DataScienceProj

Ethan Jose

2023-12-14

```r
# package loads
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(dplyr)
library(arrow)
```

```
##
## Attaching package: 'arrow'
##
## The following object is masked from 'package:lubridate':
##
##     duration
##
## The following object is masked from 'package:utils':
##
##     timestamp
```

```r
library(purrr)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
library(httr)
```

```
##
## Attaching package: 'httr'
##
## The following object is masked from 'package:caret':
##
##     progress
```

```r
library(e1071)
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```r
library(ggplot2)
library(rpart)
library(rpart.plot)
library(ipred)
library(partykit)
```

```
## Loading required package: grid
## Loading required package: libcoin
## Loading required package: mvtnorm
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
# data read ins


# sample data for testing
sample_parquet<- read_parquet("https://intro-datascience.s3.us-east-2.amazonaws.com/SC-data/2023-houseDa
#sample_weather_data<- read.csv("https://intro-datascience.s3.us-east-2.amazonaws.com/SC-data/weather/2

# get s3 directories for weather and individual house datasets and weather
house_file_url <-"https://intro-datascience.s3.us-east-2.amazonaws.com/SC-data/2023-houseData/"
weather_file_url <- "https://intro-datascience.s3.us-east-2.amazonaws.com/SC-data/weather/2023-weather-


# get our static dataset

#fname <- file.choose()
static_house_data <- fread("/Users/ethan/Downloads/static_house_info_387.csv")
```

```r
final_data <- data.frame()

for (i in 1:nrow(static_house_data)) {
  bldg_id <- static_house_data$bldg_id[i]
  county <- static_house_data$in.county[i]

  # Load parquet data
  parquet_url <- paste0("https://intro-datascience.s3.us-east-2.amazonaws.com/SC-data/2023-houseData/",
  parquet_data <- arrow::read_parquet(parquet_url)

  parquet_data <- parquet_data %>% filter(month(time) == 7)

  # Load weather data
  weather_file_url <- "https://intro-datascience.s3.us-east-2.amazonaws.com/SC-data/weather/2023-weathe
  file_name <- paste0(county, ".csv")
  weather_data <- fread(paste0(weather_file_url, "/", file_name))
  weather_data <- as.data.frame(weather_data)

  weather_data$date_time <- as.POSIXct(weather_data$date_time, format = "%Y-%m-%d %H:%M:%S")
  weather_data <- weather_data %>% filter(month(date_time) == 7)
```

```
  # Combine data
  combined_data <- cbind(static_house_data[i, ], parquet_data, weather_data)



  # Append to final_data
  final_data <- rbind(final_data, combined_data)
}
```

```
# clean up names in weather data
final_data$temperature <- final_data$`Dry Bulb Temperature [°C]`
final_data$wind_speed <- final_data$`Wind Speed [m/s]`
final_data$wind_dir <- final_data$`Wind Direction [Deg]`
final_data$horizontal_rad <- final_data$`Global Horizontal Radiation [W/m2]`
final_data$normal_rad <- final_data$`Direct Normal Radiation [W/m2]`
final_data$diffuse_rad <- final_data$`Diffuse Horizontal Radiation [W/m2]`
final_data$diffuse_rad <- final_data$`Diffuse Horizontal Radiation [W/m2]`
```

```
########## get total energy consumption

########## check if there are any columns without any variance/all zeroes

# Check if any column contains all zeroes
zero_columns <- colSums(final_data == 0) == nrow(final_data)

# get all column names- using sample parquet because these are columns
# we are worried about, everything else we want to keep for now
all_column_names <- colnames(sample_parquet)

# Create a list of column names with all zeroes
columns_with_all_zeroes <- names(zero_columns[zero_columns])

# remove NA in list
columns_with_all_zeroes <- columns_with_all_zeroes[2:15]

# Create a list of column names with values not all zeroes
columns_with_values_not_all_zeroes <- setdiff(all_column_names, columns_with_all_zeroes)

# Print the list of columns with values not all zeroes
if (length(columns_with_values_not_all_zeroes) > 0) {
  print("Columns with values not all zeroes:")
  print(columns_with_values_not_all_zeroes)
} else {
  print("All columns contain all zeroes.")
}
```

```
## [1] "Columns with values not all zeroes:"
##  [1] "out.electricity.ceiling_fan.energy_consumption"
##  [2] "out.electricity.clothes_dryer.energy_consumption"
##  [3] "out.electricity.clothes_washer.energy_consumption"
##  [4] "out.electricity.cooling_fans_pumps.energy_consumption"
##  [5] "out.electricity.cooling.energy_consumption"
##  [6] "out.electricity.dishwasher.energy_consumption"
```

```
##  [7] "out.electricity.freezer.energy_consumption"
##  [8] "out.electricity.heating_fans_pumps.energy_consumption"
##  [9] "out.electricity.heating.energy_consumption"
## [10] "out.electricity.hot_tub_heater.energy_consumption"
## [11] "out.electricity.hot_tub_pump.energy_consumption"
## [12] "out.electricity.hot_water.energy_consumption"
## [13] "out.electricity.lighting_exterior.energy_consumption"
## [14] "out.electricity.lighting_garage.energy_consumption"
## [15] "out.electricity.lighting_interior.energy_consumption"
## [16] "out.electricity.mech_vent.energy_consumption"
## [17] "out.electricity.plug_loads.energy_consumption"
## [18] "out.electricity.pool_heater.energy_consumption"
## [19] "out.electricity.pool_pump.energy_consumption"
## [20] "out.electricity.pv.energy_consumption"
## [21] "out.electricity.range_oven.energy_consumption"
## [22] "out.electricity.refrigerator.energy_consumption"
## [23] "out.electricity.well_pump.energy_consumption"
## [24] "out.natural_gas.fireplace.energy_consumption"
## [25] "out.natural_gas.grill.energy_consumption"
## [26] "out.natural_gas.hot_tub_heater.energy_consumption"
## [27] "out.natural_gas.lighting.energy_consumption"
## [28] "out.natural_gas.pool_heater.energy_consumption"
## [29] "time"
```

```r
# clean up our list to remove the time column
columns_with_values_not_all_zeroes <- columns_with_values_not_all_zeroes[1:28]



# since these are all part of the hourly energy consumption we can
# calculate the hourly total based on all of the other columns
# in each parquet data set (the hourly energy consumption is the goal)

# add all of the rows together and create a new column for energy consumption
column_name <- "total_electric_energy_used"
final_data.1 <- final_data[, (column_name) := rowSums(.SD), .SDcols = columns_with_values_not_all_zeroes
```

```r
# make the window types a ranked factor

# my interpretation of window rankings I may consider using less than 10
# such as a value of 1-5 but will explore later if needed
ranked_window_types <- c("Triple, Low-E, Non-metal, Air, L-Gain",
                  "Double, Low-E, Non-metal, Air, M-Gain",
                  "Double, Clear, Non-metal, Air, Exterior Clear Storm",
                  "Double, Clear, Non-metal, Air",
                  "Single, Clear, Non-metal, Exterior Clear Storm",
                  "Single, Clear, Non-metal",
                  "Double, Clear, Metal, Air, Exterior Clear Storm",
                  "Double, Clear, Metal, Air",
                  "Single, Clear, Metal, Exterior Clear Storm",
                  "Single, Clear, Metal")

window_ranking <- factor(final_data.1$in.windows, levels = ranked_window_types)
final_data.1$window_ranking_integer <- as.integer(window_ranking)
```

I want to decipher the different windows into rankings based on their value in in.windows - which likely would impact energy consumption: from my research I would rank as follows: (keep in mind this is not my area of expertise and I would recommend a consultant to review these to improve the modeling)

From stongest efficiency to weakest:

Triple, Low-E, Non-metal, Air, L-Gain Double, Low-E, Non-metal, Air, M-Gain Double, Clear, Non-metal, Air, Exterior Clear Storm Double, Clear, Non-metal, Air Single, Clear, Non-metal, Exterior Clear Storm Single, Clear, Non-metal Double, Clear, Metal, Air, Exterior Clear Storm Double, Clear, Metal, Air Single, Clear, Metal, Exterior Clear Storm Single, Clear, Metal

