-POWER PULSE HARMANOY-

FINAL PROJECT REPORT

IST 687- INTRODUCTION TO DATA SCIENCE

M003 GROUP 04

FALL 2023

GROUP MEMBERS:

NIDHI DEVENDRA CHAUHAN
MIHIR NILESH HOLMUKHE
KRUTI KOTADIA
CHINMAY UDAY RANADE
ZHIPENG ZHAO

CONTENTS:

1. INTRODUCTION

In response to mounting concerns about global warming and its potential influence on energy demand, our client, an energy company known as eSC, has recruited our help to address a significant issue. eSC is a South Carolina-based electric utility that also serves a tiny portion of North Carolina. Because of the consequences of global warming, the corporation expects a spike in electricity demand throughout the forthcoming summer. They are especially concerned about the strain that this increasing demand would put on their electrical grid, which could result in unsettling blackouts. Rather than relying on traditional solutions such as building additional power plants, eSC is taking a proactive and ecologically sensitive approach. The company strives to understand the critical elements influencing energy consumption and to investigate ways to persuade customers to reduce their energy consumption. The main goal is to systematically cut energy use in order to lessen the likelihood of blackouts during very hot summers. July, which is known to have the highest energy use on average, becomes the focus of their efforts. In addition to preserving the dependability of their electrical infrastructure, eSC hopes to contribute to a sustainable and environmentally friendly energy future by exploring the nuances of why energy is consumed and encouraging a culture of energy saving.

2. Project Goal

This project's main objective is to provide a thorough plan of action that an energy company (eSC) can use to efficiently control and lessen the spike in summertime electricity use, especially in July. The main goals are to identify the factors that influence energy consumption and to put policies in place that motivate users to use less energy. By accomplishing this, eSC hopes to reduce the demand for new energy production facilities, reduce the likelihood of future blackouts, and enhance the dependability of their entire grid.

Objectives:

1. Determine the Main Factors Affecting Energy Use:
- Perform a thorough examination of past data to identify the key variables affecting summertime energy use, paying particular attention to July.
- Determine trends and patterns in consumer behavior, the state of the weather, and other pertinent factors that affect the energy demand.
2. Analysis of Customer Behavior:
- Recognize how customers behave when it comes to energy use, especially during times of high demand.

- Examine consumer preferences, demographics, and past usage trends to customize energy-saving programs for certain client groups.

3. Create Models That Predict:

- Create prediction models to estimate energy consumption based on past performance, seasonal changes, and any outside variables (heatwaves, for example).
- Make use of these models to foresee times of increased demand and take preventative action.

4. Energy-Sparing Initiatives:

- Create and carry out energy-saving initiatives to inform and encourage users to use less energy during peak hours.
- To increase the efficacy of conservation efforts, look into forming alliances with nearby companies, governments, and communities.

5. Smart Technology Integration

- Examine how smart technology, such home automation systems and smart meters, can be integrated to give customers feedback on their energy usage in real time.
- Examine whether time-of-use pricing may be used to incentivize the use of energy during off-peak hours.

6. Outreach and Communication:

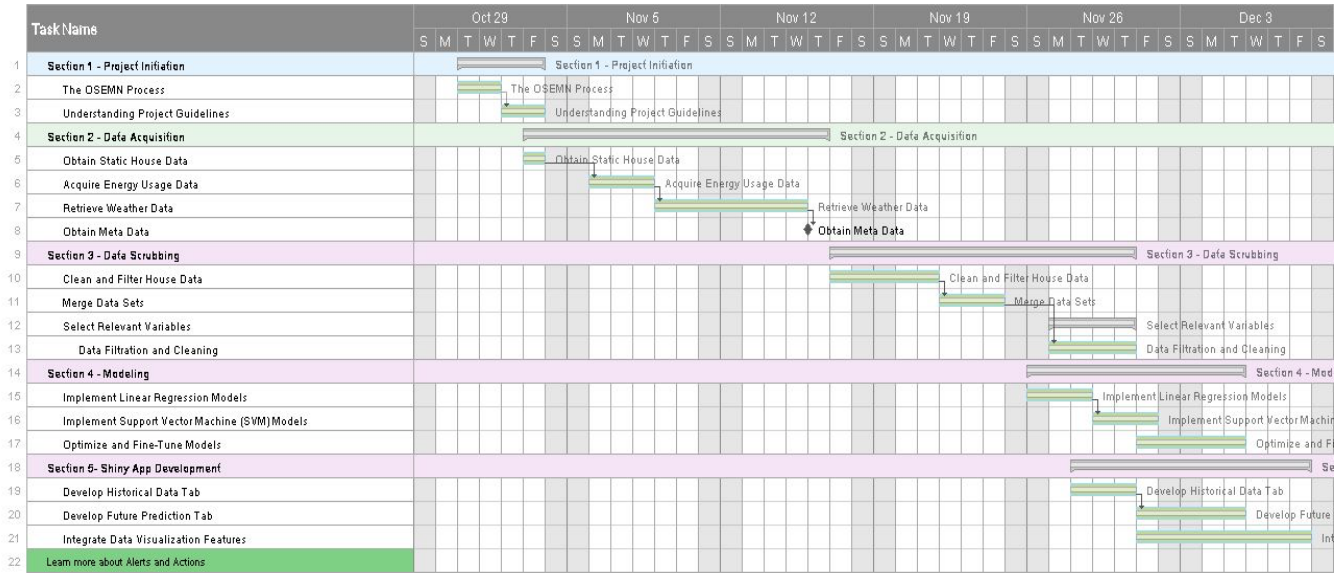
- Create a thorough communication strategy to educate consumers on the value of energy conservation, the possible effects of rising demand, and the advantages of consuming less energy.
- Employ many channels, such as social media, newsletters, and community gatherings, to include clients and encourage energy-conserving behaviors.

