NYC LL84 ML

Eden Axelrad

7/17/2022

Contents

SECTION 1: Introduction and Overview

- Data retrieved from NYC Open Data as part of Local Law 84 (LL84)
- Data described as Data and metrics on water and energy consumption in privately owned buildings over 25,000 ft2 and in City-owned buildings over 10,000 ft2.
- R script used to retrieve, wrangle, and clean the data frame used in this main report can be found in the repo, or accessed via this link.

SECTION 2: Setup

In this section we set up the initial data 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, including data cleaning, data exploration and visualization, insights gained, and modeling approach

SECTION 3.1: Exploratory Analysis

```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, ~
## $ 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.~
```

```r
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
```

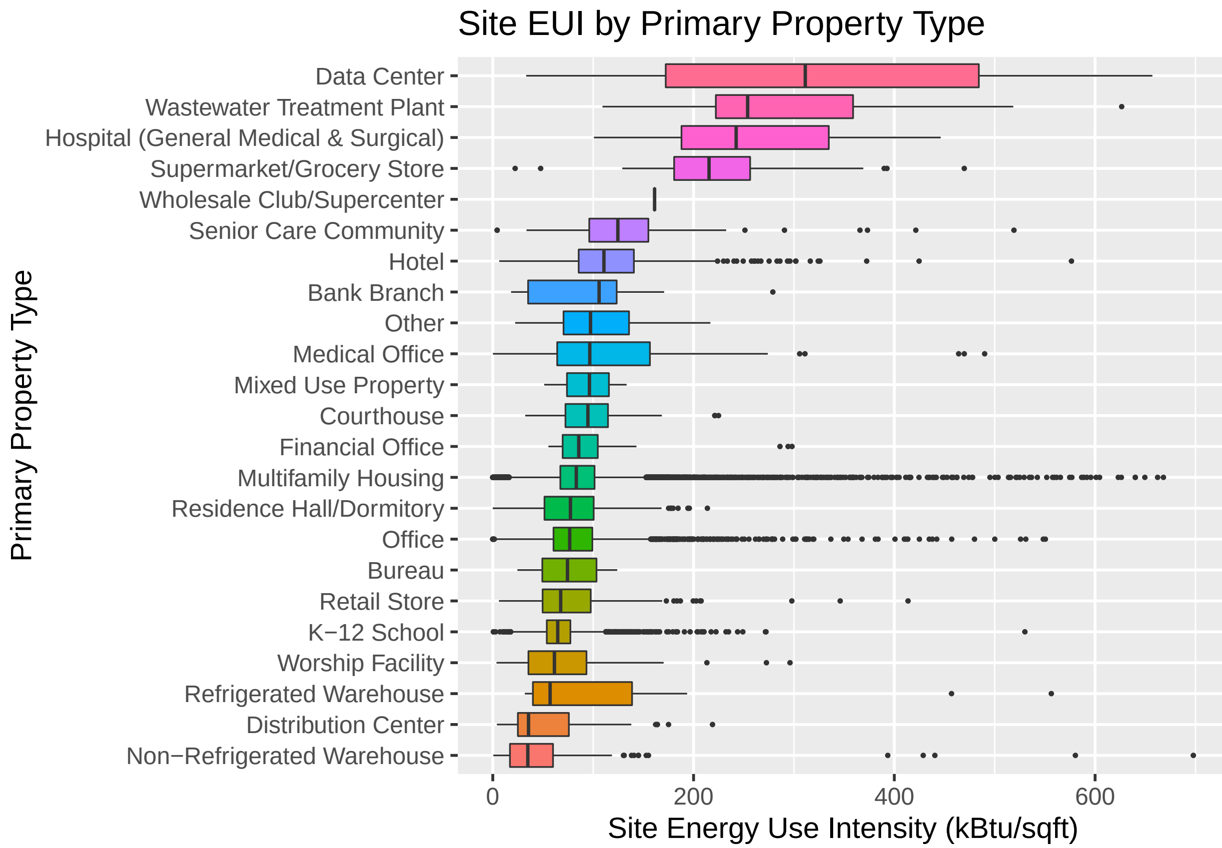
```r
# EUI and GHG by Property Type
NYC_LL84_ML %>%
  group_by("Property Type" = primary_property_type) %>%
  summarise("Site EUI (kBtu/sqft)" = round(mean(site_eui),
                                           digits = 1),
            Count = n())  %>%
  arrange(desc(Count)) %>%
  knitr::kable()
```

