



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich



Institut für  
Technische Informatik und  
Kommunikationsnetze

# Prediction Models for Indoor Solar Energy Harvesting

Semester Thesis

Colin Berner

bernerc@ethz.ch

Computer Engineering and Networks Laboratory  
Department of Information Technology and Electrical Engineering  
ETH Zürich

## **Supervisors:**

Dr. Rehan Ahmed

Stefan Drašković

Lukas Sigrist

Prof. Dr. Lothar Thiele

January 31, 2019

# Acknowledgements

I would like to thank my supervisors Rehan Ahmed, Stefan Drašković and Lukas Sigrist for their support throughout this thesis. In our weekly meetings they gave me valuable advice and guided me in the right directions.

I also thank Professor Lothar Thiele and the Computer Engineering and Networks Laboratory for giving me the opportunity to work on this semester thesis.

# Abstract

Energy harvesting systems have gained in popularity over the recent years. For these systems, in order to efficiently schedule tasks for an application, it is important that we can predict the harvested energy in the near future. While this is a well-studied topic for outdoor solar power forecasting, the situation is different indoors, where the light consists of a mixture of artificial light and sunlight.

In this thesis, we develop and evaluate models to predict the power of an indoor solar energy harvesting system. In particular, we look at linear regression models, multivariate adaptive regression splines, decision trees and random forest models.

As an evaluation, we analyze model performances when changing various model parameters. This includes limiting the number of input variables or varying the prediction interval. We also estimate the runtime complexity and memory footprint of the different models.

# Contents

<b>Acknowledgements</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Preliminaries</b>	<b>3</b>
2.1 Stations . . . . .	3
2.2 Available Measurements . . . . .	3
2.3 Variable Selection . . . . .	5
2.4 Models . . . . .	6
2.4.1 Linear Regression . . . . .	6
2.4.2 Multivariate Adaptive Regression Splines . . . . .	6
2.4.3 Decision Tree . . . . .	7
2.4.4 Random Forest . . . . .	8
<b>3 Prediction</b>	<b>9</b>
3.1 Accuracy . . . . .	10
3.1.1 Performance Metrics . . . . .	10
3.1.2 Model Configuration . . . . .	12
3.2 Input Variables . . . . .	12
3.2.1 Recursive Feature Elimination . . . . .	12
3.2.2 Importance of Variables . . . . .	13
3.2.3 Variable Classes . . . . .	13
3.3 Prediction Interval . . . . .	14
3.4 Global Model . . . . .	14
3.5 Model complexity . . . . .	15
3.6 Memory Footprint of Runtime Variables . . . . .	16



<b>4</b>	<b>Evaluation</b>	<b>17</b>
4.1	Accuracy . . . . .	17
4.1.1	Performance Metrics . . . . .	17
4.1.2	Errors over the day . . . . .	18
4.2	Input Variables . . . . .	18
4.2.1	Results of the Recursive Feature Elimination . . . . .	18
4.2.2	Importance of Variables . . . . .	21
4.2.3	Variable Classes . . . . .	24
4.3	Prediction Interval . . . . .	25
4.4	Global Model . . . . .	26
4.4.1	Effects of Sunlight . . . . .	27
4.5	Model Complexity . . . . .	27
4.6	Memory Footprint of Runtime Variables . . . . .	29
<b>5</b>	<b>Conclusion</b>	<b>31</b>
5.1	Future Work . . . . .	31
	<b>Bibliography</b>	<b>32</b>
<b>A</b>	<b>Additional Plots</b>	<b>1</b>

# Introduction

---

Energy harvesting systems have become increasingly more popular over the recent years. These devices harvest energy from the environment to prolong the lifetime of a battery or in the case of batteryless systems to directly power an application. An example could be a wireless sensor node powered by a small solar panel.

For these systems, in order to schedule tasks for an application, it is desirable that we can predict the harvested energy in the near future. This allows implementing more efficient schedules and ultimately smaller batteries. This prediction can be a difficult task, since the availability of environmental energy is highly variable. If we have a schedule very dependent on the predictions and predict too little energy, the schedule will be inefficient, but if the estimate is too high, the device could run out of power entirely.

In this thesis, we develop prediction models for indoor solar energy harvesting. For outdoor solar power forecasting, this is a well-studied topic, also with power grid balance in mind [2, 10]. However, indoors, the situation is an entirely different one. Here, the light consists of a mixture of artificial light, indirect sunlight and sometimes direct sunlight. The artificial light is mostly dependent on human presence, as lights are turned on depending whether a person is in the room.

## Problem Description

As a basis for the energy prediction, multiple solar harvesting devices have been deployed in different rooms of the ETZ building at ETH Zurich. In particular, we look at four stations with different light environments. For each station, a solar power trace over a year was recorded. Based on this historical data, prediction models can be built and evaluated.

The power received by the stations is sampled around once per second. Since we do not focus on predictions over such a short time, the measurements are first aggregated with different aggregation intervals between 10 minutes and 1 day.

The goal is then to predict the energy received within the next time interval. For a scale independent of the length of the interval, we instead consider the average power over the interval.

### **Contribution**

The main contribution of this thesis is the development and evaluation of different models to predict the power of an indoor solar energy harvesting system. In particular, we look at linear regression models, multivariate adaptive regression splines, decision trees and random forest models.

As an evaluation, we perform different experiments to analyze the properties of the models. These include:

1. Overall accuracy of the different models by multiple performance metrics.
2. Performance trade-offs when limiting the number of input variables.
3. Most important variables by different variable importance metrics.
4. Performance when predicting for different prediction intervals.
5. Performance of a global model, that is trained by data from other stations than the one it predicts for.
6. Runtime complexity and resource requirements for the models.

### **Thesis Outline**

We first give some background information about the stations and the available measurements in Chapter 2. Furthermore, we introduce the models that were used for the prediction. Next, we describe the different experiments that were conducted to assess the characteristics of the models in Chapter 3. In Chapter 4, we evaluate and discuss the results of these experiments. Finally, Chapter 5 gives a conclusion and short summary of the gained insights.

# Preliminaries

---

In this chapter, we first introduce the stations in Section 2.1. We then discuss the available measurements in Section 2.2 and their derived variables in Section 2.3. We also give some theoretical background on the models that were used for the prediction in Section 2.4.

## 2.1 Stations

For this project, we worked with four stations deployed in different rooms. These stations have different light environments, which leads to unique characteristics of the harvested power. To visualize this behavior, we plot the hourly aggregated power by time of day for each station in Figure 2.1. An overlaid color density representation helps identifying a general trend among the many data points from a year of data. The station numbers represent host names that are not relevant to this project.

For stations 6 and 13, we can make out a clear distinction between two power levels. These are the times when the light in the room is switched on or off. For stations 14 and 17, we see an additional effect taking place. In the morning and the evening respectively, these stations can experience direct sunlight, which results in much higher amounts of energy than the artificial light. This effect is also dependent on the time of the year.

## 2.2 Available Measurements

Over the course of a year, the following features were measured in parallel, which are then used to predict the harvested power. We group them into indoor and outdoor measurements. The indoor measurements are measured directly at the deployed device. The outdoor measurements are obtained from the Zurich Fluntern weather station, which is around a kilometer from the ETZ building. Additionally, a timestamp of the measurements is recorded.

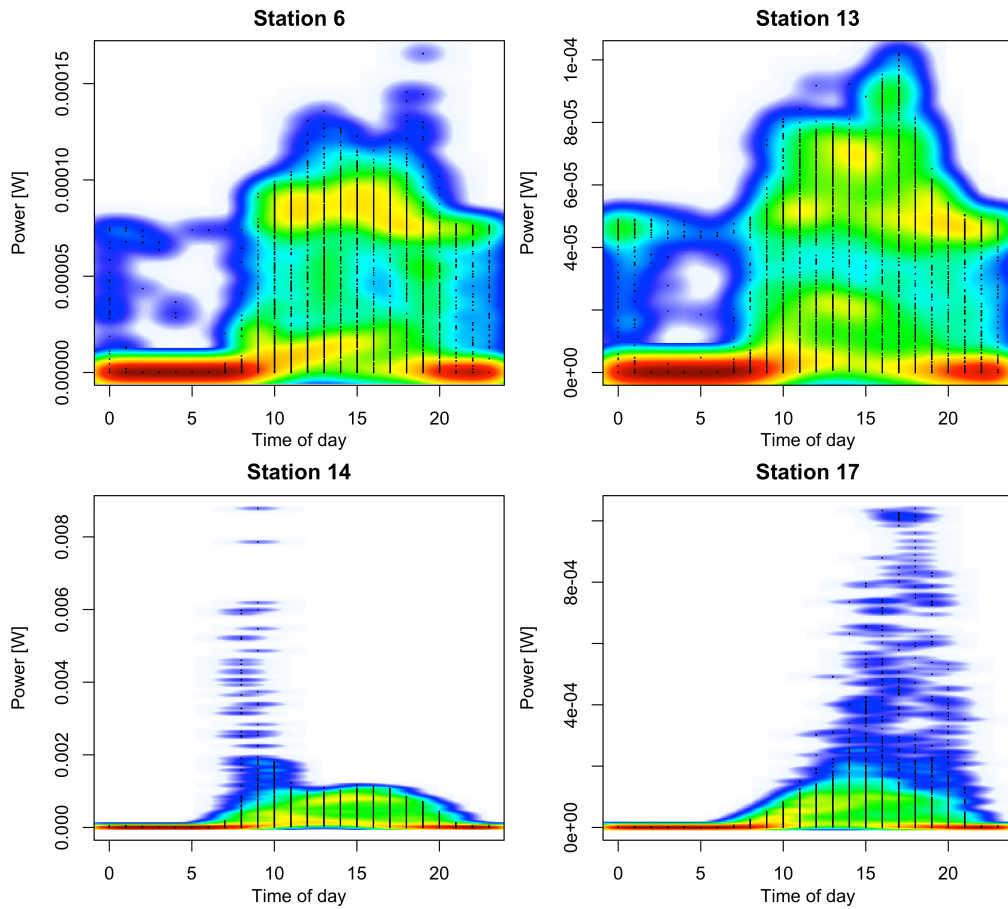


Figure 2.1: Power by time of day for each station.

### Indoor measurements

**Power ( $P_{in}$ )** The harvested power in watts.

**Illuminance ( $E_v$ )** The illuminance at the station in lux. Two sensors are used for this, of which we use the average.

**Temperature ( $T_{amb}$ )** The temperature at the station in degrees Celsius.

**Atmospheric pressure ( $P_{amb}$ )** The atmospheric pressure at the station in pascal.

**Relative humidity ( $RH_{amb}$ )** The relative humidity at the station.

### Outdoor measurements

**Temperature ( $meteo\_T$ )** The outdoor temperature in degrees Celsius.

**Atmospheric pressure (meteo\_P)** The atmospheric pressure in hectopascal.

**Irradiance (meteo\_Irr)** The solar irradiance in  $W/m^2$ .

**Sunshine duration (meteo\_Sunshine)** The sunshine duration in minutes of sunshine per interval [11].

**Dew point (meteo\_Dew)** The dew point in degrees Celsius.

## 2.3 Variable Selection

For the models we use, we have to extract predictor variables from the time series data to be able to apply the models. For each time interval, we can use the information of all features from the past intervals. As an example, an intuitive variable is the power harvested in the last interval. We name this variable `P_in.lag1`, as it is lagged by one interval compared to the power to be predicted.

In the following, we describe all variables that are used in the models. This is for a prediction interval of 1 hour. Similar variables are used for other intervals.

### Power variables

As power is the target variable, we put emphasis on past values of it. These are the power harvested in the past one, two, three and four hours respectively (`P_in.lag1`, `P_in.lag2`, `P_in.lag3`, `P_in.lag4`), but also the power in the same interval on the past day or week (`P_in.lag24h`, `P_in.lag1w`).

### Sensor and meteo variables

For all other features listed in Section 2.2, we use a subset of the time lags described above. For example, for the indoor temperature this includes `T_amb.lag1`, `T_amb.lag2` and `T_amb.lag24h`.

### Time variables

A last group of variables is derived from the timestamp. These include simple variables such as the time of day (`par.time`, values: 0-23), the day of the week (`par.weekday`, values: 1-7) or the day of the year (`par.yearday`, values: 1-366). Additionally, we also extract some information about the working hours. Firstly, a binary variable `par.workday` that is 1 for Monday through Friday and 0 on weekends. We do not consider special holidays here. Secondly, a binary variable `par.workhour` that is 1 if it is a work day and the time of day is between 09:00 and 18:00.

## 2.4 Models

### 2.4.1 Linear Regression

The first model we consider is a multiple linear regression. The target variable is modeled as a linear combination of all predictors with an added intercept. This can be described as

$$\hat{Y} = \beta_0 + \beta\mathbf{X},$$

where  $\hat{Y}$  is the predicted value,  $\mathbf{X}$  the vector of predictors,  $\beta$  the vector of coefficients and  $\beta_0$  the intercept.

