New York City LL84 - Site EUI Model

Eden Axelrad

7/17/2022

Contents

SECTION 1: Introduction and Overview

This introductory section describes the data set, variables, project goals, and key steps.

The data set used for this machine learning exercise combines building information data and weather data for New York City. The building data was retrieved from NYC Open Data and is part of the city's Local Law 84 (LL84). LL84 is contributes to the city's building performance standard benchmarking efforts and applies to any privately owned buildings over 25,000 square feet and municipal buildings over 10,000 square feet. Most importantly, the data set includes metrics on energy use and consumption by building during a reporting year. This data was supplemented with weather data from 3 local weather stations - Central Park, LaGuardia, and Kennedy. The weather data was aggregated up to a calendar year and used as an average of all three stations.

The final data set (which is used here) is a cleaned and aggregated data set consisting of LL84 data and weather data for 2016, 2017, 2018, and 2019. The R script I used to retrieve, wrangle, and clean the data can be found in the project repo, or accessed via this link.

While there were many more (hundreds) metrics available through LL84, I wanted to crop the final variable list to include data that is publicly available and easy to retrieve for most buildings. This provided the following list of parameters:

- Reporting Year (year)
- Primary Property Type (primary_property_type)
- Year Built (year_built)
- Number of Buildings (number_of_buildings)
- ENERGY STAR Score (energy_star_score)
- Property Gross Floor Area in in square feet (property_gfa)
- Parking Gross Floor Area in in square feet (parking_gfa)
- LEED Project (leed_project)
- Site Energy Use Intensity (EUI) in kBtu/sqft (site_eui)
- Cooling Demand Days with base 65F (CDD)
- Heating Demand Days with base 65F (HDD)
- Mean Maximum Annual Temperature (MeanMaxTemp)
- Mean Minimum Annual Temperature (MeanMinTemp)

The project goal was to predict a building's Site Energy Use Intensity (EUI) using common and publicly available variables. Site EUI is metric commonly used for describing a buildings energy efficiency, typically with a unit of kBtu per square foot. Sites with lower EUI, use less energy per square foot. This can be particularly useful for benchmarking across similar property types, setting energy efficiency reduction targets, or assessing the impacts of energy efficiency improvements.

To achieve the above goal, I first obtained a better understanding of the data set with exploratory analysis. Next, I applied several modeling techniques with increasing complexity in order to observe changes in performance (using root mean square error, RMSE, as a primary metric). These approaches include a naive model, using property type EUI averages, simple linear regression, multiple linear regression, classification and regression tree, and random forest.

I decided to utilize 2016, 2017, and 2018 reporting year data for my train and test set, with 2019 left out for a final validation.

SECTION 2: Setup

In this section I load the required packages, set up the initial data set, and split it in to an initial machine learning set and final validation set.

```r
# Load the cleaned up data for reporting years 2016 through 2019
NYC_LL84 <- read.csv("Data/NYC_LL84_Clean.csv")

# Create our test and train data for machine learning from reporting years 2016-18
NYC_LL84_ML <- NYC_LL84 %>%
  filter(year %in% c("2016", "2017", "2018")) %>%
  select(year, primary_property_type, year_built,
         number_of_buildings, energy_star_score, property_gfa, parking_gfa,
         leed_project, CDD, HDD, MeanMaxTemp, MeanMinTemp, site_eui)

# Create the validation set to be used in the very end
NYC_LL84_Validation <- NYC_LL84 %>%
  filter(year == "2019") %>%
  select(year, primary_property_type, year_built,
         number_of_buildings, energy_star_score, property_gfa, parking_gfa,
         leed_project, CDD, HDD, MeanMaxTemp, MeanMinTemp, site_eui)
```

SECTION 3: Methods + Analysis

This section explains the process and techniques used for this analysis, including data exploration and visualization and modeling approaches.

SECTION 3.1: Exploratory Analysis

I began the exploration process by using `glimpse()` on the entire data set and obtaining summary stats for the Site EUI column. Next, I created both a table and boxplot chart to showcase Site EUI by property type. This helps us see the mean EUI and distribution for each distinct type of building. For example we can easily identify the least energy efficient building types - data centers, waste water treatment plants, hospitals, and grocery stores. I also examined average EUI by building size and found that buildings in the first size quartile perform worst. Lastly, I take a look at the correlation coefficients for numerical predictors. This shows that, unsurprisingly, ENERGY STAR score is the most correlated with Site EUI.[1]

```r
# Summary of data frame
glimpse(NYC_LL84_ML)
```

```
## Rows: 34,470
## Columns: 13
## $ year                  <int> 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, ~
## $ primary_property_type <chr> "Multifamily Housing", "Multifamily Housing", "M~
## $ year_built            <int> 1989, 1939, 1967, 1955, 1944, 1900, 1914, 1936, ~
## $ number_of_buildings   <int> 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ energy_star_score     <int> 13, 1, 2, 86, 74, 90, 20, 64, 96, 98, 100, 60, 2~
## $ property_gfa          <dbl> 248830, 57994, 56100, 52700, 63940, 56900, 32001~
## $ parking_gfa           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13036, 0, 0, 0, 0,~
## $ leed_project          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ CDD                   <dbl> 1532, 1532, 1532, 1532, 1532, 1532, 1532, 1532, ~
## $ HDD                   <dbl> 4162, 4162, 4162, 4162, 4162, 4162, 4162, 4162, ~
## $ MeanMaxTemp           <dbl> 64.8, 64.8, 64.8, 64.8, 64.8, 64.8, 64.8, 64.8, ~
```

[1]ENERGY STAR score shows a building's energy consumption on a scale from 1-100. Scores are relative to building type (nationwide), and 50 denotes a median value. Benchmarking data for the score comes from the U.S. Department of Energy's Energy Information Administration (EIA) Commercial Building Energy Consumption Survey (CBECS).

```
## $ MeanMinTemp           <dbl> 49.56667, 49.56667, 49.56667, 49.56667, 49.56667~
## $ site_eui              <dbl> 132.6, 458.8, 165.5, 57.6, 50.3, 58.8, 58.0, 52.~
```

