Julian Gomez

Energy Prediction with Shrinkage Methods

How much energy does a building consume every day? A company specializing in energy usage, ASHRAE, collected hourly data on energy usage as well as square footing and weather factors such as wind speed, wind direction, air temperature, sea level pressure, etc. These observations come from several buildings in different sites from around the world throughout the year 2016. The goal of this project is to predict energy usage for the following two years. My predictive method consists of applying ridge regression, lasso regression, and elastic net regression to each site. Specifically, I intend to train a subset of a site's observations with all three methods, select the one that gives the lowest test error rate, and use it to predict the meter readings for the following years in that site. Once I get the predictions for each individual site, I combine them into one data set, back-transform them, and upload the results to Kaggle.

Background

A group of data sets about energy usage in 16 sites around the world consist of a training set of observations collected throughout 2016, a test set for the years 2017 and 2018, a building data set, weather training set, and weather test set. The building set consists of over 1000 observations, with a total of 1449 buildings with six variables: site id, building id, primary purpose, square footing, year built, and floor count. The weather data contains hourly measurements on air temperature (°C), dew temperature, hourly precipitation (in mm), sea level pressure (in Mlb/kPa), wind direction (in m/s), wind direction (in degrees), cloud coverage (in oktas), and site id. The training data contains the variables of building id, meter id (electricity, chilled water, steam, and hot water), timestamp, and meter reading (in kBTU for site 0 and kWh for the others). For this project, the target variable to work with is the log transformation of the meter reading variable with the following formula, since there are meter readings of 0:

$$Log\ meter\ =\ log(meter\ reading\ +\ 1).$$

To get started, I joined the datasets by their common id variables. First, I joined the building data set with the training data by building id. Later, I joined this data set with the weather data by both site id and timestamp. After that, I created a date and month variable out of the time stamp for graphing and missing data imputation purposes. With the combined data set, I grouped and summarized the quantitative variables by date, and site for some exploratory data analysis plots. The bar graph in **figure 1** shows the number of buildings in each site. It looks like that the sites with the most buildings are the ones that have the most data. In other words, site 3 has over 2 million observations as well as sites 2, 9, 13, and 14. The line graphs in **figure 2** show the log meter readings averaged by day in 2016 faceted by site ID. The plot for site 0 is different from the others since those meter readings were originally measured in different units. Some of the other plots show somewhat similar patterns in daily meter reading while sites 9 and 10 show occasional large drops. The box plots in **figure 3** show the distributions of the site transformed meter readings. In this figure, some distributions seem normal while others seem more skewed.

Methods

To get started, I handled the missing data by imputing the daily or monthly mean average of certain variables. In order to do this, I defined two functions that would conditionally replace a missing observation with an average, in this case, one by date and another by month. Next, I started splitting the combined training data into the 16 sites, 0 to 15. Then, I defined a function that would take a data frame as an argument so it would first plot the percentage of missing values and then impute the daily average for temperature, sea level pressure, wind direction, precipitation variables, wind direction and cloud coverage. Additionally, if a variable still any missing values after imputing daily mean averages, the function would then impute the monthly averages for that variable. To impute the averages, I used the ave() function, where I specified a variable with missing values, followed by a grouping variable, date or month, and one of the average functions I defined earlier. For example, to replace the air temperature

missing values, I specified the air temperature variable, the date as the grouping variable, and the daily average function I defined earlier.

With the missing values accounted for, now it is time to start determining the optimal shrinkage regression method for each site. For this step, I built a function that would take a data frame, name of site, and a vector of the quantitative variables to include in the models in order to fit the ridge, lasso, and elastic net models. The output of this function is a table of log mean square error estimates for each method as well as a statement saying which method to use. For example, I input site 1, name = 'site 1', and a vector containing square feet, air temperature, dew temperature, precipitation, etc. The result of this is the log MSE values for ridge, lasso, and elastic net as well as a sentence saying, 'Use an elastic net model.' The function starts by selecting the columns in the site data frame with respect to the terms argument. Then, the function samples 75% of the site data set for training and the remaining 25% for testing based on a seed of 1. In order to work with ridge, lasso, and elastic net regression, the function defines a model matrix with the site data set and a response variable with the log meter observations. Before fitting the ridge regression model, I used cv.glmnet with the training set to get the optimal estimate of lambda, which is part of the glmnet function as well as the training data and the alpha coefficient of 0. Next, I used the lambda estimate and the glmnet model to get the predictions on the test data in order to calculate its log MSE estimate. For lasso regression, similarly, I used cv.glmnet() to find the lambda estimate for glmnet, fit the model with the training data and an alpha of 1, and predicted with the same lambda on the test data to get the model's log MSE. For elastic net, I used cross validation to find not only an optimal value of lambda for glmnet, but also an optimal value of alpha. For this, I used the trainControl function from the caret package, where I specified the cross validation method and the number of folds, 10 since this number is also the default number of folds in cv.glmnet. After this, I used the train() function, also from the caret package, with the training data, glmnet method, the default tuneLength of 3, and the trainControl() item to find the optimal value of alpha and lambda. Like lasso and ridge

regression, I fit the elastic net model with glmnet with the optimal values of alpha and lambda from the previous step. And then, I predicted on the test data with this model to get the log MSE. For the output of the defined function, a table that collects the three log MSE estimates as well as if-then statements that would print out which of the three methods is best as well as its lambda coefficients.

After receiving the optimal shrinkage prediction methods for each site, I proceeded to work with the missing values in the test data the same way as the training data. To do this, I split the combined test dataset into the 16 sites and defined another function to impute both the mean daily and monthly average on the quantitative predictors. Since the test data set is twice as large as the training set, the function is set to add the date and month variables at the beginning.

After imputing missing values of the test datasets, I started to predict the log meter reading for the test data with respect to the results of **table 1**. To do the predictions, I defined three functions: one that would predict with elastic net regression, another with lasso regression, and another with ridge regression. Each function takes four arguments: the site training data, the site test data, the training data variables, and the test data variables. The variables for the test data would be the same as those of the training data except the log meter reading since the test data does not include it. The elastic net prediction function starts by creating a model matrix with the training data and convert the test data into a matrix. The trainControl() and train() functions from the caret package use the training data to determine the optimal values of lambda and alpha through 10-fold cross validation. Then, the glmnet() would use the alpha and lambda values as well as the training data matrix in order to compute the predictions on the test data matrix. In the end, the function binds the predictions to the site test data. The lasso and ridge prediction functions have the same arguments as that of the elastic net function. They perform the same steps as that of the elastic net function except the part where they compute the optimal lambda parameters for the glmnet() model with cv.glmnet() with 10 as the default

number of folds. Then, both functions fit the parameters with the site training data matrices to predict the log meter reading with the test matrices. Once I am done with the predictions, I row-binded the site datasets, selected the row id and prediction values, and back transformed them to their original scale before exporting to csv.

Results

The missing value plot in **Figure 4a** shows the percentage of missing values for each variable by site and **figure 4b** shows the percentage of missing values after doing my imputations. We can see from this plot that the year built is mostly missing in 8 of the sites, floor count in 9 sites, cloud coverage in two, precipitation in three, and sea level pressure in one. To do my predictions, I would not use a variable that is completely empty for one site, since I would be unable to get any averages to replace those missing values. The plots in **figure 5a)-o)** show plots of the percentage of missing values by site. From these plots, floor count will be a predictor in sites 1, 4, 5, 7, 8, and 10. However, precipitation depth will not be a predictor in sites 1, 5, and 12. Cloud coverage will not be included in the models for sites 7 and 11. And finally, sea level pressure will not be part of the site 5 model.

The results in **table 1** show each site's log mean square errors from the three shrinkage regression methods as well as their optimal prediction method. Site 10 has ridge regression from a log MSE of 4.398709. Sites 5, 7, and 13 have lasso regression with log MSEs of 1.010425, 9.764676, and 5.525768 respectively. And additionally, the remaining 14 sites' lowest log MSEs correspond to elastic net regression. Therefore, the final models for meter reading predictions are ridge regression for site 10, lasso regression for sites 5, 7, and 13, elastic net for the others. It seems interesting that any value for alpha for elastic net can outperform a ridge or lasso regression, even when some differences with the other log MSEs are not significant. Besides the mean squared errors, **table 2** contains the alpha and lambda parameters used for each prediction. All of the predictive models used the same alpha values as those shown in **table 1**, except site 8, which used a value of 1 to determine the optimal prediction method and

0.55 to predict the log meter reading. **Table 3** contains the model coefficients used for each prediction as well as which variables were used. Most of the coefficients were rounded to four decimal places, while most of the square footing coefficients were rounded to six. After I joined and back transformed my predictions, I managed to get a score of 1.89 in the Kaggle leaderboard.

Conclusion/Future Work

Predicting meter reading by site seems quite reasonable since each site is a different place in the world and **figure 2** and **figure 3** show somewhat different meter reading patterns by site. The three methods of ridge regression, lasso regression, and elastic net regression proved to predict reasonable observations for over 40 million observations over 2017 and 2018 since they took a while to cross-validate, fit, and predict. In addition to the methods applied in this project, I could also create additional variables that could possibly affect energy usage, such as relative humidity, heat index, wind chill index, etc. Another thing could also be to learn more about missing data methods such as the ones from the MICE package. Such methods might give me better imputations for the weather variables and even fill some of the empty year built and floor count variables shown in **Figure 3b**. Another thing that might be interesting is apply Keras regression here, since I started to find very interesting recently. This project was about applying any statistical predictive modeling method, and I chose one of my favorite predictive topics. Therefore, I look forward to learning even more about predictive modeling and apply my skills in projects like this one.