With a given training sample, we use the least squares method to fit the model.

One of the assumptions of a linear regression model is that the relationship between dependent and independent variables is linear. This assumption is not necessarily justified in our case, which can make the linear model inaccurate.

In this project, we used the R packages `caret` and `stats` to fit linear models.

### 2.4.2 Multivariate Adaptive Regression Splines

As an extension to linear regression, multivariate adaptive regression splines (MARS) [7] can capture nonlinear relations and interactions between predictors. Where a linear model is limited to fitting a single line through all observations, a MARS model can include kinks in the predicted function.

We can describe a MARS model as follows:

$$\hat{Y} = \sum_{i=1}^k \beta_i B_i(x_i)$$

Each term in the sum includes a constant coefficient  $\beta_i$  and a basis function  $B_i(x_i)$ , where  $x_i$  is one of the predictor variables. The basis functions can take the following forms:

1. Constant 1, as the intercept.
2. A hinge (or hockey stick) function of the form  $\max(0, x_i - c_i)$  or  $\max(0, c_i - x_i)$ .
3. A product of multiple hinge functions for different variables, modeling interactions between variables. The interaction degree is the number of hinge functions.

A MARS model can include an arbitrary number of terms and does not have to include a term for each variable.

Training a MARS model consists of a forward pass and a pruning pass. During the forward pass, the model repeatedly adds basis functions that give the maximum reduction in residual sum of squares (RSS). This is done until a specified number of terms is reached or the decrease in RSS is below a defined threshold.

Since this forward pass tends to overfit, the model is then pruned in a backward pass. This pruning pass iteratively removes terms to select the subset of terms that minimizes the generalized cross-validation statistic (GCV) [6].

In this project, we used the R package `earth` to fit MARS models. The bounds of the forward pass were defined by a maximum number of terms of 120 and an RSS threshold of  $10^{-6}$ . We also specify a maximum interaction degree of 2 to avoid overfitting.

### 2.4.3 Decision Tree

A different type of prediction models are tree-based models. These are fundamentally different from linear regression or MARS models in that they do not calculate continuous values. Instead, they predict by selecting one of many predetermined values, similar to a classification. The simplest form of this is a decision tree, specifically a regression tree in our case [5]. It can be described as a sequence of decisions. Traversing the tree from the root, each node represents a decision based on the predictors. Each leaf then contains a predicted value for the target variable.

Each node  $N_i$  is a binary decision rule of the form  $(x_i >^? c_i)$ , where  $x_i$  is one of the predictor variables and  $c_i$  is a threshold.

The process of training a decision tree consists of recursively partitioning the entire training sample into smaller subsets. At each node, all possible splits across all predictors and thresholds are tested. The best split is chosen, measured by the Gini impurity [8]. With more splits, the tree grows continually until the process is terminated by a specified condition. This can be a maximum depth, a maximum number of nodes or a minimum number of samples in a leaf. The value of a leaf is then chosen to minimize the squared error.

In our case, after tuning the maximum number of leaf nodes, we set the limit for the number of leaves at 64.

Decision trees are not very robust. A small change in the training sample can lead to completely different trees. Additionally, they tend to overfit if the tree size is not sufficiently limited.

In this project, we used the R packages `rpart` and `randomForest` to fit decision trees.



#### 2.4.4 Random Forest

A much more robust tree-based model is the random forest [4]. The idea is to build many uncorrelated decision trees and output the mean of all predicted values. This makes random forest very resistant to overfitting and allows building much deeper trees.

Each tree is trained on a subset of the entire training set, obtained by sampling the training set  $n$  times with replacement, where  $n$  is the size of the training set. This results in each tree covering approximately 63% of the training set.

Building the trees is then similar to the single decision tree described above, with one difference: At each split, only a fraction of all variables is considered. This is to further force trees to be unique. We found that setting this to a third of all variables leads to good results for different numbers of variables.

In random forest, we do not need to tune the depth of the trees, as overfitting is prevented by the many trees. We therefore only set the minimum number of samples per leaf to 5. More trees are always better, but we settle for 500, as improvements in accuracy are minimal beyond that.

In this project, we used the R package `randomForest` to fit random forest models.

# Prediction

---

In this chapter, we look at ways of assessing different properties of the prediction models. We start by outlining the relevant characteristics that were evaluated for the models:

**Accuracy** The most important quality of a model is the prediction accuracy. In Section 3.1, we describe different performance metrics that were used to measure the accuracy of the models.

**Input variables** Naturally, some features are more important than others, and some might even be redundant. By using fewer variables, we end up with a less complex model, need fewer sensors to measure the features and use less memory to store the variables. On the flipside, we expect a decrease in performance, as less information is available for prediction. In Section 3.2, we describe a process to evaluate the performance when iteratively removing variables. We also determine the most important variables for the prediction.

**Prediction interval** In all other experiments, we always work with an aggregation and prediction interval of 1 hour. This might however vary depending on the application. In Section 3.3, we describe how we evaluate the performance when predicting for different intervals, from 10 minutes to a day.

**Global model** So far, models are trained for each station on its own data. As a consequence, we need historical data for each station before prediction models can be built. We therefore have the problem that a newly deployed station cannot immediately start predicting its power, but instead has to first gather training data. It would be desirable to instead have a global model that can be deployed right away for a new station. A solution would be a model that is trained by data from other, already existing stations. In Section 3.4, we describe such a global model and how its performance was evaluated.

**Model complexity** If a model is to be deployed in practice, the complexity of it will be constrained. That is the computational complexity at runtime when predicting new values and also the space a model takes up in memory. While a linear model is very efficient in both of these aspects, this is not the case for all models. In Section 3.5, we describe the evaluation of how the different model types compare regarding complexity and how performance is affected by reducing model complexity for each type.

**Memory footprint of runtime variables** In addition to the memory used by the model itself, past values used for prediction also have to be stored. The more different variables are used and the longer the values are stored, the more memory is consumed. In Section 3.6, we describe the evaluation of how the size of this variable memory influences the performance of the models.

## 3.1 Accuracy

### 3.1.1 Performance Metrics

To evaluate the accuracy of the models, we can calculate different performance metrics. We selected the following metrics, as these are commonly used in the state of the art. In the definitions,  $\hat{Y}$  is a vector of  $n$  predicted values and  $Y$  is the vector of the true observed values.

**Mean absolute error**

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |\hat{Y}_i - Y_i|$$

**Median absolute error**

$$\text{MedianAE} = \text{median}(|\hat{Y} - Y|)$$

**Mean squared error**

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - Y_i)^2$$

**Root-mean-square error**

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - Y_i)^2}$$

For all evaluation experiments following in this chapter, we use the root-mean-square error to measure the accuracy of the models.

### Mean absolute percentage error

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{\hat{Y}_i - Y_i}{Y_i} \right|$$

It should be noted that the mean absolute percentage error is a poor performance metric for multiple reasons [16]. The most important problem in our case is that the observed values in the denominator can be close to zero. To avoid that the MAPE is dominated by these resulting extreme values, we only consider predictions where the observed value is greater than 10% of the maximum observed value. This still gives lower absolute values significantly more weight and we only include MAPE and the median absolute percentage error to give some sense of the dimensions of the errors.

### Median absolute percentage error

$$\text{MedianAPE} = \text{median} \left( 100\% \cdot \left| (\hat{Y} - Y) \oslash Y \right| \right)$$

Similar to the mean absolute percentage error, observations smaller than 10% of the maximum observed value are not taken into account for calculating the error.

### Residual sum of squares

$$\text{RSS} = \sum_{i=1}^n \left( \hat{Y}_i - Y_i \right)^2$$

### $R^2$

The coefficient of determination,  $R^2$ , measures the percentage of the variability of the target variable that is explained by the model. This ranges from 0 to 1 with 1 being the optimum.

$$R^2 = 1 - \frac{\text{RSS}}{\sum_{i=1}^n (Y_i - \bar{Y})^2}$$

where  $\bar{Y}$  is the mean of the observed data:  $\bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_i$

### 3.1.2 Model Configuration

To evaluate these metrics, we use 5-fold cross-validation to train models of each type on each station. The model parameters were set as follows:

**Linear Model** All variables and an intercept are included.

**MARS** Maximum interaction degree: 2.

**Decision Tree** Number of leaf nodes: 64.

**Random Forest** Minimum number of samples per leaf node: 5. Number of trees: 500.

The results of this evaluation are presented and discussed in Section 4.1.

## 3.2 Input Variables

### 3.2.1 Recursive Feature Elimination

To determine how the number of variables affects the performance of the models, we train models of all sizes for each model type. Since it is computationally infeasible to exhaustively search for the optimal combination of variables for each model size, we have to use a faster variable selection strategy. We therefore use recursive feature elimination (RFE) [12]. The idea of RFE is to rank features based on some importance criteria, remove the least important one and start over. This process is described in Algorithm 1.

---

**Algorithm 1** Recursive Feature Elimination (RFE)

---

**Input:**  $n$  variables, model fitting algorithm, variable importance metric

start with the full set of  $n$  variables

**for all** subset sizes  $i = n \dots 1$  **do**

    train the model using the remaining  $i$  predictors

    evaluate model performance using cross validation

    calculate variable importance and rank the variables

    eliminate the least important variable

**end for**

---

With RFE, we calculate a model of each model type for each subset size and evaluate them by RMSE. The configurations for the models are the same as described in Section 3.1.2. The variable importance ranking is calculated based on the following metrics:

**Linear Model** The absolute value of the t-statistic for each variable.

**MARS** For each variable, calculate the total decrease in GCV [6] caused by adding a term, summed over all terms that include the variable. Unused variables are assigned the value zero. Variables with larger decrease in GCV are considered more important.

**Decision Tree** For each variable, calculate the decrease in prediction accuracy (measured by the increase in MSE) caused by randomly permuting that variable in the testing sample. This permutation essentially nullifies the predictive power of a variable. Then, variables with larger decrease in accuracy are considered more important [4].

**Random Forest** Similar to the decision tree, but the decrease in accuracy is measured for each tree and its out-of-bag sample. This is then averaged over all trees.

Results and discussion of this experiment are in Section 4.2.1.

### 3.2.2 Importance of Variables

We can then also look at which are the most important variables. For this, we take the variable importance measures calculated in the first iteration of Algorithm 1 for each model. We cannot compare the metrics directly, as the scales are different. For this reason, we assign each variable a score so that the highest scoring variable is the most important. For variables with the same importance value, the scores are averaged. This happens for the decision tree and MARS model, where multiple excluded variables are given the value zero. As an example, with 5 excluded variables, they are all assigned the average score of 3. Results and discussion of this experiment are in Section 4.2.2.

### 3.2.3 Variable Classes

To get a more complete view of how the models perform when including certain features, we group the variables in classes. This allows us to exhaustively evaluate all combinations of variable classes and analyze the impact of each class on the prediction accuracy. We define the following classes:

**Power** Variables of past power values. (P\_in.lag1, P\_in.lag2, P\_in.lag3, P\_in.lag4, P\_in.lag24h, P\_in.lag1w)

**Time** Variables derived from the current timestamp. (par.time, par.weekday, par.yearday, par.workhour, par.workday)

**Sensors** Variables extracted from the ambient sensor at the station. (Ev\_amb.lag1, Ev\_amb.lag2, Ev\_amb.lag24h, T\_amb.lag1, T\_amb.lag2, T\_amb.lag24h, P\_amb.lag1, P\_amb.lag2, P\_amb.lag24h, RH\_amb.lag1, RH\_amb.lag2, RH\_amb.lag24h)

**Meteo** Variables extracted from the measurements at Zurich Fluntern weather station. (meteo\_lrr.lag1, meteo\_lrr.lag2, meteo\_lrr.lag24h, meteo\_T.lag1, meteo\_T.lag2, meteo\_T.lag24h, meteo\_P.lag1, meteo\_P.lag2, meteo\_P.lag24h, meteo\_Sunshine.lag1, meteo\_Sunshine.lag2, meteo\_Sunshine.lag24h, meteo\_Dew.lag1, meteo\_Dew.lag2, meteo\_Dew.lag24h)

We then train and cross-validate models for all combinations of these classes. The model configurations are as described in Section 3.1.2. Results and discussion of this experiment are in Section 4.2.3.

### 3.3 Prediction Interval

To analyze the relation between performance and prediction interval, we evaluate models for intervals of 10 minutes, 30 minutes, 1 hour, 4 hours and 1 day. For simplicity, we only look at the linear and random forest models. To optimize model training time of random forest, we limit the tree depth for the shorter prediction intervals. This has no negative impact on prediction accuracy. We set the minimum leaf size to 50 for 10 minute intervals, to 10 for 30 minutes, to 5 for 1 and 4 hours and to 3 for 1 day. The number of trees in a forest is 500 and the accuracy is evaluated by the cross-validated RMSE. Results and discussion of this experiment are in Section 4.3.

