A Data Science Approach to Short-Term Energy Demand Forecast

Louis Long (z5162359) Mark Budrewicz (z5353932) Hao Chiou (z5131909)
Charlet Jeyapaul (z5375906)

07/10/2023

Contents

1 Abstract

2 Motivation

Electricity demand forecasting is a topic that is within the interest of government and market bodies, with the consensus of accurate forecasts of energy demand driving profitability. Electricity in Australia is generated primarily using oil, coal, and gas as fuel[@BhattacharyaMita2020Coep]. Also, with oil prices increasing, this implies that the cost to generate energy will also increase[@KpodarKangni2022Tdio]. Therefore, profit maximization is achieved when energy production is minimized to be near actual energy demand. Thus, if electricity demand can be modelled accurately, government and market bodies can then utilize such information to tailor policies as well as optimize pricing. A recent phenomenon in the electricity grid has been the shift towards renewable energy sources such solar photovoltaic and wind generation. While these technologies are indeed cleaner they bring with them risks[@arenawebsite] due to their intermittency and inability to be dispatchable as compared to electricity produced from fossil fuels. As of 2021, renewables accounted for 23% of total electricity generation in the state of NSW[@energywebsite]. As it is non dispatchable, renewable electricity generation acts to lower the energy demand of the grid, so much so, that the Australian Energy Market Operation (AEMO) has already acknowledged that the increasing uptake in rooftop solar will shift maximum load demand to later in the day at sunset as temperatures remain high but roof top solar output falls to nil[@AEMOdemandwebsite]. The increase in renewable electricity generation will see a parallel requirement for flexible electricity generation such as gas, pumped hydro and grid scale batteries with quick ramp up ability to cope with the variability in supply[@AGLwebsite]. And with that, a more robust shorter term electricity demand forecasting process will be required to ensure that the quick start alternative electricity generators have sufficient warning to ramp up when required.

3 Introduction

The aim of this paper is to investigate different modelling techniques to generate a rolling 60 min forward demand forecast for the state NSW electricity grid that could be used by flexible quick start generators (gas, pumped hydro, and grid scale batteries) to help minimise operational costs and give a competitive edge in pricing which will benefit its profitability and flow down to lower costs for the end users of the electricity grid.

Quick start electricity generators are essential to the electricity generation mix as they add flexibility to the grid to cope with highly variable demand [@AGLwebsite]. This flexibility ensures a sufficient balance of supply and demand as these quick start generators can supply with a shorter ramp up period. While the ramp up period is shorter than conventional base load generators, it still important for an electricity generator to be able to estimate how much electricity is required by the grid within a short forecast window so that it can strategically plan generation output to minimise its startup and shut down costs and maximise profit. Times of high load requirement in the grid inevitable lead to higher spot prices, so it is in the interests of an electricity supplier to be able to accurately forecast and capitalise on such situations, inline with its operational constraints (startup and shutdown times, maintenance requirements). While an electricity demand forecast is already generated by AEMO [@AEMOforecastwebsite] for participants in the grid, this analysis aims to investigate a forecasting model that will outperform AEMO's demand forecast, allowing a supplier to take advantage of potential misalignment in AEMO's forecast to actual load demand.

AEMO produces a short term forecast model in 30 minute intervals (the 'Half-hourly demand model') [@AEMOforecastwebsite], that uses features such as a categorical value for each half hour block of the day and month of the year, dummy variables for weekends and public holidays and transformations of half hourly temperature inclusive of interaction terms. This model uses a Least Absolute Shrinkage and Selection Operator (LASSO) regularisation algorithm which is a regression analysis method that automatically performs feature selection by including a penalty term to the loss function which acts to shrink the non significant variable coefficients to zero.

Our models seek to build upon the AEMO 'Half-hourly demand model', to test if other meteorological data such as solar exposure can capture the impact of household solar photovoltaic generation in the grid, and whether rainfall has an impact on indoor electricity consumption. Our investigation seeks to compare the performance various time series and machine learning models such as LASSO, Autoregressive integrated moving average (ARIMA), Support Vector Regression (SVR) and Random Forrest Regression with the aim of improving the short term electricity demand forecast with a slight increase in computational cost. Each of the models will then be further refined via training with simulated data. The final model will be selected after comparison of various generated models and features, specifically through consideration of feature selection, accuracy, and interpretability and performance relative to AEMO's forecast.

4 Literature review

Initial review of the literature on electricity demand forecasting pointed to a well understood relationship between temperature and electricity demand that has already been incorporated by the AEMO's forecasting procedures [@AEMOforecastwebsite]. This relationship has been widely explored with a specific focus on higher frequency time intervals. [@McCullochJames2020IEDa] showed a strong relationship when modelling Australian 5-minute electricity demand with intraday temperature using a Generalised Additive Model. This same research also indicated that sensitivity of demand to temperature was heavily dependent on the time of day with higher demand sensitivity during hours of high human activity. In [@ZhangGuoqiang2020ANMf], their support vector machine (SVM) model found other meteorological features such as wind speed, relative humidity, precipitation and air pressure impacted electricity demand. Interestingly the behaviour of these variables on demand was non linear, which the authors hypothesised could be due to their impact on relative temperature (ie high wind can act to help cool on warmer days, but make it feel colder on cooler days). [@VuD.H.2015Avif] regression model for monthly electricity demand for the State of NSW showed that demand was "predominantly dependent on temperature, humidity and the number of rainy days". While the study was done on monthly electricity demand data, it points to a link in meteorological features impacting NSW electricity demand which may also transfer to more granular time periods.

Other literature included additional features and characteristics of the data such as seasonality and cyclicality (from hourly to monthly) in the electricity demand time series. As described in [@AEMOforecastwebsite], AEMO's half hourly model includes categorical values for half hourly block of the day and month of the year, and include dummy variables for non work days. In [@ClementsA.E.2016Fdel] study on Queensland electricity demand, seasonality of demand including hour of the day, day of the week were identified as important features that needed to be factored into the modelling. One approach by [@JiangPing2020Ance] was to remove the seasonality as a prepossessing step through a Fast Fourier Transform method which converted the electricity demand from time series domain to frequency domain. In this model, the seasonality was removed so as not to affect the SVM algorithm. Other approaches such as [@AlonsoAndr?sM.2020ASSL], chose to let their Long Short-Term Memory (LSTM) recurrent neural network to capture the seasonal idiosyncrasy naturally. The daily seasonality of hourly blocks was identified in the data exploration, so an LSTM with 24 hour time steps was used to capture the seasonality. As our paper sets out to investigate multiple forecasting techniques, the seasonality will need to be incorporated through different methods.

As the complexity of electricity demand has evolved so has the research into techniques for its forecasting. Earlier models explored Box and Jenkins time series modelling [@HaganMartinT.1987TTSA] or what is commonly known today as ARIMA. They found that a seasonal ARIMA model was well suited to electricity load forecasting where the forecast model was a linear combination of previous days electricity load with 24 hour (daily) and 168 hour (weekly) autoregressive lag included in their model. It was noted that including a transfer function to model the non linear relationship between temperature and electricity load should improve the forecasting error. A similar autoregressive approach in [VuD.H.2017Sedf] using NSW daily electricity demand data looked at grouping days of the week with similar electricity demand profiles together (weekends) in the modelling to improve the robustness of the forecast and to vary its coefficients with time to better capture the relationship to more recent data points.

As previously discussed, AEMO's half hourly model is a linear regression technique that uses the LASSO

regularisation algorithm to produce a forecast which aims to 'describe the relationship between underlying demand and key explanatory' [@AEMOforecastwebsite]. In this respect it is more of an explanatory model in contrast to the predictive ARIMA methods above that have no explanatory power. Multiple regression techniques have provided a simple, yet powerful approach to forecast short term electricity demand. In [@CharltonNathaniel2014Arpm] a study was performed on 20 different locations, with 24 linear regression models generated to forecast electricity demand for each hour of the day at each location, which included variables such as temperature (and polynomial transformation), dummy variables for season and weekday/weekend with good performance of 0.70 R2. In the Australian context, [@ShuFan2012SLFB] regression study on Victorian and South Australian electricity demand, produced a separate model (48) for each half-hourly period of the day, and included variables such as temperature (current and lagged, 24hr min and max), previous demand observations (24hr min, max and average) and calendar variables to forecast current electricity demand. A stepwise variable selection was carried out from a wider set of variables with a process of elimination and analysis of the effect on mean absolute percentage error (MAPE) used as the selection criterion. In [@2019Uiem], a linear regression forecasting model was used to predict 30min electricity pricing utilising the LASSO algorithm for its variable selection. This allowed an almost 'unlimited' number of variables to be used in the forecasting. In keeping with the approach by AEMO, we look to also implement a LASSO regularisation algorithm but our approach will investigate additional transformations and meteorological variables.

While there was originally more focus on regression modelling of short term electricity demand, the research has shifted to techniques that can model the complex nonlinear relationship of electricity demand to temperature and seasonality. One of the most recent papers in the literature, [@9282124], forecasts 24 hour ahead electricity load for the city of Toronto by combining a LSTM network using historical electrical demand with a deep Feedforward Neural Network (FNN) using forecasted meteorological inputs (temperature). The combination of the two artificial neural networks outperformed each individual network as measured by Root Mean Square Error (RMSE). Another approach to model the non-linear relationships is forecasting with a Support Vector Regression (SVR) which uses a kernel to convert non-linear data to a high-dimensional space and then separates the data with a hyperplane. [@JiangPing2020Moed] utilised SVM algorithms with 30min time series electricity demand data to generate a forecast for both NSW and Singapore electricity demand, performing better than their benchmarking models. In [@LiRanran2021Mlss], a least squares support vector machine was used to model half hourly electricity demand for New South Wales, Queensland and South Australia which as a method is less computationally expensive than the standard SVM technique. For this model, the data was split between work days and weekends and a lagged time series was used with a radial basis function (RBF) kernel to model each half hour forecast. The authors did not include temperature, which we seek to utilise for our SVM analysis.

Finally, Random Forest regression models have also been used as a machine learning technique for short term electricity demand forecasting. They are a type of ensemble model that utilises classification and regression trees to apply bagging to generate a series or decision/regression trees that each vote on a result. One of the benefits of random forest models is they provide a feature importance metric that can be utilised for feature selection in high dimensions of data. In [@HuangNantian2016APIF], particular attention was paid to removing outliers as they could have a 'significant effect on the importance of the features'. The paper's analysis found that for 1 hour forward electricity demand forecast, the random forest model's most important features were the previous 24 hours of actual demand, whether it was a workday, and the day of the week. A more recent paper [@DudekGrzegorz2022ACSo] applied the random forest method to forecasting short term electricity demand but adding additional features for daily and weekly seasonality. Again, only calendar features were used in the modelling for both papers, with no temperature data utilised. While there has been the use of random forest techniques for short term electricity demand forecasting in the Australian context [@FanGuo-Feng2021Fsel], they have been used within combined hybrid machine learning models, so we believe our research into applying soley random forest regression model for short term forecasting is novel, which will also seek to use meteorological data.

Our paper seeks to build upon the success in the literature above by applying similar machine learning methods (with additional meteorological data) to generate a comparison of the techniques to identify the best performing model specific to the NSW grid. In [@WangZhe2021Pcde], it is apparent that the best model approach and subset of features are heavily dependent on characteristics of that grid, so while these

machine learning methods above have been heavily researched, we seek to understand which one of them can outperform the AEMO demand forecast.

5 Material and Methods

5.1 Software

Our analysis and modelling were conducted using the R programming language, known for its proficiency in statistical analysis and data manipulation. We leveraged several essential R packages to streamline data management and analysis, empowering us to build robust predictive models. Some of the key R packages we used include:

- **tidyverse:** This comprehensive package suite equipped us with a powerful set of tools for data manipulation, visualization, and analysis. Its flexibility and integration with other packages were essential for our data preparation and exploration.
- **lubridate:** Essential for working with date and time data, a crucial aspect of time series analysis. This package allowed us to handle temporal information efficiently and accurately.
- **data.table:** We harnessed the efficiency of the data.table package for data manipulation tasks, particularly when dealing with large datasets. Its speed and concise syntax were instrumental in aggregating and selecting columns.
- **crayon:** Used for adding colourful text to our visualizations, enhancing their interpretability, and making our reports more engaging.
- **zoo:** As a pivotal package for handling time series data, zoo played a crucial role in our analysis. It provided the tools needed to process, manipulate, and visualize time-based data structures.
- **tsibble:** Facilitating the management of time series data, tsibble served as the foundation for our time-based operations, ensuring consistency and accuracy in our analyses.
- **gridExtra, grid, patchwork:** These packages enabled us to create complex grid layouts for our visualizations, enhancing the presentation of our findings and insights.
- **GGally:** This package expanded our ggplot2-based visualizations by offering additional plot types and features, enriching our data exploration.