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

```r
# Explore EUI
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
```

```r
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()
```




```r
# 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
```

```r
NYC_LL84_ML %>%
  mutate(size_quartile = case_when(
    `property_gfa` > 113686 ~ 4,
    `property_gfa` > 63968 & `property_gfa` <= 113686 ~ 3,
    `property_gfa` > 40760 & `property_gfa` <= 63968 ~ 2,
    `property_gfa` <= 40760 ~ 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	100.8	4901
2	89.2	7407
3	80.9	11132

size_quartile	Site EUI (kBtu/sqft)	Count
4	81.5	11030

```
# Look at correlation by feature for total GHG emissions
# Examine EUI data and check for outliers
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

insert text

SECTION 3.2: Actual Methods

insert text

```
# 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)
```

```
# Start off with a naive model that uses the average rating to predict movie ratings
mu_hat <- mean(train_set$site_eui)
```

```r
# Save the model RMSE
rmse_naive <- RMSE(pred = mu_hat,
                   obs = test_set$site_eui)
```

Method #1: Naive Model

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, 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
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

# Save the model RMSE
rmse_eui_avg <- RMSE(pred_eui_avg,
                     obs = test_set$site_eui,
                     na.rm = TRUE)
```

```r
# Create a linear model
set.seed(92, sample.kind = "Rounding")
model_lr <- train(site_eui ~ energy_star_score,
               data = train_set,
               method = "lm")

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

# Save the model RMSE
rmse_lr <- RMSE(pred = pred_lr,
                    obs = test_set$site_eui,
                    na.rm = TRUE)
```

Method #3: Simple Linear Regression Using ENERGY STAR Score

```r
# Resampling: Cross-Validated (10 fold, repeated 3 times)
# No pre-processing
set.seed(92, sample.kind = "Rounding")
```

```
model_mlr <- train(site_eui ~ .,
                   trControl = trainControl(method = "cv"),
                   data = train_set,
                   preProc = c("center", "scale"),
                   method = "lm")

