# A Machine Learning Approach to Photovoltaic System Optimization

2023-08-03


## Abstract

As the world transitions from fossil fuels to renewable energy, it is critical to maximize the efficiency in which we harness this energy. To address this concern, we examined the effect of weather on solar irradiance, with our goal to ultimately predict the energy output of a hypothetical solar farm. In this study, we collected solar irradiance data, and utilized multiple machine learning models to predict photovoltaic power production based on weather. Different models were compared in order to find an ideal model that provides both high accuracy and low computational cost. We found that the XGBoost model provided the lowest RMSE . Using the XGBoost model, a heat map was created to find the most efficient location for solar panel placement in Santa Fe.


## Setup

```
knitr::opts_chunk$set(eval = FALSE, echo = TRUE)
if (!require("pacman")) {
    install.packages("pacman")
}
```

```
## Loading required package: pacman
```

```
pacman::p_load(dplyr, ggplot2, ggrepel, data.table, lubridate, ggpubr,
               patchwork, car, glmnet, e1071, caret, xgboost, keras,
               randomForest, sf, Ckmeans.1d.dp, DiagrammeR, DiagrammeRsvg, rsvg)
```

## Cleaning

```
# This function removes the first two rows of a given dataframe
remove_first_two <- function(x) {
    x[-(1:2)]
}

# will be used later on
get_first_two <- function(x) {
    x[(1:2)]
}

# read in the raw files
```

```r
lv_files <- list.files(path = "data/Las_Vegas_Solar_Irradiation/raw",
                       pattern = "\\.csv$", full.names = TRUE)
lv_files_no_path <- list.files(path = "data/Las_Vegas_Solar_Irradiation/raw",
                               pattern = "\\.csv$", full.names = FALSE)
lv_read <- lapply(lv_files, readLines)

# remove the info lines
lv_read <- lapply(lv_read, remove_first_two)

# rewrite to a new file
for (i in 1:length(lv_files)) {
    fwrite(lv_read[i],
           file = (paste0("data/Las_Vegas_Solar_Irradiation/cleaned/",
                          lv_files_no_path[i])), quote = FALSE)
}

# join all Las Vegas data into one dataframe
lv_files_cleaned <- list.files(path = "data/Las_Vegas_Solar_Irradiation/cleaned",
                               pattern = "\\.csv$", full.names = TRUE)
lv_cleaned <- lapply(lv_files_cleaned, fread)
lv_full <- lv_cleaned %>%
    reduce(full_join)
fwrite(lv_full, file = "data/Las_Vegas_Solar_Irradiation/LV.csv")

# same process for New York and Chicago
ny_files <- list.files(path = "data/NY_Solar_Irradiation/raw",
                       pattern = "\\.csv$", full.names = TRUE)
ny_files_no_path <- list.files(path = "data/NY_Solar_Irradiation/raw",
                               pattern = "\\.csv$", full.names = FALSE)
ny_read <- lapply(ny_files, readLines)

ny_read <- lapply(ny_read, remove_first_two)

for (i in 1:length(ny_files)) {
    fwrite(ny_read[i],
           file = (paste0("data/NY_Solar_Irradiation/cleaned/",
                          ny_files_no_path[i])), quote = FALSE)
}

ny_files_cleaned <- list.files(path = "data/NY_Solar_Irradiation/cleaned",
                               pattern = "\\.csv$", full.names = TRUE)
ny_cleaned <- lapply(ny_files_cleaned, fread)

ny_full <- ny_cleaned %>%
    reduce(full_join)

fwrite(ny_full, file = "data/NY_Solar_Irradiation/ny.csv")

chicago_files <- list.files(path = "data/Chicago_Solar_Irradiation/raw",
                            pattern = "\\.csv$", full.names = TRUE)
chicago_files_no_path <- list.files(path = "data/Chicago_Solar_Irradiation/raw",
                                    pattern = "\\.csv$", full.names = FALSE)
chicago_read <- lapply(chicago_files, readLines)
```

```r
chicago_read <- lapply(chicago_read, remove_first_two)

for (i in 1:length(chicago_files)) {
    fwrite(chicago_read[i],
           file = (paste0("data/Chicago_Solar_Irradiation/cleaned/",
                          chicago_files_no_path[i])), quote = FALSE)
}


chicago_files_cleaned <- list.files(path = "data/Chicago_Solar_Irradiation/cleaned",
                                    pattern = "\\.csv$", full.names = TRUE)
chicago_cleaned <- lapply(chicago_files_cleaned, fread)

chicago_full <- chicago_cleaned %>%
    reduce(full_join)

fwrite(chicago_full, file = "data/Chicago_Solar_Irradiation/chicago.csv")

chicago_full <- chicago_full$Location = "Chicago"
ny_full <- ny_full$Location = "New York"