7. Observation and Assessment:

- Put in place a reliable monitoring mechanism to determine how well client engagement and energy conservation programs are working.
- Consistently assess the effectiveness of actions taken and modify plans in response to criticism and performance indicators.

By achieving these objectives, eSC aims to proactively address the challenges posed by increased energy demand during the summer, promote sustainable energy practices, and ultimately avoid the need for costly expansions in energy production capacity.

3. Workflow Breakdown



4. Code Explanation

Package Used in Coding
Linear Modeling Side

“arrow”
This package contains functions for reading various files including .parquet and .csv.
“tidyverse”
This package contains useful functions for data preparation and data filtering.
“writexl”
This package is installed so the data generated and stored in global environment can be exported as excel file.
“usmap”
Simplifies the creation of US maps in R, particularly useful for visualizing data at the state level.
“ggplot2”
This package is installed for plotting visualizations including line plot, histogram, boxplot and mapping.
“maps”
Provides a collection of geographical maps for the United States and world regions.
“ggmap”
This package is installed for mapping visualization.
“mapproj”
This package is included for map projection purpose.

“kernlab”
This package is installed for modeling purpose.
“caret”
This package is installed for modeling purpose.

Shiny App Side

“shiny”
Provides a web framework for building interactive web applications using R.
“shinydashboard”
Extends the 'shiny' package to create dashboards with a tabbed layout and various components.
“RCurl”
A package for making HTTP requests, including handling URLs, cookies, and authentication.
“jsonlite”
This package is installed for file reading.
“tidyverse”
This package contains useful functions for data preparation and data filtering.
“ggmap”
This package is installed to enable the creation of maps with ggplot().
“dplyr”
Offers a set of functions for data manipulation and transformation, making it easier to work with data frames in R.
“DT”
Provides an R interface to the DataTables JavaScript library, allowing for the creation of interactive tables in Shiny applications.
“plotly”
Enables the creation of interactive plots using the 'plotly.js' library, suitable for Shiny applications.
“ggplot2”
This package is installed for plotting visualizations including line plot, histogram, boxplot and mapping.
“sf”
Stands for Simple Features, this package supports the creation, representation, and manipulation of spatial (geographical) data.
“sp”
Offers classes and methods for spatial data, providing a framework for handling spatial data structures.
“usmap”
Simplifies the creation of US maps in R, particularly useful for visualizing data at the state level.
“maps”
Provides a collection of geographical maps for the United States and world regions.
“readxl”
Allows reading data from Excel files (both '.xls' and '.xlsx') into R, facilitating data import from Excel spreadsheets.

5. File Reading Iteration

The static house data file contains 5710 rows, and each row corresponds to a specific house. These houses are associated with their respective hourly energy usage data. Additionally, each house is identified by a county ID, which is located in column 27 of the static house data file. The weather data is recorded on an hourly basis for each county. Both the energy usage data and weather data cover the time period from January 1, 2018, 0:00:00, to December 31, 2018, 23:00:00.
To create a suitable dataset for future modeling, we have to develop a script that can automatically bring together the static house data, energy usage data, and weather data. This script will help consolidate all the necessary information into a unified dataset, making it easier for us to analyze and model the data effectively.

Iteration Script (comment removed)

```
for (i in 1:nrow(temp_house)){

  identifier <- paste0("This is ",as.character(i)," iteration")
  print(identifier)

  building_id <- temp_house$bldg_id[i]
  county_id <- temp_house$in.county[i]
  energyAddress <- paste0('https://intro-datascience.s3.us-east-2.amazonaws.com/SC-data/2023-houseData/',building_id,'.parquet')
  weatherAddress <- paste0('https://intro-datascience.s3.us-east-2.amazonaws.com/SC-data/weather/2023-weather-data/',county_id,'.csv')

  addressLog <- add_row(addressLog,bldg_id = building_id, county_id = county_id,energyAddress=energyAddress,weatherAddress=weatherAddress)

  tempdf_energy <- read_parquet(energyAddress)
  tempdf_weather <- read_csv(weatherAddress,show_col_types = FALSE)

  tempdf <- merge(tempdf_energy,tempdf_weather,all.x=TRUE,by.x="time",by.y="date_time")
  time_str <- tempdf[1] %>%
    mutate(time_as_string = format(time, "%Y-%m-%d %H:%M:%S"),.keep="unused")
  tempdf <- data.frame(tempdf,time_str)
  tempdf$month <- as.numeric(substr(tempdf$time_as_string,6,7))
  tempdf <- tempdf[complete.cases(tempdf$month),]

  tempdf$time_se <- as.numeric(paste0(substr(tempdf$time_as_string,6,7),substr(tempdf$time_as_string,9,10)))
  tempdf <- arrange(tempdf,tempdf$time_se)

  tempdf_clean <- tempdf[complete.cases(tempdf$Dry.Bulb.Temperature...C.),]

  for (i in 1:nrow(tempdf_clean)){
    tempdf_clean$sum_energy[i] <- sum(tempdf_clean[i,2:43])
  }

  tempdf_clean <- tempdf_clean %>%
    group_by(month) %>%
    summarise(monthly_usage = sum(sum_energy),monthly_temp = mean(Dry.Bulb.Temperature...C.),monthly_humidity = mean(Relative.Humidity....),
              monthly_windspeed = mean(Wind.Speed..m.s.))

  outputList <- add_row(outputList,bldg_id = building_id,peak_consumption = max(tempdf_clean$monthly_usage), peak_month = which.max(tempdf_clean$monthly_usage),
                        mean_temperature = tempdf_clean$monthly_temp[peak_month], mean_humidity = tempdf_clean$monthly_humidity[peak_month],
                        mean_wind_speed = tempdf_clean$monthly_windspeed[peak_month])

  rm(tempdf_energy)
  rm(tempdf_weather)
  rm(time_str)

}
```

We use a process called iteration by using a for loop. It goes through each row in the static house data file. From this data, we extract two important variables: "bldg_id" and "in.county". These variables are used to create the addresses for the energy usage data file and the county weather data file for the specific house we're focusing on. These files are temporarily stored as "tempdf_energy" and "tempdf_weather" while waiting for the next step of filtering.
Both the energy data file and weather data file have a common variable – "time" for energy data and "date_time" for weather data. We first merge these two datasets based on time to make sure they share the same timeline. Then, we use a method from the tidyverse to calculate the monthly average data for energy consumption and weather information. This is important because each row in the static house data file represents a specific house. It's ideal to have a single row of energy and

weather information for each house, which we then store in a pre-set data frame called "outputList" during each iteration. At the end of each iteration, we remove the temporary files.
The resulting data frame contains 5710 rows. Each row contains the building id, energy usage data, and weather data for a specific house.