5.2 Data Acquisition

The data sets used in the analysis contained historical data from the state of NSW inclusive of the period 01-01-2010 to 31-01-2021 comprising historical electricity demand time series and meteorological observations over the period. Each data set has been stored in our public Github [@githubrepo]. The data includes the following extracts;

. Total Electricity Demand (totaldemand_nsw.csv 42.9mb) Actual total electricity demand for the state of NSW in 5 minute increments measured in megawatts (MW). This data was sourced from the Energy Market Management System database of AEMO [@EMMS] made available by UNSW for the purposes of this research project.

. Total Forecasted Electricity Demand (forecastdemand_nsw.csv.zip.partaa 94.4mb, forecastdemand_nsw.csv.zip.partab 21mb) Also known as the 'Half Hourly forecast', it is the forecasted electricity demand measured in MW for the state of NSW for a series of future 'half hourly' time series increments generated by AEMO per the 'Forecasting Approach Electricity Demand Forecasting Methodology' paper [@AEMOforecastwebsite]. This AEMO forecast data will be used as a benchmark to compare against the other forecasting models that will be investigated. This data was sourced from the Energy Market Management System database of AEMO [@EMMS] made available by UNSW for the purposes of this research project. The 1 hour forward accuracy of the forecast vs the actuals as measured by error rate was calculated as ###%.

. Air Temperature (temperature_nsw.csv 8mb) Also known as 'dry bulb air temperature' [@bomtemp-website], air temperature in Celsius was taken from periodic (approximately 30min) observations from Bankstown Airport weather station and is used as a proxy to represent statewide temperature in NSW. It has been included in our analysis to include the well-known impact that temperature has to electricity demand [@McCullochJames2020IEDa]. This data was sourced from the Australian Data Archive for Meteorology [@bomarchive], made available by UNSW for the purposes of this research project.

. Solar Exposure (solar_nsw.csv 416.4kb) Solar exposure is the total amount of solar energy falling on a horizontal surface, measured in megajoules per square metre (MJ/m*m) [@bomarchive], with 'values highest in clear sky conditions during the summer, and lowest during winter or very cloudy days'. Solar exposure was included in this analysis to analyse the impact of solar photo voltaic electricity generation which acts as a net reduction to total electricity demand. The data used is based on daily solar observations derived from satellite data for the coordinates of Bankstown Airport weather station which will be a proxy for statewide solar exposure for NSW. This data was sourced from the Australian Data Archive for Meteorology [@bomarchive].

. Rainfall (rainfall_nsw.csv 744.5kb) Rainfall in millimetres (mm) was taken from observations from Bankstown Airport weather station, with the data the representing daily amount of rainfall [@bomarchive]. It will be used as a proxy for rainfall for the state of NSW. It has been included in the analysis to study the potential indoor/outdoor impact that rainfall may have on electricity demand. This data was sourced from the Australian Data Archive for Meteorology [@bomarchive].

5.3 Pre-processing Steps

R package

In this project, data will be pre-processed via R. Data transformation will be completed via the tidyverse package [@WickhamHadley2019WttT]. Whereas date time related transformations will be completed via utilising the Lubridate package [@GarrettGrolemund2011DaTM].

Data cleaning and preparation

All data files used in this project will be in csv format. Data files will be read into R and processed to become fit for purpose via having the correct format and sufficient data integrity. Data will be processed to ensure consistency can be met, as well as having appropriate treatment for null values.

In terms of consistency, it would be to ensure each dataset has a consistent time gap between each measure when ordered by timestamp. This is important as the dataset will eventually be aggregated from by per minute to hourly, and inconsistent timestamp would result in hours having an unequal number of measures, where the number of measures per hour need to be equal in order for each hour to be unbiased. To achieve such consistency, each dataset used will be joined via left join to date-matrix of relevant timestamp intervals. This will effectively retain measures if they exist in the original data file, as well as surfacing timestamps where measures are null. Date-matrix data frames with intervals one, five, thirty minutes, as well as one day will be created.

Prior transformation takes place, all data files will be read into R, converted into dataframes. Each dataframe will be deduplicated to ensure all rows are unique. From counting the number of unique rows, the total demand dataset contains 39 duplicated rows therefore will be deduplicated within R. The R data frames generated will be filtered via relevant region columns to ensure data used is for NSW region only. Additionally, the intervals between each measure will be calculated via finding the datetime difference between current and previous measures, this column is used to determine the most suitable date matrix to join the dataset to.

Data transformation In this project, the five primary data sources utilised will be total energy demand, historical forecast, historical temperature, daily rainfall, and daily solar exposure. The timeframe of interest will be from 2010-01-01 to 2022-08-01. Data transformation will be explained separately by each dataset individually as per below:

Total Energy Demand The key columns required from the total energy demand dataset will be total energy demand and datetime. There are no extreme outliers in terms of total energy demand, therefore no filtering from the data is required. Given measures are mostly taken with five minute intervals, a five minute interval date-matrix will serve as the base table for left join.

Forecast Demand Forecasts generated by AEMO are nearly always on a half-hourly basis. Within each unique timestamp, AEMO will make multiple forecasts with the new forecast taking account of the previous, where forecasts are sequences via PERIODID (indexed from 1 ~ n, 1 being the first forecast, and n-th forecast being the final). Within the forecast demand dataset, all forecast values are within a reasonable range of having no outlier, therefore no filtering is required. In this project only the first and final forecast will be used, via transformation of the tables from long to wide format, removing the PERIODID and forecast measure columns, whilst adding two new columns for initial and final forecast. The final table post transformation will have three columns, datetime, initial forecast, and final forecast. The transformed table will then be left joined onto a date-matrix of thirty minutes.

Temperature The temperature dataset contains temperature measures with sporadic timestamps, usually between zero to thirty minutes, as well as larger gaps of up to three days. The temperature dataset contains extreme outliers such as -9999 degrees, whilst historically the lowest temperature ever recorded in NSW is -23 degrees celsius [@bomRecordRainfallTemp]. Therefore, the temperature data will filter out rows where temperatures are below -23 degrees celsius. The columns required from the temperature dataset will be datetime and temperature. The filtered temperature table with required column only will then be joined via left join to date-matrix with one minute intervals, due to sporadic timestamps mentioned above.

Daily Rainfall and Daily Solar Exposure For daily rainfall and solar datasets are recorded in daily format, with mostly 1 day intervals between each measure. The recorded measurements do not contain any extreme outliers therefore do not need to be filtered.

One thing to note with solar and rainfall datasets however, would be that the fields year, month, and day are in separate columns, hence to ensure the datasets can be joined to the date-matrix, the three fields will be combined into a single date field with yyyy-mm-dd format. Once the date field is created, the tables will then be joined via left join to date-matrix of one day interval.

5.4 Data Cleaning

To clean the data to ensure there are no null measurements for any timestamp, for any column that contains measurement, methods such as linear interpolation and last observation carried forward will be utilised [@ZhangYifan2022Hmdi]. Initially, simple imputation was considered due to simplicity, however because such a method is also prone to bias therefore will not be applied in this case [@EmmanuelTlamelo2021Asom]. Another idea was populating missing temperature via regression based interpolation methods [@XuCheng-Dong2013IoMT]. However, given the temperature data covers only one region, as well as missing timegaps being relatively small in almost all cases, linear interpolation will be sufficient due to simplicity. To summarise, missing total demand, temperature, daily rainfall, and daily solar exposure will be filled via linear interpolation, and missing AEMO forecast will be populated via last observation carried forward to prevent skewing from original forecast.

5.5 Consolidation and Aggregation of final dataset

Dataset Joins

All data frames will then be joined into a single table in preparation for the modelling step. The cleaned dataset of total energy demand dataset will be used as base for join for other tables, given total energy demand will be considered the dependent variable for the model. Temperature and forecast dataset will be joined via datetime field. To join solar and rainfall dataset, a date field that does not include hour and minute will be derived from datetime column with yyyy-mm-dd format, solar exposure and rainfall datasets will then be joined to this field via relevant days.

Dataset Aggregation

Given the objective of this project is to create a model capable of forecasting hourly demand, the combined data will be aggregated to hourly level. Given such, the total and forecast energy demand will be aggregated by summing the total/forecasted energy per hour. Temperature on the other hand will be averaged. Rainfall and solar will be untouched given the measures were initially at daily level.

Additional fields derived from existing data:

- Year, Month, Day, Hour = Extracted from datetime field
- Seasons = Derived from months, Spring = {9,10,11}, Summer = {12,1,2}, Autumn = {3,4,5}, and Winter = {6,7,8}
- During Day = If the hour is between 8 am ~ 8 pm then 1, else 0.(this timeframe is selected due to having positive correlation to total energy demand, refer to Figure 9)
- BusinessDay = If day between 1~5 then 1, else 0.
- pub_holiday_flag = 1 if it is a public holiday, else 0. NSW Public Holiday table is retrieved via utilising the tsibble package in R [@WangEaro2020ANTD]

Extreme temperatures of day and night fields will be derived via guidelines provided by the Australian government [@bomExtremeTemp]. The fields will be in {1,0}.

- Extreme Hot Day = Over 40 degrees and during day equals to 1
- Hot Day = Over 35 degrees during day and during day equals to 1
- Extreme Hot Night = Over 25 degrees and during day equals to 1
- Hot Night= Over 20 degrees and during day equals to 1
- Extreme Cold day = Under 10 degrees and during day equals to 0
- Cold day = Under 15 degrees during day and during day equals to 0
- Extreme Cold Night= Under 0 degrees and during day equals to 0
- Cold Night= Under 5 degrees and during day equals to 0

5.6 Assumption

Based on the data used in this project, as well as various studies completed in the past, three main assumptions could be made with regards to behaviour of total energy demand upon interaction with various factors.

The first assumption is energy demand will increase during colder or hotter temperatures. Based on research completed in the past, it has been confirmed that winter and summer seasons have higher demand for energy during heating and cooling needs [@SavicStevan2014Cawa]. Additionally, according to research completed by Zhang et al. suggests energy demand tends to increase when temperature increases drastically [@ZhangShaohui2022Etar].

The second assumption would be in order for energy to be consumed, it would need to occur during the time of household resident activity. Past studies suggest that energy demand does indeed to be higher between 6 am to 9 pm, in contrast to 10 pm ~ 5 am [@SavicStevan2014Cawa].

For the third assumption, given energy demand seems to increase during hotter and colder temperatures, this raises the possibility that if the range of global temperature were to increase over time, it would lead to the possibility of energy demand increasing over the years. According to a study completed by Jakob and Walland to investigate Australian extreme temperatures, it has been established that the extremely hot temperatures are becoming more frequent [@JakobDorte2016Valc], which raises the question of potential increase in energy demand over the years.

5.7 Modelling Methods

We implemented three different predictive models to forecast electricity demand:

Lasso Regression:

- To address overfitting and improve model interpretability, we applied Lasso regression with cross-validation to select the best regularization parameter (lambda).
- The selected model was trained on the training data and evaluated using Mean Absolute Error (MAE) and Mean Squared Error (MSE) on the test data.

Support Vector Regression (SVR):

- We explored SVR models with various hyperparameters (epsilon and kernel) to capture non-linear relationships in the data.
- Multiple SVR models were trained and evaluated on the test data, with performance metrics reported.

Random Forest:

- Hyperparameter tuning was performed for Random Forest models, including parameters like the number of trees, mtrices, and sample rate.
- We trained two Random Forest models, one with all features and one with a reduced feature set based on variance importance.
- Model performance was assessed using test data, and predictions were compared with actual demand values.

5.8 Model Selection and Evaluation

Our evaluation process was meticulous, aiming to gauge the predictive accuracy of our models. We employed well-established error metrics, including Mean Absolute Error (MAE) and Mean Squared Error (MSE), to assess the performance of our predictive models rigorously. These metrics served as valuable guides in the selection of the most suitable model.

We benchmarked our models against the Australian Energy Market Operator (AEMO) forecast to evaluate their performance in a real-world context.

5.9 Data Exploration

Key EDA Visualizations:

In addition to model evaluation, we conducted a comprehensive Exploratory Data Analysis (EDA) that unveiled essential insights into our dataset. Some of the pivotal plots used in our EDA included:

- **Boxplot of Hourly Total Demand by Month:** This visualization shed light on the monthly variations in electricity demand, offering valuable insights.
- **Line Chart of Forecast vs. Actual (2010 & 2021):** A comparison of forecasted demand against actual demand for the years 2010 and 2021, aiding in the assessment of predictive accuracy.
- **Line Charts of Total Demand vs. Temperature:** These visualizations illustrated the relationship between electricity demand and temperature fluctuations, uncovering crucial patterns.
- **Total Demand vs. Temperature Pair plot:** An analytical tool that revealed intricate correlations between total demand and temperature.
- **Multivariate pair plot:** A multidimensional exploration of data relationships, helping us identify complex dependencies.

