Predicting Electricity Demand with Precision: Machine Learning Models at Work (Winter 2023)

Courtney Manhart, Martin Jamouss, Arturs Apinis

***Abstract*— Almost any economic activity relies on stable energy supply. This study seeks to construct a machine learning model that predicts demand for electricity in Victoria, Australia. Utilizing daily electricity demand data spanning over 5 years, we construct several different machine learning models and compare their performance in forecasting how much electricity is used on a certain day. After computing the mean squared errors (MSE) of all evaluated models, we find that the model that performs the best is a stacked ensemble model using scaled data and a testing set size of 50%. The stacked ensemble model in this study consists of five distinct machine learning models: a Linear Regression model, an Elastic Net model, a K-Nearest Neighbors (KNN) model, a Regression Tree model, and a Feed-Forward Neural network model, all of which were trained and tested at an earlier stage in the study. With an R^2 of 48.99%, our model has the potential to provide useful information for policymakers in the energy sector.**

I. INTRODUCTION

THE energy sector is the engine of economic activity and almost all businesses and households rely on stable energy supply. In order for utility companies to correctly choose how much energy should be generated on a certain day, it is of vital importance to have a clear picture of what determines energy demand. The purpose of this project is to explore different machine learning algorithms and evaluate their effectiveness at predicting the demand of electricity in megawatt-hours (MWh). The algorithms used throughout this report are used to solve supervised, regression learning problems where the output is continuous and known. To accomplish this, we first explored and preprocesses the data. Next, we utilized training and testing sets to train multiple machine learning algorithms and stored their mean squared errors (MSEs) to be used for model comparison. Then, we compared the models across training and testing set sizes, across models, and across scaled and unscaled data to select the best performing model overall. At the end of this project, we evaluate the best performing model and report our conclusions.

II. DATA EXPLORATION

The data used for this project includes 2016 observations about the demand for electricity and thirteen features that are correlated with electricity demand, spanning from January 1, 2015, until October 6, 2020. The data is collected in Victoria, the second largest state in Australia[1].

A. *Variable Description*

This dataset includes fourteen different features (including the target features) that are used to create a predictive model for demand of electricity. The feature *date* records the date of the observation and *demand* is our target variable which records the total daily electricity demanded (in MWh). Additionally included in the dataset are two demand related features, *demand_pos_RRP* and *demand_neg_RRP,* that correspond to the total daily demand at a positive recommended retail price (RRP) and the total daily demand at a negative RRP, respectively. Similarly, the dataset includes a feature that records the recommended retail price (RRP) in Australian dollars per MWh (AUD/MWh), *RRP*, along with the related averaged positive RRP and negative RRP, weighted by the corresponding intraday demand in AUD/MWh, *RRP_positive* and *RRP_negative* respectively. The dataset also records the fraction of a day when electricity was traded at a negative RRP, *frac_at_neg_RRP*.

The dataset also includes other features that are characteristics of the date of the observation. These include the maximum and minimum temperatures of the day (in Celsius), *max_temperature* and *min_temperature* respectively. Total daily sunlight energy (in MJ/m^2), *solar_exposure*, and daily rainfall (in mm), *rainfall*, are also recorded in the dataset. Lastly, the dataset contains binary variables for whether students were in school, *school_day*, and whether the day in question was a state or national holiday, *holiday*.

B. Data Processing

For machine algorithms to work properly, data needs to be cleaned and formatted to fulfill all necessary assumptions of machine learning models. This includes creating indicator variables for categorical variables, scaling data when appropriate, and creating new features from existing ones that could be useful in a model (i.e., transforming a date). Although the dataset was relatively clean to begin with, it was necessary to add new features and create indicator variables for the models to run smoothly.

First, to utilize features of dates, we created two new features - month and day - to capture relationships between demand for electricity and time. The goal of this transformation is to enable predictions for specific days at different time during a year. Hence, we are not using the time-series trend or measures of seasonality for our predictions. After transforming the data, we deleted the previous time variable.

In addition to creating variables for months and days, we created indicator variables for the school day and holiday features in the original dataset. These features originally record 'Yes' or 'No' results. Consequently, we turned these into binary variables by replacing 'Yes' with one and 'No' with zero. These binary variables are useful in the regression to account for possible differences in the demand for electricity on days that are school days and similarly holidays.

