NYC LL84 ML

Eden Axelrad

7/17/2022

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 here

SECTION 2: Setup

In this section we set up the initial data set

```
# 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

```
# 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.~
```

```
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
```

```
# 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()
```

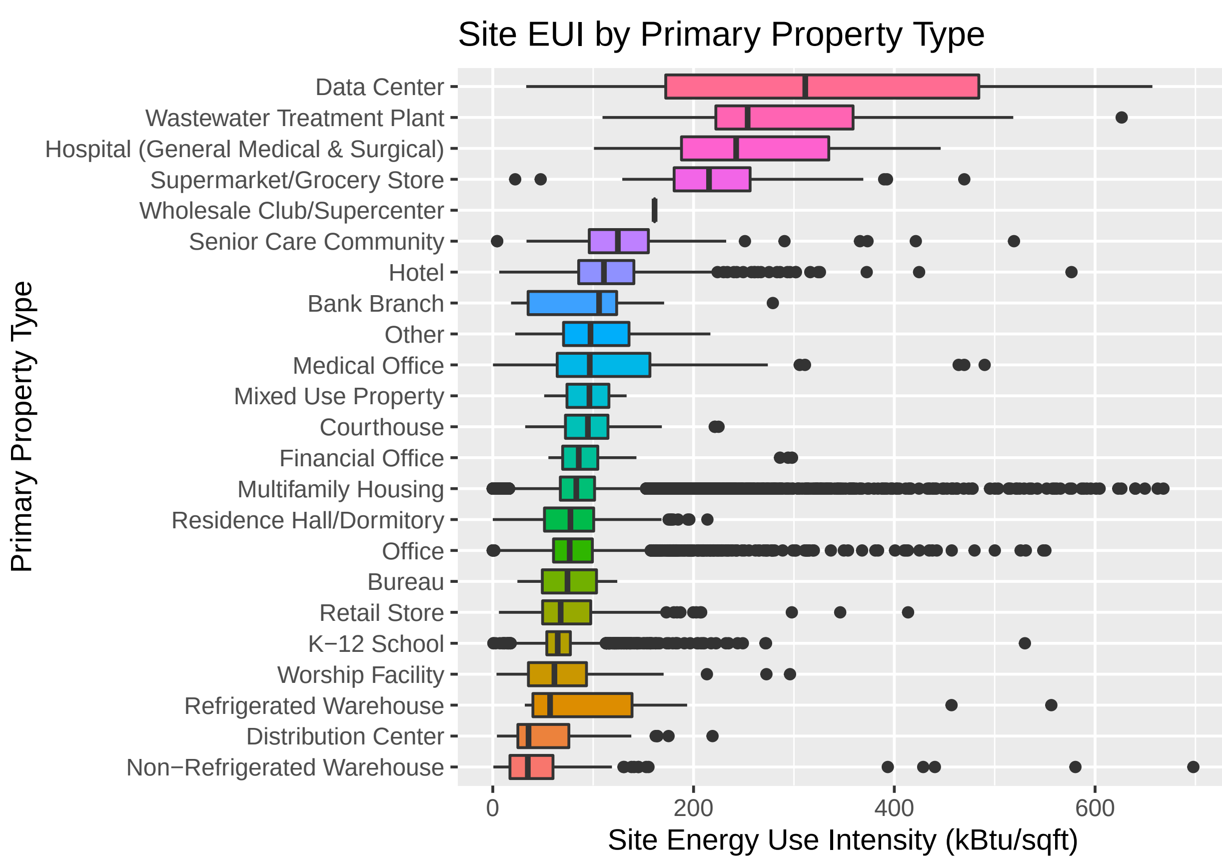
Property Type	Site EUI (kBtu/sqft)	Count
Multifamily Housing	86.7	25175
K-12 School	67.9	3881
Office	86.2	3169
Hotel	118.3	747
Residence Hall/Dormitory	74.5	300
Non-Refrigerated Warehouse	50.3	291
Senior Care Community	132.0	185
Distribution Center	51.8	135
Retail Store	83.2	130
Medical Office	125.7	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

Property Type	Site EUI (kBtu/sqft)	Count
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() +
  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
4	81.5	11030