lv_full <- lv_full$Location = "Las Vegas"


# put all data together into one large dataframe
data = list(lv_full, ny_full, chicago_full)
all_data <- data %>%
    reduce(full_join)
fwrite(all_data, file = "data/SolarFull.csv")
```

## EDA on Cleaned Data

```r
# read in the 3 datasets
LV <- fread("data/Las_Vegas_Solar_Irradiation/LV.csv")
NY <- fread("data/NY_Solar_Irradiation/ny.csv")
Chi <- fread("data/Chicago_Solar_Irradiation/chicago.csv")

# loop through them as all the code will be the same
city_data <- list(LV, NY, Chi)

# this will contain all of the plots
allPlots <- list(NULL, NULL, NULL)

for(i in (1:3)) {
  head(city_data[[i]])
  str(city_data[[i]])

  city_data[[i]]$Date <- make_datetime(year = city_data[[i]]$Year,
                                       month = city_data[[i]]$Month,
                                       day = city_data[[i]]$Day,
                                       hour = city_data[[i]]$Hour,
                                       min = city_data[[i]]$Minute)
```

```r
city_data[[i]] %>%
  ggplot(aes(x = Date, y = Temperature)) + geom_line(color = "blue")

plots <- NULL

# create a graph for each of the 20 years
temp <- city_data[[i]] %>%
  # average by year-month
  group_by(Year, Month) %>%
  summarise(Temperature=mean(Temperature, na.rm = TRUE),.groups = "drop") %>%
  ggplot() +
  geom_line(aes(x = Month, y = Temperature, color = factor(Year))) +
  scale_x_continuous(breaks = 1:12, labels = month.abb, minor_breaks = NULL) +
  labs(title="Average Temperature by Month",x="Month",y="Temp (°C)",colour="Year")

dew <- city_data[[i]] %>%
  # average by year-month
  group_by(Year, Month) %>%
  summarise(`Dew Point` = mean(`Dew Point`, na.rm = TRUE), .groups = "drop") %>%
  ggplot() +
  geom_line(aes(x = Month, y = `Dew Point`, color = factor(Year))) +
  scale_x_continuous(breaks = 1:12, labels = month.abb, minor_breaks = NULL) +
  labs(title = "Average Dew Point by Month",
       x="Month", y="Dew Point (°C)", colour = "Year")

humidity <- city_data[[i]] %>%
  # average by year-month
  group_by(Year, Month) %>%
  summarise(`Relative Humidity` = mean(`Relative Humidity`, na.rm = TRUE),
            .groups = "drop") %>%
  ggplot() +
  geom_line(aes(x = Month, y = `Relative Humidity`, color = factor(Year))) +
  scale_x_continuous(breaks = 1:12, labels = month.abb, minor_breaks = NULL) +
  labs(title = "Average Relative Humidity by Month",
       x = "Month", y = "Relative Humidity (%)", colour = "Year")

albedo <- city_data[[i]] %>%
  # average by year-month
  group_by(Year, Month) %>%
  summarise(`Surface Albedo` = mean(`Surface Albedo`, na.rm = TRUE),
            .groups = "drop") %>%
  ggplot() +
  geom_line(aes(x = Month, y = `Surface Albedo`, color = factor(Year))) +
  scale_x_continuous(breaks = 1:12, labels = month.abb, minor_breaks = NULL) +
  labs(title = "Average Surface Albedo by Month",
       x = "Month", y = "Surface Albedo (%)", colour = "Year")

wind <- city_data[[i]] %>%
  # average by year-month
  group_by(Year, Month) %>%
  summarise(`Wind Speed` = mean(`Wind Speed`, na.rm = TRUE), .groups = "drop") %>%
  ggplot() +
  geom_line(aes(x = Month, y = `Wind Speed`, color = factor(Year))) +
```

```
  scale_x_continuous(breaks = 1:12, labels = month.abb, minor_breaks = NULL) +
  labs(title = "Average Wind Speed by Month", y="Wind Speed (m/s)", colour = "Year")

pressure <- city_data[[i]] %>%
  # average by year-month
  group_by(Year, Month) %>%
  summarise(`Pressure` = mean(`Pressure`, na.rm = TRUE), .groups = "drop") %>%
  ggplot() +
  geom_line(aes(x = Month, y = `Pressure`, color = factor(Year))) +
  scale_x_continuous(breaks = 1:12, labels = month.abb, minor_breaks = NULL) +
  labs(title = "Average Pressure by Month", y="Pressure (mbar)", colour = "Year")

windD <- city_data[[i]] %>%
  # average by year-month
  group_by(Year, Month) %>%
  summarise(`Wind Direction` = mean(`Wind Direction`, na.rm = TRUE),
            .groups = "drop") %>%
  ggplot() +
  geom_line(aes(x = Month, y = `Wind Direction`, color = factor(Year))) +
  scale_x_continuous(breaks = 1:12, labels = month.abb, minor_breaks = NULL) +
  labs(title = "Average Wind Direction by Month",
       y="Wind Direction (°)", colour = "Year")

precip <- city_data[[i]] %>%
  # average by year-month
  group_by(Year, Month) %>%
  summarise(`Precipitable Water` = mean(`Precipitable Water`, na.rm = TRUE),
            .groups = "drop") %>%
  ggplot() +