Finally, we transformed the data using a standard scalar that normalizes the data. For each feature, the standard scalar scales the data by subtracting the mean from every observation and then dividing the difference by the standard deviation of the feature:

$$standard\,scalar(X) = \frac{x - \mu_X}{\sigma_X}$$

The purpose of normalization and scaling the features is to guarantee that all features are on a similar scale when passed into machine learning algorithms, i.e., all features have the same range of values. While this hinders the direct interpretation of potential coefficients, - we would need to descale the coefficients to be able to interpret them - scaling the features is helpful in determining the importance of features in the overall prediction as well as the directional relationship of the feature and the target. Excluding the target feature from scaling and normalization, we scale and normalize all input features in both the training and testing sets. Scaling has several advantages, such as faster computational speed and lower computational expense. Furthermore, scaling can improve the performance and accuracy of a machine learning model and prevent biased and inaccurate results.

C. Feature Selection

The purpose of feature selection is to decrease the number of features in the model to achieve better model performance and to avoid overfitting. While more features could help enhance the model by reducing bias, we could be fitting the model too well so that potential outliers and new observations become sensitive to the model and break the model down. By selecting the features that are the most important to the model, we can extract more reliable results from our analysis.

To complete feature selection, we first ran a cross-validation on a Lasso regression. A Lasso regression helps to handle the bias-variance tradeoff. The bias-variance tradeoff explains that, as a model gets more complex, the bias of the model decreases while the variance of the model increases. A Lasso regression helps finding the optimal model complexity where the bias and variance are simultaneously minimized. To do this, a Lasso regression helps shrink irrelevant estimators (coefficients of features) to zero by assigning a penalty term, called lambda, to the loss function. Consequently, a Lasso regression tries to minimize the following cost function:

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{i,j}\beta_j)^2 + \lambda \sum_{j=1}^{p}|\beta_j|$$

where the y_i represent the target for every sample i = 1, 2, ..., n, $x_{i,j}$ is the jth feature for every observation, j is the coefficient of the jth feature in the regression and λ's the lasso penalty term. The goal of this minimization is to reduce the estimators that are irrelevant for the model down to zero. For Lasso to reduce the estimators properly, the features supplied to the model must be normalized to the same scale as the magnitude of the coefficients matters.

Cross-validation is the process of using the training set for both training and validation purposes. Cross-validations are useful to evaluate a model's performance, flag potential problems of overfitting, and to identify optimal hyperparameters. To perform cross-validation, the algorithm splits the data into a specified number of "folds" or sets of equal size (i.e., K-folds). Then, the algorithm will fit a model to K-1 of the folds, leaving out one fold on every iteration to be used for testing/validation. We used five folds in our estimation. The algorithm then assigns a score to each fold and stores the results. These scores can then be used to compare model performance across different hyperparameters to determine the optimized results.

From our cross-validation, we identified λ=13.3 as optimal for a Lasso model. Using this optimal, we fit a Lasso regression on a training set that consisted of 80% of the original scaled data. From this regression, the coefficients were stored and visualized to evaluate which coefficients reduce to zero and are thus eliminated from the model. The visualization can be seen in Figure 1 below. From these results, it appears that the only features important in predicting the demand for electricity are *demand_pos*, which represents the total daily demand at a positive recommended retail price, and *demand_neg,* which represents the total daily demand at the negative recommended retail price. After further investigation into the data, it was determined that these two features sum to demand for every observation in our dataset. Therefore, we decided to remove these observations as it impacts the results of the analysis.

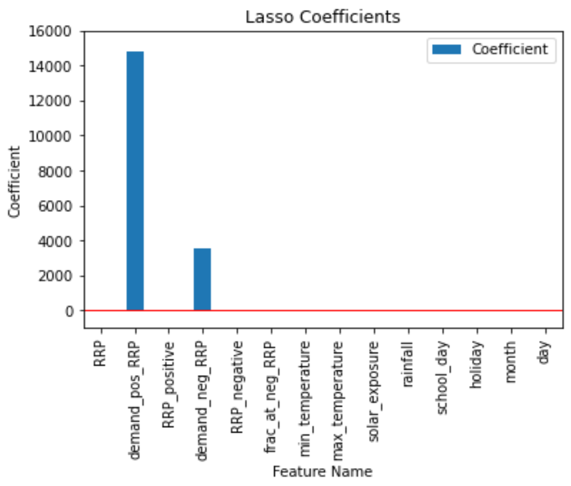


Figure 1. Lasso Feature Selection using all features of the dataset.