6. Data Set Merging & Modeling Analysis

The output from iteration will then be merged with the cleaned static house data with respect to the building id for the modeling and prediction process. After data merging we got the resulting data frame called “rawdata” with 5710 rows and 56 columns.

Linear Modeling
Modeling Attempt # 1. Suppot Vector Machine (SVM)

In our initial modeling attempt, we used the SVM (Support Vector Machine) method. To prepare for this, we first applied a linear regression model using the "lm()" function to explore the relationship between energy consumption and all variables in our raw dataset ("rawdata"). Afterward, we used the "summary()" function to examine the details of the linear regression model. Based on the significance level of each variable, we performed a second round of data cleaning. We kept only the variables deemed significant, creating a refined dataset called "svmData." This dataset, containing 15 independent variables and 1 dependent variable, is now ready to be used for training our SVM model.

SVM Data Preparation Script (comment removed)

```
df <- rawdata
lmOut <- lm(formula=energy_consumption~.,data=df)
summary(lmOut)

coefs <- summary(lmOut)$coefficients
vars <- rownames(coefs)[which(coefs[, 4] < 0.01)]
relevant <- c(
  "in.sqft",
  "in.clothes_dryer",
  "in.cooling_setpoint",
  "in.county",
  "in.geometry_foundation_type",
  "in.geometry_wall_type",
  "in.hvac_has_ducts",
  "in.infiltration",
  "in.insulation_ceiling",
  "in.orientation",
  "in.occupants",
  "in.insulation_wall",
  "mean_temperature",
  "mean_humidity",
  "mean_wind_speed",
  "energy_consumption"
)

svmData <- df[,relevant]
```

The initial phase of our SVM modeling looks promising. We trained the model using 80% of our observations in “svmData”, and when we applied the trained model to the remaining 20% observations, the predictions showed an accuracy rate between 75% and 85%. This accuracy is within the error range we set beforehand, which is 30%. In simpler terms, if the actual energy

consumption is 1000 kWh, and our prediction falls between 700 kWh and 1300 kWh, we consider it a successful prediction.

SVM Modeling and Accurate Output Script (comment removed)

```
trainList <- createDataPartition(y=svmData$energy_consumption,p=.80,list=FALSE)
trainSet <- svmData[trainList,]
testSet <- svmData[-trainList,]
svm_model <- ksvm(energy_consumption ~ ., data = trainSet,C = 2,cross = 5,prob.model = TRUE)
svm_model
svmPred <- predict(svm_model, newdata = testSet, type = "response")
ifSuccess <- data.frame(svmPred,testSet$energy_consumption)
ifSuccess <- ifSuccess %>%
  mutate(success=between(svmPred,0.7*testSet.energy_consumption,1.3*testSet.energy_consumption))
tbl <- table(ifSuccess$success)
cbind(tbl,prop.table(tbl))
```

Accurate Output of SVM Model

```
       tbl
FALSE 1251 0.2190893
TRUE  4459 0.7809107
```

But when we attempted to make a future input dataset with different values for appliances, house geometry, and weather info, the SVM model didn't perform well. Because of this, we decided to stop using this model and instead move forward with linear regression.

Modeling Attempt # 2. Linear Regression

In our linear regression modeling, we followed a similar process. After training the model with 80% of our data, we tested it on the remaining 20%, and the predictions were accurate, ranging between 75% and 85%. This accuracy is a bit better than what we achieved with the SVM model.
The linear regression model works well with changing variables, and its outputs align with our expectations. For example, as the temperature rises in July, the energy consumption should also increase, and our model captures this relationship. This result is promising, and we've decided to use this model for future predictions in our Shiny App feature.

Linear Regression Modeling and Accurate Output Script (comment removed)

```r
colName <- c(
  "in.sqft",
  "in.clothes_dryer",
  "in.cooling_setpoint",
  "in.geometry_foundation_type",
  "in.geometry_wall_type",
  "in.hvac_has_ducts",
  "in.infiltration",
  "in.occupants",
  "in.insulation_wall",
  "mean_temperature",
  "mean_humidity",
  "mean_wind_speed",
  "energy_consumption"
)
july <- rawdata[,colName]
trainList <- createDataPartition(y=july$energy_consumption,p=0.8,list=FALSE)
trainSet <- july[trainList,]
testSet <- july[-trainList,]
lmJuly <- lm(formula=energy_consumption~.,data=trainSet)
summary(lmJuly)
prediction <- predict(lmJuly,testSet)
ifSuccess <- data.frame(actual=testSet$energy_consumption,pred=prediction)
ifSuccess <- ifSuccess %>%
  mutate(success=between(pred,0.7*actual,1.3*actual))
tbl <- table(ifSuccess$success)
cbind(tbl,prop.table(tbl))
```

Accurate Output of Linear Regression Model

```
      tbl
FALSE 199 0.1745614
TRUE  941 0.8254386
```

7. Shiny App

Our Shiny App has three features, each can be accessed by click on respective tab button on dashboard.