  geom_line(aes(x = Month, y = `Precipitable Water`, color = factor(Year))) +
  scale_x_continuous(breaks = 1:12, labels = month.abb, minor_breaks = NULL) +
  labs(title = "Average Precipitable Water by Month",
       y = "Precipitable Water (cm)", colour = "Year")

zenith <- city_data[[i]] %>% ggplot(aes(x=`Solar Zenith Angle`)) +
  geom_histogram(bins=360) +
  scale_x_continuous(breaks=seq(0,360,30)) +
  labs(title="Solar Zenith Angle Histogram", x = "Degrees", y = "Count")

ghi <- city_data[[i]] %>%
  # average by year-month
  group_by(Year, Month) %>%
  summarise(`GHI` = mean(`GHI`, na.rm = TRUE), .groups = "drop") %>%
  ggplot() +
  geom_line(aes(x = Month, y = `GHI`, color = factor(Year))) +
  scale_x_continuous(breaks = 1:12, labels = month.abb, minor_breaks = NULL) +
  labs(title = "Average GHI by Month", colour = "Year", y = "Avg GHI (w/m^2)")

cloud <- city_data[[i]] %>% ggplot(aes(x=`Cloud Type`)) +
  geom_histogram(bins=11) + scale_x_continuous(breaks=seq(0,10,1)) +
  labs(title="Cloud Type")

vsTemp <- city_data[[i]] %>%
```

```
  group_by(Year, Month) %>%
  summarise(`GHI` = mean(`GHI`, na.rm=TRUE),
            `Temperature` = mean(`Temperature`, na.rm=TRUE)) %>%
  ggplot(aes(x=`Temperature`, y=GHI)) + geom_point() +
  labs(title="Average GHI by Month vs. Temperature")

vsDew <- city_data[[i]] %>%
  group_by(Year, Month) %>%
  summarise(`GHI` = mean(`GHI`, na.rm=TRUE),
            `Dew Point` = mean(`Dew Point`, na.rm=TRUE)) %>%
  ggplot(aes(x=`Dew Point`, y=GHI)) + geom_point() +
  labs(title="Average GHI by Month vs. Dew Point")

vsHumid <- city_data[[i]] %>%
  group_by(Year, Month) %>%
  summarise(`GHI` = mean(`GHI`, na.rm=TRUE),
            `Relative Humidity` = mean(`Relative Humidity`, na.rm=TRUE)) %>%
  ggplot(aes(x=`Relative Humidity`, y=GHI)) + geom_point() +
  labs(title="Average GHI by Month vs. Humidity")

vsAlbedo <- city_data[[i]] %>%
  group_by(Year, Month) %>%
  summarise(`GHI` = mean(`GHI`, na.rm=TRUE),
            `Surface Albedo` = mean(`Surface Albedo`, na.rm=TRUE)) %>%
  ggplot(aes(x=`Surface Albedo`, y=GHI)) + geom_point() +
  labs(title="Average GHI by Month vs. Surface Albedo")

vsWindS <- city_data[[i]] %>%
  group_by(Year, Month) %>%
  summarise(`GHI` = mean(`GHI`, na.rm=TRUE),
            `Wind Speed` = mean(`Wind Speed`, na.rm=TRUE)) %>%
  ggplot(aes(x=`Wind Speed`, y=GHI)) + geom_point() +
  labs(title="Average GHI by Month vs. Wind Speed")

vsPressure <- city_data[[i]] %>%
  group_by(Year, Month) %>%
  summarise(`GHI` = mean(`GHI`, na.rm=TRUE),
            `Pressure` = mean(`Pressure`, na.rm=TRUE)) %>%
  ggplot(aes(x=`Pressure`, y=GHI)) + geom_point() +
  labs(title="Average GHI by Month vs. Pressure")

vsWindD <- city_data[[i]] %>%
  group_by(Year, Month) %>%
  summarise(`GHI` = mean(`GHI`, na.rm=TRUE),
            `Wind Direction` = mean(`Wind Direction`, na.rm=TRUE)) %>%
  ggplot(aes(x=`Wind Direction`, y=GHI)) + geom_point() +
  labs(title="Average GHI by Month vs. Wind Direction")

vsPrecip <- city_data[[i]] %>%
  group_by(Year, Month) %>%
  summarise(`GHI` = mean(`GHI`, na.rm=TRUE),
            `Precipitable Water` = mean(`Precipitable Water`, na.rm=TRUE)) %>%
  ggplot(aes(x=`Precipitable Water`, y=GHI)) + geom_point() +
```

```r
  labs(title="Average GHI by Month vs. Precipitable Water")

vsZenith <- city_data[[i]] %>%
  group_by(`Solar Zenith Angle`) %>%
  summarise(average_ghi = mean(GHI, na.rm=TRUE)) %>%
  ggplot(aes(x=`Solar Zenith Angle`, y=average_ghi)) +
  geom_point() + labs(title="Average GHI by Month vs. Precipitable Water")

vsCloud <- city_data[[i]] %>%
  group_by(`Cloud Type`) %>%
  summarise(average_ghi = mean(GHI, na.rm=TRUE)) %>%
  ggplot(aes(x=`Cloud Type`, y=average_ghi)) +
  geom_point() + labs(title="Average GHI by Month vs. Cloud Type")

city_data[[i]]$Zenith_Bins <-
  cut(city_data[[i]]$`Solar Zenith Angle`, breaks = seq(0, 180, by = 10))

# Plotting irradiance vs zenith angle
# The gray dots are outliers of solar irradiance, likely caused by clouds
p1 <- ggplot(data = city_data[[i]], aes(x = Zenith_Bins, y = GHI)) +
  geom_boxplot(outlier.color = "gray", size = 0.5) +
  labs(x = "Solar Zenith Angle (degrees)",
       y = "Global Horizontal Irradiance (W/m²)",
       title = "Distribution of GHI for Different Solar Zenith Angle
       (Grouped by bins of10 degrees)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# plotting solar zenith against time of day
p2 <- ggplot(data = city_data[[i]], aes(x = Hour,