APPENDIX

Figure 1 - Number of buildings by site

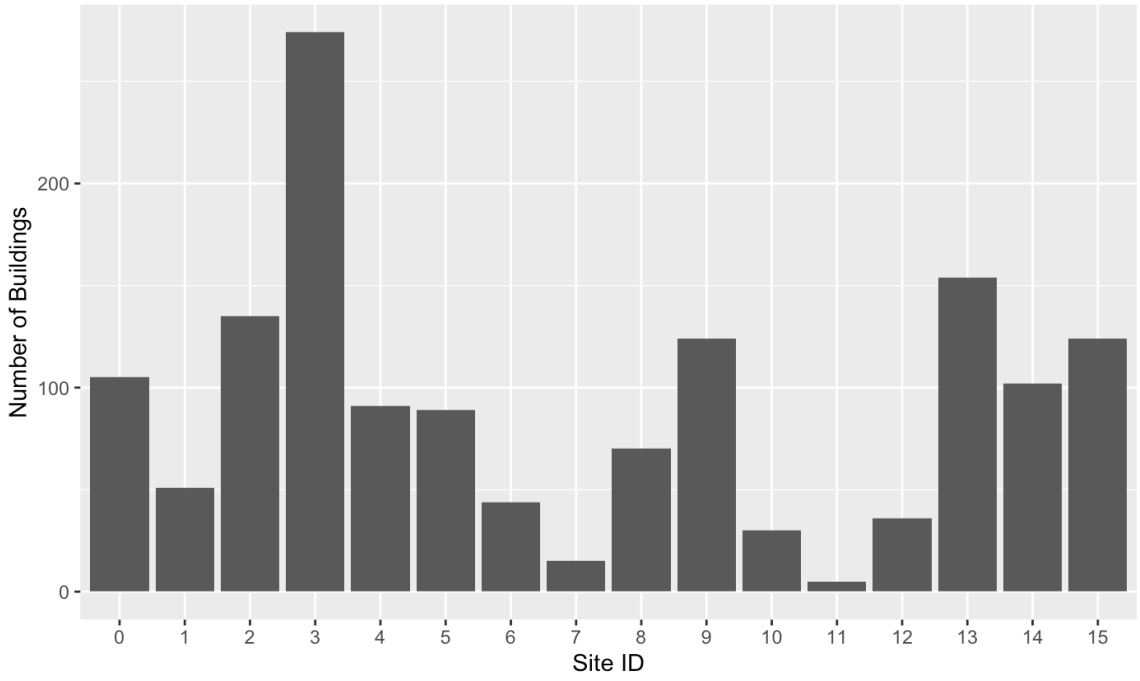


Figure 2 - Average daily meter readings in the log scale by site ID

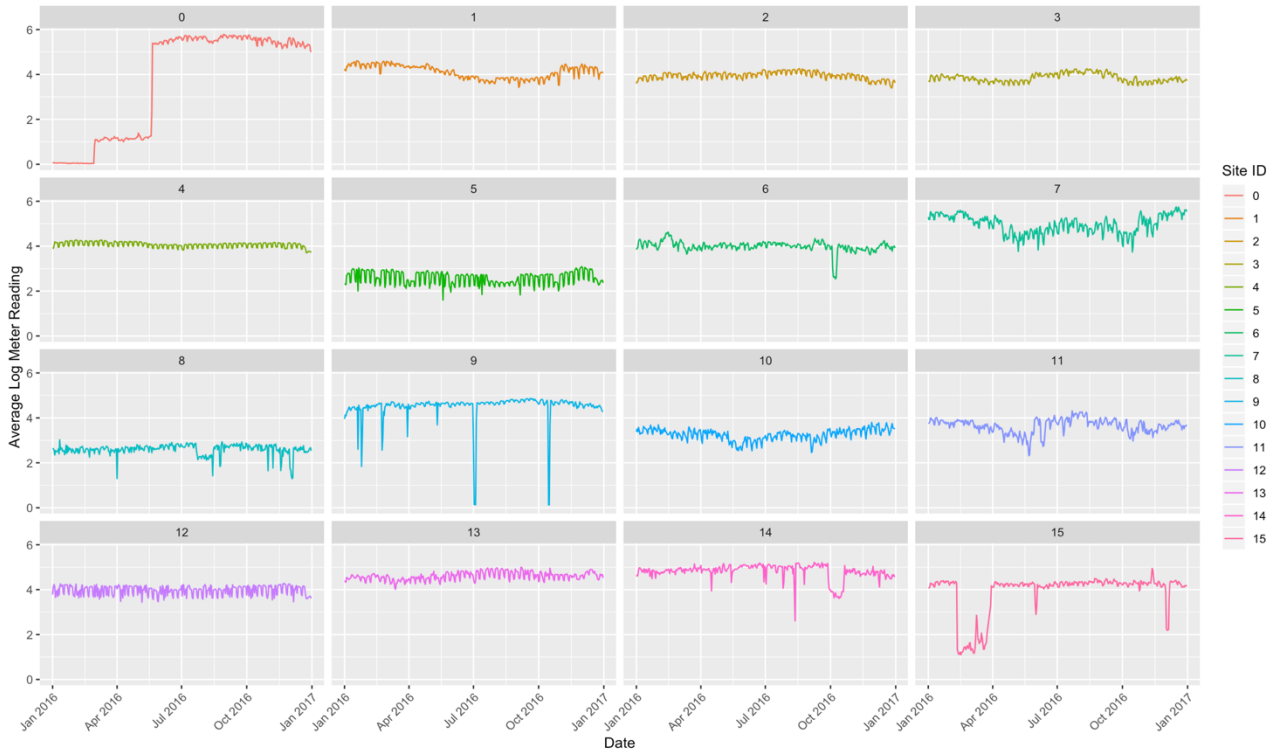


Figure 3 - Boxplot distribution of log meter reading by site

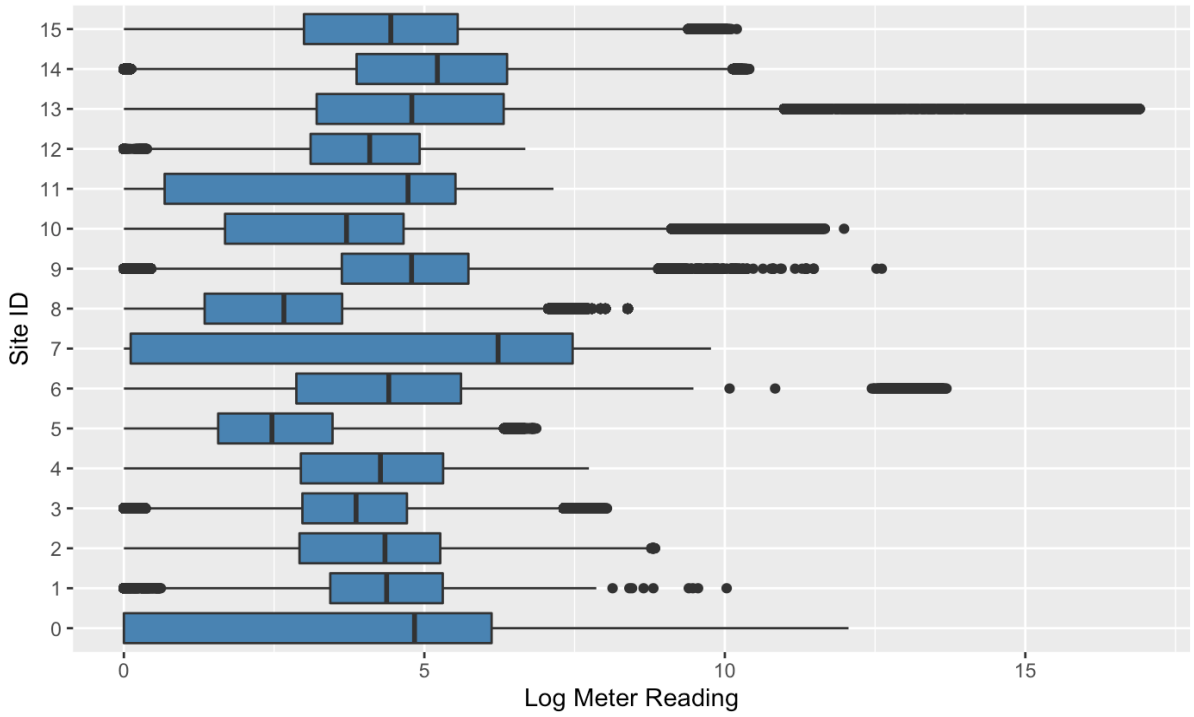


Figure 4a) - Missing data plot of the combined training set by site ID without imputations