With the removal of *demand_pos* and *demand_neg*, another cross-validation was performed to find the new optimal λ = 948.8. A new Lasso regression was trained using the new optimal lambda and a scaled training set that no longer included these two features. From this regression, we then selected those features whose coefficients are not equal to zero. The features that were selected include: the recommended retail price (*RRP)*, the fraction of the day when demand was traded at a negative RRP (*frac_at_neg_RRP)*, the minimum temperature during the day (*min_temperature)*, the total daily sunlight energy (*solar_exposure)*, the daily rainfall (*rainfall*), if students were in school on the day (*school_day*), if the day was a state or national holiday (*holiday*), and the month (*month)*, visualized below in Figure 2.

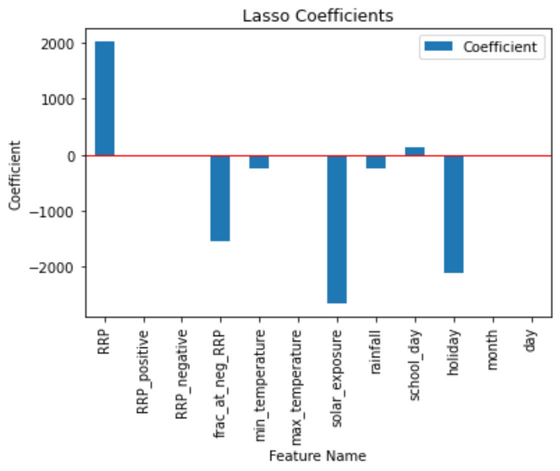


Figure 2. Lasso Feature Selection without *demand_pos* and *demand_neg*.

D. *Training and Testing Sets*

To train a machine learning model and evaluate its performance, the data needs to be split into two sets, a training and a testing set. The training set represents "past" knowledge or information and is used to train the model and determine estimators. The testing set represents "future" knowledge or information. The model estimators from the training model are applied to the testing set to predict outcomes. Because this project involves supervised learning, we have the true outcomes that we can use to determine the performance of the model's predictions.

To create the training and testing sets, we first select the size, or proportion of data, for the testing set. Based on this size, the data is randomly selected without replacement into the training set or the testing set. For this project, we varied the size of the testing set to 10%, 20%, 30%, 40%, and 50% of the original data.

The size of training size when splitting the data can play a significant role in the performance of a machine learning model. One of the goals of a machine learning model is to generalize well to unseen data. A training size that is too small might not provide enough data for the model to effectively learn the underlying patterns, leading to underfitting; however, a training size too large might result in overfitting a model. Consequently, there is a tradeoff between generalization and overfitting that occurs when determining training and testing set sizes. However, in practice, by Empirical Risk Management, the larger the sample for training gets (increasing the size of the training set), the more likely the training set will represent the distribution of the features and the model will reflect the true labeling function. Further, as we increase the size of our training set, the training errors will be increasingly closer to the true error.

III. Machine Learning Models

Using the preprocessed and cleaned data, various supervised, regression machine learning models were trained on the data. Different training-testing split sizes were used for each model as well as both scaled and unscaled data. The mean squared error (MSE) scores were stored for each model profile to be used later for model comparison. In addition, for each model, we performed a 5-fold cross-validation on the testing set and computed the MSEs for each fold. We then took the average of these MSEs to store as part of our model comparison.

A. *Linear Regression*

The Linear Regression model tries to fit a linear trend to the data through the "line of best fit", given the training data provided. To do so, the algorithm tries to minimize a cost function to determine the optimal estimates for prediction. The linear regression model with multiple features was set up in the following way, where y_i represents the target for every sample i=1, 2, ..., n, $x_{i,j}$ is the jth feature for every observation (with $x_{i,0}$= 1 being a constant), j is the coefficient of the jth feature in the regression:

$$y_i = f_\theta(x_i) = \theta_0 + \theta_1 x_{i,1} + \dots + \theta_p x_{i,p} = \sum_{j=0}^{p} \theta_j x_{i,j}$$

The optimal θ_j's were determined by gradient descent with the goal to minimize the following cost function:

$$F(\theta) = \frac{1}{2p}\sum_i (y_i - f_\theta(x_i))^2$$

To implement the Linear Regression model on our data, we first set the training and testing set sizes. We then fit the linear regression algorithm on the training set to determine the optimal θ_j's. The trained model was then applied to the testing set to calculate predicted demand values. With these predicted demands, the MSEs were calculated to later be compared across training set sizes, with other models, and between scaled and unscaled data. The MSEs for both the In-sample (training set), Cross-Validation, and Out-of-Sample (testing set) sets as well as for the scaled data and the unscaled (normal) data are visualized in the following chart. The best performing Linear Regression model uses either scaled or unscaled data with a testing set size of 50%.