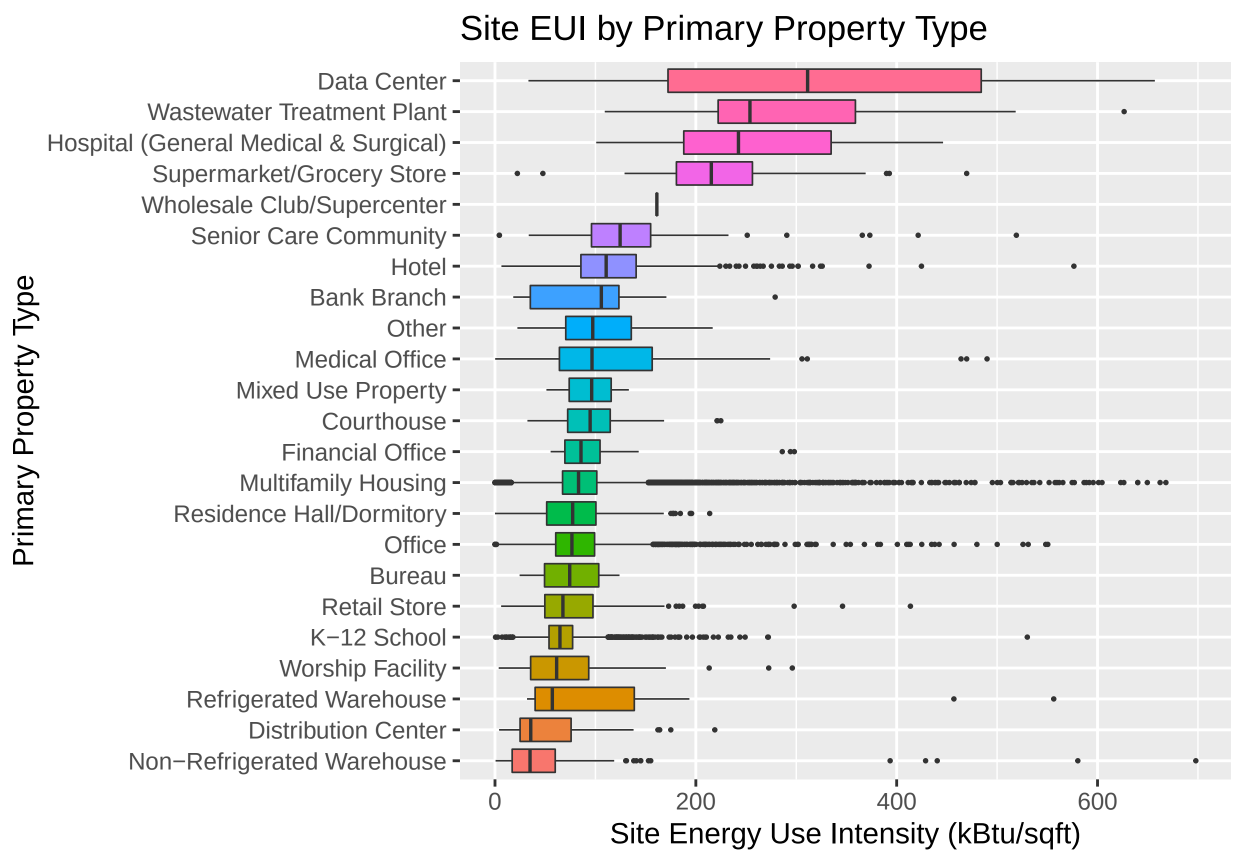
```
# Summary stats for the Site EUI column
summary(NYC_LL84_ML$site_eui)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    0.00   63.60   80.40   85.69  100.30  697.90
```

```
# View EUI by property type and frequency of occurrence
NYC_LL84_ML %>%
  group_by("Property Type" = primary_property_type) %>%
  summarise("Site EUI (kBtu/sqft)" = round(mean(site_eui),
                                          digits = 1),
            Count = n())  %>%
  arrange(desc(Count)) %>%
  knitr::kable()
```

Property Type	Site EUI (kBtu/sqft)	Count
Multifamily Housing	86.7	25175
K-12 School	67.9	3881
Office	86.2	3169
Hotel	118.3	747
Residence Hall/Dormitory	74.5	300
Non-Refrigerated Warehouse	50.3	291
Senior Care Community	132.0	185
Distribution Center	51.8	135
Retail Store	83.2	130
Medical Office	125.7	79
Hospital (General Medical & Surgical)	258.3	72
Worship Facility	70.5	72
Courthouse	97.7	64
Supermarket/Grocery Store	225.0	47
Financial Office	102.5	36
Mixed Use Property	94.5	18
Other	110.0	16
Refrigerated Warehouse	129.8	16
Wastewater Treatment Plant	310.4	13
Bank Branch	103.0	10
Bureau	74.8	10
Data Center	333.9	3
Wholesale Club/Supercenter	161.2	1

```
# Create boxplot of site EUI by property type
NYC_LL84_ML %>%
  mutate(primary_property_type = fct_reorder(primary_property_type,
                                             site_eui)) %>%
  ggplot(aes(x = primary_property_type,
             y = site_eui,
             fill = primary_property_type)) +
  geom_boxplot(size = .3, outlier.size = 0.3) +
  labs(x = "Primary Property Type",
       y = "Site Energy Use Intensity (kBtu/sqft)",
       title = "Site EUI by Primary Property Type") +
  theme(legend.position = "none") +
  coord_flip()
```



```
# Size of building versus EUI
summary(NYC_LL84_ML$property_gfa)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     3190    53900    80000   142723   135575 27882654
```

```
# Examine average EUI by building area quartile
NYC_LL84_ML %>%
  mutate(size_quartile = case_when(
    `property_gfa` > 135575 ~ 4,
    `property_gfa` > 80000 & `property_gfa` <= 135575 ~ 3,
    `property_gfa` > 53900 & `property_gfa` <= 80000 ~ 2,
    `property_gfa` <= 53900 ~ 1)) %>%
  group_by(size_quartile) %>%
  summarise("Site EUI (kBtu/sqft)" = round(mean(site_eui),
                                          digits = 1),
            Count = n())  %>%
  arrange(size_quartile) %>%
  knitr::kable()
```

size_quartile	Site EUI (kBtu/sqft)	Count
1	96.6	8619
2	85.2	8620
3	78.7	8614
4	82.2	8617

```r
# Look at correlation by predictor for site EUI
NYC_LL84_ML %>%
  select_if(., is.numeric) %>%
  cor() %>%
  as.data.frame() %>%
  rownames_to_column(var = "Feature") %>%
  select(Feature, site_eui) %>%
  drop_na() %>%
  arrange(desc(abs(site_eui))) %>%
  knitr::kable()
```

