# Forecasting Electricity Consumption Using Supervised Learning in the UK

### Team: Manjiri Girish Satam, Miriam Runde, Jackson Luckey, Aditya Narayan Rai

### Machine Learning Group Project - MSc Data Science for Public Policy

```
In [2]: from IPython.display import Image, display
```

## Introduction

Electicity grids need to consistently match electricity production and consumption to prevent variations in voltage (brownouts) and grid failure (blackouts) becuase they lack storage capacity, making accurate forecasting of consumption essential for electrical grid consumptions. Also, accurate forecasting is challenging due to the high volatility and variability in electricity consumption patterns at the household level. To address these challenges, our study focuses on forecasting electricity consumption in the UK at the household level using data from the electricity smart meters. We focus on forecasting electricity consumption from 18:00 to 18:30 in the UK and we have choosen this specific time window due to its distinct annual variation: it is light out at 18:00 in the summer and dark out in the winter, providing a useful case study for capturing seasonal patterns in electricity usage. We also choose this time slot for the efficiency in running the models since our raw data comes as a large CSV file with one row per 30 minute period per household i.e., 48 half-hour timepoints which was making it difficult for us to run the models and analysis.

A key aspect of modern electricity demand forecasting is the shift from deterministic to probabilistic forecasting (Taieb et al., 2017). Probabilistic forecasting provides a full distribution of potential future values rather than a single point estimate, offering a more comprehensive assessment of uncertainty. This is particularly important for grid operators who need to plan for various scenarios to ensure stability and reliability. By incorporating probabilistic forecasts, grid operators can better manage the inherent uncertainties and variabilities in electricity consumption, leading to more informed decision-making and improved grid stability. However, for this specific paper we have only relied on the point estimate and our aim for the next cycle for this project will be to move to the probabilistic forecasting.

Our approach aims to aid electrical grid operators in forecasting future consumption for relatively new households. Given the limited historical data for these households, we excluded the household ID from the model and designed training datasets that only require a few months of previous electricity consumption data. This method leverages the rich probabilistic information to enhance forecast accuracy and reliability, ultimately supporting better grid management practices.

The next section of this paper discusses in detail about the data cleaning and analysis strategy adopted during the course of the project followed by exploratory data analysis discussion and conclusion.

## Data - Cleaning and Analysis Strategy

The raw data comes as a large CSV file with one row per 30 minute period per household containing electricity usage and one small CSV file with one row per household containing demographic and geographic information. We had many difficulties working with the larger dataset. It is too large to load into memory on any of our devices and has erroneous datapoints (e.g. malformed dates and duplicated rows). Furthermore, it is not sorted, so it is challenging to aggregate by either time or household. We eventually decided to focus on 18:00-18:30 so that we could easily subset the dataset. To do so, we loaded the dataset in chunks and saved all the rows covering electricity usage from 18:00 to 18:30 in a new CSV file. We then read that CSV file into Pandas and constructed our daily and monthly datasets.

Our features are previous consumption (lagged and summary statistics), the date (day, day of the week, month, and year), which part of the UK the household is in, and household demographics. We decided against calculating more complicated features because our models took an extremely long time to run even with the simplest datasets. From experimentation (looking at the feature importance when running prototype models on a fraction of the dataset), including the date and the lagged columns was the most important.

Our daily dataset has one row per household per day. Each row has electricity use that day, the previous 90 days of electricity use (1 column per day), summary statistics of energy use over the last week, month, and year, and dummy variables for the demographic and geographic variables. We dropped the first 90 days of use per household because we did not want missingness in the lagged columns. The label is that day's electricity consumption from 18:00 to 18:30. These datasets are large (~2GB for training).

Our monthly dataset has one row per household per month. Each row has 28 lagged columns covering electricity use over the previous 4 weeks and dummy variables for the demographic and geographic variables. The label is average electricity consumption from 18:00 to 18:30 over the next month. These datasets are small (~30MB for training).

We decided to split both datasets into training, validation, and test. All utilization data from 2010 or later is in test. Data from 2008 or 2009 are in training (70%) or validation (30%). We included a validation dataset in case we wanted to run neural network models (and it reduced the size of our training dataset, making it faster to run models).

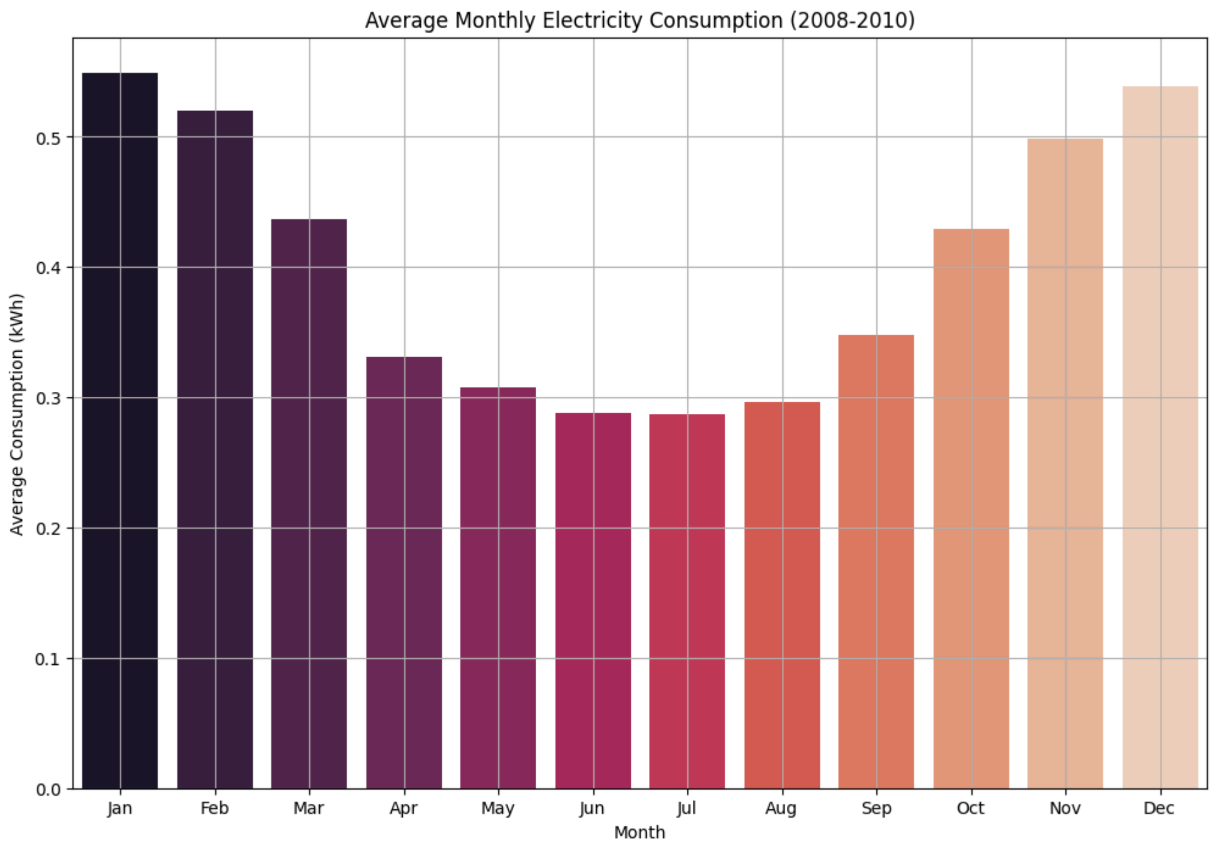
Our evaluation startegy is similar for both the monthly and daily datasets: we calculate the mean squared error for electricity usage from 18:00 to 18:30 for every household in 2010 or later. For the monthly dataset, we forecast the average electricity consumption from 18:00 to 18:30 across the month and then calculate the MSE for the monthly average. For the daily dataset, we forecast electricity consumption from 18:00 to 18:30 for every household-day in 2010 or later. To compare daily and monthly models, we have two options. We can either take the monthly model, predict mean usage per month from 18:00 to 18:30, and then calculate the mean squared error for every day in that month. Alternatively, we can take the daily model, predict daily usage from 18:00 to 18:30, aggregate predicted usage to the monthly model, and then calculate the mean squared error. In either case we can compare the models trained on the daily and monthly datasets