- **Lagged Correlations:** Insights into how past values of variables correlated with future electricity demand, providing a deeper understanding of temporal dependencies.

These visualizations played a pivotal role in our analysis by helping us uncover insights, identify outliers, and validate the effectiveness of our modelling approach. In the subsequent 'EDA Section,' we will delve deeper into the insights gained from our exploratory data analysis, which involved the creation of various plots and visualizations to uncover patterns and relationships within the data.

6 Exploratory Data Analysis

Historical Trend of All Variables In figure 1, line charts are created at the most granular level of year-month-date-hour. However, at this level of detail, it is difficult to identify trends clearly due to the amount of noise presented, suggesting further aggregation is required in order to identify the trends clearly. Also to note that the final period recorded of 2022-08 is omitted from the charts, as the only datapoint from this particular month is incomplete at hour level, and would cause significant dropoff of trend.

Looking at the general trend over time for all variables from figure 1, the only variable that displays seasonality without significant variability is temperature per hour. Total energy demand per hour and daily solar exposure displayed seasonality albeit having a large amount of variability. Daily rainfall on the other hand did not display any seasonality, or any resonance towards changes of trend occurred in total energy demand.

Figure 2 displays a line chart of data aggregated to a month level. Aggregating data to monthly level displays a clear smoothing impact to the line charts.

Looking at Trend via individual years

When the data is viewed at a monthly level via line chart in Figure 2, clear seasonality can be identified through having smoother lines in total energy demand, temperature, and solar exposure. Total rainfall viewed at monthly level still does not exhibit any seasonality. From an assumption perspective, traces of energy demand being affected by temperature are starting to appear. Also with regards to assumption three, the trend chart of monthly total energy demand over time is starting to display a slow decline in total energy demand over the years.

When the monthly trend is viewed at individual year level as per Figure 3, it can be identified that the trend did not change significantly between 2010 and 2021, for energy demand, average temperature, and average solar exposure. The reason why 2022 is not used despite being the final year recorded is because the data used in this project does not contain all months of 2022. It can also be noted that total energy demand peaks when average monthly temperature is at its highest or lowest point. Additionally, solar exposure when magnified to monthly level surfaces its relationship with temperature clearly. Furthermore, rainfall's trend remained stochastic, exhibiting no relationship with other variables.

Total Energy Demand via Boxplot at Yearly, Monthly, and Hourly level

Through using boxplot, it is possible to see the distribution of total energy demand across time. Figure 4 displays total energy demand is plotted against various time variables, specifically year, month, and hour. Each of these views will be discussed individually as per below.

Year

At yearly view we can see the fluctuation of max and min total energy demand across time. The most key point taken from this view would be the slight decrease of average total energy consumption per hour across the years. This is an interesting point because it contradicts assumption 3, where an increase of total energy demand is expected due to an increase of extreme temperature occurrences. Thus, we can rule the assumption to be false, where yearly average hourly total energy demand is indeed decreasing.

Month

At a monthly level, it can be identified that months of June, July, August, January, and February have higher average hourly total energy demand compared to the other months. This further verifies assumption 1 to be true, where total energy demand is expected to increase during hotter and colder months, as the months mentioned usually have the highest or lowest temperatures as per displayed in figure 2 and 3. Out of the previously highlighted months, June and July have the highest average total energy demand. This potentially shows that most people are more tolerant to hotter rather than colder temperatures, as June and July happen to have the coldest temperatures as per Figure 2 and 3. Furthermore, the boxplot of total energy by month suggests that months would be the best variable to use in terms of predictive modelling rather than seasons, as not necessarily all months within the season have higher total energy demand. For example, August is part of winter but has lower average total energy demand when compared to June and July.

Hour

Looking at hourly boxplot of total energy demand, the key point identified would be total energy demand is highest between 8 am and 7 pm, and is lowest between 2 am to 6 am. This verified assumption 2, of total energy demand to be higher during hours of human activity. Hourly boxplot verifies that hour can be used as a variable in terms of modelling, as there is clearly a pattern exhibited by the chart particularly around the mean of total energy demand per hour.

Upon inspecting the densities of total energy demand by seasons and months via Figure 5, it can be further identified that the only group that has a visible different mean compared to the other seasons is winter. However upon looking at this at a monthly level, it can be identified that August does indeed have much wider distribution compared to June and July, as shown in figure 4 also. This further indicates that months should be used instead of seasons in predictive models in order to reduce noise.

Total energy demand by temperature, rainfall, and solar exposure

Scatter Plot of figure 6 has been created to display the relationship between total energy demand and temperature, rainfall, and solar exposure. At top level, all scatter plots created in figure 6 are exhibiting great spread, suggesting correlation to be more useful in terms of strength of relationship. Nevertheless, it is still possible to get some characteristics from these relationships, particularly for temperature. From the total energy demand against temperature scatterplot, it is possible to identify that total energy demand increases when temperature shifts either left or right from 20 degrees. Rainfall and Solar scatter plots on the other hand exhibited little trend, where rainfall scatterplot is very much just stochastic when compared to total energy demand.

Correlation Matrix of Total Energy Demand

Using the correlation matrix, it is possible to identify how strong are the relationships between each variable [@HaddAlexandria2021Ucm]. When the data is inspected at correlation perspective via figure 7, from total energy demand perspective, the highest correlated variables are nearly always associated with cold temperature. For example, it is understandable where temperature does not hold a strong correlation with total energy demand, given in figure 6, where the scatter plot does not exhibit a linear trend; however, from the correlation matrix of figure 7, fields such as cold day, extreme cold day, and winter flag hold some of the highest correlation with total energy demand, indicating triggering those flags would result in higher total energy demand. Another field that holds a high correlation of 0.5 with total energy demand is during_day (flagging hours between 8 am ~ 8 pm), and this again strengthens assumption 2 of energy consumption being higher during human active hours. Furthermore, the negative correlation of -0.2 of datehour with total energy demand again contradicts assumption 3, hence we can settle with the statement that total energy demand is indeed gradually reducing over time.

In the correlation matrix of total energy demand against month flags, via figure 8, the highest correlation can be identified in July, as well as August, indicating higher energy demand during these two months. This is quite surprising as August had much less increase in terms of total energy demand compared to June and July as per Figure 4. This indicates that perhaps both July and August should be kept for creating the predictive model.

Upon inspecting the correlation matrix of total energy demand against hour flags in Figure 9, it can be identified that hour 3 to 5 have the highest negative correlation, whilst hour 17 ~ 9 have the highest positive correlation. The correlation matrix of the hour flag against total energy demand again demonstrates that there is a pattern between total energy demand and hour of day.

Overall, the correlation matrix from figure 7,8 and 9 shows us that in terms of predictive model, the fields that definitely should be used are extreme temperature flags, hour flags, and month flags.

Lagged Correlation of Total Energy Demand

Finally, lagged correlation of total energy demand will be investigated. Lagged correlation looks at the correlation between time shifted time series data [@ZagourasAthanassios2015Otro]. Essentially what this section is after is finding the best lag that can be used to predict future total energy demand. In this case, lags of 0 to 24 will be tested. The result as per figure 10, is that the more correlated lag would be lag of 1

and 24, which 24 will be ignored as it is essentially the same time given 24 hours. Therefore, total energy demand has the highest lagged correlation when shifted with a lag of 1.

7 Analysis and Results

7.1 Modelling setup

A few pre-processing steps were required before performing the demand forecast modelling, which included defining additional features, removing data where lagged variables do not exist, and splitting the dataset into training, test and holdout sets.

As our models will prioritise predictive performance over interpretability, it is important to include any potential features which may help in more accurately predicting energy demand. However, potential issues such as model bias or overfitting may arise, so an appropriate experimental setup was implemented to ensure the final model selected would minimise these occurrences. Some additional features include:

- Dummy variables for **hour of day** (denoted `HOUR_1`, `HOUR_2`, ..., `HOUR_23`, with midnight being the baseline hour)
- Dummy variables for **day of week** (denoted `weekday_2`, `weekday_3`, ..., `weekday_7`, with Monday being the baseline day of week)
- Dummy variables for **month of year** (denoted `MONTH_2`, `MONTH_3`, ..., `MONTH_12`, with January being the baseline month)
- **Lagged demand**, which includes demand from the previous five periods as well as the corresponding demand 24 hour ago. From the data exploration section, these periods demonstrated the highest correlation with present demand

To account for the inclusion of lagged demand, the range of the dataset had to be adjusted slightly to remove any rows which had missing lagged demand due to constraints in the date range of the data. Thus, the first 24 hours of the dataset was removed for modelling. From an overall perspective, this should have minimal impact on the trained models when compared with the overall coverage of data available.

Finally, the final dataset was split into three main sets: training, test, and holdout. This was done in order to fairly evaluate the performance of each model.

The training set would be used to determine the model based on the model type and selected hyperparameters if present. The test set would then be used to evaluate which set of hyperparameters for a particular model type has the best predictive performance. Finally, the holdout set is used to simulate a production environment where the model will be used to evaluate data after the model was created.

It is important to note that unlike non time series related modelling which would involve random sampling at a particular ratio, the split of training, test and holdout sets here are determined by set time periods due to the inherent correlation between adjacent data points.

The table below summarises the split used in our modelling section. Due to the large time range present and granularity of our data, there were no concerns on insufficient coverage across any sets. However, it was important to include at least a full year's worth of data in each step to account for potential seasonality within a calendar year.

Table 1: Split between modelling sets

Model.set	Datetime.start	Datetime.end
Training	2010-01-02 00:00:00	2018-07-31 23:00:00
Test	2018-08-01 00:00:00	2020-07-31 23:00:00
Holdout	2020-08-01 00:00:00	2022-08-01 00:00:00

7.2 LASSO

The first model that was trialed was the lasso model, which includes a shrinkage hyper-parameter `lambda` to determine the penalty of additional features to prevent over-fitting.

$$y_i = \sum_j x_{ij}\beta_j + \lambda \sum_j |\beta_j|$$

In order to ensure that the shrinkage feature works as intended, the input data was first scaled using the `scale` function. In this way, certain coefficients would not be penalised harder for having naturally larger numbers over others such that each variable would be equally considered for having their coefficients reduced.

The code snippet below shows the list of `lambda` considered. The key design decision was to ensure that the `lambdas` considered spanned across several orders of magnitude.

```
##  [1]  0.01000000  0.01258925  0.01584893  0.01995262  0.02511886  0.03162278
##  [7]  0.03981072  0.05011872  0.06309573  0.07943282  0.10000000  0.12589254
## [13]  0.15848932  0.19952623  0.25118864  0.31622777  0.39810717  0.50118723
## [19]  0.63095734  0.79432823  1.00000000  1.25892541  1.58489319  1.99526231
## [25]  2.51188643  3.16227766  3.98107171  5.01187234  6.30957344  7.94328235
## [31] 10.00000000
```

The process of deciding which lambda to consider was achieved using cross-validation. Thus, both the training and test sets defined in the previous section was used here. As there was a large span of historical data for cross-validation, the number of folds chosen of five was relatively low.

From the results of cross-validation in the chart below, it can be seen that there is a clear upward trend in MSE as lambda increases. This shows that penalising coefficients of variables has had an adverse effect on predictive performance. Thus, it made sense to choose the smallest lambda from the set of considered values for our final lasso regression model.

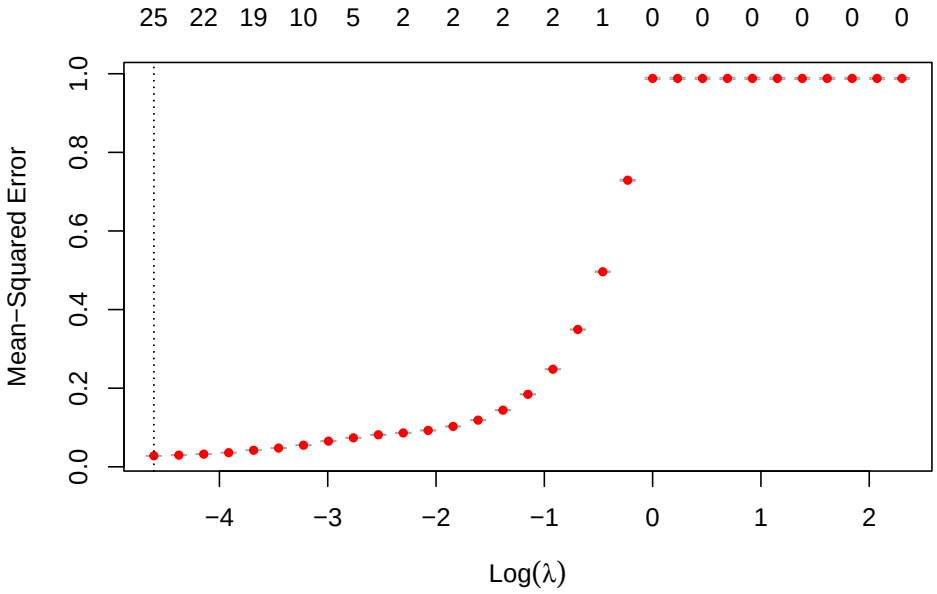


Figure 1: Effect of lambda on MSE

When inspecting the coefficients, it is interesting to see some variables which are known determinants of demand to have shrunk to zero such as temperature. However, this can be explained by the fact that other variables captured such as lagged demand and hour of the day captures these more effectively, resulting in a superior predictive model.