Feature	site_eui
site_eui	1.0000000
energy_star_score	-0.6536021
parking_gfa	0.0697990
MeanMaxTemp	-0.0632620
year	0.0629778
MeanMinTemp	-0.0624447
HDD	0.0558473
year_built	-0.0480500
number_of_buildings	0.0393701
CDD	-0.0225226
property_gfa	0.0149992
leed_project	-0.0032529

SECTION 3.2: Modelling Methods

As stated in the introduction, I decided to use a series of modelling approaches/methods with increasing complexity. They include a naive model, using property type EUI averages, simple linear regression, multiple linear regression, classification and regression tree, and random forest. To begin the modelling process, I partition the data in to a test set with 20% of our data and train set with 80% of our data. This was chosen over a 10%/90% since it is preferable that all buildings types are represented in each set.

```r
# Partition the data in to a test set with 20% and train set with 80%
# Set seed to 92 for reproducing results
set.seed(92, sample.kind = "Rounding")
test_index <- createDataPartition(NYC_LL84_ML$site_eui,
                                  times = 1, p = 0.2, list = FALSE)
test_set <- NYC_LL84_ML %>% slice(test_index)
train_set <- NYC_LL84_ML %>% slice(-test_index)
```

Method #1: Naive Model I start off with a naive model that uses the train set average EUI as a predictor. This value will act as our predicted values for every observation of the test set EUI. The naive model is an important reference point for other models.

```r
# Generate predicted Site EUI values using the train set mean EUI
mu_hat <- mean(train_set$site_eui)
```

Method #2: Mean EUI by Property Type In the field, this is a common way of improving an estimate for EUI or GHG emissions. For this reason, I wanted to include it alongside the other methods. To execute it, I simply find the average EUI by property type and left join with the test set. The assumption here is that property type is an easy way to break down the data in to categories that significantly impact EUI.

```r
# Generate predicted Site EUI using the test set predictors and property type averages
pred_eui_avg <- left_join(test_set,
                          train_set %>%
                            group_by(primary_property_type) %>%
                            summarize(pred_avg_eui = mean(site_eui)),
                          by = "primary_property_type") %>%
  .$pred_avg_eui
```

Method #3: Simple Linear Regression Using ENERGY STAR Score Next up, I wanted to construct a simple linear model using ENERGY STAR score. I chose this variable since we saw that it is by far the most correlated with Site EUI. To train this model I used a 10-fold cross validation and no pre-processing.

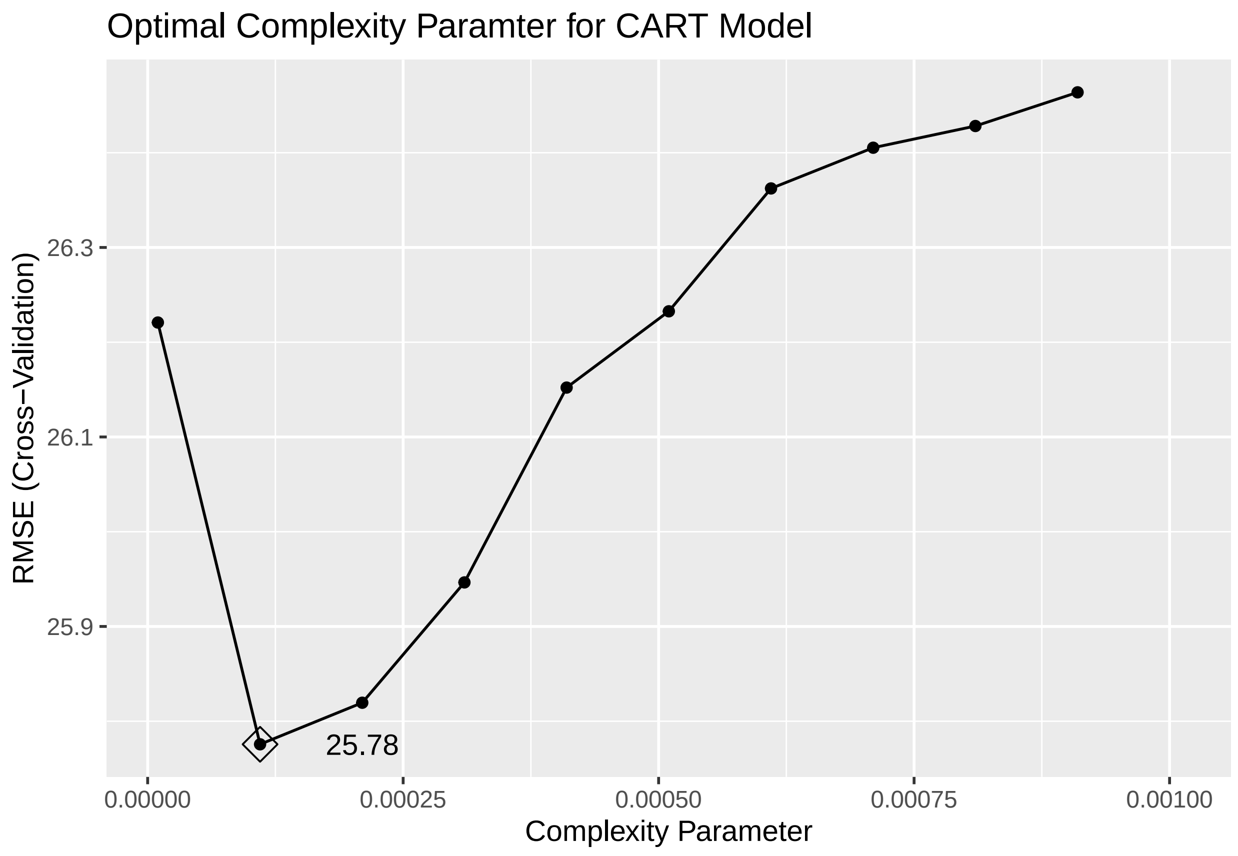
```r
# Train a linear model using only ENERGY STAR score
set.seed(92, sample.kind = "Rounding")
model_lr <- train(site_eui ~ energy_star_score,
                  data = train_set,
                  method = "lm",
                  trControl = trainControl(method = "cv", number = 10))

# Generate predicted Site EUI using the test set predictors and simple linear regression
pred_lr <- predict(model_lr,
                  newdata = test_set %>%
                    select(site_eui, energy_star_score))
```