	0.1	**0.2**	**0.3**	**0.4**	**0.5**
In-sample MSE unscaled	1.483082e+08	1.463671e+08	1.383394e+08	1.368715e+08	1.351122e+08
Out-sample MSE unscaled	1.364229e+08	1.503473e+08	2.241257e+08	2.049151e+08	1.871158e+08
Cross-val. MSE unscaled	1.364229e+08	1.503473e+08	2.241257e+08	2.049151e+08	1.871158e+08
In-sample MSE scaled	1.483082e+08	1.463671e+08	1.383394e+08	1.368715e+08	1.351122e+08
Out-sample MSE scaled	1.364229e+08	1.503473e+08	2.241257e+08	2.049151e+08	1.871158e+08
Cross-val. MSE scaled	1.364229e+08	1.503473e+08	2.241257e+08	2.049151e+08	1.871158e+08

Table 1. Linear Regression Mean Squared Error results.

B. Elastic Net

The Elastic Net model is a form of linear regression algorithm that includes an L1 (Lasso) and L2 (Ridge) penalties. The L1 and L2 penalties are defined as follows:

$$L1 = \lambda \sum_{j=1}^{p} \left|\beta_j\right| \qquad L2 = \lambda \sum_{j=1}^{p} \beta_j^2$$

The Elastic Net model uses a weighted combination of the L1 and L2 punishments that allows the linear regression to act more like a Ridge regression or more like a Lasso regression. In either case, the elastic net imposes a stricter punishment on the linear regression's cost function:

$$F(\beta) = \frac{1}{2p}\sum_i (y_i - f_\beta(x_i))^2 + \alpha\lambda \sum_{j=1}^{p} \left|\beta_j\right| + (1-\alpha)\lambda \sum_{j=1}^{p} \beta_j^2$$

The gradient descent algorithm can be performed on this cost function for the elastic net to determine the optimal β_j's for the elastic net model. Additionally, a cross validation can be performed to determine the optimal α(L1 ratio) and λ's (the penalty term) for the model.

To implement the Elastic Net algorithm on our data, we first set the training and testing set sizes. We then performed a cross-validation to determine the optimal L1 ratio and the penalty term. Using these optimal hyperparameters, we then fit the elastic net algorithm on the training set to determine the optimal β_j's. The trained model was then applied to the testing set to calculate predicted demand values. With these predicted demands, MSEs were calculated to later be compared across training and testing set sizes, with other models, and between scaled and unscaled data. The MSEs for both the In-sample (training set), Cross-Validation, and Out-of-Sample (testing set) sets as well as for the scaled data and the unscaled (normal) data are described in Table 2. Our best performing Elastic Net model with scaled data has a testing size of 50% with a learning rate of 0.1 and an L1 ratio of 0.9. Similarly, the best performing Elastic Net model with unscaled data has a testing size of 50% with a learning rate of 0.1 and an L1 ratio of 0.9.

	0.1	**0.2**	**0.3**	**0.4**	**0.5**
In-sample MSE unscaled	1.527445e+08	1.505094e+08	1.416393e+08	1.400831e+08	1.381912e+08
Out-sample MSE unscaled	1.375387e+08	1.553584e+08	2.417212e+08	2.215387e+08	1.997692e+08
Cross-val. MSE unscaled	1.830431e+08	1.760276e+08	1.443003e+08	1.437149e+08	1.426678e+08
In-sample MSE scaled	1.523795e+08	1.504807e+08	1.383587e+08	1.368950e+08	1.351323e+08
Out-sample MSE scaled	1.362089e+08	1.527469e+08	2.174270e+08	1.986240e+08	1.830578e+08
Cross-val. MSE scaled	1.545413e+08	1.530096e+08	1.409056e+08	1.402903e+08	1.395484e+08

Table 2. Elastic Net Regression Mean Squared Error results.