### 3.4 Global Model

We define a global model as a model that is not trained by any data of the station it is used to predict the power for. Therefore, for each station we trained models on data from all other stations. These models are then used to predict the power received by the remaining station. We can compare the resulting errors to the cross-validation errors from training and testing on the same station. For this evaluation, we use the model configurations as described in Section 3.1.2. Here, we again use prediction intervals of 1 hour. Results and discussion of this experiment are in Section 4.4.

### 3.5 Model complexity

To measure the complexity of the prediction models, we are interested in two metrics:

**Runtime complexity** The number of operations that have to be performed at runtime to predict a new power value for the next interval. An issue is that we cannot directly compare the different operations used in the models. These are addition, multiplication and comparison. Therefore we have to make some simplifying assumptions. Assuming an ARM Cortex-M4 processor architecture [1], a typical embedded processor, add (ADD), multiply (MUL) and compare (CMP) operations all take up one cycle. However, the architecture also supports a single-cycle multiply-accumulate (MLA) operation, that can be used for the linear and MARS models. We measure the runtime complexity as the number of operations of these four types.

**Model size** The size in bytes a model takes up in memory. We assume that values are stored in single-precision floating-point format and use 32 bits. For this analysis, the memory required to store the structure of the model and trees is omitted, as it is negligible compared to the memory of the values.

In the following, we describe how the complexity of each model is adjusted and how the two described metrics are calculated in each case:

**Linear Model** Here, we simply define the number of predictor variables. This is the same process as described in Section 3.2.1.

The number of operations equals the number of variables, since we require one MLA operation for each variable.

The number of stored values equals the number of variables plus one, since we need one value for each variable and the intercept.

**MARS** For MARS, we adjust the number of terms in the model after pruning from 0 to 40. Here, we do not allow any interactions between variables to keep the complexity constant for each number of terms.

The number of operations needed to evaluate a MARS model is two times the number of terms. For each term we require one comparison and one multiply-accumulate.

The number of stored values is two times the number of terms plus the intercept.



**Decision Tree** For a single tree we adjust the depth of the tree. Instead of limiting the maximum depth directly, we set the minimum size of the leaves in the tree. Subsequently, the average depth of all leaves can be calculated. The number of operations is then the depth of the tree, as traversing each level costs one comparison.

The number of stored values is the total number of nodes in the tree.

**Random Forest** Here, we adjust the number of trees in the forest from 1 to 1000, but keep the depth of the trees constant by limiting the number of leaves to 400. It is important to note though that this does not optimize the performance by complexity and does not create a perfect pareto front. For this, a two-dimensional search in number of trees and tree depth would have to be performed.

The number of operations needed for this type of model is the number of trees multiplied by the depth of each tree plus one addition for each tree to combine the results.

The number of stored values equals the number of trees multiplied by the number of nodes in a tree.

All the obtained models are then evaluated for their cross-validated accuracy measured by the RMSE. Results and discussion of this experiment are in Section 4.5.

### 3.6 Memory Footprint of Runtime Variables

In all the models, for each variable used, a sliding window of its measurements is kept, that has the length of the maximum time looked back. As an example, working with 1 hour aggregation intervals, assume the power at the same time on the past day is used for prediction. This requires that 24 values are stored, one for each hour. From these values, the power of the past one or two hours can also be extracted without needing extra memory.

To analyze this behavior, we extend the evaluation described in Section 3.2. For each model generated, the amount of memory required to store the variables is computed. We can then see how much memory is needed for a model of a desired accuracy. This is under the caveat that the variable selection does not optimize for memory. During the recursive feature elimination (Algorithm 1), variables are not weighted in any way, meaning the algorithm could select one “expensive” variable over two “cheap” variables that might have more combined predictive power with less memory used. Results and discussion of this experiment are in Section 4.6.

# Evaluation

---

In this chapter, we present and discuss the results of the experiments that were described in Chapter 3. We mostly focus on one station (Station 6) for the presentation of the results and expand more where stations show unique behavior.

## 4.1 Accuracy

### 4.1.1 Performance Metrics

We first present the evaluation of the performance metrics as described in Section 3.1.1. Table 4.1 shows the metrics for the different models as they predict the received power of station 6 with a prediction interval of 1 hour. Additional tables for the other stations can be found in Appendix A.1 - A.3.

Table 4.1: Performance metrics for the different models predicting for station 6. The best result for each metric is highlighted.

	Linear Model	MARS	Decision Tree	Random Forest
MAE	9.06e-06	7.84e-06	7.93e-06	<b>7.46e-06</b>
MedianAE	2.94e-06	2.10e-06	1.86e-06	<b>1.68e-06</b>
MSE	2.47e-10	2.17e-10	2.48e-10	<b>2.08e-10</b>
RMSE	1.57e-05	1.47e-05	1.57e-05	<b>1.44e-05</b>
MAPE	34.3%	33.4%	35.2%	<b>32.5%</b>
MedianAPE	20.3%	17.4%	17.2%	<b>16.3%</b>
RSS	4.56e-07	4.00e-07	4.58e-07	<b>3.83e-07</b>
R <sup>2</sup>	0.802	0.827	0.802	<b>0.834</b>

We can observe a general trend that the random forest model produces the most accurate results, followed by the MARS model. There are no large discrepancies between the metrics. The random forest is the best in every metric. The greatest differences between the models can be seen in the median absolute

error, where the linear model produces 75% higher errors than the random forest model.

### 4.1.2 Errors over the day

To better understand where the errors are originating, we now analyze the distribution of the errors over a day. The configuration is the same as above, but we are only looking at the errors of a random forest fit, as this is the most robust model. Figure 4.1 shows the distribution of the errors over a day. The first subplot shows the average power received per time of day to give some perspective about the magnitude of the errors. In the second subplot, the absolute error by the time of day is visualized by a box plot. This is a traditional Tukey box plot, with the box representing the first, second and third quartile. The whiskers represent the maximum and minimum datum still within 1.5 interquartile range of the third and first quartile. The interquartile range is the difference between the third and the first quartile. The third subplot shows a box plot of the mean absolute percentage error (MAPE) by the time of day.

The MAPE is filtered as described in Section 3.1.1, which is why there are no values for some hours. Because the MAPE tends to be higher with lower true values, it produces the lowest errors when the average power is high. This is in contrast to the absolute error. During the night, absolute errors are very low, as the power received is consistently near zero. We can make out 3 peaks of the absolute errors over the day: in the morning at 10, at noon at around 13 to 14 and in the evening at 19. At these times it is harder to predict whether someone is in the office, as people do not always come in and leave at the same time.

## 4.2 Input Variables

### 4.2.1 Results of the Recursive Feature Elimination

From the RFE described in Section 3.2.1 we obtain models for each model type and number of variables. We now evaluate their performance by the RMSE. Figure 4.2 shows the accuracy of each model when reducing the number of predictor variables with RFE. Additional plots for the other stations can be found in Appendix A.1. The decision tree model clearly stands out as its error increases rapidly when removing variables. This has to be attributed to the fact that the calculation of variable importance is suboptimal for decision trees. This leads to important variables being removed before some of the weaker predictors. With optimal subset selection, for example an exhaustive search, we expect the behavior of the decision tree to be similar to the other models.

A second observation we can make is that for the other models, the error only starts to increase significantly when removing the last few remaining variables.

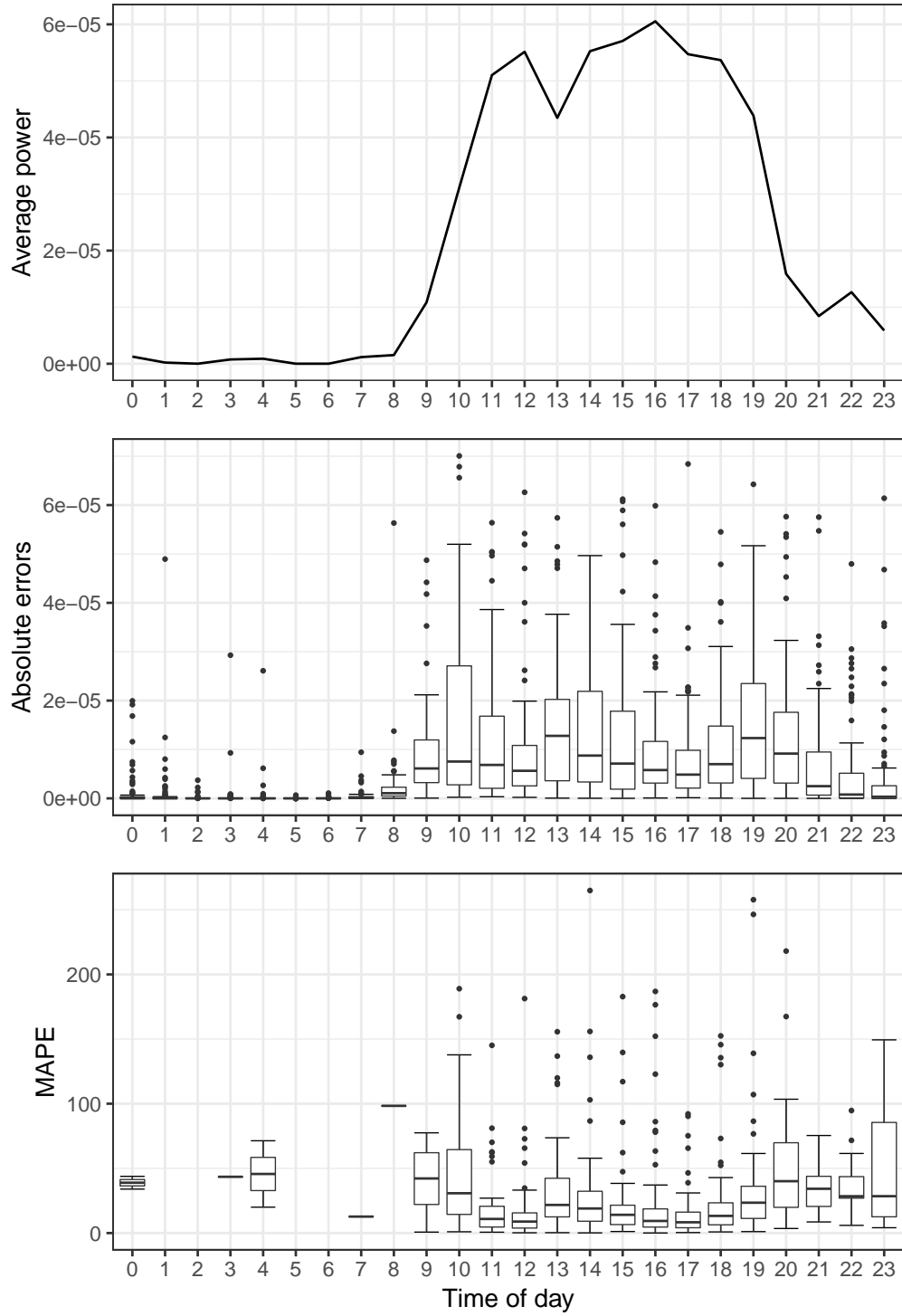


Figure 4.1: Errors by time of day with random forest on station 6.

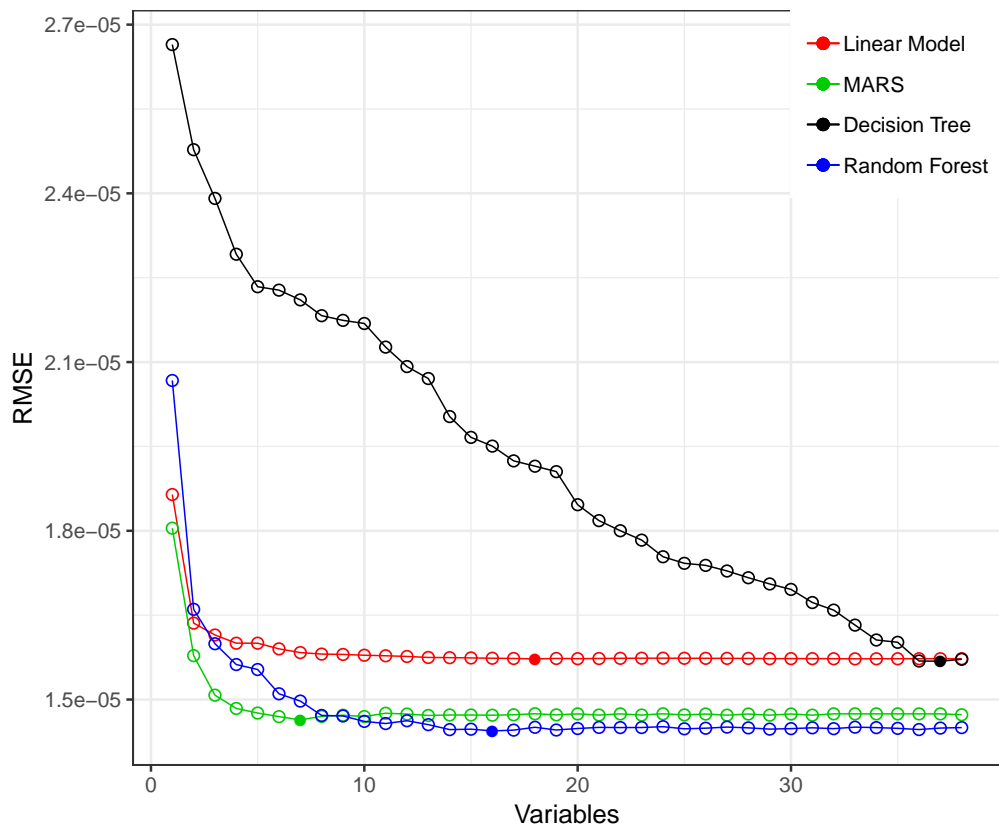


Figure 4.2: Model performance by number of variables for station 6. The minimum errors are highlighted for each model.