I think this is a valid method however I think my own ordinal rankings may hurt the models performance - i one hot encoding may be better if my rankings are not correct

```r
final_data.1$window_areas <- as.factor(final_data.1$in.window_areas)
ranked_window_ratio <- c("F6 B6 L6 R6", "F9 B9 L9 R9", "F12 B12 L12 R12",
                  "F15 B15 L15 R15", "F18 B18 L18 R18" , "F30 B30 L30 R30")
final_data.1$in.window_areas <- factor(final_data.1$in.window_areas, levels = ranked_window_ratio)
final_data.1$windowAreaRanking <- as.integer(final_data.1$in.window_areas)
```

In terms of window area, I can safely assume that the higher the F, B, L, and R values the larger the wall to window area ratio so I can reorder them as such

```r
final_data.1$in.income <- as.factor(final_data.1$in.income)
rankedIncome <- c("<10000", "10000-14999", "15000-19999", "20000-24999", "25000-29999", "30000-34999",
                  "40000-44999", "45000-49999", "50000-59999", "60000-69999", "70000-79999", "80000-9999
                  "100000-119999", "120000-139999",  "140000-159999", "160000-179999",  "180000-199999",
final_data.1$in.income <- factor(final_data.1$in.income, levels = rankedIncome)
final_data.1$rankedIncome <- as.integer(final_data.1$in.income)
```

Similar to windows and window area, I can turn income into a ranked factor for each income bracket

```r
final_data.1$in.hvac_cooling_efficiency <- as.factor(final_data.1$in.hvac_cooling_efficiency)
efficiency_ranking <- c("None", "AC, SEER 8", "AC, SEER 10", "Room AC, EER 10.7", "Heat Pump", "AC, SEE
final_data.1$in.hvac_cooling_efficiency <- factor(final_data.1$in.hvac_cooling_efficiency,
                                                  levels = efficiency_ranking)
final_data.1$rankedHVAC_Efficiency <- as.integer(final_data.1$in.hvac_cooling_efficiency)
```

ranking cooling efficiency based on my research and assumptions of HVAC systems

```r
final_data.1$in.ducts <- as.factor(final_data.1$in.ducts)
rankedDucts <- c("None", "30% Leakage, Uninsulated", "20% Leakage, Uninsulated",
                  "10% Leakage, Uninsulated", "30% Leakage, R-4", "20% Leakage, R-4",
                  "10% Leakage, R-4", "30% Leakage, R-6", "20% Leakage, R-6",
                  "10% Leakage, R-6", "30% Leakage, R-8", "20% Leakage, R-8",
                  "10% Leakage, R-8" )
final_data.1$in.ducts <- factor(final_data.1$in.ducts, levels = rankedDucts)
final_data.1$rankedDucts <- as.integer(final_data.1$in.ducts)
```

Ranking duct leakage based on leakage first then insulation level

```r
final_data.1$in.usage_level <- as.factor(final_data.1$in.usage_level)
rankedUsage <- c("Low", "Medium", "High")
final_data.1$in.usage_level <- factor(final_data.1$in.usage_level, levels = rankedUsage)
final_data.1$usageLevel <- as.integer(final_data.1$in.usage_level)
```

ranking usage level of appliances against national average (this one is easy)

```
final_data.1$in.insulation_wall <- as.factor(final_data.1$in.insulation_wall)
insulation_ranking <- c("Wood Stud, Uninsulated", "Brick, 12-in, 3-wythe, Uninsulated", "Wood Stud, R-7
                        "Brick, 12-in, 3-wythe, R-7", "Wood Stud, R-11", "CMU, 6-in Hollow, R-11",
                        "Brick, 12-in, 3-wythe, R-11", "Wood Stud, R-15", "CMU, 6-in Hollow, R-15",
                        "Brick, 12-in, 3-wythe, R-19", "Wood Stud, R-19", "CMU, 6-in Hollow, R-19")
final_data.1$in.insulation_wall <- factor(final_data.1$in.insulation_wall,
                                          levels = insulation_ranking)
final_data.1$wallInsulRank <- as.integer(final_data.1$in.insulation_wall)
```

my ranking for wall insulation

```
final_data.1$in.insulation_ceiling <- as.factor(final_data.1$in.insulation_ceiling)
ceiling_insulation_ranking <- c("Uninsulated", "R-7", "R-13", "R-19", "R-30", "R-38", "R-49")
final_data.1$in.insulation_ceiling <- factor(final_data.1$in.insulation_ceiling,
                                             levels = ceiling_insulation_ranking)
final_data.1$ceilingInsulRank <- as.integer(final_data.1$in.insulation_ceiling)
```

rank for ceiling insulation

```
final_data.1$in.insulation_floor <- as.factor(final_data.1$in.insulation_floor)
floor_insulation_ranking <- c("None", "Uninsulated", "Ceiling R-13", "Ceiling R-19")
final_data.1$in.insulation_floor <- factor(final_data.1$in.insulation_floor,
                                           levels = floor_insulation_ranking)
final_data.1$floorInsulRank <- as.integer(final_data.1$in.insulation_floor)
```

floor insulation is either none or uninsulated so we wont use this

```
final_data.1$in.insulation_slab <- as.factor(final_data.1$in.insulation_slab)
slab_ranking <- c("None", "Uninsulated", "2ft R5 Under, Horizontal", "2ft R5 Perimeter, Vertical",
                  "2ft R10 Under, Horizontal", "2ft R10 Perimeter, Vertical")
final_data.1$in.insulation_slab <- factor(final_data.1$in.insulation_slab,
                                          levels = slab_ranking)
final_data.1$slabInsulRank <- as.integer(final_data.1$in.insulation_slab)
```

other insulations are mostly useless: foundation wall has incomplete data,

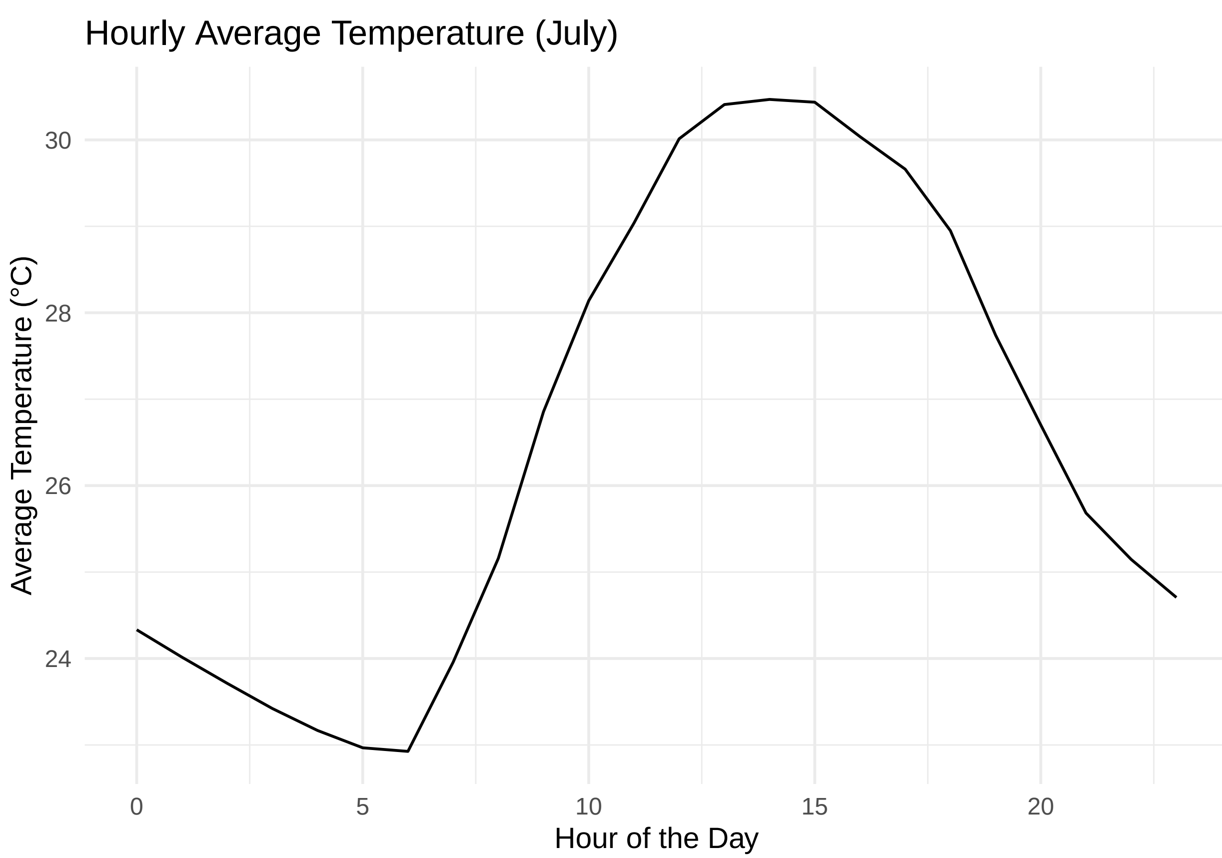
```
final_data.1$in.geometry_wall_type <- as.factor(final_data.1$in.geometry_wall_type)
wall_type_rank <- c("Steel Frame", "Concrete", "Brick", "Wood Frame")
final_data.1$in.geometry_wall_type <- factor(final_data.1$in.geometry_wall_type,
                                             levels = wall_type_rank)
final_data.1$wallTypeRank <- as.integer(final_data.1$in.geometry_wall_type)
```

```
final_data.2 <- final_data.1 %>%
  mutate(month = lubridate::month(time),
        day = lubridate::day(time),
        hour = lubridate::hour(time)) %>%
  select(bldg_id, day, hour, temperature, wind_speed,
         horizontal_rad, normal_rad, diffuse_rad, window_ranking_integer,
         windowAreaRanking, rankedIncome, in.bedrooms, rankedDucts,
```