C. Regression Tree

A decision tree is a supervised learning algorithm used to solve both classification and regression problems. As its name suggests, this algorithm has a hierarchical tree structure and can have different maximum depths, which is a hyperparameter that should be tuned to attain optimal predictions. Since regression trees are considered weak learners in machine learning, we implemented AdaBoost, which is a type of ensemble learning, to improve the accuracy and performance of our regression tree model. Subsequently, we fine-tuned the hyperparameters alpha (learning rate), the number of estimators (n_estimators), and maximum depths using cross-validation grid search to achieve optimal results. As summarized in Table 3, Our best performing Regression Tree model with unscaled data has a testing set size of 20% and has a max depth of 6, a learning rate of 0.1, and uses 300 estimators. Similarly, the best performing Regression Tree model with scaled data has a testing set size of 20% and it has a max depth of 6, a learning rate of 0.1, and uses 300 estimators.

	0.1	**0.2**	**0.3**	**0.4**	**0.5**
In-sample MSE unscaled	6.665698e+07	6.534182e+07	6.410580e+07	6.227244e+07	5.924673e+07
Out-sample MSE unscaled	1.010460e+08	9.824585e+07	1.008188e+08	1.000611e+08	1.004821e+08
Cross-val. MSE unscaled	9.391086e+07	9.258978e+07	9.471615e+07	9.604293e+07	9.696260e+07
In-sample MSE scaled	6.670282e+07	6.559857e+07	6.555218e+07	6.156974e+07	5.780130e+07
Out-sample MSE scaled	1.000416e+08	9.761402e+07	1.011643e+08	9.879201e+07	9.986103e+07
Cross-val. MSE scaled	9.432779e+07	9.282675e+07	9.505582e+07	9.622380e+07	9.682564e+07

Table 3. Regression Tree Mean Squared Error results.

D. K-Nearest Neighbors (KNN)

The K-Nearest Neighbor (KNN) algorithm is used to predict data points based on the K data points closest to it using Euclidean distances. To predict a value, the algorithm takes the average of the K nearest points outcomes.

To implement the KNN algorithm on our data, we first set the training and testing set sizes. We then performed a cross-validation to determine the optimal number of neighbors (K) to use in order to determine the optimal predictions. Using the optimal K, we then fit the KNN algorithm on the training set. The trained model was then applied to the testing set to predict electricity demand. With these predicted demands, the mean squared errors (MSEs) were calculated to later be compared across training and testing set sizes, with other models, and between scaled and unscaled data. The MSEs for both the In-sample (training set), Cross-Validation, and Out-of-Sample (testing set) sets as well as for the scaled data and the unscaled (normal) data are summarized in Table 4. The best performing KNN model with unscaled and scaled data has a testing set size of 20% with 10 neighbors.

	0.1	**0.2**	**0.3**	**0.4**	**0.5**
In-sample MSE unscaled	9.406714e+07	9.194903e+07	9.266247e+07	9.280826e+07	9.524464e+07
Out-sample MSE unscaled	1.190887e+08	1.341609e+08	1.298858e+08	1.330325e+08	1.298374e+08
Cross-val. MSE unscaled	1.157701e+08	1.147662e+08	1.154312e+08	1.156168e+08	1.177788e+08
In-sample MSE scaled	8.187684e+07	8.140194e+07	8.229258e+07	8.422209e+07	8.621376e+07
Out-sample MSE scaled	1.066534e+08	1.112351e+08	1.102405e+08	1.133300e+08	1.128440e+08
Cross-val. MSE scaled	1.037193e+08	1.025153e+08	1.041894e+08	1.044910e+08	1.068516e+08

Table 4. K-Nearest Neighbor Mean Squared Error results.

E. Feed-Forward Neural Network (FFNN)

The Feed-Forward Neural Network (FFNN) model is the simplest form of an artificial neural network as data is processed only in one direction. An input layer, one or more hidden layers, and an output layer make up the fundamental components of an FFNN. A neural network with multiple hidden layers is known as a deep neural network. Activation functions such as Softplus, sigmoid, and rectified linear unit (ReLU) are typically applied in the hidden layers. The connections between nodes contain weights and biases that are optimized during the training of a neural network.

To identify the best architecture for our model, we performed cross-validation grid search across several architectures. The optimal architecture for our FFNN with unscaled data uses a testing set size of 30% and consists of four hidden layers with 10, 20, 40, and 60 neurons, respectively, and a learning rate of 0.1. Additionally, the optimal architecture for our FFNN with scaled data uses a testing set size of 30% and consists of four hidden layers with 10, 30, 50, and 70 neurons, respectively and a learning rate of 0.1.