We conclude that the five to ten most important variables already contain almost all of the predictive power. Beyond a certain point, including more variables makes the model even slightly worse. These variables therefore only contain irrelevant or redundant information on which the model overfits.

### 4.2.2 Importance of Variables

We now evaluate the variable importance as described in Section 3.2.2. The resulting scoring for the different models on station 6 is shown in Figure 4.3. Additional plots for the other stations can be found in Appendix A.2. The bars are sorted by the average score within a variable. Section 2.3 gives an introduction to all the variables.

This plot must be interpreted with care, as the rankings are not very robust and the differences between models come mostly from the different metrics. Still, an observation that is also consistent across the different stations is that the time of day (`par.time`) is less important for the linear model than for the other models. This can be explained by the fact that the linear model is the only model that cannot incorporate nonlinear effects. The relation between time of day and received power is nonlinear as can be seen in Figure 4.1.

Another notable effect comes from the strongly correlated predictor variables. As an example, the information between the illuminance of the last hour (`Ev.lag1`) and the power of the last hour (`P.in.lag1`) is mostly redundant. A model selecting either of the two features first will then value the other less, as it has already included the information. This effect can be seen with the linear model and the decision tree. In random forest, it is counteracted by only sampling a fraction of the variables at each split, as well as averaging over many trees.

To see how the variables rank across the different stations, we now only look at the random forest variable importance, since it is the most robust. We then again take the variable importances calculated in the first iteration of RFE. The resulting scoring is shown in Figure 4.4.

The most important variables are consistent across the stations and unsurprisingly these are power and illuminance of the last hour. It seems that most of the meteorological and ambient sensor information is not as important as historical power data and time, which is again not surprising. Here, we also see the effects of sunlight at the stations. Variables that are related to working days (`par.workday`, `par.weekday`, `par.workhour`) are significantly more important for stations 6 and 13 than they are for stations 14 and 17. The latter stations are more influenced by sunlight and are thus less reliant on a person being in the room.

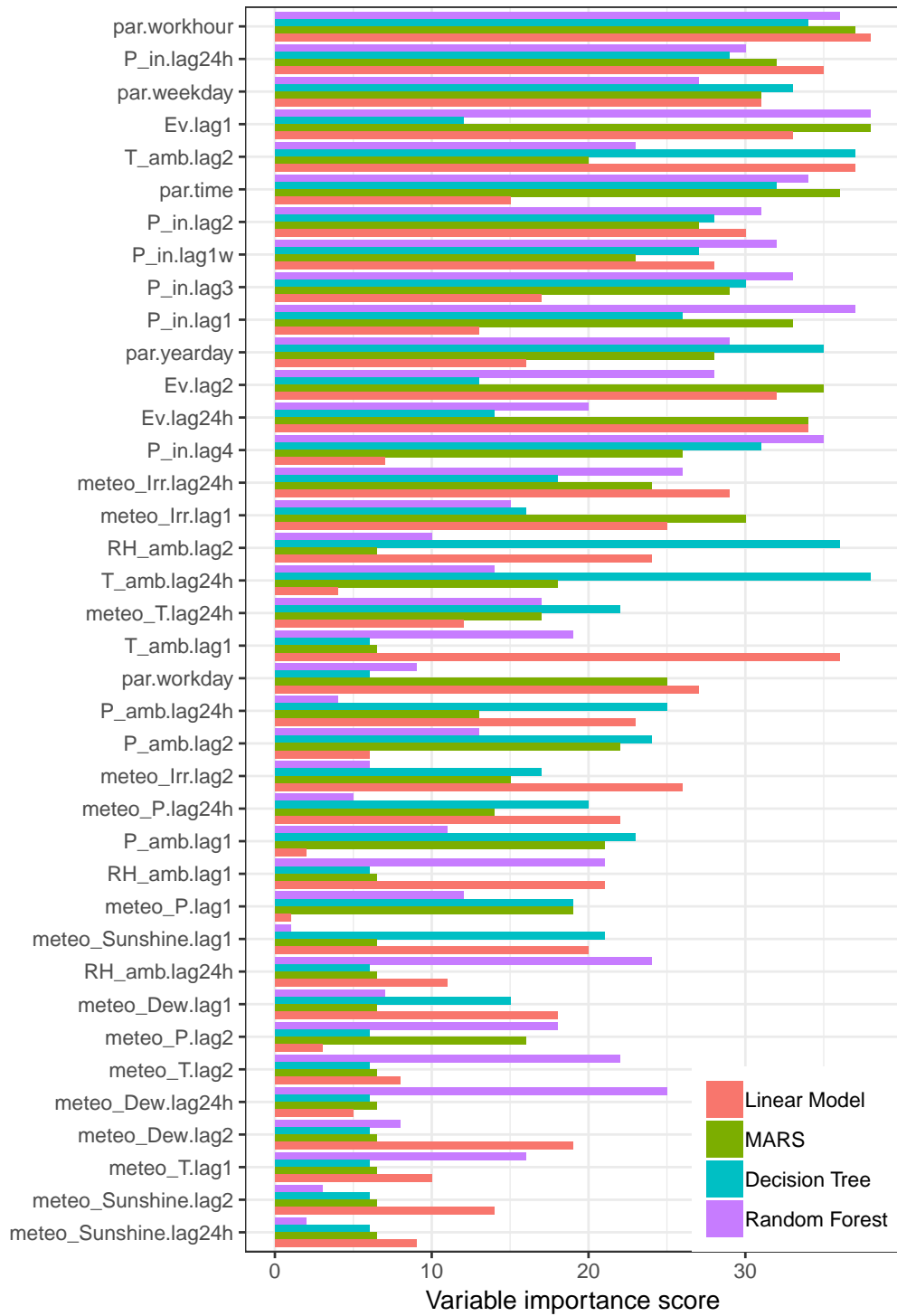


Figure 4.3: Variable importance calculated using the different models on station 6. Larger values mean higher importance.

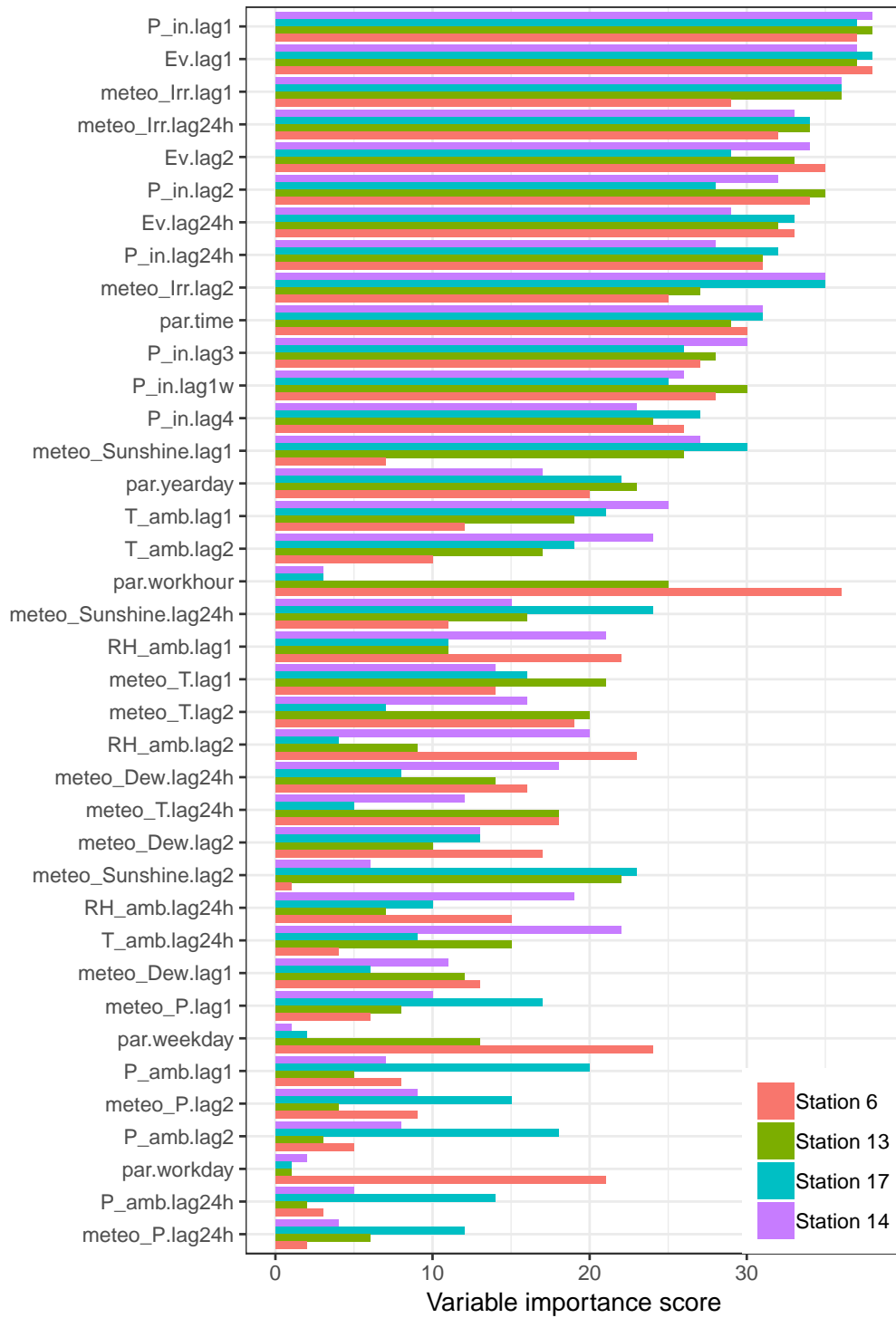


Figure 4.4: Variable importance calculated using random forest on the different stations. Larger values mean higher importance.



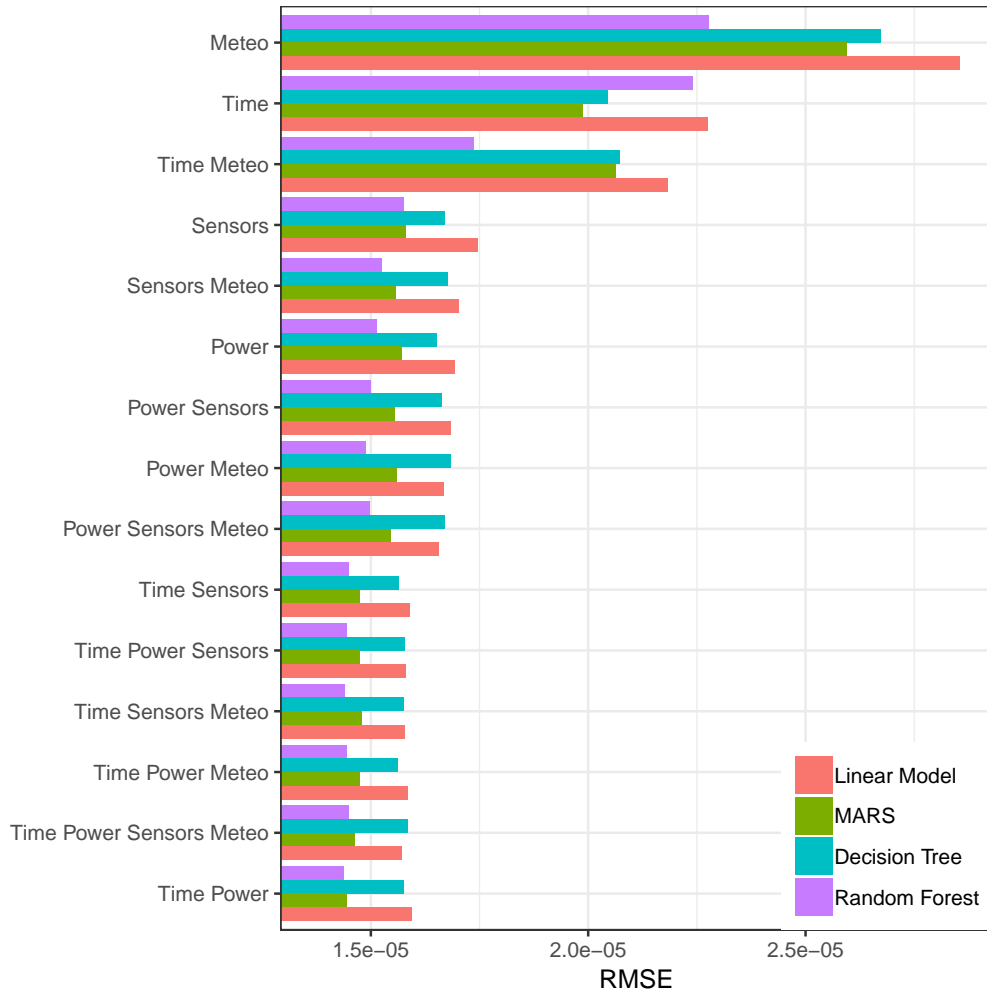


Figure 4.5: Model performance by variable classes for station 6.