                                        y = `Solar Zenith Angle`,
                                        group = interaction(Month, Day),
                                        color = factor(Month))) + geom_line() +
  labs(x = "Time of Day (Hour)", y = "Solar Zenith Angle (degrees)",
       title = "Solar Zenith Angle by Time of Day (Spaghetti Plot)") +
  scale_color_discrete(name = "Month",
                       labels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                                  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")) +
  theme_minimal() +
  theme(legend.position = "top", axis.text.x = element_text(angle = 45, hjust = 1))


# shows the difference between solstices, and their effect on solar zenith angle
filtered_data <- city_data[[i]] %>%
  filter(Month %in% c(6, 12))

p3 <- ggplot(data = filtered_data, aes(x = Hour, y = `Solar Zenith Angle`,
                                       group = interaction(Month, Day),
                                       color = factor(Month))) +
  geom_line() +
  labs(x = "Time of Day (Hour)", y = "Solar Zenith Angle (degrees)",
       title = "Solar Zenith Angle by Time of Day (June and December Only)") +
  scale_color_discrete(name = "Month", labels = c("June", "December")) +
```

```
  theme_minimal() +
  theme(legend.position = "top", axis.text.x = element_text(angle = 45, hjust = 1))


#plotting how solar irradiance is affected by time of month
city_data[[i]]$Month <- factor(city_data[[i]]$Month, levels = 1:12,
                               labels = c("Jan", "Feb", "Mar", "Apr", "May",
                                          "Jun", "Jul", "Aug", "Sep", "Oct",
                                          "Nov", "Dec"))
agg_data <- city_data[[i]] %>%
  group_by(Month, Hour) %>%
  summarize(Avg_GHI = mean(GHI))
p4 <- ggplot(data = city_data[[i]], aes(x = Hour, y = GHI,
                                        group = interaction(Month, Hour),
                                        color = Month)) +
  geom_line(size = 1.5, alpha = 0.7) +  # Set the size to 1.5 (adjust as needed)
  labs(x = "Hour of the Day", y = "Average Global Horizontal Irradiance (W/m²)",
       title = "Average Solar Irradiance by Hour for Each Month (Spaghetti Plot)") +
  scale_x_continuous(breaks = seq(0, 23, by = 1)) +
  theme_minimal() +
  theme(legend.position = "top", axis.text.x = element_text(angle = 45, hjust = 1))


city_data[[i]] <- city_data[[i]] %>%
  rename(Cloud_Type = `Cloud Type`)
city_data[[i]]$Cloud_Type <- as.factor(city_data[[i]]$Cloud_Type)

# Effect of clouds on GHI
p5 <- ggplot(data = city_data[[i]], aes(x = Cloud_Type, y = GHI)) +
  stat_summary(data = subset(city_data[[i]], `Solar Zenith Angle` <= 80),
               fun = "mean", geom = "bar", fill = "skyblue", color = "black") +
  labs(x = "Cloud_Type", y = "Mean Global Horizontal Irradiance (W/m²)",
       title = "Effect of Cloud Type on Irradiance (Solar Zenith: 0-80 degrees)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

names(city_data[[i]])

# temperature vs irradiance boxplot
p6 <- ggplot(data = subset(city_data[[i]], `Solar Zenith Angle` <= 80),
       aes(x = cut(Temperature, breaks = seq(min(Temperature),
                                             max(Temperature) + 2, by = 2)), y = GHI)) +
  geom_boxplot() +
  labs(x = "Temperature (°C)", y = "Global Horizontal Irradiance (W/m²)",
       title = "Global Horizontal Irradiance vs. Temperature
                (Solar Zenith: 0-80 degrees)") +
  theme_minimal()


#Surface Albedo vs Irradiance for sunlight hours
filtered_data <- city_data[[i]] %>%
  filter(`Solar Zenith Angle` >= 0 & `Solar Zenith Angle` <= 80,
         `Surface Albedo` >= 0.10 & `Surface Albedo` <= 0.25)
```

```
  filtered_data <- filtered_data %>%
    mutate(Albedo_Bin = cut(`Surface Albedo`, breaks = seq(0.10, 0.25, by = 0.005)),)
  p7 <- ggplot(data = filtered_data, aes(x = Albedo_Bin, y = GHI)) +
    geom_boxplot(outlier.color = "gray", size = 0.5) +
    labs(x = "Surface Albedo", y = "Global Horizontal Irradiance (W/m²)",
         title = "Distribution of GHI for Different
         Surface Albedo (Grouped by bins of 0.005)\n(Zenith: 0-80 degrees)") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  #Relative humidity vs Solar Irradiance for zentih 0-80
  filtered_data <- filtered_data %>%
    mutate(RelHumidity_Bin = cut(`Relative Humidity`, breaks = seq(0, 100, by = 2.5)))