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

# Save the model RMSE
rmse_mlr <- RMSE(pred = pred_mlr,
                 obs = test_set$site_eui,
                 na.rm = TRUE)
```

Method #4: Multiple Linear Regression with all variables

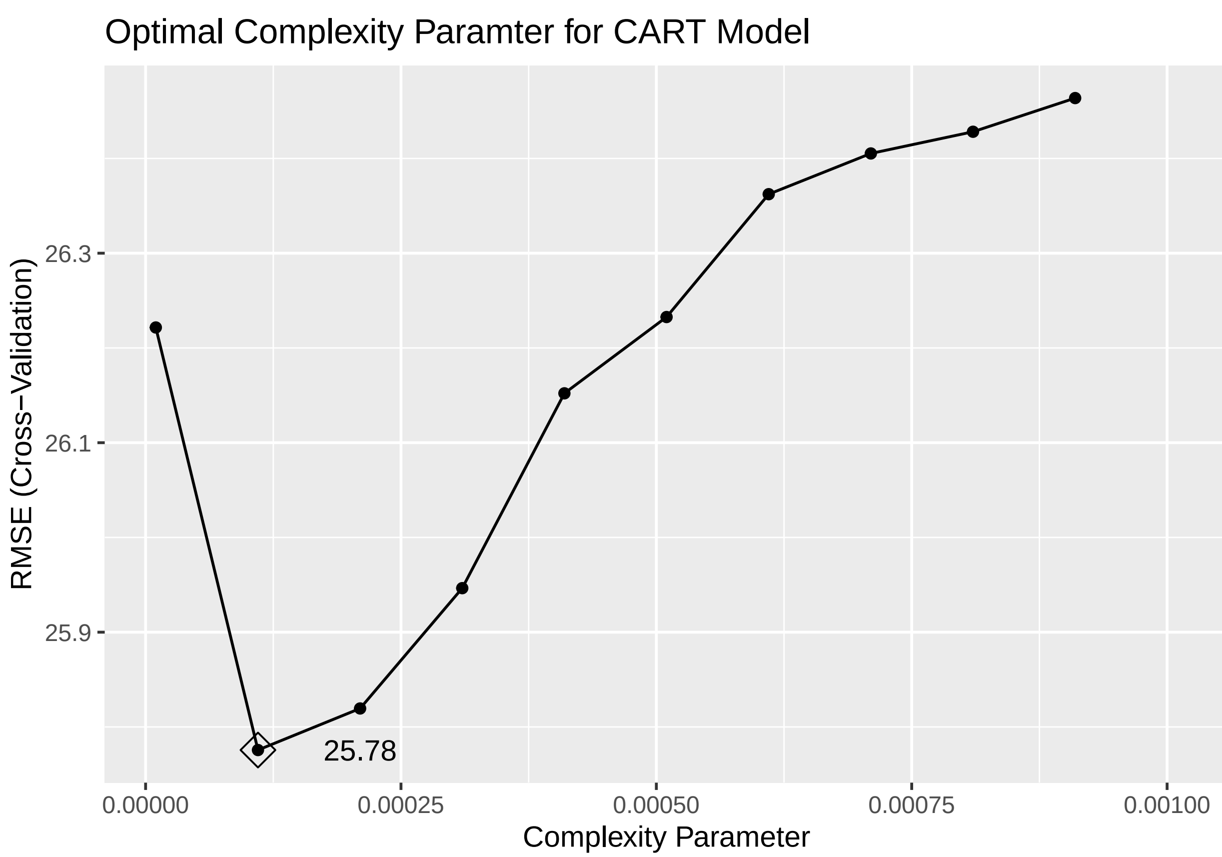
```
# Need to add CART to packages list
set.seed(92, sample.kind = "Rounding")
model_rpart <- train(site_eui ~ .,
                     data = train_set,
                     trControl = trainControl(method = "cv"),
                     method = "rpart",
                     preProc = c("center", "scale"),
                     tuneGrid = data.frame(cp = seq(.00001, .001, .0001)))

model_rpart
```

Method #5: Classification and Regression Tree (CART)

```
## CART
##
## 27575 samples
##    12 predictor
##
## Pre-processing: centered (33), scaled (33)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 24818, 24818, 24818, 24818, 24818, 24817, ...
## Resampling results across tuning parameters:
##
##   cp       RMSE      Rsquared   MAE
##   0.00001  26.22161  0.6577994  15.22196
##   0.00011  25.77563  0.6666304  14.71078
##   0.00021  25.81951  0.6649183  14.82929
##   0.00031  25.94650  0.6611772  14.92873
##   0.00041  26.15212  0.6557173  15.05774
##   0.00051  26.23263  0.6533668  15.14446
##   0.00061  26.36229  0.6498692  15.19644
##   0.00071  26.40526  0.6483165  15.24576
##   0.00081  26.42817  0.6476261  15.32687
```

```
##   0.00091  26.46371  0.6466478  15.39054
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.00011.
```

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

# Save the model RMSE
rmse_rpart <- RMSE(pred = pred_rpart,
                   obs = test_set$site_eui,
                   na.rm = TRUE)

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")
```