We do not have significant fairness or explainability concerns with this ML task. While we are forecasting at the household level, our forecast does not directly affect the household. It would instead be aggregated by the electrical grid operator to determine how much electricity to produce for the entire UK. Explainability is not a concern because the utility can treat it as a black box that helps optimize their operations. The utility does not need to explain the model to stakeholders or the public. Therefore we can choose the best model by picking the best parameters for each model class and then choosing the model class that has the smallest mean squared error on the test dataset.

# Exploratory Data Analysis

## 1) Average Monthly Energy Consumption

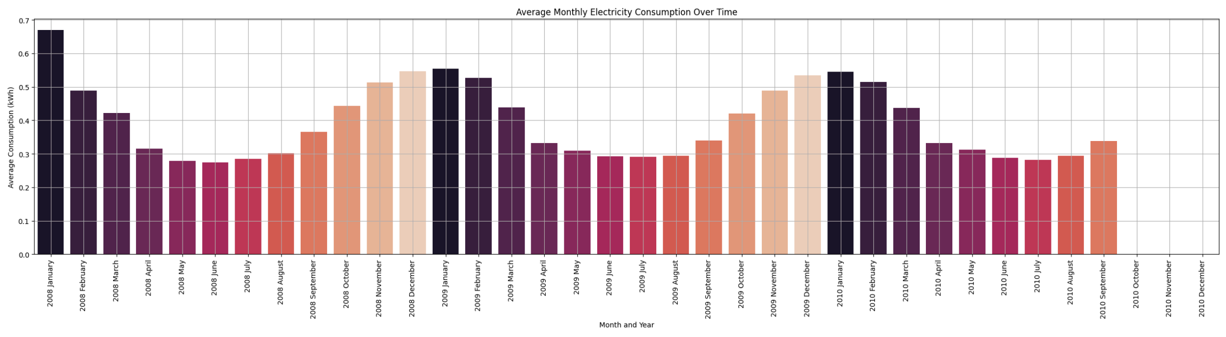
In [3]:
```
img_path = '/Users/miriam/Documents/GitHub/ml-power/output_year.png'
display(Image(filename=img_path))
```



The bar chart "Average Monthly Electricity Consumption" depicts the average electricity consumption in kilowatt-hours (kWh) at 18:00 hrs across the twelve months of the year, based on data from 2008 to 2010. The x-axis represents the months from January to December, while the y-axis indicates the average consumption in kWh. The highest consumption occurs in January and December, exceeding 0.5 kWh, reflecting increased heating needs in winter evenings. The lowest consumption is in July, slightly above 0.3 kWh, indicating reduced energy usage during summer evenings.
A decline in consumption is observed from March to July, followed by a gradual increase from September to November, peaking again in December. This chart illustrates a clear seasonal trend with higher electricity usage at 18:00 hrs in winter and lower usage in summer.

## 2) Average Monthly Electricity Consumption Over Time

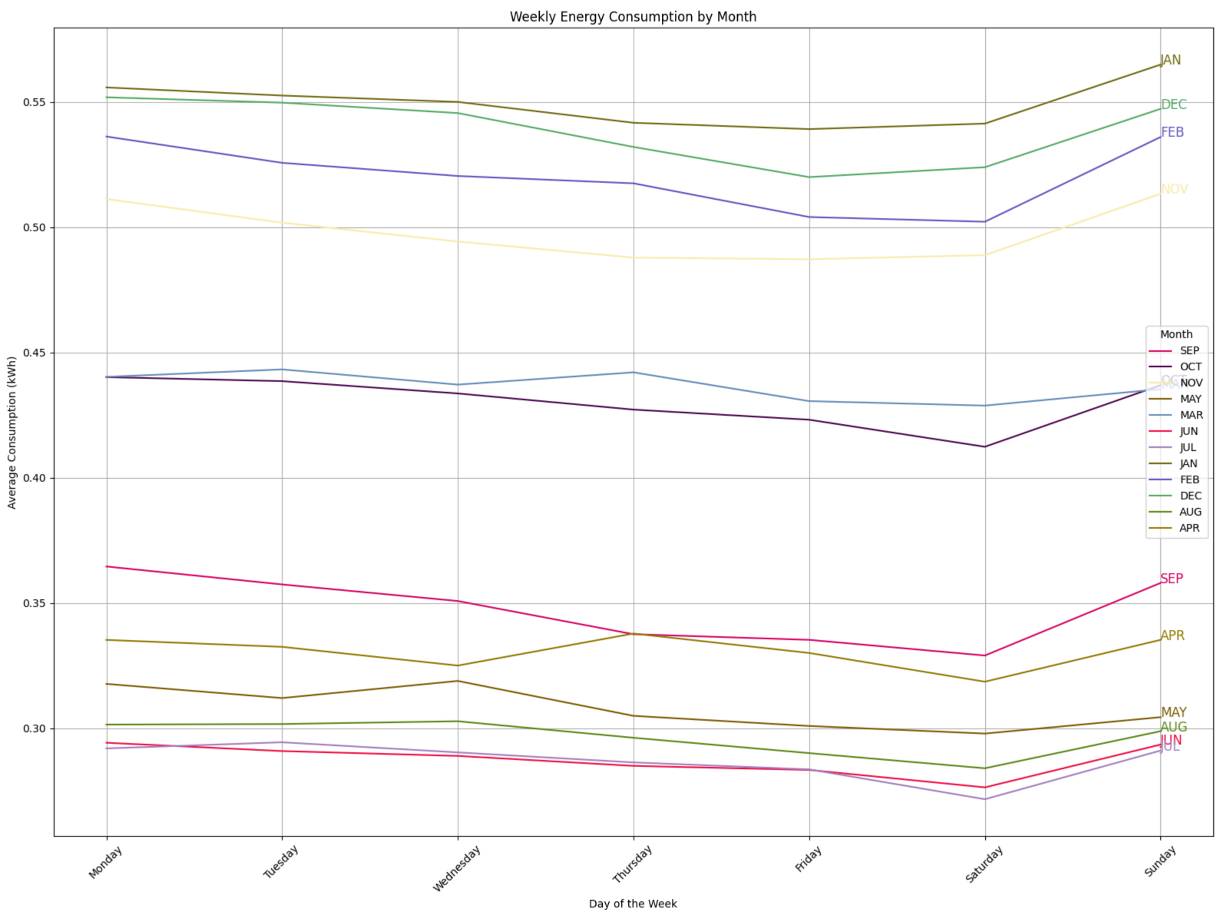
In [6]:
```
img_path = '/Users/miriam/Documents/GitHub/ml-power/output3.png'
display(Image(filename=img_path))
```



The bar chart shows the average electricity consumption in kilowatt-hours (kWh) at 18:00 hrs from January 2008 to December 2010. In line with the previous plot, it highlights a consistent seasonal trend with higher consumption in the winter months (January, February, November, December) due to increased heating needs, and lower consumption in the summer months (June, July, August) reflecting reduced energy usage. Yearly comparisons reveal similar seasonal patterns, though January 2008 has the highest peak in consumption, with slight variations in peak and trough values in subsequent years.
This underscores the significant impact of seasonal changes on electricity consumption over the three-year period.