  p8 <- ggplot(data = filtered_data, aes(x = RelHumidity_Bin, y = GHI)) +
    geom_boxplot(fill = "skyblue", color = "black") +
    labs(x = "Relative Humidity (%)", y = "Global Horizontal Irradiance (W/m²)",
         title = "GHI vs. Relative Humidity (Zenith: 0-80 degrees, Bin Size: 2.5%)") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  #Precipitable Water vs GHI (for hours during sunlight)
  filtered_data <- filtered_data %>%
    mutate(PrecipitableWater_Bin = cut(`Precipitable Water`,
                                       breaks = seq(min(`Precipitable Water`),
                                                    max(`Precipitable Water`),
                                                    by = 0.15)))
  p9 <- ggplot(data = filtered_data, aes(x = PrecipitableWater_Bin, y = GHI)) +
    geom_boxplot(fill = "skyblue", color = "black") +
    labs(x = "Precipitable Water (Bin Size: 0.05)",
         y = "Global Horizontal Irradiance (W/m²)",
         title = "GHI vs. Precipitable Water (Zenith: 0-80 degrees)") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  p <- list(temp, dew, humidity, albedo, wind, pressure, windD, precip, zenith,
            ghi, cloud, vsTemp, vsDew, vsHumid, vsAlbedo, vsWindS, vsPressure,
            vsWindD, vsPrecip, vsZenith, vsCloud, p1, p2, p3, p4, p5, p6, p7, p8, p9)
  allPlots[[i]] <- p
}
```

```
# 1 is LV, 2 is NY, 3 is Chicago
allPlots[[1]]
allPlots[[2]]
allPlots[[3]]
```

## Preprocessing

```
# read in the full dataset
solarData <- fread("data/SolarFull.csv")
solarData$`Cloud Type` <- as.factor(solarData$`Cloud Type`)
```

```r
# remove zenith angle above 90
solarData <- (solarData[which(solarData$`Solar Zenith Angle` < 90)])

# strip out location data as it is useless
data.processed <- solarData %>% select(-Location)

# one hot encoding on the cloud type using caret
dummy <- dummyVars(" ~.", data=data.processed)
data.processed <- data.frame(predict(dummy, data.processed))

# cyclical encoding on wind direction, month, day, hour, minute
data.processed <- data.processed %>%
  mutate(sinWD= sin(X.Wind.Direction. * pi / 180),
         cosWD = cos(X.Wind.Direction. * pi / 180)) %>% select(-X.Wind.Direction.)
data.processed <- data.processed %>% mutate(sin.month = sin(2 * pi * Month / 12),
                                            cos.month = cos(2 * pi * Month / 12),
                                            sin.day = sin(2 * pi * Day / 31),
                                            cos.day = cos(2 * pi * Day / 31),
                                            sin.hour = sin(2 * pi * Hour / 24),
                                            cos.hour = cos(2 * pi * Hour / 24),
                                            sin.minute = sin(2 * pi * Minute / 60),
                                            cos.minute = cos(2 * pi * Minute / 60))
data.processed <- data.processed %>% select(-c(Month, Day, Hour, Minute))

# min max standardization on X variables (do not change y variable)
ghi_col <- data.processed$GHI
data.processed <- data.processed %>% select(-GHI)
processed <- preProcess(data.processed, method=c("range"))
data.processed <- predict(processed, data.processed)
data.processed$GHI <- ghi_col
```

### Splitting Data

```r
N <- length(data.processed$GHI)
# n1 <- floor(.8*N)
n1 <- 300000
n2 <- 100000
set.seed(10)
idx_train <- sample(N, n1)
idx_no_train <- (which(! seq(1:N) %in% idx_train))
idx_test <- sample(idx_no_train, n2)
idx_val <- (which(! idx_no_train %in% idx_test))
data.train <- data.processed[idx_train,]
data.test <- data.processed[idx_test,]
data.val <- data.processed[idx_val,]
```

### Train and Run Models

```r
# create x and y matrices for training, testing, validation
train_x <- data.matrix(data.train[,-ncol(data.train)])
train_y <- data.matrix(data.train[,ncol(data.train)])
test_x <- data.matrix(data.test[,-ncol(data.train)])
test_y <- data.matrix(data.test[,ncol(data.train)])
val_x <- data.matrix(data.val[,-ncol(data.train)])
val_y <- data.matrix(data.val[,ncol(data.train)])
```

```r
# SVR
fitSVR <- svm(GHI ~ ., data.train)

# Random Forest
fitRF <- randomForest(GHI ~ ., data=data.train, ntree=200)

# LM + Backwards Elimination
fitLM <- lm(GHI ~ ., data.train)
fitLM <- update(fitLM, .~. - cos.minute)
fitLM <- update(fitLM, .~. - X.Cloud.Type.10)
fitLM <- update(fitLM, .~. - X.Cloud.Type.4)
fitLM <- update(fitLM, .~. - sin.day)
fitLM <- update(fitLM, .~. - cos.day)
fitLM <- update(fitLM, .~. - X.Wind.Speed.)