```r
# 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

Feature	site_eui
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)
```

```


### Method #1: Naive Model
# Start off with a naive model that uses the average rating to predict movie ratings
mu_hat <- mean(train_set$site_eui)

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.

```
### Method #2: Using average EUI by property type

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

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

```r
### Method #3: Linear Regression with Energy Star Score
# Create a linear model
set.seed(92, sample.kind = "Rounding")
```

Method #3: Simple Linear Regression Using ENERGY STAR Score

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

```r
model_lr <- train(site_eui ~ energy_star_score,
              data = train_set,
              method = "lm")

model_lr
```

```
## Linear Regression
##
## 27575 samples
##     1 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 27575, 27575, 27575, 27575, 27575, 27575, ...
## Resampling results:
##
##   RMSE      Rsquared  MAE
##   33.38022  0.434889  18.93076
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# 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)
```

```r
### Method #4: Multiple Linear Regression with all variables
# Create a linear model
set.seed(92, sample.kind = "Rounding")
model_mlr <- train(site_eui ~ .,
                   trControl = trainControl(method = "repeatedcv", repeats = 3),
                   data = train_set,
                   method = "lm")

model_mlr
```

Method #4: Multiple Linear Regression with all variables

```
## Linear Regression
##
## 27575 samples
##    12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 24818, 24818, 24818, 24818, 24818, 24817, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   30.91294  0.5178551  17.60797
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# 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)
```

```r
### Method 6: glment
set.seed(92, sample.kind = "Rounding")
model_glmnet <- train(site_eui ~ .,
                   data = train_set,
                   trControl = trainControl(method = "cv", repeats = 3),
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = seq(.01, 1, .01)),
                   method = "glmnet")

