IDS Team 3 Code

```
# Install required packages if not already installed
# install.packages(c("shiny", "shinydashboard", "dplyr", "arrow", "forecast", "openxlsx", "ggplot2",
"xgboost", "corrplot"))

# Load required libraries
library(shiny)
library(shinydashboard)
library(dplyr)
library(arrow)
library(forecast)
library(ggplot2)
library(corrplot)
library(xgboost)

# UI
ui <- fluidPage(
  dashboardPage(
    dashboardHeader(title = "Energy Consumption Analysis"),
    dashboardSidebar(
      sidebarMenu(
        menuItem("Plots", tabName = "Plots"),
        menuItem("Results", tabName = "results"))),
    dashboardBody(
      tabItems(
        tabItem(tabName = "Plots",
                fluidRow(
                  box(title = "Monthly Energy Consumption",
                      plotOutput("monthly_energy_plot", height = 400)),
                  box(title = "Correlation Plot",
                      plotOutput("correlation_plot", height = 400)),
                  box(title = "Scatter Plot",
                      plotOutput("scatter_plot", height = 400)),
                  box(title = "Building Size Distribution",
                      plotOutput("building_size_distribution", height = 400)))),
        tabItem(tabName = "results",
                fluidRow(
                  box(title = "XGBoost Model RMSE",
                      verbatimTextOutput("xgb_rmse")),
                  box(title = "Time Series Model RMSE",
                      verbatimTextOutput("ts_rmse")))))))

# Server
server <- function(input, output) {
  # 1. Load Static House Data
```

IDS Team 3 Code

```
  url_static <-
"https://intro-datascience.s3.us-east-2.amazonaws.com/SC-data/static_house_info.parquet"
  staticHouseData <- arrow::read_parquet(url_static)

  # 2. Load Energy Usage Data
  url_base <-
"https://intro-datascience.s3.us-east-2.amazonaws.com/SC-data/2023-houseData/"
  unique_building_id <- unique(staticHouseData$bldg_id)[1:5]
  energyUsageData <- data.frame()

  for (building_id in unique_building_id) {
    url <- paste0(url_base, building_id, ".parquet")
    tryCatch({
      energy_data <- arrow::read_parquet(url)
      energyUsageData <- rbind(energyUsageData, energy_data)},
      error = function(e) {
        cat("Error for building_id:", building_id, "\n")})}

  energy_columns <- grep("\\.energy_consumption$", names(energyUsageData), value = TRUE)
  energyUsageData$total_energy_consumption <-
rowSums(energyUsageData[energy_columns], na.rm = TRUE)
  energyUsageData <- energyUsageData[, !(names(energyUsageData) %in% energy_columns)]
  energyUsageData$time <- as.POSIXct(energyUsageData$time, format="%Y-%m-%d
%H:%M:%S")
  colnames(energyUsageData)[colnames(energyUsageData) == "time"] <- "date_time"

  grouped <- energyUsageData %>%
    group_by(date_time) %>%
    summarise(hourly_energy_consumption = sum(total_energy_consumption))

  hourly_usage <- as.data.frame(grouped)

  hourly_usage$date_time <- format(hourly_usage$date_time, "%Y-%m-%d %H")
  hourly_usage <- na.omit(hourly_usage)

  # 3. Load Weather Data
  unique_county_id <- unique(staticHouseData$in.county)
  unique_county_id <- unique_county_id[1:5]
  weatherData <- data.frame()

  for(county_id in unique_county_id) {
    url <-
paste('https://intro-datascience.s3.us-east-2.amazonaws.com/SC-data/weather/2023-weather-d
ata/', county_id, '.csv', sep='')
```

IDS Team 3 Code

```
    weather <- read.csv(url)
    weatherData <- rbind(weatherData,weather)}

  weatherData$date_time <- as.POSIXct(weatherData$date_time, format="%Y-%m-%d
%H:%M:%S")

  grouped <- weatherData %>%
    group_by(date_time) %>%
    summarise_all(funs(mean(., na.rm = TRUE)))

  hourly_weather <- as.data.frame(grouped)
  hourly_weather$date_time <- format(hourly_weather$date_time, "%Y-%m-%d %H")

  # 4. Merge Energy Usage and Weather Data
  merged_data <- merge(hourly_usage, hourly_weather, by = "date_time", all = TRUE)
  merged_data <- na.omit(merged_data)

  columns_to_convert <- setdiff(names(merged_data), "date_time")
  merged_data[, columns_to_convert] <- lapply(merged_data[, columns_to_convert],
as.numeric)

  # 5. New Train-Test Split
  training_start_date <- as.Date("2018-01-01")
  training_end_date <- as.Date("2018-06-30")

  test_start_date <- as.Date("2018-07-01")
  test_end_date <- as.Date("2018-07-31")

  training_data <- subset(merged_data, date_time >= training_start_date & date_time <=
training_end_date)
  test_data <- subset(merged_data, date_time >= test_start_date & date_time <=
test_end_date)

  X_train <- as.matrix(training_data[, -which(names(training_data) ==
"hourly_energy_consumption")])
  y_train <- training_data$hourly_energy_consumption

  X_test <- as.matrix(test_data[, -which(names(test_data) == "hourly_energy_consumption")])
  y_test <- test_data$hourly_energy_consumption

  non_numeric_cols <- sapply(training_data, function(x) !is.numeric(x))
  X_train <- as.matrix(training_data[, !non_numeric_cols])
  X_test <- as.matrix(test_data[, !non_numeric_cols])
```