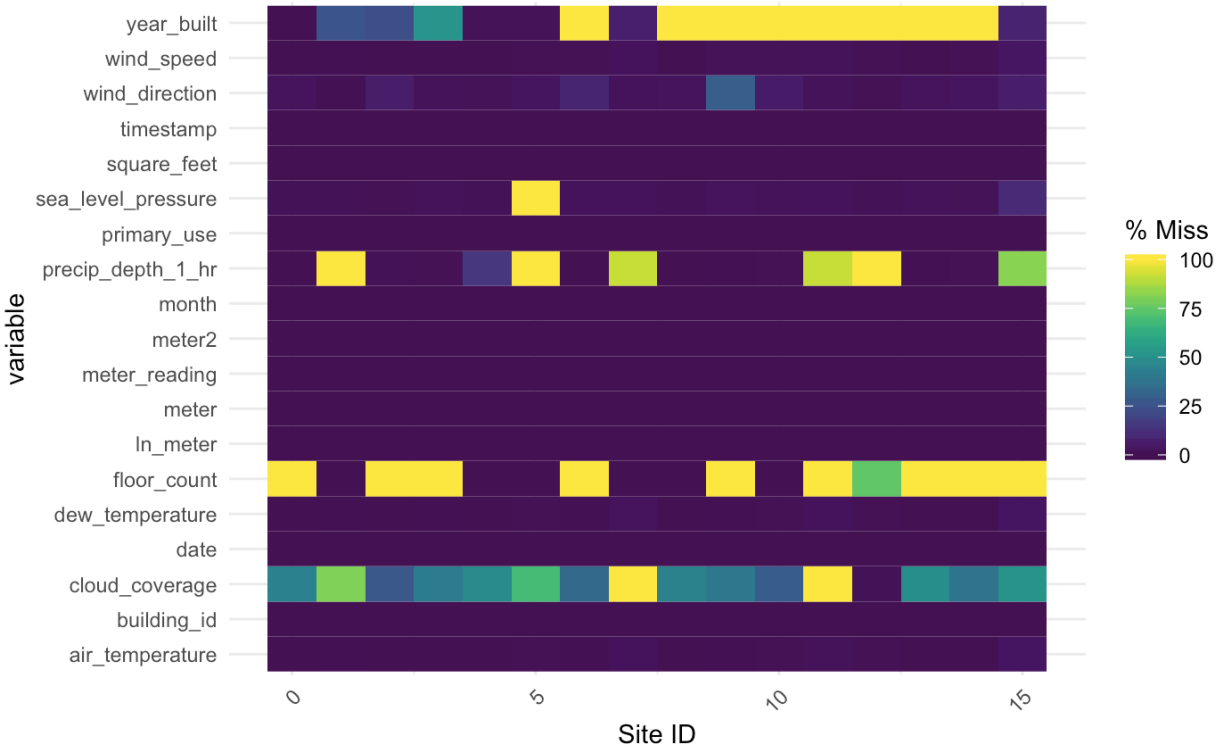


Figure 4b) - Missing data plot with daily and monthly average imputations by site ID

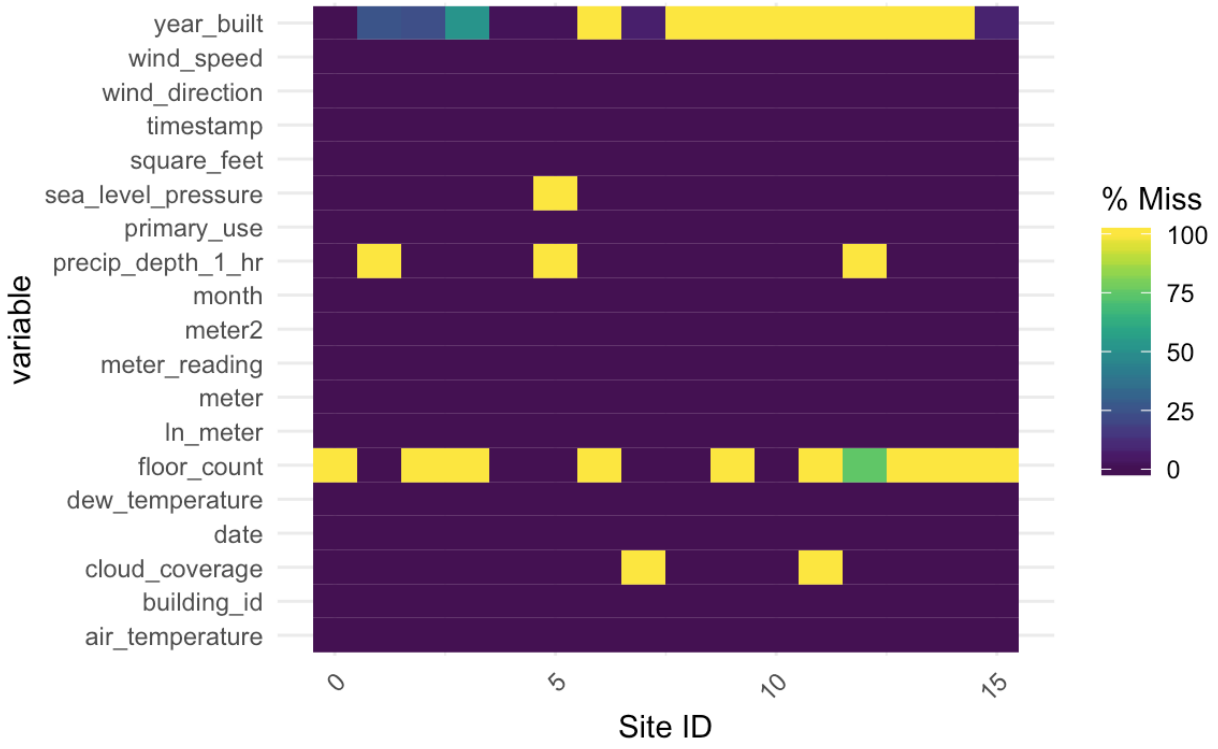
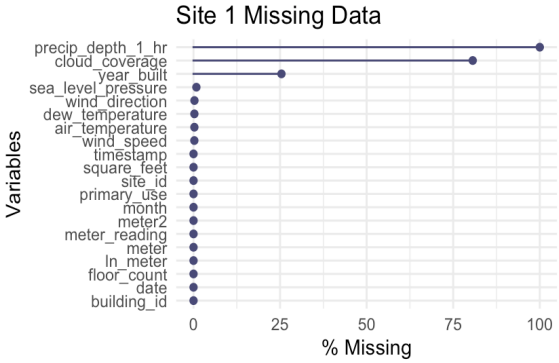