Method #4: Multiple Linear Regression with all variables I build upon the simple linear model and add in the remaining predictors to create a multiple linear regression model. I was curious to see the impact the remaining variables had on RMSE (and other performance metrics). To train this model I used a 10-fold cross validation and no pre-processing.

```r
# Train a linear model with all variables included
# Use 10 fold cross-validation for resampling
set.seed(92, sample.kind = "Rounding")
model_mlr <- train(site_eui ~ .,
                   data = train_set,
                   method = "lm",
                   trControl = trainControl(method = "cv", number = 10))

# Generate predicted Site EUI using the test set predictors and MLR model
pred_mlr <- predict(model_mlr,
                    newdata = test_set)
```

Method #5: Classification and Regression Tree (CART) Moving on to more complex methods, I decided to implement a decision tree algorithm. The classification and regression tree (CART), as the name implies can be used for both classification and regression models. The CART model uses complexity parameters for tuning, which dictates the size of the tree. A smaller complexity parameter reduces the threshold at which another node is added, but going too far can overfit the data. Results for optimal number complexity parameters are shown in the graph below.

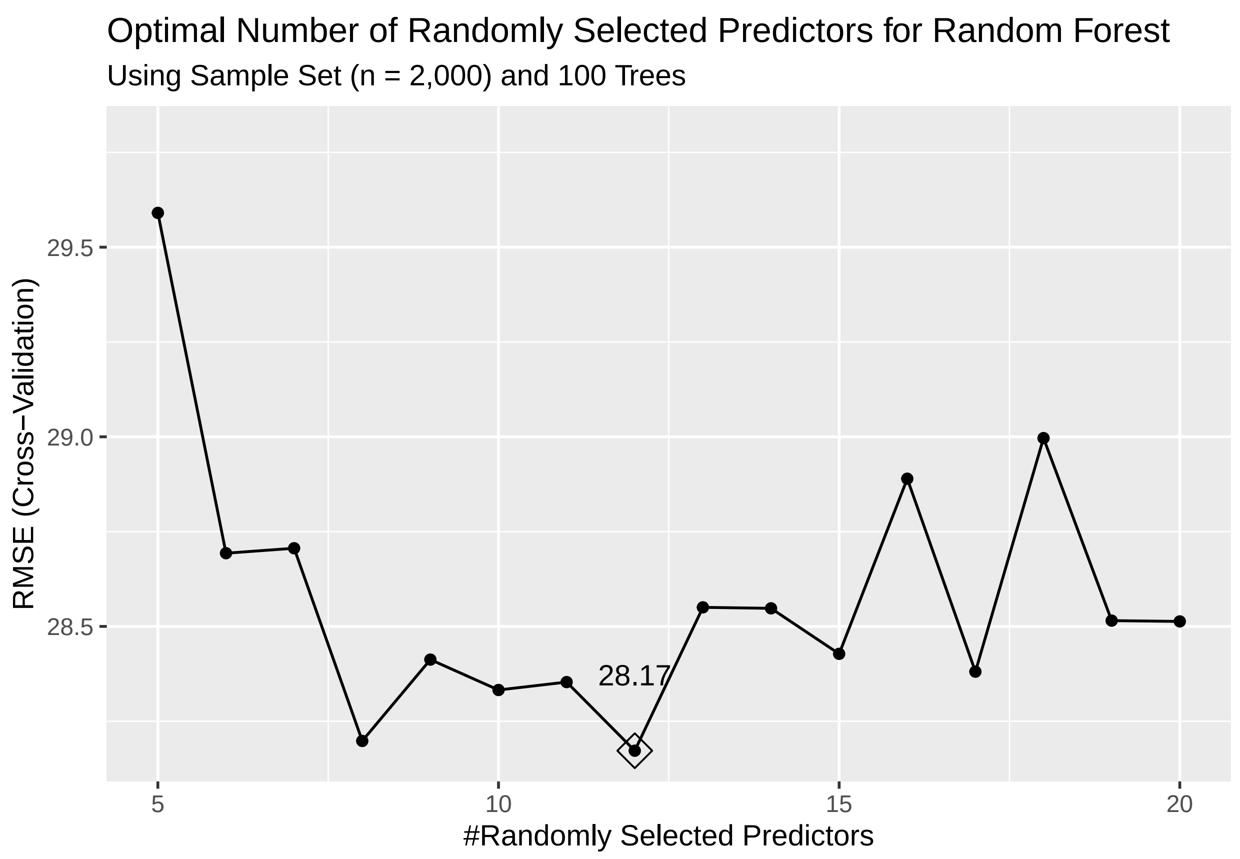
```r
# Train a model using a CART method from 'rpart' package
# Use 10 fold cross-validation for resampling
set.seed(92, sample.kind = "Rounding")
model_rpart <- train(site_eui ~ .,
                     data = train_set,
                     method = "rpart",
                     trControl = trainControl(method = "cv", number = 10),
                     tuneGrid = data.frame(cp = seq(.00001, .001, .0001)))

# Generate predicted Site EUI using the test set predictors and CART model
pred_rpart <- predict(model_rpart,
                      newdata = test_set)

# Generate a CART model plot which highlights our optimal tuning parameter (cp)
ggplot(model_rpart, highlight = TRUE) +
  geom_text(position = position_nudge(x = 0.0001),
            aes(label = ifelse(cp == model_rpart$bestTune[[1]],
                                   round(min(model_rpart$results$RMSE), digits = 2), ""))) +
  labs(title = "Optimal Complexity Paramter for CART Model")
```



Method #6: Random Forest Because there was a lot of success using the decision tree, a random forest model felt like a logical next step. To my understanding, a random forest machine learning algorithm is an ensemble of decision trees created on random samples of the data set. This technique is well known for its predictive power. The only down side is that it is computationally taxing. Since I am working with a failing computer from 2016, I had to piecemeal the model tuning using a sample set. To tune this model, I took a 2,000 observation sample from the train set, used 5 fold cross-validation instead of 10, and split up the tuning for number of trees and number of randomly selected predictors. This allowed me to define the best tuning parameters which I then used to train the original data set.