## 3) Average 6pm Electricity Consumption per Month

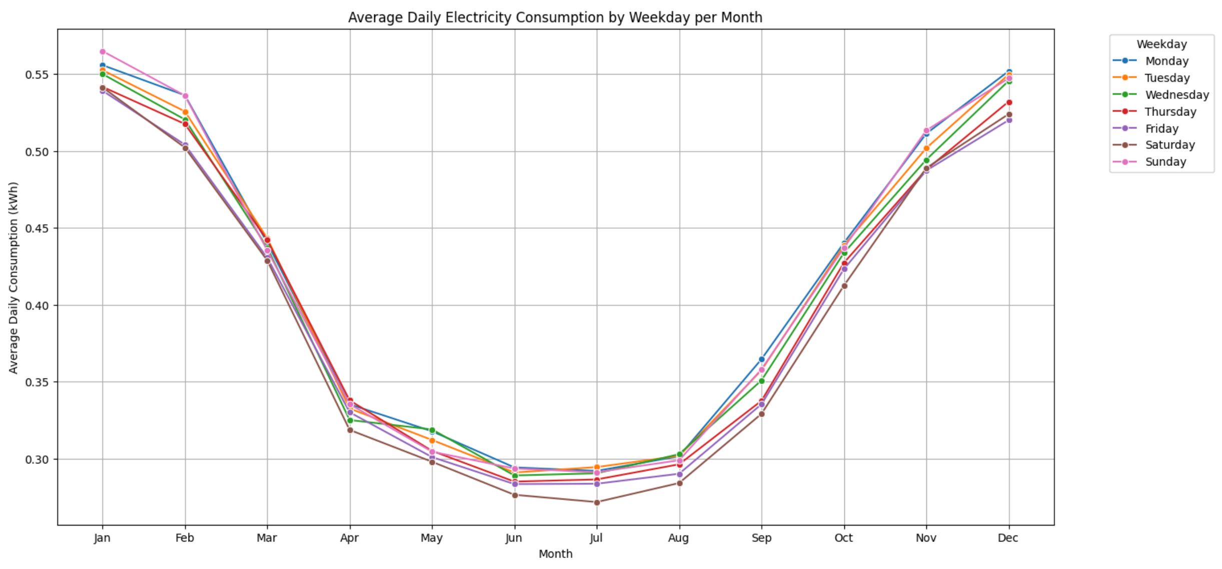
In [8]:
```
img_path = '/Users/miriam/Documents/GitHub/ml-power/output_av_month_year.png'
display(Image(filename=img_path))
```



The line chart titled illustrates the average daily electricity consumption in kilowatt-hours (kWh) at 18:00 hrs for each day of the week, with different lines representing various months. The x-axis represents the days of the week from Monday to Sunday, while the y-axis indicates the average consumption in kWh.
Winter months (January, February, December) show the highest consumption, starting above 0.55 kWh on Monday and gradually declining through the week, with a slight rise on Sunday. Spring and autumn months (March, April, May, September, October, November) exhibit moderate consumption levels, starting around 0.45 kWh and also declining through the week with a small increase on Sunday. Summer months (June, July, August) have the lowest consumption, ranging from 0.30 to 0.35 kWh, remaining stable through the week with a slight dip towards the weekend and a small rise on Sunday.
The chart shows a consistent pattern of mid-week dips (Wednesday to Friday/Saturday) and increased consumption on weekends, especially Sunday, with higher overall usage in winter due to heating needs and lower usage in summer.

## 4) Average 6pm Electricity Consumption by Weekday per Month"

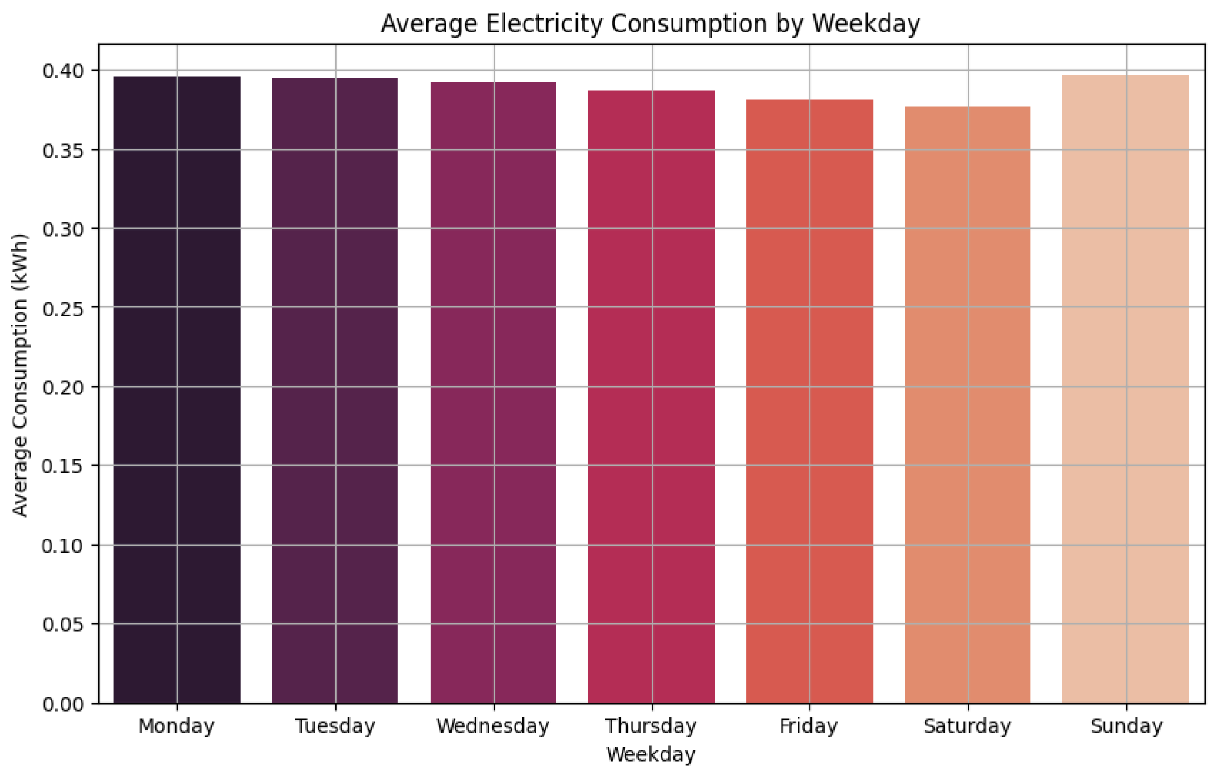
In [9]:
```
img_path = '/Users/miriam/Documents/GitHub/ml-power/output_year_days.png'
display(Image(filename=img_path))
```



The line chart shows the average daily electricity consumption in kilowatt-hours (kWh) at 6 pm from 2008 to 2010 for each day of the week, with different lines representing various months. The x-axis represents the months from January to December, while the y-axis indicates the average daily consumption in kWh. The chart reveals minimal variation in consumption across the days of the week, with a strong seasonal pattern being the primary driver of changes in electricity usage.
Winter months (January, February, December) show the highest average consumption, starting above 0.55 kWh and gradually declining through the week, with a slight rise on Sunday. Spring and autumn months (March, April, May, September, October, November) exhibit moderate consumption levels, starting around 0.45 kWh and showing a gradual decline through the week with a small increase on Sunday. Summer months (June, July, August) have the lowest consumption, ranging from 0.30 to 0.35 kWh, remaining stable throughout the week with a slight dip towards the weekend and a small rise on Sunday. This consistent pattern across all weekdays indicates that daily electricity usage at 6 pm is primarily influenced by seasonal factors rather than the specific day of the week.