	0.1	**0.2**	**0.3**	**0.4**	**0.5**
In-sample MSE unscaled	9.383349e+08	9.553988e+08	8.133723e+08	1.000276e+09	1.120747e+09
Out-sample MSE unscaled	8.967690e+08	9.271882e+08	2.167350e+09	3.683910e+09	3.491485e+09
Cross-val. MSE unscaled	1.363423e+09	1.507343e+09	1.083541e+09	1.318759e+09	1.646440e+09
In-sample MSE scaled	1.235538e+08	1.477600e+08	1.345348e+08	1.591133e+08	2.272530e+08
Out-sample MSE scaled	1.501263e+08	1.477106e+08	4.867793e+08	4.197632e+08	3.016764e+08
Cross-val. MSE scaled	2.241372e+08	2.680486e+08	2.004765e+08	2.269159e+08	2.437519e+08

Table 5. Feed Forward Neural Network Mean Squared Error results.

F. Stacked Ensemble Model

The stacking method is an ensemble learning technique that combines multiple models to improve overall performance. The intuition behind ensemble learning is based on the concept of 'wisdom of the crowds', which suggests that the decision of a group is more likely to be better than that of a single experienced individual. Stacking utilizes the strength of well-performing models while reducing the impact of those that hinder performance by assigning different weights to the supplied model predictions. The ultimate objective of stacking is to produce better predictions than any single model could achieve on its own.

Our Stacked Ensemble model consists of five distinct machine learning models: the optimal linear model, elastic net model, K-nearest neighbors (KNN) model, regression tree model, and feed-forward neural network model that were trained previously in the project. For both scaled and unscaled data, the best ensemble model utilizes a testing set size of 50%.

	0.1	**0.2**	**0.3**	**0.4**	**0.5**
In-sample MSE unscaled	6.267055e+07	6.102480e+07	6.064674e+07	5.857942e+07	5.470953e+07
Out-sample MSE unscaled	1.026459e+08	9.715614e+07	1.093868e+08	1.124014e+08	1.078997e+08
Cross-val. MSE unscaled	6.294383e+07	6.125483e+07	6.102788e+07	5.906977e+07	5.521241e+07
In-sample MSE scaled	6.093464e+07	5.945427e+07	6.066718e+07	5.953733e+07	5.430537e+07
Out-sample MSE scaled	1.028752e+08	9.628568e+07	9.943442e+07	9.697376e+07	9.978735e+07
Cross-val. MSE scaled	6.123277e+07	5.992963e+07	6.135765e+07	6.012015e+07	5.501462e+07

Table 6. Ensemble Model Mean Squared Error results.

IV. Model Comparison

We can compare models across various dimensions - training and testing sizes, across models, and between scaled and unscaled data - by looking at their in-sample, out-of sample, and cross-validation MSEs. First, we compared the same models on different proportions of the data, ranging from 50% to 90% for training, to get the in-sample MSE and their corresponding testing sets for the out-of-sample MSE. Additionally, we performed a 5-fold Cross-Validation on the model using the training set which calculated the MSE for every fold. These five MSE scores were then averaged to get the cross-validation MSE which is used for model comparison. Finally, we evaluated the performance of our models on both scaled and unscaled data. The full comparison of the MSEs across models, training and testing set sizes, and scaled and unscaled data can be found on the bar charts in the Appendix.

A. Across Training and Testing Size

We can compare the various models learned across training and testing size by holding model type and scale or unscaled constant. To do so, we compared models by taking the minimum Cross-Validation MSE results. Doing so, we can determine that the optimal testing set sizes for each model using scaled and unscaled data are as follows:

	Scaled Data	**Unscaled Data**
Linear	10%	10%
Elastic Net	50%	50%
Regression Tree	20%	20%
KNN	20%	20%
FFNN	30%	30%
Stacked Ensemble	50%	50%

Table 7. Model Comparison – best training set size for every model by scaled and unscaled data.