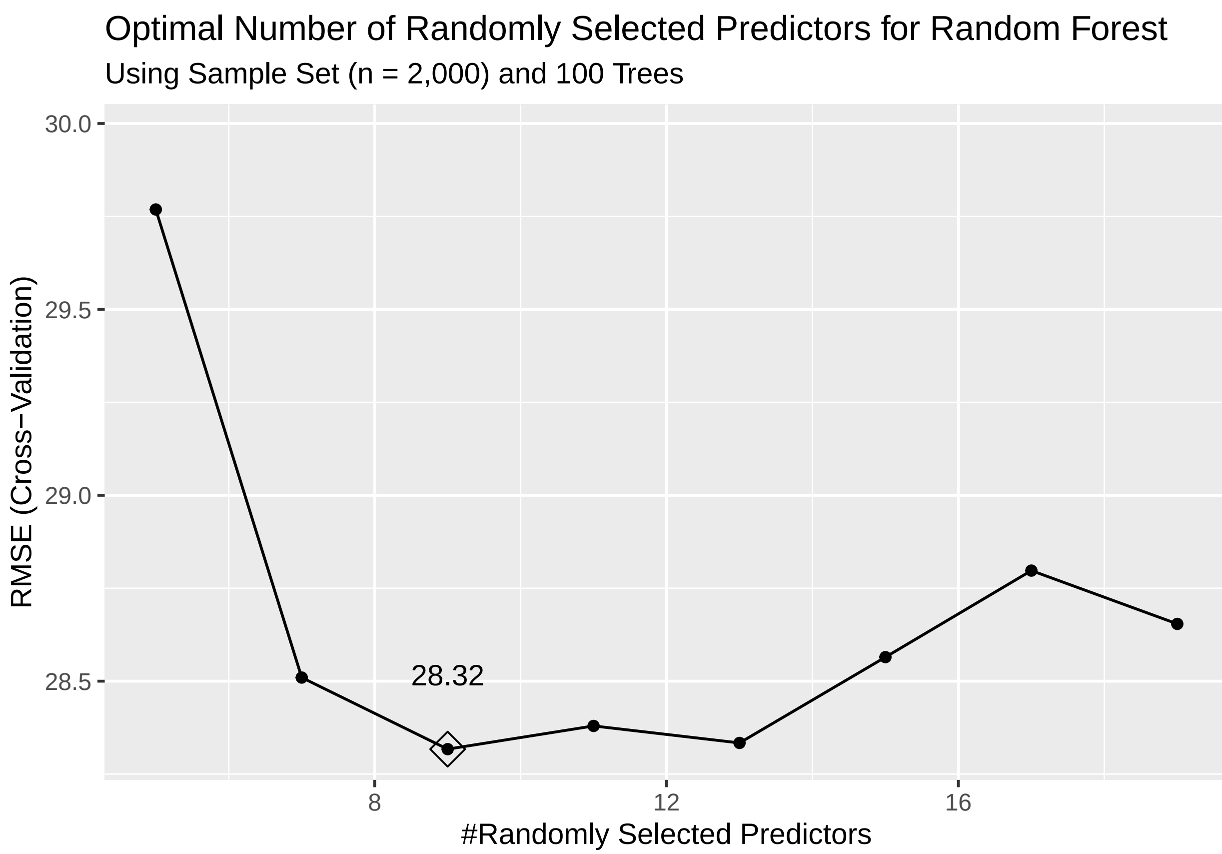

```
# Create a sample of the train set in order to speed up the training and tuning process
set.seed(92, sample.kind = "Rounding")
sample_index <- sample(nrow(train_set), 2000)
```

```r
sample_set <- train_set %>% slice(sample_index)


# Begin with sample set and minimal load test for optimal mtry
set.seed(92, sample.kind = "Rounding")
model_rf_default <- train(site_eui ~ .,
                   data = sample_set,
                   method = "rf",
                   ntree = 100,
                   tuneGrid = data.frame(mtry = seq(5, 20, 2)),
                   trControl = trainControl(method = "cv", number = 5))

model_rf_default
```

Method #6: Random Forest

```
## Random Forest
##
## 2000 samples
##   12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1600, 1600, 1600, 1600, 1600
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##    5    29.76887  0.5501355  15.99386
##    7    28.50973  0.5695297  15.03490
##    9    28.31715  0.5688013  14.77394
##   11    28.37960  0.5653051  14.71219
##   13    28.33389  0.5670283  14.65317
##   15    28.56469  0.5603048  14.84453
##   17    28.79745  0.5536640  14.86486
##   19    28.65406  0.5586150  14.83960
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 9.
```

```r
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")
```