### 4.2.3 Variable Classes

As described in Section 3.2.3, we now train and cross-validate models for all combinations of variable classes. Figure 4.5 shows the resulting RMSE values for station 6. Additional plots for the other stations can be found in Appendix A.3.

The relative performance of the models is very consistent across different combinations of classes. We can make out three groups of combinations. The best models contain at least the time class and either power or sensor class. Slightly worse are the models containing power or sensors but no time information. Much worse are models that do not contain either power or sensors. In this analysis, power and sensors are largely interchangeable. This is because the

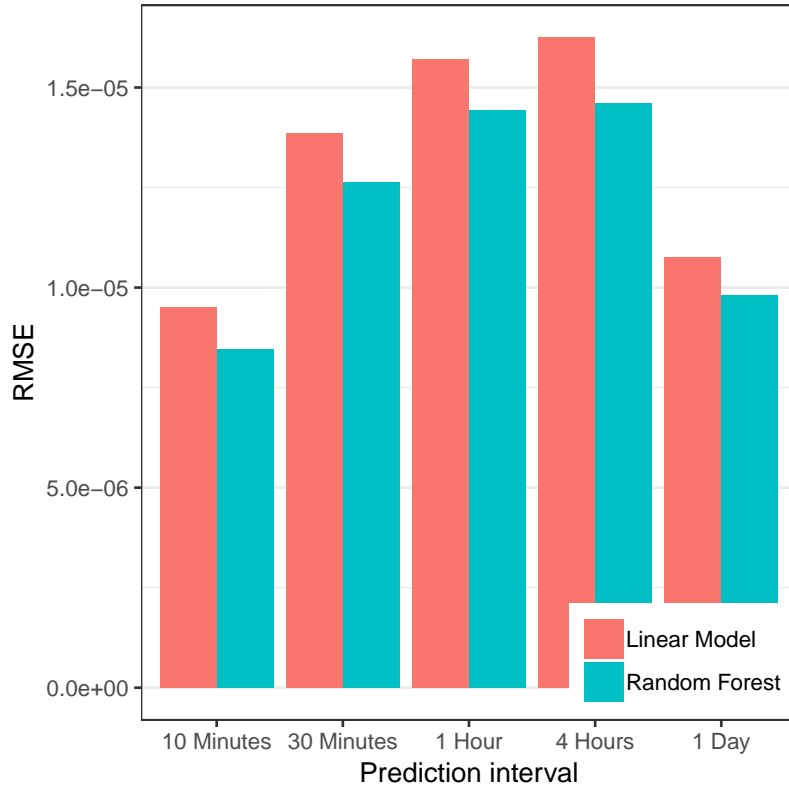


Figure 4.6: Model performance by prediction interval for station 6.

sensors class includes illuminance, which is very strongly correlated to received power. We verified this by excluding illuminance from the sensors, in which case the class expectedly became irrelevant.

We conclude that beyond the time class and either power or illuminance class, adding more variables does not improve the model significantly, if at all.

### 4.3 Prediction Interval

We now evaluate the performance for different prediction intervals, as described in Section 3.3. Since the predicted average power received in an interval does not have a time component, we can directly compare the error values. Figure 4.6 shows the model performance, measured by the RMSE, for different prediction intervals on station 6. Plots for the other stations can be found in Appendix A.4.

We can observe that both 10 minute intervals as well as 1 day can be more accurately predicted than the intermediate intervals. The effect is the same for

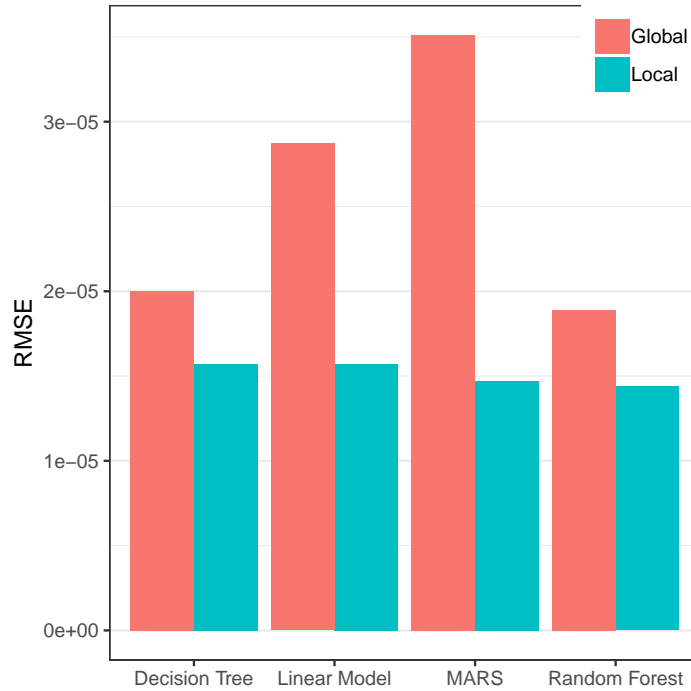


Figure 4.7: Performance of the global and local models compared for station 6.

both model types. For the daily interval, this can be explained by the lower variance of the aggregated power. There is much less variance of the average power over an entire day when compared to the other intervals, which makes it easier to predict. For 10 minutes, the power is more strongly correlated to past values, as it is likely to stay the same for a period of time. This strong temporal correlation also makes for a more accurate prediction.

#### 4.4 Global Model

In Section 3.4, we describe a global model that does not use a station's own data for training. We now compare global models of each type to the locally trained models. Figure 4.7 shows the prediction accuracy of the global and local models for station 6. Plots for the other stations can be found in Appendix A.5.

We observe that the global models are significantly worse for all model types, especially the MARS and linear model. A global model cannot capture the effects specific to this station and could also overfit on the individual situations of the other stations. The MARS and linear model are more prone to these overfitting effects as they will also extrapolate beyond the range of the training data.

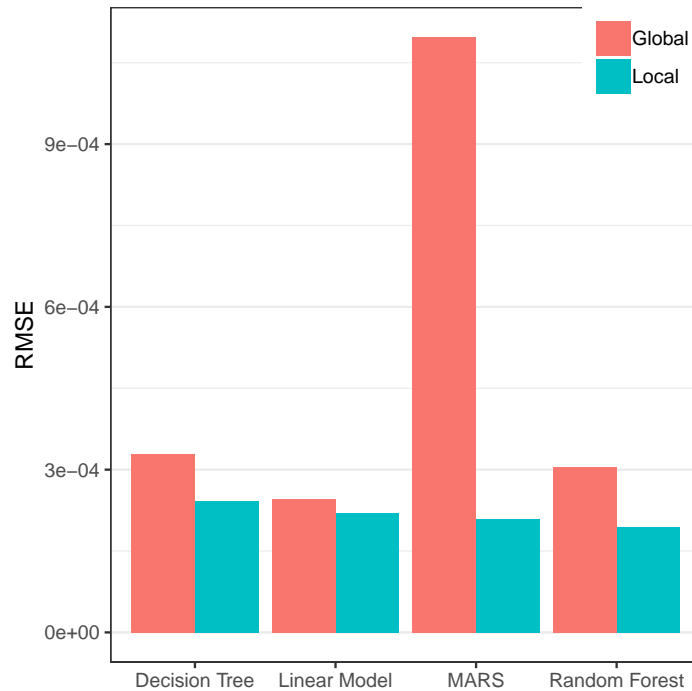


Figure 4.8: Performance of the global and local models compared for station 14.

#### 4.4.1 Effects of Sunlight

We see this even more pronounced for station 14, which receives much more power due to it being more exposed to sunlight. Figure 4.8 shows the same comparison as described above for station 14. MARS immediately stands out with a terrible performance of the global model. In this case, the MARS model frequently overestimates the received power by a large amount, as the behavior of the splines is uncontrolled outside the range of the training set. This makes MARS worse than a linear model, as it is not limited to a linear fit across the entire training set, but instead can model nonlinear kinks and interactions. The tree-based models are also restricted in that they cannot predict power values outside of the range contained in the training set. As a result, they under- or overestimate here if the true value is larger/smaller than the maximum/minimum of the training set.

### 4.5 Model Complexity

As described in Section 3.5, we now analyze how the different model types compare regarding complexity and how performance is affected by reducing model complexity for each type. The results for station 6 with a prediction interval of

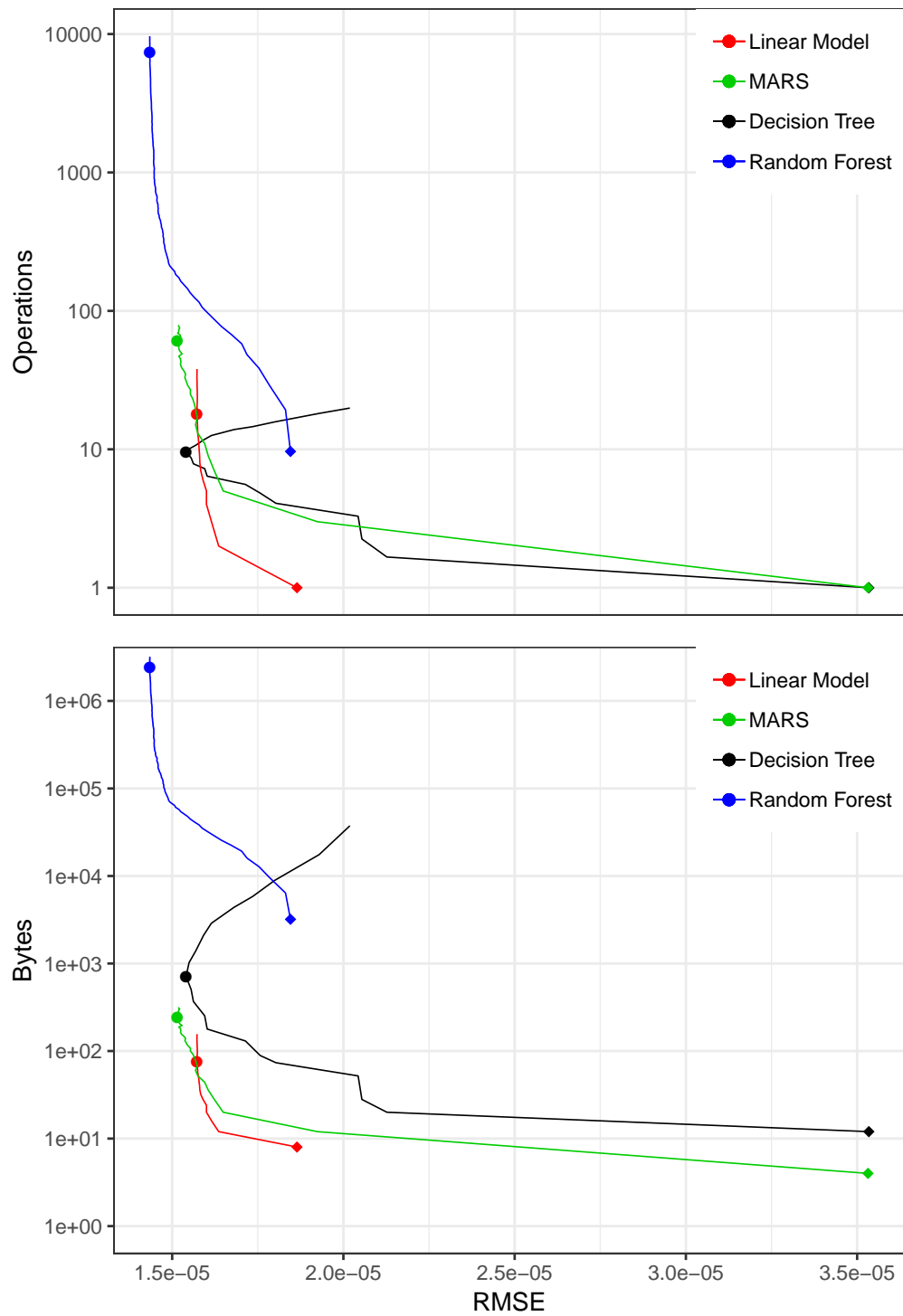


Figure 4.9: Memory and runtime complexity for each model compared to their performance on station 6.