fitLM <- update(fitLM, .~. - sin.minute)

# XGB training
xgb_train <- xgb.DMatrix(data = train_x, label = train_y)
xgb_test <- xgb.DMatrix(data = test_x, label = test_y)
xgb_val <- xgb.DMatrix(data=val_x, label=val_y)
watchlist <- list(train=xgb_train, test=xgb_test)
fitXGB <- xgb.train(data = xgb_train, max.depth = 11, nrounds = 361, watchlist=watchlist)

## simple NN train
set.seed(10)
p <- dim(train_x)[2] # number of input variables
fitNN <- keras_model_sequential() %>%
  layer_dense(units = 64, activation = "relu", input_shape = c(p)) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 1)

fitNN %>% compile(
  optimizer = "Adam",
  loss = "MeanSquaredError",
)

history <- fitNN %>% fit(
  train_x,
  train_y,
  epochs = 15,
  batch_size = 512,
  validation_split = .15
)
```

## Results

```
# Create predictions and output both r^2 and RMSE
predictions <- predict(fitSVR, data.test)
RMSE(test_y, predictions)
R2(test_y, predictions)

predictions <- predict(fitRF, data.test)
RMSE(test_y, predictions)
R2(test_y, predictions)

predictions <- predict(fitLM, data.test)
RMSE(test_y, predictions)
R2(test_y, predictions)

predictions <- predict(fitXGB, xgb_test)
RMSE(test_y, predictions)
R2(test_y, predictions)

predictions <- predict(fitNN, test_x)
RMSE(test_y, predictions)
R2(test_y, predictions)
```

## Visualizations

```
## This outputs the epoch training graph and the importance table for XGB
fitXGB$evaluation_log[10:nrow(fitXGB$evaluation_log),] %>% ggplot(aes(x=iter)) +
  geom_point(aes(y=test_rmse), color="red") +
  geom_point(aes(y=train_rmse), color="blue") + labs(y="RMSE")

importance <- xgb.importance(colnames(xgb_train), model=fitXGB)
xgb.ggplot.importance(importance)

tree1 <- xgb.plot.tree(model=fitXGB, trees=1, plot_width = 1000, plot_height=1000)
tree1 %>% export_svg %>% charToRaw %>% rsvg_pdf("graph.pdf")
xgb.ggplot.shap.summary(data=test_x, model=fitXGB)
```

## Map

```
# Read in the Raw Data (same cleaning procedure as above)
files <- list.files(path = "data/MapData/raw",
                    pattern = "\\.csv$", full.names = TRUE)
files_no_path <- list.files(path = "data/MapData/raw",
                            pattern = "\\.csv$", full.names = FALSE)
read <- lapply(files, readLines)
first_two <- lapply(read, get_first_two)
read <- lapply(read, remove_first_two)

for (i in 1:length(files)) {
```

```
    fwrite(read[i],
           file = (paste0("data/MapData/cleaned/", "data",
                          files_no_path[i])), quote = FALSE)
    fwrite(first_two[i],
           file = (paste0("data/MapData/cleaned/", "info",
                          files_no_path[i])), quote = FALSE)
}


files_cleaned <- list.files(path="data/MapData/cleaned",
                            pattern="data.*\\.csv$", full.names=TRUE)
f_cleaned <- lapply(files_cleaned, fread)

# we keep the info csv instead of just removing it and add in the extra data
# which includes location ID as well as long and lat
# every point has a unique location ID
info_cleaned <- list.files(path="data/MapData/cleaned",
                           pattern="info.*\\.csv$", full.names=TRUE)
i_cleaned <- lapply(info_cleaned, fread)
for(i in 1:length(f_cleaned)) {
  f_cleaned[[i]]$ID <- i_cleaned[[i]]$`Location ID`
  f_cleaned[[i]]$Long <- i_cleaned[[i]]$Longitude
  f_cleaned[[i]]$Lat <- i_cleaned[[i]]$Latitude
}

# join together all datasets and remove everything above 90 as usual
full <- f_cleaned %>% reduce(full_join)
full <- full[which(full$`Solar Zenith Angle` < 90)]

# do the same preprocessing in order to input into the model
data.processed <- full
data.processed$`Cloud Type` <- as.factor(data.processed$`Cloud Type`)
# one hot encoding on the cloud type using caret
dummy <- dummyVars(" ~.", data=data.processed)
data.processed <- data.frame(predict(dummy, data.processed))

# cyclical encoding on wind direction, month, day, hour, minute
data.processed <- data.processed %>%
  mutate(sinWD= sin(X.Wind.Direction. * pi / 180),
         cosWD = cos(X.Wind.Direction. * pi / 180)) %>% select(-X.Wind.Direction.)