URL Link to shiny app: https://pisin.shinyapps.io/energyPredApp/

My Dashboard Tabs Script (comment removed)

```r
sidebar <- dashboardSidebar(
  sidebarMenu(
    menuItem("Static House Data Review", tabName = "past"),
    menuItem("Energy Usage Hotspot", tabName = "monitor"),
    menuItem("Future Prediction", tabName = "predict")
  )
)
```

Static House Data Review

This tab serves as a showcase for house data. In our project, we currently have access only to historical data. If we could connect to real-time online data, this tab could display the current status of appliances, house geometry, and energy consumption.
When you're on this tab, two visualizations pop up in the main area based on the "rawdata" we've created after sorting and combining the data. The one on the left is a histogram, giving an overview

of the general conditions among all houses in the data table. The one on the right is a boxplot, showing the connection between energy consumption and resident choices for a specific variable. For interactivity, we've included three radio buttons in this tab. They allow users to pick their areas of interest, making the visualizations more specific. Different radio button selections will pick different columns in the data table for visualization. At the bottom of the panel, there's a data table generated to provide users with a detailed view regarding the visualization.

UI Side Script of Static House Data Review Tab (comment removed)

```
tabItems(
  tabItem(tabName = "past",
          h2("Static House Data Review"),
          fluidRow(
            column(3, radioButtons("radio", h3("Select Your Concern"),
                                   choices = list("Appliance" = 1, "Cooling Setpoint" = 2,"House Geometry" = 3),selected = 1)),
            br(),br(),br(),br(),br(),br(),br(),br(),br(),br(),
            splitLayout(cellWidths = c("50%", "50%"), plotOutput("plot1"), plotOutput("plot2")),
            br(),
            h2("House Attribute Data Table"),
            fluidRow(box(DTOutput('table', width = '100%')))
          )
  )
```

Server-Side Script of Static House Data Review Tab (comment removed)

```
output$table <- renderDT(
  tab1Data() |>
    datatable()
)
output$plot1 <- renderPlot({
  if (input$radio == 1) {
    return(ggplot(tab1Data(), aes(x=in.clothes_dryer, y=pct,label = scales::percent(pct))) + geom_col() + scale_x_discrete(guide = guide_axis(n.dodge=3))
           + labs(title="Clothes Dryer Appliance")+ geom_text(position = position_dodge(width = .9),vjust = -0.5,size = 3)
           + scale_y_continuous(labels = scales::percent)+xlab("Clothes Dryer Types") + ylab("Percentage"))
  }
  else if (input$radio == 2) {
    return(ggplot(tab1Data(), aes(x=in.cooling_setpoint, y=pct,label = scales::percent(pct))) + geom_col() + scale_x_discrete(guide = guide_axis(n.dodge=3))
           + labs(title="Cooling Setpoints")+ geom_text(position = position_dodge(width = .9),vjust = -0.5,size = 3)
           + scale_y_continuous(labels = scales::percent)+xlab("Cooling Setpoint") + ylab("Percentage"))
  }
  else if (input$radio == 3) {
    return(ggplot(tab1Data(), aes(x=in.geometry_wall_type, y=pct,label = scales::percent(pct))) + geom_col() + scale_x_discrete(guide = guide_axis(n.dodge=3))
           + labs(title="House Geometry")+ geom_text(position = position_dodge(width = .9),vjust = -0.5,size = 3)
           + scale_y_continuous(labels = scales::percent)+xlab("Wall Types") + ylab("Percentage"))
  }
})
output$plot2 <- renderPlot({
  if (input$radio == 1) {
    return(ggplot(data=newData) + aes(x=energy_consumption, y=in.clothes_dryer) + geom_point() + geom_boxplot()+xlab("Energy Usage") + ylab("Clothes Dryer Types"))
  }
  else if (input$radio == 2) {
    return(ggplot(data=newData) + aes(x=energy_consumption, y=in.cooling_setpoint) + geom_point() + geom_boxplot()+xlab("Energy Usage") + ylab("Cooling Setpoint"))
  }
  else if (input$radio == 3) {
    return(ggplot(data=newData) + aes(x=energy_consumption, y=in.geometry_wall_type) + geom_point() + geom_boxplot()+xlab("Energy Usage") + ylab("Wall Types"))
  }
})
```

Energy Usage Hot Spot

This tab offers a mapping visualization that with ability to display real-time hotspots of energy usage. We create the base map using ggmap(), and then add a layer of scatter plot on top. The scatter plot uses different sizes and filled gradient colors to represent various aspects of the data.

UI Side Script of Energy Hot Spot Tab (comment removed)

```
tabItem(tabName = "monitor",
        h2("Energy Consumption Hotspot"),
        plotOutput("plot3",click="clickMap")
)
```

Server-Side Script of Energy Hot Spot Tab (comment removed)

```
output$plot3 <- renderPlot({
  register_stadiamaps("3361de07-653a-4211-ac7c-67613b31c4a7")
  RawData <- rawdata
  selectCol <- c("in.county","in.weather_file_latitude","in.weather_file_longitude","energy_consumption")
  countyPos <- RawData[,selectCol]
  countyPos <- countyPos %>%
    group_by(in.county) %>%
    summarise(aveEnergy=mean(energy_consumption),aveLon=mean(in.weather_file_longitude),aveLat=mean(in.weather_file_latitude))
  countyPos <- arrange(countyPos,countyPos$aveEnergy)
  us <- c(left = -83.2, bottom = 32.2, right = -78.2, top = 35.7)
  map <- get_stadiamap(us, zoom = 7, maptype = "stamen_toner_lite") %>% ggmap()
  tab2plot <- map+geom_point(countyPos,mapping=aes(x=aveLon,y=aveLat,color=aveEnergy,size=aveEnergy))+scale_size(range = c(1,30))
      +scale_colour_gradient(low="green", high="red")
  tab2plot
},height = 600, width = 1000)
```

Future Prediction Tab

This tab features an interactive visualization that responds to user input. It has three main parts:

1. Parameter Modification:

Users can adjust temperature change and daytime using slider bars.
There are two preset data sets: "Original Preference" and "Energy Saver." The original one keeps the existing preferences, while the energy saver tends to set residences with lower energy consumption.

