of coding experience, lagged variables will be used instead. Using either approach, the model can learn from past information.

To create the lagged variables, the values of the selected features HUMIDITY, WINDSPEED, HOLIDAY, SOLAR, RRP, OUTPUT, TEMPAVE, and TOTALDEMAND are to be shifted backwards in time. The distance of this ‘lookback’ time will be experimented on, specifically observing model performance with lookback between 1 and 7 days. The lookback that produces the highest accuracy will be used in the final model. Additionally, as we are interested in predicting demand one day ahead, the response variable will be defined as TOTALDEMAND shifted one day forwards in time. Resulting from value shifting, missing values will appear at the head and tail of the dataset. These rows will be removed.

The X matrix will contain HUMIDITY, WINDSPEED, HOLIDAY, SOLAR, RRP, OUTPUT, TEMPAVE, TOTALDEMAND, HUMIDITY (t-i), WINDSPEED (t-i), HOLIDAY (t-i), SOLAR (t-i), RRP (t-i), OUTPUT (t-i), TEMPAVE (t-i), TOTALDEMAND (t-i), where t represents the current point in time, and i represents the the number of lookback days (i=1,2,...,7). The response variable will be TOTALDEMAND (t+1).

To split time series data into train and test sets, the latest records will be used as the test set. As the data ends in August 2022, the test set will range from August 2021 to August 2022, with the training set consisting of the remaining prior data.

The X training and X test sets will be standardised to ensure all features are within a common scale. Each set will be standardised separately to avoid bias. The transformation will be performed using Sckit-Learn’s MinMaxScaler function, which individually scales each feature between 0 and 1.

The model will be defined using the “xgboost” Python library. The hyperparameters colsample\_bytree, gamma, learning\_rate, n\_estimators, reg\_alpha, reg\_lambda, and subsample will be grid-searched in order to find the most optimal combination, consequently minimising error and maximising model accuracy. The best hyperparameters will then be fitted to the model and trained on the test set. Finally, predictions can be made.

The following hyperparameters are defined according to the official XGBoost online documentation (XGBoost Documentation, 2022). N\_estimators represents the number of base learners (decision trees) a model uses. Each of these trees are iteratively added to the model. Colsample\_bytree represents the percentage of features that are randomly selected to build each tree. For example, if there are 10 features and colsample\_bytree = 0.8, only 8 features will be sampled for each tree. Gamma represents the minimum loss reduction that is required to split a tree’s leaf node. Learning\_rate represents the step size that the model takes each time before updating the weights of new trees. Reg\_alpha refers to Least Absolute Shrinkage and Selection Operator (Lasso) regularisation (L1), which adds a penalty term to the loss function, proportional to the absolute value of the coefficients of the features. Lasso regularisation will remove less important features to improve model performance. Reg\_lambda refers to Ridge regularisation (L2), which adds a penalty term to the loss function, proportional to the square of the coefficients of the features. Ridge regularisation does not remove features but instead shrinks less important features to improve model performance. The subsample hyperparameter represents the percentage of observations used in each tree (XGBoost Documentation, *XGBoost Parameters*, 2022).

To assess model performance, we will firstly observe R squared and the mean absolute error. Prediction error assessments will be performed by observing plots that compare prediction values and actual values. These results will also be compared with forecasted values determined by a prior model.

Finally, to explain an XGBoost model, which is the premise of our research question, it is necessary to highlight the most important features when making predictions. Feature importance will be assessed using the “xgboost” and “SHAP” python libraries. We aim to observe the differences between the results from each package and if feature importances are similar to the results of the LSTM model.

### 3.6.2 Long Short Term Memory (LSTM) Model for Time Series Analysis

According to the official online Research Article (International Journal of Sustainable Engineering, *Electricity demand and price forecasting model for sustainable smart grid using comprehensive long short term memory,* 2021), an Artificial Neural Network (ANN) model is designed to mimic the neuronal structure of the human brain. An ANN model consists of an input layer, one or more hidden layers, and an output layer. The input layer accepts the input variables in a specified multi-dimensional array format. It sits on one or more hidden layers that do the processing which then passes their combined output to the output layer that simplifies the results and produces the output variables. Each layer contains multiple neurons and their relevant activation functions. Multiple hidden layers can enhance the ability to handle non-linear and time series data.

There are 3 main types of artificial neural networks currently in use: ANN’s (Artificial Neural Network), CNN’s (Convolutional Neural Networks) and RNN’s (Recurrent Neural Networks). Among them, RNN’s are used to process and interpret time series data where the output of a processed node is fed back into nodes of the same or previous layers. In terms of the current research, a Long Short-Term Memory (LSTM) network model, which is one of the most advanced types of RNN, has been adopted. It has many characteristics that render it an appropriate tool for forecasting electricity demand.

LSTM’s are particularly suitable for learning long range dependencies, while the conventional recurrent network cannot handle a large amount of sequential (sequence-to-sequence) input data. Given the significant time lag between the inputs and the corresponding outputs, LSTM’s can effectively learn data containing long-range temporal dependencies, and thus, makes it an ideal option for this research.

Figure 3 illustrates the structure of an LSTM at a single time step, along with the equations for a time unrolled representation. The LSTM structure comprises of three main gates: forget gate (), input gate () and output gate ()

The model takes a single timestep input and output and can be fed with the output of a CNN or the input sequence directly as x(t). The inputs from the previous LSTM timestep are represented by h(t-1) and c(t-1). The output of the LSTM for the current timestep is o(t). Whereas 𝛿 and tanh are the activation function, W and U are the weight of forget gate, and b is bias vector. Additionally, the LSTM generates c(t) and h(t) for the next timestep LSTM to consume.

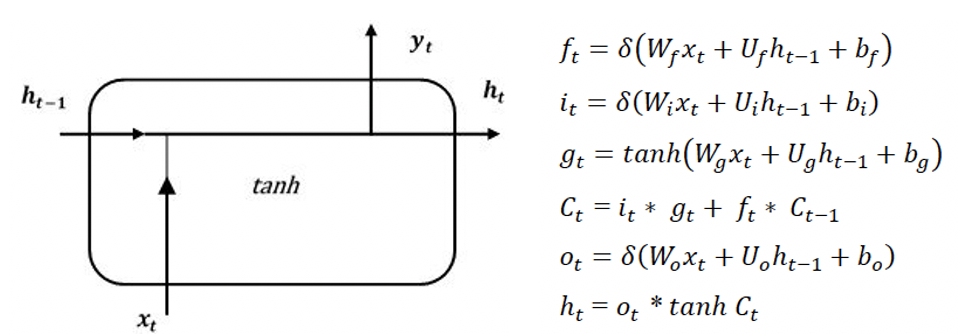


Figure 3: Standard Recurrent Neural Network (RNN) (Kuo and Huang 2018)

After the data has been cleaned, weekday data is encoded using one-hot encoding, all features will then be normalised and re-shaped into a 3D tensor format that can be input into the LSTM model.

The 3D tensor format has a shape based on batch\_size, time\_steps and features:

* Batch\_size: the number of samples in a batch
* Time\_steps:  the number of time steps in each sequence
* Features is the number of features in each time step.

A Sequential Keras Tensorflow model will be built in an Anaconda notebook. Initially, the model will be constructed using basic model structure. Based on these initial results, a process of trial-and-error will be carried out to improve the model performance by changing the number of layers, the training epochs, and adjusting the hyperparameters.

The model complexity and hyperparameters will be adjusted if the model becomes prone to overfitting (the training accuracy being much higher than the validation accuracy) in an attempt to find the optimal values.

After the basic LTSM model is constructed, an enhancement/tune-up process will be conducted for all hyperparameters in order to achieve the best model. This will be implemented by three different methods: GridSearchCV, experimenting manually with different hyperparameters, and through the Bayesian Optimisation approach. The results of the three methods will be compared, and will inform the study of the best combination of hyperparameters to use in the finalised model.

Important factors to consider during the trial-and-error phase to achieve the best LSTM model are:

* Model complexity: the number of layers and hidden units in each layer. Starting with a single layer and minimal hidden units and gradually increasing the complexity to see if the results improve.
* “Dropout” regularisation method:  adding Dropout as a function to LSTM layers to prevent overfitting. This technique randomly drops out (sets to zero) some units in a particular layer during training. This is designed to enhance the robustness and reduce the feature dependency for the remaining units in the layer.
* Dropout rate: the percentage of units in the layer that randomly drop out during the training process.
* Activation function for each layer: is applied to the output of each unit in a layer to introduce nonlinearity into the model and allow the LSTM model to learn complex patterns in the data.

There are 4 activation functions that can be used in LSTM models: sigmoid, tanh, relu and softmax. The goal of this analysis is to forecast rather than classify, considering the complex patterns of the data and the solving problem; therefore, only 3 activations are used in building the LSTM model: sigmoid, tanh, relu. And it is a good practice to use different activation functions for different layers.

* Learning rate: determine how long until the model converges with the optimal solution. If it is too large, the model may oscillate or diverge and inversely, the model will take long to converge. Learning rate was closely chosen during the process of experimenting and tuning the model.
* Epoch: a full iteration over all training examples of the dataset. The number of epochs contributes to the overfitting or underfitting of the model. Number of epochs chosen should yield the best generalisation performance for the model.
* Batch size: depends on the amount of training data and the available memory of the training machine. Generally, batch size of 32 or 64 is commonly used in training a LSTM model.

Note that due to time constraints and compute limitations, other advanced hyperparameters will not be taken into account. The model will then be compiled and trained based on the defined training data set and evaluated based on test data to compute the Mean Squared Error and R-squared.

Diagnostics / Performance measures:

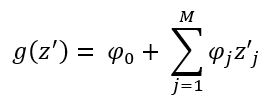
* R-squared coefficient (R2): the higher R2 indicates the predicted values fit well with the actual values. In a regression problem, R2 value implies the accuracy of the model.
* Mean Absolute Error (MAE): measure the difference (on average absolute) between predicted values and actual values. Model with lower MSE means better performance.

### 3.6.3 Shapley Additive exPLanations (SHAP)

Shapley Additive Explanations, or SHAP, is a method introduced by Lundberg and Lee (2017) to assist with the interpretation and explanation of predictions of Machine Learning models based on the cooperative game theory concept of Shapley values.

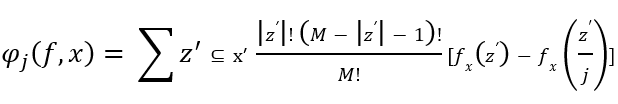
In cooperative game theory, the Shapley value of a player is a measure of their contribution to the overall outcome of the game. In the context of our paper, we can think of the features in our datasets as players in a cooperative game where each feature contributes a certain amount to the prediction made by the model. SHAP uses this concept to compute the contribution of each feature to the final prediction by considering all possible subsets of features and their corresponding predictions.

The SHAP explanation model can be defined in the following way:



where g is the explanation function, z'∈〖{0,1}〗^M is the vector of simplified binary features and M is the number of features present in the model. φ\_j∈ R is the feature attribution for a feature j, the Shapley values, and acts as the coefficient in the explanation model. φ\_0 is a constant and is the mean model prediction for the dataset.

As stated by Bialek et al. (2022), “this form of the model gives particularly desired property – sum of attributions of all important features is approximately equal to the output of the original model f, therefore f(x) ≅ g(x)” and that feature attribution, SHAP values, are calculated in the following way:



Bialek et al. (2022) explains that the attribution of the j-th feature is the mean difference of running the model multiple times with this particular feature, and multiple times without this feature. It is this difference calculated that represents the attribution of the feature to the model. As the attribution is relative to the model prediction, unlike other methods such as Permutation Importance which measures the impact of the feature’s presence on the model’s error, the attribution is much easier to comprehend. SHAP is also able to cope with models that train on many features, irrespective of type, or whether or not they are correlated - such as in the case of the present study which deals with time-delayed temperature, humidity and solar radiation features. SHAP can also handle time-related features such as day, month and season.

The main drawback of the implementing SHAP is that the computation time grows exponentially with the number of features. However, in the context of this study, where we are modelling using a limited number of variables, implementing SHAP to derive feature importance appears to be an appropriate choice.

# 4 Exploratory Data Analysis

## 4.1 Descriptive Statistics

[Table 12](#Table12) in the Appendix provides an overview of the essential statistics that describe the features used in the current study. It can be seen that each of the features are represented with quite different magnitudes, which led us to require all features to be normalised before analysis.

Whilst most of the features enjoy similar means to their medians, Energy output (in kwh) from solar panels (OUTPUT) appears to be somewhat different, with a larger difference between them. This may reflect the increasing usage of solar panels over the period of this study, skewed towards more recent years, which will perhaps exert a greater impact over time in its role as a predictive feature in the model.

## 4.2 Time Series Plots

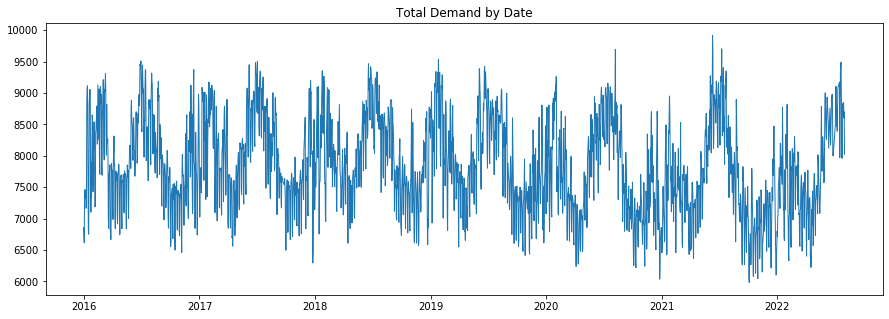


Figure 4: Total Electricity Demand time series plot

Total electricity demand shows strong seasonality, with a slight downward trend over time (Figure 4). It also appears to be increasing in variability as time progresses, with lower minimums and higher maximums at the extremes.