data.processed <- data.processed %>%
  mutate(sin.month = sin(2 * pi * Month / 12), cos.month = cos(2 * pi * Month / 12),
         sin.day = sin(2 * pi * Day / 31), cos.day = cos(2 * pi * Day / 31),
         sin.hour = sin(2 * pi * Hour / 24), cos.hour = cos(2 * pi * Hour / 24),
         sin.minute = sin(2 * pi * Minute / 60),
         cos.minute = cos(2 * pi * Minute / 60)) %>%
  select(-c(Month, Day, Hour, Minute))

# min max standardization on X variables (do not change y variable)
ghi_col <- data.processed$GHI
data.processed <- data.processed %>% select(-c(GHI, Year, ID, Long, Lat))
processed <- preProcess(data.processed, method=c("range"))
data.processed <- predict(processed, data.processed)
```

```
data.processed$GHI <- ghi_col
data.processed$X.Cloud.Type.2 <- 0
data.processed$X.Cloud.Type.10 <- 0
data.processed$Year <- (21/20)

# reorder the columns to be the same as the original set
data.processed <- data.processed[, c(32, 1, 15, 18, 13, 2, 14, 17, 16, 3, 4, 30,
                                     5, 6, 8, 9, 10, 11, 31, 19, 20, 21, 22, 23,
                                     24, 25, 26, 27, 28, 29)]

data_x <- data.matrix(data.processed[,-ncol(data.processed)])
data_y <- data.matrix(data.processed[,ncol(data.processed)])

# predict with our model
predictions <- predict(fitXGB, data_x)
full$PredGHI <- predictions

# find the "center"
avg_long = mean(full$Long)
avg_lat = mean(full$Lat)

# sum up the GHI and make it only the location number of points
sums <- full %>% group_by(ID) %>% mutate(sumGHI = sum(PredGHI)) %>%
  select(ID, Long, Lat, sumGHI) %>% unique()

# create SF points for the long lat with the 4326 projection
stFull <- st_as_sf(sums, coords=c("Long", "Lat"), crs=4326)

# make the center an SF object as well
avg.df = data.frame(Long=numeric(), Lat=numeric())
avg.df[1,] = c(avg_long, avg_lat)
center <- st_as_sf(avg.df, coords=c("Long", "Lat"), crs=4326)

# calculate the distances in meter
stFull <- stFull %>% mutate(dist = as.numeric(st_distance(center, geometry)[1]))

# output the final heat map
stFull <- stFull %>% mutate(Power = sumGHI * (0.9999)^(dist / 1000))
heatMap <- stFull %>%
  ggplot() + geom_sf(aes(color=Power, size = Power)) +
  fscale_color_gradient(low="blue", high="red")
ggsave("heatmap_final.png", heatMap)
```

### Tilt Graph

```
solar_df <- fread("data/MapData/cleaned/data1841034_35.70_-106.11_2021.csv")
solar_df$Date <- as.POSIXct(paste(solar_df$Year, solar_df$Month, solar_df$Day,
                                  solar_df$Hour, solar_df$Minute, sep = "-"),
                            format = "%Y-%m-%d-%H-%M")
filtered_solar_df <- solar_df %>%
  filter(Solar.Zenith.Angle <= 90)
```

```r
lat <- 35.6870

solar_df <- solar_df %>%
  mutate(Day_Index = as.numeric(format(Date, "%j")))

hourly_solar_df_1 <- solar_df %>%
  group_by(Day_Index, Hour) %>%
  summarise(Hg_hourly = sum(Clearsky.GHI))

hourly_solar_df_2 <- solar_df %>%
  group_by(Day_Index, Hour) %>%
  summarise(Hd_hourly = sum(Clearsky.DHI))

daily_solar_df_1 <- hourly_solar_df_1 %>%
  group_by(Day_Index) %>%
  summarise(Hg = sum(Hg_hourly))

daily_solar_df_2 <- hourly_solar_df_2 %>%
  group_by(Day_Index) %>%
  summarise(Hd = sum(Hd_hourly))

daily_albedo_df <- solar_df %>%
  group_by(Day_Index) %>%
  summarise(Albedo = mean(Surface.Albedo))

decl_angle_form <- data.frame(Day = numeric(),
                      DeclinationAngle = numeric())
for (n in 1:365) {
  declination_angle_degr <- (23.45*sin(2*pi*(284+n)/365))
  decl_angle_form <- decl_angle_form %>%
    add_row(Day = n, DeclinationAngle = declination_angle_degr)
}
decl_angle_form$lat <- lat
decl_angle_form$lat_rad <- decl_angle_form$lat * pi / 180
decl_angle_form$delta_rad <- decl_angle_form$DeclinationAngle * pi / 180
decl_angle_form$hour_angle_sunset <- acos(-tan(decl_angle_form$lat_rad) * tan(decl_angle_form$delta_rad
decl_angle_form$hour_angle_sunset <- acos(-tan(decl_angle_form$lat_rad) * tan(decl_angle_form$delta_rad
decl_angle_form <- decl_angle_form %>%
  left_join(daily_solar_df_1, by = c("Day" = "Day_Index"))
decl_angle_form <- decl_angle_form %>%
  left_join(daily_solar_df_2, by = c("Day" = "Day_Index"))