2. Polygon Mapping Visualization:

With the user's input parameters, the script generates a polygon map of South Carolina with county borders.
Counties are filled with gradient colors. Green represents lower energy usage, while red indicates higher energy usage.

3. Interactive Text Output:

Users can click on the map, and the code will identify the clicked location as the county name.
The text output at the bottom of the map displays the county name and the energy consumption for that county.

UI Side Script of Static House Data Review Tab (comment removed)

```
tabItem(tabName = "predict",
        h2("Future Prediction"),
        box(
          "Please use slider bar to change parameters", br(),"Energy Prediction Unit: kWh",
          sliderInput("temperature_slider", "Temperature Change", -10, 10, 0),
          selectInput("house_setup","Attributes:",choices=c("Original Preference","Energy Saver")),
          sliderInput("daytime_slider", "Time in a Day", 0, 23, 12),
          plotOutput("plot4",click="clickMap2"),
          fluidRow(box(textOutput('textTab3'))),
          fluidRow(box(textOutput('textTab4')))
        )
)
```

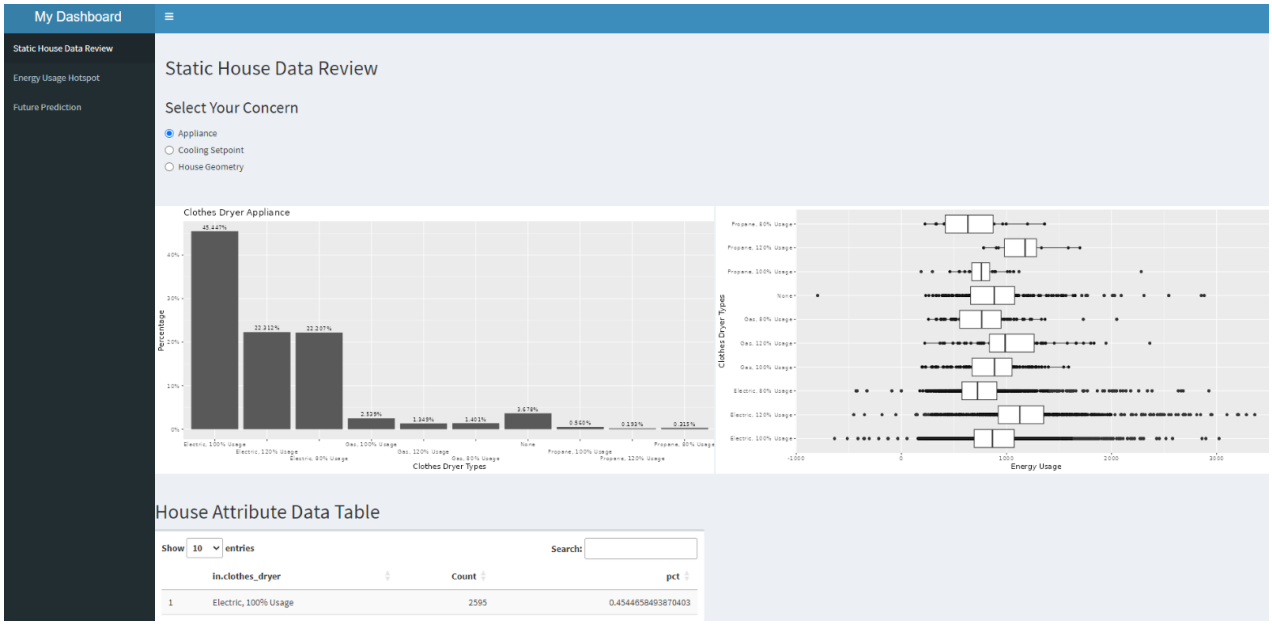
Server-Side Script of Static House Data Review Tab - Visualization (comment removed)

```
output$plot4 <- renderPlot({
  sc_counties <- map_data("county","south carolina")
  countyMap <- ggplot(sc_counties) + aes(long,lat, group=group) + geom_polygon(fill = "white", color = "black")
  prediction <- predict(lmJuly,tab3Data())
  result <- data.frame(county=tab3Data()$countyName,pred=prediction)
  plot4 <- merge(result,sc_counties,all.x=TRUE, by.x="county",by.y="subregion")
  b <- c(300, 500, 700, 1000, 1200, 1400, 2000)
  c <- c("skyblue", "green", "yellow", "orange", "purple", "red")
  tab3plot<-ggplot(plot4) + aes(long,lat, group=group) + geom_polygon(aes(fill= pred), color = "black",) + scale_fill_distiller(direction = 1)
    +scale_fill_gradientn(colors = c, values = scales::rescale(b),limits=c(300,2000))
  tab3plot
})
```