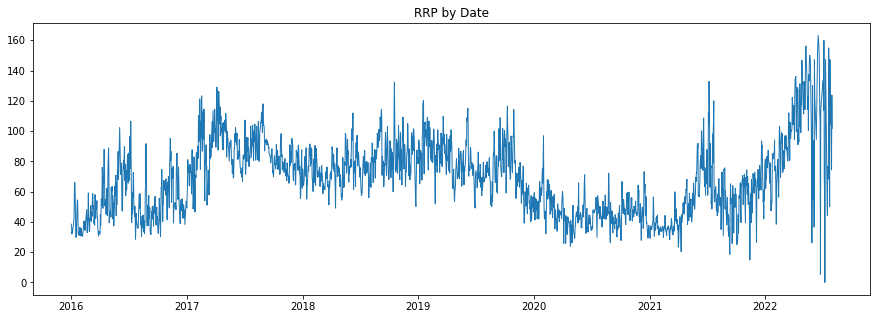


Figure 5: Total Electricity Recommended Retail Price time series plot

Electricity pricing (Recommended Retail Price - RRP) appears to be highly variable over the 11 years (Figure 5) that the data describes, with more recent extremes at both high and low price points. That this variation contains information relevant to predicting electricity demand, and to what extent, is definitely of interest.

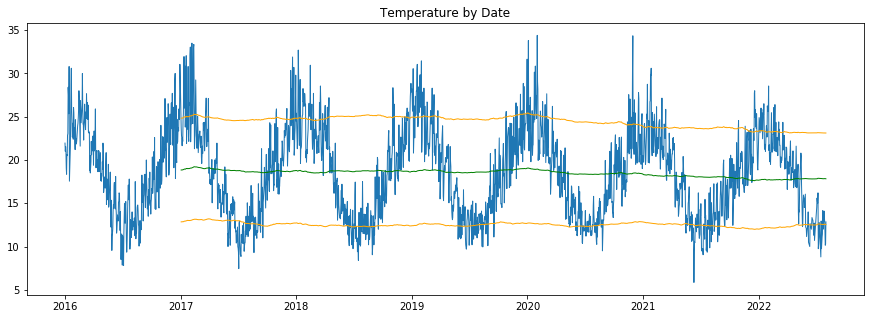


Figure 6: Total Temperature time series plot

As would be expected, the temperature data shows strong seasonality over time (Figure 6). There also appears to be a slight downward trend over time. Whether this reflects the NSW regional situation has not been analysed, and in fact is one of the study’s assumptions.

The line plots superimposed on this chart highlight the gradual decrease of yearly rolling average of minimum, mean and maximum daily temperature values. This is also seen in the temperature ranges appearing to decrease over time.

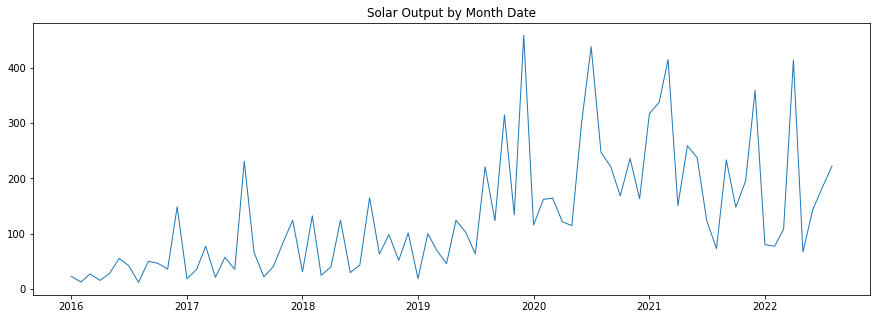


Figure 7: Solar Output time series plot

There has been a marked increase in solar electricity output over the last 3 years, which may have an impact on electricity demand as measured by the electricity retailers. Figure 7 shows the highly erratic nature of this measure, which would not only be affected by seasonal factors (ie available sunshine) but also by the number of solar panel installations which varies due to price, availability, perceived benefit and government subsidies.

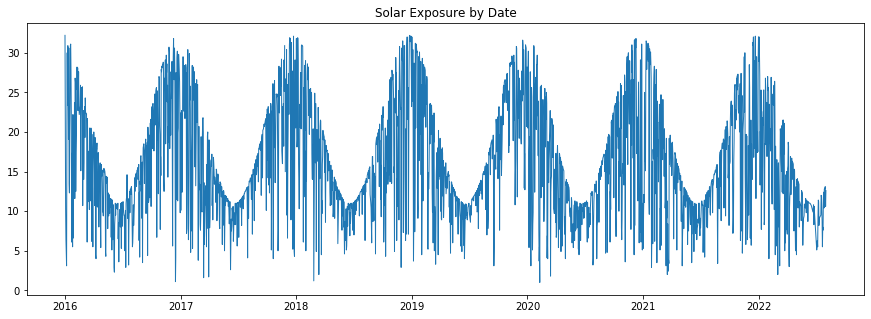


Figure 8: Solar Exposure time series plot

As seen in Figure 8, solar exposure appears to be highly seasonal, with no apparent outliers. Despite this there appears to be increased distortion in the last 1-2 years of the data.

In [Figure 9](#Figure9) from the Appendix, humidity data displays likely seasonality with high variation, and appears to be increasing slightly in the last 2-3 years.

Wind speed displays a seasonal pattern with high variation, as seen in [Figure 10](#Figure10) from the Appendix. This variability appears to be higher in the last 1-2 years.

## 4.3 Outlier Analysis

Outlier analysis was performed by observing boxplots from before and after outlier handling for each feature (Appendix [Figure 11](#Figure11)). Some of the features, including the dependent variable, contained outliers which were mostly removed as they consisted of very few data points. The only feature from which the outliers were not removed was solar output (OUTPUT) as this would have represented an entire month of data and there is reason to believe that these outliers actually represent valid data, which possibly could go some way towards explaining some of the electricity demand fluctuations during that time.

## 4.4 Correlation Analysis

### 4.4.1 Correlation Heatmap

A correlation heatmap (Appendix, [Figure 12](#Figure12)) was produced to observe correlations between variables. A moderate positive correlation between retail price and demand can be seen as well as between temperature and humidity and temperature and solar electricity generation. Moderate negative correlations can be seen between solar electricity generation and electricity demand and retail price as well as electricity demand and temperature and solar electricity generation. Very weak correlations are seen between wind speed and demand, and humidity and demand.

### 4.4.2 Pairplot

Delving into correlations further, we produced a pairplot for non-categorical variables (Appendix, [Figure 13](#Figure13)). As the time series plots highlighted seasonal trends, we decided to categorise the data into two groups: (1) data that fell within the first and last quarter of the year, and (2) data that fell within the second and third quarter of the year. This was to observe any trends when separating the warmer months from the cooler months.

The pairplot showed us that there were some differences between the two groups. As expected, variables such as HUMIDITY, SOLAR, and TEMPAVE increased during the warmer months. Variation of RRP was greater during the colder months. For each category, TOTALDEMAND showed a linear relationship with TEMPAVE. As the temperature increases above the median, demand increases, and similarly, as temperature decreases below the median, demand increases. Based on these results, we suspect that the variables TOTALDEMAND, HUMIDITY, SOLAR, TEMPAVE, RRP and time of year will be among the main influences for electricity demand forecasting.

# 5 Analysis and Results

## 5.1 Prior Model

Prior forecasting values were provided, which contained an R squared value of 96 and an MAE of 112.8. Figure 14 compares the forecast with the test set.

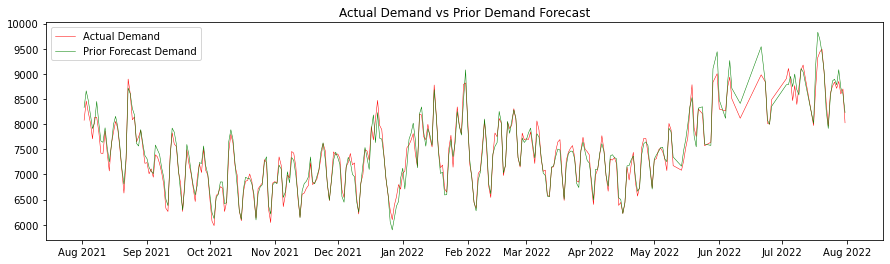


Figure 14: Actual Demand and Prior Demand Forecast time series plot

## 5.2 XGBoost for Time Series Analysis

For this report, we used an XGBoost model to make day ahead predictions of electricity demand. We chose this model based on our literature findings that XGBoost is an algorithm that produces highly accurate results.

The categorical variable, WEEKDAY, was one-hot encoded, where each variable was separated into individual binary features. This is to avoid bias in the model, giving each category equal weight.

Lagged variables were created for HUMIDITY, WINDSPEED, HOLIDAY, SOLAR, RRP, OUTPUT, TEMPAVE, and TOTALDEMAND using the shift function from the Pandas library. The amount of lookback was experimented on, specifically values between 1 and 4 days. As we are interested in making day-ahead predictions, the response variable was defined as TOTALDEMAND shifted one day forwards in time. Shifting the values resulted in some missing values at the head and tail of the dataset, so these rows were removed.

Data standardisation was performed using Scikit-Learn’s “minmaxscaler” function, which scales each feature within the range 0 and 1.

To implement model training, a grid search was performed on a baseline model (no lookback) and models with 1 to 7 days lookback. The results were then compared and displayed in [Table 13](#Table13) from the Appendix.

The model with 7 days lookback had the best performance, with an R squared value of 74 and MAE of 294.7, and the model with 3 days lookback had the worst performance, with an R squared value of 70 and MAE of 315.5.

The most optimal hyperparameters with a lookback of 7 days were fitted to the XGBoost model and predictions were calculated using the “predict” function from “sklearn”. To visualise how our predictions compared to actual demand and a prior model’s demand forecast, these plots were produced.

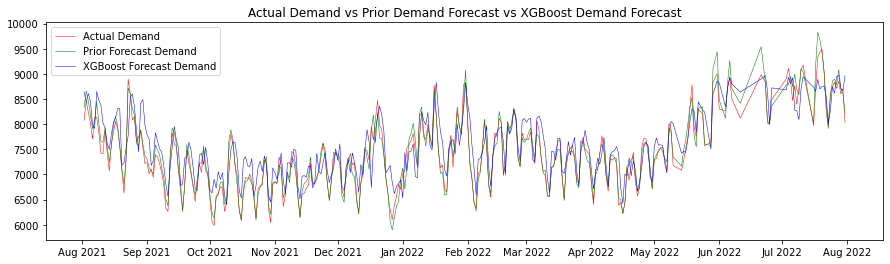


Figure 14: Actual Demand vs Prior Demand Forecast vs XGBoost Demand Forecast time series plot

We can see in figure 14 that the general pattern of the XGBoost predictions was correct, but there is some error in comparison to the actual values and the prior model.

### 5.2.1 Prediction Error

To further assess the errors in prediction by the XGBoost model, two graphs were created. The first graph plots actual demand against predicted demand, and the second graph displays error distribution (Figure 15).

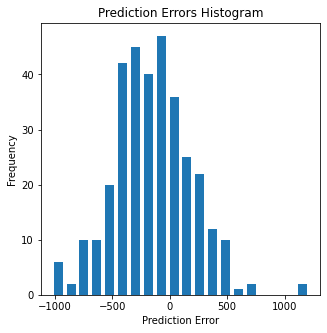
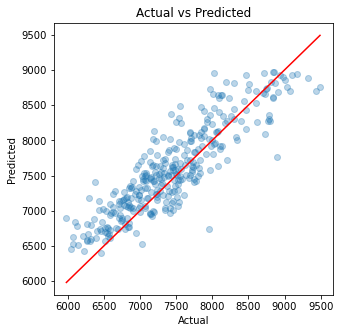


Figure 15: Actual vs Predicted values scatterplot (left) and Prediction Errors Histogram (right) for the XGBoost model

These plots show that the errors are approximately normal, and demand was often overestimated using the XGBoost model. We can also see this in [Figure 16](#Figure16) from the Appendix, where the predictions don’t drop as far as the actual values at the local minimums of the Actual Demand curve.

### 5.2.2 Feature Importance / XAI (with prior total demand variables)

To explain an XGBoost model, an F score was calculated for each feature, which counts the number of times a feature is split on in the model. The feature with the highest frequency is the most important. Feature importance was illustrated using the “plot\_importance” function from “xgboost” (Appendix, [Figure 17](#Figure17)).

From this graph, we see that total electricity demand, humidity, windspeed, solar exposure, average temperature, and RRP values from the day before have a strong influence on day-ahead predictions. Month and day also have a strong influence. Holidays clearly have the least influence, with the original variable and all lookback variables ranking last.