1 hour are shown in Figure 4.9. Additional plots for the other stations can be found in Appendix A.6. In the first subplot, each curve represents the possible combinations of prediction accuracy and runtime complexity. A diamond marks the model with the lowest complexity and a dot marks the most accurate model. The second subplot is the same but for the resource requirements of the models. For each RMSE value, we can find the two complexity metrics required by the different model types to reach that accuracy.

As expected, for models of very low memory and runtime complexity, a linear model will be the best. Already beyond around 10 operations and 100 bytes, a MARS model performs better. An interesting trend can be observed for the decision tree, where the model performs worse with high complexity. This is because a tree of too high depth will overfit, while a too shallow tree underfits. In this case, the optimal tree depth is around 8. For a tree, the number of operation scales linearly with depth, while the model size grows exponentially. We see that the decision tree can compete with the MARS model in runtime complexity, but a tree for the same accuracy always needs more memory than MARS. The random forest reaches the lowest RMSE but with considerably higher complexity than the other model types. A random forest of 1000 trees consumes more than a megabyte in memory.

## 4.6 Memory Footprint of Runtime Variables

As our last experiment described in Section 3.6, we evaluate the memory needed to store the variables for prediction. Figure 4.10 shows the memory used to store the variables for each model generated by the RFE for station 6. Additional plots for the other stations can be found in Appendix A.7.

We observe that beyond a few hundred bytes, a lot of memory is used for small improvements to accuracy. The special behavior of the decision tree has been explained in Section 4.2.1. For MARS and the linear model, the space used to store the variable makes up most of the total memory footprint. For the decision tree, variable memory and model size are roughly similar and for the random forest, the model consumes significantly more memory than the variables.

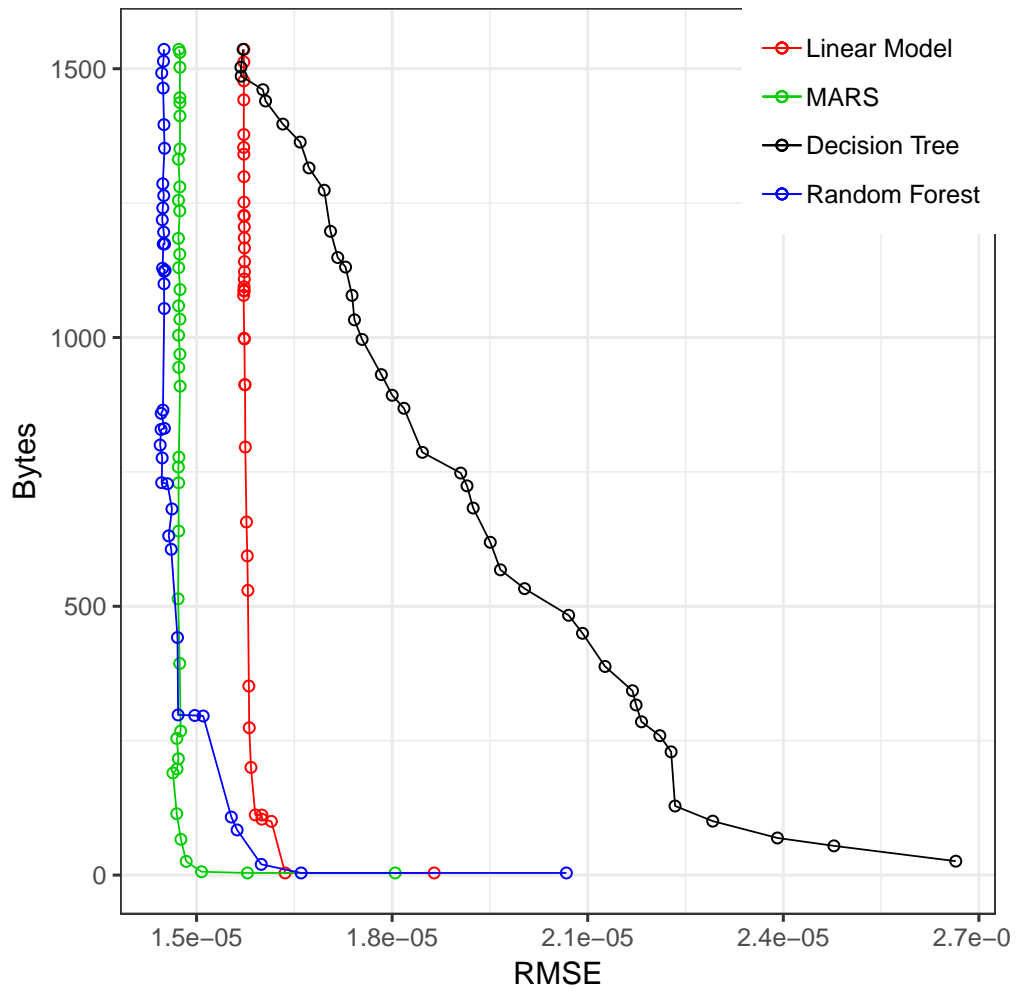


Figure 4.10: Variable memory and performance of each model generated by recursive feature elimination on station 6.

# Conclusion

---

In this thesis, we developed and evaluated models to predict the power of an indoor solar energy harvesting system. We considered linear regression models, multivariate adaptive regression splines, decision trees and random forest models.

The evaluation has shown that the random forest model is overall the most accurate and robust. Compared to a linear regression model, we observe around 10% reduction in RMSE consistently across the experiments.

A second conclusion is that only few of the variables contain almost all the predictive power. These are the variables derived from the time and past power values.

When analyzing different prediction intervals, we discovered that intervals of 1 to 4 hours are the most difficult to predict. These are short enough to have high variance, but not short enough for a high temporal correlation.

The unique environments of each location proved to be significant, making it difficult to implement a single model for all stations.

The random forest model is significantly more complex than the others, requiring more than 1000 operations at runtime and a megabyte of memory to store the model for the best accuracy.

## 5.1 Future Work

The models we developed struggled to predict when direct sunlight will hit the device. Only occurring at specific days of the year and times of the day depending on the location, our generally trained models are not a good fit for such an effect. However, the course of the sun relative to a specific location can be modeled very accurately. Together with established meteorological forecasting models, this could be used to more accurately predict solar influence at the stations.

Different models could be considered and evaluated, that might offer more accurate predictions. Some examples would be single- or multivariate ARMA or ARIMA [3], Markov models, or even long short-term memory approaches [9].



# Bibliography

- [1] *ARM Cortex-M4 Technical Reference Manual*, 2010. ARM 100166\_0001\_00\_en.
- [2] BACHER, P., MADSEN, H., AND NIELSEN, H. A. Online short-term solar power forecasting. *Solar Energy* 83, 10 (2009), 1772–1783.
- [3] BOX, G. E., JENKINS, G. M., REINSEL, G. C., AND LJUNG, G. M. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [4] BREIMAN, L. Random forests. *Machine learning* 45, 1 (2001), 5–32.
- [5] BREIMAN, L. *Classification and regression trees*. Routledge, 2017.
- [6] CRAVEN, P., AND WAHBA, G. Smoothing noisy data with spline functions. *Numerische mathematik* 31, 4 (1978), 377–403.
- [7] FRIEDMAN, J. H. Multivariate adaptive regression splines. *The annals of statistics* (1991), 1–67.
- [8] GINI, C. Variabilità e mutabilità. *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi* (1912).
- [9] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
- [10] INMAN, R. H., PEDRO, H. T., AND COIMBRA, C. F. Solar forecasting methods for renewable energy integration. *Progress in energy and combustion science* 39, 6 (2013), 535–576.
- [11] JARRAUD, M. *Guide to Meteorological Instruments and Methods of Observation (WMO-No. 8)*. 2008.
- [12] KOHAVI, R., AND JOHN, G. H. Wrappers for feature subset selection. *Artificial intelligence* 97, 1-2 (1997), 273–324.
- [13] KULLBACK, S., AND LEIBLER, R. A. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.
- [14] QUINLAN, J. R. Induction of decision trees. *Machine learning* 1, 1 (1986), 81–106.

- [15] STURGES, H. A. The choice of a class interval. *Journal of the american statistical association* 21, 153 (1926), 65–66.
- [16] TOFALLIS, C. A better measure of relative prediction accuracy for model selection and model estimation. *Journal of the Operational Research Society* 66, 8 (2015), 1352–1362.

# Additional Plots

Table A.1: Performance metrics for station 13

	Linear Model	MARS	Decision Tree	Random Forest
MAE	6.04e-06	5.32e-06	5.44e-06	<b>4.88e-06</b>
MedianAE	2.48e-06	2.03e-06	<b>1.48e-06</b>	1.54e-06
MSE	1.03e-10	9.12e-11	1.09e-10	<b>8.40e-11</b>
RMSE	1.01e-05	9.54e-06	1.04e-05	<b>9.16e-06</b>
MAPE	29.7%	28.0%	30.2%	<b>25.9%</b>
MedianAPE	17.5%	15.4%	15.3%	<b>13.3%</b>
RSS	1.79e-07	1.60e-07	1.90e-07	<b>1.47e-07</b>
R <sup>2</sup>	0.841	0.859	0.833	<b>0.870</b>

Table A.2: Performance metrics for station 14

	Linear Model	MARS	Decision Tree	Random Forest
MAE	8.38e-05	7.44e-05	7.06e-05	<b>5.55e-05</b>
MedianAE	3.65e-05	2.03e-05	8.68e-06	<b>6.58e-06</b>
MSE	4.84e-08	2.66e-07	6.12e-08	<b>3.76e-08</b>
RMSE	2.17e-04	2.94e-04	2.42e-04	<b>1.89e-04</b>
MAPE	22.3%	20.9%	22.6%	<b>17.3%</b>
MedianAPE	15.4%	14.7%	14.4%	<b>10.7%</b>
RSS	8.74e-05	4.80e-04	1.11e-04	<b>6.78e-05</b>
R <sup>2</sup>	0.664	0.705	0.610	<b>0.747</b>

Table A.3: Performance metrics for station 17

	Linear Model	MARS	Decision Tree	Random Forest
MAE	1.83e-05	1.56e-05	1.55e-05	<b>1.33e-05</b>
MedianAE	5.60e-06	1.85e-06	9.51e-07	<b>7.46e-07</b>
MSE	2.13e-09	2.36e-09	2.61e-09	<b>1.73e-09</b>
RMSE	4.60e-05	4.84e-05	5.09e-05	<b>4.15e-05</b>
MAPE	33.4%	35.3%	38.7%	<b>31.1%</b>
MedianAPE	26.5%	26.4%	30.3%	<b>22.3%</b>
RSS	3.91e-06	4.34e-06	4.81e-06	<b>3.18e-06</b>
R <sup>2</sup>	0.738	0.721	0.693	<b>0.783</b>