Table 2: Resulting coefficients after shrinkage (part 1)

Variable	Coefficient
TEMPERATURE	.
HOT_DAY	0.0093
HOT_NIGHT	.
COLD_DAY	.
COLD_NIGHT	.
EXTREME_HOT_DAY	.
EXTREME_HOT_NIGHT	.
EXTREME_COLD_DAY	.
EXTREME_COLD_NIGHT	.
PUBLIC_HOLIDAY	0.0018
RAINFALL	.
SOLAR_EXPOSURE	.
HOUR_1	-0.0291
HOUR_2	-0.0283
HOUR_3	.
HOUR_4	0.0152
HOUR_5	0.0563
HOUR_6	0.0817
HOUR_7	0.044
HOUR_8	0.0017
HOUR_9	.
HOUR_10	.
HOUR_11	.
HOUR_12	.
HOUR_13	0.0039
HOUR_14	0.0068
HOUR_15	0.0179
HOUR_16	0.0284
HOUR_17	0.04
HOUR_18	0.0092
HOUR_19	-0.018
HOUR_20	-0.02
HOUR_21	-0.0117
HOUR_22	.
HOUR_23	-0.0088
weekday_2	.
weekday_3	.
weekday_4	.
weekday_5	.
weekday_6	-0.0278

Table 3: Resulting coefficients after shrinkage (part 2)

	Variable	Coefficient
41	weekday_7	-0.0152
42	MONTH_2	.
43	MONTH_3	.
44	MONTH_4	.
45	MONTH_5	.
46	MONTH_6	.
47	MONTH_7	.
48	MONTH_8	.
49	MONTH_9	.
50	MONTH_10	.
51	MONTH_11	.
52	MONTH_12	.
53	demand_lag_1	0.9863
54	demand_lag_2	.
55	demand_lag_3	-0.1714
56	demand_lag_4	.
57	demand_lag_5	.
58	demand_lag_24	0.1351

A quick inspection of the lambda pathway plot reaffirms this, with recent demand being the strongest indicators of future demand. In comparison, the lasso model found that most other coefficients were not as significant, and were therefore shrunk to zero or kept at a very low magnitude.

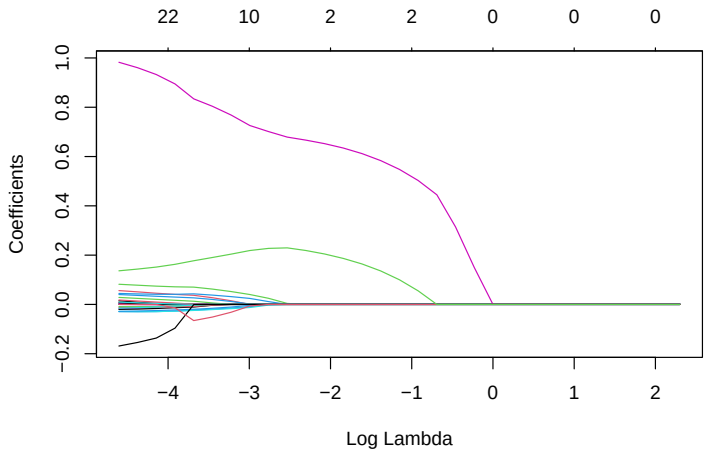


Figure 2: Lambda pathway plot

7.3 Support vector regression (SVR)

Support vector regressions uses support vectors (lines) to help define data points. Similar to lasso regression, the variables first had to be scaled to ensure direct comparisons when separating data points of different classes.

Another consideration when modelling using SVRs is constraints surrounding computational resources. In particular, kernel machines are sensitive to the amount of training data used. Using the full training dataset here resulted in unreasonably long computation times for hyperparamter tuning and large amounts of memory required to store the models. To reduce the impact on computational resources, the training data was cut to the start of **August 2014** onwards.

In addition, the set of features considered in the model was more targeted in SVRs for the same reason. From guidance in the data exploration section, the features used in this model included temperature and the lagged demands established.

Both of these adjustments significantly improved runtime with minimal impacts on the predictive performance of the model.

There were a couple hyperparameters that were considered for SVRs in this project.

- Kernel: algorithm used for pattern analysis. A simple linear kernel was considered, as well as a more complex radial kernel to account for non-linear relationships
- Epsilon: penalisations for errors. The larger the epsilon, the larger the penalisation from errors.

The table below shows the full set of SVR models considered in this project.

Table 4: List of SVR models

Model ID	Epsilon	Kernel
1	0.0	linear
2	0.0	radial
3	0.5	linear
4	0.5	radial
5	1.0	linear
6	1.0	radial

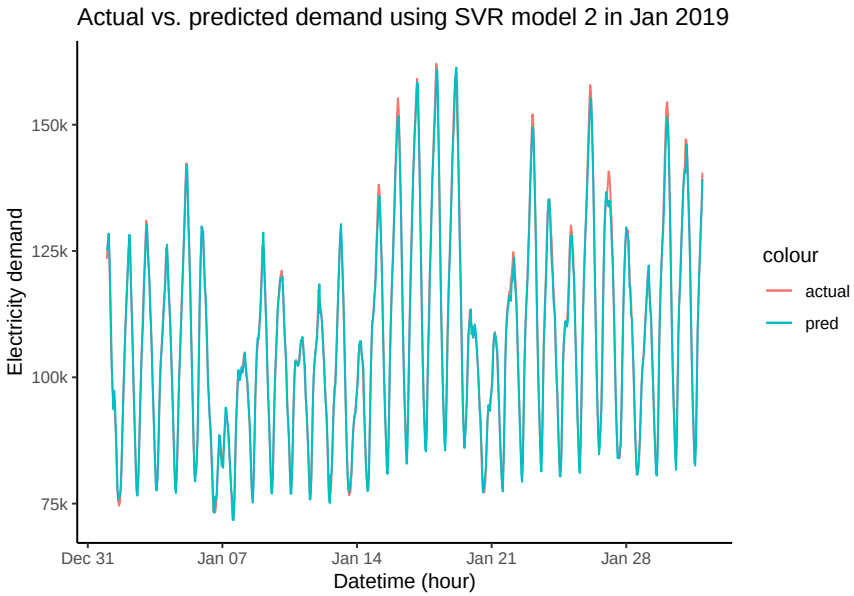
The tuning process involves fitting a model with all the hyperparameter combinations shown above using training data and evaluating their performances using the test data. These metrics would be used as the final SVR model to consider.

Table 5: SVR model performances

model_name	MAPE	MSE	RMSE	MAE	R-Squared
svr_1	1.90%	6,198,630	2,490	1,838	97.13%
svr_2	1.33%	2,952,711	1,718	1,263	98.63%
svr_3	2.34%	7,635,083	2,763	2,223	96.47%
svr_4	2.55%	8,493,121	2,914	2,362	96.07%
svr_5	5.57%	33,413,035	5,780	5,070	84.53%
svr_6	6.37%	45,642,602	6,756	5,819	78.87%

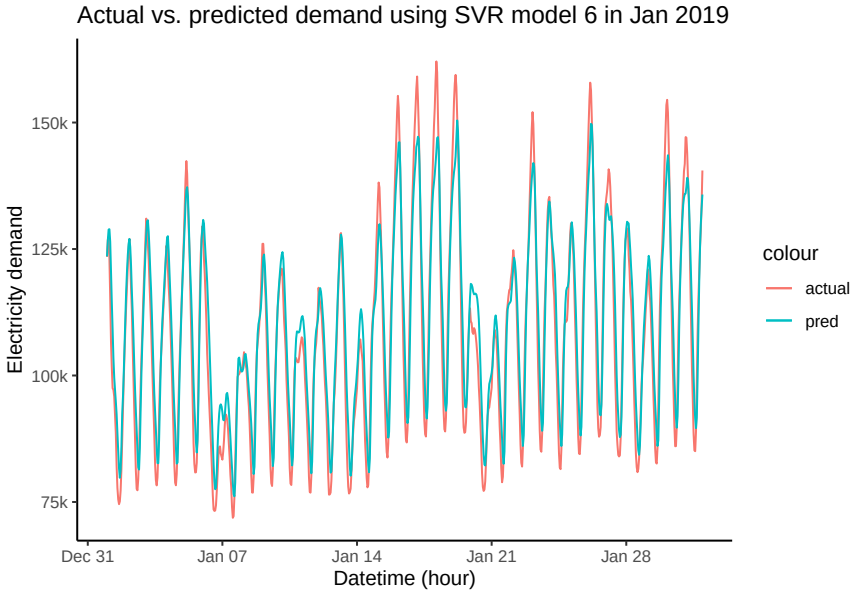
The table above shows that radial SVRs outperform linear SVRs, which can be explained by accounting for non-linear trends. Furthermore, models with lower epsilon values also attributed to better performance. Overall, the second SVR model (radial kernel with epsilon value of zero) performed best and was considered in the overall model selection process.

The charts below compare the fitted versus actual values for the best and worst SVR models from the tuning process:

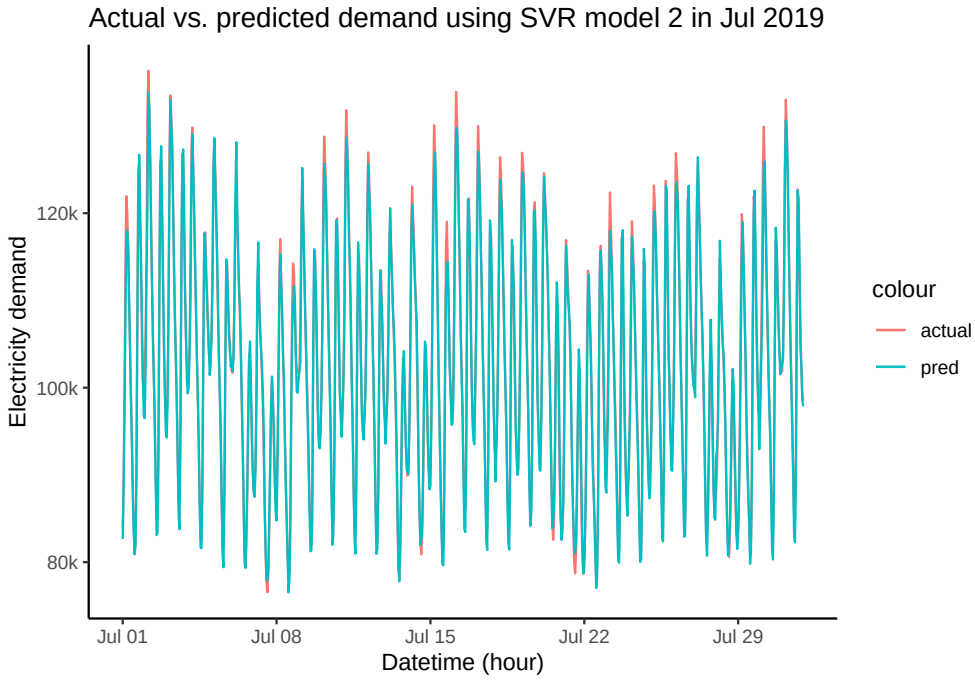


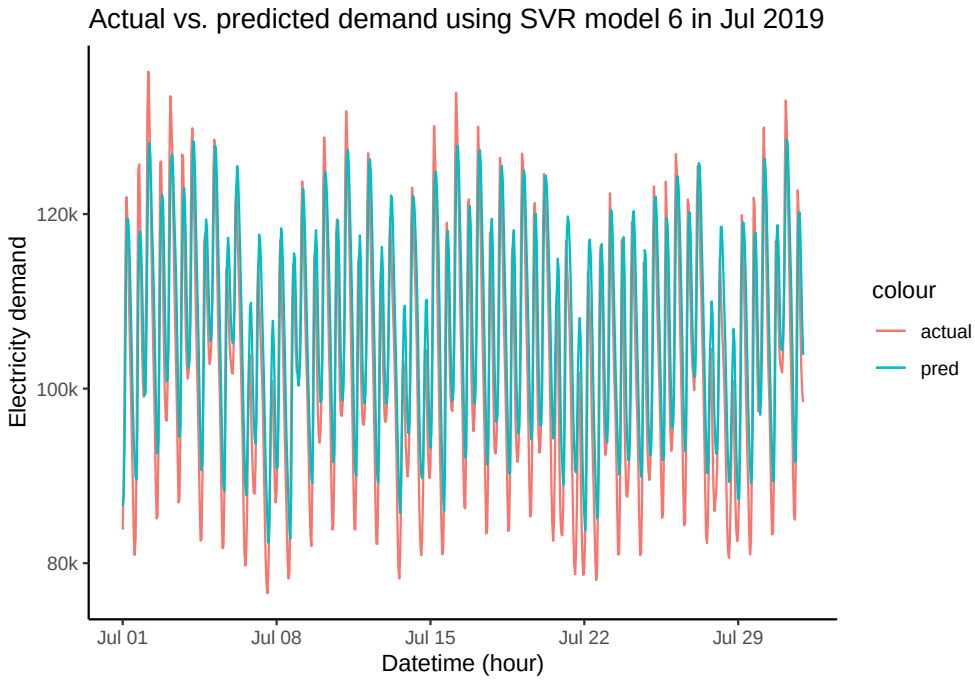
The chart above demonstrates that the best SVR model had strong forecasting performances throughout the month of Jan 2019.