SHAP Mean Importance SHAP Summary Plot

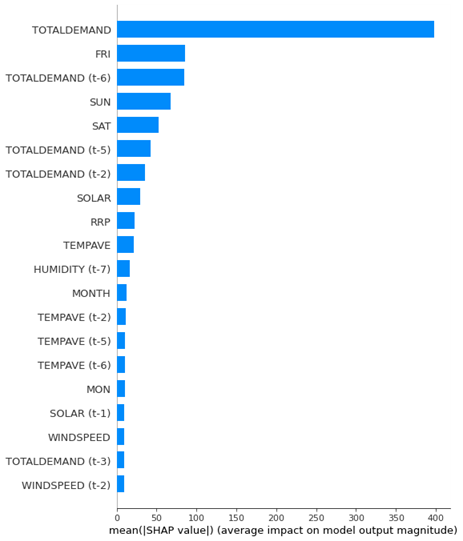
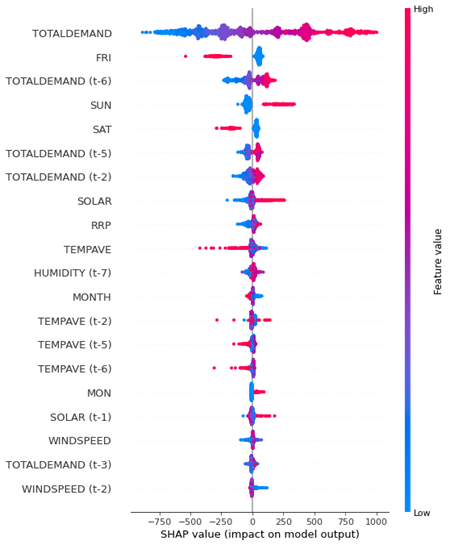
 

Figure 18: SHAP Mean Importance bar plot (left) and SHAP Summary Plot (right) for XGBoost model

When extrapolating the feature importance of the model via SHAP, prior total electricity demand (TOTALDEMAND) had by far the largest average absolute Shapley value compared to all other features, and much more pronounced compared to the F score feature importance analysis above as can be seen in the bar chart of mean importance (Figure 18).

Outside of TOTALDEMAND, we also note that the subsequent feature importance rankings do not share any similarity to the above F score analysis. We see Friday, 6-day prior TOTALDEMAND, Sunday and Saturday round out the Top 5 with the first weather related feature, amount of solar radiation (SOLAR), ranked 8th along with average temperature (TEMPAVE) ranked 10th in feature importance. HOLIDAY and solar panel electricity output (OUTPUT) clearly had little to no influence in the prediction as these two features did not appear on the mean importance chart. The same could be said for days Tuesday to Thursday, inclusive, and to a lesser degree the Humidity and Wind Speed features.

The summary plot above shows the direction and magnitude of each feature’s effect to the demand forecast output. TOTALDEMAND’s effect is evenly distributed across the full range of positive and negative impact scores to the model output. Looking at the dependence plot below (Figure 19), there is clearly a positive linear relationship between TOTALDEMAND and forecasted demand as high TOTALDEMAND values had a large positive impact to forecasted demand, whilst low TOTALDEMAND values had a large negative impact to forecasted demand.

SHAP Dependence Plots

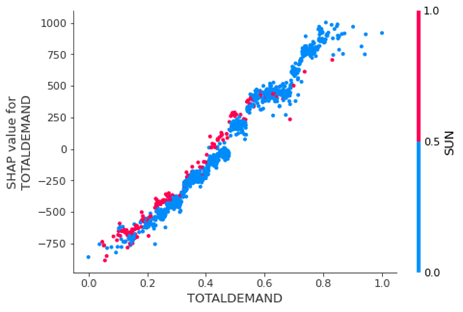
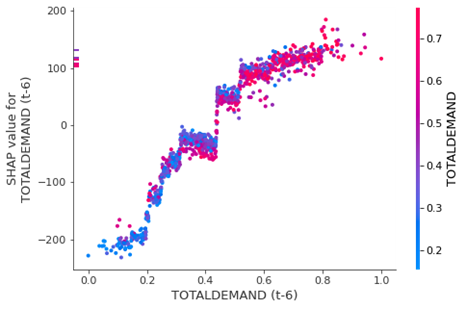
 

Figure 19: SHAP Dependence Plots for XGBoost model

In Figure 19, TOTALDEMAND (t-6) seems to follow a similar distribution, however, with a much smaller range of both positive and negative impact to demand forecast output. It seems that both Fridays and Saturdays had a negative impact on the forecasted demand, however, the opposite could be said for Sundays as this day gave a positive impact on the demand forecast.

### 5.2.3 Feature Importance / XAI (without prior total demand variables)

When TOTALDEMAND is removed as a predictor, average temperature (TEMPAVE) becomes the most significant contributor to the model prediction. We note that Friday (4th) and Saturday (2nd) remained in the Top 5 by feature importance and had a similar negative, although more pronounced, impact on the forecasted demand (Figure 20).

|  |  |
| --- | --- |
| **SHAP Mean Importance** | **SHAP Summary Plot** |
|  |  |

Figure 20: SHAP Mean Importance bar plot (left) and SHAP Summary Plot (right) for the XGBoost model without TOTALDEMAND

From the F score analysis (Appendix, [Figure 21](#Figure21)), we see a completely different list for feature importance rankings with day, humidity, solar and wind speed having the strongest influence on day-ahead predictions. It can be noted, however, that the average temperature, day before RRP values and the month both appear to have a strong contribution to forecasted demand as these features were among the top by feature importance for both the F score analysis and for their mean SHAP values.

HOLIDAY and solar panel electricity output (OUTPUT), once again, had little to no influence to the forecasted demand prediction as can be seen for both the F score analysis and mean SHAP values computed.

In the TEMPAVE dependence plot (Figure 22), it is interesting to see that there is a large positive impact to the forecasted demand when the temperature is either very low or very high. We can also see that around the median temperature, there is a slight negative impact to forecasted demand. As for the electricity pricing (RRP), there seems to be a logarithmic relationship between the electricity pricing and its impact on forecasted demand. For low prices, the forecasted demand is impacted negatively, however, for large prices forecasted demand is impacted positively by this feature.

**SHAP Dependence Plots**

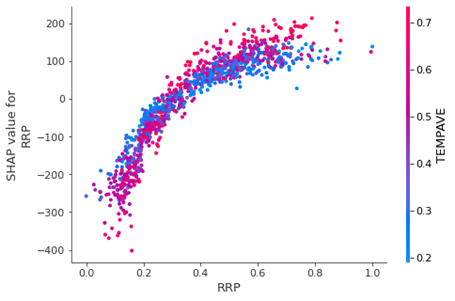
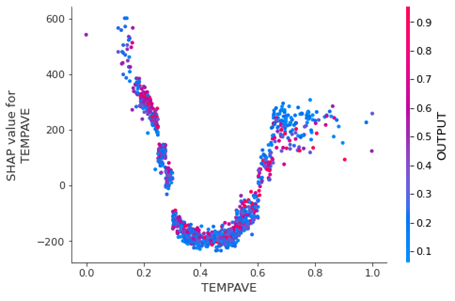


Figure 22: SHAP Dependence Plots without TOTALDEMAND for the XGBoost model

We do note, however, that removing the TOTALDEMAND, and all subsequent lag variables of TOTALDEMAND, produced an inferior model with an R Squared of 0.15 and a Mean Absolute Error of 565.99.

## 5.3 LSTM for Time Series Analysis

The second model used in this report was an LSTM (Long Short-Term Memory) model, for its time series analytical properties.

### 5.3.1 Modelling and optimisation

After the data was cleaned with 14 features including the target feature, the one-hot encoder was used to encode the “WeekDay'' feature to separate it into 7 columns corresponding to the 7 days of the week.This technique helps to capture nonlinear relationships between the day of the week and the target variable TOTALDEMAND. The final data used for model training and test had 17 features plus 1 target variable.

Data was then split into training and test sets before normalisation. After that, sliding windows with a time step n=1 was applied to the data so as to facilitate the next day prediction of TOTALDEMAND. To prepare for LSTM training, the input data was also reshaped to a 3D tensor format at this step. Note that lagged features were not engineered for this model, as the LSTM model is intrinsically designed to determine the best auto-correlations automatically.

### 5.3.2 Manual Model Training Approach

A Long Short Term Memory (LSTM) model was built first as a simple, single layer with a small number of hidden units, the default activation function (ReLU) using the common RMSprop optimizer and a set learning rate of 0.001. The result was initially not impressive, with a high loss MSE and low R-squared which indicated underfitting. The model was then developed further to overcome initial limitations by:

* Increasing model complexity by adding more layers
* Adding activation functions ‘tanh’ or ‘relu’ on each layer
* Adding dropout layers with a dropout rate of 0.2 after each layer to prevent overfitting
* Changing the optimizer to Adam to enhance model convergence and performance

The model hyperparameters were then tuned through experimentation with three different methods: Trial-and-Error, GridSearchCV and Bayesian Optimization. We called them respectively LSTM Model V1, LSTM Model V2 and LSTM Model V3.

### *5.3.3 Model Version 1*: Manual Optimisation

Refer to file ‘LSTM Modelling V1.0 - No GridSearchCV.ipynb’ within the Github repository

This final V1 model was constructed with 4 layers and 1 Dense output layer.

It is noted in [Table 14](#Table14) from Appendix that a Mean Absolute Error of 0.066 is relatively low while an R-squared of 0.7733 is quite high and demonstrates that the model explains about 77.33% of the variance in the target variable. These metrics indicate that the overall performance of this LSTM model is pretty good in predicting the electricity demand for the next day.

In [Figure 23](#Figure23) from the Appendix, the training converged within 50 epochs and both training loss and validation loss decreased steadily with each epoch showing that the model improved its performances on both the training and validation sets to similar extents.

The validation loss is slightly lower than the training loss, indicating there may be some noise in the data.

The general trend in [Figure 24](#Figure24) from the Appendix illustrates that the LSTM model predicts the electricity demand quite well since the LSTM Model Predicted Demand is quite close to the Supplied Forecast Demand. However, there are some points of time, where the error or differences between the two are recognisable.

### 5.3.4 Model Version 2: GridSearch

Refer to file ‘LSTM Modelling V2.0 - With GridSearchCV.ipynb’ on Github repository

In this version of the LSTM model, the basic structure of the LSTM model keeps the same with 4 layers and 1 Dense output layer.

GridSearchCV technique has been applied to find the best possible hyperparameters for the designed LSTM model. The runtime turn-around for this search is approximately 12 hours due to the model complexity as well as the variety of searched parameters.

The original searched values for each hyperparameter is 3. However, the searching took up to 3 days to complete. Therefore, some of the gridsearch values are reduced to 2 values only to reduce the searching time.

The search on hyperparameters produced results (Appendix, [Table 15](#Table15)) which are pretty close to hyperparameters in Model V1 (Appendix, [Table 14](#Table14)). The R-squared is slighter lower and Mean Absolute Error is slightly higher. Overall, this Model V2 also performs quite well.

As can be seen in [Figure 25](#Figure25) [and 26](#Figure26) from the Appendix, LSTM optimization using GridSearchCV method yields similar results to the trial-and-error method in Model V1 which performs well. Nevertheless, taking the very long runtime into account, this approach is not really effective.

### 5.3.5 LSTM Model Version 3: Bayesian Optimisation

Refer to the files ‘LSTM Modelling V3.0 - Bayesian Optimisation (With Total Demand).ipynb’ and ‘LSTM Modelling V3.0 - Bayesian Optimisation (Without Total Demand).ipynb’ on Github repository

The optimal set of hyperparameters for this model was finally arrived at through a mixture of Bayesian Optimisation (Appendix, [Table 16](#Table16)) and manual tweaking. This resulted in a 5-layer model, with 3 hidden layers sandwiched between the input and hidden layers.

The runtime for the Bayesian Optimisation process averaged between 40 minutes and two hours when run on the GPU-equipped machine.

The final model definition can be found in [Table 17](#Table17) in the Appendix.

The best learning\_rate parameter was found to be 0.001039. The number of epochs settled on was 50 as well as a batch size of 64. These final two parameters were arrived at using trial and error - repeatedly running the Bayesian Optimisation code with a batch size of 16, 32 and 64, and noting where the model began to converge in Figure 27.

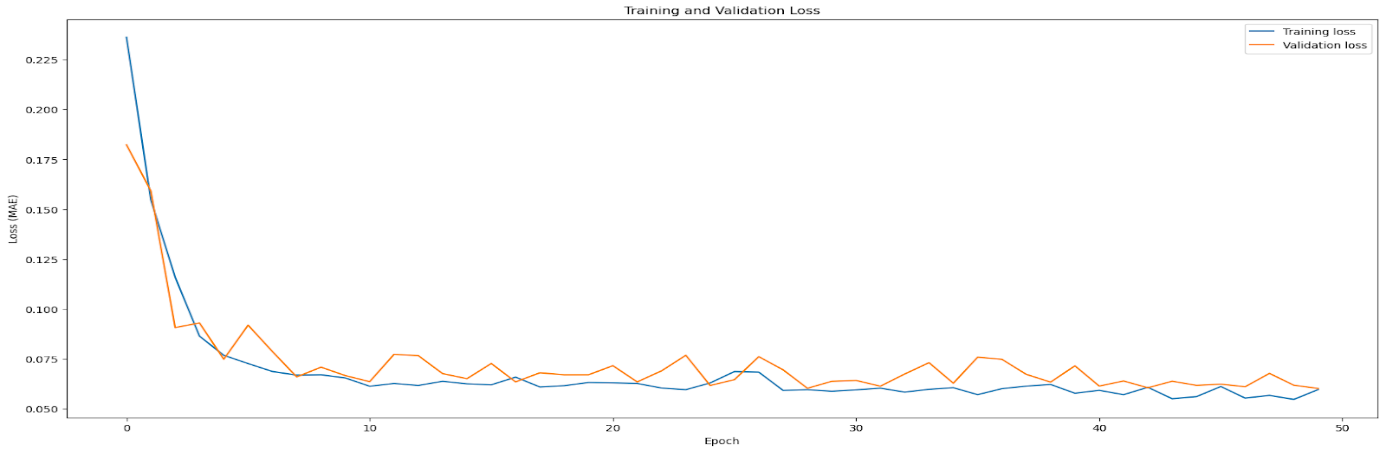
****

Figure 27: Training vs validation loss for each epoch

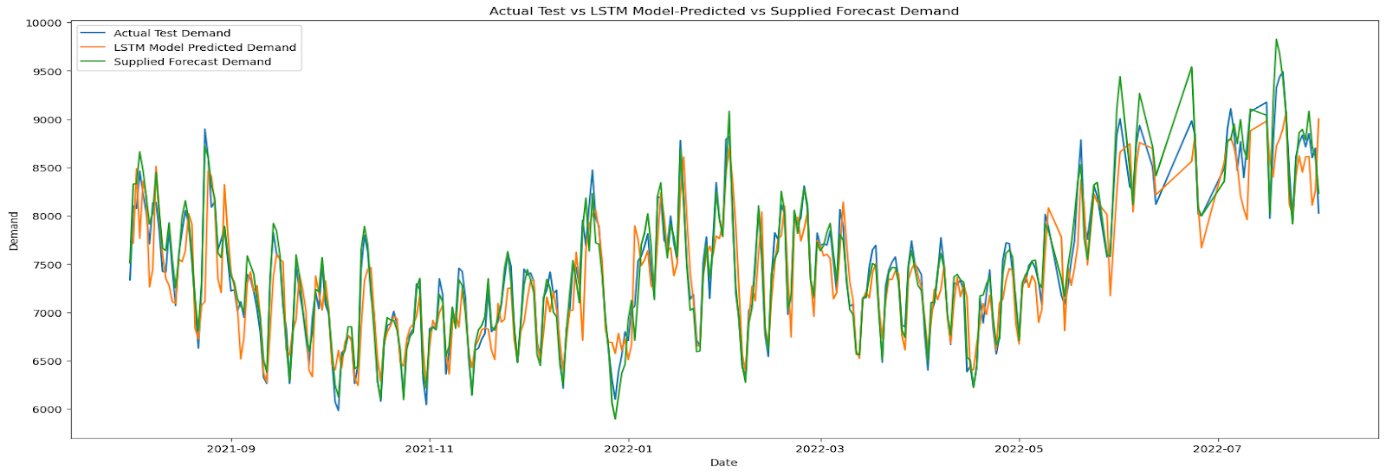
****

Figure 28: Actual Test Demand vs LSTM Model-Predicted Demand vs Supplied Forecast Demand time series plot