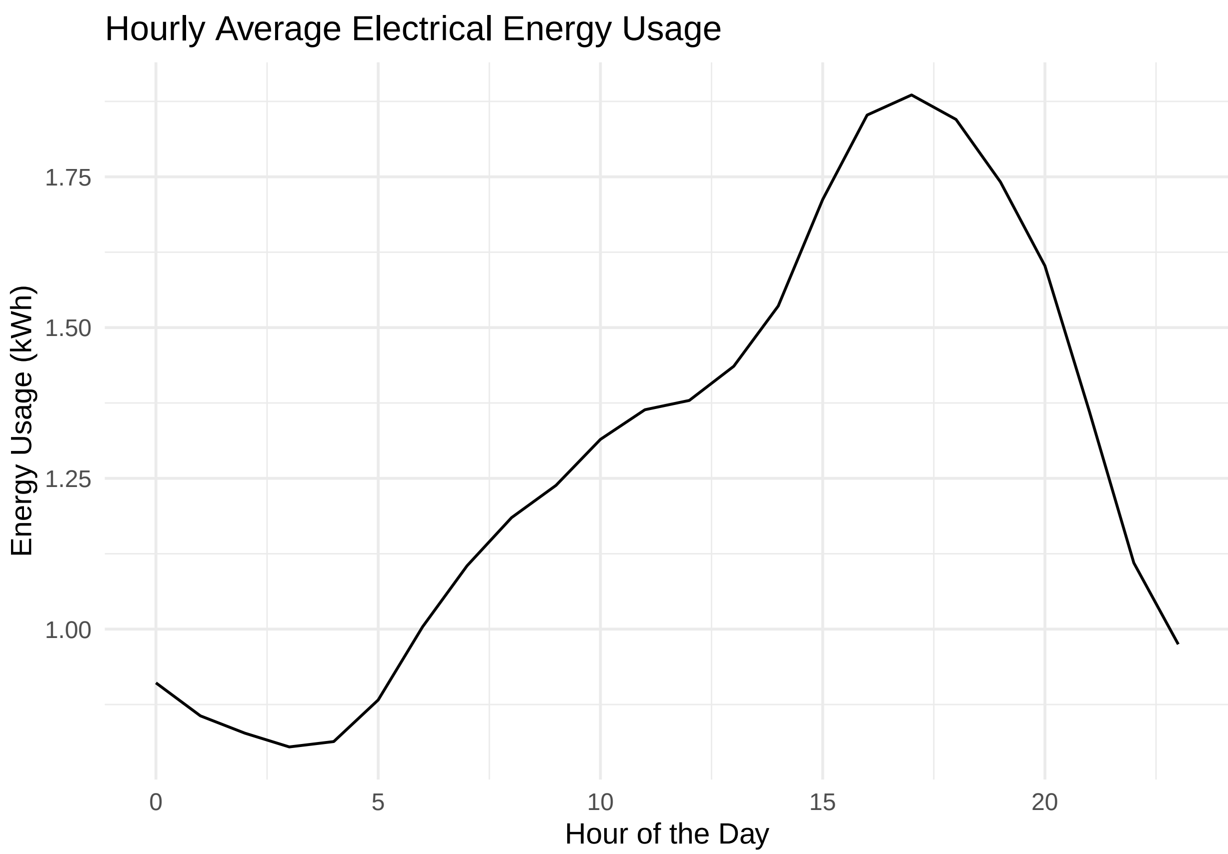
```
         rankedHVAC_Efficiency, usageLevel, in.weather_file_latitude, in.weather_file_longitude,
         wallInsulRank, ceilingInsulRank, floorInsulRank, slabInsulRank,wallTypeRank,
         in.geometry_stories, in.county, in.sqft, total_electric_energy_used)


# we want to predict something and we already know total electric energy use
# so we can base this on energy use per square foot instead to apply our models
# to an unknown target variable
final_data.3 <- final_data.2 %>%
  mutate(energy_by_sqft = total_electric_energy_used / in.sqft)
```

DESCRIPTIVE PLOTS

```
avg_temp <- final_data.3 %>%
  group_by(hour) %>%
  summarize(avgTemp = mean(temperature))

# plot avg hour temp
ggplot(avg_temp, aes(x = hour, y = avgTemp)) +
  geom_line() +
  labs(title = "Hourly Average Temperature (July)",
       x = "Hour of the Day",
       y = "Average Temperature (°C)") +
  theme_minimal()
```

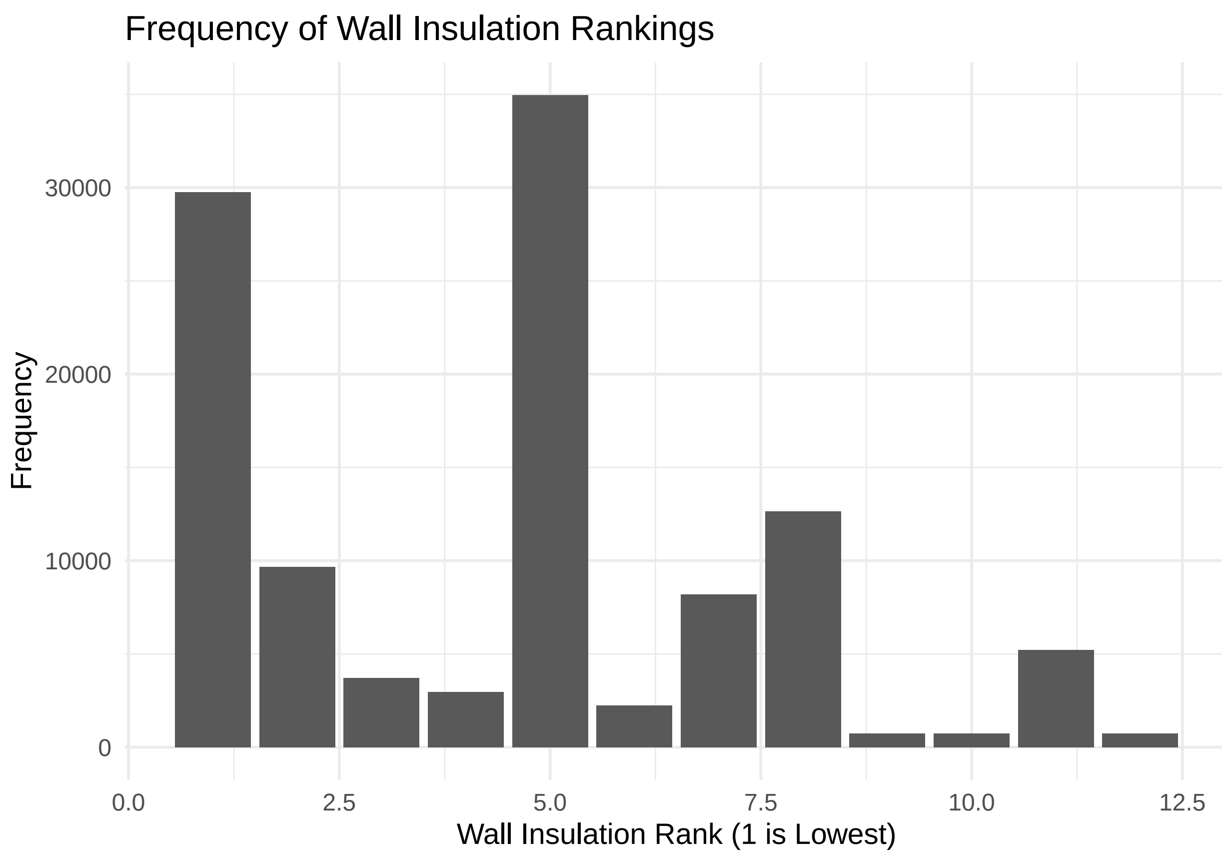


```r
avg_hourly_energy <- final_data.3 %>%
  group_by(hour) %>%
  summarize(avg_Hourly_Energy_Usage = mean(total_electric_energy_used))

# average electrical energy usage
ggplot(avg_hourly_energy, aes(x = hour, y = avg_Hourly_Energy_Usage)) +
  geom_line() +
  labs(title = "Hourly Average Electrical Energy Usage",
       x = "Hour of the Day",
       y = "Energy Usage (kWh)") +
  theme_minimal()
```