Meanwhile, the chart below, which used an SVR with a linear kernel and an epsilon value of one had difficulties estimating the peaks and troughs of each day, consistently underestimating the magnitudes in either directions.



A similar trend can be seen in winter as well as seen in the two charts below for July 2019.





7.4 Random forest

Random forest regression is a machine learning algorithm that consists of many decision trees.

Similar to SVR, this section aims to find the best hyperparameters as well as perform some light feature selection. The hyperparamters considered for random forest regression include

- number of trees (100, 200, 300, 400, and 500 trees considered)
- mtries, which specifies the number of columns to randomly select at each level (3, 4, or 5)
- sample rate (ranged from 0.5 to 0.8)
- max depth of tree (ranged from 5 to 10)

The tuning process was performed using the `h2o` package in R. Using this package, a search algorithm can be specified to control the runtime of the tuning process. The search algorithm implemented here included:

- using a random discrete strategy when selecting hyperparameters
- set a max runtime of 600 seconds
- stop the process if the MSE has not improved after 5 models

The first model was tuned using all features available in the dataset. Afterwards, a second model was tuned by selecting variables which scored 0.01 or greater in the variance importance metric, which is a measure of the significance of each considered feature. By considering a smaller subset of features, the model may reduce over-fitting, leading to better predictive performance. The chart below shows the variance importance of the first model.

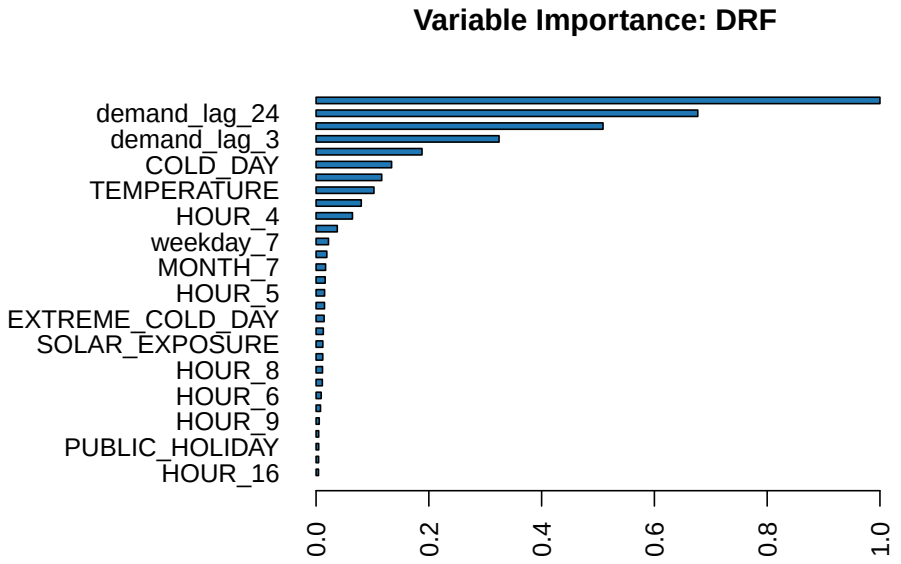


Figure 3: Variance Importance Plot considering all features

The most important measure according to the model was the demand 24 hours prior, which makes intuitive sense as they are highly correlated. Furthermore, temperature and whether it was a cold day also had high importance according to the model.

The second model had a reduction in number of features from 58 to 23 after considering their variance importance measures. The variance importance plot can be seen below.

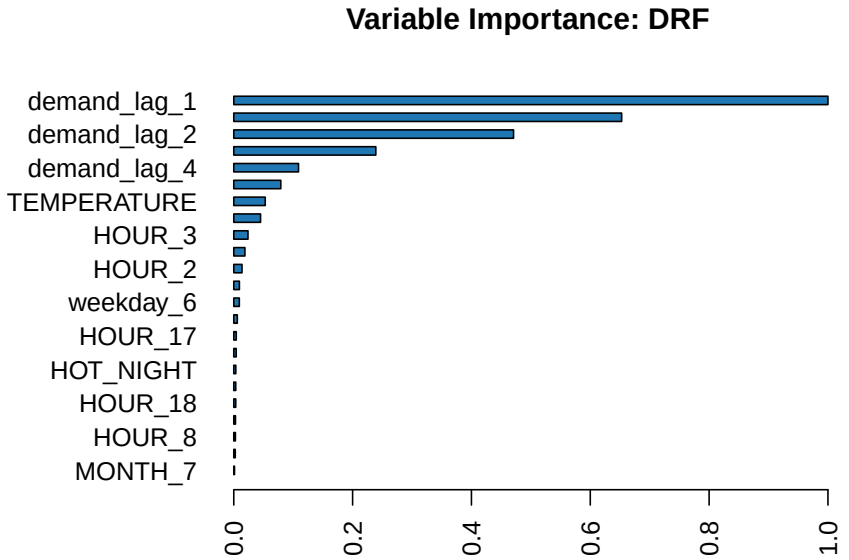


Figure 4: Variance Importance Plot after feature selection

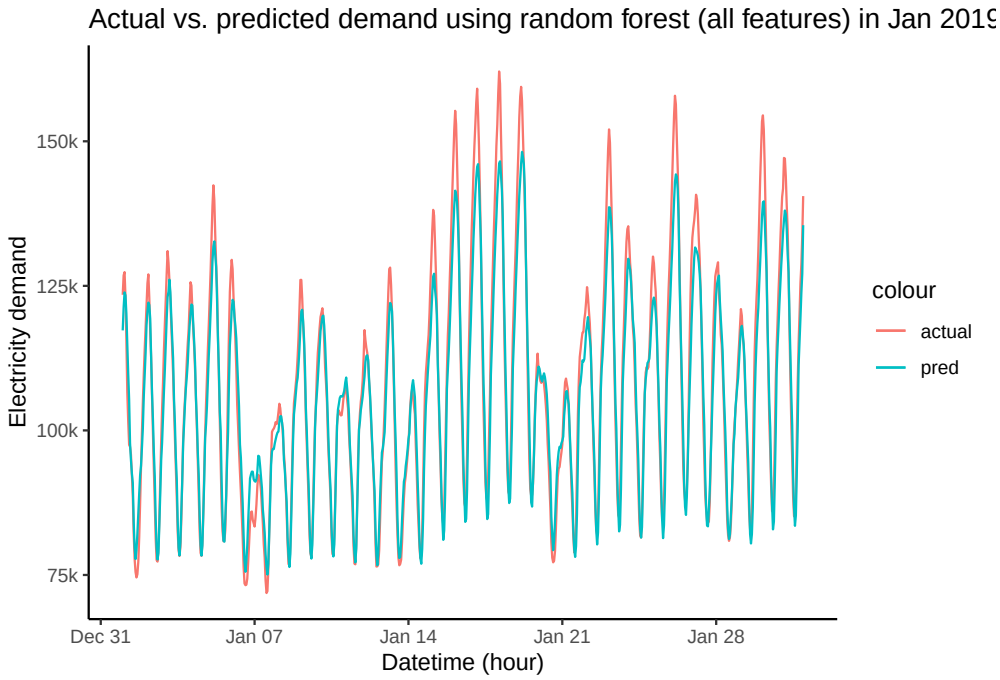
After performing feature selection, the top variables are recent demand values. This is consistent with what was found in the correlation analysis in the data exploration section.

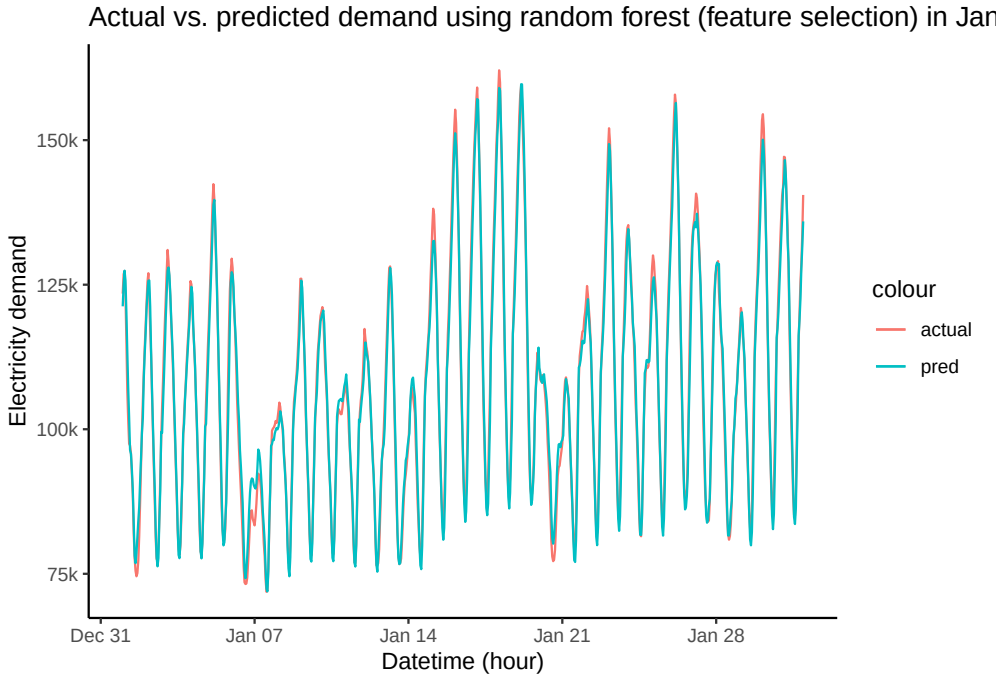
Evaluating the performances of the model with all features compared to the model after feature selection reveals that removing some features did improve the overall performance of the model. However in both cases, there may be some overfitting to the training data as the test set had a relatively significant reduction in performance compared to the training set.