The comparison has been made among the 3 methods of building and optimising the LSTM model (Figure 28). It is clear that the LSTM model with Bayesian Optimisation brings about the best model performance (MAE: 0.0601, R2: 81%) with a moderate runtime of approximately 1 hour and 50 minutes. Therefore, through the three experiments, the LSTM Model V3 was chosen as the final version and used for further analysis.

To further confirm the final LSTM model was optimal, a number of comparative models were run for different ranges of sliding windows (1 day look-back, 3 days look-back, 5 day look-back, 7 day look-back and 15 day look-back) to ensure the model was able to look back sufficiently into the past to learn the patterns in the data. This additional confirmation step was conducted because it was felt that just using the day-before data may not have been enough for the model to understand the auto-correlated relationships. Similar to using the lagged variables in the XGBoost model, 5 models using different sliding windows were coded and run. The results of these executions for the final LSTM model are outlined in Table 18 in the Appendix.

As can be seen in Table 18 from the Appendix, the 1-day look-back window performed the best, and was therefore confirmed as the best LSTM model used for subsequent analysis.

### 5.3.6 Comparing the LSTM forecast to the University supplied forecasts

The final selected LSTM model tended to slightly overestimate electricity demand, whereas the supplied forecast values tended to slightly underestimate them.

When the average differences in forecast were compared (Figure 28 and 29) among the , the LSTM model over-estimated with an average actual difference of around 74 megawatts of electricity, whereas the supplied forecast under-forecasted by about 20.5 megawatts of electricity (Appendix, [Table 19](#Table19)).

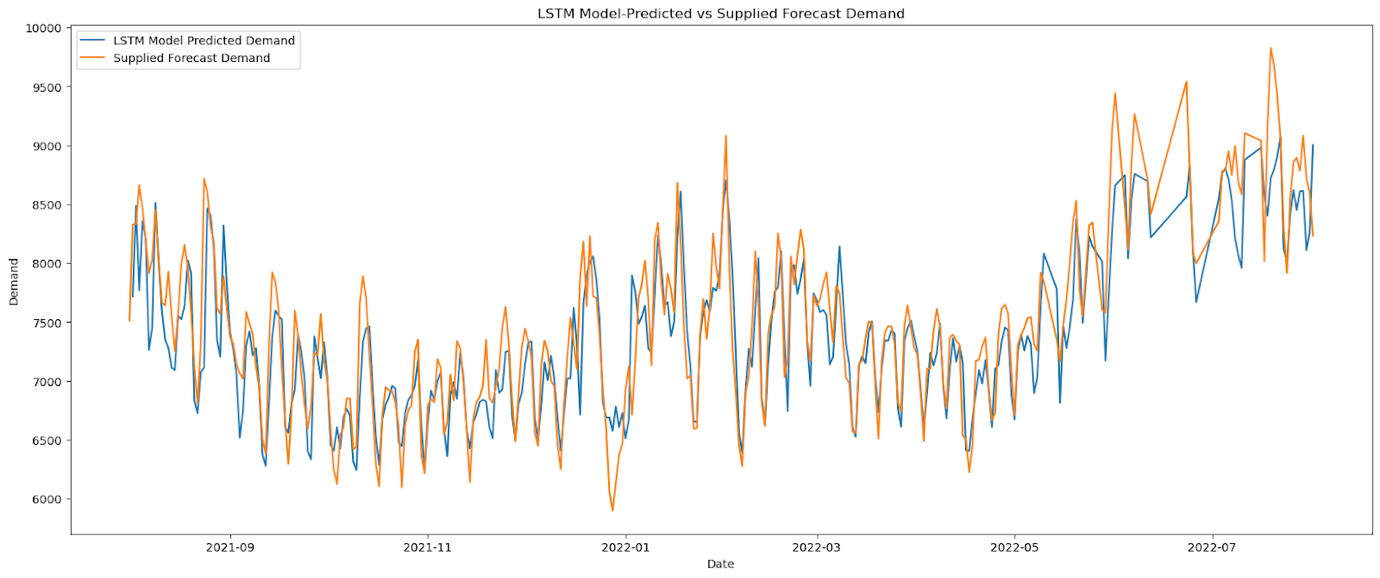


Figure 29: LSTM Forecast Demand vs Prior Model Forecast Demand time series plot

***5.3.7 Prediction Error***

To further assess the errors in prediction by the LSTM model, two graphs were created (Figure 30). The first graph plots actual demand against predicted demand, and the second graph displays error distribution.

|  |  |
| --- | --- |
|  |  |

Figure 30: Actual vs Predicted values scatterplot (left) and Prediction Errors Histogram (right) for the LSTM model

These plots show that the errors are approximately normal, and demand was slightly overestimated using the LSTM model. We can also see this in the time series plot (Appendix, [Figure 31](#Figure31)), where the predictions don’t drop as far as the actual values at the local minimums of the Actual Demand curve.

### 5.3.8 Feature Importance / XAI (with prior total demand variables)

Since SHAP analysis has not yet been developed for LSTM neural networks, we were forced to adopt an alternate method to discover feature importance for the LSTM model.

We utilised the Approximation Model method also used by Bouktif et al (2018) and Nakagawa et al (2019) in order to interpret their LSTM models, which allows for the reverse engineering of the results of the LSTM model using a simpler model type. For this study, we selected a Random Forest Regressor model. Although this approach is likely to miss some of the sophistication inherent in the LSTM, it was thought to still give a near-enough approximation of the importance each of the features contribute to the model.

By utilising the predictions obtained from the LSTM model as the dependent variable for the random forest regressor model, we were able to train this simpler model to predict the electricity demand forecast and then utilise the Python shap library on it to derive the relative feature importance scores.

The random forest regressor used was from Scikit-Learn, and was trained on the first 10 months of predicted electricity demand between August 2021 and May 2022 inclusive. The final two months of predicted electricity demand between June and July 2022 (inclusive) were used as the testing set.

This data was scaled using the MinMaxScaler, modelled using the RandomForestRegressor and trained against the training data set, and then tested against the testing data set.

An R squared score of 0.81 indicated that the regression model made a moderate approximation of the LSTM model.

|  |  |
| --- | --- |
| **SHAP Mean Importance** | **SHAP Summary Plot** |
|  |  |

Figure 32: SHAP Mean Importance bar plot (left) and SHAP Summary Plot (right) for the LSTM model

You can see in the mean SHAP importance chart in Figure 32, prior total electricity demand (TOTALDEMAND) is the obvious dominant feature contributing to the LSTM model’s forecasted demand output. This is a similar finding to the XGBoost model, however, the LSTM model is more heavily reliant on TOTALDEMAND and the subsequent feature importance rankings and magnitude differ greatly. We see humidity, amount of solar radiation (SOLAR), average temperature (TEMPAVE), and recommended retail electricity price (RRP) round out the Top 5. It also appears that holidays have barely any effect on the model predictions, nor do the other days of the week outside of perhaps Sunday.

The summary plot in Figure 32 shows the direction and magnitude of each feature’s effect to the predicted demand forecast output. TOTALDEMAND’s effect is evenly distributed across the full range of positive and negative impact scores to model output. Looking at the dependence plot in Figure 33, there is clearly a positive linear relationship between TOTALDEMAND and forecasted demand. We do note that the XGBoost model showed the same positive linear relationship.

|  |  |
| --- | --- |
| **SHAP Dependence Plots** | |
|  |  |

Figure 33: SHAP Dependence Plots for the LSTM model

In Figure 33, humidity seems to have a positive linear relationship to the LSTM’s model output, however, we do note that the relationship was weak and although humidity ranked 2nd by feature importance, this feature had a small impact on model output due to the over reliance of the model to TOTALDEMAND.

### 5.3.8 Feature Importance / XAI (without prior total demand variables)

When prior total electricity demand (TOTALDEMAND) is removed as a predictor, the month of the year (MONTH), the weekend (SAT & SUN), solar panel electricity output (OUTPUT) and recommended retail electricity price (RRP) become significant contributors to the forecasted demand (Figure 34). This is then followed by weather related features with humidity, average temperature (TEMPAVE), wind speed and amount of solar radiation (SOLAR) making notable, but smaller, contributions to forecasted demand.

Consistent with all models experimented in this study, the feature holiday has little to no contribution to the model output regardless of model type or any changes made to model feature selection.

|  |  |
| --- | --- |
| **SHAP Mean Importance** | **SHAP Summary Plot** |
|  |  |

Figure 34: SHAP Mean Importance bar plot (left) and SHAP Summary Plot (right) without TOTALDEMAND for the LSTM model

|  |  |
| --- | --- |
| **SHAP Dependence Plots** | |
|  |  |
| **image** | **image** |

Figure 35: SHAP Dependence Plots without TOTALDEMAND for the LSTM model

Figure 35 contains a series of SHAP dependence plots without total demand from the previous day. In the average temperature (TEMPAVE) dependence plot, it is interesting to see that there is a large positive impact to the forecasted demand when the temperature is either very low or very high. We can also see that around the median temperature, there is a slight negative impact to forecasted demand. This relationship between temperature and forecasted demand output can also be seen in the experimented XGBoost model (without TOTALDEMAND).

As for the electricity pricing (RRP), there seems to be a linear relationship between electricity price and its impact on forecasted demand. For low prices, the forecasted demand is impacted negatively, however, for large prices forecasted demand is impacted positively by this feature. Humidity, followed a similar but weak linear relationship which saw higher humidity scores had a positive impact on the forecasted demand.

Finally, it is evident that there is a strong relationship between day of the week and model output. You can see in the dependence plot that the early days of a week, between Monday and Thursday, have close to no impact on forecasted demand. This is also evident in the mean importance and summary plot where these days were ranked near the bottom. There is a clear and significant contribution made to forecasted demand, however, for later days of the week and the weekend. Saturdays and Sundays are in the Top 3 for feature importance, with Friday further down in ranking although ahead of Mondays to Thursdays inclusive. There is an upward trend evident in the dependence plot, with a notable positive impact to model forecasted demand, from Friday and especially Saturday and Sunday.

Like the XGBoost model, we do note that removing the TOTALDEMAND predictor from the LSTM model did produce an inferior model with an R Squared of 0.428 and a Mean Absolute Error of 0.1081.

# 6 Discussion

The original idea for this paper was to analyse how two effective electricity demand models, intended for use in the electricity industry, compare in the way the features used to train them individually impact on the model output. The intention was to use a relatively under-used methodology within data science called SHapely Additive exPlanations (SHAP), to explain how these various features contribute to the model outcome. It was thought that in addition to the relative poverty of model explanations in the literature regarding time series forecasting models, and in machine learning and artificial intelligence research in general, that we would apply this game theory-derived approach to an electricity demand forecasting problem, and in this way contribute to the slowly growing sub-field of model explanation.

Explaining machine learning and AI models is seen as becoming increasingly relevant in the research and commercial worlds, as customers increasingly question why a particular model has decided to predict or forecast in a certain way. No longer content with a ‘black-box’ approach where magic was seen to be done utilising algorithmic wizardry, which is trusted  implicitly merely due to the mathematical sophistication of the model, clients now demand more from their high-end analytics teams.

The decision to focus on SHAP was due to its ability to generalise explanations across the whole model instead of just concentrating on explaining how a single prediction or group of predictions are generated. It was also proposed as a model-agnostic explanation method, and so should have worked with any model. However, whilst we were able to utilise the Python shap library for XGBoost, the current version of this library does not appear to function with LSTM time-sequential models. Neither an attempt to reach the author of this software, Scott Lundberg of the University of Washington nor an attempt to reach Jakub Bialek of the Warsaw University of Technology, an author of a study (Bialek, 2022) which claimed to have used the shap library with a non-specified neural network were answered before submission of this paper. It was hoped one of these academics would be able to assist us to get shap working.