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

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$bestTune[[1]]),
                    nSamp = 750)
  min(model_rf$results$RMSE)
})

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

```
# Create the actual rf model using our tuning parameters
# 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"))

model_rf
```

```
## Random Forest
##
## 27575 samples
##    12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 24818, 24818, 24818, 24818, 24818, 24817, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   23.11584  0.7323913  13.40331
##
## Tuning parameter 'mtry' was held constant at a value of 9
```

```r
# Generate predicted Site EUI using the test set
pred_rf <- predict(model_rf,
                   newdata = test_set)

rmse_rf <- RMSE(pred = pred_rf,
                obs = test_set$site_eui,
                na.rm = TRUE)
```

SECTION 3.3: Validation

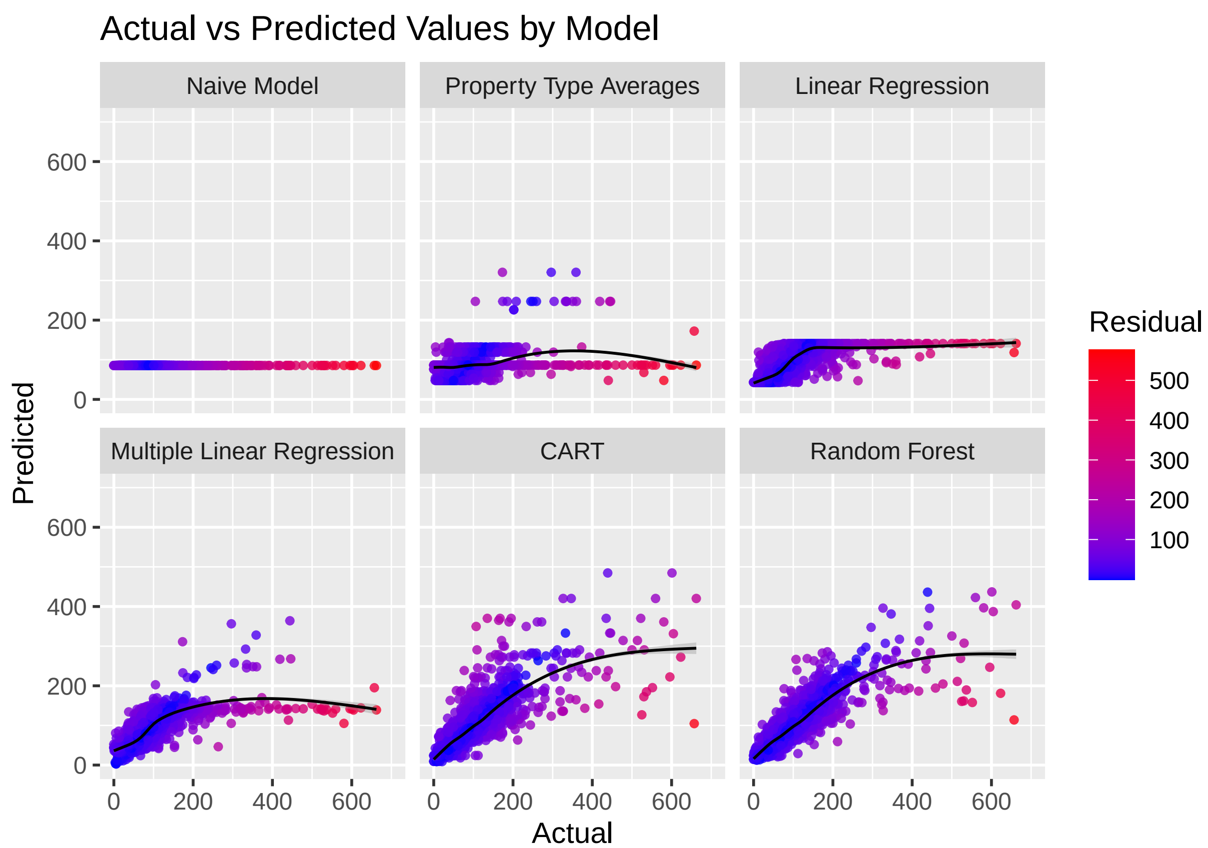
```r
# Generate predicted Site EUI using the test set
pred_validation <- predict(model_rf,
                           newdata = NYC_LL84_Validation)

# Save the model RMSE
rmse_validation <- RMSE(pred = pred_validation,
                   obs = NYC_LL84_Validation$site_eui,
                   na.rm = TRUE)
```

SECTION 4: Results

Comparing all models against the actual EUI values for reporting year 2019.