model_glmnet
```

Method #5: GLM with Regularization

```
## glmnet
##
## 27575 samples
##    12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 24818, 24818, 24818, 24818, 24818, 24817, ...
## Resampling results across tuning parameters:
##
##   lambda  RMSE      Rsquared   MAE
##   0.01    30.89316  0.5185709  17.59792
```

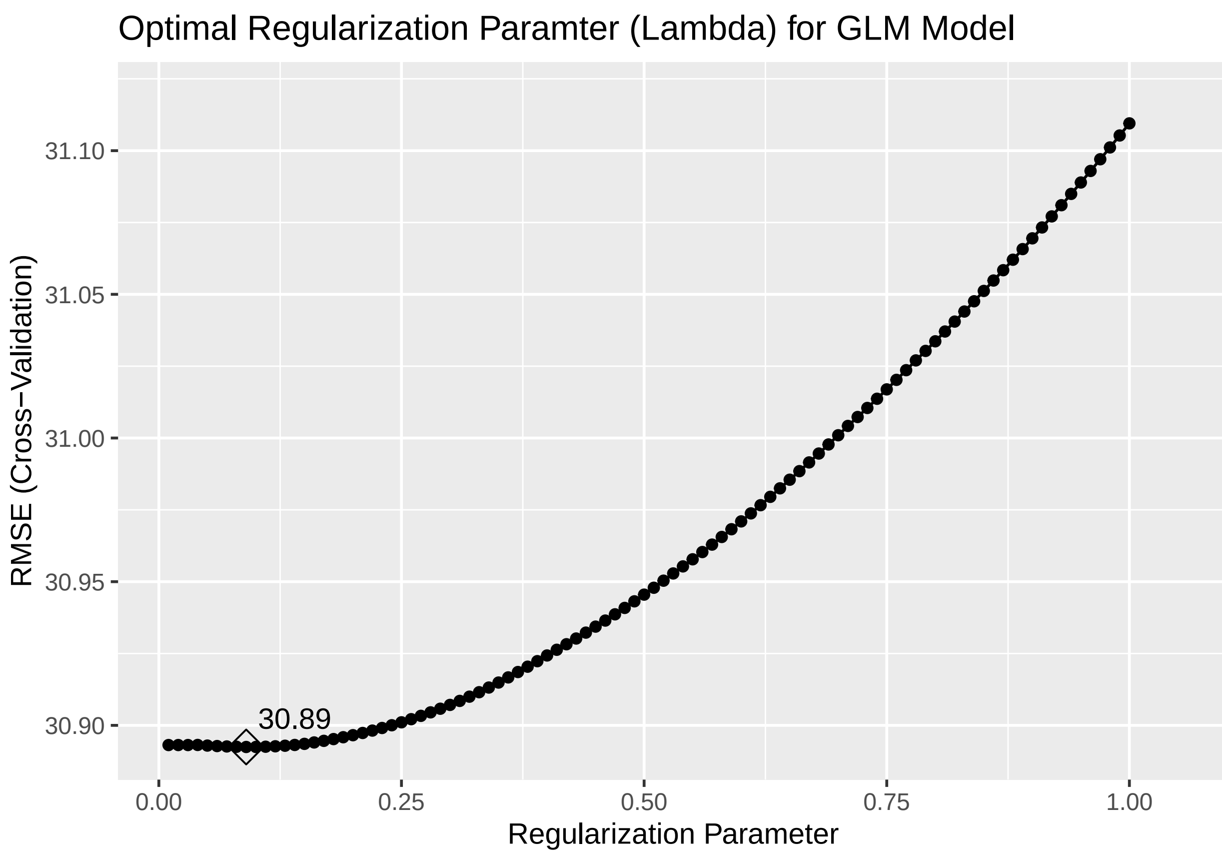
```
## 0.02 30.89316 0.5185709 17.59792
## 0.03 30.89316 0.5185709 17.59792
## 0.04 30.89318 0.5185702 17.59780
## 0.05 30.89297 0.5185757 17.59496
## 0.06 30.89280 0.5185796 17.59211
## 0.07 30.89266 0.5185831 17.58937
## 0.08 30.89255 0.5185865 17.58667
## 0.09 30.89247 0.5185893 17.58390
## 0.10 30.89248 0.5185901 17.58115
## 0.11 30.89257 0.5185889 17.57848
## 0.12 30.89271 0.5185866 17.57587
## 0.13 30.89291 0.5185830 17.57332
## 0.14 30.89318 0.5185779 17.57078
## 0.15 30.89356 0.5185702 17.56832
## 0.16 30.89405 0.5185593 17.56593
## 0.17 30.89461 0.5185470 17.56358
## 0.18 30.89522 0.5185334 17.56126
## 0.19 30.89587 0.5185190 17.55895
## 0.20 30.89658 0.5185034 17.55663
## 0.21 30.89735 0.5184862 17.55434
## 0.22 30.89818 0.5184675 17.55214
## 0.23 30.89909 0.5184471 17.54999
## 0.24 30.90004 0.5184255 17.54789
## 0.25 30.90106 0.5184023 17.54587
## 0.26 30.90215 0.5183774 17.54395
## 0.27 30.90331 0.5183508 17.54208
## 0.28 30.90453 0.5183226 17.54028
## 0.29 30.90579 0.5182936 17.53853
## 0.30 30.90711 0.5182629 17.53690
## 0.31 30.90851 0.5182306 17.53533
## 0.32 30.90999 0.5181957 17.53381
## 0.33 30.91154 0.5181593 17.53234
## 0.34 30.91316 0.5181209 17.53092
## 0.35 30.91489 0.5180793 17.52955
## 0.36 30.91669 0.5180361 17.52821
## 0.37 30.91855 0.5179913 17.52690
## 0.38 30.92042 0.5179468 17.52560
## 0.39 30.92234 0.5179008 17.52437
## 0.40 30.92432 0.5178532 17.52319
## 0.41 30.92630 0.5178059 17.52204
## 0.42 30.92823 0.5177609 17.52097
## 0.43 30.93022 0.5177143 17.51997
## 0.44 30.93227 0.5176660 17.51901
## 0.45 30.93436 0.5176170 17.51808
## 0.46 30.93647 0.5175677 17.51720
## 0.47 30.93864 0.5175168 17.51639
## 0.48 30.94087 0.5174642 17.51565
## 0.49 30.94316 0.5174100 17.51495
## 0.50 30.94550 0.5173548 17.51426
## 0.51 30.94790 0.5172979 17.51363
## 0.52 30.95035 0.5172394 17.51303
## 0.53 30.95287 0.5171793 17.51245
## 0.54 30.95532 0.5171216 17.51189
## 0.55 30.95779 0.5170636 17.51135
```