Consequently, we turned to using a model approximation method that others (Bouktif et al,  2018 and Nakagawa et al, 2019) have used in an attempt to explain models too sophisticated to use with the shap library. In this method a simpler (non-neural network) model is built over the same feature set as the neural network, but it uses the predicted variable as the training variable, which is forecast electricity demand in our case. Enough predicted values is required to be generated by the LSTM to feed into the approximation model, and in our case we had almost a year of forecast data which we believe is sufficient. We decided to use a random forest regressor for the approximation model, and when we ran the shap code against the resultant model, we were then able to compare the generalised feature contributions to the XGBoost model to that of the LSTM model.

## 6.1 Data Quality

With respect to our data cleaning approach, we opted to remove most outliers. As there were not so many in our data, and outliers typically have the potential to hinder models, this seemed like a viable option at the time. In hindsight, this approach likely had a negative effect on our results. We realised that removing a row removes an entire day from our dataset, which leads to information loss. This could also negatively impact lagged variables and sliding windows.By removing a given record, the lags generated in relation to downstream records would be an extra day removed than would otherwise have been thought. In future, we would recommend alternative and more advanced techniques to handle outliers, such as linear or seasonal interpolation, which are methods of interpolating seasonal data from neighbouring future and past timesteps. This could be implemented by using the Python library SciPy’s interpolate function (Analytics Vidhya, 2021).

Another concern we had was regarding the weather data and how it has only been sourced from Bankstown Airport and Sydney Airport with the assumption that these locations are representative of the entirety of NSW. Our recommendation would be to extend the study by gathering data that represents different regional areas across NSW, which could have a positive influence on model output.

## 6.2 XGBoost

Lagged variables were used to implement XGBoost for time series analysis, which produced reasonable results. The model’s performance would likely be improved by implementing a different approach, such as a sliding window method. This method was attempted initially, but we found that it required a higher level of coding experience for us to implement. While we would have preferred this method, it ultimately wasn’t a feasible option due to time restrictions. Using lagged variables was found to be an acceptable approach for time series analysis according to our literature review, so we opted for this method instead.

The number of lookback days were experimented on. We compared a baseline model with no lookback variables to a range of models containing 1 to 7 days lookback variables. We found that a model with 7 days lookback had the best performance, and a model with 3 days lookback had the worst performance. 3 days lookback may have worsened model performance as it does not look back far enough to identify significant trends, whereas 7 days lookback might be identifying weekly trends.

We would recommend future researchers who are interested in the lagged variables approach to experiment with further lookback than 7 days and to observe the effects this has on model performance. In our case, due to computational limitations and time restrictions, we stopped testing at 7 days. Although, when experimenting on further lookback, it’s important to take into consideration the number of features being created. Each day of lookback added, exponentially increases the dataset dimensions. This leads to the “curse of dimensionality” and overfitting, which occurs when analysing an extreme number of features. In this case, we’d recommend implementing a feature projection method called Principal Component Analysis (PCA) (---UNSW Ref---). PCA reduces dataset dimensions by handling redundancies and maximising variance between data points (---Harvard Ref---). A negative side-effect is that this method involves a data transformation, which could make explainability more difficult, unless you are an expert in your field. PCA wasn’t implemented in this study due to time restrictions and limited expertise. Another method that can be considered is feature subset selection, which selects a subset of features, determined by feature relevancy, to be used in the model (---UNSW Ref---).

The results also showed that while the predictions matched the patterns of actual demand, they were often overestimating. This could be due to data quality or model complexity issues. Experimenting with our previous suggestions might have an impact on model performance and may reduce overestimation.

## 6.3 LSTM

As outlined in the literature review, Long Short-Term Memory (LSTM) network models are a form of recurrent neural network that have become increasingly popular for time series problems. The literature also highlights that there are relevant examples of LSTM being used in the electricity industry.

Due to its recurrent nature, LSTMs require the generation of sliding windows in the data to ensure the model is able to look back at autocorrelated variable values to different points in the past. As such, the data preparation work involved was similar to the XGBoost model to ensure that day-ahead forecasts could be generated, though instead of lagged variables, sliding windows were generated. The data also had to be cleaned and reshaped for processing, and similar to the data ingested by the XGBoost model, one-hot encoding and value standardisation were also applied.

Due to the number of possible hyperparameter combinations, we turned to a Bayesian Optimisation approach to find the optimal configuration for the model. Executing on a GPU-equipped machine, this process took between 30 minutes and 2 hours for each run, which was superior to a GridSearchCV approach, which took up to 3 days to complete, and significantly more effective than manual hyperparameter selection. There was still a manual element to the model design in order to determine how many layers the LSTM model would perform best with. This was done by running the optimisation multiple times with 2, 3 and 4 neural network LSTM layers. A similar process was run to determine how best to set the batch sizes and number of epochs.

In addition, a number of versions of the LSTM model were generated that utilised various ranges of sliding windows. This was done to take advantage of the linear sequential attributes of the LSTM algorithm. The sliding window sizes used were for 1, 3, 5 and 15 days in the past. As it turned out, the most accurate window size was for 1 day, which is what we settled upon.

As the model derived through a mixture of Bayesian and manual optimisation performed the best, it was selected as the final LSTM model to compare to the XGBoost model. The results of applying this model indicated that the previous total electricity demand was by far the greatest contributor to the next-day demand predictions. This accords with previous work that found similar results.

## 6.4 Model Explainability / XAI

The results from both the F score feature importance analysis and the SHAP analysis saw that the most important feature for the XGBoost model was total electricity demand from the previous day. Evidently, the SHAP analysis of the Approximation Model over our LSTM model also found that total electricity demand from the previous day was the largest contributor to the day-head demand forecast output, although the reliance on this feature was much more pronounced than the XGBoost model.

Weather related features contributed to a lesser degree, particularly in the XGBoost model, which is rather surprising in light of all we have read about the effect of temperature on electricity demand in the literature. This could mean that the model is preferentially relying on other correlated measures, or that the models are incorrectly configured in some way. However, we discovered a study by Bouktif et al (2018) who found that temperature contributed to an insignificant degree when compared to previous electricity demand lags..

It was evident from both models that there was a clear positive linear relationship between total electricity demand from the previous day and the day-head demand forecast output. This is not surprising as it is obvious that total electricity demand today would have some correlation with the expected electricity demand tomorrow. The same can be said for demand up to 6-days prior, however, the correlation would be to a lesser extent.

As there was an over reliance on total electricity demand from the previous day for both models, we found that we had largely mixed results when analysing the contribution of all other features. It was found that when total electricity demand was removed from both models, the feature importance and overall contributions changed drastically. We found that there was the possibility of using parallel features to produce the forecasts, however, when this total electricity demand was removed from both models, a less effective but possibly improvable model was generated, which relied instead on the full gamut of exogenous variables used in this study. Unfortunately, due to time restrictions we were not able to concentrate on fully improving this model, despite being somewhat successful through applying the Bayesian Optimisation approach.

With the total demand variable removed, the XGBoost model had the average temperature as the most significant contributor to the model’s demand predictions. To a lesser extent, the LSTM model had several weather related features making notable contributions to the model’s output. These features include humidity, average temperature, wind speed and amount of solar radiation. This is in line with the literature where Tribble (2003), in her PhD thesis, highlighted that “weather has a significant impact on load demand and load forecasting” and named temperature, apparent temperature, relative humidity, solar radiation and wind speed in her findings.

The relationship between the average temperature and demand forecast output, for both models, was non-linear and showed elevated demand forecasts for the days at the two extremes of the temperature distribution. This is also in line with the literature with Manderson and Considine (2021), who found the same relationship in their study between monthly electricity consumption for the residential and commercial sectors and temperature.

It was evident in both models that certain calendar related features had notable contributions to the model’s forecast demand prediction. Both the XGBoost and the LSTM model had the weekend as significant contributors, with the month of the calendar year the top contributor for the LSTM model and the weekend, both Saturdays and Sundays, in the top 3 for overall contribution to the prediction for the XGBoost model. This is also in line with the literature as noted in Shah, Iftikhar and Ali’s 2022 study who emphasised the importance of including yearly, seasonal and calendar effects to efficiently forecast electricity demand and prices. Interestingly, holidays were found to be the least important feature for both models. A possible influence could be that each holiday only occurs once and at a specific time per year, so the data is not substantial enough for it to have a significant effect on each model. We would also recommend avoiding holiday lag variables, as this intuitively shouldn’t have any effect on the model.

As another recommendation for future researchers, It would be interesting to observe model performance if the least important features were removed from the model. At least it should be beneficial to reduce the dimensions of the dataset if concerned about memory, computational limitations, and the “curse of dimensionality”, as mentioned earlier..

### 6.4.1 XAI for XGBoost

The results of the XGBoost feature importance analysis showed that the most important feature was total electricity demand from the previous day, followed by average temperature, solar radiation, day of the week, month of the year and recommended retail price. Upon reflection, the model could have been improved if we had utilised day-ahead features for holiday and weekday variables. In this case, feature importance analysis might produce a different result. Additionally, the results of feature importance might differ if PCA were to be implemented, as suggested previously.

### 6.4.2 XAI for LSTM

There is a degree of concern within the study that we did not have sufficient time to complete the study to the degree of rigour we would have liked. Due to the large number of moving parts involved in bringing the study to fruition, including the gathering of exogenous data, conducting the literature review, cleaning and feature engineering the data, selecting and coding the models to be compared, researching and applying explainability methods to the models, and conducting the analysis and write-up, we feel that we were not able to conduct as much testing as we should have. This was also undoubtedly a major factor in explaining the results we obtained.

# 7 Conclusion and Further Issues

The results of this study show that the XGBoost and LSTM models are effective tools for forecasting short-term (day-ahead) electricity demand, with the LSTM model being somewhat more accurate than XGBoost for this time series formatted data.

The study also revealed that the way each model utilises each of the data features internally is unique, with different ordering of those features that contribute to generating the model output revealed by explainability analysis.

Interestingly, we found that although the models both relied heavily on the past, auto-correlated values of the dependent variable - TOTALDEMAND, they were also able to be generated without using this feature. By the time we concluded this study we had managed to generate both an XGBoost and LSTM model that completely excluded total electricity demand as a predictor. Although the accuracy of these models was rather low, it appears that in future work there is much more that could be done to improve them. The LSTM model appears to have more potential here, but given our time limitations this remains to be seen. Whilst analysing these results it was also apparent that once the overarching effect of TOTALDEMAND was absent, the models relied on the remaining exogenous features on which they were trained differently to when TOTALDEMAND was present.

For example, when trained with TOTALDEMAND, the LSTM model prioritised humidity, then solar radiation, average temperature and then recommended retail price. Without TOTALDEMAND as a predictor, the model prioritised the current month, Sundays and Saturdays, solar panel electricity output, recommended retail price and then humidity. Only then did it find the average temperature to make a contribution. More work definitely needs to be done in this area, especially to learn in which circumstances the features are prioritised to contribute to a model, and in which circumstances they are relied upon less.

We feel the time limitation for this project did affect the extent to which we were able to test and refine each model, and consequently the accuracy of each is not what we believe we could have achieved. For instance, we could also combine the multiple LSTMs and train on different subsets of data, using different feature sets. In addition, our expertise in coding these models was also a factor, and we would have ideally liked to spend some time training in the further enhanced ways to prepare the data and construct the models.

In retrospect, we would have approached the data preparation slightly differently. Instead of merely deleting the outliers, the time series nature of the data requires that we would now recommend alternative and more advanced techniques of handling them in the future. Given the chance, we would instead apply linear or seasonal interpolation, which interpolates data from neighbouring future and past timesteps of a variable.

Additionally, we assumed that the weather as captured for a particular location was generalisable across all of NSW. This is a highly untested assumption, and needs to be further tested. It is also likely that running similar studies for smaller, substation-scale local areas may be more accurate. By only including the service area related to a particular substation, the effect of the exogenous variables may be more apparent, especially when considering the differential consequences of supplying electricity to areas that are largely residential versus commercial versus industrial.

In terms of future work, we see much promise in tuning both the XGBoost and LSTM models, both to improve their accuracy and to apply other explainability methods to them to elucidate what exactly is going on. We are hopeful that explainability libraries that can be applied more broadly across both deep learning and lower-sophistication models will be developed and become more widely available within the Python ecosystem. The use of explainability in machine learning needs to be generally encouraged and become a core part of the data science curriculum so that new practitioners are aware of and can rise to the challenge of illuminating black box models for both their customers and fellow modelers.

References

Analytics Vidhya (2021) *Interpolation Techniques Guide & Benefits | Data Analysis (Updated 20*23)

Available at: <https://www.analyticsvidhya.com/blog/2021/06/power-of-interpolation-in-python-to-fill-missing-values/>

(Accessed: 12 April, 2023)

Belle, V. and Papantonis, I., 2021. Principles and practice of explainable machine learning. *Frontiers in Big Data*, p.39.

Białek, J., Bujalski, W., Wojdan, K., Guzek, M. and Kurek, T., 2022. Dataset level explanation of heat demand forecasting ANN with SHAP. *Energy, 261*, p.125075.

Bouktif, S., Fiaz, A., Ouni, A. and Serhani, M.A., 2018. Optimal deep learning lstm model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. *Energies, 11*(7), p.1636.

Boulaire, F., Higgins, A., Foliente, G. and McNamara, C., 2014. Statistical modelling of district-level residential electricity use in NSW, Australia. *Sustainability science, 9*, pp.77-88.

Divina, F., Garcia Torres, M., Goméz Vela, F.A. and Vazquez Noguera, J.L., 2019. A comparative study of time series forecasting methods for short term electric energy consumption prediction in smart buildings. *Energies, 12*(10), p.1934.