Since this is the final model, I decided to also examine the variable/feature importance. In the last plot, I present the top 10 most important features. At the top you can see ENERGY STAR score, property area, and year built.

```
# Create a sample of the train set in order to speed up the tuning process
set.seed(92, sample.kind = "Rounding")
sample_index <- sample(nrow(train_set), 2000)
sample_set <- train_set %>% slice(sample_index)


# Begin with sample set and minimal load test (low ntree and 5-fold cv) for optimal mtry
set.seed(92, sample.kind = "Rounding")
model_rf_default <- train(site_eui ~ .,
                          data = sample_set,
                          method = "rf",
                          ntree = 100,
                          trControl = trainControl(method = "cv", number = 5),
                          tuneGrid = data.frame(mtry = 5:20))

# View optimal tuning parameter (mtry) in a plot
ggplot(model_rf_default, highlight = TRUE) +
  geom_text(position = position_nudge(y = 0.2),
            aes(label = ifelse(mtry == model_rf_default$bestTune[[1]],
                               round(min(model_rf_default$results$RMSE),
                                     digits = 2), ""))) +
  labs(title = "Optimal Number of Randomly Selected Predictors for Random Forest",
       subtitle = "Using Sample Set (n = 2,000) and 100 Trees")
```



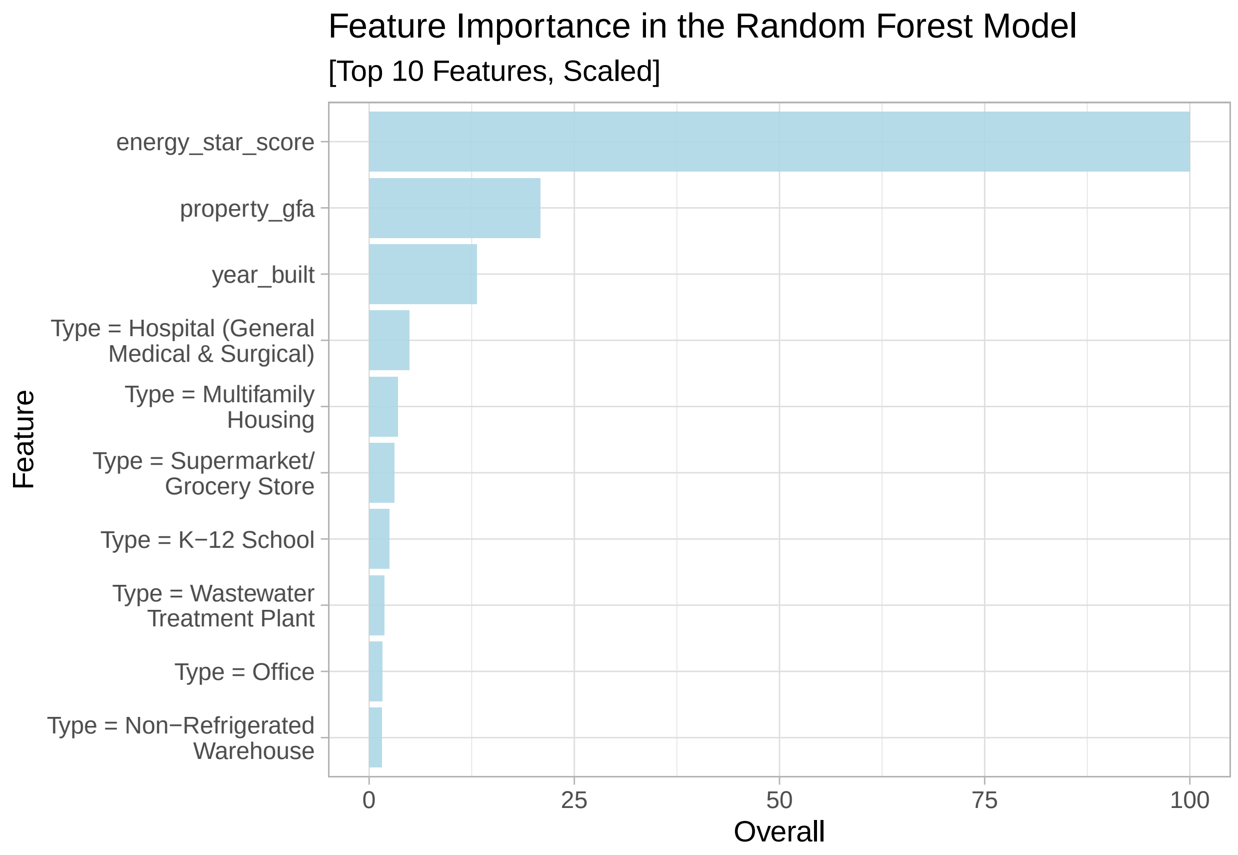
```r
# Find number of optimal trees to include in the model
ntrees <- c(100, 250, 500, 750, 1000)

# Use the sample test and optimal mtry to test out various tree sizes
tree_test <- sapply(ntrees, function(ntrees) {
  set.seed(92, sample.kind = "Rounding")
  model_rf <- train(site_eui ~ .,
                    data = sample_set,
                    method = "rf",
                    ntree = ntrees,
                    trControl = trainControl(method = "cv", number = 5),
                    tuneGrid = data.frame(mtry = model_rf_default$bestTune[[1]]))
  min(model_rf$results$RMSE)
})

tree_results <- data.frame("ntrees" = ntrees,
                           "rmse" = tree_test)
```

```r
# Create the actual random forest model using our tuning parameters above
# WARNING - Takes quite a long time to execute... 37.7 minutes
set.seed(92, sample.kind = "Rounding")
```

```
## Warning in set.seed(92, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
model_rf <- train(site_eui ~ .,
                  data = train_set,
                  method = "rf",
                  ntree = tree_results %>% slice_min(order_by = rmse) %>% .$ntrees,
                  tuneGrid = data.frame(mtry = model_rf_default$bestTune[[1]]),
                  trControl = trainControl(method = "cv"))

# Generate predicted Site EUI using the test set predictors and random forest model
pred_rf <- predict(model_rf,
                   newdata = test_set)
```

```r
# Find variable importance for the random forest model
importance_df <- varImp(model_rf)
importance_df <- importance_df[["importance"]] %>%
  rownames_to_column(var = "Feature")

# Display top 10 most important variables in a bar plot
importance_df %>%
  mutate(Feature = gsub(pattern = "primary_property_type",
                        replacement = "Type = ",
                        x = Feature),
         Feature = fct_reorder(Feature, Overall)) %>%
  slice_max(order_by = Overall, n = 10) %>%
  ggplot(aes(Feature, Overall)) +
  geom_col(fill = "lightblue", alpha = .9) +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 25)) +
  coord_flip() +
  theme_light() +
  labs(title = "Feature Importance in the Random Forest Model",
       subtitle = "[Top 10 Features, Scaled]")
```