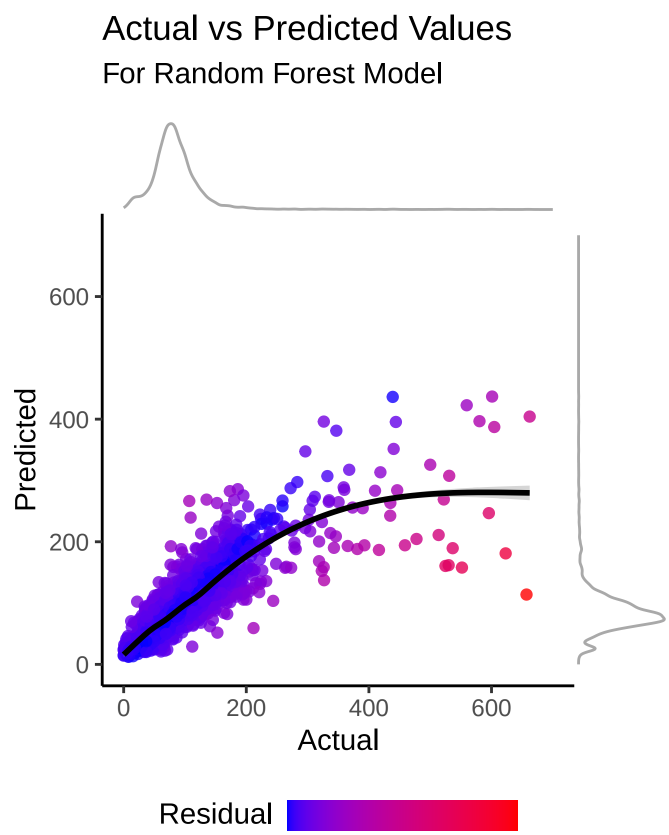
```r
Results <- cbind.data.frame(Actual = test_set$site_eui,
                            `Naive Model` = mu_hat,
                            `Property Type Averages` = pred_eui_avg,
```

```r
                            `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 factors based on order of our methods
Results$Model = factor(Results$Model, levels = c("Naive Model",
                                                 "Property Type Averages",
                                                 "Linear Regression",
                                                 "Multiple Linear Regression",
                                                 "CART",
                                                 "Random Forest"))

# 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(color = "Residual",
       title = "Actual vs Predicted Values by Model")
```




```r
# Actual vs predicted for random forest only, showing marginal distribution
ggExtra::ggMarginal(Results %>%
                      filter(Model == "Random Forest") %>%
                      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(color = "Residual",
                           title = "Actual vs Predicted Values",
                           subtitle = "For Random Forest Model") +
                      theme_classic() +
                      theme(legend.position = "bottom") +
                      guides(color = guide_colourbar(ticks = FALSE,