Figure 5 - Percentage of missing data by site ID

a) Site 0

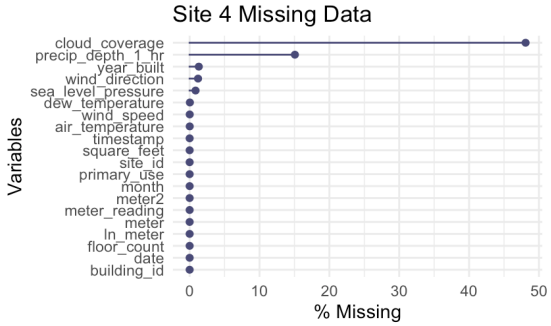
b) Site 1



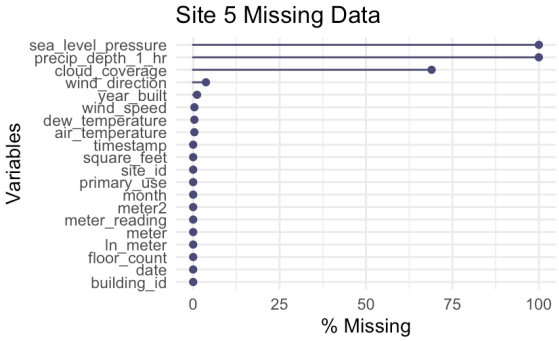
b) Site 2

c) Site 3

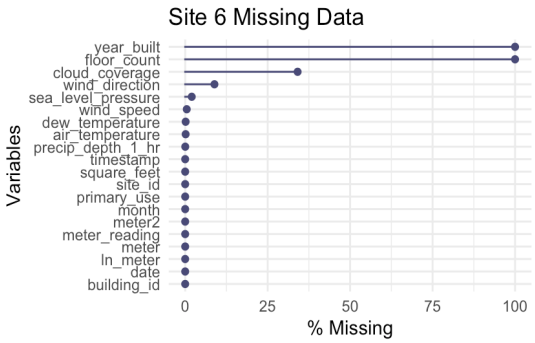
d) Site 4



e) Site 5



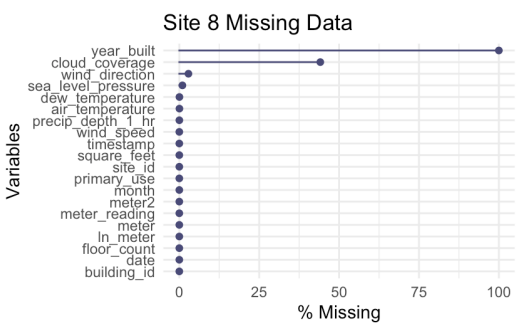
f) Site 6



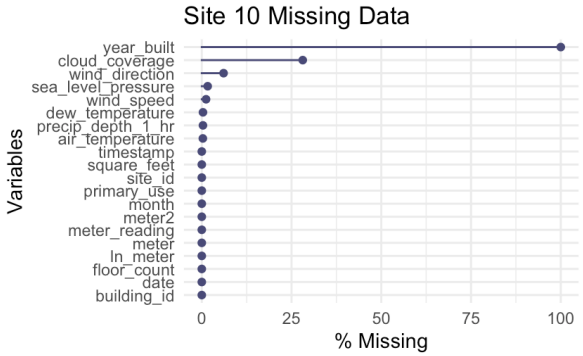
g) Site 7

h) site 8

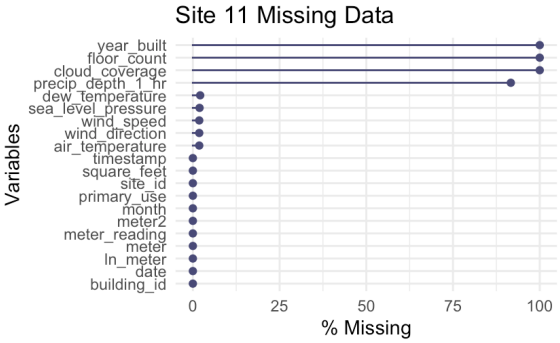
i) site 9



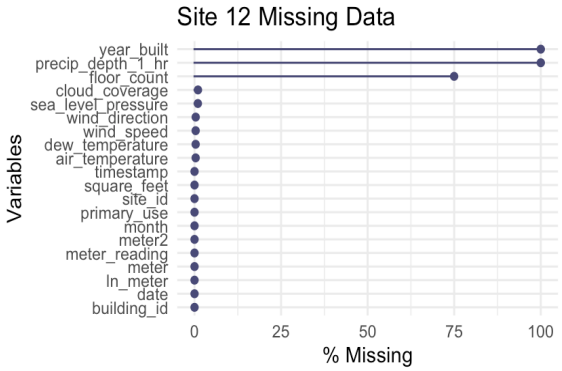
j) Site 10



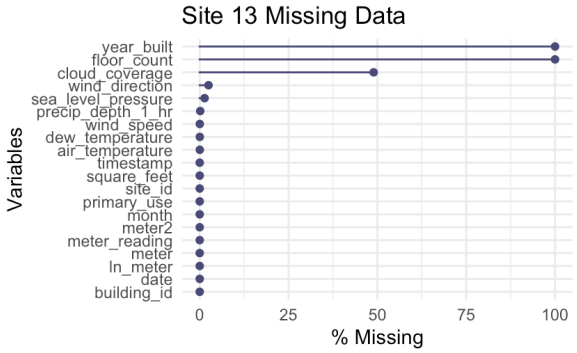
k) Site 11



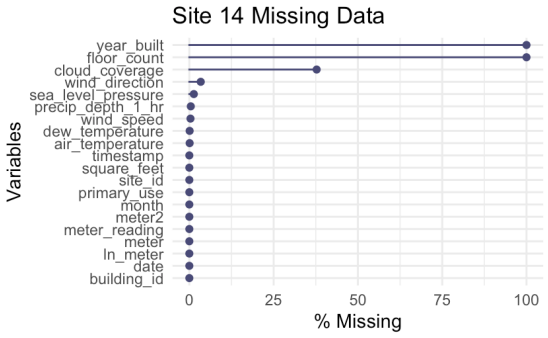
l) Site 12



m) Site 13



n) Site 14



o) Site 15

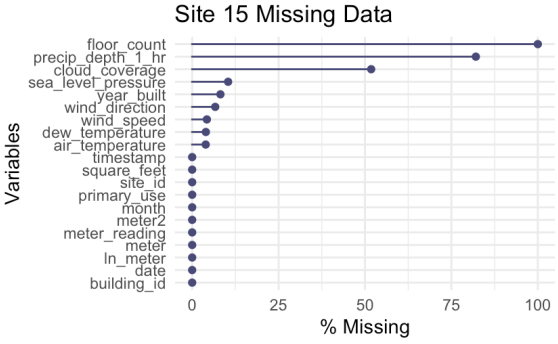


Table 1 - Results of Log Mean Squared Errors for each individual site. The optimal shrinkage method for each site was used to predict their respective energy usage.

Site ID	Ridge Log MSE	Lasso Log MSE	Elastic Net Log MSE	Optimal Method	Alpha	Lambda
0	6.59186	6.584918	6.584815	Elastic Net	0.1	0.0026
1	2.32807	2.327088	2.327084	Elastic Net	1	0.0015
2	3.365343	3.36518	33.365176	Elastic Net	1	0.0012
3	1.212749	1.210229	11.210181	Elastic Net	0.1	0.0017
4	1.232391	1.229118	1.229016	Elastic Net	0.1	0.0025
5	1.014243	1.010425	1.010426	Lasso	1	0.001
6	5.30177	5.301293	5.301275	Elastic Net	1	0.0012
7	9.764821	9.764676	9.764718	Lasso	1	0.0005
8	2.12055	2.119522	2.119514	Elastic Net	1	0.0015
9	2.53173	2.530696	2.530677	Elastic Net	0.55	0.0016
10	4.398709	4.398803	4.398785	Ridge	0	0.0447
11	5.384148	5.383582	5.383556	Elastic Net	0.1	0.0006
12	0.862517	0.8585468	0.8585151	Elastic Net	0.1	0.0016
13	5.527642	5.525768	5.525789	Lasso	1	0.0009
14	4.764875	4.763352	4.763335	Elastic Net	0.1	0.0015
15	3.280038	3.279709	3.279706	Elastic Net	0.1	0.0026

Table 2- list of alpha and lambda parameters used to predict log meter reading by site

Site ID	Prediction Model	Alpha	Lambda
0	Elastic Net	0.1	0.0026
1	Elastic Net	1	0.0015
2	Elastic Net	1	0.0012
3	Elastic Net	0.1	0.0017
4	Elastic Net	0.1	0.0025
5	Lasso	1	0.001
6	Elastic Net	1	0.0012
7	Lasso	1	0.0005
8	Elastic Net	0.55	0.0015
9	Elastic Net	0.55	0.0016
10	Ridge	0	0.0445
11	Elastic Net	0.1	0.0006
12	Elastic Net	0.1	0.0016
13	Lasso	1	0.0009
14	Elastic Net	0.1	0.0015
15	Elastic Net	0.1	0.0012

Table 3 - Prediction model equations by site ID and methods

Site ID	Regression	Model Equation
0	Elastic Net	$f(x) = -103.5441 + 0.000002sq_{ft} + 0.0789air_t - 0.1795cloud_{cov} + 0.1985dew_t + 0.0004precip + 0.1007sea_{pr} + 0.0012wind_{dir} - 0.0149wind_{sp}$
1	Elastic Net	$f(x) = 6.669 + 0.00002sq_{ft} - 0.004air_t + 0.024floor_{ct} + 0.0057cloud_{cov} - 0.038dew_t - 0.0035sea_{pr} - 0.00015wind_{dir} + 0.027wind_{sp}$
2	Elastic Net	$f(x) = 4.704 + 0.000006sq_{ft} - 0.0014air_t - 0.0141cloud_{cov} - 0.012dew_t - 0.0018precip - 0.0013sea_{pr} - 0.0005wind_{dir} + 0.0069wind_{sp}$
3	Elastic Net	$f(x) = 1.58 + 0.0000073sq_{ft} + 0.007air_t - 0.001cloud_{cov} + 0.0024dew_t - 0.0003precip + 0.002sea_{pr} + 0.00013wind_{dir} - 0.0024wind_{sp}$
4	Elastic Net	$f(x) = 2.7857 + 0.000008sq_{ft} - 0.2954floor_{ct} - 0.013air_t - 0.01cloud_{cov} + 0.0115dew_t - 0.0007precip - 0.009sea_{pr} - 0.0002wind_{dir} - 0.014wind_{sp}$
5	Lasso	$f(x) = 1.5678 + 0.000015sq_{ft} + 0.0303air_t + 0.0861floor_{ct} + 0.0238cloud_{cov} - 0.0373dew_t - 0.0003wind_{dir} + 0.0221wind_{dir}$
6	Elastic Net	$f(x) = 4.756 + 0.000009sq_{ft} - 0.0034air_t + 0.0054cloud_{cov} = 0.00002dew_t - 0.0004precip - 0.0014sea_{pr} - 0.00002wind_{dir} - 0.0043wind_{sp}$
7	Lasso	$f(x) = 6.878 + 0.000002sq_{ft} - 0.0705floor_{ct} - 0.0447air_t + 0.0242dew_t + 0.0052precip - 0.0017sea_{pr} + 0.0001wind_{dir} + 0.0043wind_{sp}$
8	Elastic Net	$f(x) = 1.322 + 0.000007sq_{ft} + 0.4488floor_{ct} + 0.0019air_{temp} - 0.0014cloud_{cov} + 0.0047dew_t + 0.0005sea_{pr} + 0.000006wind_{dir} - 0.00026wind_{sp}$
9	Elastic Net	$f(x) = -1.883 + 0.000006sq_{ft} + 0.002air_t + 0.0238cloud_{cov} + 0.0055dew_t + 0.0013precip + 0.0053sea_{pr} - 0.00003wind_{dir} - 0.0373wind_{sp}$
10	Ridge	$f(x) = -0.2482 + 0.000008sq_{ft} - 0.2206floor_{ct} - 0.0126air_t + 0.001cloud_{cov} - 0.00127dew_t - 0.0021precip + 0.0035sea_{pr} - 0.0003wind_{dir} + 0.0146wind\_sp$
11	Elastic Net	$f(x) = -2.0809 + 0.00001 - 0.0144air_t + 0.0141dew_t + 0.0074precip + 0.0045sea_{pr} - 0.000003wind_{dir} + 0.0193wind_{sp}$
12	Elastic Net	$f(x) = 3.1854 + 0.00001 + 0.0328air_t + 0.0074cloud_{cov} - 0.0339dew_t - 0.0003sea_{pr} - 0.00012 + 0.0101wind_{sp}$
13	Lasso	$f(x) = -1.7344 + 0.000007sq_{ft} - 0.0117air_t - 0.0083cloud_{cov} + 0.0178dew_t - 0.0006precip + 0.0053sea_{pr} + 0.00006wind_{dir} + 0.0073wind_{sp}$
14	Elastic Net	$f(x) = 12.1672 + 0.000009 + 0.0293air_t + 0.0039cloud_{cov} - 0.0231dew_t + 0.0029precip - 0.0084sea_{pr} - 0.00015wind_{dir} + 0.0064wind_{sp}$
15	Elastic Net	$f(x) = 1.639 + 0.000009sq_{ft} - 0.001air_t + 0.0087cloud_{cov} + 0.0009dew_t + 0.0015precip + 0.0018sea_{pr} - 0.00003wind_{dir} + 0.0023wind_{sp}$