Energy.gov.au (2023) *Australian Energy Statistics, Table O Electricity generation by fuel type 2020-21 and 2021*

Department of Industry, Science, Energy and Resources

Available at: <https://www.energy.gov.au/publications/australian-energy-statistics-table-o-electricity-generation-fuel-type-2020-21-and-2021>

(Accessed: 10 April 2023)

Engle, R.F., Mustafa, C. and Rice, J., 1992. Modelling peak electricity demand. *Journal of forecasting, 11*(3), pp.241-251.

Fatema, I., Kong, X. and Fang, G., 2021. Electricity demand and price forecasting model for sustainable smart grid using comprehensive long short term memory. *International Journal of Sustainable Engineering, 14*(6), pp.1714-1732.

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation, 9*(8), pp.1735-1780.

Le, T., Vo, M.T., Vo, B., Hwang, E., Rho, S. and Baik, S.W., 2019. Improving electric energy consumption prediction using CNN and Bi-LSTM. *Applied Sciences, 9*(20), p.4237.

Lisi, F. and Shah, I., 2019. Forecasting next-day electricity demand and prices based on functional models. *Energy Systems, 11*(4), pp.947-979.

Lu, H., Cheng, F., Ma, X. and Hu, G., 2020. Short-term prediction of building energy consumption employing an improved extreme gradient boosting model: A case study of an intake tower. *Energy, 203*, p.117756.

Lundberg, S.M. and Lee, S.-I. (2017) “A Unified Approach to Interpreting Model Predictions,” Advances in Neural Information Processing Systems, pp. 4768–4777.

Maia-Silva, D., Kumar, R. and Nateghi, R., 2020. The critical role of humidity in modeling summer electricity demand across the United States. *Nature communications, 11*(1), p.1686.

Manderson, E. and Considine, T., 2021. *The Effect of Temperature on Energy Demand and the Role of Adaptation* (No. 2112). Economics, The University of Manchester.

Muratori, M., Schuelke-Leech, B.A. and Rizzoni, G., 2014. Role of residential demand response in modern electricity markets. *Renewable and Sustainable Energy Reviews, 33*, pp.546-553.

Nakagawa, K., Ito, T., Abe, M. and Izumi, K., 2019. Deep recurrent factor model: interpretable non-linear and time-varying multi-factor model. *arXiv preprint arXiv:1901.11493.*

Nia, A.R., Awasthi, A. and Bhuiyan, N., 2021. Industry 4.0 and demand forecasting of the energy supply chain: A literature review. *Computers & Industrial Engineering, 154*, p.107128.

Rai, A., Esplin, R., Nunn, O. and Nelson, T., 2019. The times they are a changin’: Current and future trends in electricity demand and supply. *The Electricity Journal, 32*(6), pp.24-32.

Schoen, C., 2005. A new empirical model of the temperature–humidity index. *Journal of Applied Meteorology and Climatology, 44*(9), pp.1413-1420.

Shah, I., Iftikhar, H. and Ali, S., 2022. Modeling and forecasting electricity demand and prices: A comparison of alternative approaches. *Journal of Mathematics*, 2022.

Son, H. and Kim, C., 2017. Short-term forecasting of electricity demand for the residential sector using weather and social variables. *Resources, conservation and recycling*, 123, pp.200-207.

Statista (2023) *Electricity consumption in Australia from financial year 2010 to 2022* Available at: <https://www.statista.com/statistics/986178/australia-electricity-usage/> (Accessed: 6 April 2023)

Tribble, A.N., 2003. *The relationship between weather variables and electricity demand to improve short-term load forecasting*. The University of Oklahoma.

Verwiebe, P.A., Seim, S., Burges, S., Schulz, L. and Müller-Kirchenbauer, J., 2021. Modeling energy demand—a systematic literature review. *Energies, 14*(23), p.7859.

Wu, K., Gu, J., Meng, L., Wen, H. and Ma, J., 2022. An explainable framework for load forecasting of a regional integrated energy system based on coupled features and multi-task learning. *Protection and Control of Modern Power Systems, 7*(1), pp.1-14.

XGBoost Documentation (2022) *XGBoost Parameters*

Available at: [XGBoost Parameters — xgboost 1.7.5 documentation](https://xgboost.readthedocs.io/en/stable/parameter.html) (Accessed:13 April 2023)

XGBoost Documentation (2022) *Introduction to Boosted Trees*

Available at: <https://xgboost.readthedocs.io/en/stable/tutorials/model.html> (Accessed:13 April 2023)

Zhao, X., Li, Q., Xue, W., Zhao, Y., Zhao, H. and Guo, S., 2022. Research on Ultra-Short-Term Load Forecasting Based on Real-Time Electricity Price and Window-Based XGBoost Model. *Energies, 15*(19), p.7367.

Kuo, P.-H. & Huang, C.-J. 2018, 'An electricity price forecasting model by

hybrid structured deep neural networks', Sustainability, vol. 10,

no. 4, p. 1280. Doi: 10.3390/su10041280

Appendix

Tables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Library | Version | Comments | | |
| OS |  | for working directory navigation and management | | |
| BeautifulSoup | 3.2.2 | for web scraping | | |
| Selenium | 4.8.3 | for web scraping using a chrome driver | | |
| 7-Zip |  | opens and reads content of zip files | | |
| Pandas | 1.5.3 | for easier data handling. Tasks include cleaning, merging, feature engineering. | | |
| NumPy | 1.23.5 | for easier handling of datasets. Similar functionality to the Pandas package | | |
| SciKit-Learn | 1.2.1 | for performing various machine learning tasks | | |
| Matplotlib | 3.7.0 | for graphs/visualisation tasks | | |
| Seaborn | 0.12.2 | for graphs/visualisation tasks | | |
| XGBoost | 1.7.5 | to build an xgboost model | | |
| Tensorflow | 2.11 | to build an LSTM model | | |
| Skopt | 0.9.0 | for hyperparameter tuning and global optimization | | |
| Shap | 0.41 | for analysing feature importance | | |
| Computer | Owner | Processor | RAM | OS |
| Dell Laptop | Mia | Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz 2.59 GHz | 8 GB | Microsoft Windows 10 Home |
| Dell Laptop | Mia | Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz | 32 GB | Microsoft Windows 10 Enterprise |
| Alienware Laptop | Daniel | Intel(R) Core(TM) i7-10750H CPU/GPU @ 2.60GHz 2.59 GHz | 32 GB | Microsoft Windows 11 Pro |
| iMac | Kevin | 3.5 GHz Quad-Core Intel Core i7 | 24 GB | MacOS Catalina |
| Acer Laptop | Christian | Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 2591 Mhz, 6 Core(s), 12 Logical Processor(s) | 8 GB | Microsoft Windows 11 Home |

Table 1: Python libraries and Computer Specifications

|  |  |  |  |
| --- | --- | --- | --- |
| Total Electricity Demand (1,323,398 rows) | | | |
| Variables | Data Type | Nulls | Additional details |
| DATETIME | Date time | 0 | Format: YYYY-MM-DD HH:MM:SS. Date range 2010-01-01 to 2022-08-01, all times 00:00:00. |
| REGIONID | Text | 0 | Categorical. Only one value: “NSW1”. Can be removed. |
| TOTAL DEMAND | Float | 0 | Units: megawatts (MW) |

Table 2: Total Electricity Demand data description

|  |  |  |  |
| --- | --- | --- | --- |
| Forecast Demand (11,619,503 rows) | | | |
| Variables | Data Type | Nulls | Additional details |
| DATETIME | Date time | 0 | Format: YYYY-MM-DD HH:MM:SS. Date range 2010-01-01 to 2022-08-01, all times 00:00:00. |
| REGIONID | Text | 0 | Categorical. Only one value: “NSW1”. Can be removed. |
| FORECASTDEMAND | Float | 0 | Units: megawatts (MW) |
| PREDISTPATCHSEQNO | Integer | 0 | Unique identifier of predispatch run. 10 digits, format YYYYMMDDPP |
| PERIODID | Integer | 0 | Period count. Range 1 to 79. |
| LASTCHANGED | Date time | 0 | Format: YYYY-MM-DD HH:MM:SS. Date time range 2009-12-30 12:31:49 to 2022-07-31 23:31:52. |

Table 3: Forecast Demand data description

|  |  |  |  |
| --- | --- | --- | --- |
| Air Temperature (247,646 rows) | | | |
| Variables | Data Type | Nulls | Additional details |
| DATETIME | Date time | 0 | Format: YYYY-MM-DD HH:MM:SS. Date range 2010-01-01 to 2022-08-01, all times 00:00:00. |
| LOCATION | Float | 0 | Categorical. Only one value: “94766.”. Can be removed. |
| TEMPERATURE | Float | 0 | Units: degrees celsius |

Table 4: Air Temperature data description

|  |  |  |  |
| --- | --- | --- | --- |
| Price and Demand (302,448 rows) | | | |
| Variables | Data Type | Nulls | Additional details |
| REGION | Text | 0 | Categorical. Only one value: “NSW1”. Can be removed. |
| SETTLEMENTDATE | Date time | 0 | Format: YYYY-MM-DD HH:MM:SS |
| TOTALDEMAND | Float | 0 | Units: megawatts (MW) |
| RRP | Float | 0 | Units: Australian dollar value |
| PERIODTYPE | Text | 0 | Categorical. Only one value: “TRADE”. Can be removed. |

Table 5: Price and Demand data description

|  |  |  |  |
| --- | --- | --- | --- |
| Weather in Bankstown (371,126 rows) | | | |
| Variables | Data Type | Nulls | Additional details |
| HUMIDITY | Text | 52 | Mostly integer values with “%” which must be stripped. Contains remaining error values contain unknown strings which must be removed. |
| WINDDIRECTION | Text | 0 | Non-alphanumeric. Only one character. Can be removed. |
| TEMP | Text | 0 | Integer values with “°C” which must be stripped. |
| TIME | Time | 0 | Half-hourly, 24 hours. |
| BAROMETER | Text | 0 | Not required. Can be removed. |
| WEATHER | Text | 0 | Not required. Can be removed. |
| WINDSPEED | Text | 45 | Integer values with “km/h” which must be stripped. |
| DATE | Date | 0 | Format: d Mmmm YYYY, Date range 1 January 2016 to 31 December 2022. |

Table 6: Weather in Bankstown data description

|  |  |  |  |
| --- | --- | --- | --- |
| Solar Panel Installations (302 rows) | | | |
| Variables | Data Type | Nulls | Additional details |
| Small Unit Installation Postcode | Integer | 0 | Only one value “2200”. Can be removed. |
| Variable | Date | 0 | Month and year. Range Jan 2010 to Dec 2022. |
| Output | Float | 0 | SGU rated output of solar panels. Units: kW (kilowatts) |
| Quantity | Integer | 0 | Number of solar panels installed |

Table 7: Solar Panel Installations data description

|  |  |  |  |
| --- | --- | --- | --- |
| Solar Exposure (4,816 rows) | | | |
| Variables | Data Type | Nulls | Additional details |
| Product Code | Text | 0 | Only one value “IDCJAC0016”. Can be removed. |
| Bureau of Meteorology station number | Integer | 0 | Only one value “66137”. Can be removed. |
| Year | Integer | 0 | Range: 2010 to 2022 |
| Month | Integer | 0 | Range: 1 to 12 |
| Day | Integer | 0 | Range: 1 to 31 |
| Daily global solar exposure | Float | 2 | Units: MJ/m^2 (megajoules per square metre) |

Table 8: Solar Exposure data description

|  |  |  |  |
| --- | --- | --- | --- |
| Min Temperature (4,830 rows) | | | |
| Variables | Data Type | Nulls | Additional details |
| Product Code | Text | 0 | Only one value “IDCJAC0011”. Can be removed. |
| Bureau of Meteorology station number | Integer | 0 | Only one value “66137”. Can be removed. |
| Year | Integer | 0 | Range: 2010 to 2022 |
| Month | Integer | 0 | Range: 1 to 12 |
| Day | Integer | 0 | Range: 1 to 31 |
| Minimum temperature | Float | 17 | Units: degrees celsius |
| Days of accumulation of minimum temperature | Integer | 17 | Only one value “1”. To be removed. |
| Quality | Text | 18 | Categorical. Values “Y” and “N”. Not required. Can be removed. |

Table 9: Min Temperature data description

|  |  |  |  |
| --- | --- | --- | --- |
| Max Temperature (4,816 rows) | | | |
| Variables | Data Type | Nulls | Additional details |
| Product Code | Text | 0 | Only one value “IDCJAC0010”. Can be removed. |
| Bureau of Meteorology station number | Integer | 0 | Only one value “66137”. Can be removed. |
| Year | Integer | 0 | Range: 2010 to 2022 |
| Month | Integer | 0 | Range: 1 to 12 |
| Day | Integer | 0 | Range: 1 to 31 |
| Maximum temperature | Float | 20 | Units: degrees celsius |
| Days of accumulation of maximum temperature | Integer | 20 | Only one value “1”. To be removed. |
| Quality | Text | 23 | Categorical. Values “Y” and “N”. Not required. Can be removed. |

Table 10: Max Temperature data description

|  |  |  |  |
| --- | --- | --- | --- |
| Public Holidays (823 rows) | | | |
| Variables | Data Type | Nulls | Additional details |
| Date | Date | 0 | Format: YYYYMMDD. Date range 20100101 to 20221231 |
| Holiday Name | Text | 0 | Not required. Can be removed. |
| Information | Text | 0 | Description of holiday. Not required. Can be removed. |
| More Information | Text | 82 | Web link. Not required. Can be removed. |
| Jurisdiction | Text | 0 | States applicable to. Some values contain multiple states separated by the “|” delimiter. |