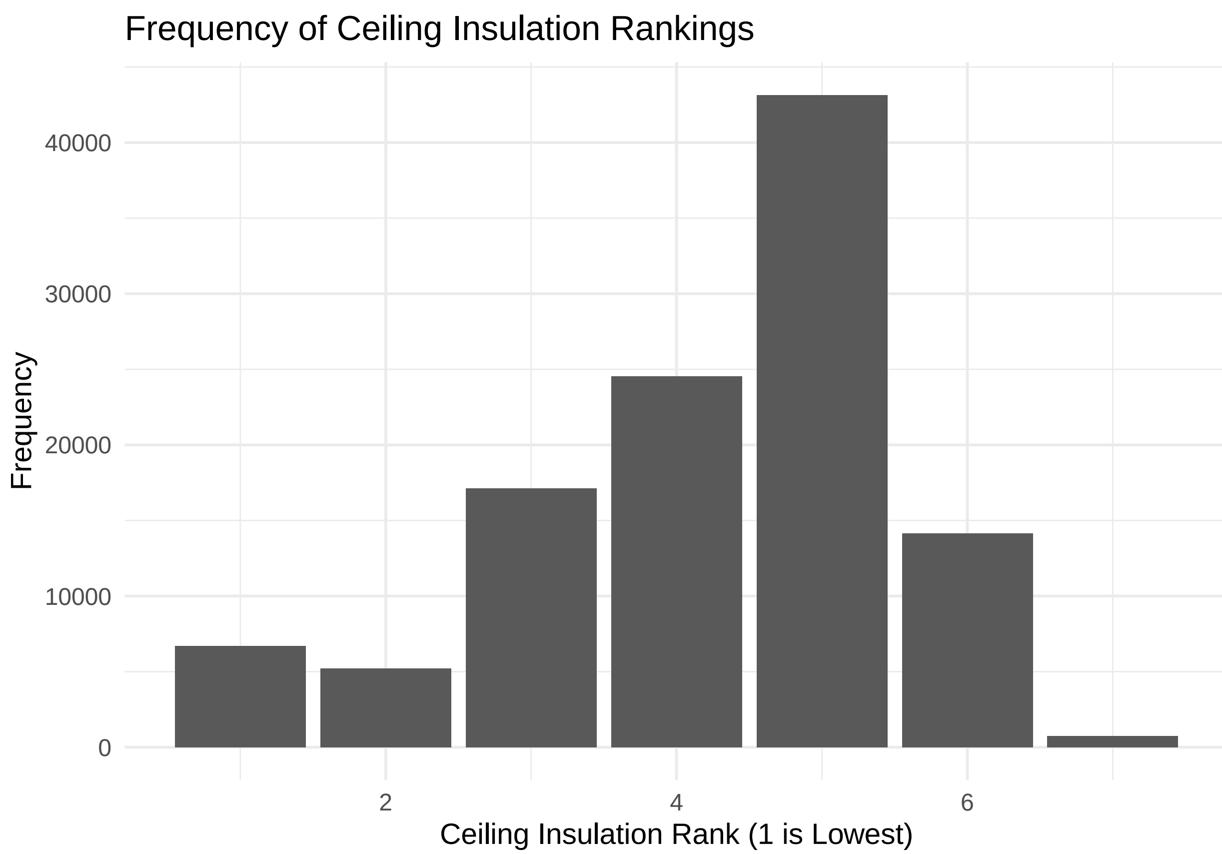


BOXPLOT OF RANKING OF INSULATION

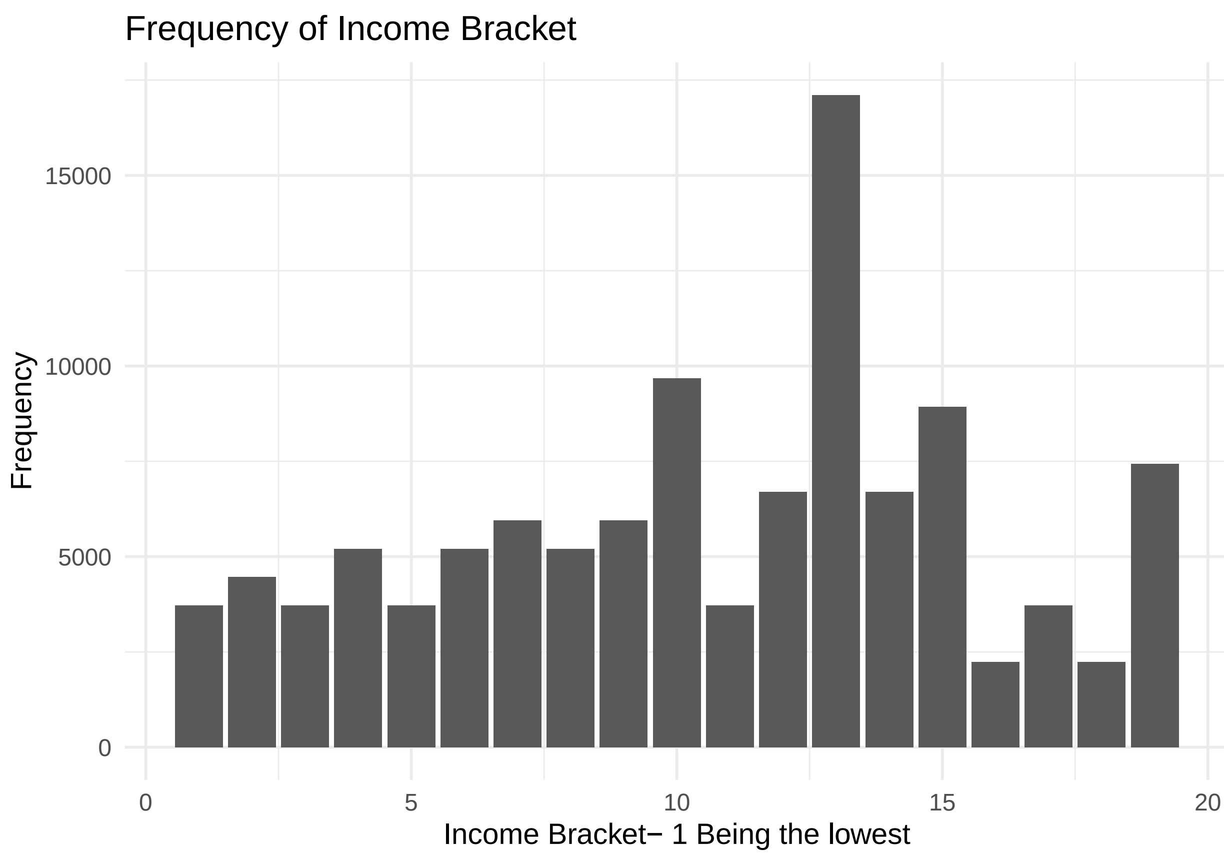
```r
ggplot(final_data.3, aes(x = wallInsulRank)) +
  geom_bar() +
  labs(title = "Frequency of Wall Insulation Rankings",
       x = "Wall Insulation Rank (1 is Lowest)",
       y = "Frequency") +
  theme_minimal()
```




```r
ggplot(final_data.3, aes(x = ceilingInsulRank)) +
  geom_bar() +
  labs(title = "Frequency of Ceiling Insulation Rankings",
       x = "Ceiling Insulation Rank (1 is Lowest)",
       y = "Frequency") +
  theme_minimal()
```




```
ggplot(final_data.3, aes(x = rankedIncome)) +
  geom_bar() +
  labs(title = "Frequency of Income Bracket",
       x = "Income Bracket- 1 Being the lowest",
       y = "Frequency") +
  theme_minimal()
```




```
# remove descriptive columns that we wont use in modeling

modelingData <- final_data.3 %>%
  select(-bldg_id, -day, -hour, -in.county, -in.weather_file_latitude,
         -in.weather_file_longitude)


# well use this when we increase the temp
modelingData_dupe <- final_data.3 %>%
  select(-bldg_id, -day, -hour, -in.county, -in.weather_file_latitude,
         -in.weather_file_longitude)

modelingData$total_electric_energy_used <- NULL


summary(modelingData)
```

```
##   temperature       wind_speed     horizontal_rad     normal_rad
##  Min.   :13.89   Min.   : 0.000   Min.   :   0.0   Min.   :  0.0
##  1st Qu.:23.90   1st Qu.: 1.190   1st Qu.:   0.0   1st Qu.:  0.0
##  Median :26.10   Median : 2.100   Median :  74.5   Median : 32.0
##  Mean   :26.57   Mean   : 2.333   Mean   : 248.5   Mean   :206.4
##  3rd Qu.:29.40   3rd Qu.: 3.480   3rd Qu.: 493.0   3rd Qu.:378.0
##  Max.   :38.30   Max.   :11.300   Max.   :1040.0   Max.   :957.0
##   diffuse_rad     window_ranking_integer windowAreaRanking  rankedIncome
##  Min.   :  0.0   Min.   : 1.00          Min.   :1.000     Min.   : 1.00
```

```
##  1st Qu.:  0.0   1st Qu.: 2.00         1st Qu.:2.000     1st Qu.: 7.00
##  Median : 51.0   Median : 6.00         Median :3.000     Median :11.00
##  Mean   :104.0   Mean   : 5.58         Mean   :3.053     Mean   :10.57
##  3rd Qu.:170.5   3rd Qu.: 8.00         3rd Qu.:4.000     3rd Qu.:14.00
##  Max.   :488.5   Max.   :10.00         Max.   :6.000     Max.   :19.00
##   in.bedrooms   rankedDucts     rankedHVAC_Efficiency   usageLevel
##  Min.   :1.0   Min.   : 1.000   Min.   :1.00          Min.   :1.000
##  1st Qu.:3.0   1st Qu.: 3.000   1st Qu.:5.00          1st Qu.:1.000
##  Median :3.0   Median : 5.000   Median :6.00          Median :2.000
##  Mean   :3.3   Mean   : 5.893   Mean   :5.36          Mean   :2.007
##  3rd Qu.:4.0   3rd Qu.: 8.000   3rd Qu.:6.00          3rd Qu.:3.000
##  Max.   :5.0   Max.   :13.000   Max.   :8.00          Max.   :3.000
##  wallInsulRank    ceilingInsulRank floorInsulRank  slabInsulRank
##  Min.   : 1.000   Min.   :1.000    Min.   :1.000   Min.   :1.000
##  1st Qu.: 1.000   1st Qu.:3.000    1st Qu.:1.000   1st Qu.:1.000
##  Median : 5.000   Median :5.000    Median :1.000   Median :2.000
##  Mean   : 4.473   Mean   :4.233    Mean   :1.733   Mean   :1.733
##  3rd Qu.: 7.000   3rd Qu.:5.000    3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :12.000   Max.   :7.000    Max.   :4.000   Max.   :6.000
##   wallTypeRank  in.geometry_stories    in.sqft      energy_by_sqft
##  Min.   :1.00   Min.   :1.000       Min.   : 633   Min.   :-0.0036656
##  1st Qu.:4.00   1st Qu.:1.000       1st Qu.:1220   1st Qu.: 0.0004090
##  Median :4.00   Median :1.000       Median :1690   Median : 0.0006116
##  Mean   :3.72   Mean   :1.507       Mean   :2114   Mean   : 0.0007170
##  3rd Qu.:4.00   3rd Qu.:2.000       3rd Qu.:2176   3rd Qu.: 0.0009011
##  Max.   :4.00   Max.   :3.000       Max.   :8194   Max.   : 0.0212338
```