Figure 7 – My Kaggle submission score

Kaggle Project

Julian Gomez

11/5/2019

```
library(tidyverse)
library(lubridate)
library(leaps)
library(randomForest)
library(glmnet)
library(Hmisc)
library(pls)
library(caret)
library(naniar)
library(data.table)

buildings <- read.csv('~/Documents/STAT 488/ashrae-energy-prediction/building_metadata.csv')
b_train <- read.csv('~/Documents/STAT 488/ashrae-energy-prediction/train.csv')

b_train$ln_meter <- log(b_train$meter_reading + 1)

w_train <- read.csv('~/Documents/STAT 488/ashrae-energy-prediction/weather_train.csv')


b_test <- read.csv('~/Documents/STAT 488/ashrae-energy-prediction/test.csv')
w_test <- read.csv('~/Documents/STAT 488/ashrae-energy-prediction/weather_test.csv')
```

```
## combine training data with weather data
b_train2 <- b_train %>% left_join(buildings, by = 'building_id')
rm(b_train)
b_train3 <- b_train2 %>% left_join(w_train, by = c('site_id', 'timestamp'))
rm(b_train2)
## Create date
b_train3$date <- as.Date(b_train3$timestamp)
b_train3$month <- format(as.Date(b_train3$timestamp), '%Y-%m')
b_train3 <- b_train3 %>% mutate(meter2 = case_when(
    meter == 0 ~ 'electricity',
    meter == 1 ~ 'chilled water',
    meter == 2 ~ 'steam',
    meter == 3~'hot water'
  ))
```

```
b_test2 <- b_test %>% left_join(buildings, by = 'building_id')
rm(b_test)
b_test3 <- b_test2 %>% left_join(w_test, by = c('site_id', 'timestamp'))
rm(b_test2)
```

```
## create weather summary by day
weather_sum <- b_train3 %>% group_by(site_id,date) %>%
  summarise(avg_air_temperature = mean(air_temperature, na.rm = TRUE),
            avg_dew_temperature = mean(dew_temperature, na.rm = TRUE),
            avg_sea_level_presssure = mean(sea_level_pressure, na.rm = TRUE),
            avg_cloud_coverage = mean(cloud_coverage, na.rm = TRUE),
```

```
            avg_precipitation_depth = mean(precip_depth_1_hr, na.rm = TRUE),
            avg_wind_speed = mean(wind_speed, na.rm = TRUE),
            avg_ln_meter_reading = mean(ln_meter))
```

```
## building data
buildings %>% ggplot() +
  geom_bar(aes(x = primary_use)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

buildings %>% ggplot() +
  geom_bar(aes(x = floor_count), na.rm = TRUE) + xlab('Floor Count')

buildings %>% ggplot() +
  geom_histogram(aes(x = square_feet)) +
  xlab('Square Feet')

## how many buildings on sites
lo <- b_train3 %>%
  distinct(site_id, building_id) %>%
  group_by(site_id) %>%
  summarise(number_of_buildings = n())

lo %>% ggplot() +
  geom_bar(aes(x = factor(site_id), y = number_of_buildings), stat = 'identity') +
  xlab('Site ID') + ylab('Number of Buildings')

## training weather data

weather_sum %>% ggplot(aes(x = date, y = avg_ln_meter_reading)) +
  geom_line(aes(col = factor(site_id))) +
  facet_wrap(~factor(site_id)) +
  labs(col = "Site ID") + xlab('Date') +
  ylab('Average Log Meter Reading') +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

weather_sum %>% ggplot(aes(x = date, y = avg_air_temperature)) +
  geom_line(aes(col = factor(site_id))) +
  labs(col = "Site ID") + xlab('Date') +
  ylab('Average Daily Air Temperature')

weather_sum %>% ggplot(aes(x = date, y = avg_dew_temperature)) +
  geom_line(aes(col = factor(site_id))) + labs(col = "Site ID") +
  xlab('Date') + ylab('Average Daily Dew Temperature')

weather_sum %>% ggplot(aes(x = date, y = avg_sea_level_presssure)) +
  geom_line(aes(col = factor(site_id))) +
  labs(col = "Site ID") +
  xlab('Date') +
  ylab('Average Daily Sea Level Pressure')

weather_sum %>% ggplot(aes(x = date, y = avg_cloud_coverage)) +
  geom_line(aes(col = factor(site_id))) +
  labs(col = "Site ID") +
  xlab('Date') + ylab('Average Daily Cloud Coverage')
```

```r
weather_sum %>% ggplot(aes(x = date, y = avg_precipitation_depth)) +
  geom_line(aes(col = factor(site_id))) +
  labs(col = "Site ID") + xlab('Date') +
  ylab('Average Daily Precipitation (mm)')

weather_sum %>% ggplot(aes(x = date, y = avg_wind_speed)) +
  geom_line(aes(col = factor(site_id))) +
  labs(col = "Site ID") + xlab('Date') + ylab('Average Daily Wind Speed')




## boxplots: meter readings
b_train3 %>% ggplot() +
  geom_boxplot(aes(x = factor(site_id), y = ln_meter), fill = 'steel blue') +
  coord_flip() + xlab('Site ID') + ylab('Log Meter Reading')


## temperature by site
weather_sum %>% ggplot() +
  geom_boxplot(aes(x = factor(site_id), y = avg_air_temperature), fill = 'steel blue') +
  coord_flip() + xlab('Site ID') + ylab('Average Daily Air Temperature')

## temperature by site
b_train3 %>% ggplot() +
  geom_boxplot(aes(x = factor(site_id), y = air_temperature), fill = 'steel blue') +
  coord_flip() + xlab('Site ID') + ylab('Air Temperature')

## missing data
gg_miss_var(b_train3)

gg_miss_fct(x = b_train3, fct = site_id) + xlab('Site ID')
```

```r
## impute missing values by site

## function that will calculate daily averages for missing values
daily_avg <- function(x) ifelse(is.na(x), mean(x, na.rm = TRUE), x)

## function that will calculate monthly averages for missing values
monthly_avg <- function(x) ifelse(is.na(x), mean(x, na.rm = TRUE), x)


## site 0
site0_tr <- b_train3 %>% filter(site_id == 0)

## site 1
site1_tr <- b_train3 %>% filter(site_id == 1)

## site 2
site2_tr <- b_train3 %>% filter(site_id == 2)

## site 3
site3_tr <- b_train3 %>% filter(site_id == 3)
```

```
## site 4
site4_tr <- b_train3 %>% filter(site_id == 4)

## site 5
site5_tr <- b_train3 %>% filter(site_id == 5)

## site 6
site6_tr <- b_train3 %>% filter(site_id == 6)

## site 7
site7_tr <- b_train3 %>% filter(site_id == 7)

## site 8
site8_tr <- b_train3 %>% filter(site_id == 8)

## site 9
site9_tr <- b_train3 %>% filter(site_id == 9)

## site 10
site10_tr <- b_train3 %>% filter(site_id == 10)

## site 11
site11_tr <- b_train3 %>% filter(site_id == 11)

## site 12
site12_tr <- b_train3 %>% filter(site_id == 12)

## site 13
site13_tr <- b_train3 %>% filter(site_id == 13)

## site 14
site14_tr <- b_train3 %>% filter(site_id == 14)

## site 15
site15_tr <- b_train3 %>% filter(site_id == 15)

site_impute <- function(df, title){

  print(gg_miss_var(df, show_pct = TRUE) + ggtitle(title))

  df$air_temperature <- ave(df$air_temperature, df$date, FUN = daily_avg)

  df$air_temperature <- ave(df$air_temperature, df$month, FUN = monthly_avg)

  df$dew_temperature <- ave(df$dew_temperature, df$date, FUN = daily_avg)

  df$dew_temperature <- ave(df$dew_temperature, df$month, FUN = monthly_avg)

  df$precip_depth_1_hr <- ave(df$precip_depth_1_hr, df$date, FUN = daily_avg)

  df$precip_depth_1_hr <- ave(df$precip_depth_1_hr, df$month, FUN = monthly_avg)

  df$wind_speed <- ave(df$wind_speed, df$date, FUN = daily_avg)
```

```
  df$wind_speed <- ave(df$wind_speed, df$month, FUN = monthly_avg)

  df$sea_level_pressure <- ave(df$sea_level_pressure, df$date, FUN = daily_avg)

  df$sea_level_pressure <- ave(df$sea_level_pressure, df$month, FUN = monthly_avg)

  df$wind_direction <- ave(df$wind_direction, df$date, FUN = daily_avg)

  df$wind_direction <- ave(df$wind_direction, df$month, FUN = monthly_avg)

  df$cloud_coverage <- ave(df$cloud_coverage, df$date, FUN = daily_avg)

  df$cloud_coverage <- ave(df$cloud_coverage, df$month, FUN = monthly_avg)

  df
}

## impute the averages according the the function

site0_tr <- site_impute(site0_tr, 'Site 0 Missing Data')

site1_tr <- site_impute(site1_tr, 'Site 1 Missing Data')

site2_tr <- site_impute(site2_tr, 'Site 2 Missing Data')

site3_tr <- site_impute(site3_tr, 'Site 3 Missing Data')

site4_tr <- site_impute(site4_tr, 'Site 4 Missing Data')

site5_tr <- site_impute(site5_tr, 'Site 5 Missing Data')


site6_tr <- site_impute(site6_tr, 'Site 6 Missing Data')


site8_tr <- site_impute(site8_tr, 'Site 8 Missing Data')


site9_tr <- site_impute(site9_tr, 'Site 9 Missing Data')


site10_tr <- site_impute(site10_tr, 'Site 10 Missing Data')


site12_tr <- site_impute(site12_tr, 'Site 12 Missing Data')

site13_tr <- site_impute(site13_tr, 'Site 13 Missing Data')

site14_tr <- site_impute(site14_tr, 'Site 14 Missing Data')

site15_tr <- site_impute(site15_tr, 'Site 15 Missing Data')
```

```
## Additional imputations: cloud coverage completely missing


## for site 7
gg_miss_var(site7_tr, show_pct = TRUE) + ggtitle('Site 7 Missing Data')

site7_tr$air_temperature <- ave(site7_tr$air_temperature, site7_tr$date, FUN = daily_avg)

site7_tr$dew_temperature <- ave(site7_tr$dew_temperature, site7_tr$date, FUN = daily_avg)

site7_tr$precip_depth_1_hr <- ave(site7_tr$precip_depth_1_hr, site7_tr$date, FUN = daily_avg)

site7_tr$precip_depth_1_hr <- ave(site7_tr$precip_depth_1_hr, site7_tr$month, FUN = monthly_avg)

site7_tr$wind_speed <- ave(site7_tr$wind_speed, site7_tr$date, FUN = daily_avg)