## 5) Average Electricity Consumption by Weekday

In [10]:
```
img_path = '/Users/miriam/Documents/GitHub/ml-power/output_week.png'
display(Image(filename=img_path))
```



The last bar chart shows the mean electricity usage at 18:00 hrs from 2008 to 2010. The x-axis represents the days of the week, while the y-axis indicates the average consumption in kilowatt-hours (kWh). The data reveals a slight decrease in electricity consumption over the course of the week, with the lowest usage occurring on Saturday, around 0.35 kWh. This is followed by an increase on Sunday, reaching just below 0.40 kWh.

This pattern can be contextualized with the typical routines of people at 6 PM. During the weekdays (Monday to Friday), people generally return home from work or school around this time, leading to relatively stable electricity usage as they engage in evening activities such as cooking dinner, using appliances, and entertainment. By Saturday at 6 PM, many people might be out for recreational activities, shopping, or dining out, resulting in a slight dip in electricity usage. On Sunday at 6 PM, people are more likely to be at home preparing for the upcoming week, which can lead to higher electricity consumption as they engage in activities like cooking, laundry, and watching television.

**Overall Interpretation:**

The analysis of electricity consumption at 18:00 hrs from 2008 to 2010 reveals that seasonal changes have the most significant impact on usage patterns, with higher consumption in winter due to heating needs and lower consumption in summer when less heating is required. The weekly routine also influences these patterns, though to a lesser extent. On weekdays, consistent evening activities such as cooking and entertainment result in stable usage. On Saturdays, lower consumption suggests people are often out, while higher usage on Sundays indicates increased home activities as people prepare for the upcoming week.

# Regression Models: Linear, Ridge, & Lasso

The results of predicting average electricity consumption per month from linear regression, ridge regression, and lasso regression using the same dataset show quite similar performance metrics.

**Linear Regression** achieved a training MSE of 0.0144, a validation MSE of 0.0151, and a test MSE of 0.0179. This indicates that the model fits the training data well and generalizes effectively to the validation set, with only a slight increase in error for the test set. The test $R^2$ score of 0.7903 suggests that the model explains about 79% of the variance in the data.

| Parameter | Value |
|---|---|
| Training MSE | 0.01436349042532913 |
| Validation MSE | 0.015068152127000613 |
| Test MSE | 0.017929546215365546 |
| Test set R2 Score | 0.7903490068810117 |

**Ridge Regression**, which uses L2 regularization to prevent overfitting, found its best alpha to be 35.56. With this alpha, it achieved a validation MSE of 0.0151 and test MSE of 0.0179, and a slightly higher test $R^2$ score of 0.7904. This indicates that ridge regression offers a minor improvement over linear regression by balancing the model complexity and reducing the risk of overfitting. The regularization helps to shrink the coefficients, which can prevent the model from becoming too complex and capturing noise in the training data.

| Parameter | Value |
|---|---|
| Best Alpha | 35.564803062231285 |
| Training MSE | 0.014365506254449751 |
| Validation MSE | 0.015063685063261882 |
| Test MSE | 0.01792498486025651 |
| Test set R2 Score | 0.7904023430121714 |

**Lasso regression**, which uses L1 regularization to both prevent overfitting and perform feature selection, found its optimal alpha to be 0.001. This resulted in a validation MSE of 0.0151, a test MSE of 0.0179, and a test $R^2$ score of 0.7907. The very small alpha suggests that the regularization effect was minimal, making the lasso regression model behave similarly to the linear regression model. In this case, the regularization was not strong enough to drive any coefficients to zero, so it didn't perform significant feature selection.

| Parameter | Value |
|---|---|
| Best Alpha | 0.0001 |
| Training MSE | 0.014373598612734517 |
| Validation MSE | 0.015072480045796094 |
| Test MSE | 0.017902566672722667 |
| Test set R2 Score | 0.7906644798908153 |

**Explanation of Alpha Values:**

For both ridge and lasso regression, we used the same range of alpha values (np.logspace(-4,4,50)). This function generates 50 values for alpha ranging from $10^{-4}$ to $10^{4}$. Alpha controls the strength of regularization: a small aplha means less regularization, while a large alpha means more regularization.

In ridge regression, the best alpha was found to be 35.56, which indicates a moderate level of regularization. This helps in preventing overfitting by shrinking the coefficients but not driving them to zero. This balance helps the model generalize slightly better than linear regression.

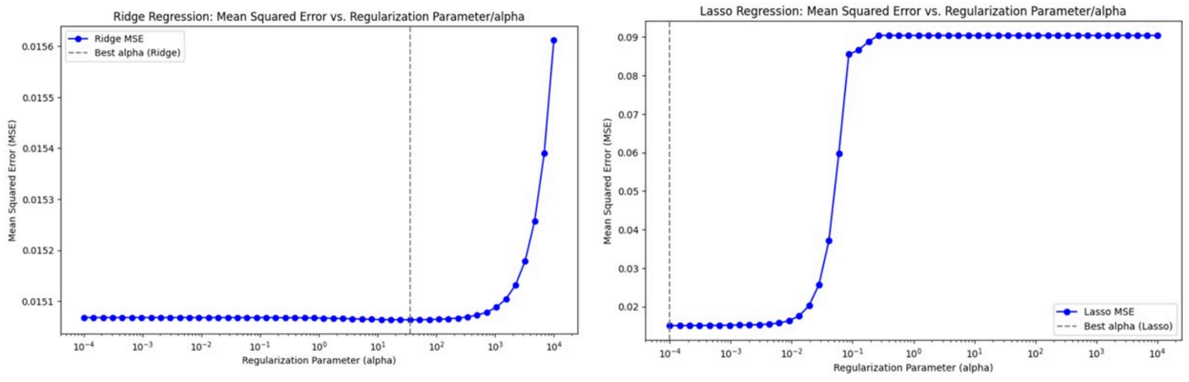
In lasso regression, the best alpha was 0.0001, which is very small. This means the regularization effect was minimal, so it didn't significantly change the coefficients from those in the linear regression model. The reason why ridge and lasso have such different optimal alphas is due to the nature of the regularization: ridge shrinks coefficients uniformly, while lasso can shrink some coefficients to zero.

In summary, all three models - linear regression, ridge regression, and lasso regression - perform similarly on this dataset, with only negligible differences in their MSE and $R^2$ scores. Ridge regression shows a slight edge in performance, suggesting it might generalize a bit better due to its moderate regularization. Lasso regression's very small alpha indicates that feature selection was not particularly important for this dataset, leading it to perform almost identically to linear regression.

Given these results, one could choose any of the three models depending on the specific needs. If a simple model is preferred with no regularization, linear regression works well. In terms of interpretability, it is the easiest to understand as it has no regularization. If one wants to add some regularization to prevent overfitting, ridge regression is slightly better. It adds a layer of complexity with its regularization, but it doesn't zero out coefficients, so all predictors are still present in the model. If one is interested in feature selection and sparsity in the model, lasso regression might be the choice. It can improve interpretability by reducing the number of predictors, but in this case, it didn't significantly differ from linear regression.