IDS Team 3 Code

```
  # Render XGBoost model RMSE
  output$xgb_rmse <- renderText({
    # 6. xgBoost Model
    xgb_model <- xgboost(data = X_train, label = y_train, objective = "reg:squarederror", nrounds
= 10)
    predictions <- predict(xgb_model, as.matrix(X_test))
    rmse <- sqrt(mean((predictions - y_test)^2))
    cat("RMSE on test data:", rmse, "\n")
    return(paste("RMSE on test data:", rmse))})

  # Render time series model RMSE
  output$ts_rmse <- renderText({
    # Create time series objects for training and test data
    ts_training_data <- ts(training_data$hourly_energy_consumption, frequency = 24)
    ts_test_data <- ts(test_data$hourly_energy_consumption, frequency = 24)

    # Fit an ARIMA model automatically
    arima_model <- auto.arima(ts_training_data)
    forecast_values <- forecast(arima_model, h = length(ts_test_data))
    ts_predictions <- forecast_values$mean
    rmse <- sqrt(mean((ts_predictions - y_test)^2))
    cat("RMSE on test data:", rmse, "\n")

    return(paste("RMSE on test data:", rmse))})

  # 6. Visualizations
  output$building_size_distribution <- renderPlot({
    ggplot(staticHouseData, aes(x = in.sqft)) +
      geom_histogram(binwidth = 30, fill = 'skyblue', color = 'black') +
      labs(title = 'Distribution of Building Sizes', x = 'Square Footage', y = 'Count')})

  output$cooling_setpoint_distribution <- renderPlot({
    ggplot(staticHouseData, aes(x = cooling_setpoint_offset_magnitude)) +
      geom_histogram(binwidth = 2, fill = 'lightgreen', color = 'black') +
      labs(title = 'Distribution of Cooling Setpoint Offset Magnitude', x = 'Cooling Setpoint Offset
Magnitude', y = 'Count')})

  merged_data$date_time <- as.POSIXct(merged_data$date_time)
  merged_data$month <- format(merged_data$date_time, "%Y-%m")

  ggplot(merged_data, aes(x = month, y = hourly_energy_consumption, fill = month)) +
    geom_bar(stat = "identity") + labs(title = "Hourly Energy Consumption by Month",
                                       x = "Month", y = "Hourly Energy Consumption") + theme(axis.text.x =
element_text(angle = 45, hjust = 1))
```

IDS Team 3 Code

```
  correlation_matrix <- cor(merged_data[, c("hourly_energy_consumption",
"Dry.Bulb.Temperature...C.", "Relative.Humidity....", "Wind.Speed..m.s.")])
  corrplot(correlation_matrix, method = "color")

  ggplot(merged_data, aes(x = Dry.Bulb.Temperature...C., y = hourly_energy_consumption)) +
    geom_point() + labs(title = "Scatter Plot of Energy Consumption vs Temperature",
                        x = "Dry Bulb Temperature", y = "Hourly Energy Consumption")

  output$monthly_energy_plot <- renderPlot({
    ggplot(merged_data, aes(x = month, y = hourly_energy_consumption, fill = month)) +
      geom_bar(stat = "identity") + labs(title = "Hourly Energy Consumption by Month",
                                         x = "Month", y = "Hourly Energy Consumption") + theme(axis.text.x =
element_text(angle = 45, hjust = 1))})

  output$correlation_plot <- renderPlot({
    correlation_matrix <- cor(merged_data[, c("hourly_energy_consumption",
"Dry.Bulb.Temperature...C.", "Relative.Humidity....", "Wind.Speed..m.s.")])
    corrplot(correlation_matrix, method = "color")})

  output$building_size_distribution <- renderPlot({
    ggplot(staticHouseData, aes(x = in.sqft)) +
      geom_histogram(binwidth = 30, fill = 'skyblue', color = 'black') +
      labs(title = 'Distribution of Building Sizes', x = 'Square Footage', y = 'Count')})

  output$scatter_plot <- renderPlot({
    ggplot(merged_data, aes(x = Dry.Bulb.Temperature...C., y = hourly_energy_consumption)) +
      geom_point() + labs(title = "Scatter Plot of Energy Consumption vs Temperature",
                          x = "Dry Bulb Temperature", y = "Hourly Energy Consumption")})
}

# Run the application
shinyApp(ui = ui, server = server)
```