```
##   0.56    30.96032  0.5170039  17.51086
##   0.57    30.96291  0.5169427  17.51040
##   0.58    30.96556  0.5168797  17.51000
##   0.59    30.96826  0.5168155  17.50962
##   0.60    30.97098  0.5167506  17.50926
##   0.61    30.97377  0.5166841  17.50895
##   0.62    30.97662  0.5166160  17.50871
##   0.63    30.97952  0.5165462  17.50850
##   0.64    30.98249  0.5164747  17.50832
##   0.65    30.98549  0.5164026  17.50813
##   0.66    30.98847  0.5163314  17.50786
##   0.67    30.99150  0.5162586  17.50760
##   0.68    30.99460  0.5161842  17.50739
##   0.69    30.99775  0.5161081  17.50726
##   0.70    31.00096  0.5160305  17.50718
##   0.71    31.00420  0.5159521  17.50717
##   0.72    31.00731  0.5158782  17.50706
##   0.73    31.01046  0.5158034  17.50697
##   0.74    31.01366  0.5157270  17.50693
##   0.75    31.01692  0.5156491  17.50693
##   0.76    31.02024  0.5155696  17.50695
##   0.77    31.02360  0.5154885  17.50701
##   0.78    31.02700  0.5154068  17.50706
##   0.79    31.03031  0.5153283  17.50697
##   0.80    31.03366  0.5152487  17.50690
##   0.81    31.03706  0.5151677  17.50691
##   0.82    31.04051  0.5150851  17.50696
##   0.83    31.04402  0.5150009  17.50707
##   0.84    31.04758  0.5149153  17.50720
##   0.85    31.05119  0.5148283  17.50735
##   0.86    31.05481  0.5147412  17.50744
##   0.87    31.05840  0.5146551  17.50743
##   0.88    31.06205  0.5145675  17.50749
##   0.89    31.06574  0.5144786  17.50764
##   0.90    31.06948  0.5143881  17.50781
##   0.91    31.07328  0.5142962  17.50801
##   0.92    31.07712  0.5142029  17.50826
##   0.93    31.08102  0.5141081  17.50854
##   0.94    31.08496  0.5140120  17.50883
##   0.95    31.08892  0.5139151  17.50911
##   0.96    31.09294  0.5138169  17.50940
##   0.97    31.09701  0.5137172  17.50972
##   0.98    31.10112  0.5136160  17.51010
##   0.99    31.10528  0.5135134  17.51051
##   1.00    31.10950  0.5134093  17.51097
##
## Tuning parameter 'alpha' was held constant at a value of 1
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 1 and lambda = 0.09.
```

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

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

ggplot(model_glmnet, highlight = TRUE) +
  geom_text(position = position_nudge(x = .05, y = .01),
            aes(label = ifelse(lambda == model_glmnet$bestTune[2]$lambda,
                               round(min(model_glmnet$results$RMSE), digits = 2), ""))) +
  labs(title = "Optimal Regularization Paramter (Lambda) for GLM Model")
```




```r
### Method #5: Classification and Regression Tree (CART) with 'rpart'
# Need to add CART to packages list
set.seed(92, sample.kind = "Rounding")
model_rpart <- train(site_eui ~ .,
                     data = train_set,
                     trControl = trainControl(method = "repeatedcv", repeats = 3),
                     method = "rpart",
                     tuneGrid = data.frame(cp = seq(.00001, .001, .0001)))

model_rpart
```