Also, computational efficiency is generally higher for linear regression because it doesn't involve any additional computations for regularization. Ridge and lasso regression might take slightly longer due to the optimization of the alpha parameter, but this difference is often negligible. Overall, while ridge regression could be preferred for its balance of regularization and predictive accuracy, the minimal performance differences suggest that any of the three models could be effectively used depending on specific requirements such as interpretability or computational efficiency.

```
In [4]: display(Image(filename='../figures/Combined.jpg'))
```



# XGBoost and Random Forest Models - Monthly and Daily

## Monthly XGBoost

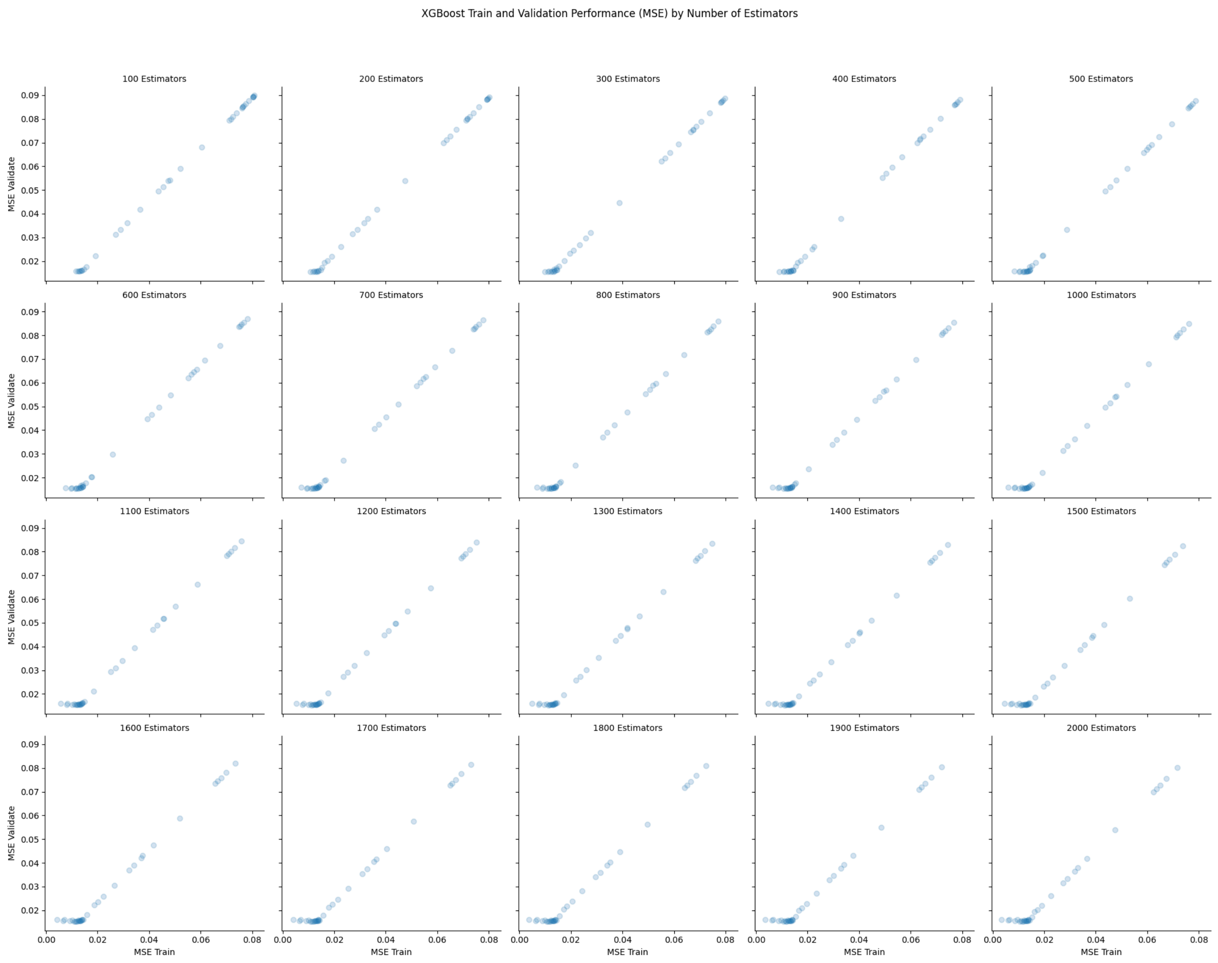
For XGBoost, we decided to try combinations of:

- 100 to 2000 estimators in increments of 100 (20 possibilities)
- Learning rates of [0.0001, .0005, .001, .005, .01, .05, .1] (7 possibilities)
- Depths of 1 to 5 (5 possibilities)

for a total of 700 possible models. For each model we calculated both the training and validation MSEs. We used the validation dataset because we already had it in case we got to neural nets and because cross-validation was taking too long to run. To identify the best model, I reran the top-performing models (as measured by validation MSE) with cross-validation and used the one standard error rule.

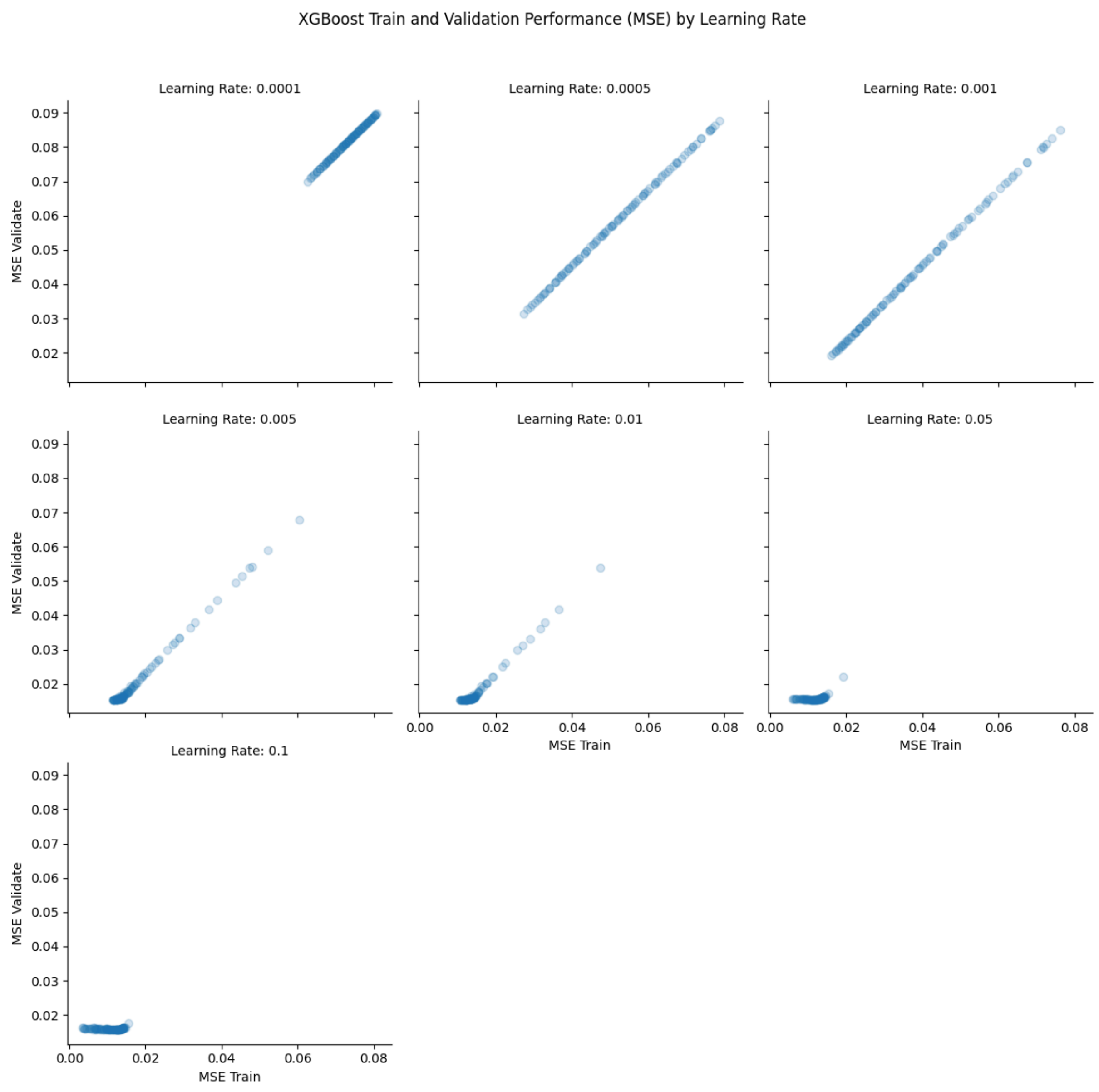
A consistent theme with this dataset is model performance looking like a hockey stick. By that, I mean that there is almost a perfectly linear relationship between the training and validation mean squared error except for models with extremely low error, where training error starts to go down while validation error stays the same. My prior is that this occurs because our data is quite predictable in a simple way (current electricity usage is highly correlated with past electricity usage). I believe the feature importance charts later in the report provide evidence that this is occuring. My interpretation of this is that we want to choose models at the inflection point when training and validation MSE diverges.

```
In [5]: display(Image(filename='../data/model_results/monthly_xgboost_by_n_estimators.png'))
```