decl_angle_form <- decl_angle_form %>%
  left_join(daily_albedo_df, by = c("Day" = "Day_Index"))




calculate_Ht <- function(Hg, Hd, Albedo, lat_rad, DeclinationAngle, hour_angle_sunset, beta_rad) {
  Rb <- (cos(lat_rad - beta_rad) * cos(DeclinationAngle) * sin(hour_angle_sunset) +
           hour_angle_sunset * sin(lat_rad - beta_rad) * sin(DeclinationAngle)) /
         (cos(lat_rad) * cos(DeclinationAngle) * sin(hour_angle_sunset) +
           hour_angle_sunset * sin(lat_rad) * sin(DeclinationAngle))
```

```r
  Ht <- (Hg - Hd) * Rb + Hg * Albedo * (1 - cos(beta_rad)) / 2 + Hd * (1 + cos(beta_rad)) / 2
  return(Ht)
}

optimal_betas <- list()

# Loop through each row of decl_angle_form
for (i in 1:nrow(decl_angle_form)) {
  row <- decl_angle_form[i, ]

  # Get the necessary variables for the Ht function
  Hg <- row$Hg
  Hd <- row$Hd
  Albedo <- row$Albedo
  lat_rad <- row$lat_rad
  DeclinationAngle <- row$delta_rad
  hour_angle_sunset <- row$hour_angle_sunset

  # Define the range of beta values from 0 to 2*pi by 0.01
  beta_range <- seq(0, pi, by = 0.01)

  # Calculate Ht for each beta value and find the maximum
  max_Ht <- -Inf
  optimal_beta <- NULL

  for (beta in beta_range) {
    Ht <- calculate_Ht(Hg, Hd, Albedo, lat_rad, DeclinationAngle, hour_angle_sunset, beta)

    if (Ht > max_Ht) {
      max_Ht <- Ht
      optimal_beta <- beta
    }
  }

  # Store the results in the list
  optimal_betas[[i]] <- list(Day = row$Day, Optimal_Beta = optimal_beta, Max_Ht = max_Ht)
}

optimal_beta_df <- lapply(optimal_betas, function(x) {
  data.frame(Day = x$Day, Optimal_Beta_Deg = x$Optimal_Beta * 180 / pi)
}) %>%
  bind_rows()

# Plot the optimal beta values against the day index
ggplot(optimal_beta_df, aes(x = Day, y = Optimal_Beta_Deg)) +
  geom_line() +
  labs(title = "Optimal Beta Angle vs. Day Index",
       x = "Day Index",
       y = "Optimal Beta Angle (Degrees)") +
  theme_minimal()

yearly_beta_opt <- decl_angle_form %>%
  summarise(yearly_Hg = mean(Hg),
```

```
            yearly_Hd = mean(Hd),
            yearly_Albedo = mean(Albedo),
            yearly_delta_rad = weighted.mean(delta_rad, Hg),
            yearly_wss = weighted.mean(hour_angle_sunset, Hg))

beta_range_for_avr <- seq(0, pi/2, by = 0.01)

Ht_results_df <- data.frame(Beta_Deg = numeric(),
                            Ht = numeric())

for (beta in beta_range_for_avr) {
  # Calculate Ht for the given beta using the yearly_beta_opt values
  Ht <- calculate_Ht(yearly_beta_opt$yearly_Hg, yearly_beta_opt$yearly_Hd,
                     yearly_beta_opt$yearly_Albedo, lat_rad, yearly_beta_opt$yearly_delta_rad,
                     yearly_beta_opt$yearly_wss, beta)

  # Save the results in the Ht_results list
  Ht_results_df <- bind_rows(Ht_results_df, data.frame(Beta_Deg = beta * 180 / pi, Ht = Ht))

}

# universal prediction
max_Ht_index <- which.max(Ht_results_df$Ht)
optimal_beta <- Ht_results_df$Beta_Deg[max_Ht_index]
optimal_Ht <- Ht_results_df$Ht[max_Ht_index]

ggplot(Ht_results_df, aes(x = Beta_Deg, y = Ht)) +
  geom_line() +
  geom_point(x = optimal_beta, y = optimal_Ht, color = "blue", size = 3) +
  labs(title = "Ht vs. Beta Angle",
       x = "Beta Angle (Degrees)",
       y = "Ht") +
  theme_minimal()
# for (result in optimal_betas) {
#   cat("Day:", result$Day, "\n")
#   cat("Optimal Beta (in radians):", result$Optimal_Beta, "\n")
#   cat("Max Ht:", result$Max_Ht, "\n\n")
# }

decl_angle <- data.frame(Day = numeric(),
                         DeclinationAngle = numeric())
for (n in 1:365) {
  declination_angle_degr <- (23.45*sin(2*pi*(284+n)/365))
  decl_angle <- decl_angle %>%
    add_row(Day = n, DeclinationAngle = declination_angle_degr)
}
calculate_optimal_tilt <- function(delta) {
  beta_opt <- 35.15 - 137 * delta/360 - 0.007 * delta^2/(360^2)
  return(beta_opt)
}

decl_angle$OptimalTiltAngle <- calculate_optimal_tilt(decl_angle$DeclinationAngle)
```

```r
ggplot(data = decl_angle, aes(x = Day, y = OptimalTiltAngle)) +
  geom_line(color = "blue") +
  labs(title = "Optimal Tilt Angle for Solar Panels",
       x = "Day of Year",
       y = "Optimal Tilt Angle (degrees)")
```