site7_tr$sea_level_pressure <- ave(site7_tr$sea_level_pressure, site7_tr$date, FUN = daily_avg)

site7_tr$wind_direction <- ave(site7_tr$wind_direction, site7_tr$date, FUN = daily_avg)

site7_tr$cloud_coverage <- ave(site7_tr$cloud_coverage, site7_tr$date, FUN = daily_avg)




## for site 11
gg_miss_var(site11_tr, show_pct = TRUE) + ggtitle('Site 11 Missing Data')

site11_tr$air_temperature <- ave(site11_tr$air_temperature, site11_tr$date, FUN = daily_avg)

site11_tr$dew_temperature <- ave(site11_tr$dew_temperature, site11_tr$date, FUN = daily_avg)

site11_tr$precip_depth_1_hr <- ave(site11_tr$precip_depth_1_hr, site11_tr$date, FUN = daily_avg)

site11_tr$precip_depth_1_hr <- ave(site11_tr$precip_depth_1_hr, site11_tr$month, FUN = monthly_avg)

site11_tr$wind_speed <- ave(site11_tr$wind_speed, site11_tr$date, FUN = daily_avg)

site11_tr$wind_direction <- ave(site11_tr$wind_direction, site11_tr$date, FUN = daily_avg)

site11_tr$cloud_coverage <- ave(site11_tr$cloud_coverage, site11_tr$date, FUN = daily_avg)

site11_tr$sea_level_pressure <- ave(site11_tr$sea_level_pressure, site11_tr$date, FUN = daily_avg)
```

```
## combine imputed training data
imputed_train <- rbind(site0_tr, site1_tr, site2_tr, site3_tr,
                       site4_tr, site5_tr, site6_tr, site7_tr,
                       site8_tr, site9_tr, site10_tr, site11_tr,
                       site12_tr, site13_tr, site14_tr, site15_tr)

gg_miss_fct(x = imputed_train, fct = site_id) + xlab('Site ID')
rm(imputed_train)
```

```r
site_train <- function(df, name, terms){
  ## input data frame, name of site for reference, and quantitative variables to include in the model
  set.seed(1)
  site_vars <- df[,terms] ## get data frame with the quantitative variables

  train = sample(1:nrow(site_vars), nrow(site_vars)*0.75)
  test = (-train)

  ln_meter_test = site_vars$ln_meter[test]


  ## Model matrix
  X = model.matrix(ln_meter ~., site_vars)[,-1]
  Y = site_vars$ln_meter

  ## Ridge regression


  cv_out <- cv.glmnet(X[train,], Y[train], alpha = 0)

  lamr <- cv_out$lambda.min

  ridge_fit <- glmnet(X[train,], Y[train], alpha = 0, lambda = lamr)

  pred_ridge <- predict(ridge_fit, s = lamr, newx = X[test,]) ## predictions in log scale
  lmse_r <- mean((pred_ridge - ln_meter_test)^2)

  ## Lasso Regression
  cv_out <- cv.glmnet(X[train,], Y[train], alpha = 1)

  laml <- cv_out$lambda.min

  lasso_fit <- glmnet(X[train,], Y[train], alpha = 1, lambda = laml)

  pred_lasso <- predict(lasso_fit, s = laml, newx = X[test,])
  lmse_l <- mean((pred_lasso - ln_meter_test)^2)


  ## elastic net
  cv_10 = trainControl(method = "cv", number = 10)

  cv_en <- train(ln_meter ~ ., data = site_vars[train,], method = 'glmnet', trControl = cv_10)

  tune_en <- cv_en$bestTune

  fit_en <- glmnet(X[train,], Y[train], alpha = tune_en[[1]], lambda = tune_en[[2]])

  en_pred <- predict(fit_en, X[test,])

  lmse_en <- mean((en_pred -  ln_meter_test)^2)

  print(name)
  lmse_df = data.frame(ridge = lmse_r, lasso = lmse_l, elastic_net = lmse_en)
```

```
  if((lmse_df$elastic_net < lmse_df$ridge) & (lmse_df$elastic_net < lmse_df$lasso)){
    print('use an elastic net regression model')
    print(tune_en)
  }else if((lmse_df$lasso < lmse_df$elastic_net) & (lmse_df$lasso < lmse_df$ridge)){
    print('use a lasso regression model')
    print(laml)
  }else{
    print('use a ridge regression model')
    print(lamr)
  }

  return(lmse_df)
  print(' ')

}


##### site 0 #####
site_train(site0_tr, 'site 0', c('ln_meter', 'square_feet', 'air_temperature',
                                 'cloud_coverage','dew_temperature', 'precip_depth_1_hr',
                                 'sea_level_pressure','wind_direction', 'wind_speed'))


##### site 1 #####

site_train(site1_tr, 'site 1', c('ln_meter', 'square_feet', 'air_temperature',
                                 'floor_count', 'cloud_coverage', 'dew_temperature',
                                 'sea_level_pressure', 'wind_direction', 'wind_speed'))

##### site 2 #####
site_train(site2_tr, 'site 2', c('ln_meter', 'square_feet', 'air_temperature',
                                 'cloud_coverage', 'dew_temperature', 'precip_depth_1_hr',
                                 'sea_level_pressure','wind_direction', 'wind_speed'))

##### site 3 #####
site_train(site3_tr, 'site 3', c('ln_meter', 'square_feet', 'air_temperature',
                                 'cloud_coverage', 'dew_temperature', 'precip_depth_1_hr',
                                 'sea_level_pressure','wind_direction', 'wind_speed'))

##### site 4 #####

site_train(site4_tr, 'site 4', c('ln_meter', 'square_feet', 'floor_count',
                                 'air_temperature', 'cloud_coverage', 'dew_temperature',
                                 'precip_depth_1_hr', 'sea_level_pressure',
                                 'wind_direction', 'wind_speed'))

##### site 5 #####

site_train(site5_tr, 'site 5', c('ln_meter', 'square_feet', 'air_temperature',
                                 'floor_count', 'cloud_coverage', 'dew_temperature',
                                 'wind_direction', 'wind_speed'))

##### site 6 #####
```

```
site_train(site6_tr, 'site 6', c('ln_meter', 'square_feet', 'air_temperature',
                                 'cloud_coverage', 'dew_temperature', 'precip_depth_1_hr',
                                 'sea_level_pressure','wind_direction', 'wind_speed'))

##### site 7 #####

site_train(site7_tr, 'site 7', c('ln_meter', 'square_feet', 'floor_count',
                                 'air_temperature', 'dew_temperature','precip_depth_1_hr',
                                 'sea_level_pressure', 'wind_direction', 'wind_speed'))

##### site 8 #####

site_train(site8_tr, 'site 8', c('ln_meter', 'square_feet', 'floor_count',
                                 'air_temperature', 'cloud_coverage', 'dew_temperature',
                                 'precip_depth_1_hr', 'sea_level_pressure',
                                 'wind_direction', 'wind_speed'))

##### site 9 #####
site_train(site9_tr, 'site 9', c('ln_meter', 'square_feet', 'air_temperature',
                                 'cloud_coverage', 'dew_temperature', 'precip_depth_1_hr',
                                 'sea_level_pressure','wind_direction', 'wind_speed'))

##### site 10 #####

site_train(site10_tr, 'site 10', c('ln_meter', 'square_feet', 'floor_count',
                                   'air_temperature', 'cloud_coverage', 'dew_temperature',
                                   'precip_depth_1_hr', 'sea_level_pressure',
                                   'wind_direction', 'wind_speed'))

##### site 11 #####
site_train(site11_tr, 'site 11', c('ln_meter', 'square_feet',
                                   'air_temperature', 'dew_temperature',
                                   'precip_depth_1_hr', 'sea_level_pressure',
                                   'wind_direction', 'wind_speed'))


#### site 12####
site_train(site12_tr, 'site 12', c('ln_meter', 'square_feet', 'air_temperature',
                                   'cloud_coverage', 'dew_temperature',
                                   'sea_level_pressure', 'wind_direction', 'wind_speed'))


##### site 13 #####
site_train(site13_tr, 'site 13', c('ln_meter', 'square_feet',
                                   'air_temperature', 'cloud_coverage',
                                   'dew_temperature', 'precip_depth_1_hr',
                                   'sea_level_pressure','wind_direction', 'wind_speed'))

##### site 14 #####
site_train(site14_tr, 'site 14', c('ln_meter', 'square_feet', 'air_temperature',
                                   'cloud_coverage', 'dew_temperature', 'precip_depth_1_hr',
                                   'sea_level_pressure','wind_direction', 'wind_speed'))
```

```
##### site 15 #####
site_train(site15_tr, 'site 15', c('ln_meter', 'square_feet', 'air_temperature',
                                   'cloud_coverage', 'dew_temperature', 'precip_depth_1_hr',
                                   'sea_level_pressure','wind_direction', 'wind_speed'))
```

```
site_impute2 <- function(df){

  df$date <- as.Date(df$timestamp) ## add date since the test set is twice as large

  df$month <- format(as.Date(df$timestamp), '%Y-%m') ## extract specific month

  df$air_temperature <- ave(df$air_temperature, df$date, FUN = daily_avg)

  df$air_temperature <- ave(df$air_temperature, df$month, FUN = monthly_avg)

  df$dew_temperature <- ave(df$dew_temperature, df$date, FUN = daily_avg)

  df$dew_temperature <- ave(df$dew_temperature, df$month, FUN = monthly_avg)

  df$precip_depth_1_hr <- ave(df$precip_depth_1_hr, df$date, FUN = daily_avg)

  df$precip_depth_1_hr <- ave(df$precip_depth_1_hr, df$month, FUN = monthly_avg)