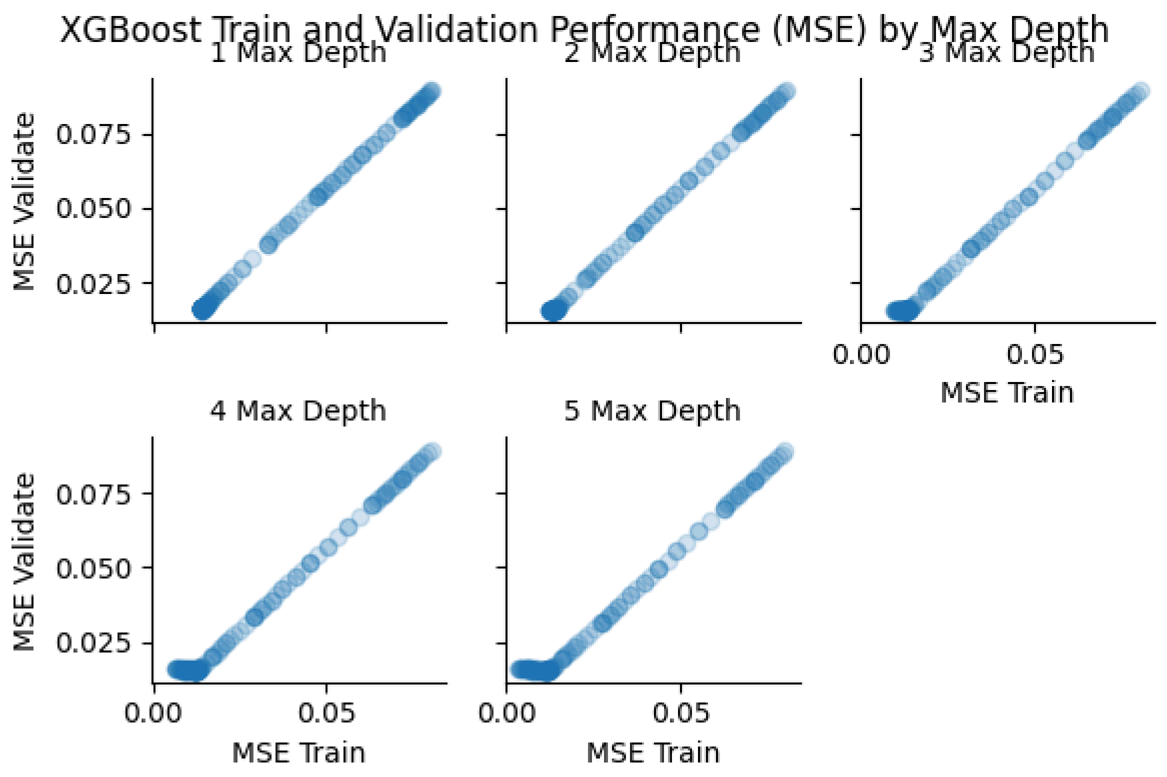
The chart above shows that there is not a clear relationship between the number of estimators and model performance. Even with only 100 estimators, the hockey stick phenomena occurs (albeit only a little bit).

```
In [6]: display(Image(filename='../data/model_results/monthly_xgboost_by_learning_rate.png'
```



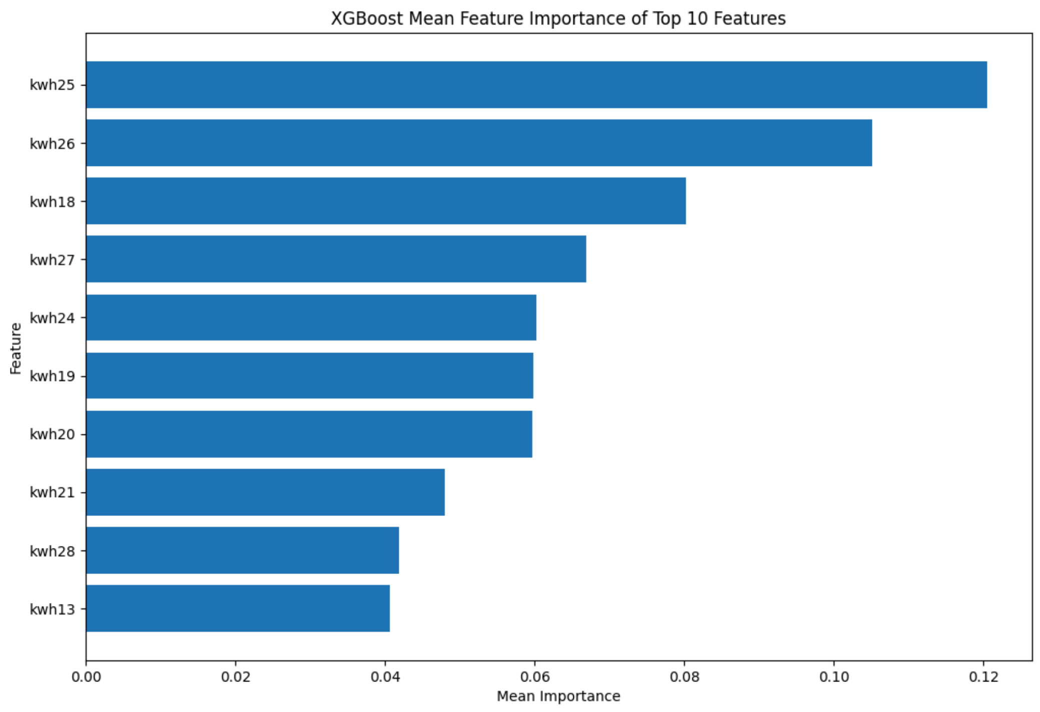
This chart shows that model performance is highly dependent on the learning rate. With the lowest 3 learning rates (0.0001, 0.0005, 0.001), the model never reaches the smallest MSEs that other models reach. With the largest two learning rates (0.05, 0.1), the model always overfit the training data and have approximately the same validation MSE. Learning rates of 0.005 and 0.01 seem to provide the best performance.