Table 6: Random forest regression model performances

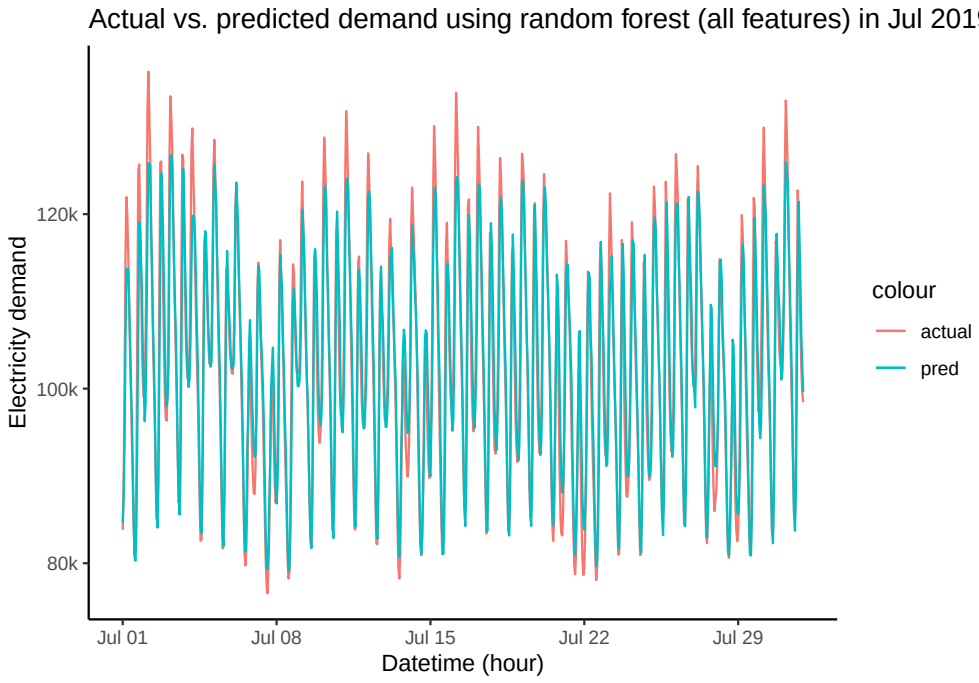
Model name	MAPE	MSE	RMSE	MAE	R-Squared
Training data (all features)	2.88%	13,495,453	3,674	2,796	94.48%
Test data (all features)	3.34%	16,709,373	4,088	3,152	92.26%
Training data (feature selection)	1.69%	4,870,574	2,207	1,658	98.01%
Test data (feature selection)	2.23%	7,868,561	2,805	2,116	96.36%

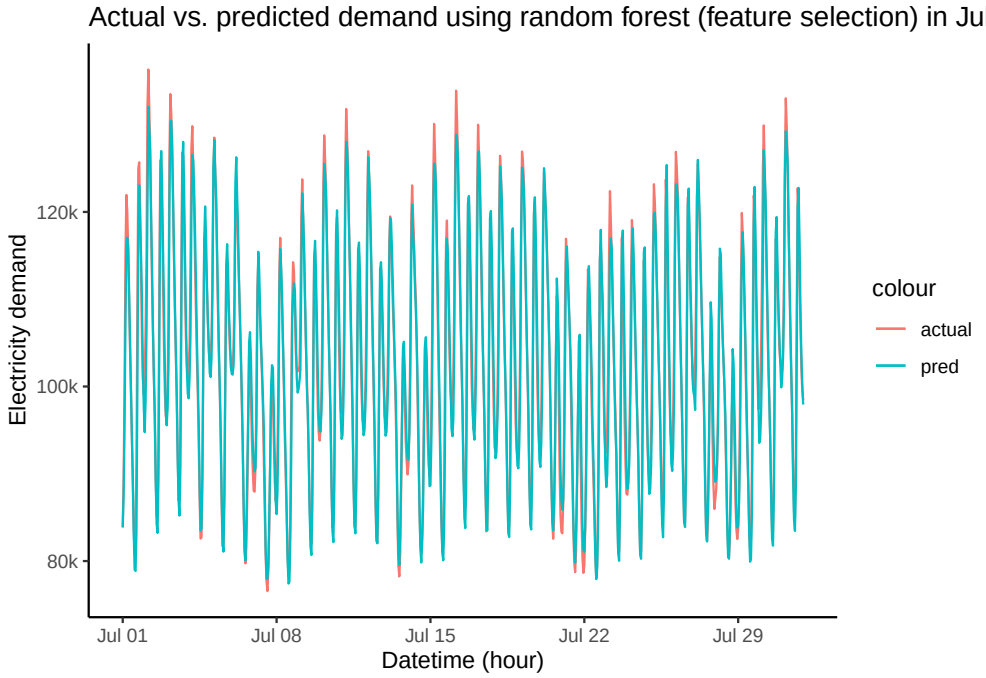
The plots below compare the actual versus fitted values in the test data. Using January 2019 data, the first random forest model had a particularly hard time forecasting the peaks of each day, consistently underestimating the true value. This issue is still present in days with high peak demands in the second random forest model, albeit to a lesser extent.





Again, a similar trend can be found when examining July 2019 fitted versus actual demand, with the feature selection model outperforming the model using all features.





7.5 Final model selection

The final model selection criteria is evaluated by the performance metrics using the holdout set. This is done to simulate a real world forecasting environment to predict unseen data. The AEMO demand forecast was also included as a benchmark to compare our models with.

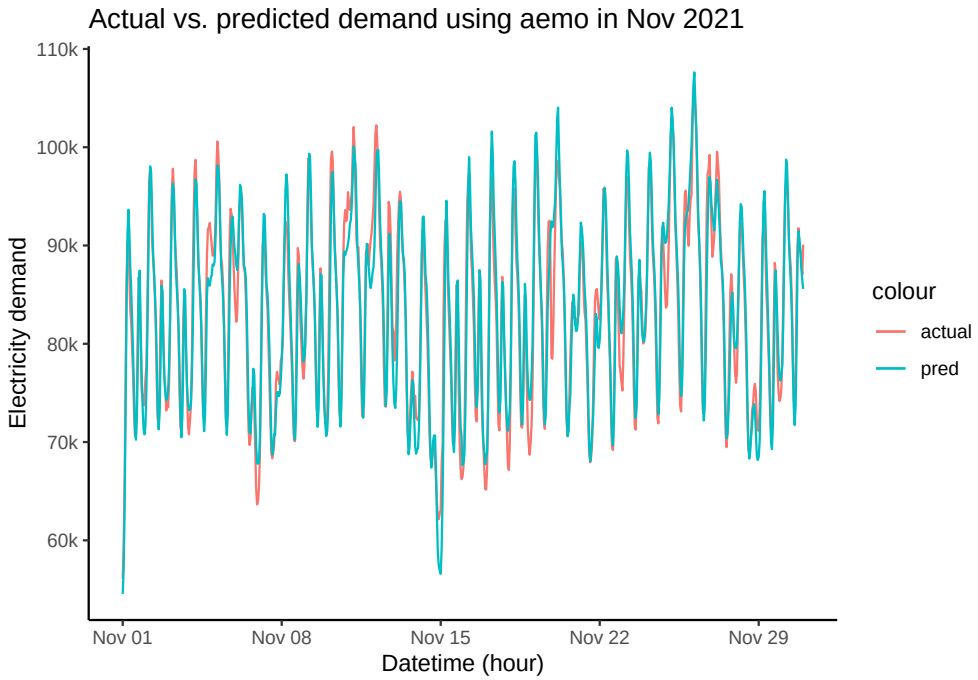
Table 7: Random forest regression model performances

Model name	MAPE	MSE	RMSE	MAE	R-Squared
aemo (benchmark)	3.30%	20,368,620	4,513	3,093	90.93%
lasso	2.38%	9,087,579	3,015	2,158	95.95%
svr	1.90%	5,928,964	2,435	1,684	97.36%
random forest	3.02%	13,631,166	3,692	2,645	93.93%

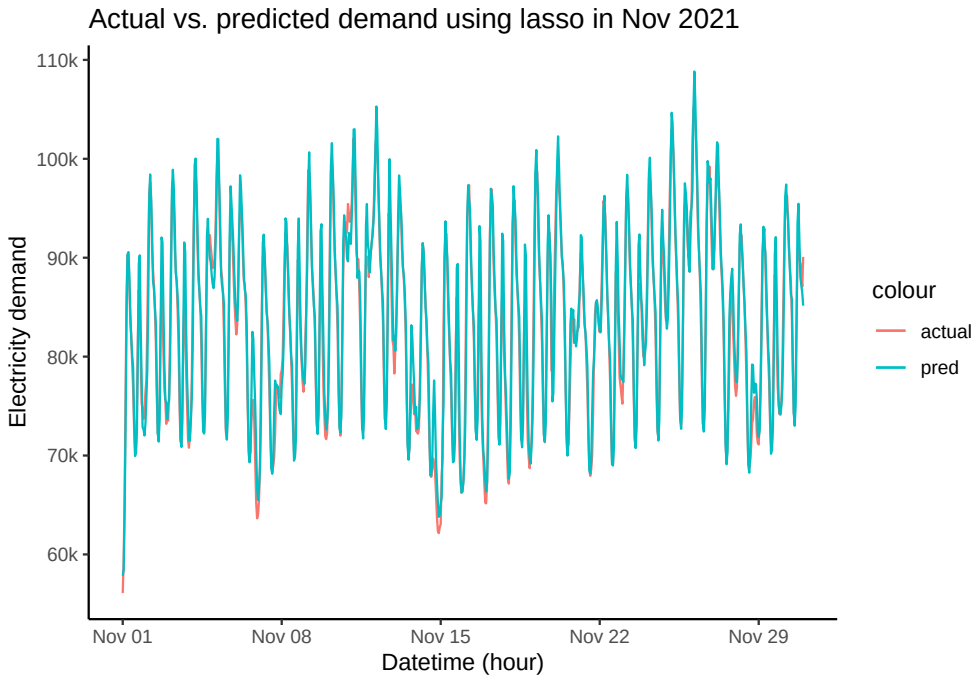
All three models considered in this report managed to out-perform the AEMO benchmark. This can be attributed to the way our models were set up to prioritise predictive performance, while AEMO's model equally considers performance and interpretability.

Interestingly, the random forest model continued to deteriorate in performance as the dataset moved away from the training data, showing some indication of overfitting to the training data. The SVR had the strongest performance overall with over 98% accuracy, and is the final model chosen for our project, being able to outperform aemo by +1.4% accuracy.

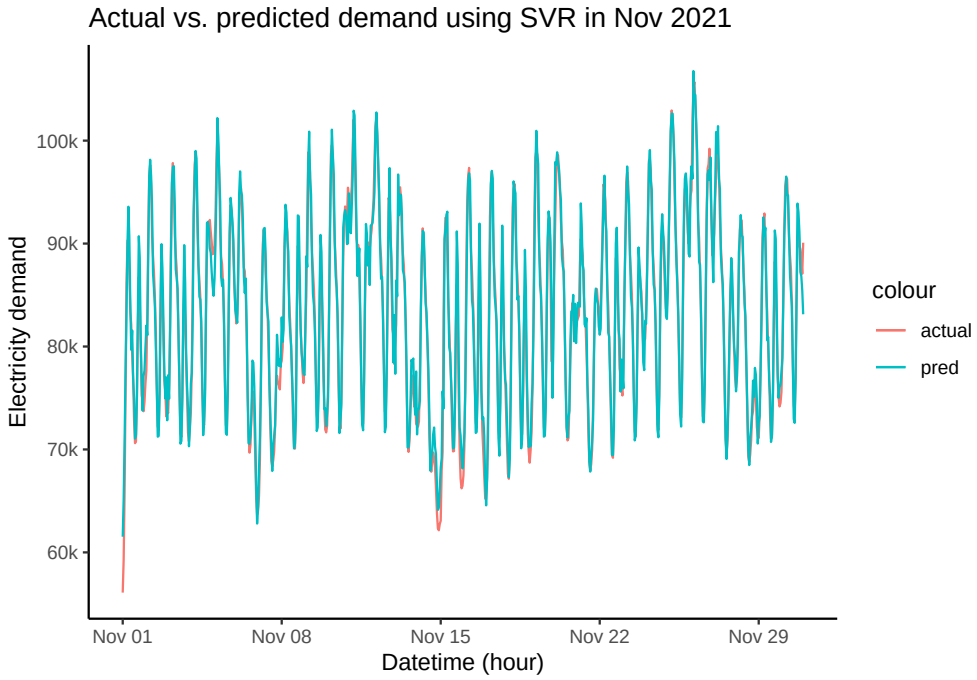
Taking a closer look at one of the months in the holdout set, it can be observed that most of the forecasting error in the aemo model occurs at the peaks and troughs of demand.



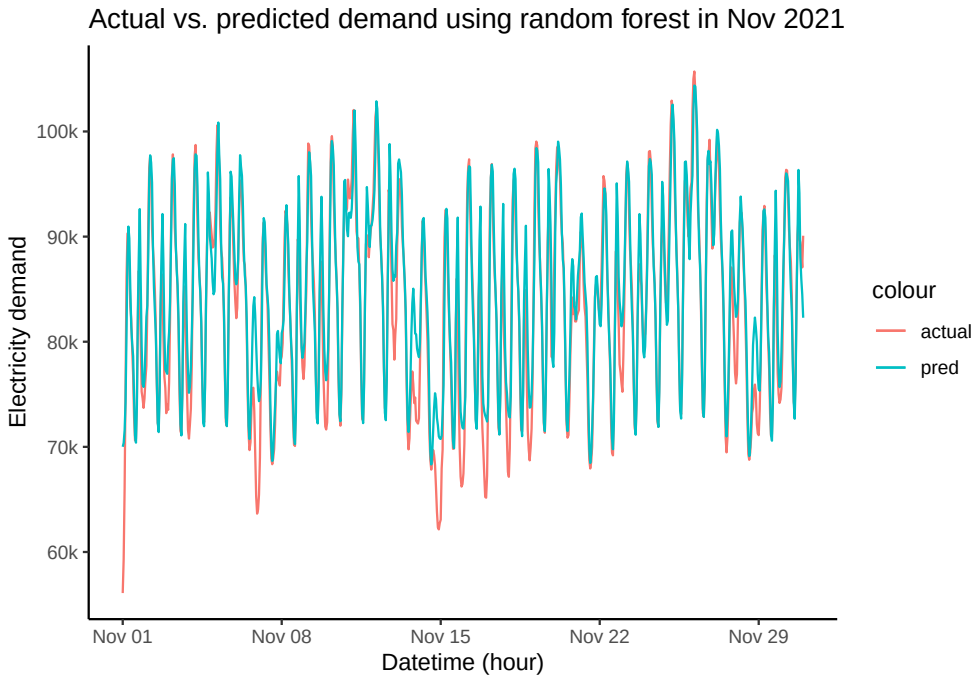
The lasso model fitted in this project shows a similar performance to the aemo model, with slightly more accurate predictions overall.