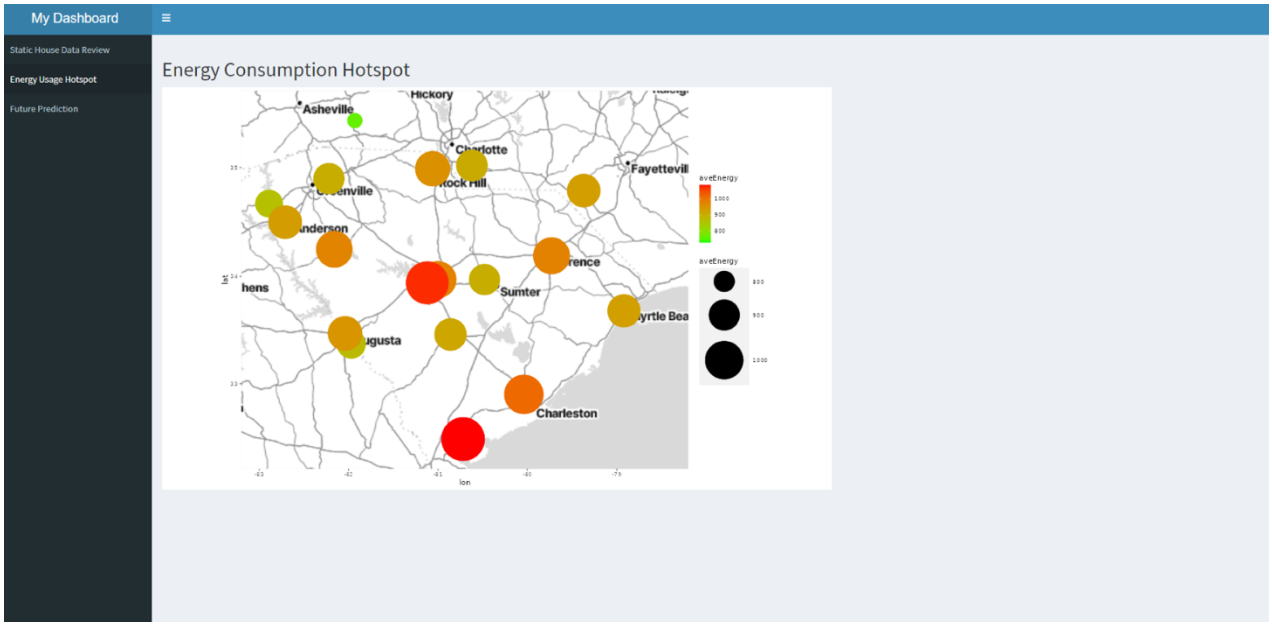
Server-Side Script of Static House Data Review Tab - Interactive Feature (comment removed)

```
countyNumber <- reactive({
  longlatGroup <- plotFinal[,2:4]
  point <- data.frame(x=input$clickMap2$x,y=input$clickMap2$y)
  for (i in 1:46){
    temp <- longlatGroup[longlatGroup$group==i,]
    check <- as.numeric(point.in.polygon(point.x=point$x,point.y=point$y,pol.x=temp$long,pol.y=temp$lat))
    countyNumber <- 1
    if(check == 1){
      countyNumber <- (i)
      break
    }
  }
  return(countyNumber)
})
output$textTab3 <- renderText({
  prediction <- predict(lmJuly,tab3Data())
  result <- data.frame(county=tab3Data()$countyName,pred=prediction)
  input$clickMap2
  req(input$clickMap2)
  str1 <- paste("In ",as.character(result[countyNumber(),1]," ","county"))
  isolate(HTML(paste(str1)))
})
output$textTab4 <- renderText({
  prediction <- predict(lmJuly,tab3Data())
  result <- data.frame(county=tab3Data()$countyName,pred=prediction)
  input$clickMap2
  req(input$clickMap2)
  str2 <- paste("The predicted energy usage is ",as.character(result[countyNumber(),2]))
  isolate(HTML(paste(str2)))
})
```

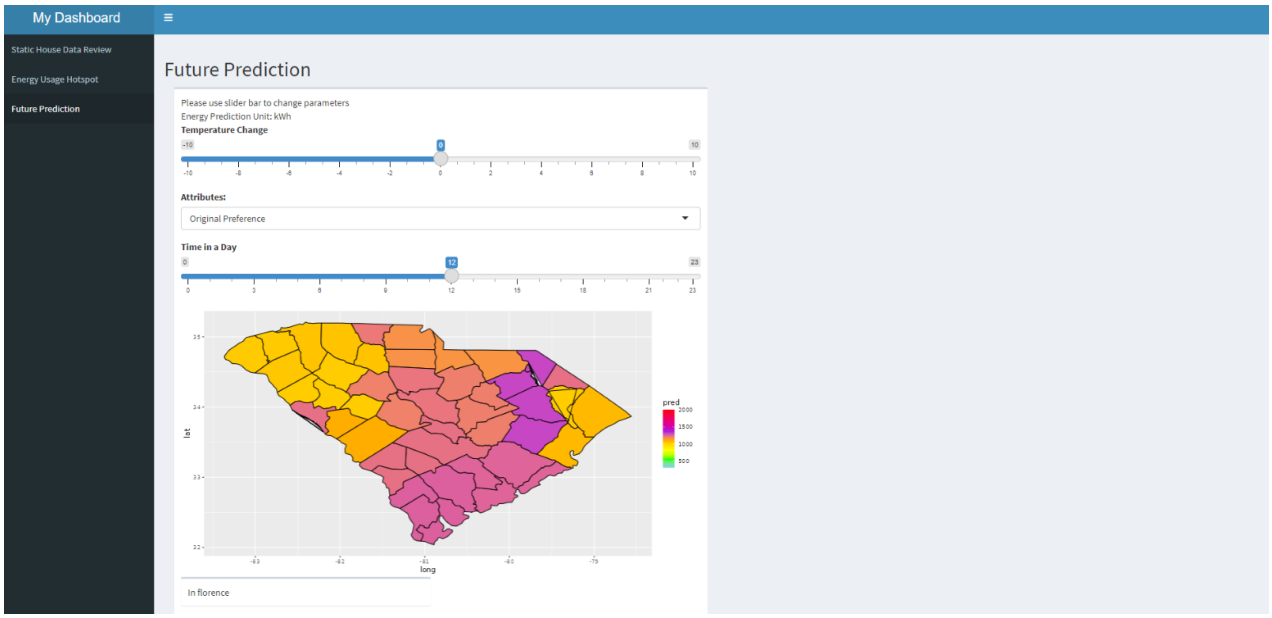
Shiny App Prototype Overview
Tab - Static House Data Review