In [7]:
```
display(Image(filename='../data/model_results/monthly_xgboost_by_max_depth.png'))
```



This chart shows that model performance is partially correlated with the max depth of the XGBoost model. The hockey stick phenomena only occurs with 3 or more levels (and really only with 4 or 5). Models with only 1 or 2 layers do not reach quite the same level of performance on the validation dataset. Based on this chart, I suspect models with 3 layers will perform the best.

```
In [8]: display(Image(filename='../data/model_results/monthly_xgboost_feature_importance.pn
```



Finally, we'll look at feature importance. To create this chart, I aggregated feature importance across all 700 models. The top 28 features are the lagged electricity usage columns. They are responsible for almost the entirety of the model's predictions (over 99%). While I was originally suprised that the month dummy columns had 0 importance and that the demographic and geographic dummy columns had near zero importance, I eventually realized that the information contained in those columns was also captured by the lagged electricity usage columns. In other words, the explanatory power of the month drops out when you include electricity usage in the previous month. I am not sure why these particular lagged columns are at the top. The order of the feature importances of the lagged electricity columns is highly dependent on the specific model.

After running cross-validation on the 20 models with the lowest validation error, this model performed the best.

| Parameter | Value |
|---|---|
| n_estimators | 1900 |
| learning_rate | 0.05 |
| max_depth | 3 |
| mse_train | 0.009979 |
| mse_validate | 0.014112 |
| mse_test | 0.013055 |

## Daily XGBoost

We only used XGBoost on the daily dataset because it runs much faster than random forest and we were highly limited by compute time (we discovered Google Colab late in the project and running a single model on the daily dataset in Colab eats up a significant portion of the monthly compute budget). For similar reasons, we were unable to do an extensive parameter search or do cross-validation. I instead chose to run a few models to compare them to our monthly models so that our time spent working on the more sophisticated dataset was not wasted. I chose model parameters based on how the monthly models performed. I tried models with 1000 estimators with a max depth of 3, 5, and 7 and a learning rate of 0.05 and 0.1. I chose higher learning rates to make the model converge faster due to compute limits.

Our best performing daily model had the following parameters:

- Number of Estimators: 1000
- Learning Rate: 0.05
- Max Depth 3

It had the following daily MSEs:

- Train Mean Squared Error: 0.09854756999257833
- Validate Mean Squared Error: 0.09663890082236168
- Test Mean Squared Error: 0.09276999969447028

The ten most important features in this model are:

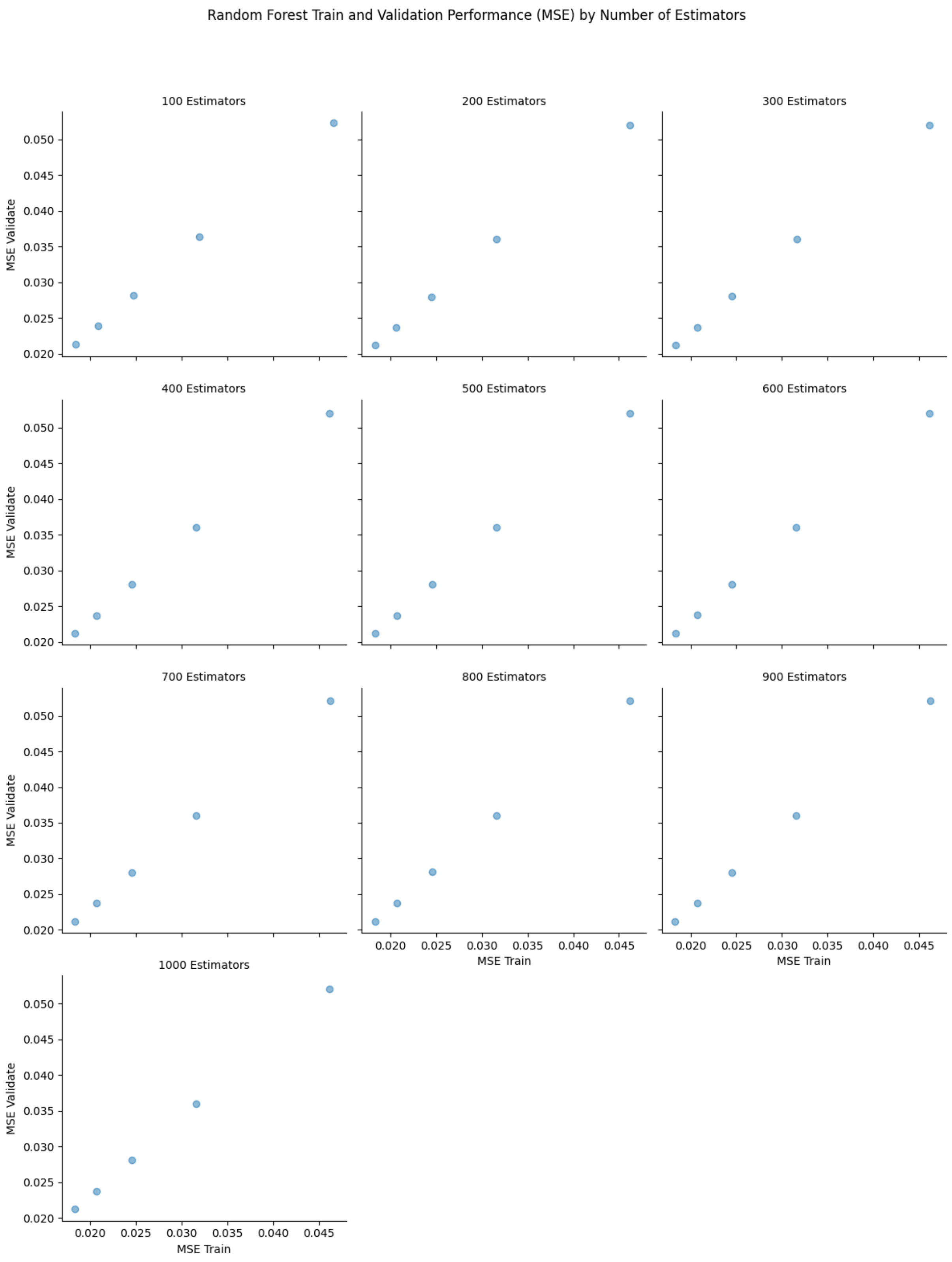
| Feature | Feature Importance |
|---|---|
| previous_month_mean | 0.513370 |
| previous_week_mean | 0.238838 |
| previous_week_median | 0.070862 |
| previous_week_min | 0.016290 |
| kwh_lag_14 | 0.012096 |
| kwh_lag_1 | 0.011610 |
| previous_year_mean | 0.011018 |
| kwh_lag_21 | 0.005179 |
| previous_month_median | 0.004988 |
| kwh_lag_7 | 0.004919 |

Like with the other models, the previous use columns are by far the most important and make up almost all of the explanatory power of the model. The previous use aggregate columns are the most important. Interestingly, the median columns are included alongside the mean columns for month and week but not year.