SECTION 3.3: Validation

Since I had the most success with the random forest model, it was the model I decided to apply to our validation set (2019 reporting year data).

```r
# Generate predicted Site EUI using the validation set and random forest model
pred_validation <- predict(model_rf,
                           newdata = NYC_LL84_Validation)
```

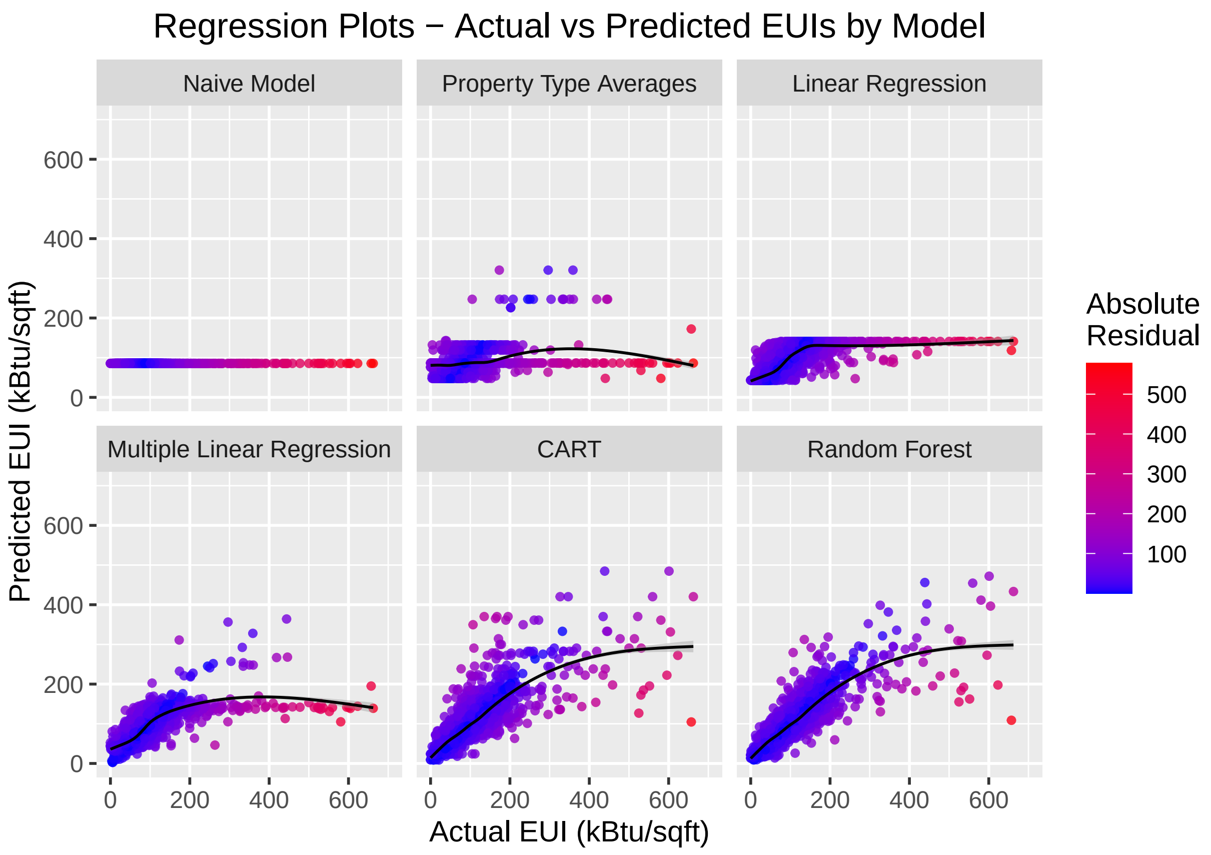
SECTION 4: Results

SECTION 4.1: Comparing All Models

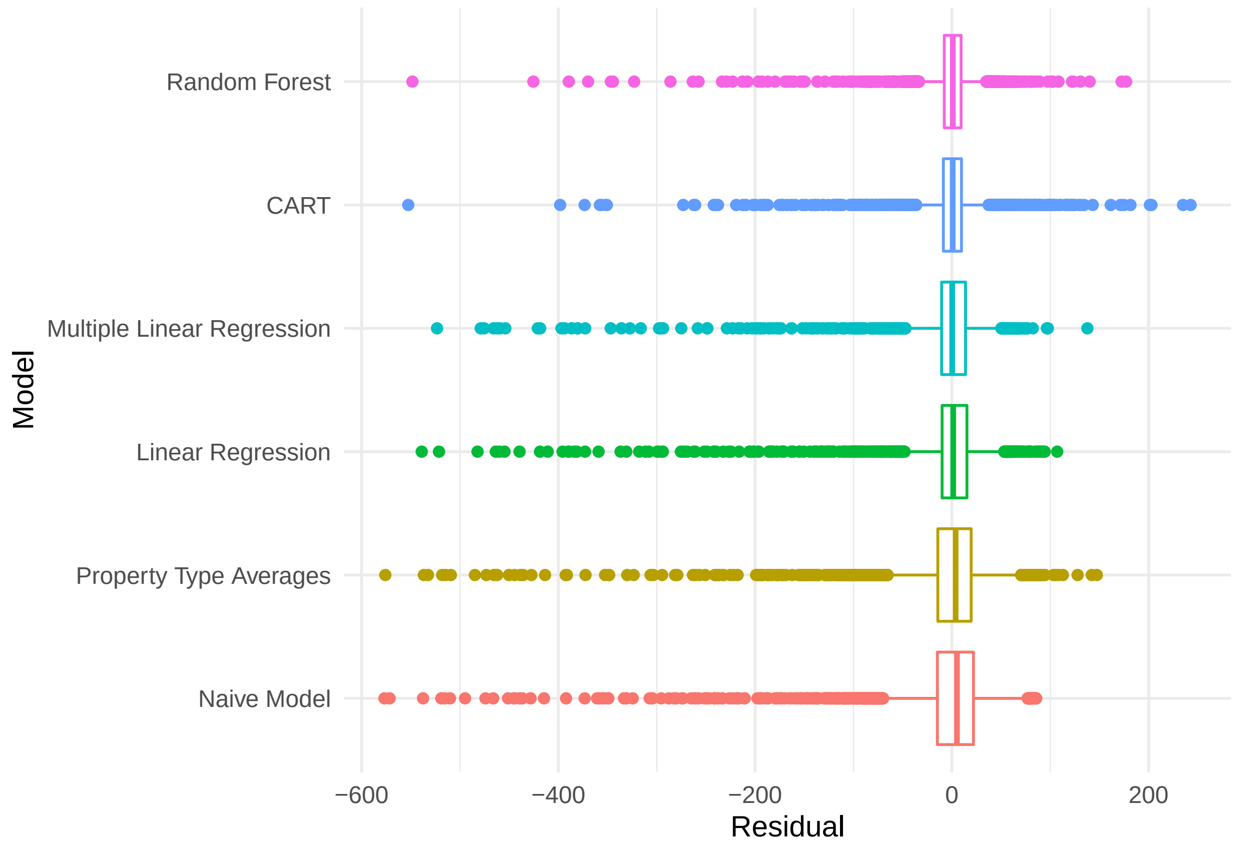
Here I show two plots which visually compare results across all of our models (excluding the validation set). First, a regression plot where we can examine the predicted and actual values by model. The absolute residual is presented through a color gradient. This plot indicates that the models had a harder time predicting the correct EUI when the actual values were high. In particular, the models assumed that buildings with very high EUI had lower EUIs. This finding is corroborated by the residual boxplot which showcases many outliers in the negative residual space.

These plots also depict an increasing predicvitve accuracy across our modelling approaches.

```r
# Save predicted values and residuals for all models in a data frame
Results <- cbind.data.frame(Actual = test_set$site_eui,
                            `Naive Model` = mu_hat,
                            `Property Type Averages` = pred_eui_avg,
                            `Linear Regression` = pred_lr,
                            `Multiple Linear Regression` = pred_mlr,
                            `CART` = pred_rpart,
                            `Random Forest` = pred_rf) %>%
  pivot_longer(cols = 2:7,
               values_to = "Predicted",
               names_to = "Model") %>%
  mutate(Residual = Predicted - Actual) %>%
  select(Model, Actual, Predicted, Residual)

# Create model factor levels based on order of our methods (increasing complexity)
Results$Model = factor(Results$Model, levels = c("Naive Model",
                                                 "Property Type Averages",
                                                 "Linear Regression",
                                                 "Multiple Linear Regression",
                                                 "CART",