As reaffirmed by the performance metrics summary, the SVR also looks the most accurate when comparing with actuals in November 2021 as well.



Finally, the random forest model seemed to have the most trouble forecasting the troughs in November 2021, over-estimating energy demands in these periods overall.



8 Discussion

Although the modelling results obtained in this report are promising and have out-performed the AEMO benchmark as we've set out to do, there are still potential improvements to consider when conducting future analysis and modelling.

Exploring more complex models as referenced in the literature review such as Generalised Additive Models [@McCullochJames2020IEDa], Long Short-Term Memory recurrent neural networks [@AlonsoAndr?sM.2020ASSL], and deep Feedforward Neural Networks [@9282124] could address some limitations experienced by the models chosen in this project, such as accounting for seasonality and being more robust against outliers.

Furthermore, some of the machine learning processes surrounding tuning hyperparameters consumed large amounts of computational resources including runtime and storage memory. Thus, some workarounds had to be implemented in this project such as limiting the possible combinations of hyperparameters considered and putting stopping conditions to prevent the algorithms from running for too long. Having access to more resources from this domain could potentially result in more optimised models and thus, better forecasting results.

Another potential aspect which could be considered for future work is utilising multivariate analysis and modelling. The models within this project are limited in scope to only forecasting one hour ahead. However, having a multivariate response variable for consecutive future hours could expand on these results by better accounting for hours of the day and giving the client a better visibility of the demand forecast horizon.

Finally, there were some challenges around data acquisition, resulting in sub-optimal features when it comes to predicting future demand. In particular, the solar and rainfall data used was at a daily level, which is less granular than the hourly demand that was being forecasted. This was evident in the coefficient and variable importance metrics in the lasso and random forest regression models respectively, where both scored relatively low in significance. Having hourly data for weather measurements such as rainfall, solar exposure, wind speed and humidity could further improve on the existing models in this report.