Table 11: Public Holidays data description

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Minimum | Maximum | Mean | Median | Skew |
| HUMIDITY | 0.223448 | 0.9664 | 0.675363 | 0.68564 | -0.39138 |
| WINDSPEED | 8.756757 | 44.59524 | 20.15829 | 19.22087 | 0.820107 |
| TOTALDEMAND | 5983.135 | 9919.418 | 7839.728 | 7758.87 | 0.119798 |
| SOLAR | 1 | 32.2 | 15.98689 | 14.6 | 0.416857 |
| RRP | 0 | 163.23 | 68.13216 | 68.39348 | 0.492613 |
| FORECASTDEMAND | 5518.989 | 10751.96 | 7825.213 | 7732.721 | 0.299895 |
| OUTPUT | 12.32 | 458.355 | 132.0547 | 102.575 | 1.236534 |
| TEMPAVE | 5.85 | 34.4 | 18.52966 | 18.65 | 0.212889 |

Table 12: Data description of each feature

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | Results | | | | | | | |
| Hyperparameter | Limits | Values Searched | 0 days | 1 day | 2 days | 3 days | 4 days | 5 days | 6 days | 7 days |
| colsample\_bytree | (0,1] | 0.8, 0.9, 1 | 0.8 | 1 | 1 | 0.9 | 1 | 0.9 | 0.9 | 0.7 |
| gamma | [0,1] | 0, 0.01, 0.1, 0.5, 0.9, 1 | 0 | 0 | 0.5 | 0.1 | 1 | 0.9 | 0.9 | 0.7 |
| learning\_rate | [0,1] | 0.04, 0.05, 0.06, 0.07 | 0.06 | 0.06 | 0.05 | 0.06 | 0.05 | 0.05 | 0.05 | 0.05 |
| n\_estimators | [0,inf] | 120, 150, 160, 170, 180, 190, 250, 300, 310, 350 | 170 | 160 | 170 | 150 | 170 | 190 | 180 | 310 |
| reg\_alpha | [0,inf] | 1, 2, 3, 4, 5, 6, 7, 10, 15, 20, 25 | 4 | 4 | 5 | 4 | 6 | 3 | 4 | 20 |
| reg\_lambda | [1,inf] | 1, 2, 3, 4, 5 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| subsample | (0,1] | 0.2, 0.3, 0.4, 0.8, 1 | 0.4 | 0.3 | 0.4 | 0.8 | 0.8 | 0.8 | 0.8 | 0.9 |
| R Squared | | | 71.2 | 72 | 71 | 70 | 72 | 72 | 73.9 | 74 |
| Mean Absolute Error | | | 309.8 | 307.8 | 315.6 | 315.5 | 304.9 | 301.4 | 298.1 | 294.7 |

Table 13: XGBoost experimentation and gridsearch results

|  |  |
| --- | --- |
| **Hyperparameter** | **Values** |
| Activation layer 1 | Tanh |
| Activation layer 2 | Relu |
| Activation layer 3 | Tanh |
| Activation layer 4 | Tanh |
| Dropout | 0.2 |
| Optimizer | Adam |
| Learning Rate | 0.001 |
| Batch size | 32 |
| Epochs | 50 |
| **Results** | |
| Runtime (minutes) - Computer iMac - 24GB RAM | **2 minutes** |
| Mean Absolute Error | **0.066** |
| R-squared | **77.33 %** |

Table 14: LSTM V1 Hyperparameters and Results

|  |  |  |
| --- | --- | --- |
| **Hyperparameter** | **Values Searched** | **Result** |
| neurons | 32, 64, 128 | 64 |
| learning\_rate | 0.0001, 0.001 | 0.001 |
| dropout | 0.2, 0.3 | 0.3 |
| batch\_size | 32, 64 | 32 |
| epoch | 50, 100 | 50 |
| R-squared | | **74.06%** |
| Mean Absolute Error | | **0.0715** |

Table 15: LSTM gridsearch results

|  |  |  |
| --- | --- | --- |
| **Hyperparameter** | **Values Searched** | **Result** |
| Learning rate | between 1e-5 and 1e-2 | 0.001039 |
| Number of units at each layer | 16,32,64,128,512 and 1024 | Varied on specific layers. Refer to model definition table below |
| Activation function at each layer | tanh, relu and sigmoid | Varied on specific layers. Refer to model definition table below |
| Dropout rate after each layer | 0.1, 0.2, 0.3, 0.4 and 0.5 | Varied on specific layers. Refer to model definition table below |
| Mean Absolute Error | | **0.0601** |
| R-squared | | **80.98%** |

Table 16: Bayesian Optimisation results

|  |  |  |
| --- | --- | --- |
| Layer | Type | Hyperparameters with the best values included |
| Input Layer | LSTM | units: best\_n\_units1 = 1024  input\_shape: (1, 18)  return\_sequences: True  activation: best\_activation1 = relu |
| Dropout Layer | Dropout | dropout\_rate: dropout\_rate1 = 0.2 |
| First Hidden Layer | LSTM | units: best\_n\_units2 = 1024  return\_sequences: True  activation: best\_activation2 = relu |
| Dropout Layer | Dropout | dropout\_rate: dropout\_rate2 = 0.0 |
| Second Hidden Layer | LSTM | units: best\_n\_units3 = 256  return\_sequences: True  activation: best\_activation3 = tanh |
| Dropout Layer | Dropout | dropout\_rate: dropout\_rate3 = 0.1 |
| Third Hidden Layer | LSTM | units: best\_n\_units4 = 256  return\_sequences: False  activation: best\_activation4 = sigmoid |
| Dropout Layer | Dropout | dropout\_rate: dropout\_rate4 = 0.0 |
| Output Layer | Dense | Output dimension = 1 |

Table 17: LSTM final model details

|  |  |  |
| --- | --- | --- |
|  | R-Squared | Mean Absolute Error |
| 1 day look-back (No Total Demand) | 0.428 | 0.1081 |
| 1 day look-back (with Total Demand) | 0.8098 | 0.0601 |
| 3 day look-back (with Total Demand) | 0.5375 | 0.0972 |
| 5 day look-back (with Total Demand) | 0.4578 | 0.1106 |
| 7 day look-back (with Total Demand) | 0.5381 | 0.0979 |
| 15 day look-back (with Total Demand) | 0.5303 | 0.1003 |

Table 18: LSTM lookback experimentation results

|  |  |
| --- | --- |
| LSTM Model Average Actual Difference: | 73.9812 |
| Supplied Model Average Actual Difference: | -20.5042 |

Table 19: Summary of Average Actual Differences

Graphs

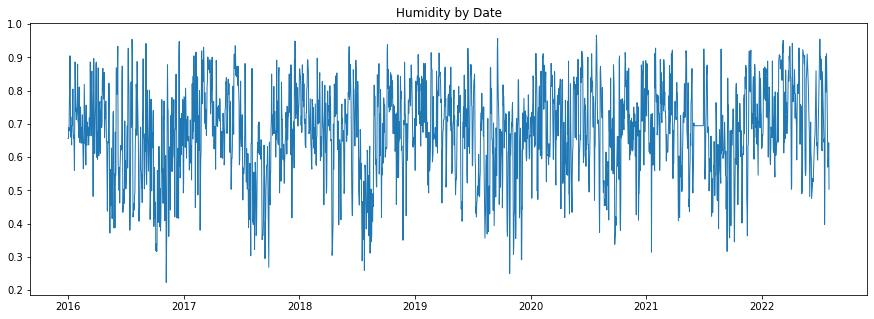


Figure 9: Humidity time series plot

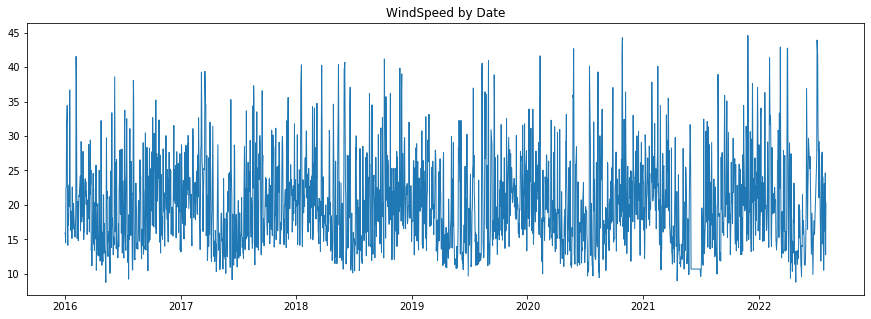


Figure 10: Wind Speed time series plot

|  |  |
| --- | --- |
| Before outlier handling | After outlier handling |
|  |  |
|  |  |
|  |  |
|  | N/A |
|  | N/A |
|  | Do not remove - removing outliers will remove an entire month of data |
|  |  |
|  |  |

Figure 11: Comparison of boxplots from before and after outlier handling for each feature