  df$wind_speed <- ave(df$wind_speed, df$date, FUN = daily_avg)

  df$wind_speed <- ave(df$wind_speed, df$month, FUN = monthly_avg)

  df$sea_level_pressure <- ave(df$sea_level_pressure, df$date, FUN = daily_avg)

  df$sea_level_pressure <- ave(df$sea_level_pressure, df$month, FUN = monthly_avg)

  df$wind_direction <- ave(df$wind_direction, df$date, FUN = daily_avg)

  df$wind_direction <- ave(df$wind_direction, df$month, FUN = monthly_avg)

  df$cloud_coverage <- ave(df$cloud_coverage, df$date, FUN = daily_avg)

  df$cloud_coverage <- ave(df$cloud_coverage, df$month, FUN = monthly_avg)

  df
}

test_data <- b_test3


site0_t <- subset(test_data, site_id == 0)
site0_t <- site_impute2(site0_t)

site1_t <- subset(test_data, site_id == 1)
site1_t <- site_impute2(site1_t) ## additional imputations

site2_t <- subset(test_data, site_id == 2)
site2_t <- site_impute2(site2_t)
```

```
site3_t <- subset(test_data, site_id == 3)
site3_t <- site_impute2(site3_t)

site4_t <- subset(test_data, site_id == 4)
site4_t <- site_impute2(site4_t)

site5_t <- subset(test_data, site_id == 5)
site5_t <- site_impute2(site5_t)

site6_t <- subset(test_data, site_id == 6)
site6_t <- site_impute2(site6_t)

site7_t <- subset(test_data, site_id == 7)


site8_t <- subset(test_data, site_id == 8)
site8_t <- site_impute2(site8_t)

site9_t <- subset(test_data, site_id == 9)
site9_t <- site_impute2(site9_t)

site10_t <- subset(test_data, site_id == 10)
site10_t <- site_impute2(site10_t)

site11_t <- subset(test_data, site_id == 11)


site12_t <- subset(test_data, site_id == 12)
site12_t <- site_impute2(site12_t)

site13_t <- subset(test_data, site_id == 13)
site13_t <- site_impute2(site13_t)

site14_t <- subset(test_data, site_id == 14)
site14_t <- site_impute2(site14_t)

site15_t <- subset(test_data, site_id == 15)
site15_t <- site_impute2(site15_t)

rm(b_test3)
rm(test_data)
rm(b_train3)
```

```
### the site impute function did not work for sites 7 and 11, so I imputed the daily and monthly averag

#### site 7 ####
site7_t$date <- as.Date(site7_t$timestamp) ## add date since the test set is twice as large

site7_t$month <- format(as.Date(site7_t$timestamp), '%Y-%m')

site7_t$air_temperature <- ave(site7_t$air_temperature, site7_t$date, FUN = daily_avg)

site7_t$air_temperature <- ave(site7_t$air_temperature, site7_t$month, FUN = monthly_avg)
```

```r
site7_t$dew_temperature <- ave(site7_t$dew_temperature, site7_t$date, FUN = daily_avg)
site7_t$dew_temperature <- ave(site7_t$dew_temperature, site7_t$month, FUN = monthly_avg)
site7_t$precip_depth_1_hr <- ave(site7_t$precip_depth_1_hr, site7_t$date, FUN = daily_avg)
site7_t$precip_depth_1_hr <- ave(site7_t$precip_depth_1_hr, site7_t$month, FUN = monthly_avg)
site7_t$wind_speed <- ave(site7_t$wind_speed, site7_t$date, FUN = daily_avg)
site7_t$wind_speed <- ave(site7_t$wind_speed, site7_t$month, FUN = monthly_avg)
site7_t$wind_direction <- ave(site7_t$wind_direction, site7_t$date, FUN = daily_avg)
site7_t$wind_direction <- ave(site7_t$wind_direction, site7_t$month, FUN = monthly_avg)
site7_t$sea_level_pressure <- ave(site7_t$sea_level_pressure, site7_t$date, FUN = daily_avg)
site7_t$sea_level_pressure <- ave(site7_t$sea_level_pressure, site7_t$month, FUN = monthly_avg)


#### site 11 ####
site11_t$date <- as.Date(site11_t$timestamp) ## add date since the test set is twice as large
site11_t$month <- format(as.Date(site11_t$timestamp), '%Y-%m')
site11_t$air_temperature <- ave(site11_t$air_temperature, site11_t$date, FUN = daily_avg)
site11_t$air_temperature <- ave(site11_t$air_temperature, site11_t$month, FUN = monthly_avg)
site11_t$dew_temperature <- ave(site11_t$dew_temperature, site11_t$date, FUN = daily_avg)
site11_t$dew_temperature <- ave(site11_t$dew_temperature, site11_t$month, FUN = monthly_avg)
site11_t$precip_depth_1_hr <- ave(site11_t$precip_depth_1_hr, site11_t$date, FUN = daily_avg)
site11_t$precip_depth_1_hr <- ave(site11_t$precip_depth_1_hr, site11_t$month, FUN = monthly_avg)
site11_t$wind_speed <- ave(site11_t$wind_speed, site11_t$date, FUN = daily_avg)
site11_t$wind_speed <- ave(site11_t$wind_speed, site11_t$month, FUN = monthly_avg)
site11_t$wind_direction <- ave(site11_t$wind_direction, site11_t$date, FUN = daily_avg)
site11_t$wind_direction <- ave(site11_t$wind_direction, site11_t$month, FUN = monthly_avg)
site11_t$sea_level_pressure <- ave(site11_t$sea_level_pressure, site11_t$date, FUN = daily_avg)
site11_t$sea_level_pressure <- ave(site11_t$sea_level_pressure, site11_t$month, FUN = monthly_avg)
options(scipen=999)
## predict with elastic net
```

```
site_predict_en <- function(df_tr, df_t, terms_tr, terms_t){
  set.seed(1)
  site_df <- df_tr[, terms_tr] ## training data

  site_df2 <- df_t[, terms_t] ## test data

  X_tr = model.matrix(ln_meter ~., site_df)[,-1]
  Y_tr = site_df$ln_meter

  X_t <- as.matrix(site_df2) ## data for test and predict

  ## elastic net
  cv_10 <- trainControl(method = "cv", number = 10)

  cv_en <- train(ln_meter ~ ., data = site_df, method = 'glmnet', trControl = cv_10)

  tune_en <- cv_en$bestTune

  print(tune_en) ## print alpha and lambdas

  fit_en <- glmnet(X_tr, Y_tr, alpha = tune_en[[1]], lambda = tune_en[[2]]) ## extract alpha and lambda

  print(coef(fit_en)) ## print model coefficients

  preds <- predict(fit_en, X_t)

  df_t$ln_meter <- as.data.frame(preds)

  df_t
}

##### site 0 #####
site0_p <- site_predict_en(site0_tr, site0_t, c('ln_meter', 'square_feet', 'air_temperature',
                                                'cloud_coverage', 'dew_temperature',
                                                'precip_depth_1_hr', 'sea_level_pressure',
                                                'wind_direction', 'wind_speed'),
                           c('square_feet', 'air_temperature',
                             'cloud_coverage', 'dew_temperature',
                             'precip_depth_1_hr', 'sea_level_pressure',
                             'wind_direction', 'wind_speed'))

rm(site0_tr)
rm(site0_t)

##### site 1 #####
site1_p <- site_predict_en(site1_tr, site1_t,c('ln_meter', 'square_feet',
                                               'air_temperature', 'floor_count',
                                               'cloud_coverage', 'dew_temperature',
                                               'sea_level_pressure', 'wind_direction', 'wind_speed'),
                           c('square_feet', 'air_temperature', 'floor_count',
                             'cloud_coverage','dew_temperature',
                             'sea_level_pressure', 'wind_direction', 'wind_speed'))
```

```
rm(site1_tr)
rm(site1_t)

##### site 2 #####
site2_p <- site_predict_en(site2_tr, site2_t,c('ln_meter', 'square_feet',
                                               'air_temperature', 'cloud_coverage',
                                               'dew_temperature', 'precip_depth_1_hr',
                                               'sea_level_pressure','wind_direction', 'wind_speed'),
                           c('square_feet', 'air_temperature', 'cloud_coverage',
                             'dew_temperature', 'precip_depth_1_hr',
                             'sea_level_pressure','wind_direction', 'wind_speed'))

rm(site2_tr)
rm(site2_t)

##### site 3 #####
site3_p <- site_predict_en(site3_tr, site3_t, c('ln_meter', 'square_feet',
                                                'air_temperature', 'cloud_coverage',
                                                'dew_temperature', 'precip_depth_1_hr',
                                                'sea_level_pressure','wind_direction', 'wind_speed'),
                           c('square_feet', 'air_temperature', 'cloud_coverage',
                             'dew_temperature', 'precip_depth_1_hr',
                             'sea_level_pressure','wind_direction', 'wind_speed'))

rm(site3_tr)
rm(site3_t)

##### site 4 #####

site4_p <- site_predict_en(site4_tr, site4_t, c('ln_meter', 'square_feet', 'floor_count',
                                                'air_temperature', 'cloud_coverage', 'dew_temperature',
                                                'precip_depth_1_hr', 'sea_level_pressure',
                                                'wind_direction', 'wind_speed'),
                           c('square_feet', 'floor_count', 'air_temperature',
                             'cloud_coverage', 'dew_temperature',
                             'precip_depth_1_hr', 'sea_level_pressure',
                             'wind_direction', 'wind_speed'))

rm(site4_tr)
rm(site4_t)

##### site 6 #####

site6_p <- site_predict_en(site6_tr, site6_t, c('ln_meter', 'square_feet', 'air_temperature',