B. Across Models

We can compare the across models by holding training and testing sizes and scaled or unscaled data constant by looking at the minimum Cross-Validation MSE results. Doing so, we can determine the that the optimal model for each testing size using scaled and unscaled data are as follows:

	Scaled Data	**Unscaled Data**
10%	Ensemble	Ensemble
20%	Ensemble	Ensemble
30%	Ensemble	Ensemble
40%	Ensemble	Ensemble
50%	Ensemble	Ensemble

Table 8. Model Comparison – best model for every testing set size by scaled and unscaled data.

C. Scaled Data vs Non-Scaled Data

Since machine learning models can be sensitive to the scale of the features, we can compare our models by scale or unscaled data holding testing size and model type constant and then observing the minimum MSE of the Cross-Validation results. Doing so, we can determine that the optimal model for each testing size using scaled and unscaled data are as follows:

	10%	**20%**	**30%**	**40%**	**50%**
Linear	Either	Either	Either	Either	Either
Elastic Net	Scaled	Scaled	Scaled	Scaled	Scaled
Regression Tree	Unscaled	Unscaled	Unscaled	Unscaled	Unscaled
KNN	Scaled	Scaled	Scaled	Scaled	Scaled
FFNN	Scaled	Scaled	Scaled	Scaled	Scaled
Stacked Ensemble	Scaled	Scaled	Scaled	Scaled	Scaled

Table 9. Model Comparison – best dataset for every model by every testing set size.

Due to the linear nature of the Linear Regression model, it is important to note that scaling features do not have an impact on its model performance and consequently result in the same MSE scores between scaled and unscaled data.

V. Model Selection and Evaluation

Combining all different ways to analyze and compare model performance, we can evaluate which model performs the best based on having the lowest average MSE score from cross-validation using the training set. The model that performs the best overall is the stacked ensemble model using scaled data and a testing set size of 50%. This model has an out-of-sample MSE score of approximately 99787350 MWh^2. Additionally, using other metrics to evaluate model performance, we can see that the best performing model has a mean absolute error (MAE) of 8082.91 MWh and an R^2 of 48.99%.

	MSE	MAE	R-Squared
In-Sample	5.430537e+07	6257.56	0.685
Out-of-Sample	9.978735e+07	8082.91	0.4899

Table 10. Best Performing Model Results.

As discussed previously, the stacked ensemble model uses a weighted combination of the predictions of the models included in the stacking. Table 11 shows the weights of these models when predicting demand for electricity.

	0.5
Linear Regression scaled	1.166130
Elastic Net scaled	-6.851798
KNN scaled	0.090260
Regression Tree scaled	6.683470
FFNN scaled	-0.083128

Table 11. Best Model – Model Weights for the Ensemble Model.

Analyzing the weights, the stacked ensemble weights the predictions of each model in a separate way to calculate a prediction that has a smaller error than all the models individually. We can also see that the Stacked Ensemble model puts a large weight in the positive direction for the Regression Tree, but nearly equally large negative weight for the Elastic Net model, almost canceling each other out. We can see that the Stacked Ensemble model uses very small weights for the KNN and FFNN model, highlighting that these model's predictions are of little importance to the Stacked Ensemble model. Lastly, we can see that the Stacked Ensemble model uses a nearly perfect weight of the Linear Regression model. This is helpful to identify the importance of the Linear Regression model to the Stacked Ensemble model's overall importance.