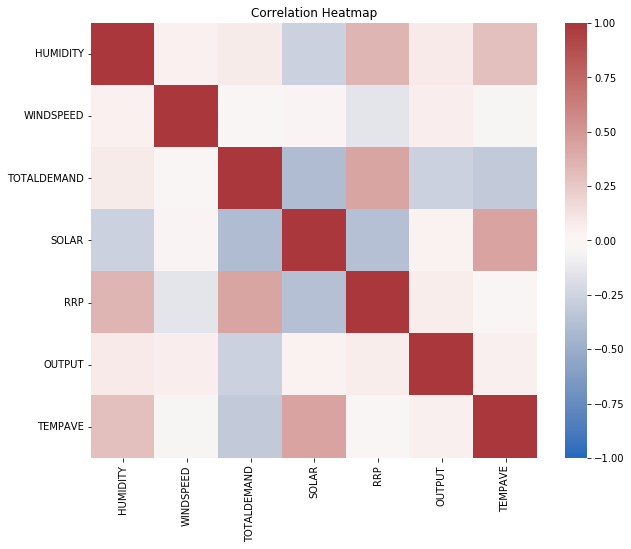


Figure 12: Feature Correlation Heatmap

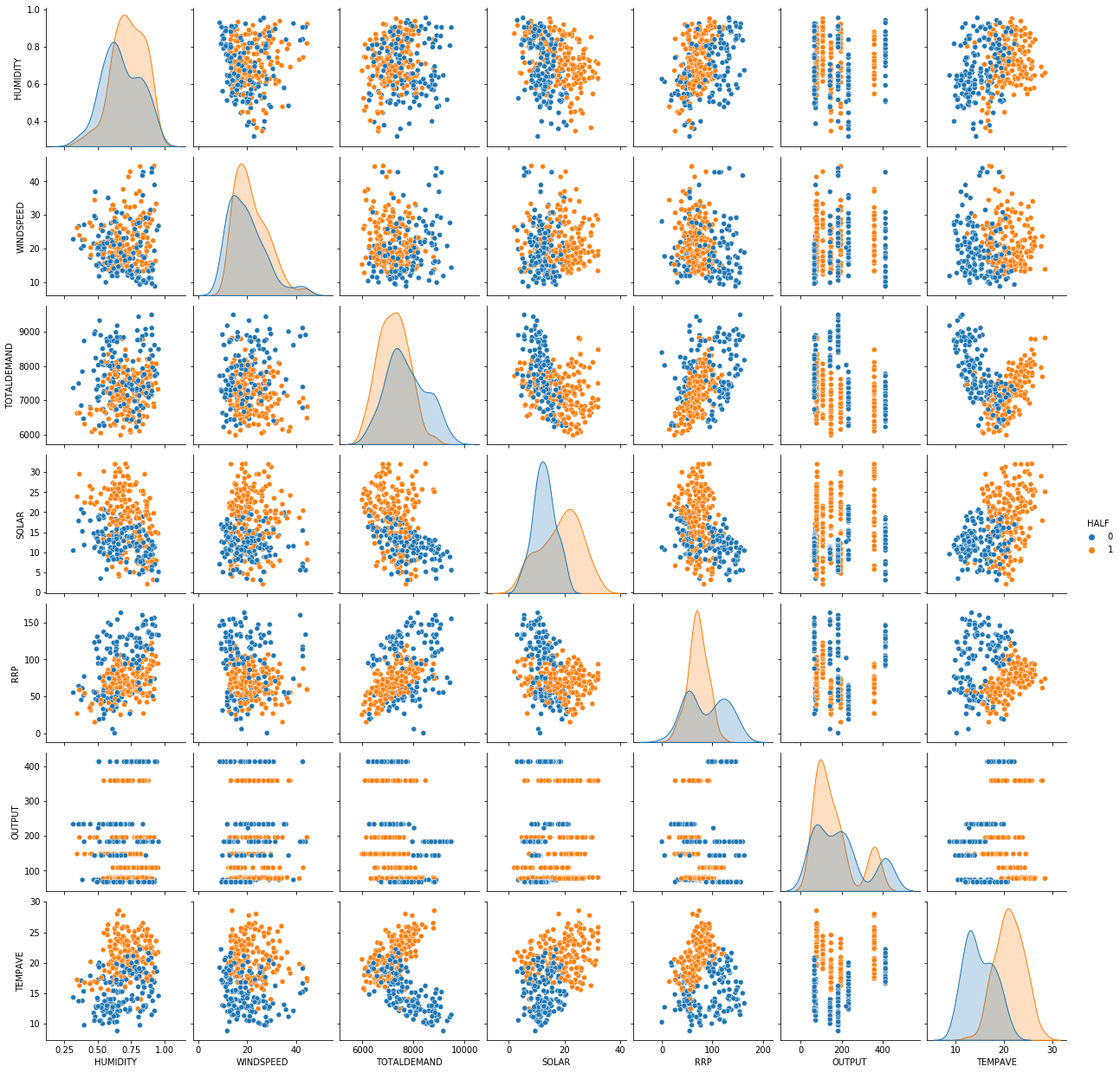


Figure 13: Feature Pairplot

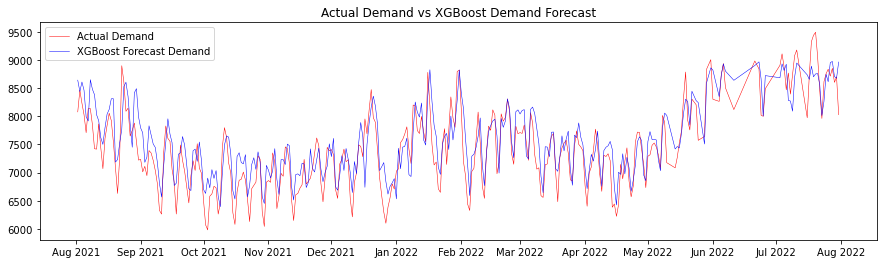


Figure 16: Actual Demand vs XGBoost Demand Forecast time series plot

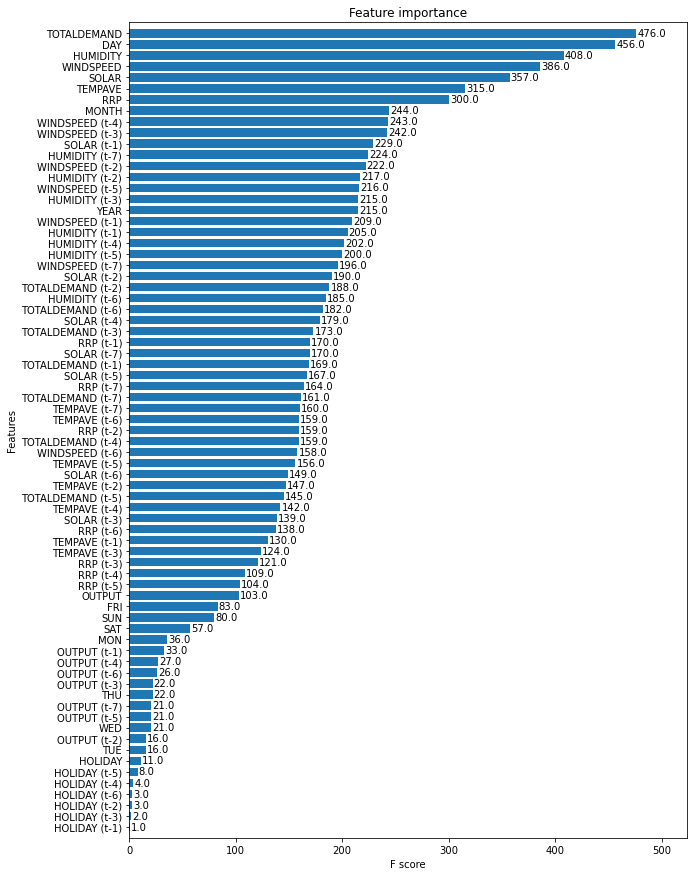


Figure 17: XGBoost feature importance bar plot

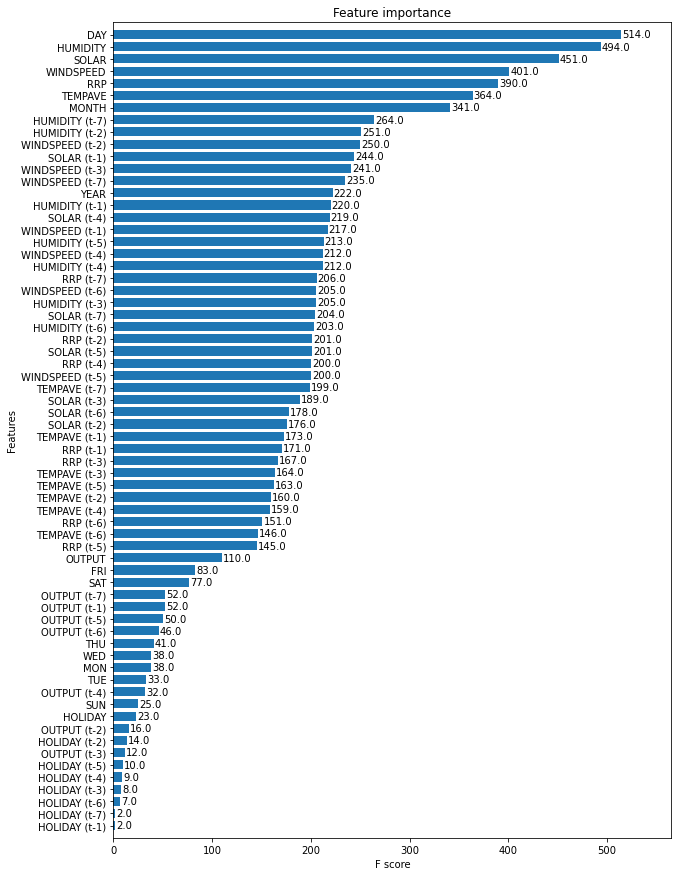


Figure 21: XGBoost feature importance bar plot without TOTALDEMAND

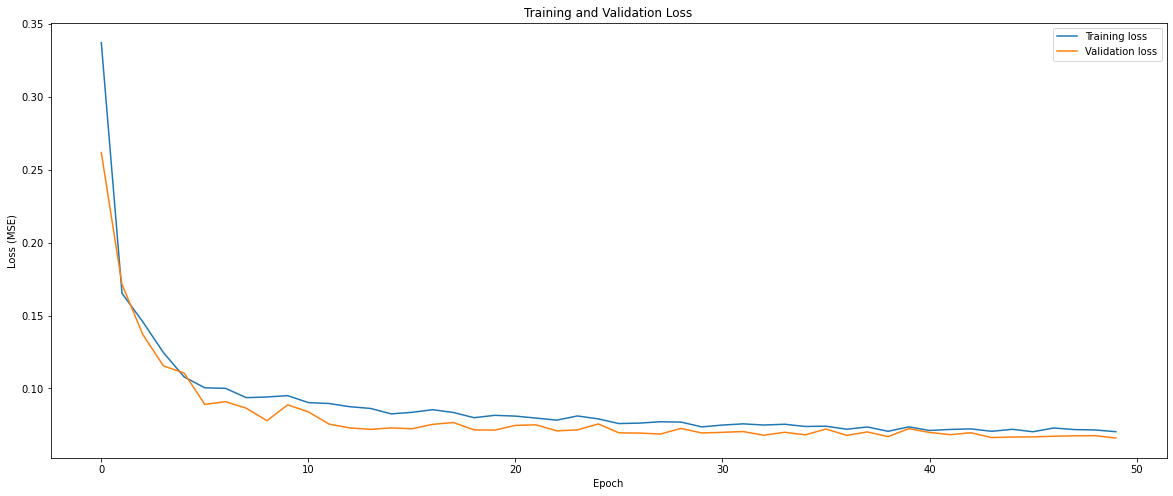


Figure 23: Training vs validation loss for each epoch

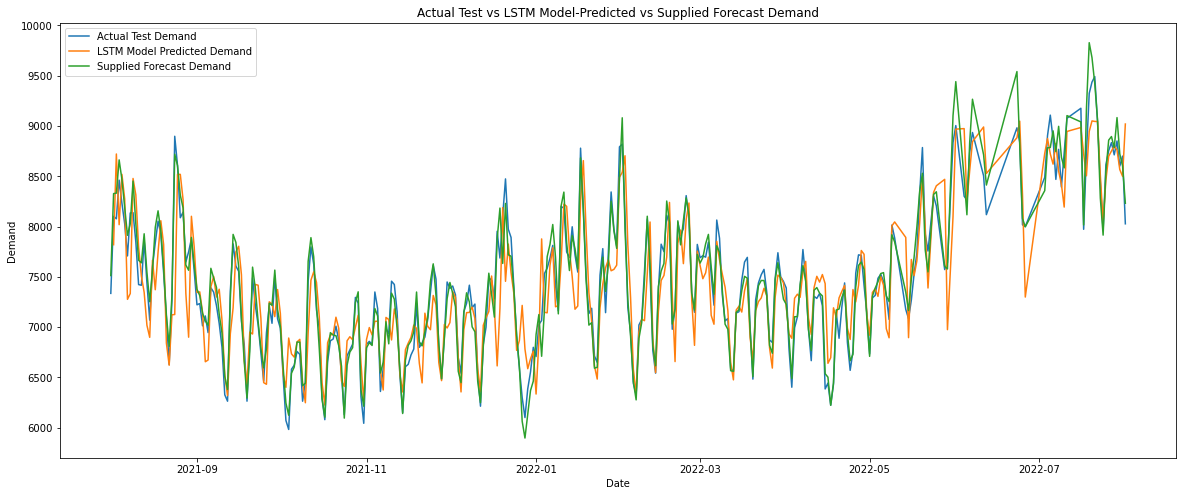


Figure 24: Actual Test Demand vs LSTM Model-Predicted Demand vs Supplied Forecast Demand time series plot

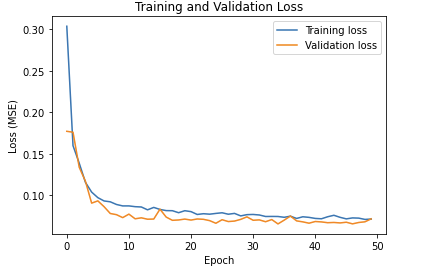


Figure 25: Training vs validation loss for each epoch:

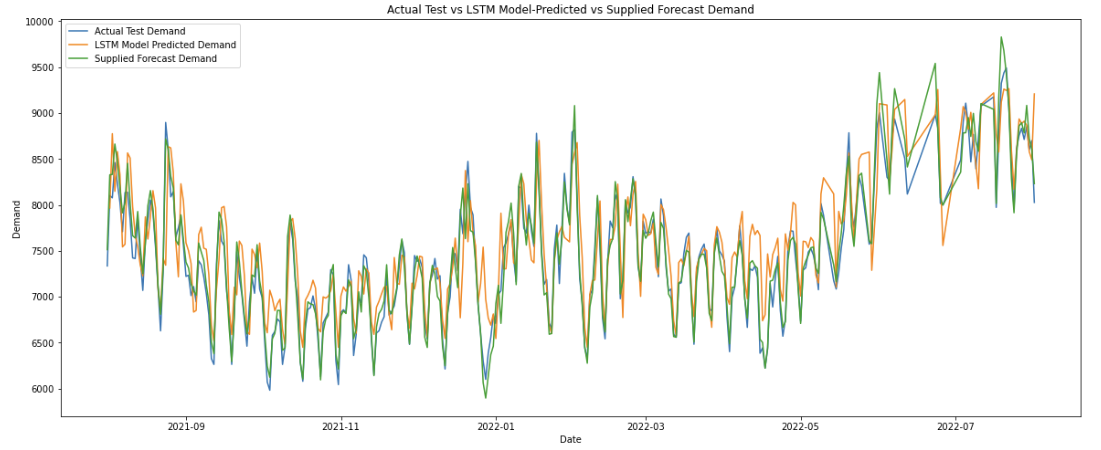
****

Figure 26: Actual Test Demand vs LSTM Model-Predicted Demand vs Supplied Forecast Demand for the 5 year study period

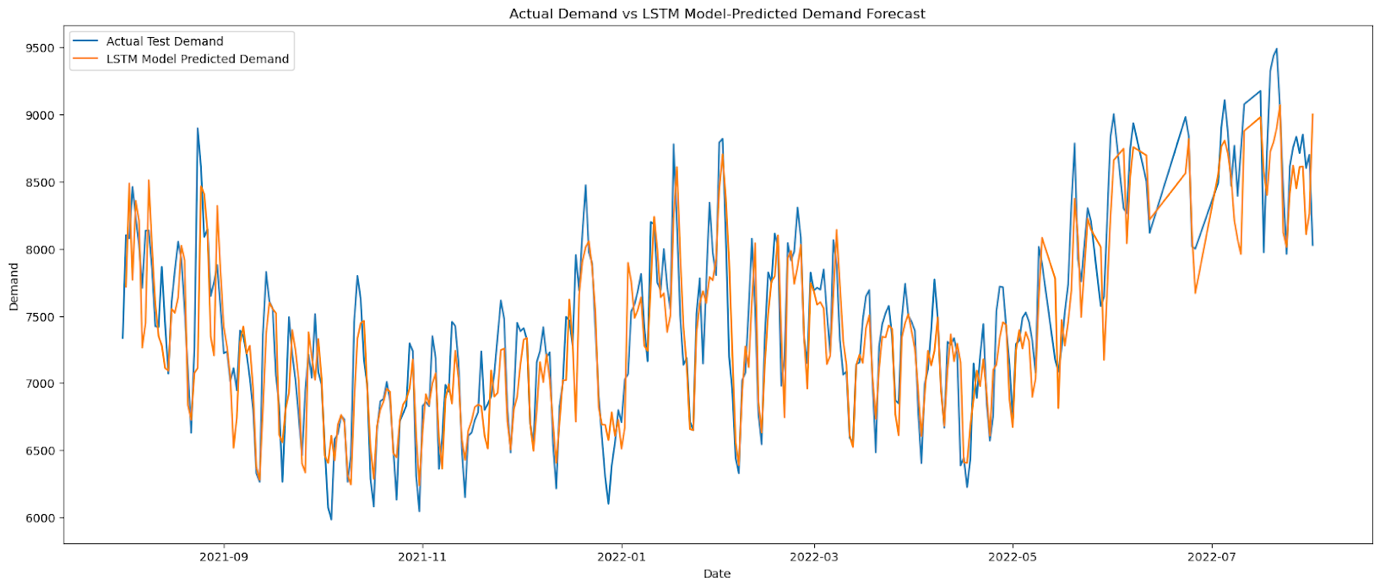


Figure 31: Actual Demand vs LSTM Demand Forecast time series plot