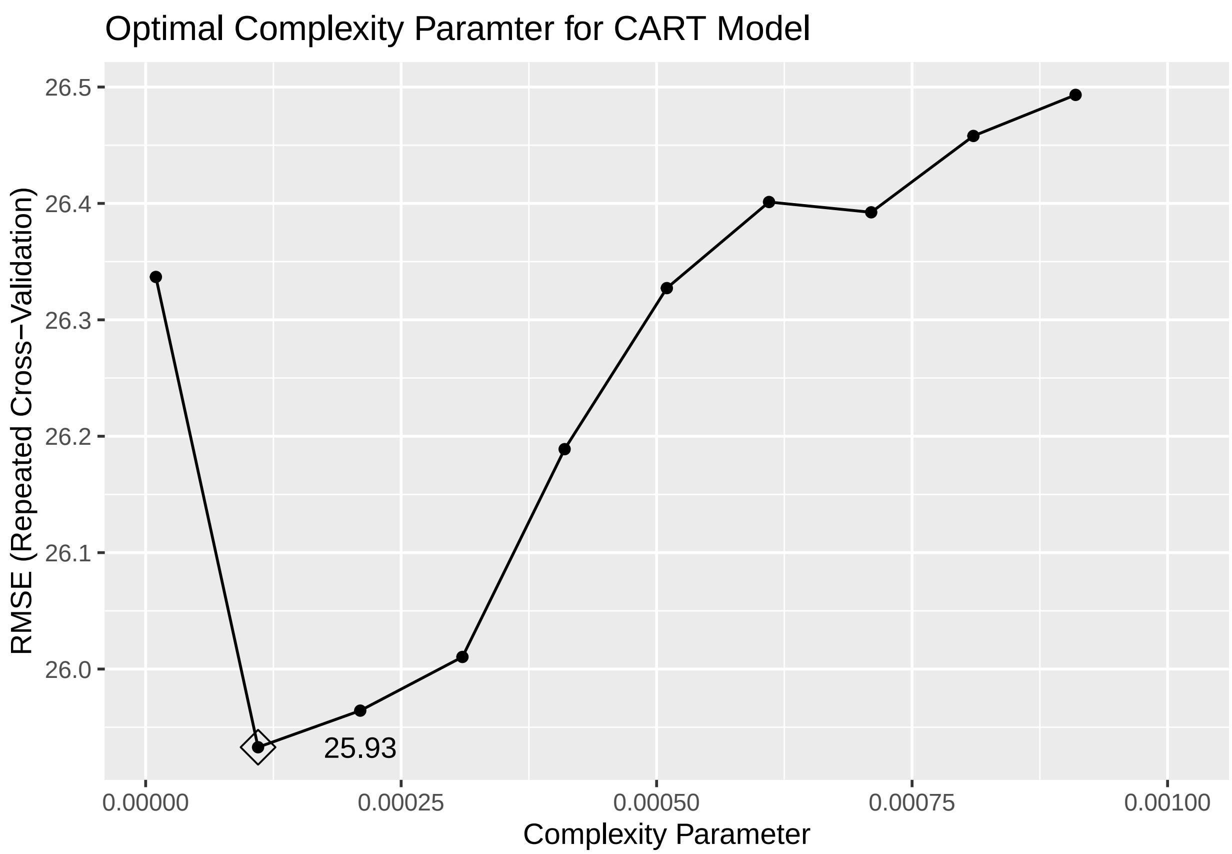
Method #5: Classification and Regression Tree (CART)

```
## CART
##
## 27575 samples
##    12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 24818, 24818, 24818, 24818, 24818, 24817, ...
## Resampling results across tuning parameters:
##
##   cp       RMSE      Rsquared   MAE
##   0.00001  26.33685  0.6563413  15.18282
##   0.00011  25.93282  0.6639376  14.72306
##   0.00021  25.96427  0.6624434  14.83284
##   0.00031  26.01039  0.6607973  14.92148
##   0.00041  26.18891  0.6558638  15.05889
##   0.00051  26.32725  0.6520565  15.14974
##   0.00061  26.40120  0.6499130  15.18645
##   0.00071  26.39238  0.6498339  15.24356
##   0.00081  26.45797  0.6480457  15.33270
##   0.00091  26.49324  0.6469890  15.39627
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.00011.
```

```r
# 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")
```