Tab - Energy Hot Spot

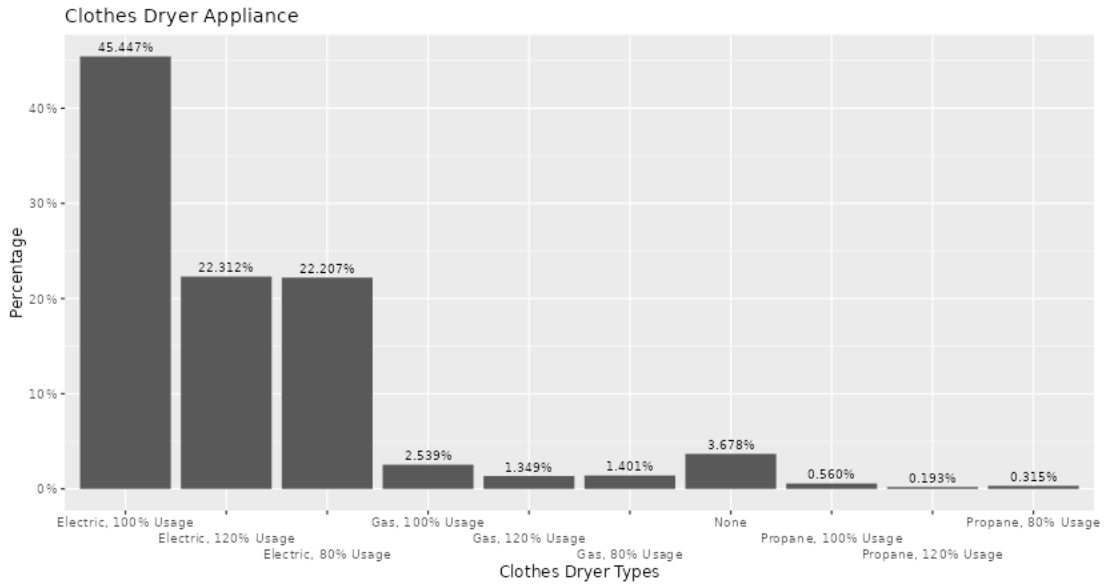


Tab - Future Prediction



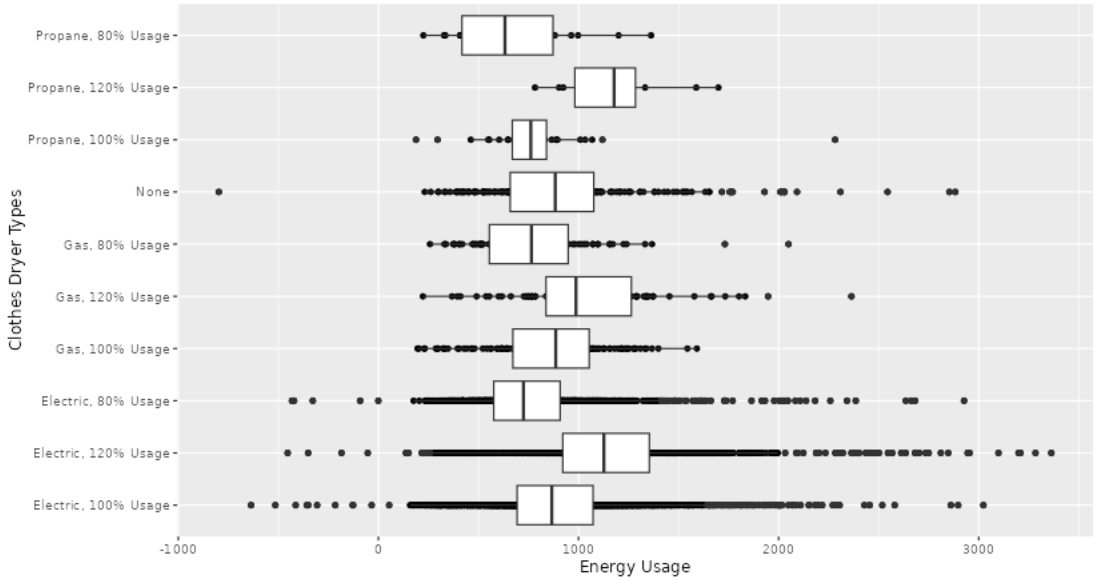
Visualization Overview

Histogram



The histogram will be generated based on user's selected variable. Each variable has several different types. The histogram will how how many count each type has and their respective percentages.

Boxplot



The boxplot further explains the relationship between the variable and energy consumption. If the box locates toward the right side of the plot, it means that type of variable will generate a higher energy consumption.

Data Table

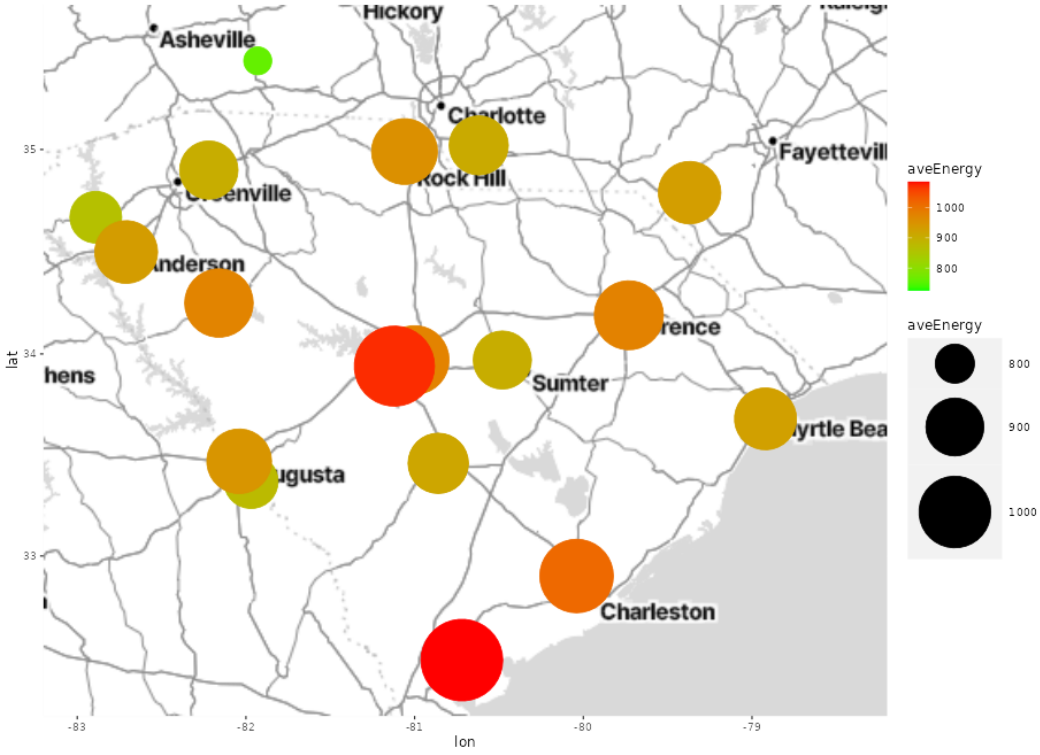
Show 10 entries Search:

	in.clothes_dryer	Count	pct
1	Electric, 100% Usage	2595	0.4544658493870403
2	Electric, 120% Usage	1274	0.2231173380035026
3	Electric, 80% Usage	1268	0.2220665499124343
4	Gas, 100% Usage	145	0.02539404553415061
5	Gas, 120% Usage	77	0.01348511383537653
6	Gas, 80% Usage	80	0.01401050788091068
7	None	210	0.03677758318739054
8	Propane, 100% Usage	32	0.005604203152364273
9	Propane, 120% Usage	11	0.001926444833625219
10	Propane, 80% Usage	18	0.003152364273204904

Showing 1 to 10 of 10 entries Previous 1 Next

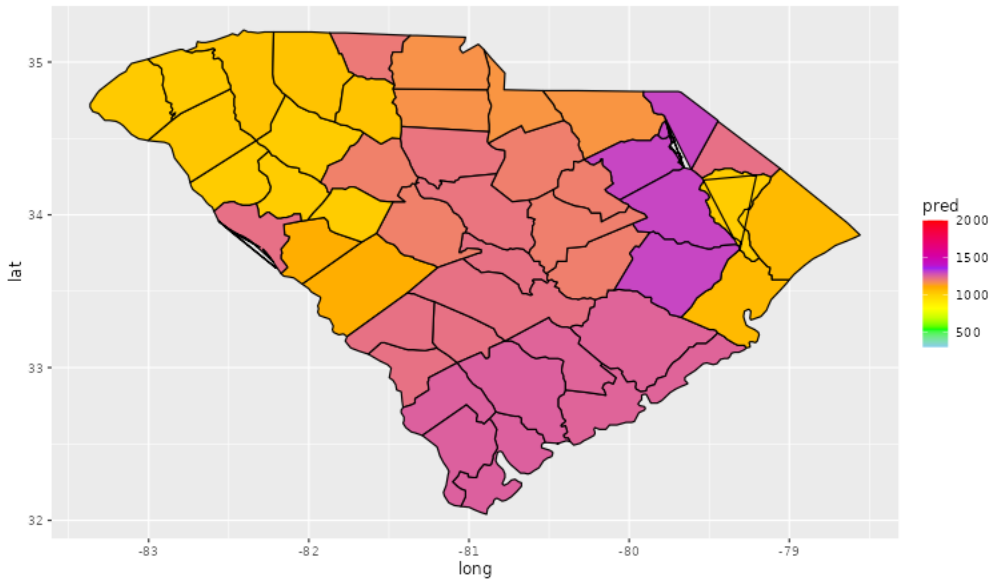
The data table is the what the plot is based on. It provides the data side of view to the user.

Hot Spot Map



A scatter plot is generated on the basis of stadia map. For larger energy usage, the size of the dot will also be larger. And the dots are filled with gradient colors. As energy consumption rises, the color will turn from green to red.

Interactive Map



The interactive map is plotted with ggplot() and US state geographic information. Counties will be filled with different colors concerning their average daily energy usage. The color will turn from green to red as energy usage rises. Users can click on any point of the map, and if the click is inside a specific county, the specific text out of that county will be generated to tell users more about their concerns.

8. Interpretations

After studying and calculating all the possible ways for getting apt energy consumption, we have narrowed down to some points which can be used for saving energy.

- Since South Carolina lies in southeastern coastal of the United States and by saying coastal, we mean humidity.
- Temperature wise January is the coldest month whereas July is the hottest where it goes up to 81.5°F.
- While optimizing the data, we found out that appliances play a very important role in energy consumption. for eg: - different types of dryer like electric, gas , propane.
- Around 2595 houses prefer electric clothes dryer over propane and gas.
- Apart from that the cooling setpoint of a house is vital when it comes to consuming energy. Till date there are 40% of houses having 60- 70 F set as their cooling setpoint which consumes more energy.
- Since energy consumption is the most in Charleston which 1320 KW/h we can come up with an idea or an awareness amongst the population for making some day-to-day life changes. Like switching to a propane clothes dryer as it consumes less energy compared to the electric one.
- Another adjustment can be setting the cool point in a range of 70-80F to provide large energy consumption.

9. Recommendations

Solar panels can improve electricity consumption in South Carolina in several ways. Here are some of the main ways solar panels can have a positive impact:

Renewable energy production:

- Solar panels use solar energy and convert it into electricity. By installing solar panels, South Carolina can generate clean, renewable energy, reducing dependence on fossil fuels and reducing the carbon footprint of electricity generation.

Lower energy costs:

- Once installed, solar panels can help lower utility bills for both residential and commercial users. The initial investment in solar panels can be compensated in the long run by saving on energy costs, since sunlight is a free and abundant resource. Online measurement programs:
- South Carolina may have net metering programs that allow solar panel owners to receive credit for excess electricity they generate and feed back into the grid. It helps promote the adoption of solar energy by providing financial incentives.