setting up data for modeling using 70% of data for run time

```r
# set a seed for consistent reproduction
set.seed(117)

trainingIndex <- createDataPartition(modelingData$energy_by_sqft, p = .7,
                                     list = F)
trainingData <- modelingData[trainingIndex, ]
testingData <- modelingData[-trainingIndex, ]
```

LINEAR REGRESSION

```r
# run the linear regression
linReg <- lm(energy_by_sqft ~ ., data=modelingData)
# results
summary(linReg)
```

```
##
## Call:
## lm(formula = energy_by_sqft ~ ., data = modelingData)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0043714 -0.0002149 -0.0000535  0.0001346  0.0198341
##
## Coefficients:
```

```
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -3.907e-04  2.167e-05 -18.033  < 2e-16 ***
## temperature              5.268e-05  6.228e-07  84.585  < 2e-16 ***
## wind_speed               1.486e-05  9.150e-07  16.236  < 2e-16 ***
## horizontal_rad          -7.421e-07  1.993e-08 -37.239  < 2e-16 ***
## normal_rad               5.064e-07  1.569e-08  32.275  < 2e-16 ***
## diffuse_rad              7.722e-07  2.695e-08  28.653  < 2e-16 ***
## window_ranking_integer -4.295e-06  5.451e-07  -7.880 3.31e-15 ***
## windowAreaRanking       -5.936e-06  1.133e-06  -5.240 1.61e-07 ***
## rankedIncome            -1.769e-06  3.419e-07  -5.175 2.29e-07 ***
## in.bedrooms             -7.455e-05  1.924e-06 -38.741  < 2e-16 ***
## rankedDucts             -5.877e-06  5.168e-07 -11.372  < 2e-16 ***
## rankedHVAC_Efficiency  -7.465e-08  9.036e-07  -0.083  0.93417
## usageLevel               1.412e-04  1.994e-06  70.800  < 2e-16 ***
## wallInsulRank            1.815e-06  6.454e-07   2.813  0.00492 **
## ceilingInsulRank        -4.906e-06  1.294e-06  -3.792  0.00015 ***
## floorInsulRank           3.190e-05  2.099e-06  15.197  < 2e-16 ***
## slabInsulRank            1.139e-05  1.991e-06   5.724 1.04e-08 ***
## wallTypeRank            -1.282e-05  2.747e-06  -4.665 3.09e-06 ***
## in.geometry_stories     -5.240e-05  2.826e-06 -18.543  < 2e-16 ***
## in.sqft                 -9.659e-08  1.322e-09 -73.067  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0004682 on 111580 degrees of freedom
## Multiple R-squared:  0.2666, Adjusted R-squared:  0.2665
## F-statistic:  2135 on 19 and 111580 DF,  p-value: < 2.2e-16
```

```r
# predict
predictions_lm <- predict(linReg, newdata=testingData)


# Calculate the residuals
residuals_lm <- predictions_lm - testingData$energy_by_sqft

# Calculate Mean Squared Error (MSE)
mse_lm <- mean(residuals_lm^2)

# Calculate Root Mean Squared Error (RMSE)
rmse_lm <- sqrt(mse_lm)

cat("Root Mean Squared Error (RMSE):", rmse_lm, "\n")
```

```
## Root Mean Squared Error (RMSE): 0.0004542246
```

26% of the change in energy per sqft is explained by our dependent variables HVAC effiency is not scientifically signifcant and wall insulation rank is almost insignificant being close to 0.05