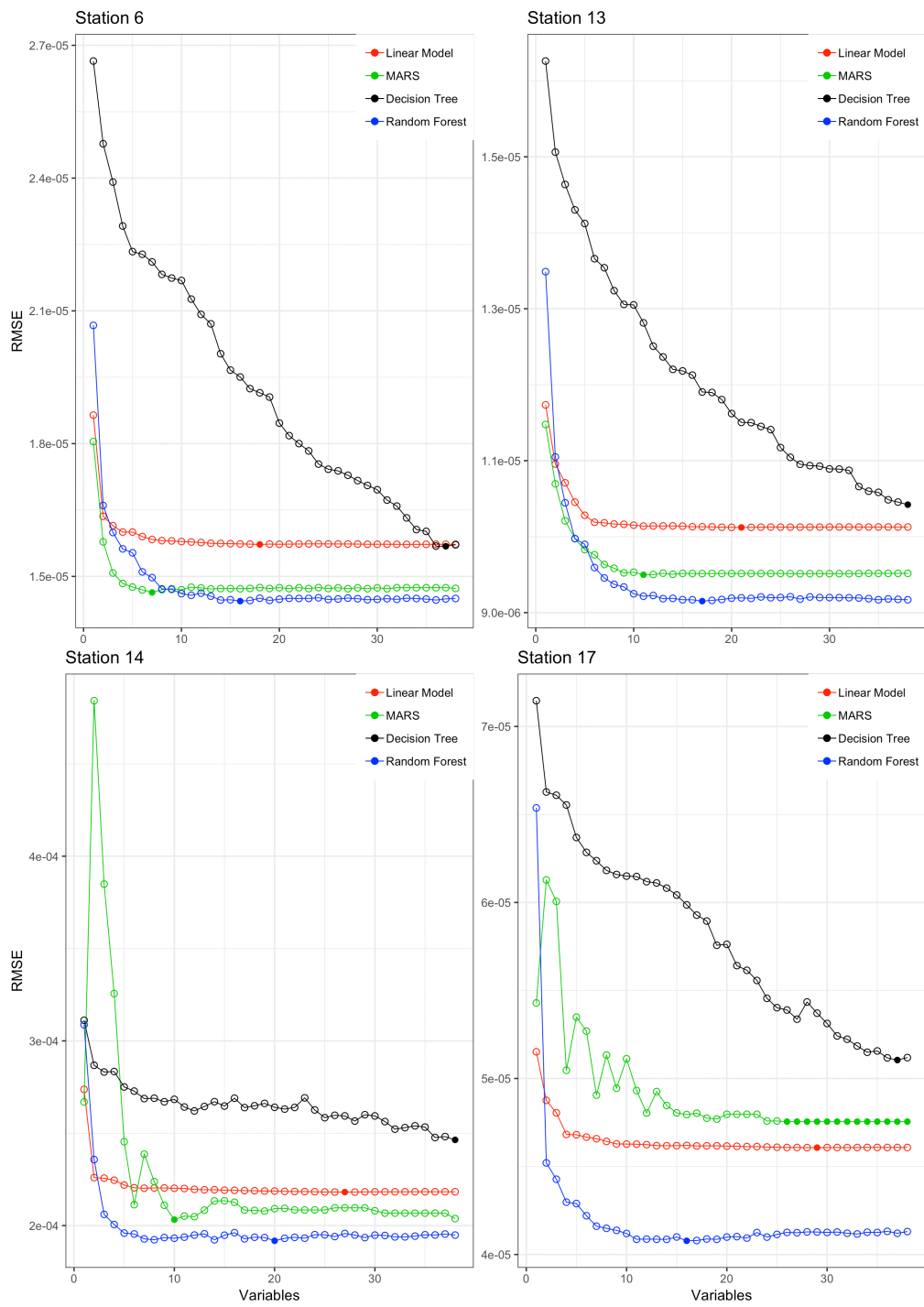


Figure A.1: Model performance by number of variables for all stations.

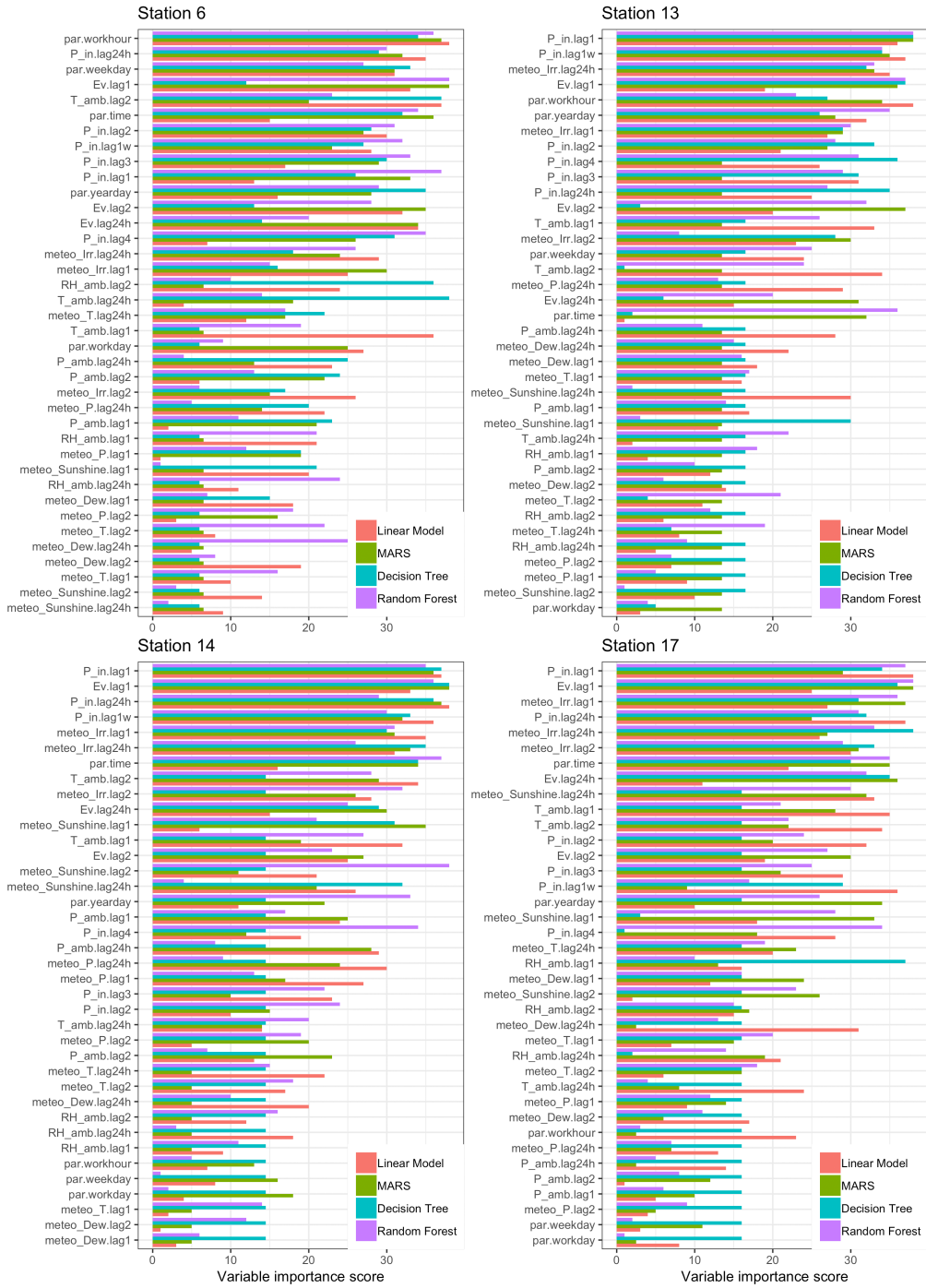


Figure A.2: Variable importance calculated using the different models on all stations.

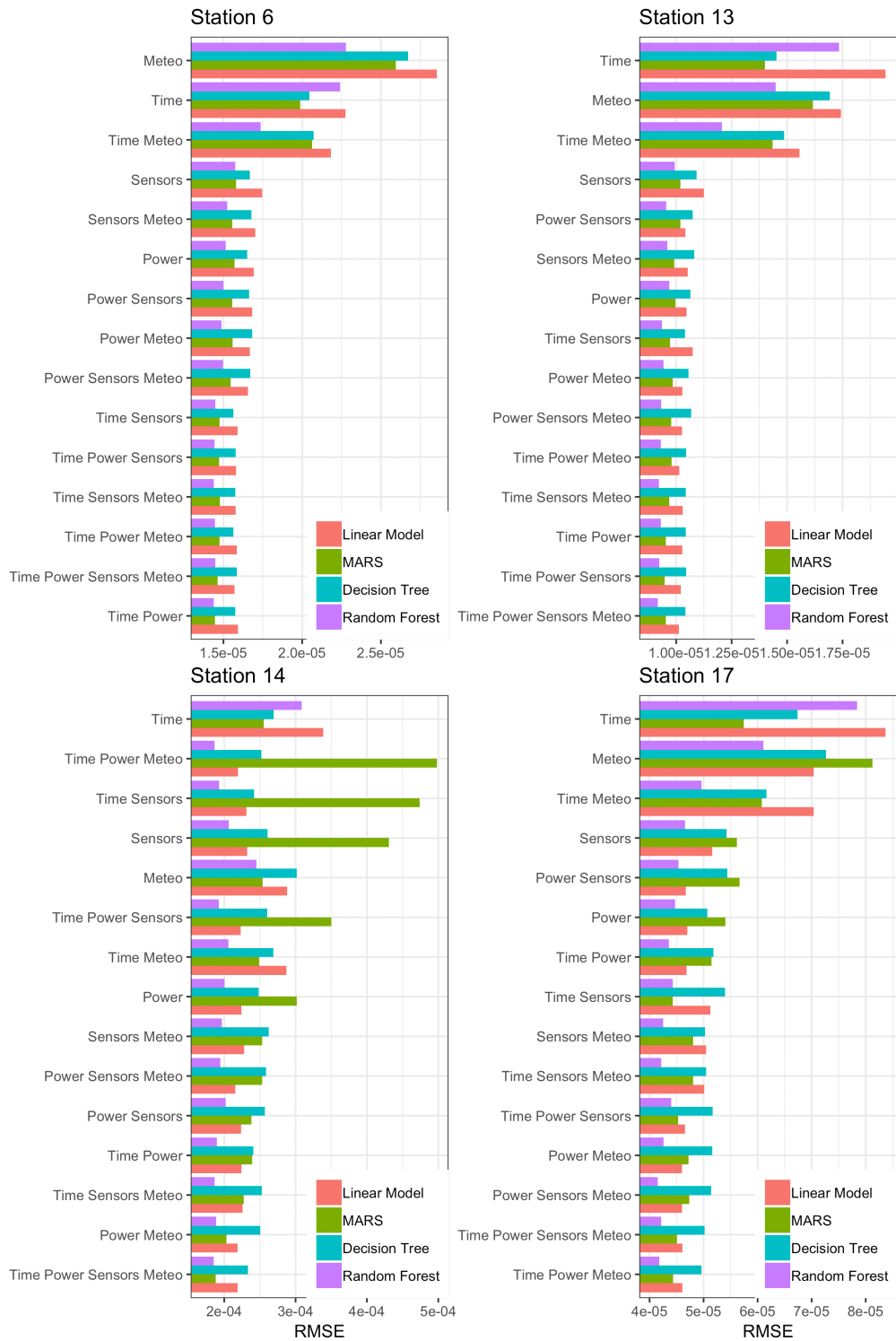


Figure A.3: Model performance by variable classes for all stations.

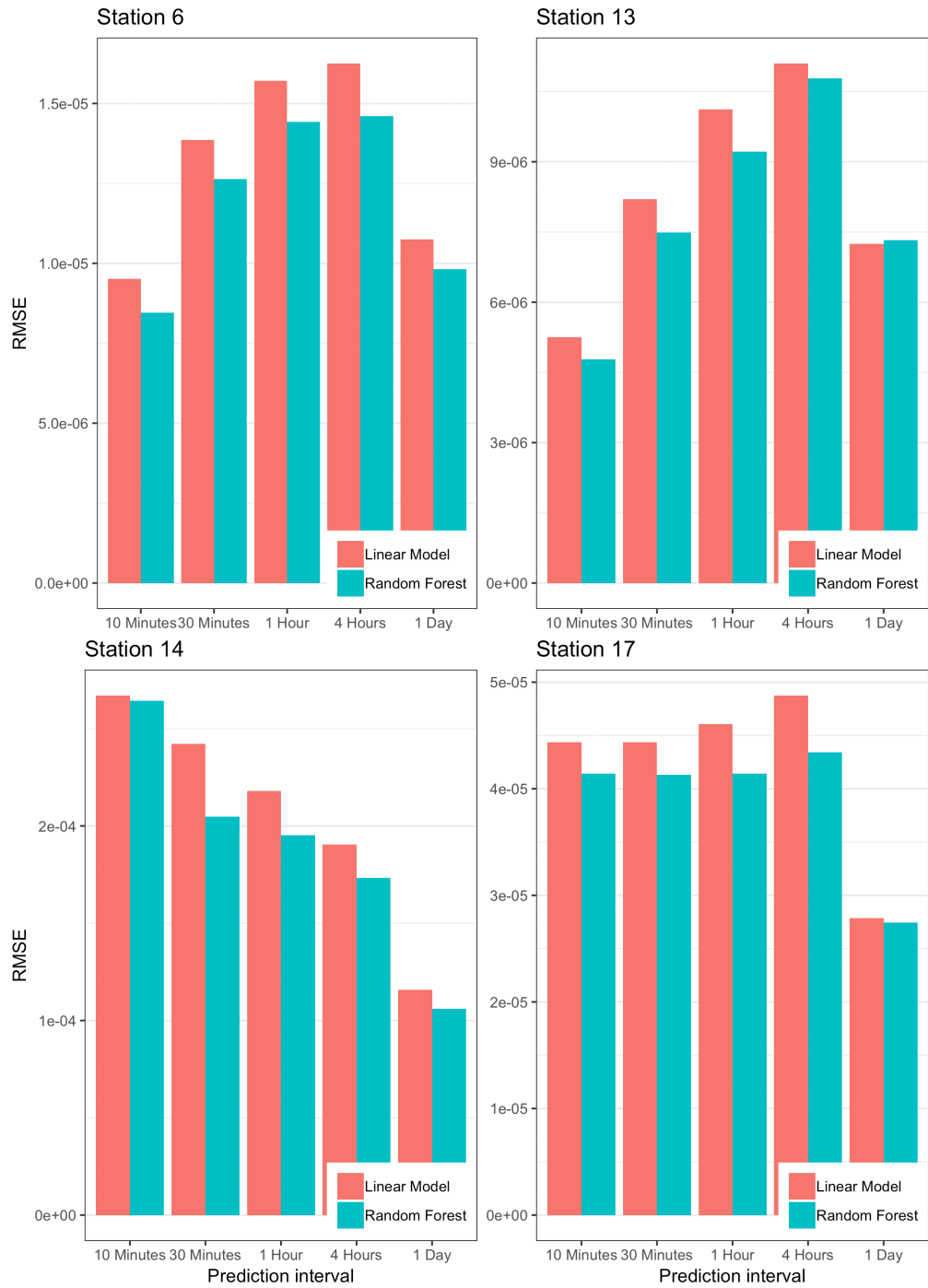


Figure A.4: Model performance by prediction interval for all stations.