                                                     label = FALSE,
                                                     barheight = 0.8,
                                                     title.vjust = 0.9)),
                    type = "density", color = "darkgray")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



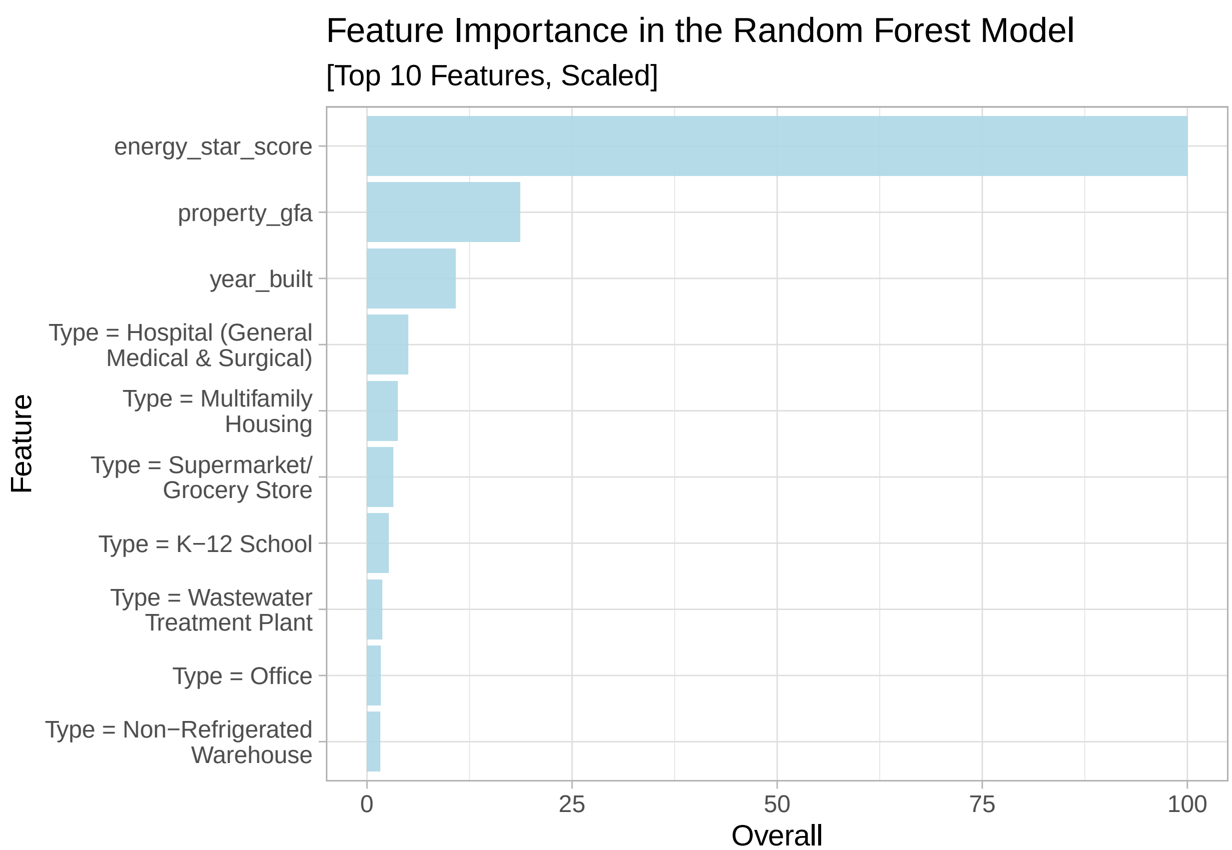
Final table depicting the RMSE results for all of our tested methods and the final validation.

```
# This section presents the modeling results and discusses the model performance
# Final results table with test on validation set
results_tbl <- 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"),
  RMSE = c(rmse_naive, rmse_eui_avg, rmse_lr, rmse_mlr,
           rmse_rpart, rmse_rf, rmse_validation)) %>%
  mutate(`RMSE Improvment` = scales::percent((RMSE-rmse_naive)/rmse_naive))

results_tbl %>%
  knitr::kable(align = c('l', 'l', 'c', 'c'), digits = 2)
```

Method	Model	RMSE	RMSE Improvment
Method #1	Naive Model	47.25	0.0%
Method #2	Property Type EUI Averages	44.98	-4.8%
Method #3	LR - ENERGY STAR Score	36.55	-22.6%
Method #4	Mulitple Linear Regression	33.98	-28.1%
Method #5	CART	28.08	-40.6%
Method #6	Random Forest	25.67	-45.7%
Validation	Random Forest	23.86	-49.5%

Examining the variable/feature importance in our final model - random forest.

```
importance_df <- varImp(model_rf)
importance_df <- importance_df[["importance"]] %>%
  rownames_to_column(var = "Feature")

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 5: Conclusion

insert text