                                                'cloud_coverage', 'dew_temperature',
                                                'precip_depth_1_hr', 'sea_level_pressure',
                                                'wind_direction', 'wind_speed'),
                           c('square_feet', 'air_temperature', 'cloud_coverage',
                             'dew_temperature', 'precip_depth_1_hr',
                             'sea_level_pressure','wind_direction', 'wind_speed'))

rm(site6_tr)
rm(site6_t)
```

```
#### site 8 ####

site8_p <- site_predict_en(site8_tr, site8_t, c('ln_meter', 'square_feet',
                                                'floor_count', 'air_temperature',
                                                'cloud_coverage', 'dew_temperature',
                                                'precip_depth_1_hr', 'sea_level_pressure',
                                                'wind_direction', 'wind_speed'),
                           c('square_feet', 'floor_count',
                             'air_temperature', 'cloud_coverage',
                             'dew_temperature', 'precip_depth_1_hr',
                             'sea_level_pressure', 'wind_direction', 'wind_speed'))

rm(site8_tr)
rm(site8_t)

##### site 9 #####
site9_p <- site_predict_en(site9_tr, site9_t, c('ln_meter', 'square_feet',
                                                'air_temperature', 'cloud_coverage',
                                                'dew_temperature', 'precip_depth_1_hr',
                                                'sea_level_pressure','wind_direction', 'wind_speed'),
                           c('square_feet', 'air_temperature',
                             'cloud_coverage', 'dew_temperature',
                             'precip_depth_1_hr', 'sea_level_pressure',
                             'wind_direction', 'wind_speed'))

rm(site9_tr)
rm(site9_t)


##### site 11 #####
site11_p <- site_predict_en(site11_tr, site11_t, c('ln_meter', 'square_feet',
                                                   'air_temperature', 'dew_temperature',
                                                   'precip_depth_1_hr', 'sea_level_pressure',
                                                   'wind_direction', 'wind_speed'),
                            c('square_feet', 'air_temperature',
                              'dew_temperature','precip_depth_1_hr',
                              'sea_level_pressure', 'wind_direction', 'wind_speed'))

rm(site11_tr)
rm(site11_t)

##### site 12 #####
site12_p <- site_predict_en(site12_tr, site12_t, c('ln_meter', 'square_feet',
                                                   'air_temperature', 'cloud_coverage',
                                                   'dew_temperature', 'sea_level_pressure',
                                                   'wind_direction', 'wind_speed'),
                            c('square_feet', 'air_temperature',
                              'cloud_coverage', 'dew_temperature',
                              'sea_level_pressure', 'wind_direction', 'wind_speed'))

rm(site12_tr)
rm(site12_t)
```

```
##### site 14 #####
site14_p <- site_predict_en(site14_tr, site14_t, c('ln_meter', 'square_feet', 'air_temperature',
                                                   'cloud_coverage', 'dew_temperature',
                                                   'precip_depth_1_hr', 'sea_level_pressure',
                                                   'wind_direction', 'wind_speed'),
                            c('square_feet', 'air_temperature', 'cloud_coverage',
                              'dew_temperature', 'precip_depth_1_hr',
                              'sea_level_pressure','wind_direction', 'wind_speed'))

rm(site14_tr)
rm(site14_t)

##### site 15 #####
site15_p <- site_predict_en(site15_tr, site15_t, c('ln_meter', 'square_feet',
                                                   'air_temperature', 'cloud_coverage',
                                                   'dew_temperature', 'precip_depth_1_hr',
                                                   'sea_level_pressure','wind_direction', 'wind_speed')
                            c('square_feet', 'air_temperature', 'cloud_coverage',
                              'dew_temperature', 'precip_depth_1_hr', 'sea_level_pressure',
                              'wind_direction', 'wind_speed'))

rm(site15_tr)
rm(site15_t)
```

```
options(scipen=999)
## predict with lasso regression
site_predict_l <- function(df_tr, df_t, terms_tr, terms_t){
  site_df <- df_tr[, terms_tr] ## training data

  site_df2 <- df_t[, terms_t] ## test data

  X_tr = model.matrix(ln_meter ~., site_df)[,-1]
  Y_tr = site_df$ln_meter

  X_t <- as.matrix(site_df2) ## data for test and predict

  ## Lasso Regression
  cv_out <- cv.glmnet(X_tr, Y_tr, alpha = 1)

  laml <- cv_out$lambda.min

  print(laml)

  lasso_fit <- glmnet(X_tr, Y_tr, alpha = 1, lambda = laml)

  print(coef(lasso_fit))

  preds <- predict(lasso_fit, X_t)

  df_t$ln_meter <- as.data.frame(preds)

  df_t
```

```
}



##### site 5 #####

site5_p <- site_predict_l(site5_tr, site5_t,
                          c('ln_meter', 'square_feet', 'air_temperature',
                            'floor_count', 'cloud_coverage', 'dew_temperature',
                            'wind_direction', 'wind_speed'),
                          c('square_feet', 'air_temperature',
                            'floor_count', 'cloud_coverage',
                            'dew_temperature', 'wind_direction', 'wind_speed'))

rm(site5_tr)
rm(site5_t)

#### site 7 ####
site7_p <- site_predict_l(site7_tr, site7_t, c('ln_meter', 'square_feet',
                                               'floor_count', 'air_temperature',
                                               'dew_temperature','precip_depth_1_hr',
                                               'sea_level_pressure', 'wind_direction', 'wind_speed'),
                          c('square_feet', 'floor_count', 'air_temperature',
                            'dew_temperature','precip_depth_1_hr', 'sea_level_pressure',
                            'wind_direction', 'wind_speed'))

rm(site7_tr)
rm(site7_t)

##### site 13 #####
site13_p <- site_predict_l(site13_tr, site13_t,
                           c('ln_meter', 'square_feet', 'air_temperature',
                             'cloud_coverage', 'dew_temperature', 'precip_depth_1_hr',
                             'sea_level_pressure','wind_direction', 'wind_speed'),
                           c('square_feet', 'air_temperature', 'cloud_coverage',
                             'dew_temperature', 'precip_depth_1_hr', 'sea_level_pressure',
                             'wind_direction', 'wind_speed'))

rm(site13_tr)
rm(site13_t)
```

```
options(scipen=999)
## predict with ridge regression
site_predict_r <- function(df_tr, df_t, terms_tr, terms_t){
  site_df <- df_tr[, terms_tr] ## training data

  site_df2 <- df_t[, terms_t] ## test data

  X_tr = model.matrix(ln_meter ~., site_df)[,-1]
  Y_tr = site_df$ln_meter

  X_t <- as.matrix(site_df2) ## data for test and predict

  ## Lasso Regression
```

```
  cv_out <- cv.glmnet(X_tr, Y_tr, alpha = 0)

  lamr <- cv_out$lambda.min

  print(lamr)

  ridge_fit <- glmnet(X_tr, Y_tr, alpha = 0, lambda = lamr)

  print(coef(ridge_fit))

  preds <- predict(ridge_fit, X_t)

  df_t$ln_meter <- as.data.frame(preds)

  df_t
}


##### site 10 #####

site10_p <- site_predict_r(site10_tr, site10_t,
                           c('ln_meter', 'square_feet',
                             'floor_count', 'air_temperature', 'cloud_coverage',
                             'dew_temperature', 'precip_depth_1_hr',
                             'sea_level_pressure', 'wind_direction', 'wind_speed'),
                           c('square_feet', 'floor_count', 'air_temperature',
                             'cloud_coverage', 'dew_temperature', 'precip_depth_1_hr',
                             'sea_level_pressure', 'wind_direction', 'wind_speed'))

rm(site10_tr)
rm(site10_t)
```

```
## bind prediction datasets together

energy_preds <- rbind(site0_p, site1_p, site2_p, site3_p,
                      site4_p, site5_p, site6_p, site7_p,
                      site8_p, site9_p, site10_p, site11_p,
                      site12_p, site13_p, site14_p, site15_p)

energy_preds$ln_meter <- unlist(energy_preds$ln_meter)



energy_preds$meter_reading <- exp(energy_preds$ln_meter) - 1

energy_preds$meter_reading <- round(energy_preds$meter_reading, 4)

energy_preds2 <- energy_preds %>% dplyr::select(row_id, meter_reading)



colnames(energy_preds2) <- c('row_id', 'meter_reading')

fwrite(energy_preds2, '~/Documents/STAT 488/energy_submission3.csv', row.names = FALSE)
```