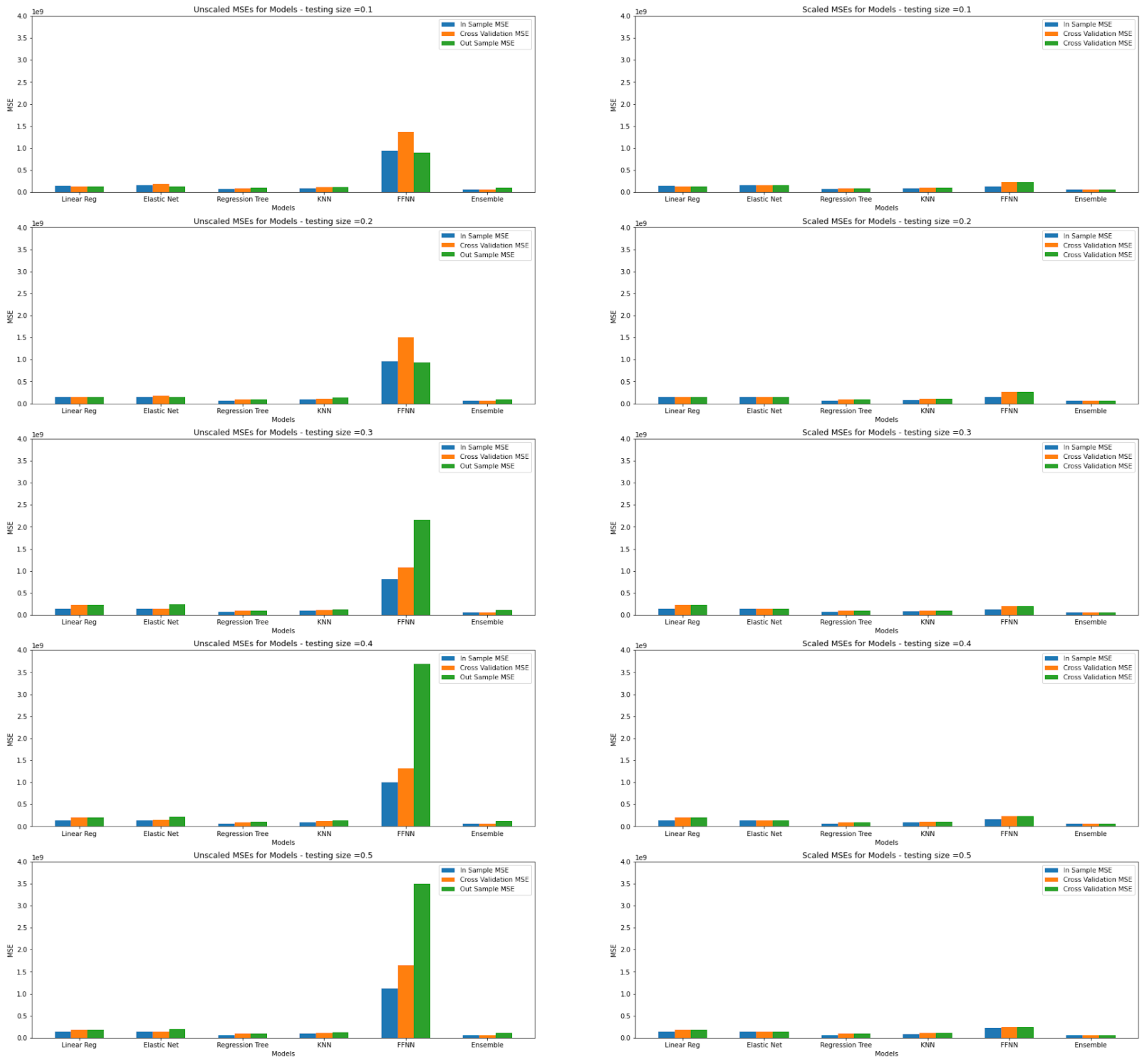
VI. Conclusion

In this study, we explored a variety of machine learning models to predict electricity demand in Victoria, Australia. After preparing the data for the machine learning models, we found that the following features were likely to have an impact on energy demand: recommended retail price, the fraction of the day when demand was traded at a negative RRP, minimum temperature during the day, total daily sunlight energy, daily rainfall, if students were in school on the day and if the day was a state or national holiday. Subsequently, demand for electricity was estimated using a Linear Regression model, an Elastic Net model, a K-nearest neighbors (KNN) model, a Regression Tree model, a Feed-Forward Neural Network model and a Stacked Ensemble model that contained all the previously included models. We then compared the MSEs of all estimated models and found that the Stacked Ensemble model using scaled data and a testing set size of 50% performs best. Given that the model has an R^2 of 48.99%, it can be used to predict almost half of the variation in electricity demand in Victoria, Australia. For future research and to offer even more precise predictions for energy demand to policymakers in the energy sector, it may be more useful to have less aggregated data to estimate energy demand. The use of hourly data could enable machine learning models to detect patterns that are conditional on certain times during the day which could further increase the performance of machine learning models.

References

[1] Kozlov, Alex. (2020). *Daily Electricity Price and Demand Data.* Kaggle. https://www.kaggle.com/datasets/aramacus/electricity-demand-in-victoria-australia?resource=download
[2] Liu, Xiao-Bai. Class Notes and Lecture Slides.
[3] Borghese, Sam. Class Notes and Lecture Slides.

APPENDIX



Appendix 1: Bar chart depicting the Mean Squared Errors of the In-sample, Out-of-Sample, and Cross-Validation of every model using both scaled and unscaled data. Each set of graphs represents a different testing set size.