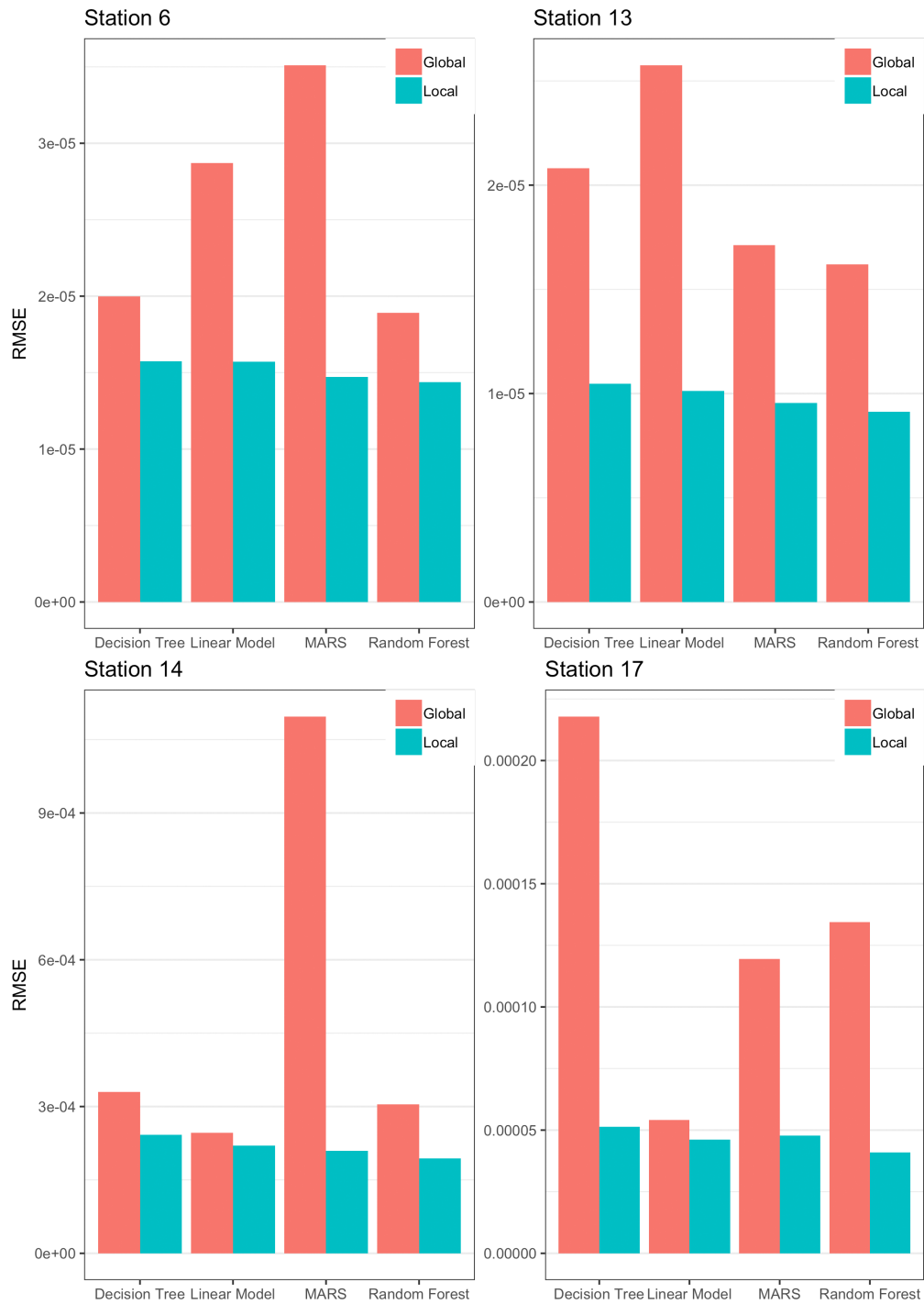


Figure A.5: Performance of the global and local models compared for all stations.



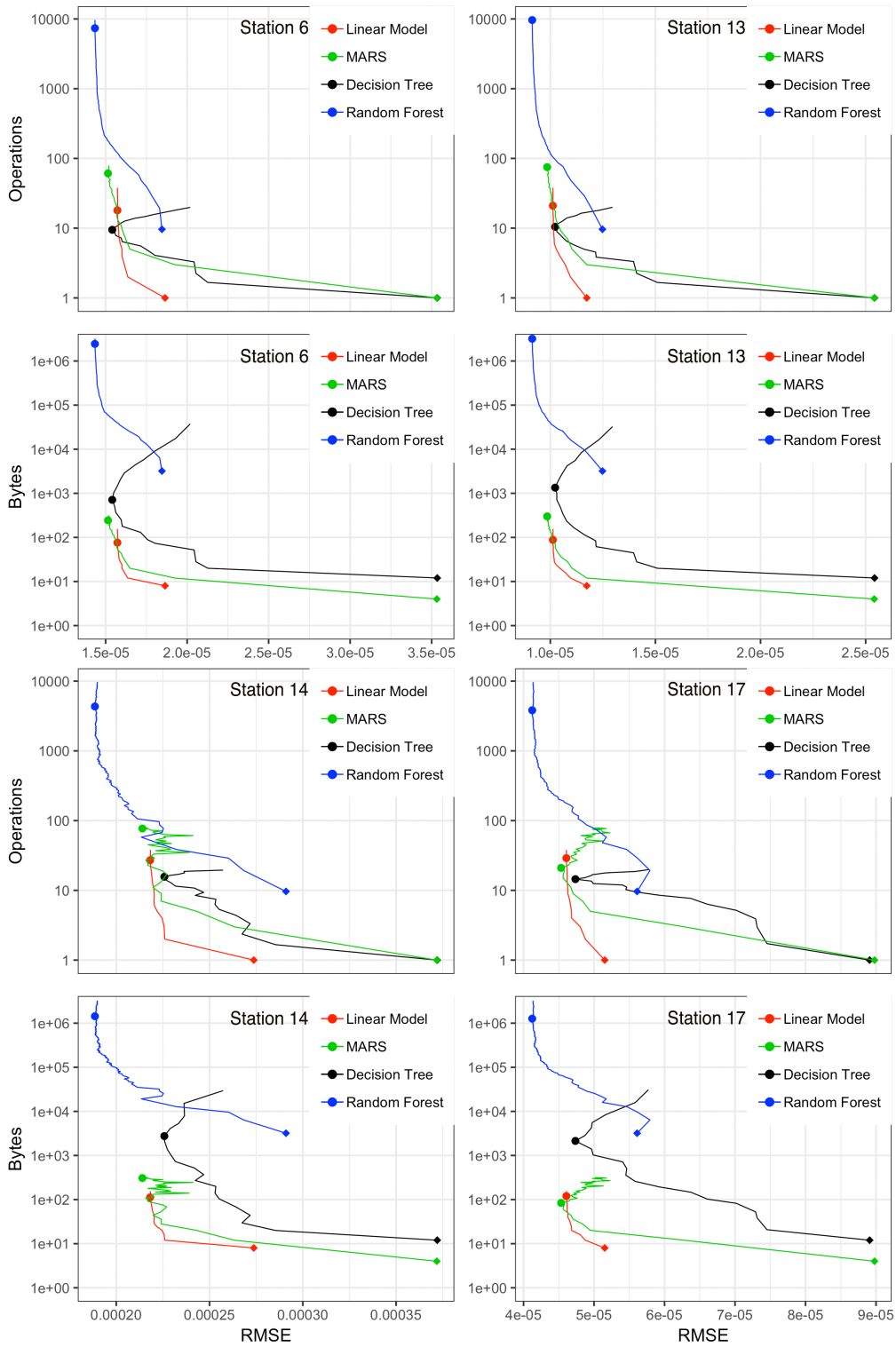


Figure A.6: Memory and runtime complexity for each model compared to their performance for all stations.

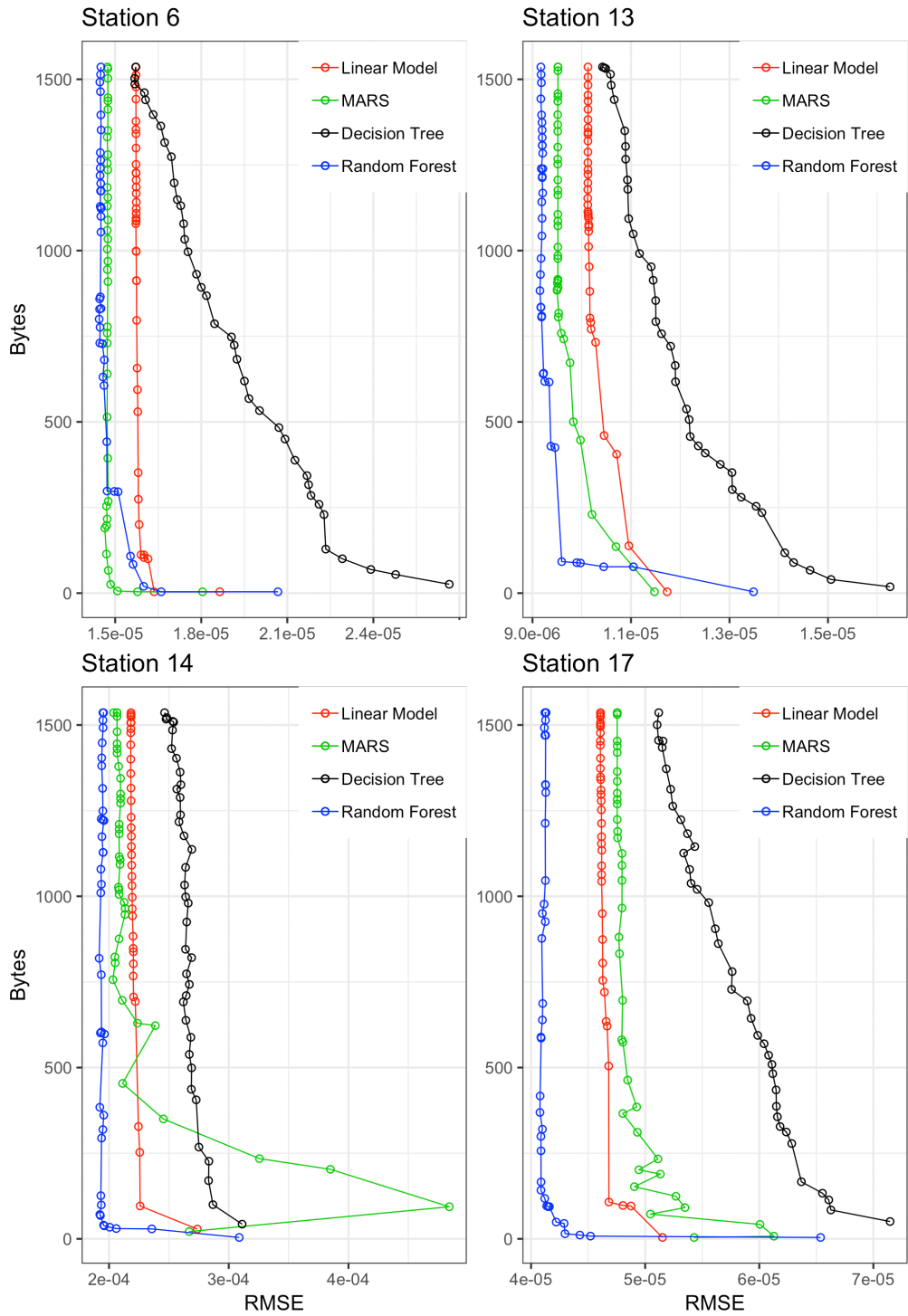


Figure A.7: Variable memory and performance of each model generated by recursive feature elimination on all stations.