9 Conclusion

–

References

Appendix

```
#------------------------------------------------------------------------#
# file: 01 setup
#
# author: Louis
#
# description: loads library and custom functions
#------------------------------------------------------------------------#


# clear environment ----------------------------------------------------
rm(list = ls())
gc()

# load packages --------------------------------------------------------

# relevant packages for script
package_list <- c(
  "tidyverse"
  , "lubridate"
  , "data.table"
  , "crayon"
  , "zoo"
  , "tsibble"
  , "timeDate"
  , "glmnet"
  , "randomForest"
  , "forecast"
  , "h2o"
  , "scales"
  , "grid"
  , "gridExtra"
  , "fastDummies"
  , "e1071"
  , "knitr"
  , "kableExtra"
)

# list of packages not installed
required_packages <- setdiff(
  package_list
  , rownames(installed.packages())
)

# install any package not installed
lapply(required_packages, install.packages)

# load required packages
suppressMessages(
```

```
  invisible(
    lapply(package_list, library, character.only = T)
  )
)

# initiate h2o for hyperparameter tuning
h2o.init(max_mem_size = "5g") # max 5 gigabytes

# specify functions
year <- lubridate::year
month <- lubridate::month
day <- lubridate::day
hour <- lubridate::hour
minute <- lubridate::minute
select <- dplyr::select

# custom functions ------------------------------------------------------

# PRIMARY_KEY_CHECK
# ensure that there are no duplicates
primary_key_check <- function(df, pk) {

  # start with data.frame
  df %>%

    # confirm data.table
    as.data.table() %>%

    # count rows by primary key
    .[, .(N = .N), by = pk] %>%

    # filter where count exceeds 1
    .[N > 1] %>%

    # count rows
    nrow() %>%

    # print message if greater than 1
    {
      if (. > 0) {
        message(
          paste0(
            "Warning: the primary key ("
            , paste(pk, collapse = ", ")
            , ") is not unique\n"
            , .
            , " cases of duplications found"
          )
        )
      }
    }

}
```

```r
compute_mape <- function(actual, pred){
  mean(
    abs(
      (actual - pred)/actual
    )
  )
}

compute_mse <- function(actual, pred) {
  mean(
    (actual - pred)^2
    , na.rm = TRUE
  )
}

compute_mae <- function(actual, pred) {
  mean(
    abs(actual - pred)
  )
}

compute_rmse <- function(actual, pred) {
  compute_mse(actual, pred) %>%
    sqrt()
}

compute_rsquared <- function(actual, pred) {
  rss <- sum((actual - pred)^2) # residual sum of squares
  tss <- sum((actual - mean(actual))^2) # total sum of squares

  1 - rss/tss
}

summarise_model_performance <- function(
    actual
    , pred
    , model_name = NULL
    , format = T
) {

  # create table of performance metrics
  data.table(
    "model_name"  = model_name
    , "MAPE"      = compute_mape(actual, pred)
    , "MSE"       = compute_mse(actual, pred)
    , "RMSE"      = compute_rmse(actual, pred)
    , "MAE"       = compute_mae(actual, pred)
    , "R-Squared" = compute_rsquared(actual, pred)
  ) %>%

    # format if specified
    {
      if(format) {
```

```
        .[, `:=` (
          MAPE          = label_percent(accuracy = 1e-2)(MAPE)
          , MSE         = label_comma()(MSE)
          , RMSE        = label_comma()(RMSE)
          , MAE         = label_comma()(MAE)
          , `R-Squared` = label_percent(accuracy = 1e-2)(`R-Squared`)
        )
        ]
      }
    } %>%

    # return object
    .[]
}

plot_predictions <- function(
    # model outputs
  actual
  , pred

  # start and end date of data - default to holdout set
  , start_date = "2020-08-01"
  , end_date   = "2022-08-01"

  # model name (for title)
  , model = "lasso"

  # filter to month year for more clarity
  , input_month = 1
  , input_year = 2021
) {

  datetime_seq <- seq(
    from = as.POSIXct(paste0(start_date, "01:00:00"))
    , to = as.POSIXct(paste0(end_date, " 00:00:00"))
    , by = 60 * 60 # every hour
  )

  # combine data into one table
  p <- data.table(
    datetime_hour = datetime_seq
    , actual = pull(as.data.table(actual))
    , pred   = pull(as.data.table(pred))
  ) %>%

    suppressWarnings() %>%

    # filter to specified month and year
    .[month(datetime_hour) == input_month & year(datetime_hour) == input_year] %>%

    # create ggplot
    ggplot(., aes(x = datetime_hour)) +
```

```
    # plot for actuals
    geom_line(aes(y = actual, colour = "actual")) +

    # plot for predictions
    geom_line(aes(y = pred, colour = "pred")) +

    # plot labels
    labs(
      x = "Datetime (hour)"
      , y = "Electricity demand"
      , title = paste0(
        "Actual vs. predicted demand using "
        , model
        , " in "
        , month(input_month, label = T), " ", input_year
      )
    ) +

    scale_y_continuous(labels = label_number(suffix = "k", scale = 1e-3)) +

    # set theme
    theme_classic()

  return(p)
}

model_svr <- function(
    training_data
    , test_data
    , model_data
    , param_epsilon
    , param_kernel = "linear"
    , id = 1
) {

  # run svr model
  svr_model <- svm(
    TOTALDEMAND
    ~ TEMPERATURE
    + demand_lag_1
    + demand_lag_2
    + demand_lag_3
    + demand_lag_4
    + demand_lag_5
    + demand_lag_24

    , data = training_data
    , kernel = param_kernel
    , epsilon = param_epsilon
    , tolerance = 0.1 # help control runtime
  )

  # fit in test set
```

```
  svr_fit_scaled <- predict(
    svr_model
    , newdata = test_data[
      , .(TOTALDEMAND
          , TEMPERATURE
          , demand_lag_1
          , demand_lag_2
          , demand_lag_3
          , demand_lag_4
          , demand_lag_5
          , demand_lag_24)
    ]
  )

  # unscale data
  demand_mean <- mean(model_data$TOTALDEMAND)
  demand_sd <- sd(model_data$TOTALDEMAND)
  svr_fit <- (svr_fit_scaled * demand_sd) + demand_mean

  # performance metrics
  svr_performance <- summarise_model_performance(
    actual =  model_data[model_set == "test", TOTALDEMAND]
    , pred = svr_fit
    , model_name = paste0("svr_", id)
  )

  # return object
  return(
    list(
      model = svr_model
      , fitted_vals = svr_fit
      , performance = svr_performance
    )
  )
}

model_rf <- function(
  # data and column splits
  training_data
  , test_data
  , x_cols
  , y_cols

  # tuning specificities
  , rf_grid
  , rf_search_criteria
  , id = 1
  , seed_num = 123
) {

  # grid identifier
  rf_grid_id <- paste0("rf_grid", id)
```

```
  # perform hyperparameter tuning
  rf_models <- h2o.grid(
    grid_id = rf_grid_id
    , algorithm = "randomForest"
    , x = x_cols
    , y = y_cols
    , seed = seed_num
    , training_frame = training_data
    , hyper_params = rf_grid
    , search_criteria = rf_search_criteria
  )

  # extract the best model
  rf_grid_performance <- h2o.getGrid(
    grid_id = rf_grid_id
    , sort_by = "mse"
    , decreasing = F
  )

  rf_best_model <- h2o.getModel(rf_grid_performance@model_ids[[1]])

  ## create predictions
  rf_pred_training <- h2o.predict(rf_best_model, newdata = training_data)
  rf_pred_test     <- h2o.predict(rf_best_model, newdata = test_data)


  ## assess model performance
  rf_metrics <- rbind(
    summarise_model_performance(training_data[, y_cols], rf_pred_training, paste0("training_", id))
    , summarise_model_performance(test_data[, y_cols], rf_pred_test, paste0("test_", id))
  )

  # plot variance importance
  rf_var <- h2o.varimp(rf_best_model)

  # collect key outputs
  out <- list(
    best_model = rf_best_model
    , metrics = rf_metrics
    , var = rf_var
    , pred = list(
      train = as.data.table(rf_pred_training)
      , test = as.data.table(rf_pred_test)
    )
  )

  # return key elements
  return(out)
}
```

```
# TODO: Hao to add once finalised data exploration code
```

```r
#------------------------------------------------------------------------#
# file: 03 modelling
#
# author: Louis
#
# description: modelling component of electricity demand
#------------------------------------------------------------------------#


# set up ----------------------------------------------------------------

# flag to run dependencies (only first time)
run_dependencies = F

# run required scripts
if (run_dependencies) {
  source("Code/01 Setup.R")
  source("Code/02 Data exploration.R")
}

## key parameters

# overwrite saved models
update_models <- F

# split data (remaining will be in holdout set)
TRAINING_CUTOFF = as.POSIXct("2018-08-01 00:00:00") # cutoff date for training set
TEST_CUTOFF     = as.POSIXct("2020-08-01 00:00:00") # cutoff date for test set

# set of lagged variables to consider
LAG_SET = c(1:5, 24)
SEED_NUM = 123

# data prep -------------------------------------------------------------

## aggregate data to an hourly level
data %>%

  # set to data.table object
  as.data.table() %>%

  # make a copy
  data.table::copy() %>%

  # select required columns
  .[
    , .(
      DATETIME_HOUR = DATEHOUR
      , HOUR
      , MONTH

      , TOTALDEMAND
      , TEMPERATURE
```

```
      , RAINFALL = DAILY_RAINFALL_AMT_MM
      , SOLAR_EXPOSURE = DAILY_GLBL_SOLAR_EXPOSR
      , HOT_DAY
      , HOT_NIGHT
      , COLD_DAY
      , COLD_NIGHT
      , EXTREME_HOT_DAY
      , EXTREME_HOT_NIGHT
      , EXTREME_COLD_DAY
      , EXTREME_COLD_NIGHT
      , PUBLIC_HOLIDAY = pub_holiday_flag
    )
  ] %>%

  .[, weekday := lubridate::wday(DATETIME_HOUR, week_start = 1)] %>%

  dummy_cols(
    select_columns = c("MONTH", "weekday", "HOUR")
    , remove_first_dummy = T
  ) %>%

  # remove columns after dummies are created
  .[, `:=` (
    HOUR = NULL
    , weekday = NULL
    , MONTH = NULL
  )
  ] %>%

  .[order(DATETIME_HOUR)] %>%

  # save output
  force() -> dt_demand_hour

# table for aemo demand forecast
data %>%

  as.data.table() %>%

  .[, .(datetime_hour = DATEHOUR, aemo_demand = FINAL_FORECAST)] %>%

  force() -> dt_aemo

## add split between training, test and holdout sets
# start with hourly table
dt_demand_hour %>%

  # create a copy
  data.table::copy() %>%

  .[, `:=` (
    model_set = fcase(
      # if before Aug 2016, set as training set
```

```
      DATETIME_HOUR <= TRAINING_CUTOFF, "training"

      # if between Aug 2016 and Aug 2017, set as test set
      , DATETIME_HOUR <= TEST_CUTOFF, "test"

      # otherwise, set as holdout set
      , default = "holdout"
    )

  )
  ] %>%

  # save output
  force() -> dt_model

## create variables for lagged demands
lapply(LAG_SET, function(x) {
  dt_model[, paste0("demand_lag_", x) := lag(TOTALDEMAND, x)]
})

# start from time period where there is no missing lag data
dt_model <- dt_model[-c(1:max(LAG_SET)), ]

# list of independent and dependent variables
x_cols <- c(
  "TEMPERATURE"
  , "HOT_DAY"
  , "HOT_NIGHT"
  , "COLD_DAY"
  , "COLD_NIGHT"
  , "EXTREME_HOT_DAY"
  , "EXTREME_HOT_NIGHT"
  , "EXTREME_COLD_DAY"
  , "EXTREME_COLD_NIGHT"
  , "PUBLIC_HOLIDAY"
  , "RAINFALL"
  , "SOLAR_EXPOSURE"
  , paste0("HOUR_", 1:23)
  , paste0("weekday_", 2:7)
  , paste0("MONTH_", 2:12)
  , paste0("demand_lag_", LAG_SET)
)

# dependent variable
y_cols <- "TOTALDEMAND"

factor_cols = c(
  "HOT_DAY"
  , "HOT_NIGHT"
  , "COLD_DAY"
  , "COLD_NIGHT"
  , "EXTREME_HOT_DAY"
  , "EXTREME_HOT_NIGHT"
```

```
  , "EXTREME_COLD_DAY"
  , "EXTREME_COLD_NIGHT"
  , "PUBLIC_HOLIDAY"
  , paste0("HOUR_", 1:23)
  , paste0("weekday_", 2:7)
  , paste0("MONTH_", 2:12)
)

model_cols <- c(x_cols, y_cols)

# get mean and variance for unscaling
demand_mean <- dt_model[, mean(TOTALDEMAND)]
demand_sd   <- dt_model[, sd(TOTALDEMAND)]

# lasso regression ----------------------------------------------------------
dt_model %>%

  data.table::copy() %>%

  # convert factor columns to numeric type
  .[, (factor_cols) := lapply(.SD, as.numeric), .SDcols = factor_cols] %>%

  # scale all variables to ensure shrinkage parameter works properly
  .[, (model_cols) := lapply(.SD, scale), .SDcols = model_cols] %>%

  # save output
  force() -> dt_lasso

# will use cross-validation, so only need one training set
lasso_training_x <- dt_lasso[model_set %in% c("training", "test"), .SD, .SDcols = x_cols] %>% as.matrix
lasso_training_y <- dt_lasso[model_set %in% c("training", "test"), .SD, .SDcols = y_cols] %>% as.matrix

if (update_models) {

  ## set of lambdas to test
  lambda_grid <- 10^seq(-2, 1, by = 0.1)

  # perform k-fold cross validation to determine best lambda
  lasso_cv <- cv.glmnet(
    x = lasso_training_x
    , y = lasso_training_y
    , alpha = 1 # specification fo lasso regression
    , lambda = lambda_grid
    , nfolds = 5
  )

  # minimum lambda
  lasso_lambda_min <- lasso_cv$lambda.min

  # run model with min lambda
  lasso_best_model <- glmnet(
    x = lasso_training_x
    , y = lasso_training_y
```

```
    , alpha = 1 # specification fo lasso regression
    , lambda = lasso_lambda_min
  )

  # collect key outputs
  lasso_model_list <- list(
    best_model = lasso_best_model
    , min_lambda = lasso_lambda_min
    , cv = lasso_cv
  )

  # save model
  saveRDS(lasso_model_list, "models/lasso.RDS")
}

# read lasso model
lasso_model_list <- readRDS("models/lasso.RDS")

# save best lasso model
lasso_best_model <- lasso_model_list$best_model

# support vector regression ------------------------------------------------

# use inputs defined in lasso as variables are scaled
svr_train <- dt_lasso[
  # control training data size for sum's
  model_set == "training" & DATETIME_HOUR >= as.Date("2014-08-01")
  , .SD, .SDcols = model_cols
]
svr_test  <- dt_lasso[model_set == "test", .SD, .SDcols = model_cols]

# combination of elsilon and cost parameters to test
svr_grid <- CJ(
  epsilon = seq(0, 1, by = 0.5)
  , kernel = c("linear", "radial")
) %>%

  .[, id := .I]

if (update_models) {
  # run svr models with each combination of tuning parameters
  lapply(1:svr_grid[, .N], function(x) {
    epsilon = svr_grid[x, epsilon]
    kernel  = svr_grid[x, kernel]

    svr_model = model_svr(
      training_data = svr_train
      , test_data = svr_test
      , model_data = dt_model
      , param_epsilon = epsilon
      , param_kernel = kernel
      , id = x
    )
```

```
    # save each model separately due to large file size
    saveRDS(svr_model, paste0("models/svr_", x, ".RDS"))
  })
}

# read models
svr_models <- lapply(1:svr_grid[, .N], function(x) {
  readRDS(paste0("models/svr_", x, ".RDS"))
})

# summary table comparing performances with different tuning parameters
svr_summary_table <- rbindlist(
  list(
    svr_models[[1]]$performance
    , svr_models[[2]]$performance
    , svr_models[[3]]$performance
    , svr_models[[4]]$performance
    , svr_models[[5]]$performance
    , svr_models[[6]]$performance
  )
)

# compare fitted with actuals
# best model
plot_predictions(
  actual = dt_model[model_set == "test", TOTALDEMAND]
  , pred = svr_models[[2]]$fitted_vals
  , model = "svr model 2"
  , start_date = as.Date("2018-08-01")
  , end_date = as.Date("2020-08-01")
  , input_year = 2019
  , input_month = 1
)

# worst model
plot_predictions(
  actual = dt_model[model_set == "test", TOTALDEMAND]
  , pred = svr_models[[6]]$fitted_vals
  , model = "svr model 6"
  , start_date = as.Date("2018-08-01")
  , end_date = as.Date("2020-08-01")
  , input_year = 2019
  , input_month = 1
)

# save best model
svr_best_model <- svr_models[[2]]$model

# random forest ------------------------------------------------------------

## split into training and test for hyperparameter testing
rf_train <- dt_model[model_set == "training", .SD, .SDcols = model_cols]
rf_test  <- dt_model[model_set == "test"    , .SD, .SDcols = model_cols]
```

```r
# get actuals in test set
actuals_demand_test <- rf_test[, .SD, .SDcols = y_cols]

## perform hyperparameter tuning
# set hyperparameter grid
rf_hyper_grid <- list(
  ntrees        = seq(100, 500, by = 100)
  , mtries      = seq(5, 35, by = 5)
  , sample_rate = c(0.5, 0.8, by = 0.1)
  , max_depth   = 5:10
)

# adjust search criteria to control runtime
rf_search_criteria <- list(
  strategy = "RandomDiscrete"

  # stops after 10 min
  , max_runtime_secs = 600

  # stops if mse has not improved after 5 models
  , stopping_metric = "mse"
  , stopping_tolerance = 1e-4
  , stopping_rounds = 5
)

if (update_models) {

  ## run on base model (all factors included)
  rf_model_1 <- model_rf(
    training_data = rf_train %>% as.h2o()
    , test_data = rf_test %>% as.h2o()
    , x_cols = x_cols
    , y_cols = y_cols
    , rf_grid = rf_hyper_grid
    , rf_search_criteria = rf_search_criteria
    , id = 1
    , seed_num = SEED_NUM
  )

  ## subset features based on variance importance
  x_new <- rf_model_1$var %>%

    # arbitrary cutoff for feature selection
    filter(scaled_importance >= 0.01) %>%

    # get list of variables
    pull(variable)

  # re-run model
  rf_model_2 <- model_rf(
    training_data = rf_train %>% as.h2o()
    , test_data = rf_test %>% as.h2o()
    , x_cols = x_new
```

```
    , y_cols = y_cols
    , rf_grid = rf_hyper_grid
    , rf_search_criteria = rf_search_criteria
    , id = 2
    , seed_num = SEED_NUM
  )

  rf_list <- list(
    model_1 = rf_model_1
    , model_2 = rf_model_2
    , x_new = x_new
  )

  # save model
  saveRDS(rf_list, "models/random_forest.RDS")
  h2o.saveModel(object = rf_model_1$best_model, path = "models", force = TRUE)
  h2o.saveModel(object = rf_model_2$best_model, path = "models", force = TRUE)
}

# read random forest models
rf_list <- readRDS("models/random_forest.RDS")
rf_list$model_1$best_model <- h2o.loadModel("models/rf_grid1_model_132")
rf_list$model_2$best_model <- h2o.loadModel("models/rf_grid2_model_23")

rf_model_1 <- rf_list$model_1
rf_model_2 <- rf_list$model_2

# save final model
rf_best_model <- rf_model_2$best_model

# model selection ------------------------------------------------------------

# set up holdout data set
aemo_holdout_demand <- dt_aemo[datetime_hour > TEST_CUTOFF, aemo_demand]
actual_holdout_demand <- dt_model[model_set == "holdout", TOTALDEMAND]

## aemo benchmark
dt_aemo_summary <- summarise_model_performance(
  actual = actual_holdout_demand
  , pred = aemo_holdout_demand
  , model_name = "aemo (benchmark)"
)

plot_aemo <- plot_predictions(
  actual = actual_holdout_demand
  , pred = aemo_holdout_demand
  , model = "aemo"
)

## forecast validation set

# lasso model
# predict using holdout set
```

```r
lasso_holdout_x <- dt_lasso[model_set == "holdout" , .SD, .SDcols = x_cols] %>% as.matrix()
lasso_holdout_y <- dt_lasso[model_set == "holdout" , .SD, .SDcols = y_cols] %>% as.matrix()

lasso_pred_scaled <- predict(
  lasso_best_model
  , lasso_holdout_x
)

# unscale data
lasso_pred <- (lasso_pred_scaled * demand_sd) + demand_mean

# performance metrics
dt_lasso_summary <- summarise_model_performance(
  actual = actual_holdout_demand
  , pred = lasso_pred
  , model_name = "lasso"
)

# compare fitted with actuals
plot_lasso <- plot_predictions(
  actual = actual_holdout_demand
  , pred = lasso_pred
  , model = "lasso"
)

# support vector regression
svr_holdout <- dt_lasso[model_set == "holdout", .SD, .SDcols = model_cols]

# fit in test set
svr_pred_scaled <- predict(
  svr_best_model
  , newdata = svr_holdout[
    , .(TOTALDEMAND
        , TEMPERATURE
        , demand_lag_1
        , demand_lag_2
        , demand_lag_3
        , demand_lag_4
        , demand_lag_5
        , demand_lag_24)
  ]
)

# unscale data
svr_pred <- (svr_pred_scaled * demand_sd) + demand_mean

# performance metrics
dt_svr_summary <- summarise_model_performance(
  actual = actual_holdout_demand
  , pred = svr_pred
  , model_name = "svr"
)
```

```
# random forest
rf_holdout <- dt_model[model_set == "holdout", .SD, .SDcols = model_cols] %>% as.h2o()
rf_pred <- h2o.predict(rf_model_2$best_model, newdata = rf_holdout)

# performance metrics
dt_rf_summary <- summarise_model_performance(
  actual = rf_holdout[, "TOTALDEMAND"]
  , pred = rf_pred
  , model_name = "random forest"
)
```