SECTION 3.3: Validation

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

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

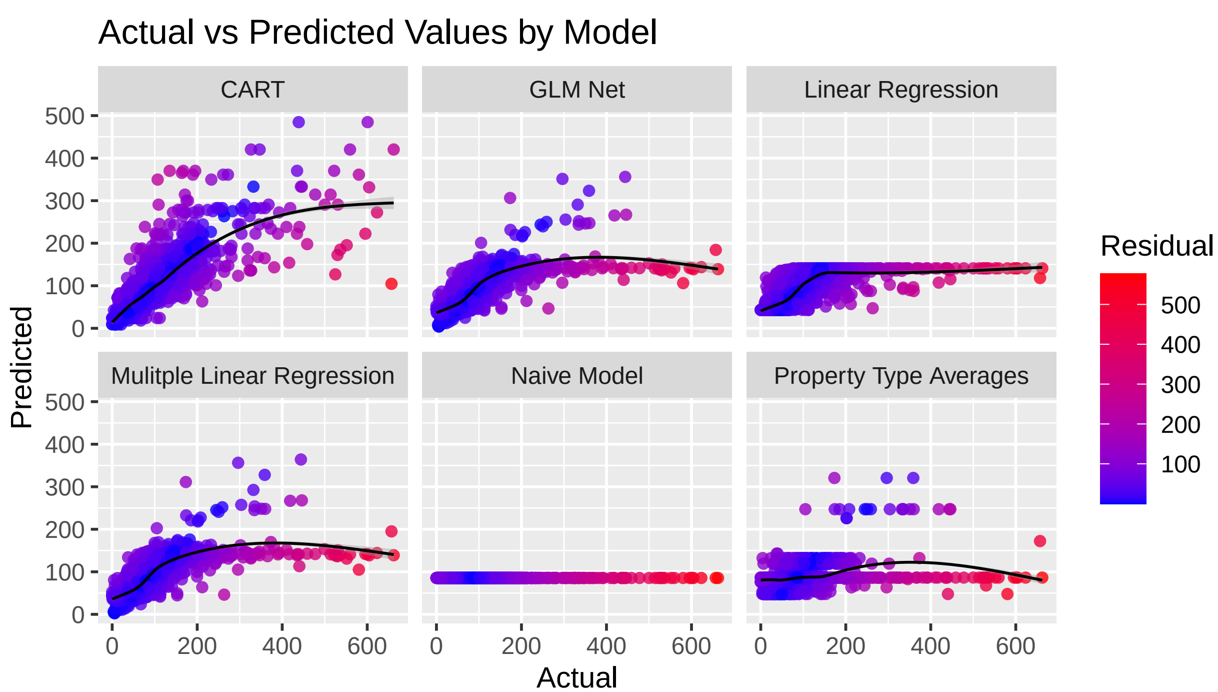
SECTION 4: Results

insert text

```
Results <- cbind.data.frame(Actual = test_set$site_eui,
                            `Naive Model` = mu_hat,
                            `Property Type Averages` = pred_eui_avg,
                            `Linear Regression` = pred_lr,
                            `Mulitple Linear Regression` = pred_mlr,
                            `GLM Net` = pred_glmnet,
                            `CART` = pred_rpart) %>%
  pivot_longer(cols = 2:7,
```

```
               values_to = "Predicted",
               names_to = "Model") %>%
  mutate(Residual = Actual-Predicted) %>%
  select(Model, Actual, Predicted, Residual)


# Actual vs predicted by model without outliers
Results %>%
  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", size = .5) +
  coord_equal() +
  facet_wrap(~Model) +
  labs(color = "Residual",
       title = "Actual vs Predicted Values by Model")
```




```
# 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", "EUI Averages", "LR - Energy Star",
            "Mulitple Linear Regression", "GLM Net", "CART", "CART"),
  RMSE = c(rmse_naive, rmse_eui_avg, rmse_lr, rmse_mlr, rmse_glmnet, rmse_rpart, rmse_validation)) %>%
  mutate(`RMSE Improvment` = scales::percent((RMSE-rmse_naive)/rmse_naive))
```

```r
results_tbl %>%
  mutate(RMSE = round(RMSE, digits = 2)) %>%
  knitr::kable()
```

Method	Model	RMSE	RMSE Improvment
Method #1	Naive Model	47.25	0.000%
Method #2	EUI Averages	44.98	-4.792%
Method #3	LR - Energy Star	36.55	-22.636%
Method #4	Mulitple Linear Regression	33.98	-28.073%
Method #5	GLM Net	34.00	-28.036%
Method #6	CART	28.08	-40.573%
Validation	CART	26.90	-43.060%

SECTION 5: Conclusion

insert text