TREEBAG MODEL

```r
fit1 <- train(energy_by_sqft ~ ., data = trainingData, method = "treebag",
              preProc = c("center", "scale"))
```

```r
# show the results of the treebag
varImp(fit1)
```

```
## treebag variable importance
##
##                         Overall
## in.sqft                 100.000
## horizontal_rad           93.793
## diffuse_rad              89.409
## temperature              73.524
## rankedIncome             70.605
## normal_rad               65.190
## usageLevel               40.748
## window_ranking_integer   38.599
## floorInsulRank           37.772
## slabInsulRank            37.558
## rankedDucts              30.440
## in.bedrooms              27.903
## windowAreaRanking        24.306
## ceilingInsulRank         10.017
## wallTypeRank              7.329
## wind_speed                6.114
## wallInsulRank             5.808
## in.geometry_stories       2.110
## rankedHVAC_Efficiency     0.000
```

```r
# precict and compare
predictions_treebg <- predict(fit1, newdata = testingData)

residuals_treebg <- predictions_treebg - testingData$energy_by_sqft

rmse_treebg <- sqrt(mean(residuals_treebg^2))

print(paste("RMSE:", rmse_treebg))
```

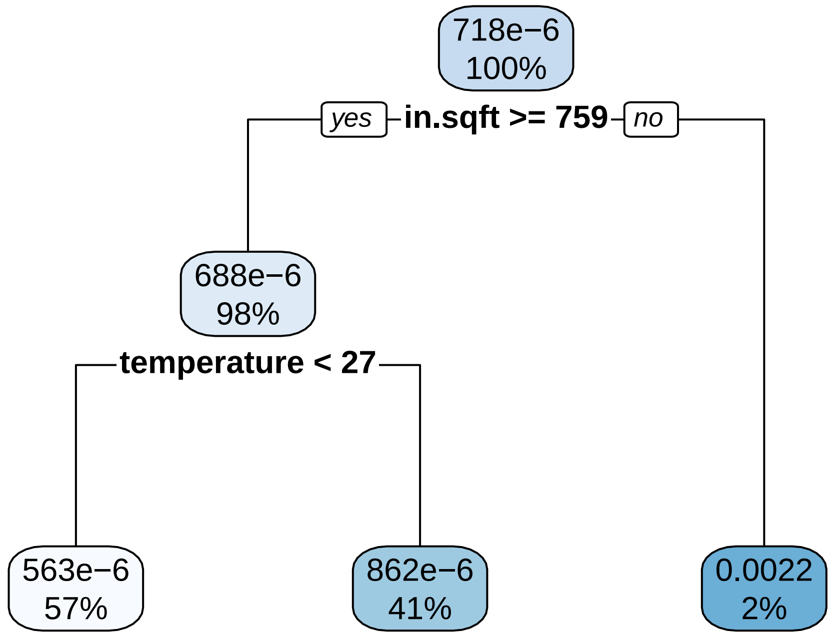
```
## [1] "RMSE: 0.000414377097613422"
```

DECISION TREE MODEL AND RPART TREE

```r
dt_model <- train(energy_by_sqft ~ ., data=trainingData, method="rpart")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

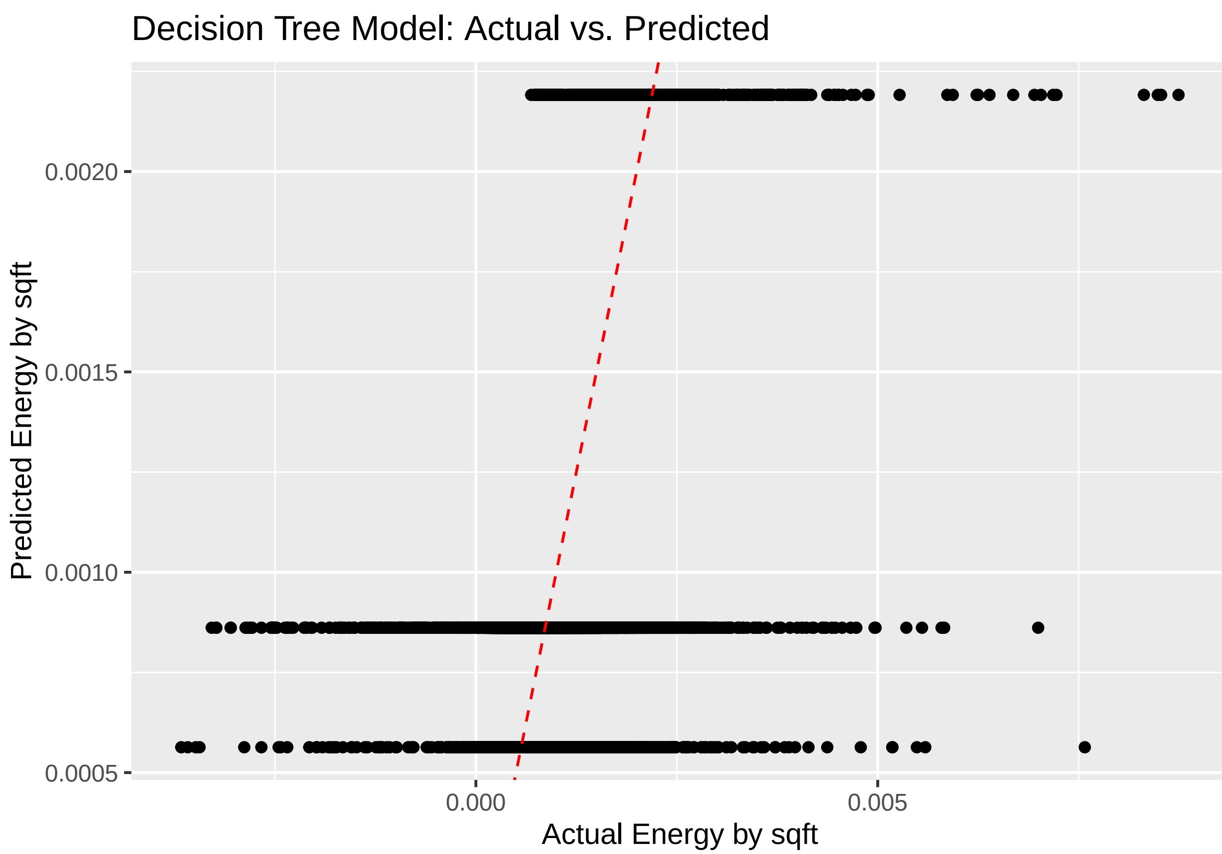
```r
rpart.plot(dt_model$finalModel)
```



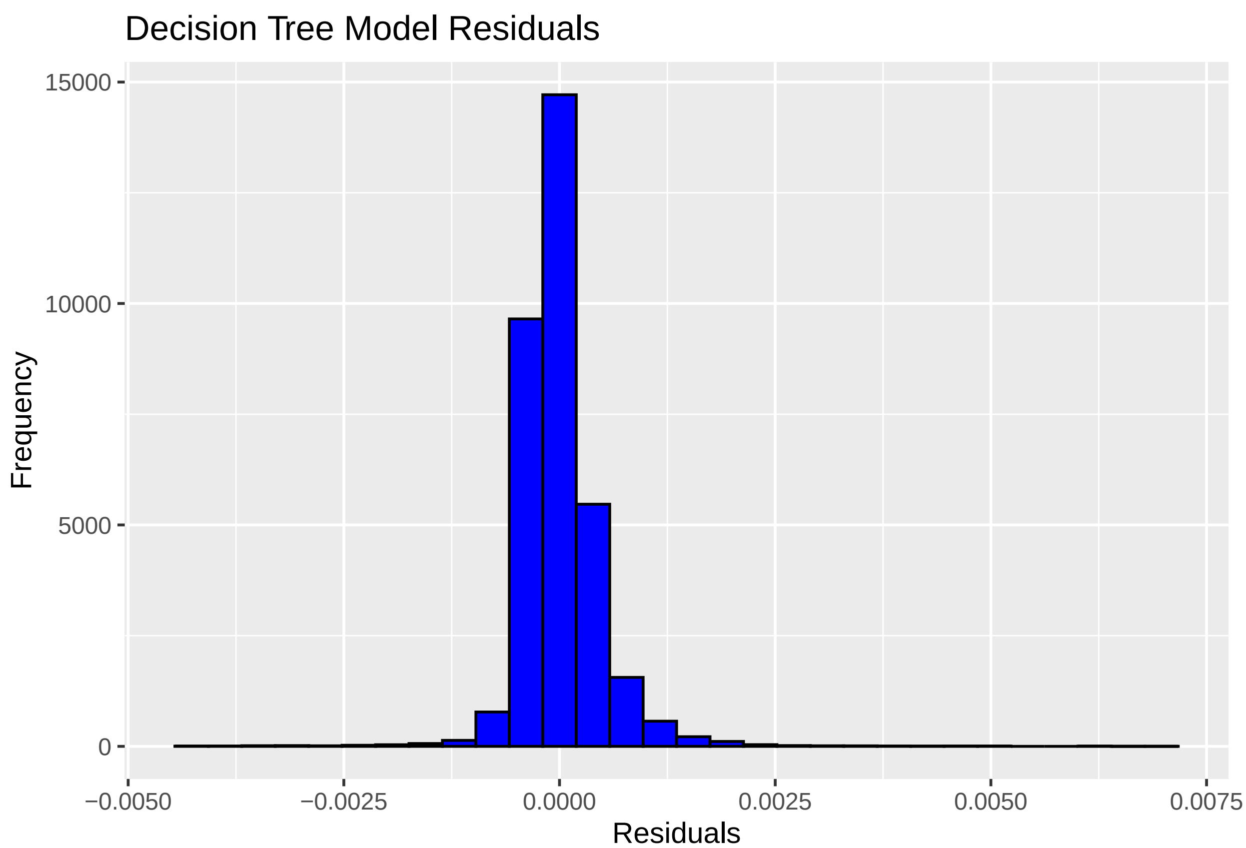