                                                 "Random Forest"))

# Regression plots for actual vs predicted by model with color as RMSE
Results %>%
  ggplot(aes(x = Actual, y = Predicted, color = abs(Residual))) +
  geom_point(alpha = 0.8, size = 1) +
  scale_color_gradient(low = "blue", high = "red") +
  geom_smooth(color = "black", size = .5) +
  ylim(0, 700) +
  xlim(0, 700) +
  coord_equal() +
  facet_wrap(~Model) +
  labs(x = "Actual EUI (kBtu/sqft)",
       y = "Predicted EUI (kBtu/sqft)",
       color = "Absolute\nResidual",
       title = "Regression Plots - Actual vs Predicted EUIs by Model") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
# Boxplot of residuals by model
Results %>%
  ggplot(aes(x = Model, y = Residual, color = Model)) +
  geom_boxplot() +
  theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_text(hjust=0.5),
        plot.subtitle = element_text(hjust=0.5)) +
  coord_flip()
```



Finally, we present a results summary table depicting the R-squared, MAE, RMSE, and RMSE improvement over naive for each method discussed in this report. This table shows that we were able to significantly improve our model performance and ability to predict Site EUI using machine learning algorithms.

The final test set RMSE was 25.31, a 46.4% improvement over the naive model. The final validation set (2019 RY) RMSE was 23.65, even lower than the test set.

```
# Set the knitr options to omit NA from table output
opts <- options(knitr.kable.NA = "")


# Creating a function that uses predicted values to calculate R-squared, MAE, and RMSE
gen_results <- function(model) {
  results <- postResample(pred = model, obs = test_set$site_eui) %>%
    as.data.frame() %>%
    t()
}

# Final results table with method, model, R-squared, MAE, and RMSE
Results_Summary <- tibble(
  Method = c("Method #1", "Method #2", "Method #3", "Method #4",
             "Method #5", "Method #6", "Validation"),
  Model = c("Naive Model", "Property Type EUI Averages", "LR - ENERGY STAR Score",
            "Mulitple Linear Regression", "CART", "Random Forest", "Random Forest"),
  rbind.data.frame(
    gen_results(mu_hat), gen_results(pred_eui_avg), gen_results(pred_lr),
    gen_results(pred_mlr), gen_results(pred_rpart), gen_results(pred_rf),
```