After predicting daily usage and aggregating that usage to the monthly level, the test MSE (that is comparable to the monthly models) is 0.021048

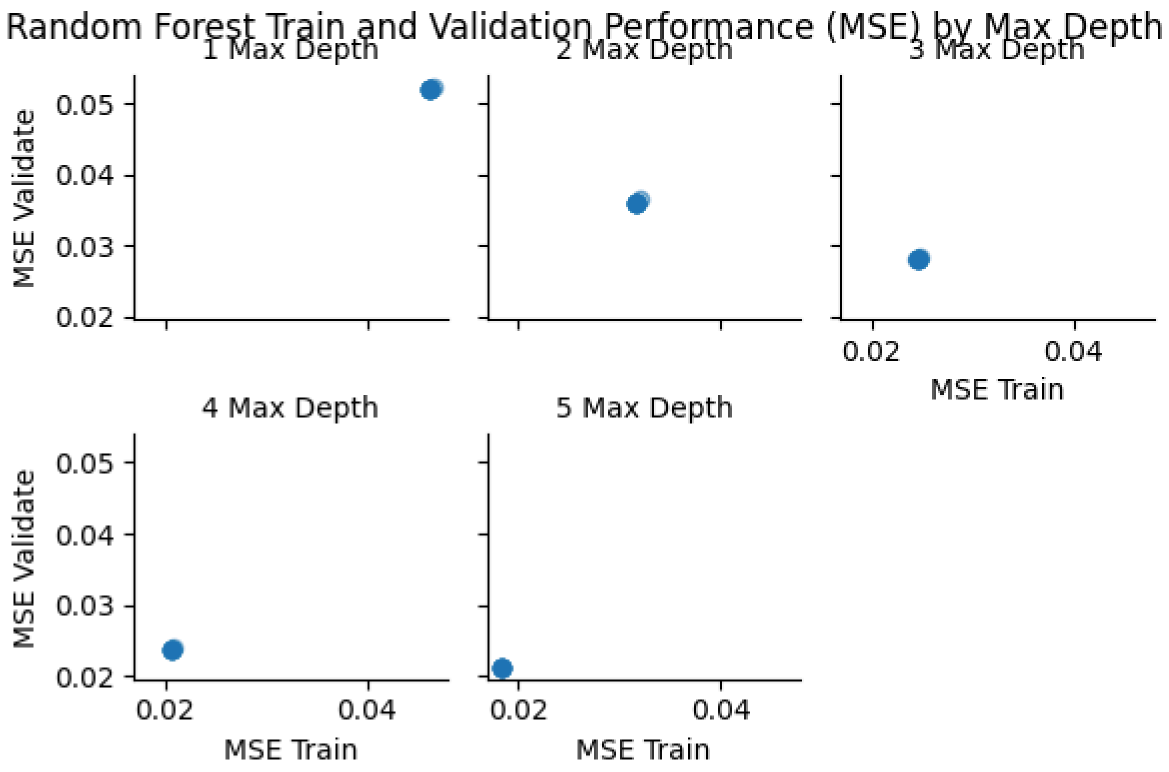
## Monthly Random Forest

```
In [10]: display(Image(filename='../data/model_results/monthly_random_forest_by_n_estimators
```



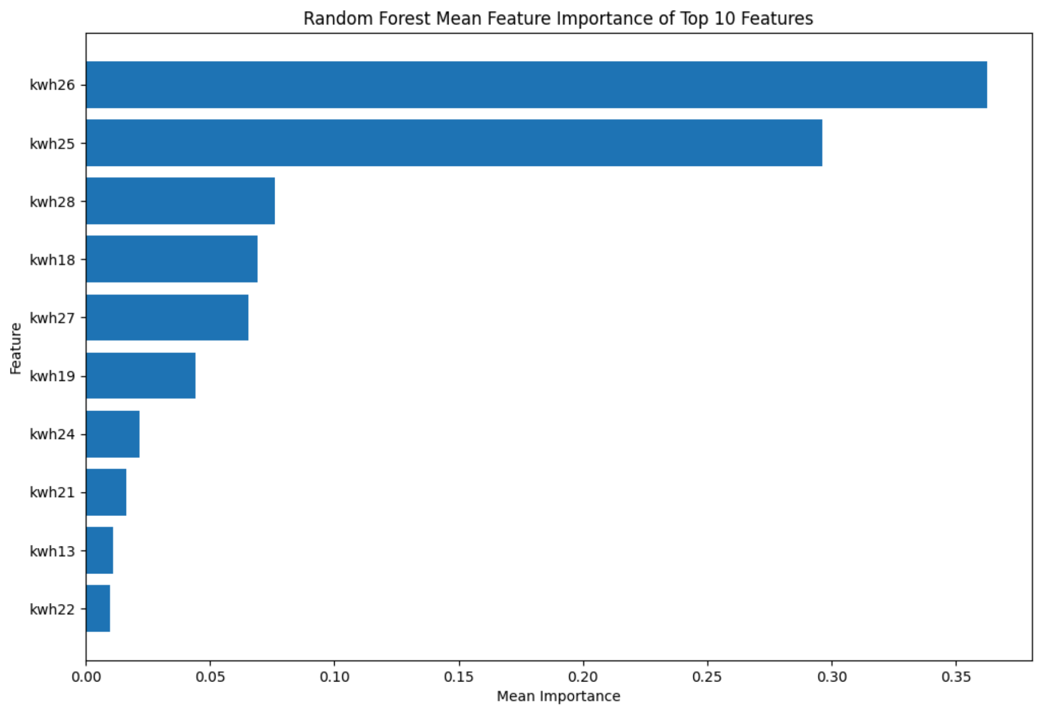
As with XGBoost, the number of estimators does not seem to be correlated with random forest model performance.

In [11]:
```
display(Image(filename='../data/model_results/monthly_random_forest_by_max_depth.pn
```



Unlike XGBoost, model performance is highly correlated with max depth with the random forest models. The models with a max depth of 5 performed the best. If we had more compute, I would have tried running models with an even greater depth. I suspect this is because a relatively small number of features provide most of the explanatory power, so being able to have more complicated splits on those features is important.

In [12]:
```
display(Image(filename='../data/model_results/monthly_random_forest_feature_importa
```



As with XGBoost, the most important features are lagged electricity consumption. The top two features are 26 and 25 days before the month we are predicting like with XGBoost, although the days are reversed here. I am suprised by the amount of overlap in feature importance between the two model classes. I am still curious why it is these specific days that are so important and am not able to think of a clear way to investigate this.

After running cross-validation on the 3 models with the best validation MSE, the top Random Forest model was:

| Parameter | Value |
|---|---|
| n_estimators | 400 |
| max_depth | 5 |
| mse_train | 0.018319 |
| mse_validate | 0.021182 |
| mse_test | 0.025573 |

## Best Model

As explained in the cleaning and analysis section, we do not have significant explainability or fairness concerns with this ML task. Therefore, we can select our model almost purely off of performance (ie test mean squared error). Our best performing model is XGBoost as it has a significantly lower test MSE (0.013055) than Random Forest (0.025573) or linear regression with or without regularization (0.01792--it was remarkably consistent across OLS, LASSO, and Ridge). While the linear regression models will be able to make forecasts faster than the top performing XGBoost model due to having lower parameters, our prior belief is that the increased model performance outweighs the increased computation cost due to the large difference in test MSE (~30% relative to XGBoost), especially since the model is forecasting an entire 30 minute chunk (the cost-performance tradeoff might be different if we were forecasting consumption 10 seconds into the future).

The high performance of the daily XGBoost model when applied to the monthly forecasting task despite trying approximately 1% as many models and not doing cross-validation suggests that it is worth further investigation. With the current models, however, the daily XGBoost model is outperformed by both the monthly XGBoost model and linear regression.