```r
# make predictions on the testing data
test_pred_dt <- predict(dt_model, newdata=testingData)


residuals_dt <- test_pred_dt - testingData$energy_by_sqft
mse_dt <- mean(residuals_dt^2)
rmse_dt <- sqrt(mse_dt)

cat("Root Mean Squared Error (RMSE):", rmse_dt, "\n")
```

```
## Root Mean Squared Error (RMSE): 0.0004693346
```

```r
results_dt <- data.frame(Actual = testingData$energy_by_sqft, Predicted = test_pred_dt)

# Scatter plot of Actual vs. Predicted values
ggplot(results_dt, aes(x = Actual, y = Predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(title = "Decision Tree Model: Actual vs. Predicted",
       x = "Actual Energy by sqft",
       y = "Predicted Energy by sqft")
```




```r
# Plot of the residuals
residuals_dt <- results_dt$Actual - results_dt$Predicted
ggplot() +
  geom_histogram(aes(x = residuals_dt), bins = 30, fill = "blue", color = "black") +
  labs(title = "Decision Tree Model Residuals",
       x = "Residuals",
       y = "Frequency")
```



SUPPORT VECTOR REGRESSION

```
svr_model <- svm(energy_by_sqft ~ ., data=trainingData, type='eps-regression')
print(svr_model)
```

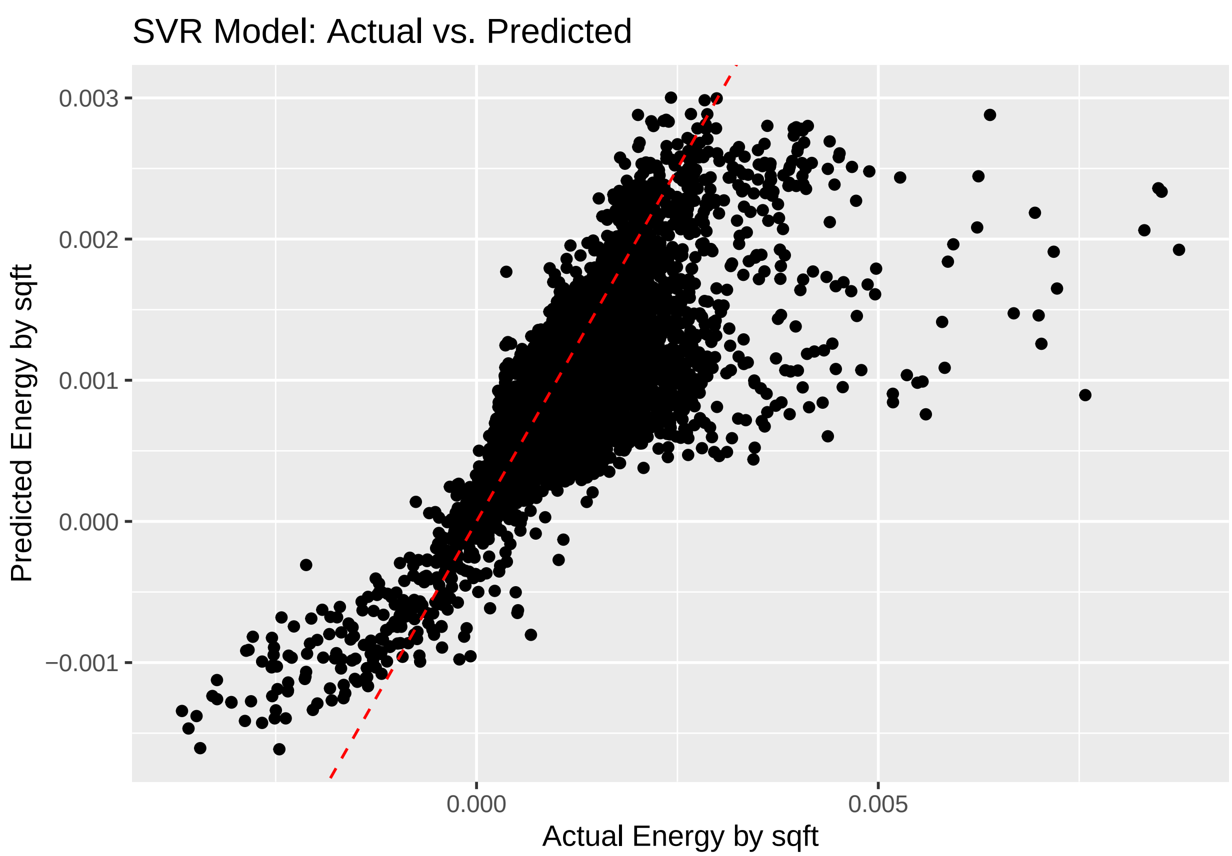
```
##
## Call:
## svm(formula = energy_by_sqft ~ ., data = trainingData, type = "eps-regression")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.05263158
##     epsilon:  0.1
##
##
## Number of Support Vectors:  48379
```

```
## Make predictions on the test set
test_predictions_svr <- predict(svr_model, newdata=testingData)

## Calculate RMSE
residuals_svr <- test_predictions_svr - testingData$energy_by_sqft
mse_svr <- mean(residuals_svr^2)
```

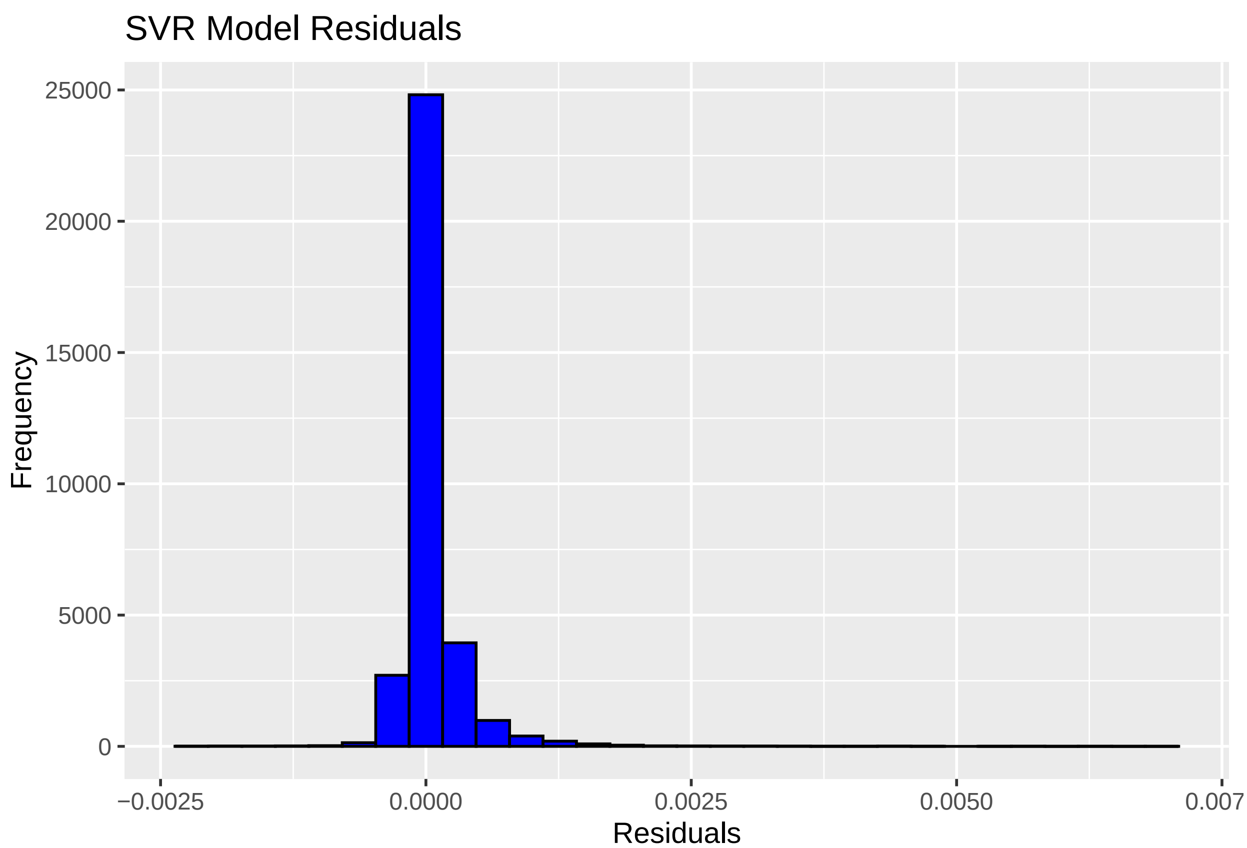
```r
rmse_svr <- sqrt(mse_svr)