Table 1: Final Summary Table for All Models

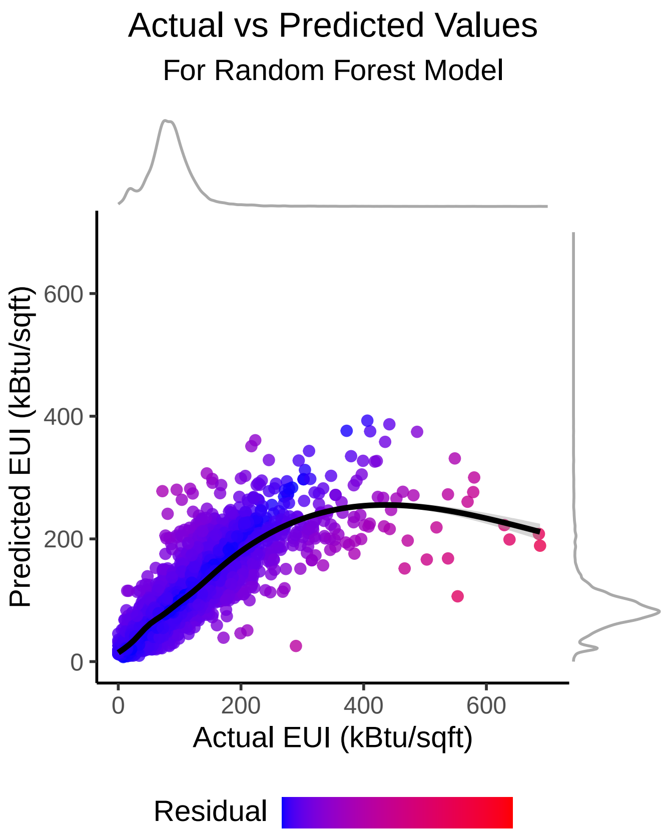
Method	Model	Rsquared	MAE	RMSE	RMSE Improvement
Method #1	Naive Model		28.03	47.25	0.0%
Method #2	Property Type EUI Averages	0.09	26.31	44.98	-4.8%
Method #3	LR - ENERGY STAR Score	0.40	19.17	36.55	-22.6%
Method #4	Mulitple Linear Regression	0.48	17.79	33.98	-28.1%
Method #5	CART	0.65	15.00	28.08	-40.6%
Method #6	Random Forest	0.72	13.51	25.31	-46.4%
Validation	**Random Forest**	**0.71**	**13.18**	**23.65**	**-49.9%**

```
    postResample(pred = pred_validation, obs = NYC_LL84_Validation$site_eui) %>%
      as.data.frame() %>% t())
) %>%
  mutate(`RMSE Improvement` = scales::percent((RMSE-RMSE[[1]])/RMSE[[1]])) %>%
  select(Method, Model, Rsquared, MAE, RMSE, `RMSE Improvement`)

Results_Summary %>%
  knitr::kable(align = c('l', 'l', rep('c', 4)),
               digits = 2,
               booktabs = TRUE,
               caption = "Final Summary Table for All Models") %>%
  kableExtra::row_spec(7, bold = TRUE) %>%
  kableExtra::row_spec(6, hline_after = TRUE)
```

SECTION 4.2: Validation Set Exploratory Analysis

I added this additional section because I was curious about the validation set as well. I created a regression plot similar to the ones above, and examined the RMSE by property type. I found that RMSE was greatest for data centers waste water treatment plants, and hospitals - some of the property types with the highest Site EUI. This lines up with our previous findings.

```
# Validation set - actual vs predicted with marginal distribution
ggExtra::ggMarginal(cbind.data.frame(Actual = NYC_LL84_Validation$site_eui,
                                     `Predicted` = pred_validation) %>%
                      mutate(Residual = Predicted - Actual) %>%
                      ggplot(aes(x = Actual, y = Predicted, color = abs(Residual))) +
                      geom_point(alpha = 0.8) +
                      scale_color_gradient(low = "blue", high = "red") +
                      geom_smooth(color = "black") +
                      ylim(0, 700) +
                      xlim(0, 700) +
                      coord_equal(ratio = 1) +
                      labs(x = "Actual EUI (kBtu/sqft)",
                           y = "Predicted EUI (kBtu/sqft)",
                           color = "Residual",
                           title = "Actual vs Predicted Values",
                           subtitle = "For Random Forest Model") +
                      theme_classic() +
                      theme(legend.position = "bottom",
                            plot.title = element_text(hjust=0.5),
                            plot.subtitle = element_text(hjust=0.5)) +
                      guides(color = guide_colourbar(ticks = FALSE,
                                                     label = FALSE,
                                                     barheight = 0.8,
                                                     title.vjust = 0.9)),
                    type = "density",
                    color = "darkgray")
```



```
# RMSE by property type for the validation set
cbind.data.frame(Actual = NYC_LL84_Validation$site_eui,
                 `Predicted` = pred_validation,
                 `Property Type` = NYC_LL84_Validation$primary_property_type) %>%
  mutate(Residual = Predicted - Actual) %>%
  group_by(`Property Type`) %>%
  summarize(RMSE = round(RMSE(Predicted, Actual, na.rm = TRUE), digits = 2),
            Count = n()) %>%
  arrange(desc(Count)) %>%
  knitr::kable()
```

Property Type	RMSE	Count
Multifamily Housing	21.66	14137
Office	23.77	1667
K-12 School	14.11	1468
Hotel	29.52	417
Non-Refrigerated Warehouse	35.32	175
Residence Hall/Dormitory	22.00	148
Retail Store	35.95	129
Worship Facility	27.86	116
Senior Care Community	47.72	110
Distribution Center	28.58	85
Hospital (General Medical & Surgical)	89.07	52
Medical Office	44.17	51
Supermarket/Grocery Store	53.49	36
Courthouse	35.21	22
Financial Office	35.54	17
Refrigerated Warehouse	38.76	16
Wastewater Treatment Plant	173.03	13
Mixed Use Property	17.34	9
Bank Branch	16.29	6
Data Center	299.79	3
Wholesale Club/Supercenter	19.63	2

SECTION 5: Conclusion

The goal of this project was to predict a building's Site EUI with machine learning algorithms. To find a final model with optimal performance, I tested out several methods with increasing complexity. They consist of a naive model, using property type EUI averages, simple linear regression, multiple linear regression, classification and regression tree, and random forest. **The random forest model had the best performance, with a 46.4% RMSE improvement over the initial naive model.**

This model can be used for many things including predicting energy use in future years with temperature changes, predicting site EUI for new properties or ones with missing energy consumption calculations, or flagging potential errors in user inputs. All applications which can assist New York City in achieving it's energy reduction targets.

There are a few notable limitations. The first two have to do with the data quality. The weather data is not localized. It would be beneficial to have annual data by zip code rather than using a single value for the whole region. I suspect that improving this data quality would increase the importance of weather variables. I also did my best to remove errors when cleaning the data but there is no way to know if some of the remaining outliers are accurate or contain erroneous EUI calcs. Lastly, I ran in to a processing power roadblock when tuning and training the random forest model. A better computer would unlock more opportunities for optimizing the model.

Future work can use similar modeling approaches to try and parse out EUI by energy use type - namely electricity and natural gas. This would provide additional value for calculating and projecting GHG emissions over time.