cat("Root Mean Squared Error (RMSE):", rmse_svr, "\n")
```

```
## Root Mean Squared Error (RMSE): 0.0003133251
```

```r
results <- data.frame(Actual = testingData$energy_by_sqft, Predicted = test_predictions_svr)

ggplot(results, aes(x = Actual, y = Predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(title = "SVR Model: Actual vs. Predicted",
       x = "Actual Energy by sqft",
       y = "Predicted Energy by sqft")
```




```r
# Plot of residuals
residuals_plot_svr <- results$Actual - results$Predicted
ggplot() +
  geom_histogram(aes(x = residuals_plot_svr), bins = 30, fill = "blue", color = "black") +
  labs(title = "SVR Model Residuals",
       x = "Residuals",
       y = "Frequency")
```



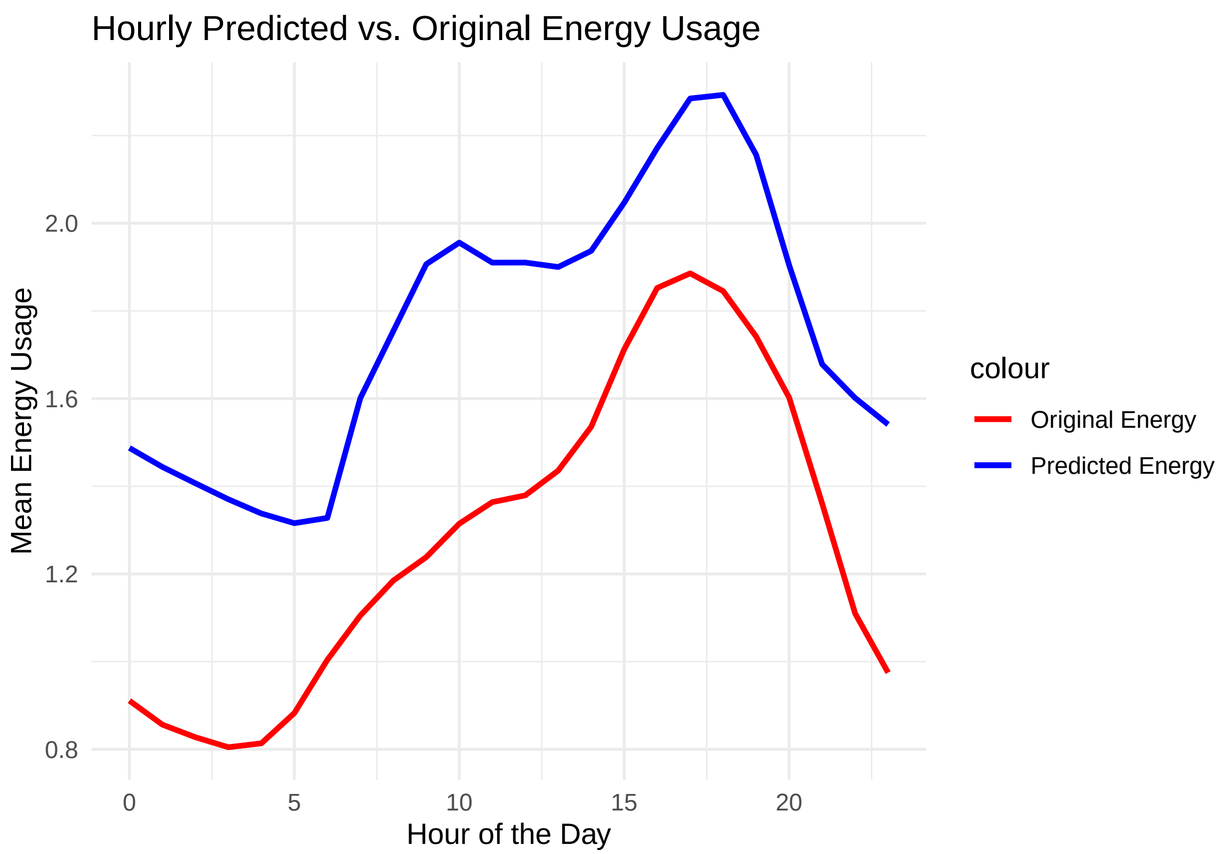
MODELS ON 5 DEGREE INCREASE

```
# increase overall temp by 5 degrees
modelingData_dupe$temperature <- modelingData_dupe$temperature + 5
modelingData_dupe$total_electric_energy_used <- NULL

# base predictions on 5 degree increase

# gather our predictions using the same models but with new data
# this is done in order to present a model driven result that is consistent
# with our original predictions


# I am going to focus on svr in this case, where my RMSE value was lowest
new_svr_pred <- predict(svr_model, newdata = modelingData_dupe)

# we want to not calculate energy per square feet, but overall energy consumption
# since that is essentially the question we are answering we just used per
# square feet as our target independent variable initially to run and develop
# our models

# we can tac each list of predictions onto our duplicated dataframe
modelingData_dupe <- modelingData_dupe %>%
  mutate(new_svr_pred)

# add our prediction and turn it back into total energy not by sqft
```

```r
# also bring back in our hour column
modelingData_dupe <- modelingData_dupe %>%
  mutate(totalEnergyPred = new_svr_pred * in.sqft,
         originalEnergyUse = final_data.3$total_electric_energy_used,
         hour = final_data.3$hour)


# group by hour and average predicted energy usage
modelingDupeSummary<- modelingData_dupe %>%
  group_by(hour) %>%
  summarize(meanPredHourlyEnergy = mean(totalEnergyPred),
            meanOriginalHourlyEnergy = mean(originalEnergyUse),
            temperature = mean(temperature))
```

```r
ggplot(modelingDupeSummary, aes(x = hour)) +
  geom_line(aes(y = meanPredHourlyEnergy, color = "Predicted Energy"), size = 1) +
  geom_line(aes(y = meanOriginalHourlyEnergy, color = "Original Energy"), size = 1) +
  labs(title = "Hourly Predicted vs. Original Energy Usage",
       x = "Hour of the Day",
       y = "Mean Energy Usage") +
  scale_color_manual(values = c("Predicted Energy" = "blue", "Original Energy" = "red")) +
  theme_minimal()
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```




```r
# this viz does not work due to the y scale being mixed between avg energy
# consumption and temperature so it is scrapped.

# ggplot(modelingDupeSummary, aes(x = hour)) +
#   geom_line(aes(y = meanPredHourlyEnergy, color = "Predicted Energy"), size = 1) +
#   geom_line(aes(y = meanOriginalHourlyEnergy, color = "Original Energy"), size = 1) +
#   geom_line(aes(y = temperature, color = "Temperature"), size = 1, linetype = "dashed") +
#   labs(title = "Hourly Predicted vs. Original Energy Usage with Temperature",
#        x = "Hour of the Day",
#        y = "Mean Values") +
#   scale_color_manual(values = c("Predicted Energy" = "cyan", "Original Energy" = "blue", "Temperature
#   theme_minimal()
```